OVERVIEW AND SOFTWARE ARCHITECTURE OF THE COPERNICUS TRAJECTORY DESIGN AND OPTIMIZATION SYSTEM

Jacob Williams[1], Juan S. Senent[2], Cesar Ocampo[3], Ravi Mathur[3], and Elizabeth C. Davis[4]

[1]*ERC, Inc. Engineering and Science Contract Group, Houston, Texas*
[2]*Odyssey Space Research, Houston, Texas*
[3]*Department of Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin, Austin, Texas*
[4]*Jacobs Technology, Engineering and Science Contract Group, Houston, Texas*

ABSTRACT

The Copernicus Trajectory Design and Optimization System represents an innovative and comprehensive approach to on-orbit mission design, trajectory analysis and optimization. Copernicus integrates state of the art algorithms in optimization, interactive visualization, spacecraft state propagation, and data input-output interfaces, allowing the analyst to design spacecraft missions to all possible Solar System destinations. All of these features are incorporated within a single architecture that can be used interactively via a comprehensive GUI interface, or passively via external interfaces that execute batch processes. This paper describes the Copernicus software architecture together with the challenges associated with its implementation. Additionally, future development and planned new capabilities are discussed.

Key words: Copernicus, Spacecraft Trajectory Optimization Software.

1. INTRODUCTION

Copernicus [11, 12, 13] is a generalized spacecraft trajectory design and optimization system. It is capable of solving a wide range of trajectory design and optimization problems such as planet or moon centered trajectories, libration point trajectories, planet-moon transfers and tours, and all types of interplanetary and asteroid/comet missions. Impulsive and finite burn (low to high thrust) propulsion systems based on chemical, solar electric, or nuclear powered engines can be modeled. A unified architecture (shown schematically in Figure 1) allows the formulation and solution to many classes of problems of practical interest. Unique aspects of the Copernicus architecture include:

- Continuous feedback during the trajectory design and optimization process.
- Generalized trajectory segment building blocks, with which the user can construct a wide range of problems (e.g., simulation, targeting, and optimization), and which can be combined and reused as needed to solve new problems in the future.
- A modular design which allows for the inclusion of new algorithms, models, and techniques as they become available.
- The ability to be scaled from a single desktop computer using the Graphical User Interface (GUI), to computer clusters where no user interaction or graphical feedback is required.
- A batch processing library that can be incorporated into user-created programs or invoked by other tools.
- Platform independence. Currently, Copernicus works on both Linux and Windows platforms.

The Copernicus Project started at the University of Texas at Austin in August 2001. In June 2002, a grant from the NASA Johnson Space Center (JSC) was used to develop the first prototype which was completed in August 2004. In the interim, support was also received from NASA's In Space Propulsion Program [15, 8] and from the Flight Dynamics Vehicle Branch of Goddard Spaceflight Center. The first operational version was completed in March 2006 (v1.0). The initial development team consisted of Dr. Cesar Ocampo and graduate students at the University of Texas at Austin Department of Aerospace Engineering and Engineering Mechanics. Since March 2007, primary development of Copernicus has been at the Flight Mechanics and Trajectory Design Branch of JSC. The latest version (v2.2.1) was released in November 2009.

2. TRAJECTORY BUILDING BLOCKS

The *basic segment* concept introduced in [12] is the fundamental building block of the trajectory optimization problem in Copernicus. Any number of segments can be defined in a mission, and can represent multiple spacecraft, multiple stages of a single spacecraft, or can simply be computational segments used to obtain information about selected states in the mission (for example to compute orbital elements or altitude). Segments can be inde-

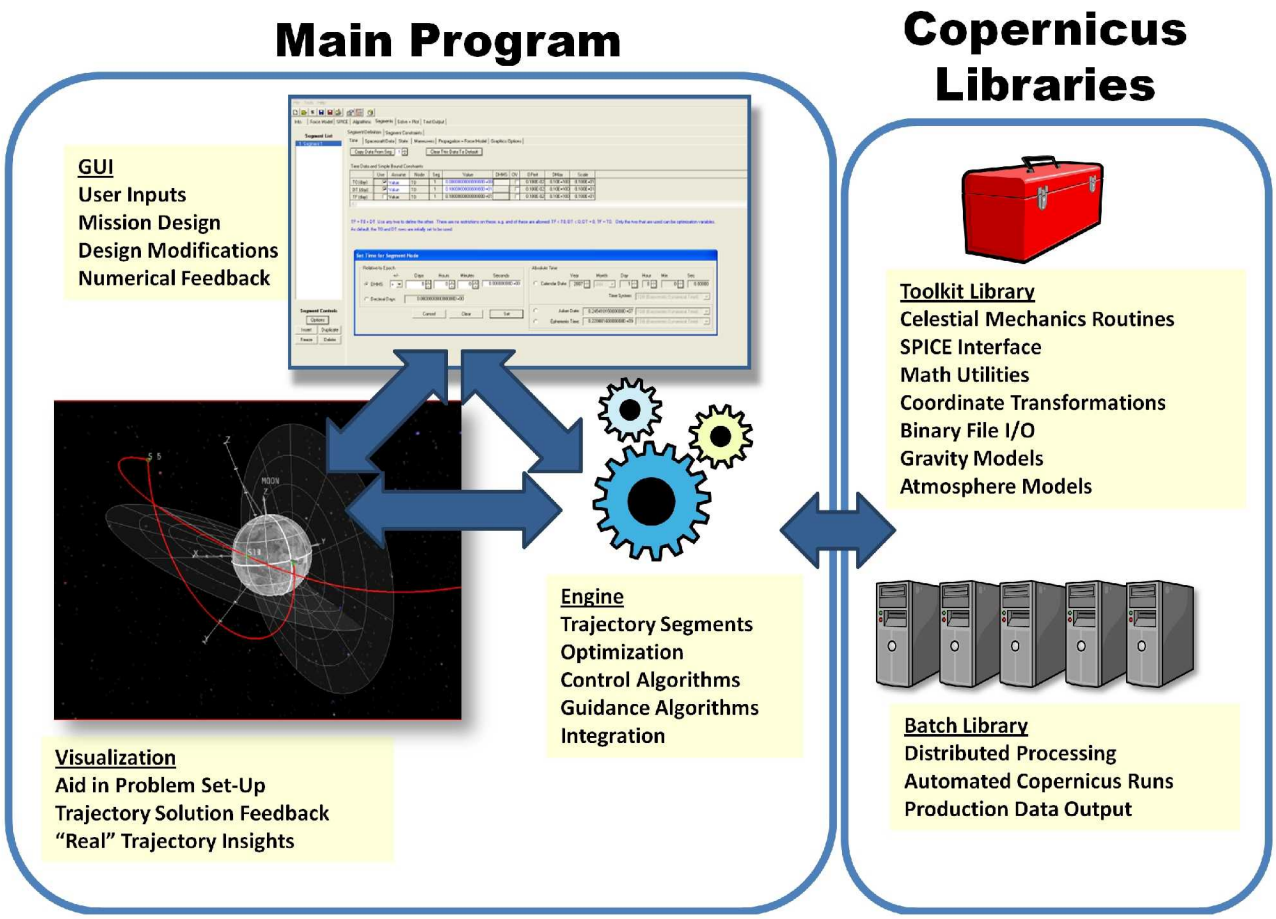

Figure 1. Overview of the Copernicus software architecture. The core of the program consists of the GUI, Visualization, and Engine routines. The Toolkit and Batch libraries also provide services to the program (and can also be used in other programs).

pendent, connected via inheritance in complicated ways, or initially disconnected and constrained to be connected during the optimization process. A segment can contain impulsive maneuvers, finite burn maneuvers, and user-defined force models including: high-order gravity and pointmass gravity fields, atmospheric drag, and solar radiation pressure. Additionally, segment definition parameters such as times, initial mass, mass discontinuities due to staging, initial state, engine parameters, impulsive maneuver and finite burn control law histories [13] can all be optimization variables. Each segment can have a different integration or propagation method (a variety of different fixed and variable step size method are provided) and segments can be propagated either forward or backwards in time.

All of these building blocks can be combined to design very complicated missions. The same problem can be solved using different levels of complexity and fidelity as needed within the same program. The user can start with simplified models and can gradually build a more complex and realistic solution to the problem. Some examples of this include:

- Solving a trajectory problem using simplified force models (e.g., pointmass gravity), and then activating a higher-fidelity force model (e.g., atmospheric drag, high-order gravity, or solar radiation pressure) and re-solving.
- A Δv maneuver can first be estimated using Lambert targeting and then added to the optimization problem.
- Gravity assists can be modeled using a zero sphere of influence patched conic model and then converted to actual planetary flybys.
- A complex trajectory problem can be solved using optimized Δv maneuvers, which are then converted to finite burn maneuvers.

3. USER INTERACTION VIA 3D GRAPHICAL VISUALIZATION

The standard method used by Copernicus for presenting trajectory data to the user is through the 3D Graphical Visualization window. This window presents the user with a graphical representation of the system being simulated, complete with planets, moons, spacecraft, trajectory segments, and other quantities useful to visualizing the com-

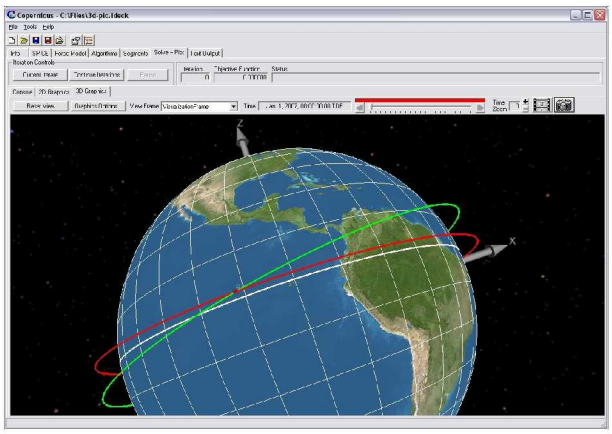

Figure 2. The Copernicus 3D Graphics Tab.

puted solutions. The user is able to interactively view the scene using the mouse or keyboard, and can (in real time) rotate around to view the scene from any angle, zoom in on a particular area of interest, pan around to see a different area of the scene, and even instantly change the simulation time to see where all bodies would be at any given time.

The reference frame in which all visualization is presented is the *visualization frame*, defined using the standard Copernicus frame definition dialogs. This allows the user to visualize the mission in a variety of frames including rotating frames and frames attached to moving celestial bodies. One important aspect of the visualization frame is that it may be completely different from the reference frames used to define the mission segments. Changing the visualization frame allows the user to quickly see the time evolution of bodies and trajectories in the mission with respect to different reference frames. For example, the user can initially specify the simulation to be viewed in an inertial Earth-centered frame. If the user then changes the visualization frame to a rotating Earth-Moon frame, the simulation can be re-run and the newly computed trajectories can be seen instantly.

Copernicus provides the user with several automatically created *views* that can be selected to instantaneously move the user's point of view to a particular location. For example, a spacecraft-centered view can be selected in which the user will automatically follow the spacecraft along a particular segment of the mission. Alternatively, a planet-centered view can be selected in which the user's viewpoint will be constrained to a body-fixed frame regardless of the motion of other planets and spacecraft. These simple views allow the user to see how the simulation evolves from the viewpoint of one of the simulated objects.

The 3D graphics window is handled by three main 3rd party Application Programming Interfaces (APIs). In short, an API is a set of programming subroutines that can be used to perform a defined set of tasks without resorting to low-level operating system calls. All of the drawing for the Copernicus 3D graphics window is done using the OpenGL [19] graphics API. OpenGL is an open source, widely accepted, and well supported API for rendering 3D scenes that is available for almost all mainstream computing platforms and hardware configurations. In addition to OpenGL, the OpenSceneGraph [23] and OpenFrames [10] APIs are used to define the scene to be visualized. This scene consists of all objects to be displayed (e.g. planets, spacecraft, trajectories, etc.) and specifies the positions and orientations of each of the objects. The benefit of using OpenFrames is that it eliminates the need to have any specific computer graphics knowledge when programming a scientific visualization, since such education is generally not within the scope of most engineering programs.

The relationship between these three APIs is as follows. Copernicus interacts directly with OpenFrames, which provides the ability to define which objects should be visualized and how those objects are positioned, and also handles all user interactivity with the scene. OpenFrames then interprets this scene and uses OpenSceneGraph to create a data structure called a "scene graph", which is a tree-like representation of all objects in a scene. OpenSceneGraph then uses this scene graph to draw each element using OpenGL. Although such a level of abstraction may initially seem to hinder application runtime speed, in fact each of the APIs involved are highly optimized so as not to intrude on the available processing resources for the main simulation. In addition, both OpenFrames and OpenSceneGraph are multithreaded to take advantage of multiple-core architectures and minimize the graphics performance overhead for Copernicus.

The OpenGL API comes standard with all major operating systems, and the OpenFrames and OpenSceneGraph APIs are available free of charge (via an OpenSource license) from the internet and are included with Copernicus. The OpenFrames API was developed with scientific simulations in mind, and as such it greatly reduces the effort required to add interactive 3D visualizations to scientific simulations. It is fully integrated into Copernicus, but can be disabled if the program is launched in batch processing mode (i.e., with no graphical user interface).

4. COPERNICUS TOOLKIT

The Copernicus Toolkit was originally designed to provide a unified interface to the coordinate transformation and ephemeris manipulation routines in SPICE [20]. With time, some existing functionalities in Copernicus (such as the Lambert targeter) were integrated into the Copernicus Toolkit so that they could be used in other tools. The Toolkit has been used to develop new trajectory tools independent of Copernicus, e.g.: the Mission Assessment Post Processor (MAPP) [26] and to support various mission analysis tasks, e.g. calculation of Lunar mission abort return trajectories.[21]

Currently, it is composed of a collection of routines that

provides a variety of services:

- A unified software architecture for the definition of frequently used aerospace data. This scheme presents a simple interface to the user for the definition of: vehicle state (parameterizations of the state) and maneuvers, celestial bodies and coordinate frames. It uses most of the capabilities of the SPICE toolkit and adds new functionalities not available through SPICE. It also provides a caching scheme that reduces the number of calls to the SPICE ephemeris routines and an interpolation scheme that can eliminate most of these calls.
- Numerical integration of ODEs with DLSODE [18] (variable step-size, variable order) and rkf7-8 [4] (fixed step-size) and non-gradient based optimization routines [17] and [3] for simple optimization problems.
- Definition of arbitrarily complex gravity fields with any arbitrary number of celestial bodies and any high order terms in the central body [14].
- A library of celestial mechanics-related routines. Including: routines to solve the Kepler and Lambert problems [7], conic (two-body) propagator routines based on Stumpff functions, orbital elements-related routines, lighting models, etc.
- A library with common mathematical routines for matrix and vector calculations in aerospace applications, spline interpolation, etc.
- APIs for the generation of I/O data files in Comma Separated Value (CSV) (text format) and in the Hierarchical Data Format 5 [22] (HDF5 binary format).
- Basic exception handling for common exceptions occurred during run-time. This exception handling mechanism is integrated with the one provided by SPICE.

5. THIRD PARTY PACKAGES

The two main classes of third-party software and libraries that are integrated into Copernicus are: (1) integration and propagation methods, and (2) optimization and nonlinear equation (NLE) solver methods. This software is seamlessly integrated in the GUI, so that the user can select one optimization method, solve a problem, and then switch to another method. Integration and propagation methods are selected from a list, and each segment can use a different method. Internally, wrappers to each routine are written which provide a common interface to Copernicus for each of the third-party methods. The third-party solution method packages incorporated into Copernicus are: NS11AD, VG11AD, VG12AD, and VF13AD from the HSL Archive [1], IPOPT [24], SNOPT [6], SLSQP [9], and HYBRJ (from MINPACK) [16]. In addition to DLSODE, the RKSUITE [2] integration package is also included. Various other open source code is also used in the program. Some of these packages did require modifications in order to be incorporated into the Copernicus architecture. As an example, a modified version of IPOPT was created to allow the stop button in the GUI to terminate the optimization.

6. GRAPHICAL USER INTERFACE

The Copernicus GUI integrates the components of Copernicus, described in previous sections, into a cohesive, easy to use tool. The Copernicus program consists of one executable, which contains the entire GUI, the interactive 3D graphics, capabilities for problem setup and tuning, and control of data output and display. Copernicus was designed with the user as an integral part of the solution process, therefore the GUI and the visualization capability were given high priority during the development.

The flexibility of the Copernicus GUI facilitates, via experimentation, the modeling of many types of trajectory optimization problems and the generation of a methodology required to solve them. Though there are multiple ways in which a trajectory optimization problem can be modeled, there are some that will have better convergence properties than others for a given choice of solution method. Copernicus is not a "black box" to solve problems with little or no input from the user, but is a tool to assist the user in designing and solving the problem. Using Copernicus is a creative process, requiring the user to decide how many segments are needed to model the trajectory, how to parameterize the states and maneuvers, which algorithms to use, etc. For problem setup, the GUI provides fields and menus to enter data, dialogs for each of the solution methods, and graphical tools to provide visual adjustment of trajectory parameters. Through the GUI, the user can define states and maneuvers in any reference frame that is provided by SPICE and the Toolkit. Any planetary body available can be used in any context (e.g. for display in the 3D window, to define a state using orbital elements, to define a two-body rotating reference frame, or to assign a gravity or atmosphere model). Also, SPICE kernels which define the ephemeris models can be loaded and modified from within the GUI.

Once the problem has been defined and the iteration process has commenced, the user can stop it or pause it, modify or tune optimization variables or scales, adjust the orientation of the 3D graphic, and view the text-based output. After the iterations have finished, the user also has the option of reverting to any previous iteration (for example, if the solution has diverged). A text-based grid display of iteration and convergence data facilitates this process, providing syntax highlighting for each iteration to indicate how well the constraints are satisfied. In this way, the GUI allows the user a great deal of control over the optimization process. The design, setup, and solution process using Copernicus is summarized in Figure 3.

7. EXTERNAL INTERFACE

Copernicus can interface with other tools and programs in several ways. It can be called in batch mode from the

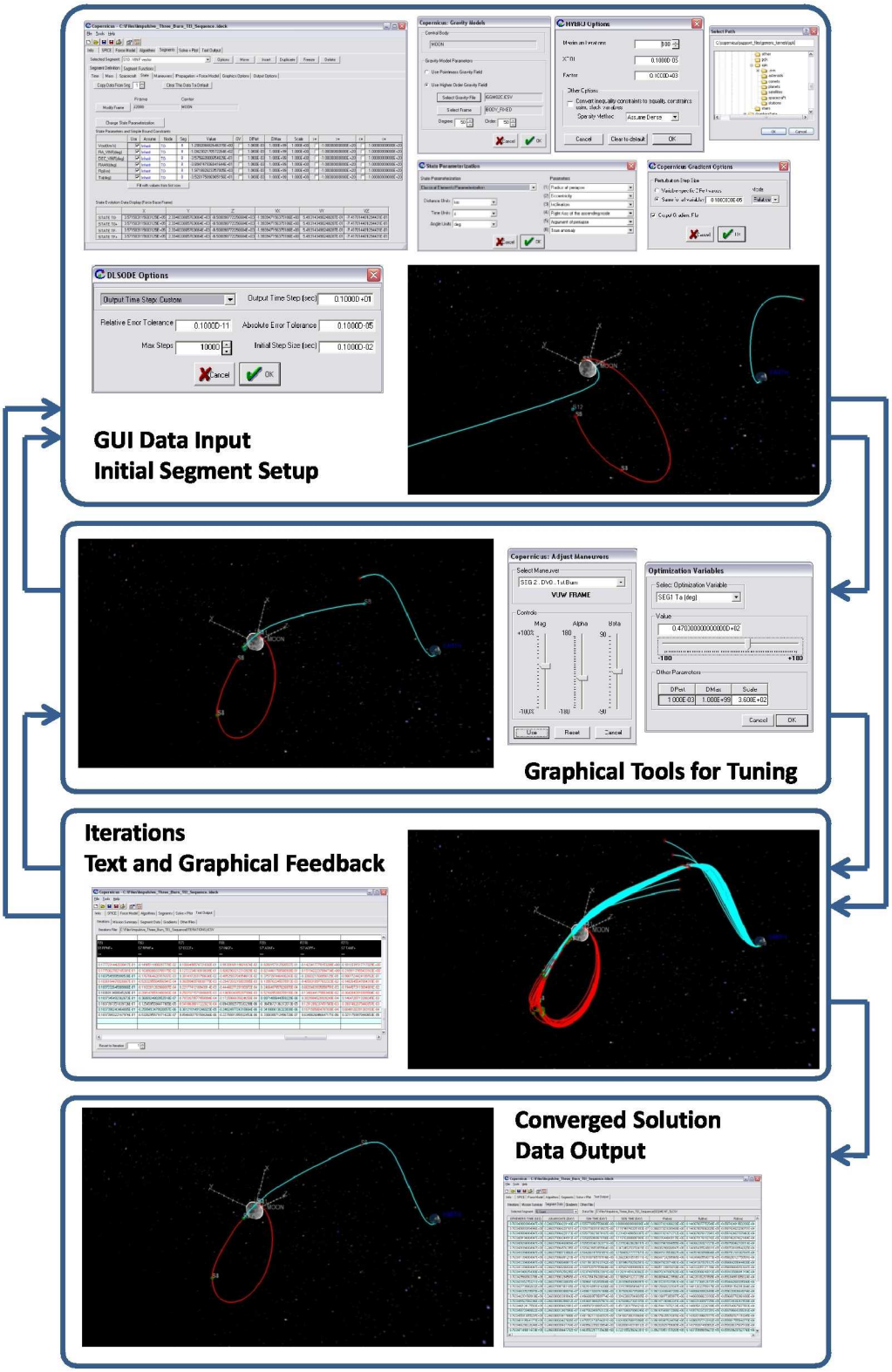

Figure 3. When using Copernicus, the user is an integral part of the solution process. The GUI is used to setup and tune the problem, and provides real-time feedback during the solution process. Graphical tools can be used to adjust the optimization variables and maneuvers and the trajectory display is instantly updated. The 3D graphic display is updated for each iteration as the optimization proceeds. Text-based output is also provided, and uses coloring to indicate convergence of each constraint.

command-line or in-the-loop from within another program to solve or just propagate an input file. An external program can be used to change input variables, then call Copernicus to solve the mission. Such a batch program can also be used to incorporate data from other simulation tools to populate input files which can be used as an initial guess for a Copernicus run. This is a powerful feature which greatly extends the applicability of the tool. As an application of this capability, databases used in the performance evaluation of the Orion vehicle were generated by calling Copernicus in batch mode [25].

A Fortran batch library has been developed which can be used within other programs to read and manipulate Copernicus input files. In addition, a program written in any language could also be used for this purpose (since Copernicus input files are simply text files) and call Copernicus in batch mode. Copernicus also has an extensive library of Matlab routines called the Copernicus Matlab Toolkit (CoMaT). The CoMaT also includes routines to read and plot data from Copernicus output files. This allows the generated data to be quickly post-processed and analyzed in Matlab. CoMaT routines also exist to convert Copernicus generated data to a format readable by other tools such as Satellite Tool Kit (STK). Additionally, there are routines to check and compare data from Copernicus runs, and generate Copernicus readable CSV files. Most of the CoMaT routines are also compatible with GNU Octave.

8. SOFTWARE DEVELOPMENT

Except for some third-party code, Copernicus (including the Toolkit) is programmed in Fortran 95 with the incorporation of some Fortran 2003 features. Although the object-oriented features of Fortran 2003 were not available when the project started, the design of the software architecture followed an object-oriented approach. As compilers implementing the full object-oriented capabilities of Fortran 2003 become available, Copernicus will be modified to take advantage of them. Most of the third-party optimization methods are written in Fortran 77, and the 3D graphics and IPOPT libraries are written in C++. The program is developed using the Intel Fortran compiler and Microsoft Visual Studio. The Copernicus GUI is designed using Winteracter, a modern GUI toolset for the Fortran 90/95 programming language.

9. EXAMPLES

Just a few examples of the kinds of trajectories that have been solved with Copernicus include: design of a 10 year 32-asteroid tour using a low-thrust propulsion system (see Figure 4), free-return flybys of the Moon, trans-lunar trajectories for Project Constellation [5], Orion trans-Earth injection (TEI) return trajectories [25], Mars sample return missions, trajectory design for the Lunar Crater Observation and Sensing Satellite (LCROSS). Copernicus has also been used extensively at numerous NASA centers in support of Project Constellation. Additionally, the system is being used as an educational tool to examine simple problems (such as impulsive transfers between two-body orbits) and complex problems (such as low thrust transfers between circular restricted three body orbits).

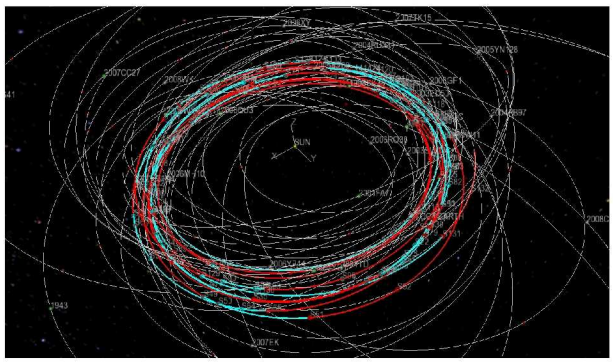

Figure 4. The 32-asteroid tour computed for the GTOC-4 competition. The blue arcs are coast periods, and the red arcs are low-thrust periods.

10. FUTURE DEVELOPMENTS

Copernicus is being actively developed at JSC in support of several NASA projects, and improvements are being made continuously. New engine and maneuver models are being added for low thrust and solar electric missions. New techniques and algorithms are continuously evaluated for possible incorporation into the tool. These include evolutionary algorithms (a differential evolution method has already been included) and other non-gradient based optimizers. Other orbit propagation methods, collocation and pseudospectral methods are also being considered. Eventually, Copernicus will allow for collocation segments to be added to the optimization problem, allowing even greater flexibility to define and solve complicated problems.

11. OBTAINING COPERNICUS

The National Aeronautics and Space Act of 1958 and a series of subsequent legislation recognized transfer of federally owned or originated technology to be a national priority and the mission of each Federal agency. In accordance with NASA's obligations under mandating legislation, NASA makes Copernicus available free of charge to other NASA centers, government contractors, and universities with contractual affiliations with NASA. To obtain a copy of Copernicus, contact the NASA Johnson Space Center Innovative Partnership Program Office.

12. ACKNOWLEDGEMENTS

The authors would like to thank everyone who contributed to the development of Copernicus at the University of Texas at Austin, and the Johnson Space Center, especially Gerald Condon, Fady Morcos, David Lee, Virginia Martin, and Robert Gottlieb.

REFERENCES

[1] HSL (2007). A collection of Fortran codes for large scale scientific computation. http://www.numerical.rl.ac.uk/hsl.

[2] R. W. Brankin and I. Gladwell. Algorithm 771: rksuite_90: Fortran 90 software for ordinary differential equation initial-value problems. *ACM Trans. Math. Softw.*, 23(3):402–415, 1997.

[3] R.P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, 1973. Englewood Cliffs, NJ.

[4] E. Fehlberg. Classical fifth-, sixth-, seventh- and eighth-order runge-kutta formulas with stepsize control. Technical report, George C, Marshall Space Flight Center, NASA, 1968. NASA TR R-287 NASA-Langly 1968 19 M524.

[5] M. Garn, M. Qu, J. Chrone, P. Su, and C. Karlgaard. NASA's Planned Return to the Moon: Global Access and Anytime Return Requirement Implications on the Lunar Orbit Insertion Burns. In *Proceedings of the AIAA Guidance, Navigation and Control conference*, August 2008.

[6] P. E. Gill, W. M., and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. In *Society for Industrial and Applied Mathematics SIAM Review*, volume 47, pages 99–131, 2005.

[7] R.H. Gooding. A Procedure for the Solution of Lambert's Orbital Boundary Value Problem. In *Celestial Mechanics and Dynamical Astronomy*, volume 48, pages 145–165, 1990.

[8] L. D. Kos, T. P. Polsgrove, R. C. Hopkins, D. Thomas, and J. A. Sims. Overview of the development for a suite of low-thrust trajectory analysis tools. In *AIAA/AAS Astrodynamics Specialist Conference*, number AIAA 2006-6743, August 21-24 2006. Keystone, CO.

[9] D. Kraft. A Software Package for Sequential Quadratic Programming. Technical Report DFVLR-FB 88-28, DFVLR, 1988.

[10] R. Mathur and C.A. Ocampo. An Architecture for Incorporating Interactive Visualizations into Scientific Simulations. In *Proceedings of the 57th IAC Conference. paper IAC-06-D1.P.1.6*, August 2006.

[11] C. Ocampo and J. Senent. The Design and Development of Copernicus: A Comprehensive Trajectory Design and Optimization System. In *Proceedings of the International Astronautical Congress*, number IAC-06-C1.4.04, 2006.

[12] C. A. Ocampo. An architecture for a generalized trajectory design and optimization system. In *Proceedings of the International Conference on Libration Points and Missions*, June 2002. Girona, Spain.

[13] C. A. Ocampo. Finite burn maneuver modeling for a generalized spacecraft trajectory design and optimization system. In *Annals of the New York Academy of Science*, volume 1017, pages 210–233, May 2004.

[14] S. Pines. A uniform representation of the gravitational potential and its derivatives for a rotating non-spherical body. Technical report, NASA, 1971. report No. 71-7, Contract No. NAS 1-9100.

[15] T. Polsgrove, L. Kos, R. Hopkins, and T. Crane. Comparison of Performance Predictions for New Low-Thrust Trajectory Tools. In *AIAA/AAS Astrodynamics Specialist Conference*, number AIAA 2006-6742, August 21-24 2006. Keystone, CO.

[16] M. J. D. Powell. A hybrid method for nonlinear equations. In *Numerical Methods for Nonlinear Algebraic Equations*, 1970.

[17] M.J.D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, Department of Applied Mathematics and Theoretical Physics, Centre for Mathematical Sciences, 2009. http://plato.asu.edu/ftp/other_software/bobyqa.zip.

[18] K. Radhakrishnan and A. C. Hindmarsh. Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations. Technical Report UCRL-ID-113855, Lawrence Livermore National Laboratory, 1993.

[19] M. Segal and K. Akeley. The OpenGL Graphics System: A Specification. Technical Report Version 4.0 (Core Profile), The Khronos Group Inc, March 2010.

[20] B. V. Semenov. SPICE Reference Frames. Technical report, Jet Propulsion Laboratory, APRIL 08 2009. http://naif.jpl.nasa.gov/pub/naif/toolkit_docs.

[21] J.S. Senent. Partial-TLI and post-TLI abort options for lunar missions. Technical report, NASA, 2006. TDS-04-013/subtask 1a and 1b.

[22] The HDF Group. HDF5 Tutorial. Technical report, The HDF Group, 2009. http://www.hdfgroup.org/HDF5.

[23] OSG Community. OpenSceneGraph, 2007. http://www.openscenegraph.org/projects/osg.

[24] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[25] J. Williams, E. Davis, D. Lee, G. Condon, T. Dawn, and M. Qu. Global Performance Characterization of the three burn trans-Earth injection maneuver sequence over the lunar nodal cycle. In *Proceedings of the 2009 Astrodynamics Specialist AAS/AIAA Joint Conference*, August 9-13 2009.

[26] J. Williams, S. M. Stewart, G. L. Condon, D. E. Lee, E. C. D., J. S. Senent, and T. F. Dawn. The Mission Assessment Post Processor (MAPP): A New Tool for Performance Evaluation of Human Lunar Missions. In *Proceedings of the 20th AAS/AIAA Space Flight Mechanics Meeting*, number AAS 10-191, February 14-17 2010.