Candidate Frameworks for TRANSP

S. Jardin

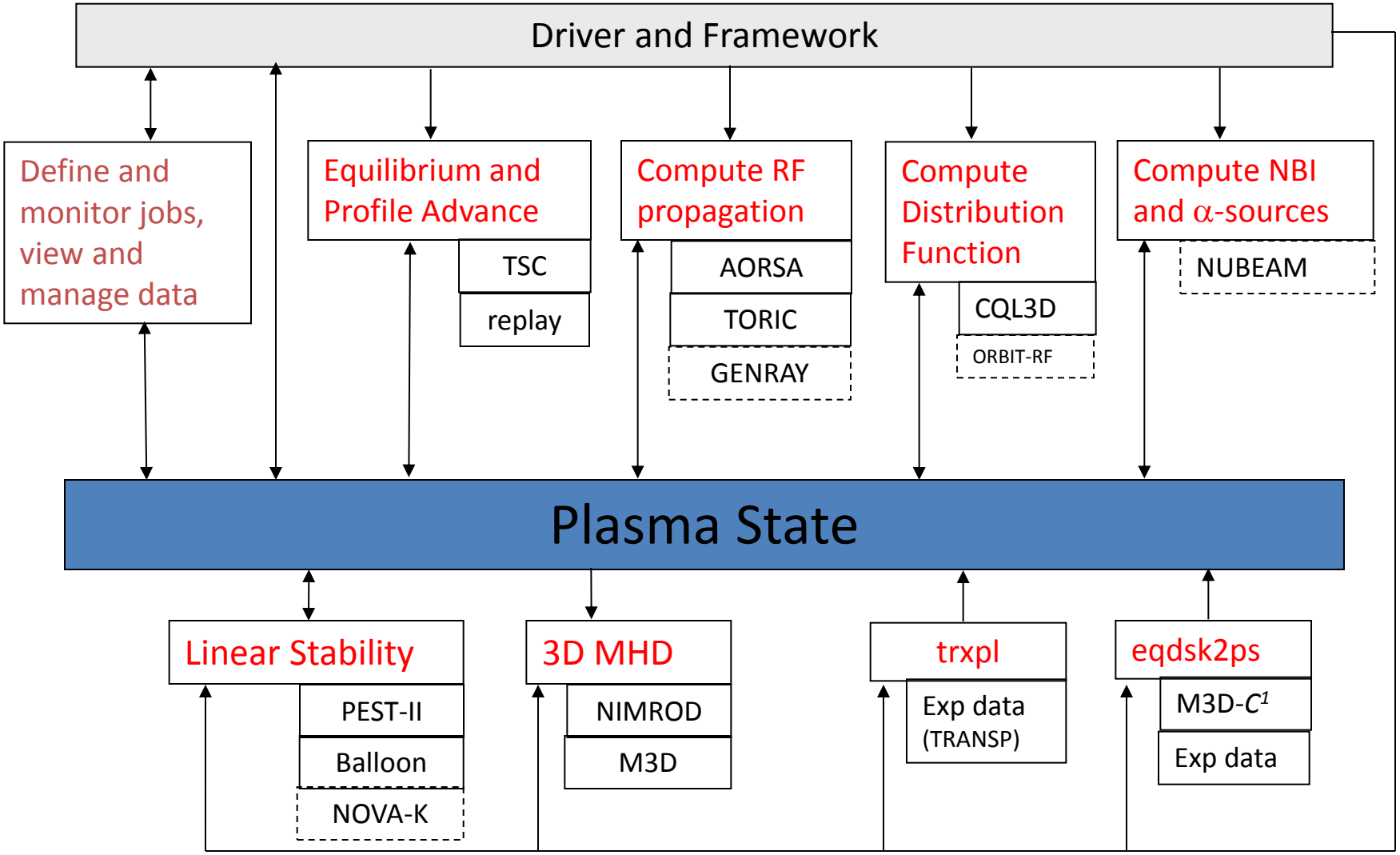
3/23/2015

PPPL

Considerations for developing a (new) Framework?

- Managing complexity: A good framework will
  - assist in isolating code components that can be individually maintained by experts
  - allow new and improved physics modules to be seamlessly integrated
  - provide user with a transparent and configurable workflow
- Common data structures
  - Aids in importing/exporting data with other codes
  - Facilitates code benchmarking and validations
- Effective parallelization
  - Efficient, load balanced, multiple levels, largely transparent to user
- Can we make use of existing Fusion Code frameworks?
  - SWIM/AToM IPS
  - ITER IMAS
  - OMFIT

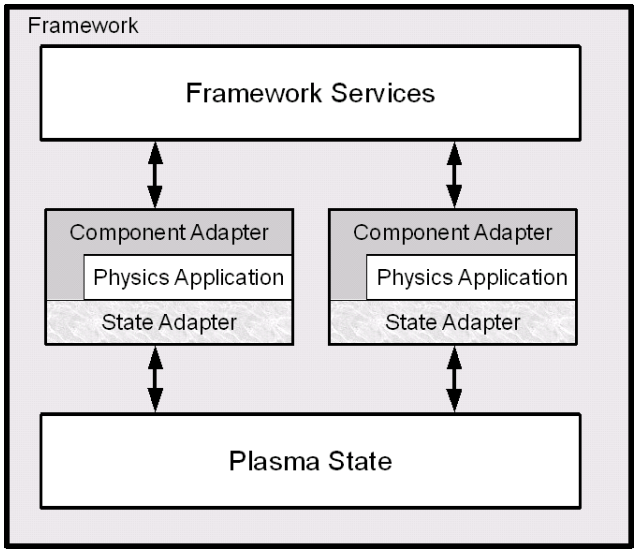
SWIM/AToM Integrated Plasma Simulator



IPS Components and Framework

- **Components**
  - Physics application code with adapters
  - State adapter converts native physics code data to/from plasma state format
  - Component adapter converts between declared interface and code inputs
  - Three interface functions: init(), step(), finalize()

- **Framework**
  - Lightweight, supports HPC environments
  - Allows multiple parallel tasks (components) to execute concurrently
  - Framework services: task manager, resource manager, data manager, configuration manager, event service
  - Implemented in Python



ITER Integrated Modeling and Analysis Suite (IMAS)

Acronyms and concepts described on following 3 vgs:

IDM	-	ITER Data Model
IDS	-	Interface Data Structure
PUAL	-	Physics User Access Language
IMAS	-	Integrated Modeling and Analysis Suite
Component		Physics code with compliant IO
Kepler	-	Software tool for connecting components

ITER DATA MODEL (IDM)

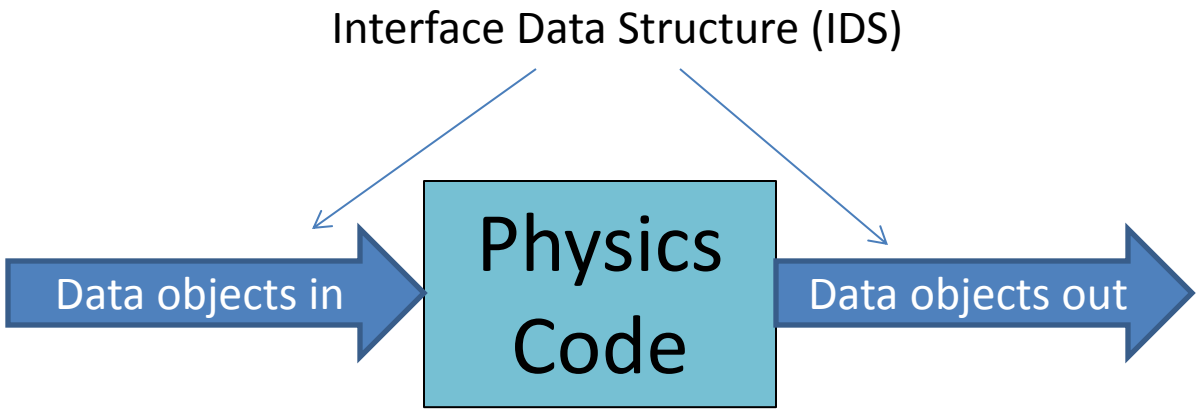
- Tree-structure data model that is already being used in the simulation codes.
- All ITER-specific data that a code (or component) reads or writes should be compatible with this model.

- For a given component, the actual data read or written by the component is called the Interface Data Structure (IDS) for that component. The IDS is a subset of the full ITER Data Model

- For example, in FORTRAN: equilibrium%profiles_1d%phi
  - 1D array that contains the toroidal flux at every flux surface

- Several new tokamaks are planning on adopting this:
  - JT-60SA, WEST, MAST?

All physics components must be of this form

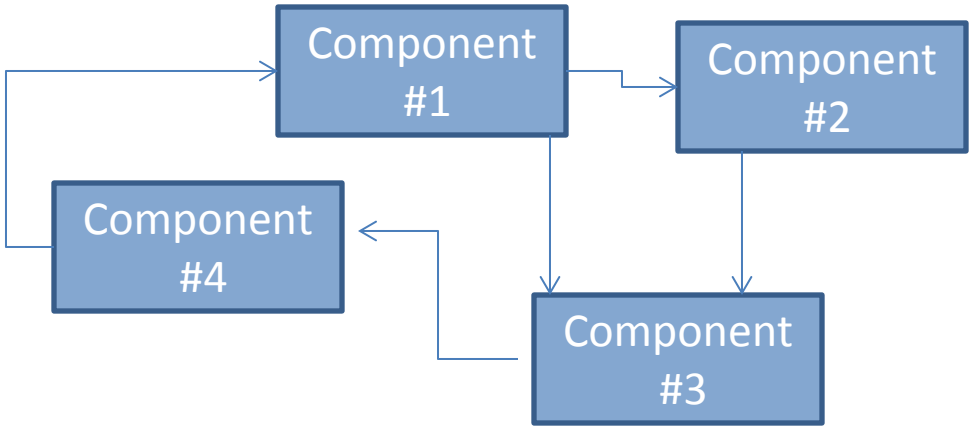


- The Physics User Access Language (PUAL) is a set of Fortran commands that provide a way to import and export data from the IDS: put(….), get(…..), etc.

- Once a physics code is IDS-compliant, a utility called a *component generator* will turn this code into a *component* which Kepler recognizes

Integrated Modeling and Analysis Suite (IMAS)

IMAS is a Kepler based framework for coupling components



Kepler allows you to graphically configure a workflow once components are defined and their interface data structure IDS is document and registered.

The ITER Integrated Modeling (IM) activity at ITER will be centered on using IMAS to develop workflows to do transport modeling, equilibrium and stability analysis, and other modeling and analysis functions.

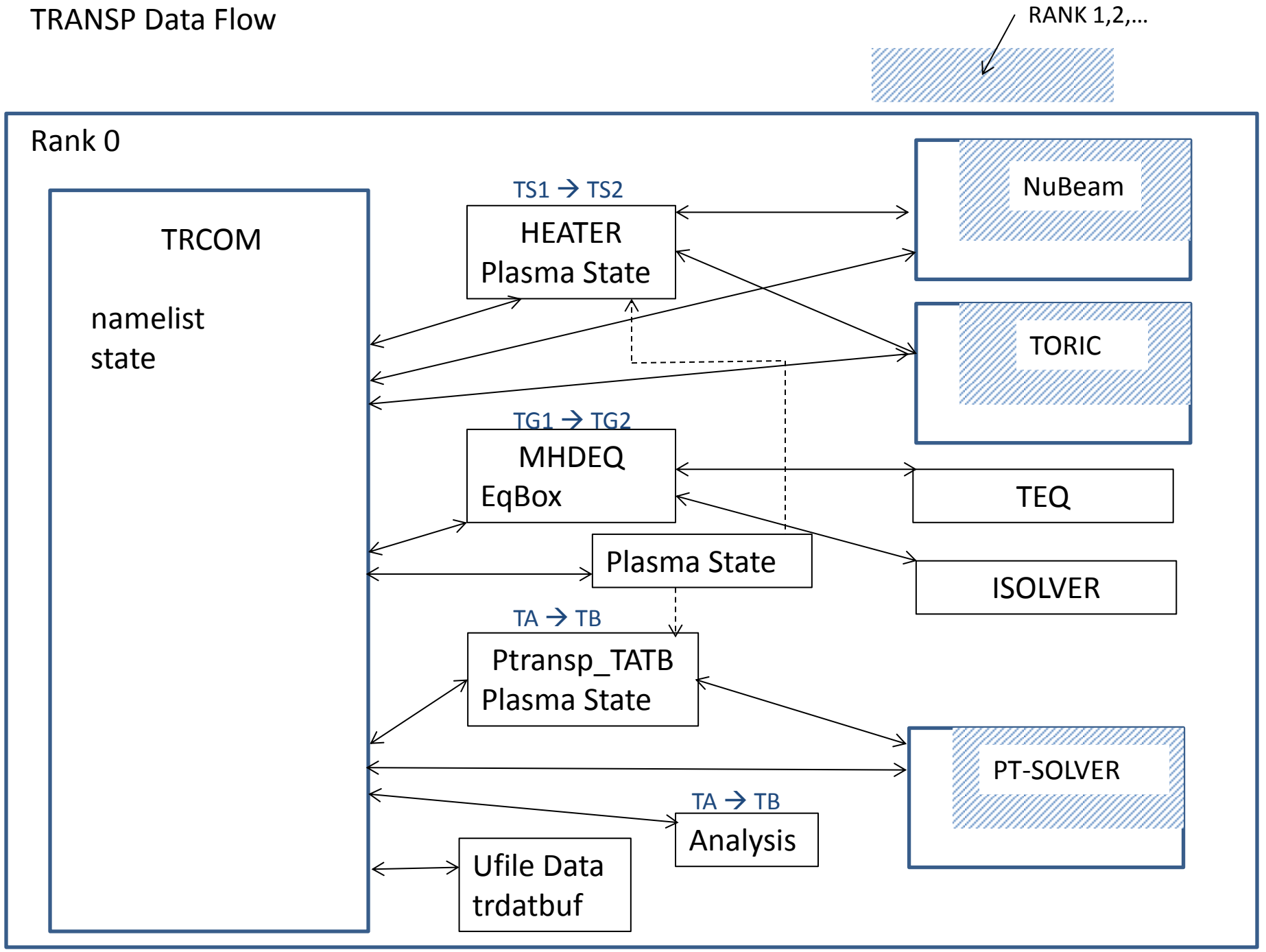
	Integrated Plasma Simulator (IPS)	ITER Integrated Modeling and Analysis Suite (IMAS)	TRANSP
Workflow manager	Python-based driver component controls Configuration, Task, Data , Resource, and Event Managers	Kepler graphical interface workflow engine	Custom UNIX shell scripts to process input, compile and run custom executable based on namelist, and process output
Data Model	Plasma State data structure is encouraged and widely used, but IPS imposes no particular data model	ITER Data Model (IDM) uses tree structure data dictionary	Input Ufiles created from MDSPlus data. Output primarily NetCDF. Utilities for extracting time slice data.(eqdsk, plasma state,…)
Packaging a physics code	Python based wrappers that read/write to plasma state and prepare and process code output	Physics User Access Language (PUAL) provides Fortran commands to import and export IDM data via Interface Data Structures (IDSs)	1. Fortran driver to call physics code as a subroutine with data passed via core. 2. Systems call to separate executable. Data via files.
Support for parallelism	4-levels: within a component, multiple instances, concurrent components, parameter studies; asynchronous tasks and event-driven work flows	Will have parallel capability, but not yet demonstrated.	Master-slave. Individual modules have multi-levels of parallelism.

	Integrated Plasma Simulator (IPS)	ITER Integrated Modeling and Analysis Suite (IMAS)	TRANSP
Output	Multiple plasma states, web portal with graphical output. Component output files.	Support for storing IDM data, graphical output	Netcdf…see above
Availability	Portable, IPS requires only Python. Components usually require Fortran. Maintained installations at PPPL, ORNL, NERSC	Now only at ITER IO. Soon to be installed on member local clusters	Installed at PPPL, NERSC, JET. Available via open science grid to users throughout the world.
Software Management tools	SVN based, Issue tracking	GIT based, Issue tracking, automatic building & regression testing	SVN based, automated issue tracking (triage), automatic building and regression testing.
Usage	SWIM/AToM IPS users with TSC, FASTRAN, EPED, TORIC, AORSA, NUBEAM, GENRAY, CQL3D, NIMROD, DAKOTA	A few users at ITER IO. Demoed with CORSICA, DINA. Plans for usage by members in FY15.	Hundreds of users with accounts.

OMFIT

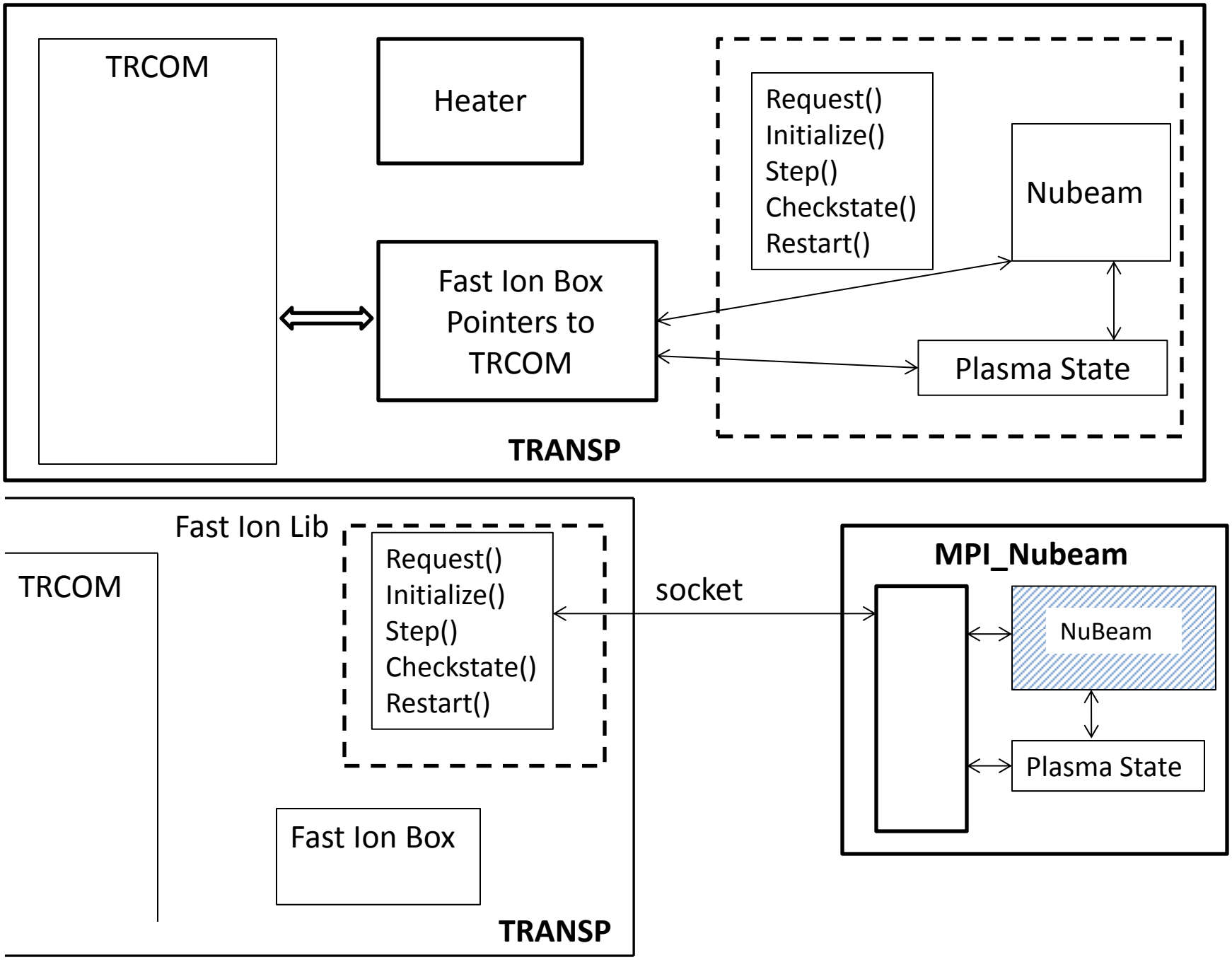
- *Workflow manager* designed to simplify and automate common workflows involving physics application codes and provide an interface to experimental data.
- Users can supply python modules (or customize existing modules) that execute a given workflow. All modules use their own data formats (not imposed by OMFIT). Sharing of modules is encouraged.
- Framework provides high-level functionalities allowing user to execute codes on remote servers, submit and monitor jobs on HPC batch systems, create custom GUIs, post-process and visualize data, seamless integration with MDSplus
- Top-level GUI to manage data, run simulations, analyze data
- Emphasis on up-to-date documentation (including YouTube videos), issue tracking
- Common workflows that have been implemented include:
  - Kinetic EFIT equilibrium reconstruction including NEO bootstrap current model
  - Core and edge stability studies involving CORISCA equilibrium, DCON and GATO stability
  - Parametric studies of transport and heating
  - RMP induced transport and rotation by coupling M3D-C1 and NTV torque models
  - Steady-state transport modeling involving TGYRO, EFIT, NEO, TGLF, ONETWO
- Emphasis has <u>not</u> been on time-dependent modeling
  - Plans to couple with the IPS in AToM to execute a ITER transport simulation
  - Ability to set up and execute a simulation and access ITM data

TRANSP Data Flow



TRANSP Plugin Framework

- TRANSP runs as a serial process
- TRANSP would no longer contain nubeam, toric, teq, …
- Dynamically loadable libraries would be responsible for calculating Fast Ions, RF heating, equilibriumj, …
  - Set in namelist ➔ dependency injection
  - Can be system or user supplied
- Subset of TRCOM available to libraries
  - Implemented with smart pointers
  - serializable


TRCOM
Heater
Request()
Initialize()
Step()
Checkstate()
Restart()
Nubeam
Fast Ion Box
Pointers to
TRCOM
Plasma State
TRANSP
Fast Ion Lib
TRCOM
Request()
Initialize()
Step()
Checkstate()
Restart()
socket
MPI_Nubeam
NuBeam
Plasma State
Fast Ion Box
TRANSP

Possible component libraries for FastIonLib

- Directly load in code, nubeam runs serially
- Invoke MPI nubeam through a system call and file communication
- Start a separate MPI nubeam process and communicate with socket
- Implement an IPS interface and invoke nubeam with task manager
- Implement ITER IMAS interfacce as a Kepler component
- Connect to users matlab code

Other libraries

- Data sent to server for exascale analysis
- Control algorithms
- Interacctive control and examination of data
- Data sent to web page or emailed to user for job monitoring