

Space Weather Modeling Framework  
Design Document  
Milestone 11K  
v11.1

**Author:**

**David R. Chesney**

**Contributors:**

**Kenneth Hansen**

**Gabor Toth**

**Kevin Kane**

**Darren DeZeeuw**

**Aaron Ridley**

## Table of Contents

Introduction .....	3
Purpose .....	3
Overview .....	3
Hardware Deployment .....	4
Framework Architecture .....	6
Layered Architecture of Space Weather Modeling Framework .....	6
Generic Structure of a Physics Component .....	7
Brief Descriptions of Individual Physics Components .....	8
Brief Descriptions of Dataflows between Physics Components .....	12
Framework State Behavior .....	19
Overall Execution Model .....	20
Execution Model for Setting Parameters .....	21
Execution Model for Reading Inputs .....	23
Execution Model for Initializing a Session .....	25
Execution Model for Executing a Session .....	27
Physics Component View of Execution Model .....	29
References .....	31

## List of Figures

Figure 1. Hardware deployment diagram for Space Weather Modeling Framework .....	5
Figure 2. Layered model of SWMF. ....	7
Figure 3. Generic structure of a Physics Component .....	7
Figure 4. Dataflow model of SWMF. ....	9
Figure 5. Overall state behavior for SWMF. ....	20
Figure 6. Flowchart for setting parameters. ....	22
Figure 7. Flowchart for reading input. ....	24
Figure 8. Flowchart for initializing a session .....	26
Figure 9. Flowchart for executing a session .....	28
Figure 10. Flowchart diagram of execution from Physics Component viewpoint. ....	30

# Introduction

## **Purpose**

The purpose of this document is to describe the design of the Space Weather Modeling Framework (SWMF). The SWMF provides a flexible Sun-Earth simulation framework serving the Sun-Earth community. In its fully developed form, the SWMF comprises a series of interoperating models of physics domains, ranging from the surface of the Sun to the upper atmosphere of the Earth. The development team provides base versions of these models, but community provided models will also be incorporated as they become available.

The intended audience of this document is the design-approving community, namely the ESS project, CCMC, NOAA SEC, and the ESMF Design Team. Additionally, the audience of this document includes the framework development community, namely the SWMF team.

The organization of this document is first, this introduction that describes the purpose of the SWMF and offers a brief overview. Next, the hardware deployment is described using a UML deployment diagram. Next, the architecture of the SWMF is described from two orthogonal perspectives. First, the final implementation of the SWMF is described from an abstract to an increasingly detailed level. Note that the final implementation is coincident with that described in Milestone 10J of the SWMF Project Proposal [1]. In describing the implemented architecture, the detailed architecture is first described using Dataflow diagrams with symbolic names as placeholders. Then, a more detailed view of the architecture is described using dataflow names and descriptions.

Second, the architecture is described from a state behavior perspective. That is, the sequential behavior of the SWMF is illustrated and described. Flowchart diagrams are used for this description. Finally, references are listed.

## **Overview**

Our intent was to address NASA's need for production-ready high-performance tools that model, analyze, and interpret space science observational data. The main SWMF design goals were defined [5] as:

- Incorporate computational physics modules with only modest modification.
- Achieve good parallel performance in the coupling of the physics components.
- Allow physics components to interact with the SWMF as efficiently as possible.

We reached this goal by engineering existing codes to interoperate in a common framework while simultaneously achieving high performance. The combination of interoperating models and underlying software framework serves the entire Sun-Earth community.

In addition, the SWMF will use NASA satellite data to test and validate the models. Possible examples include:

- data from SOHO to compare with solar coronal runs;
- data from ACE to drive magnetospheric model inputs and validate solar heliosphere runs;
- data from the following satellites to validate magnetospheric runs: Geotail, Interball, Wind, IMP-8, Polar, Image, SAMPEX, and FAST.

Further detail may be obtained from the Project Proposal [1], Software Engineering Plan [2], and Requirements Document [3].

## Hardware Deployment

This section of the Design Document addresses the hardware deployment for the framework. Using the Unified Modeling Language (UML), v 1.3, the hardware deployment is described using Deployment diagrams. For simplicity, Deployment diagrams are referred to as hardware deployment diagrams and Component diagrams are referred to as software deployment diagrams.

The notation used in a UML hardware deployment diagram is intuitive. Stick figures represent users and developers. Workstations may be single or multi-processor, as indicated by iconic multiple processor cards. All execution platforms are illustrated by a large, single icon. Finally, connections between various hardware entities in the hardware deployment diagram are simply shown with a line.

Figure 1 is the hardware deployment diagram for the framework. The development platform for the physics models at UM is Pentium 3+ and/or AMD Athlon multi-processor workstations running Linux OS.

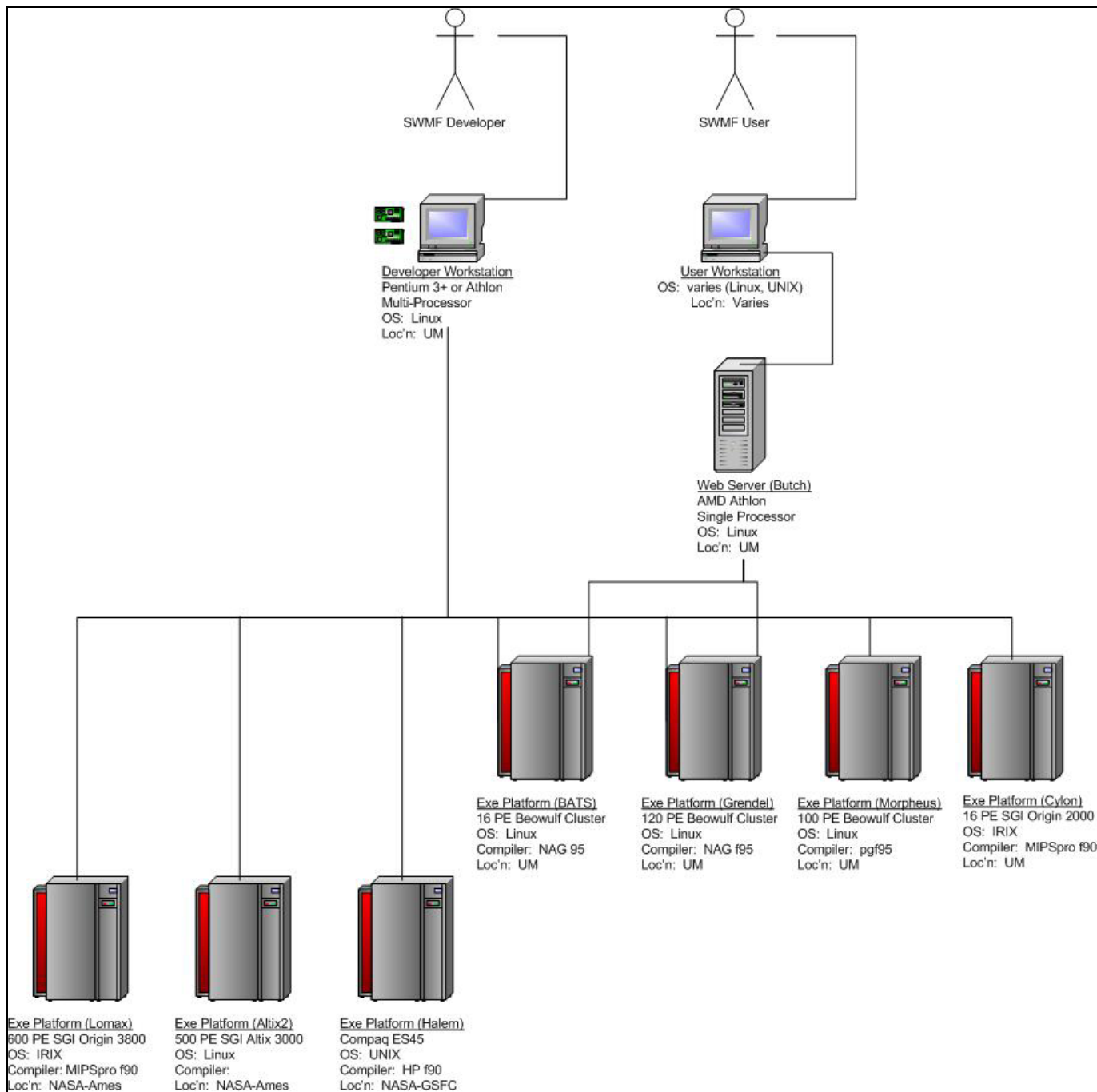
The main execution platforms and Fortran compiler combinations for the SWMF are:

- Beowulf cluster with 120 AMD processors running RedHat Linux OS. The Fortran compiler is NAG f95. The machine is local to UM and is named Grendel. This Beowulf cluster was purchased with funds from this grant. See milestone 2.O1.
- Silicon Graphics Origin 3800 with 600 processors running IRIX 64 OS. The Fortran compiler is MIPSpro f90. The machine is located at NASA Ames and is named Lomax.
- Silicon Graphics Altix 3000 with 500 processors running Linux OS. The machine is located at NASA Ames and is named Altix2.

In addition, the following execution platforms are sometimes used:

- Beowulf cluster with 16 AMD processors running Linux OS. The Fortran compiler is NAG f95. The machine is local to UM and is named BATS.
- Beowulf cluster with 100 AMD processors running Linux OS with a Fortran compiler that is Portland Group Fortran 90 (pgf90). The machine is local to UM and is named Morpheus. Note that this cluster uses a different Fortran compiler than BATS and Grendel, and so is useful for testing purposes.
- Compaq ES45 running UNIX OS (OFS1). The Compaq uses a HP f90 compiler. The machine is located at NASA Goddard Space Flight Center and is named Halem.
- Silicon Graphics Origin 2000 with 16 processors running IRIX OS. The Fortran compiler is MIPSpro f90. The machine is local to UM and is named Cylon.

The web server for the Graphical User Interface (GUI) is an AMD Athlon single-processor workstation that is local to UM. The web server's function is to house the Graphical User Interface (GUI) and control access to the framework. It is design intent that the web server only has access to the cluster-architecture machines located at the University of Michigan (16 PE and 120 PE Beowulf clusters).



**Figure 1. Hardware deployment diagram for Space Weather Modeling Framework.**

## Framework Architecture

The SWMF implementation is based on important component technology and object-oriented programming concepts emulated in Fortran 90 [10]. The SWMF is a layered and structured collection of software building blocks (collections of routines and classes) that can be used to develop Sun-Earth system modeling components, and to assemble them into an application. The SWMF contains the Control Module, which controls initialization and execution of the components. It is responsible for component registration, processor layout for each component and coupling schedules. A component is created from the user-supplied physics code by adding a wrapper, which provides the set of predefined control functions. A component performs the data exchange with its peer component via a standardized pair-wise coupling interface. Both the wrapper and coupling interface are constructed from the building blocks provided by the framework itself.

All parallel communications are based on the MPI standard. The SWMF creates a single executable. Both serial and concurrent execution modes are supported. With concurrent execution the SWMF can run faster than real-time on massively parallel machines.

This section describes the architecture of the SWMF. First, the layered architecture of the SWMF is described. Then, the generic structure of an individual physics component is described. Next, the individual physics components are enumerated and briefly described. Included in this section is a description of the dataflows between each physics component.

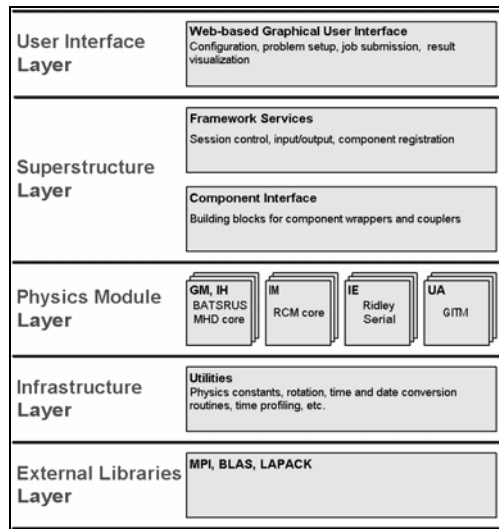
### ***Layered Architecture of Space Weather Modeling Framework***

This section describes a view of the architecture of the SWMF from a layered perspective. The milestone 10J release of the SWMF includes a broad array of physics components and a layered architecture that is similar to previously released versions.

A previously released working prototype of the SWMF (Milestone 7I) included components for five physics domains [9, 10]: Global Magnetosphere (GM), Inner Heliosphere (IH), Ionosphere Electrodynamics (IE), Upper Atmosphere (UA) and Inner Magnetosphere (IM):

- The GM and IH components are based on the University of Michigan's BATSRUS MHD module. The highly parallel BATSRUS code uses a 3D block-adaptive Cartesian grid.
- The IM component is the Rice Convection Model (RCM) developed at Rice University. This serial module uses a 2D non-uniform spherical grid.
- The IE component is a 2D spherical electric potential solver developed at the University of Michigan. It can run on 1 or 2 processors since the northern and southern hemispheres can be solved in parallel.
- The UA component is the Global Ionosphere Thermosphere Model (GITM) recently developed at the University of Michigan as a fully parallel 3D spherical model.

The framework's layered architecture [10] is shown in Figure 2. The Infrastructure consists of utilities, which define physics constants and different coordinate systems, time and data conversion routines, time profiling routines and other lower level routines. The Framework Services consist of software units (classes), which implement component registration, session control, and input/output operations of initial parameters. All of the Framework Services are performed by the Control Module (CON). The Superstructure Layer, Physics Module Layer, and Infrastructure Layer constitute the "sandwich-like" architecture similar to the Earth System Modeling Framework (ESMF) [11]. The SWMF will also contain a web-based Graphical User Interface.

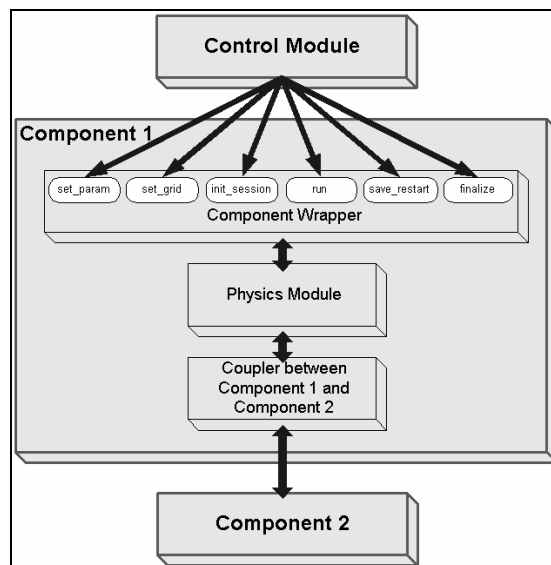


**Figure 2. Layered model of SWMF.**

### **Generic Structure of a Physics Component**

The generic structure of a physics component is shown in Figure 3. As shown, each component is divided into three partitions. From a physicist's viewpoint, the most important partition is the Physics Module. The physics module contains the core solution software for a specific physical domain the Sun-Earth simulation environment.

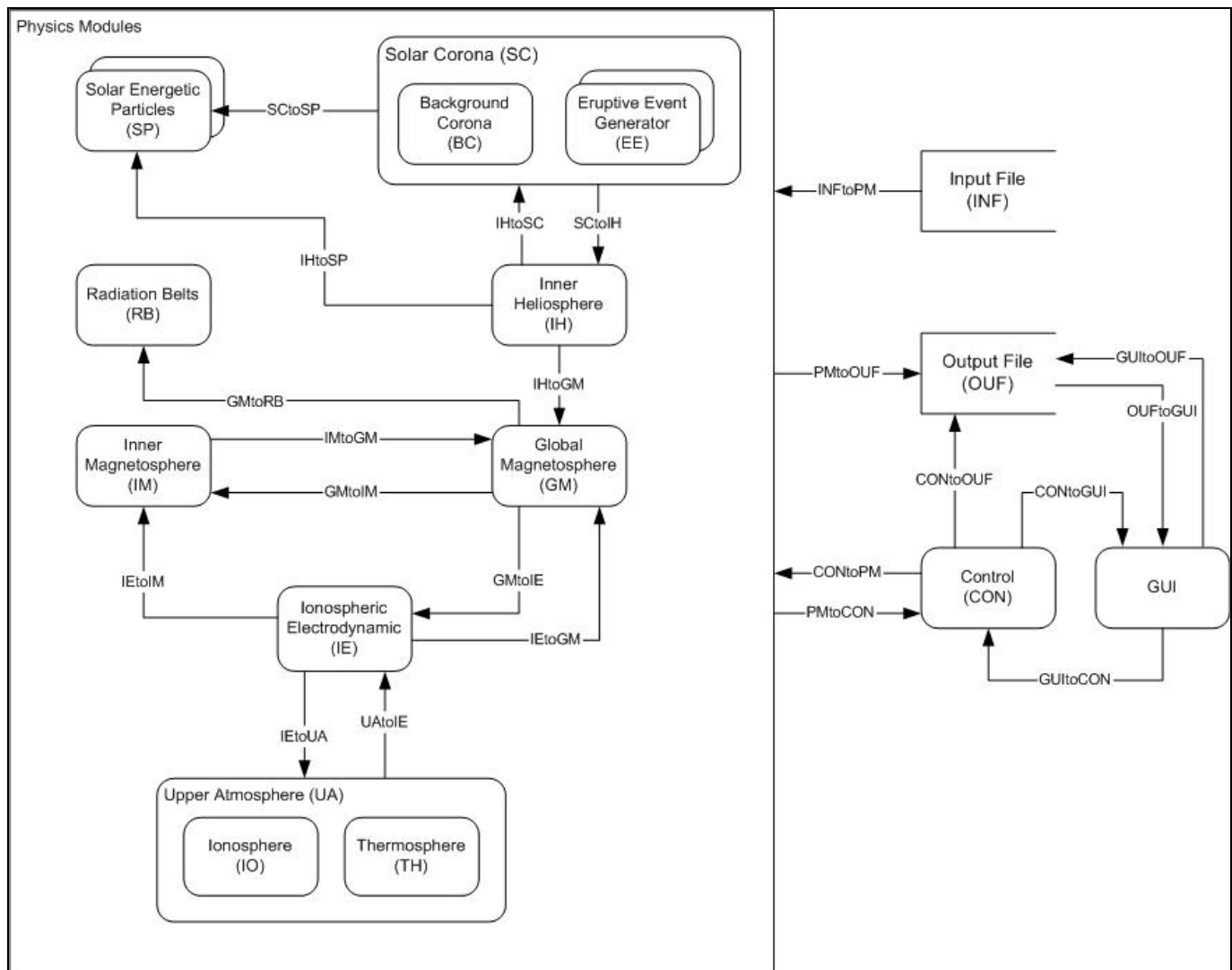
Two additional partitions of a physics component are the Wrapper and the Coupler. The purpose of the wrapper is to provide the standard interface to control the physics module. The purpose of the coupler is to perform the data exchange between peer physics components. From a software component perspective, both the wrapper and coupling interface are Component Interfaces: the wrapper is an interface with the high-level Control Module (CON) of the framework, and the coupler is an interface with another physics component. As shown in Figure 3, wrapper interface methods (subprograms) have standard names and a standard list of formal parameters (dummy arguments), which makes swapping between various versions of a component possible. Both the wrapper and the coupling interface are constructed from the building blocks provided by the framework.



**Figure 3. Generic structure of a Physics Component.**

## Brief Descriptions of Individual Physics Components

The dataflow of the SWMF as implemented for Milestone 10J is shown in **Figure 4**. The physics components (and their respective sources) are enumerated below the diagram:



**Figure 4. Dataflow model of SWMF.**

- Solar Energetic Particles (SP):** The SP component calculates the transport of energetic ions and electronics along magnetic field lines connecting Earth to the Sun. This information is critically important for astronaut safety. The model is based upon an existing numerical model and solves the appropriate Fokker-Planck equation including all important physical phenomena. The model is also able to describe so-called gradual events which are the most powerful and geoeffective SEP events. While particles in most SEP events are primarily accelerated near the Sun, in gradual events the particles are generally accelerated by interplanetary shocks driven by CMEs. J Kota, Arizona State University;
- Combined Solar Coronal and Eruptive Event Generator (CE):** As noted, this component contains both the Solar Coronal model and Eruptive Event Generator. The Solar Coronal model describes the near-solar region from the photosphere to the Solar Coronal model-Inner Heliosphere model interface. A key issue is connecting the small-scale variations on the Sun with large-scale structures in the corona. Parameters such as the source surface radius, cusp surface height, and current magnitudes will be



derived or iterated using physical models. Densities and other parameters are provided by models or assimilated from observations. Eruptive solar events (such as Coronal Mass Ejections (CMEs)) are initiated by the Eruptive Event Generator. These models are very flexible and range from simple parametric modules to MHD models describing overstressing and destabilization of the magnetic field configuration in the lower solar corona. The EEG module will also provide a description of the formation and initial evolution of the CME shock waves which generate geoeffective SEPs.

The CME is initiated within this coronal model by superimposing a 3-D Gibson-Low (GL) magnetic flux rope in the streamer belt in an initial state of force imbalance. The GL solution for this flux rope is derived by applying a mathematical stretching transformation ( $r \rightarrow r - a$ ) to an axisymmetric, spherical ball of twisted magnetic flux in equilibrium with plasma pressure. The transformation, performed in spherical coordinates ( $r, \Theta, \Phi$ ), draws space toward the origin while holding angular coordinates,  $\Theta$  and  $\Phi$  fixed. This mathematical procedure serves two important purposes. First, it generates a geometrically complex solution by distorting the originally spherical, axisymmetric flux rope into a tear-drop shape with full 3-D spatial variation. The second benefit of the stretching transformation is the introduction of Lorentz forces associated with the magnetic field that requires both the pressure and weight of plasma in a  $1/r^2$  gravitational field to be in static equilibrium. The density structure of the model possesses a dense helmet streamer containing a cavity embedded with a prominence-like density enhancement. The equilibrium state of GL requires a significant outward increasing plasma pressure to offset the magnetic pressure in the upper portion of the magnetic flux rope. The background corona is insufficient to provide this pressure, which leaves the flux rope with unbalanced magnetic forces which drive the eruption. The University of Michigan.

- **Radiation Belt Model (RB):** The RB component is based on existing models developed at Rice University. This model will trace bounce-averaged guiding center trajectories of high energy trapped particles and use Liouville's theorem to evolve a corresponding phase-space density taking into account pitch-angle scattering. The RB model will provide fluxes of precipitating relativistic (killer) electrons and ions. Since relativistic electrons are a hazard for spacewalking astronauts this module is very important for NASA. A Chan, D Wolf, S Sazykin, B Yu, Rice University;
- **Inner Heliosphere (IH):** The IH component is a 3D MHD model. None of the specific coronal features are needed for the IH component. The spatial resolution used in the IH component can be relatively low, and relatively large time steps are allowed. The model works at  $5$  to  $30 R_s \leq R \leq 1$  to  $100$  AU ( $\text{AU} \approx 215 R_s$ ). The IH component should be coupled to the SP component (one way coupling through the CE component), the SC component (two-way coupling), and the GM component (one way coupling from IH). The University of Michigan.
- **Global Magnetosphere (GM):** The GM component is an efficient and powerful parallel AMR finite-volume method for solving the semi-relativistic MHD equations to describe the area of the Earth's magnetosphere. A cell-centered upwind finite-volume formulation is adopted to solve the governing equations in divergence form. Three different approximate Riemann solvers are implemented to evaluate the numerical flux function, as well as four different methods to control the  $\nabla \cdot \mathbf{B}$  truncation errors. A block-based AMR technique is used. Computational cells are embedded in regular, structured, self-similar blocks of equal size. University of Michigan;
- **Inner Magnetosphere (IM):** The IM component calculates the dynamic behavior of the particles and the electric fields and currents in the Earth's inner magnetosphere. The physics of the region is complicated, since it contains overlapping particle distributions with a wide range of energies and characteristics. These different coexisting particle populations cannot be treated as a single fluid,

because they all move differently. IM is one of the interoperating codes where we intend to achieve significant numerical accuracy and performance improvements. D Wolf, Rice University;

- Ionospheric Electrodynamics (IE): The ionospheric electrodynamics modules capture the conductivity and electric field in the ionosphere. The conductivity is driven primarily by the sunlight on the dayside and the aurora in the high latitude region. This conductivity is either derived from statistical relationships, or specified by the upper atmosphere module. In the latter case, the UA module needs to have the auroral particle precipitation input. The IE module provides this. The aurora is typically derived from the field-aligned currents, which are passed from the GM module. The ionospheric electric potential is solved for given the field-aligned currents (specified by the GM model, IM module, and UA module) and the conductance. Currently, the “Ridley Ionosphere” is used within the framework. This is an ionospheric potential solver which can run on up to two PEs. Within this code, the auroral particle precipitation can be derived from the field-aligned currents. A Ridley, University of Michigan;
- Upper Atmosphere (UA): The upper atmosphere module is a model of the thermosphere and ionosphere. This model needs to provide height-integrated conductances and the neutral wind driven field-aligned currents to the IE module. The model can be a statistical type of model, such as IRI and MSIS, but the IRI model does not contain the aurora, so it is not as useful as a fully first-principles model of the coupled system. The Global Ionosphere-Thermosphere Model (GITM) is a fully coupled model of the IT system, extending from 95 km to 500+ km. It solves for the neutral and ion densities, temperatures, and velocities, driven by the Sun, the aurora, and the high-latitude potential pattern (the potential and aurora are specified by the IE module). A Ridley, University of Michigan.

### ***Brief Descriptions of Dataflows between Physics Components***

In **Figure 4**, the dataflows are given generic names (e.g., CEtoSP) for simplicity. However, each dataflow is described in greater detail below:

#### **SCtoSP:**

SC provides the coordinates of points along magnetic field lines. The points are advected with the plasma flow by SC. In addition, density, pressure, velocity, and magnetic field are provided by SC at each of the points. The advection is completed by SWMF using the velocities from SC.

- Point coordinates:
  - 3 reals/1D gridpoint/ field line
- Density:
  - 1 real/1D gridpoint/ field line
- Pressure:
  - 1 real/1D gridpoint/ field line
- Velocity vector:
  - 3 reals/1D gridpoint/ field line
- Magnetic field vector:
  - 3 reals/1D gridpoint/ field line

### **SCtoIH:**

The IH needs to be driven by the outward solar wind flow from SC. The IH obtains a time and space varying set of data at its inner boundary, which consists of the solar wind density, pressure, velocity, and magnetic field.

- Density:
  - 1 real/2D gridpoint
- Pressure:
  - 1 real/2D gridpoint
- Velocity vector:
  - 3 reals/2D gridpoint
- Magnetic field vector:
  - 3 reals/2D gridpoint

### **IHtoSC**

For periods of time when the solar wind is not supersonic at the outer boundary of SC, the IH provides the outer boundary conditions for SC, which consists of the solar wind density, pressure, velocity, and magnetic field.

- Density:
  - 1 real/2D gridpoint
- Pressure:
  - 1 real/2D gridpoint
- Velocity vector:
  - 3 reals/2D gridpoint
- Magnetic field vector:
  - 3 reals/2D gridpoint

### **IHtoSP**

IH provides the coordinates of points along magnetic field lines. The points are advected with the plasma flow by IH. In addition, density, pressure, velocity, and magnetic field are provided by IH at each of the points. The advection is completed by SWMF using the velocities from IH.

- Point coordinates:
  - 3 reals/1D gridpoint/ field line
- Density:
  - 1 real/1D gridpoint/ field line
- Pressure:
  - 1 real/1D gridpoint/ field line
- Velocity vector:
  - 3 reals/1D gridpoint/ field line
- Magnetic field vector:
  - 3 reals/1D gridpoint/ field line

### **IHtoGM:**

The GM needs to be driven by the solar wind flow in which it sits. The GM obtains a time and space varying set of data at its inflow boundary, which consists of the solar wind density, pressure, velocity, and magnetic field.

- Density:
  - 1 real/2D gridpoint
- Pressure:
  - 1 real/2D gridpoint
- Velocity vector:
  - 3 reals/2D gridpoint
- Magnetic field vector:
  - 3 reals/2D gridpoint

### **GMtoRB:**

GM indicates to RB if field lines connected to its 2D spherical grid points are open or closed. For closed field lines, GM provides the X and Y coordinates where the field line intersects the equatorial plane, and the magnitude of the magnetic field at the intersection.

- X equatorial:
  - 1 real/2D gridpoint
- Y equatorial:
  - 1 real/2D gridpoint
- B equatorial:
  - 1 real/2D gridpoint

### **IMtoGM:**

GM needs the total pressure calculated by IM to be applied at the GM grid connected to the ionosphere via the magnetic field lines. IM provides the pressure on its 2D spherical grid. The mapping is completed in GM.

- Total pressure:
  - 1 real/2D gridpoint

### **GMtoIM:**

The IM needs the magnetic field line flux tube volumes and the average density and pressure in the flux tubes connected to its 2D spherical grid points. GM indicates to IM if field lines are open or closed. For closed field lines, the integration along magnetic field lines is done in GM. GM also provides the X and Y coordinates where the field line intersects the equatorial plane, and the magnitude of the magnetic field at the intersection.

- Field line volume:
  - 1 real/2D gridpoint
- Average density:
  - 1 real/2D gridpoint
- Average pressure:
  - 1 real/2D gridpoint
- X equatorial:
  - 1 real/2D gridpoint
- Y equatorial:
  - 1 real/2D gridpoint
- B equatorial:
  - 1 real/2D gridpoint

### **IEtoIM:**

This is a one way coupling. IE provides the radial current and the electric potential for the 2D spherical grid of IM. The interpolation between the IE and IM grids is completed by the SWMF coupling toolkit.

- Electric potential:
  - 1 real/2D gridpoint
- Radial current:
  - 1 real/2D gridpoint

### **IEtoGM:**

IE provides the electric potential computed on its 2D spherical grid. The potential is mapped to the inner boundary of GM and is used for calculating the electric field and velocity at the inner boundary. The mapping is done by SWMF, assuming a dipole field.

- Electrical potential:
  - 1 real/2D gridpoint

### **GMtoIE:**

The IE requires the field aligned currents on its 2D spherical grid. The currents are calculated in GM at an appropriate radial distance (e.g. 3.5 Earth radii), and mapped along the dipole field lines of the planet to the ionosphere. The field aligned currents are calculated in IE from the information provided by GM.

- Current:
  - 3 reals/2D gridpoint
- Magnetic field:
  - 3 reals/2D gridpoint

### **IEtoUA:**

The electric potential calculated by IE is mapped to the upper atmosphere model in each grid cell along the magnetic field lines. The gradient of the potential is taken to calculate the electric field which is used to drive the ion motion. The mapping along magnetic field lines is completed in UA.

- Electrical potential;
  - 1 real/2D gridpoint;

### **UAtoIE:**

The upper atmosphere model provides IE the Pederson and Hall conductivities and field aligned currents (in addition to the field aligned currents obtained from GM).

- Neutral wind field-aligned current:
  - 1 real/2D gridpoint
- Conductances (Hall, Pederson, and field aligned):
  - 3 real/2D gridpoint

### **CONtoPM:**

CON initializes, schedules execution and coupling, finalizes the physics components. It also requests saving restart files periodically.

#### **Initialize**

- Number of processors:
  - 1 integer
- MPI communication group:
  - 1 integer
- Component ID:
  - string (2 chars)
- Output directive:
  - enumeration (STDOUT, FILEOUT)
- Unit number for FILEOUT option
  - 1 integer
- Input parameters: actual parameters for each active component
  - strings (100 char) / per line

**Couple components:** The components exchange data as instructed by the coupler.

- Coupling time:
  - 1 real

**Execute component:** The component executes one time step not exceeding simulation time limit.

- Simulation time:
  - 1 real
- Simulation time limit:
  - 1 real

**Save/Restart:** The component saves its current state into restart files.

- Simulation time:
  - 1 real

Finalize component: The component finalizes execution.

- Simulation time:
  - 1 real

### **PMtoCON:**

PM provides version information and its simulation time to CON.

#### Registration

- Actual or 'empty' component:
  - 1 logical
- Version name:
  - string (40 chars)
- Version number:
  - 1 real

Current simulation time: Sent after completion of each time step.

- Simulation time:
  - 1 real

### **CONtoGUI:**

CON provides the current status of execution of the framework, and runtime statistics for possible display in the GUI.

- currStat: current status of framework executable and all physics model components
- runStat: runtime statistics

### **GUItoCON:**

The GUI provides two files to CON. The first, LAYOUT.in, describes the distribution of physics components to the processing elements (PEs). The second, PARAM.in, controls execution by defining timing and session control, boundary conditions, included physics components, and required coupling, among other input parameters.

- LAYOUT.in
  - List of registered components
  - Number of processors for each registered component
  - Root processor for each registered component
  - Processor rank stride for each registered component
- PARAM.in: Note that the list of parameters in PARAM.in is lengthy. For simplicity, the following enumeration is NOT exhaustive, but is meant to be representative of actual parameters.

#### General commands

- NameIncludeFile
- UseStrict

#### Time and session control

- DoTimeAccurate
- TSimulation
- NStep
- MaxIteration
- ...

#### Testing and timing

- StringTEst
- LVerbose
- UseTiming
- DnTiming
- ...

#### Initial and boundary conditions

- IProblem
- TypeCoordSystem
- NameRestartInDir
- DoRestartBFace
- XMin

- XMax
- YMin
- YMax
- ZMin
- ZMax
- ...

#### Component control

- NameComp
- UseComp
- DnRun

#### Coupling control

- NCouple
- NameSourceTarget
- DnCouple
- DtCouple
- ...

#### Restart control

- DoSaveRestart
- DnSaveRestart
- DtSaveRestart

#### Output control

- DoEcho
- UseStdout
- NameStdOutDir

#### Planet commands

- NamePlanet
- RadiusPlanet
- MassPlanet
- OmegaPlanet
- TiltRotation
- TypeBField
- MagAxisPhiGeo
- DipoleStrength
- UseRotation
- RotationPeriod
- IsRotAxisPrimary
- RotAxisTheta
- RotAxisPhi
- IsMagAxisPrimary
- MagAxisTheta
- MagAxisPhi

### **PMtoOUF:**

Physical components write verbose information into the unit number provided by CONtoPM when output directive is set to FILEOUT. Data files and restart files are written into the directory named by the component.

- Verbose messages:
  - mixed species data
- Data files:
  - mixed species data
- Restart files:
  - mixed species data

### **CONtoOUF:**

CON saves restart information into an ASCII file (RESTART.out), which contains the following:

- Description of run:
  - string (100 chars)
- Planet name:
  - string (100 chars)
- Starting date and time:
  - 6 integers and 1 real
- Number of steps:
  - 1 integer
- Simulation time:
  - 1 real
- Version number:
  - 1 real
- Precision:
  - enumerated (4 or 8)

### **OUFtoGUI:**

The output files include raw data files (mixed species data) which can be used to generate plots. The GUI provides an interface for processing the output files and generating plot images, which are then presented to the user.

- JPEG files:
  - JPEG images, compressed and suitable for viewing in a web browser
- Postscript files
  - High-resolution images suitable for use at conferences, in journals, etc.

### **INFtoPM:**

External science data is ingested into the SWMF in various forms. The dataflow is generically represented as INFtoPM. Some examples types of data that are currently consumed by Physics components that are in the framework include:

- Magnetograms: consumed by the Solar Corona model;
- Interplanetary Magnetic Fields (IMF) Files: consumed by the Global Magnetosphere and Upper Atmosphere models; and
- Satellite Trajectory Data: consumed by the Global Magnetosphere, Upper Atmosphere, Inner Heliosphere, an Solar Corona models.



## Framework State Behavior

This section contains a detailed description of the state behavior of the SWMF. That is, the previous discussions discussed functional relationships between various architectural partitions of the SWMF. This section discusses the temporal (i.e., sequential) relationships between the various architectural partitions.

The form of this section is traditional flowchart diagrams followed by verbal descriptions. The conventions used for the flowcharts are traditional, however, a brief description is given for clarity. Starting and ending points of process steps are indicated by labeled ovals/lozenges. Subroutines are indicated by labeled rectangles. Rectangles with slightly rounded corners (e.g., delete SWMF\_STOP in Figure 5) indicate separate processes or in some cases notes that are added for clarity. The diamond is used to indicate an alternative statement or choice. A parallelogram indicates a loop variable that is typically incremented on each execution of the contained loop. The pentagon, and alternatively a labeled arrow, is used to show a continuation of execution between different pages of a flowchart. Finally, a nested calling structure is shown by nesting subroutine rectangles within each other. As an example of nesting, Figure 6 shows `set_param_comp`, which calls `set_param_id`, which calls `get_comp_info`, which queries the information for any registered component.

## Overall Execution Model

Given this brief discussion of notation conventions, the state behavior of the SWMF is described below. Figure 5 is the overall execution model for the SWMF. Consider this diagram as the ‘macro’ model of execution, where further detail is given in the subsequent models shown in Figure 6 through Figure 9. As an example, there is a ‘read\_inputs’ subroutine shown in Figure 5. Figure 7 shows greater detail for ‘read\_inputs’, and in fact describe the complexity involved in this partition of source code. That is, if Figure 5 is considered the ‘macro’ view, then Figure 6 through Figure 9 are considered the ‘micro’ view of execution of the SWMF.

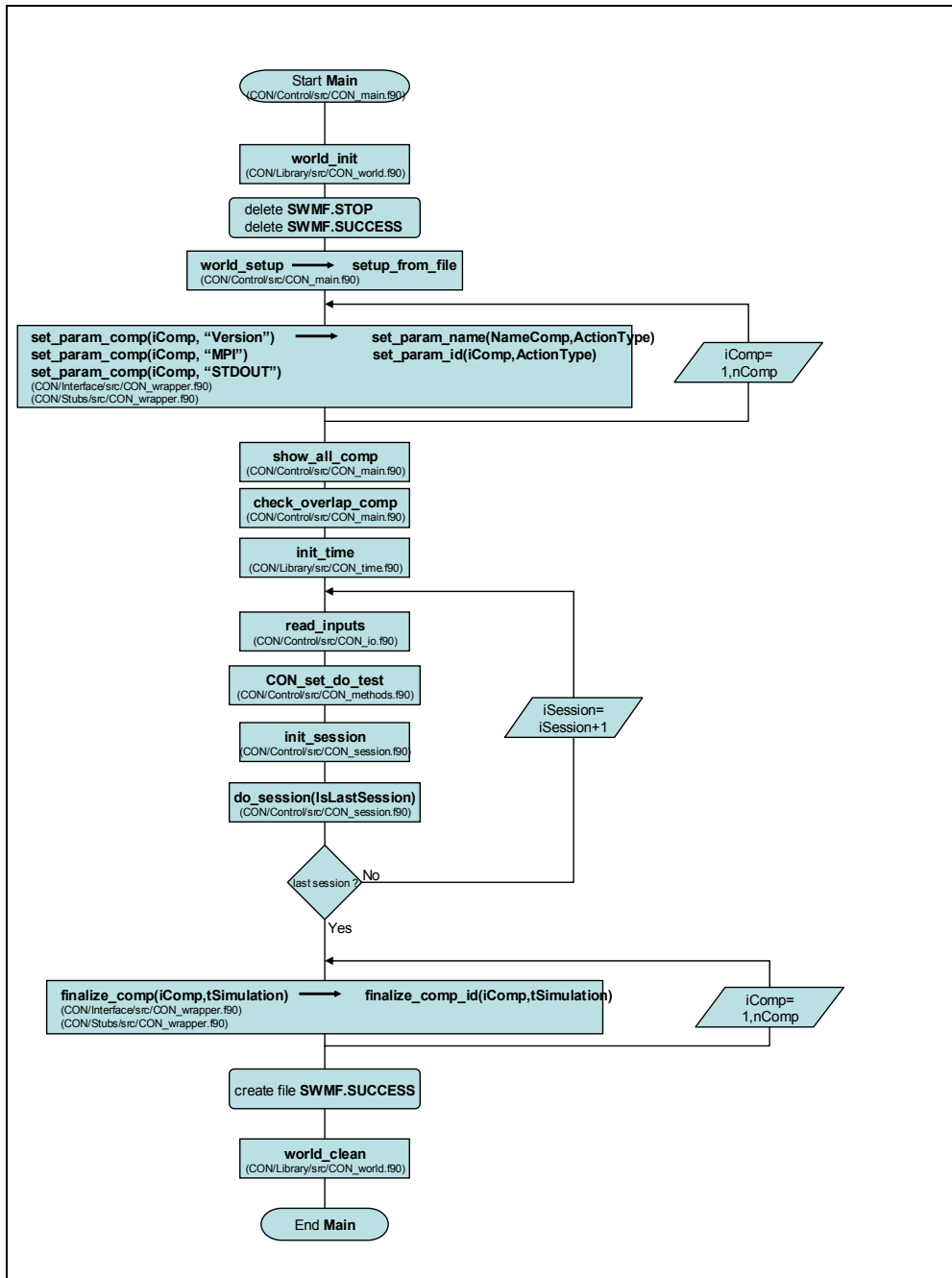


Figure 5. Overall state behavior for SWMF.

Figure 5 shows the overall flow of execution for the SWMF. The execution is described from top to bottom. The top four ‘objects’ (from Start Main to world\_setup) are initialization steps for the SWMF, including deleting artifacts from previous runs.

The set\_param\_... subroutine iterates through all components assigned to execute for this particular execution model. Further detail about set\_param\_... is given in Figure 6. After the physics components are assembled into the execution model, then further initialization is required for the included components, including time initialization and checking for more than one module is running on a given processor (aka overlap).

Next, inputs are read for each session (read\_inputs). Based upon the input test string, various debug variables and output are set (CON\_set\_do\_test). The session is initialized (init\_session) and the session is executed (do\_session). Details about read\_inputs are shown in Figure 7. Details about init\_session are shown in Figure 8. And, details about do\_session are shown in Figure 9.

After session execution is complete, each component is finalized (finalize\_comp). Finally, output files are created and computer memory is cleaned.

### Execution Model for Setting Parameters

Figure 6 shows how parameters are set for execution of the SWMF. As mentioned, Figure 6 is a ‘magnification’ of the set\_param\_... subroutine in Figure 5. As such, the process of setting parameters begins with execution of set\_param\_comp and ends execution with completion of set\_param\_comp.

The levels of nesting show the intricacies in the execution of setting parameters. The nesting indicates a calling structure to the code. As mentioned, an *enclosing* box calls the *enclosed* box. Related to Figure 6, set\_param\_comp is an interface which can be called with either a string or an integer as the first variable. If called with a string, set\_param\_name is called to resolve the integer that matches the string. set\_param\_id is then called with the integer argument. In turn, set\_param\_id calls get\_comp\_info and then executes parameter setup for each physics component that is included in the execution model by calling GM\_set\_param, IE\_set\_param, ... Within get\_comp\_info is the routine comp\_info\_get, which in turn calls get, which is a method of CON\_comp\_info. As noted in the diagram, the nesting structure results in MPI, unit, name, and version variables being set for the components.

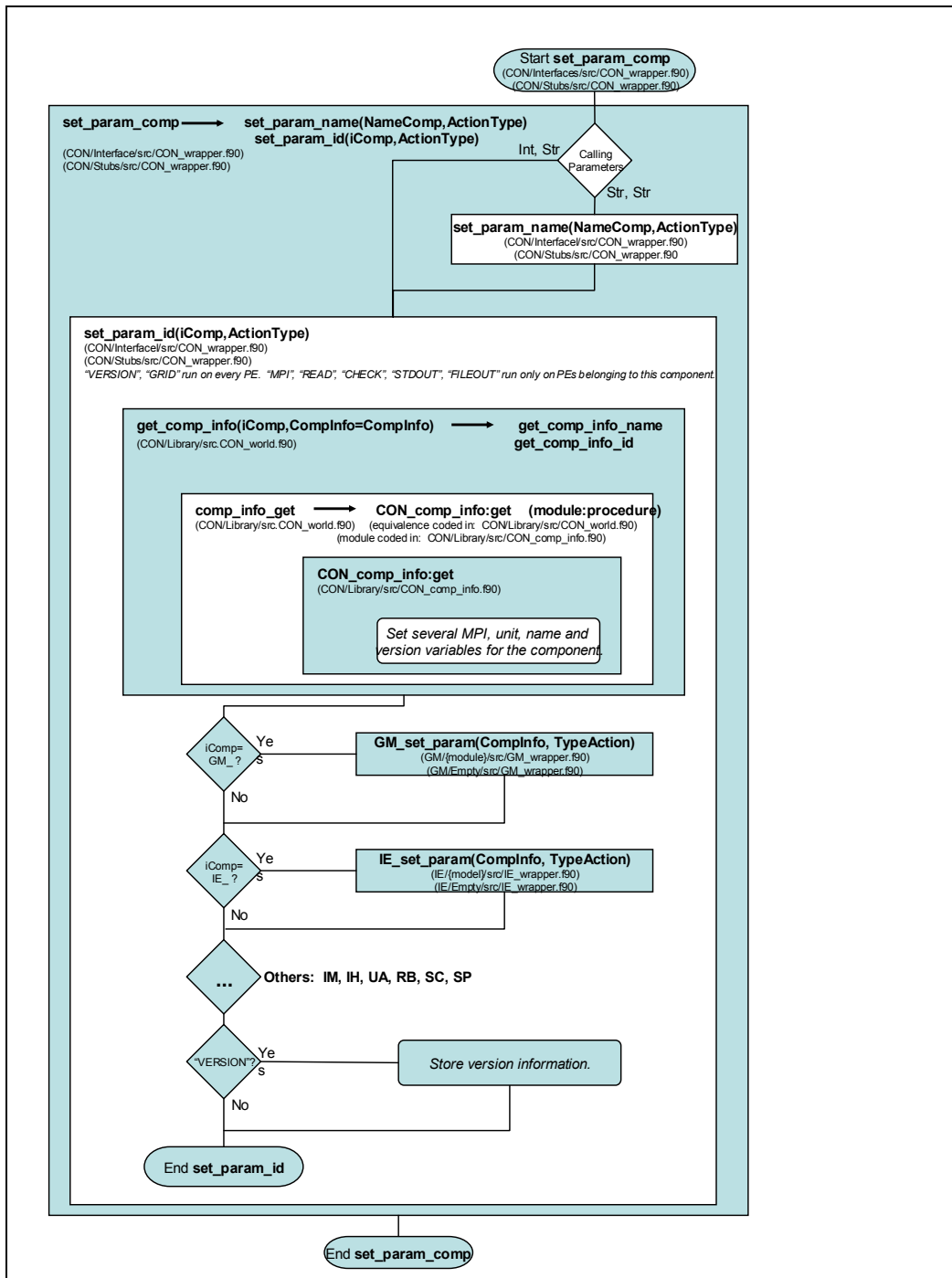


Figure 6. Flowchart for setting parameters.

## Execution Model for Reading Inputs

Figure 7 is a flowchart diagram showing how inputs are read into the SWMF. The figure is partitioned into two parts – the left-hand side (LHS) and right-hand side (RHS). Note the connection from the LHS to the RHS through the read\_inputs #1 pentagon. Also, note the connection from the RHS to the LHS through the read loop variable. The description below first describes the LHS of Figure 7, then describes the RHS of Figure 7.

Reading inputs begins by determining whether or not inputs have previously been read. After this decision, input parameters are read for CON, each physics component, and physics component coupling. Input parameters include directives for each physics component such as output file location, number of couplings between components, and frequency of coupling between components.

Frequency is adjusted as needed to enable coupling between physics components. The nested loop is required to allow for each physics component to couple with each other physics component.

Additional variables are read related to planets and axes. Finally, parameters are set for each of the physics components and overall timing variables are set. Upon completion of read\_inputs, the SWMF is ready for session initialization as described in the next section.

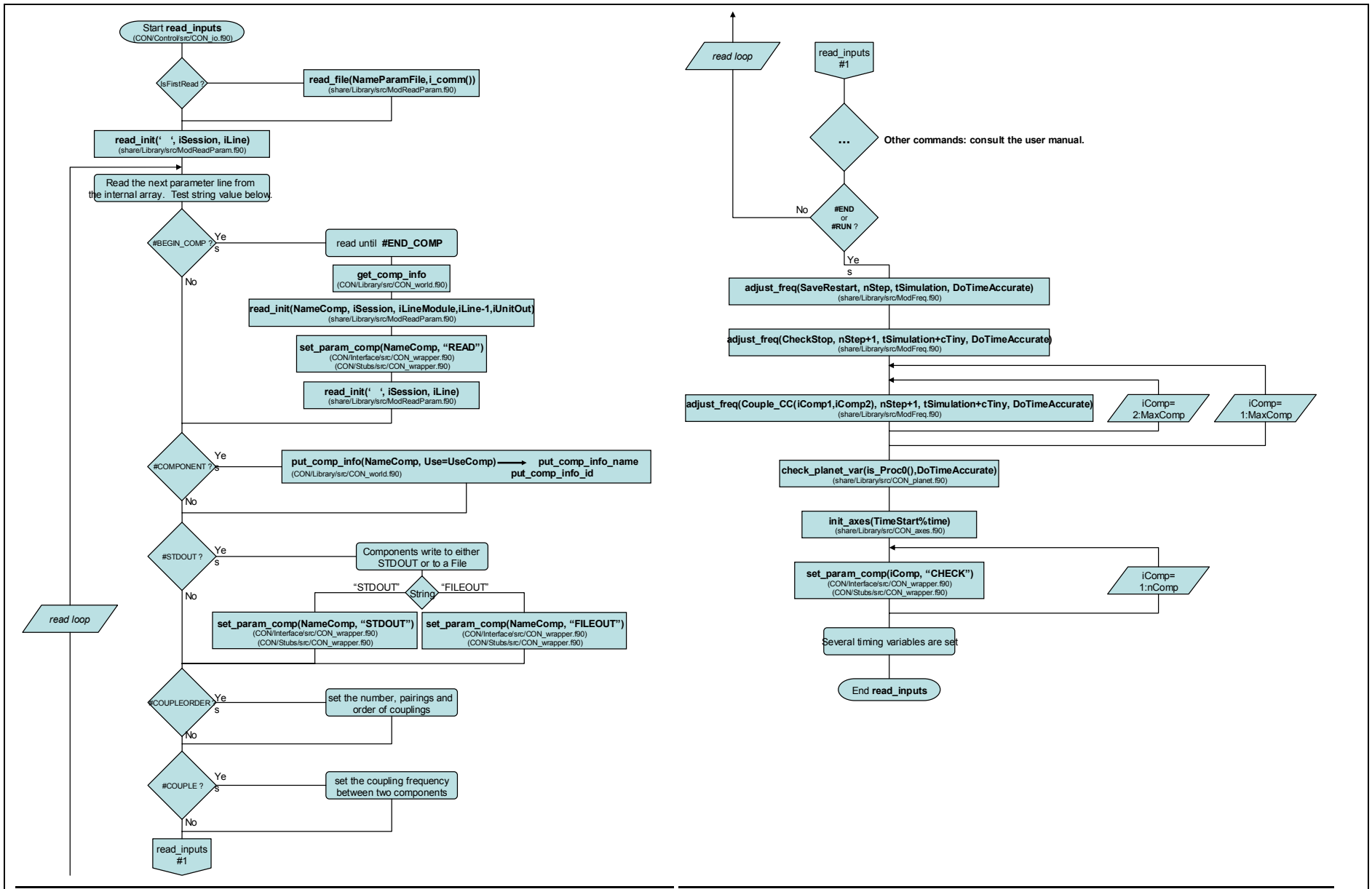


Figure 7. Flowchart for reading input.

## Execution Model for Initializing a Session

Figure 8 is a flowchart diagram showing the initialization of a session. As with the previous diagram, there is a logical flow from the bottom of the left-hand side (LHS) to the top of the right-hand side (RHS). Note the connection from the LHS to the RHS through the `init_session #1` pentagon.

`init_session` begins with the initialization of each of the physics components included in the execution. That is, if the Global Magnetosphere physics component is included in the session, then the Global Magnetosphere model is initialized. This pattern is similar for each subsequent physics component. To complete the LHS of Figure 8, the parameters for each of the physics components are set.

On the top of the RHS of Figure 8, coupling is initialized as requested between pairs of physics components. Finally, the general session is initialized after each of the individual physics components and their respective couplings are initialized. The `init_session_general` process distributes the physics components to processing elements (PEs), ensures that all available PEs are used by the run, and initiates coupling between components.

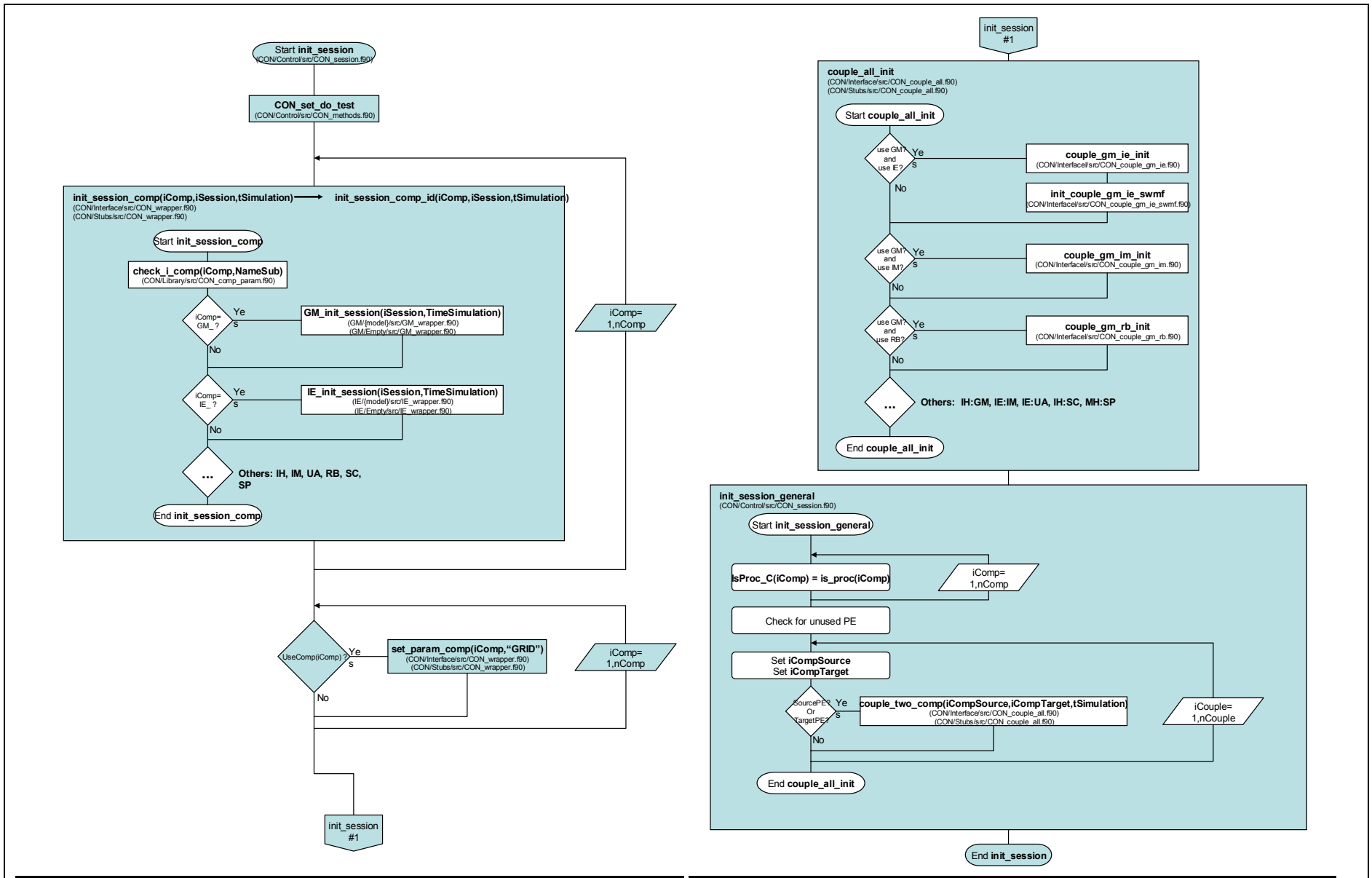


Figure 8. Flowchart for initializing a session.



## Execution Model for Executing a Session

Figure 9 is a flowchart diagram showing the execution of a SWMF session. As with some previous diagrams, the figure is partitioned into two parts – the left-hand side (LHS) and right-hand side (RHS). Note the connection from the LHS to the RHS through the `do_session #1` pentagon. Also, note the connection from the RHS to the LHS through the `TIMELoop` variable. The description below first describes the LHS of Figure 9, then describes the RHS of Figure 9.

The `do_session` is the heart of the SWMF where modules are allowed to perform computational time steps and where coupling is initiated and controlled. The general session begins execution, then conditions are checked for terminating execution (e.g., `MaxIteration`, ..., `CPUTime`). If none of the termination conditions are met, then execution continues by incrementing some counters. If the SWMF is executing in time accurate mode, then the SWMF completes additional work to determine specific details regarding the next time at which an event can interrupt execution (coupling, writing restart files, checking CPU time-based stopping conditions, etc.).

Continuing with the RHS of Figure 9, the SWMF asks each physics component to execute a single time step. The size of the step depends upon whether or not the execution is time accurate. Progress is displayed for the execution of the session.

At the end of each iteration, coupling occurs between each appropriate pair of physics component. That is, unique PEs of the first component communicate directly with unique PEs from the second component. Finally, restart files are created for each component. The process that is described continues until one of the initial termination conditions is met.

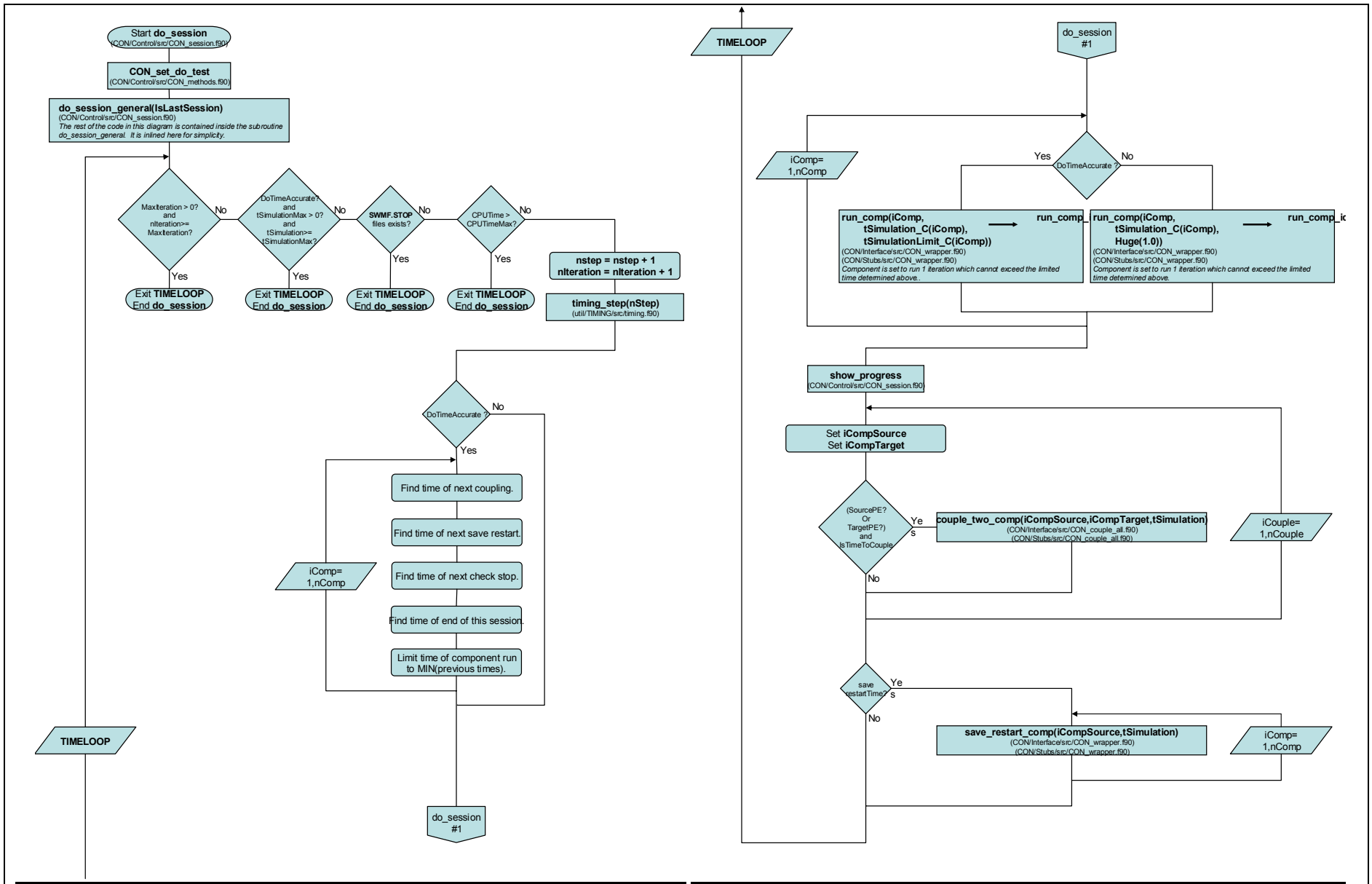


Figure 9. Flowchart for executing a session.

## Physics Component View of Execution Model

Figure 5 through Figure 9 showed the overall sequence of execution from the perspective of the SWMF. Figure 10 (below) shows the sequence of execution from the perspective of an individual physics component. That is, the earlier diagrams show an orthogonal view from the figure below. Specifically, Figure 10 shows an example sequence of subroutine calls for the execution of the Global Magnetosphere (GM) physics component. The subroutine names are consistent with those used in previous flowchart diagrams.

First, parameters for the physics component are set, such as version number, MPI, and output type. Several other parameters are set within the session loop, including read, check, and grid directives. The GM physics component is coupled with other physics components, such as the Ionospheric Electrodynamics and Inner Magnetosphere physics components. Note that the requested couplings have been previously determined in `read_inputs`.

The specific physics component (in this case, GM) executes, couples, and saves restart files in the time loop. The physics component is finalized after exiting both the session and time loop.

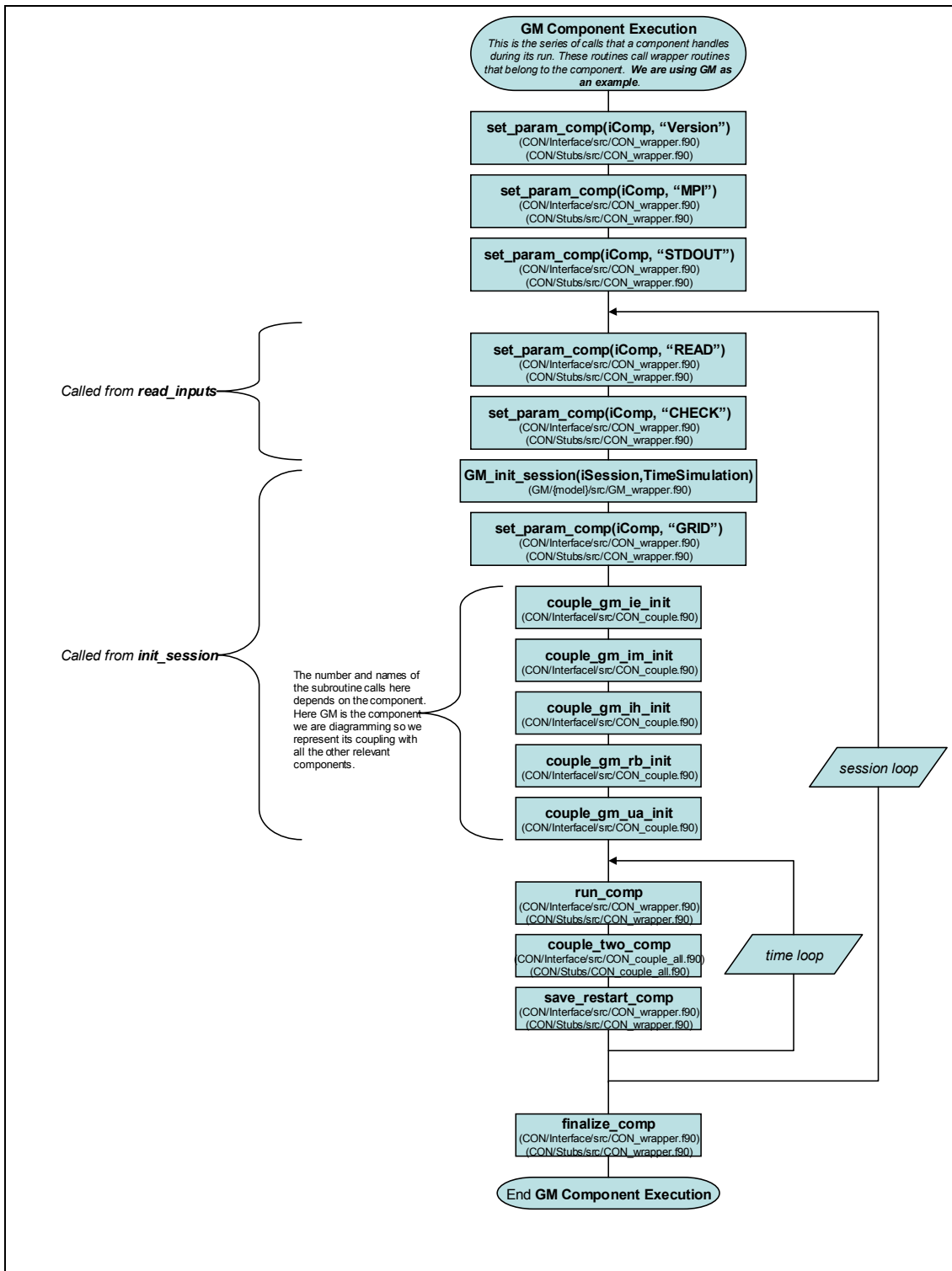


Figure 10. Flowchart diagram of execution from Physics Component viewpoint.

## References

- [1] Tamas Gombosi, et al, *Space Weather Modeling Framework Project Proposal*, Center for Space Environment Modeling, The University of Michigan, Ann Arbor, Michigan, 2001.
- [2] David Chesney, et al, *Space Weather Modeling Framework Software Engineering Plan*, Center for Space Environment Modeling, The University of Michigan, Ann Arbor, Michigan, 2001.
- [3] David Chesney, et al, *Space Weather Modeling Framework Requirements Document, v1.7*, Center for Space Environment Modeling, The University of Michigan, Ann Arbor, Michigan, 2002.
- [4] David Chesney, et al, *Space Weather Modeling Framework Design Document, v1.7*, Center for Space Environment Modeling, The University of Michigan, Ann Arbor, Michigan, 2002.
- [5] Volberg, O., D. R. Chesney, D. L. De Zeeuw, K. C. Hansen, K. Kane, R. Oehmke, A.J. Ridley, I.V. Sokolov, G. Toth, T. Weymouth, *Space Weather Modeling Framework: Design Policy for Interoperability*, Center for Space Environment Modeling, The University of Michigan, Ann Arbor, Michigan, 2002.
- [6] Kevin Kane, et al, *Space Weather Modeling Framework Graphical User Interface Requirements Document*, Center for Space Environment Modeling, The University of Michigan, Ann Arbor, Michigan, 2003.
- [7] Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide*. Addison Wesley, 1999, ISBN 0-201-57168.
- [8] James Rumbaugh, Ivar Jacobson, and Grady Booch, *The Unified Modeling Language Reference Manual*. Addison Wesley, 1999, ISBN 0-201-30998-X.
- [9] Toth, G., O. Volberg, A. Ridley, *Space Weather Modeling Framework Manual, Code version 1.0*, Center for Space Environment Modeling, the University of Michigan, Ann Arbor, Michigan, 2003.
- [10] Volberg, O., G. Toth, I. V. Sokolov, A. J. Ridley, T. I. Gombosi, D. L. De Zeeuw, K. C. Hansen, D. R. Chesney, Q. F. Stout, K. G. Powell, K. Kane, R. C. Oehmke, *Doing it the SWMF Way: From Separate Space Physics Simulation Programs to The Framework for Space Weather Simulation*, Fall AGU Meeting, San Francisco, 2003.
- [11] Hill, C., C. DeLuca, V. Balaji, M. Suarez, A. da Silva, and the ESMF Joint Specification Team, *The Architecture of the Earth System Modeling Framework*, Computing in Science and Engineering, Volume 6, Number 1, 2004.