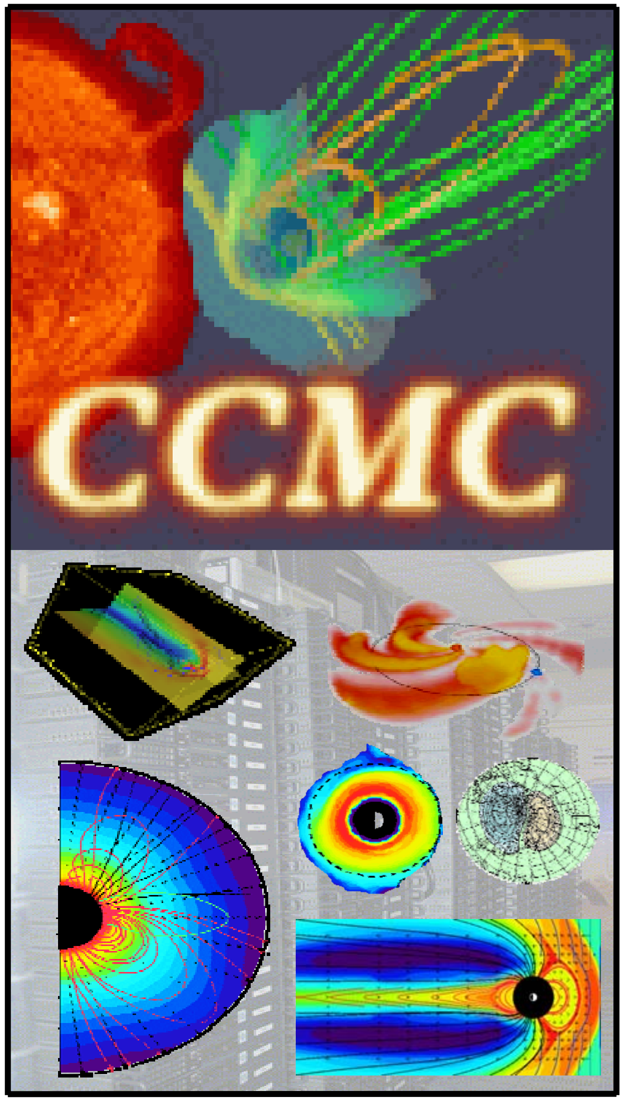


Frameworks : What are They and What Can They do

Peter MacNeice

CCMC Workshop

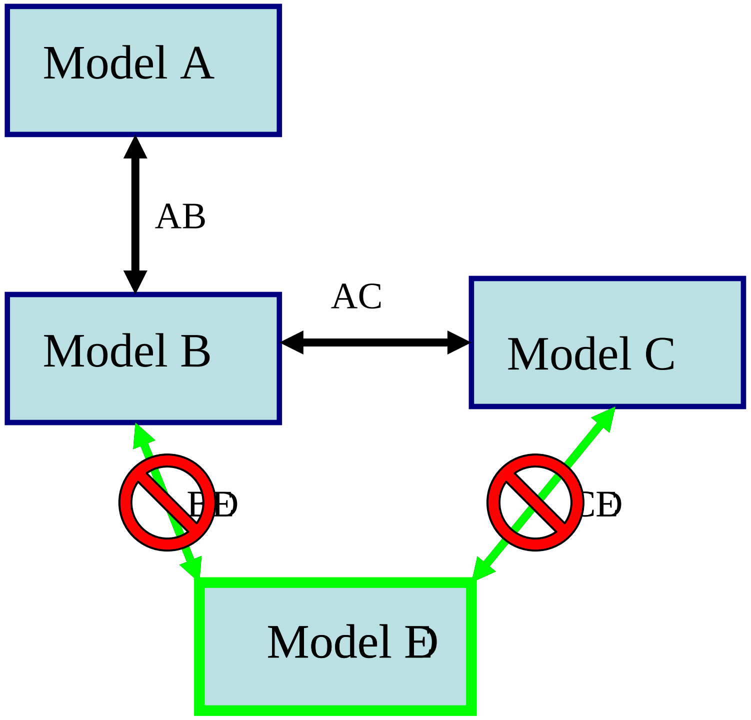
Clearwater Beach

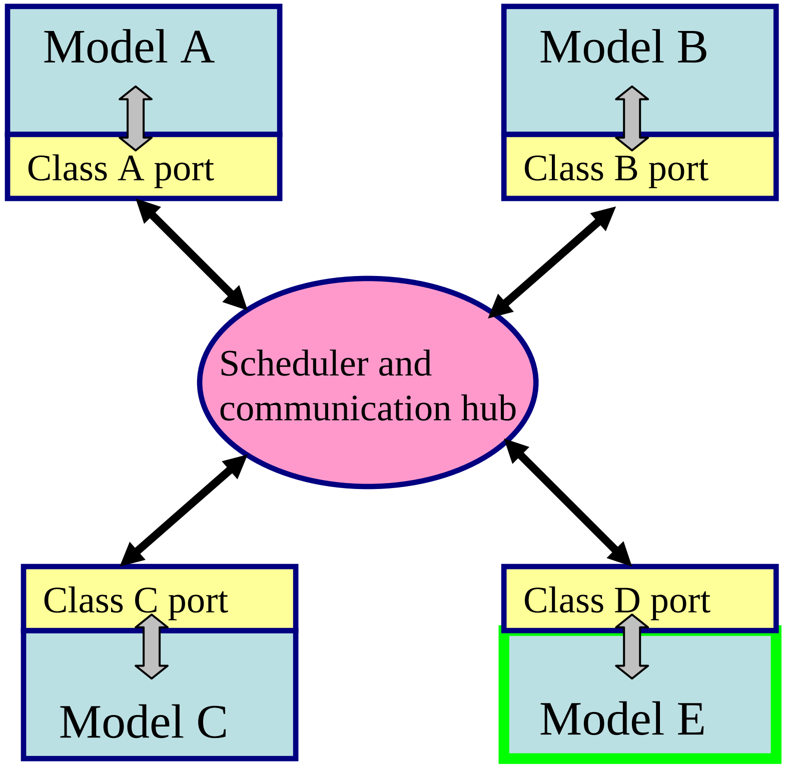
October 11, 2005.

Overview

- Framework Cartoon
- What is needed for SW forecasting
  - Patch-panel diagram
  - Candidate components
  - Summary of software properties
- The software engineering aspect
  - Key word is reuse. OOP, Components, Design Patterns, Frameworks etc. – definitions and history
- Compare and contrast the 3 relevant frameworks
  - Design
  - Status at the CCMC

A Framework ?





No Framework!

Elementary Framework!

Organizing superstructure for model set.

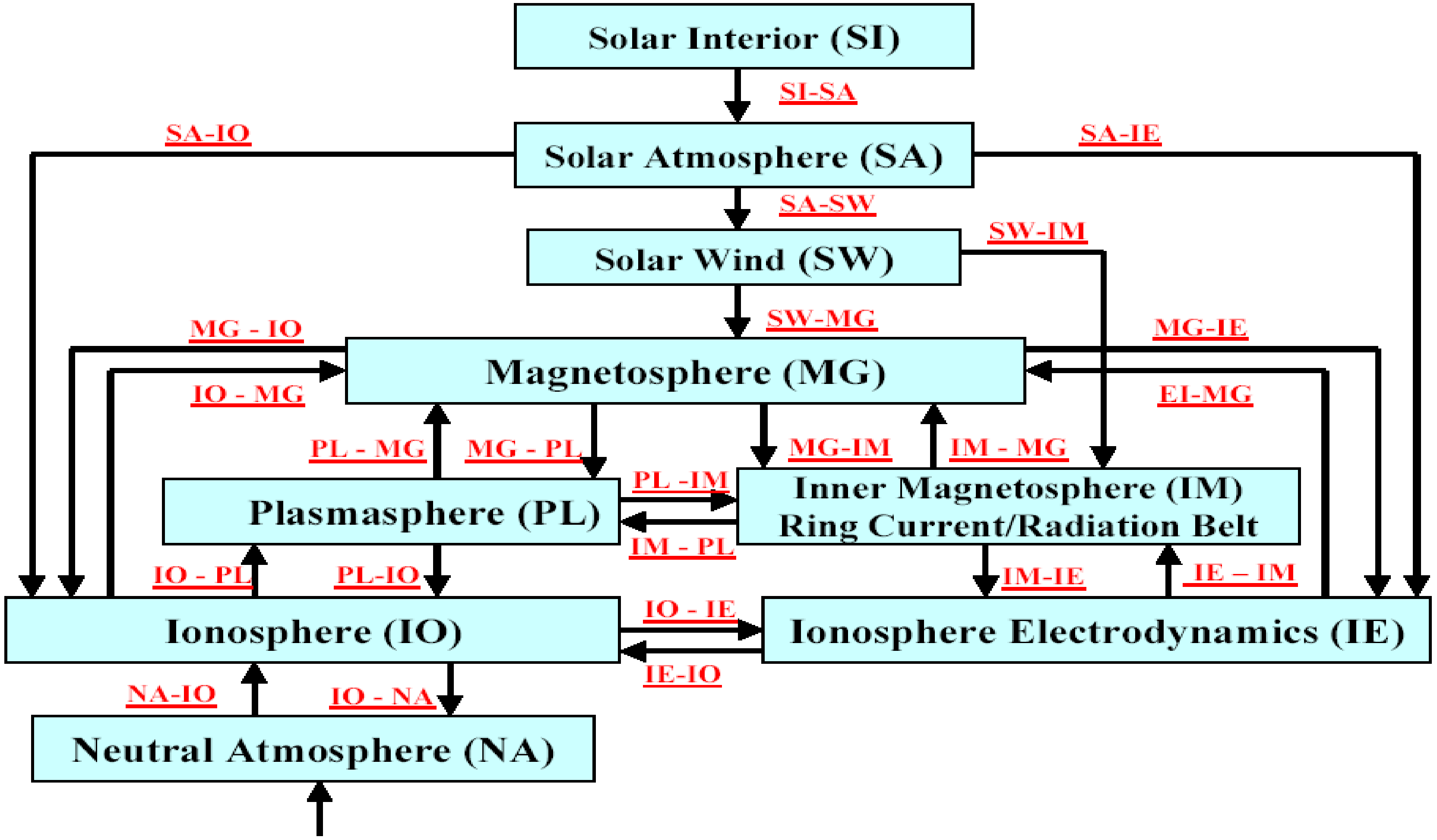
Theoretical Definitions of a Framework

"In software development, a **framework** is a defined support structure in which another software project can be organized and developed. A framework may include support programs, code libraries, a scripting language, or other software to help develop and *glue together* the different components of a software project.

The word 'framework' has become a buzzword due to recent continuous and unfettered use of the term for any generic type of libraries. Sometimes, the word **framework** is used in place of the synonymous term, library, for the purpose of impressing venture capitalists."

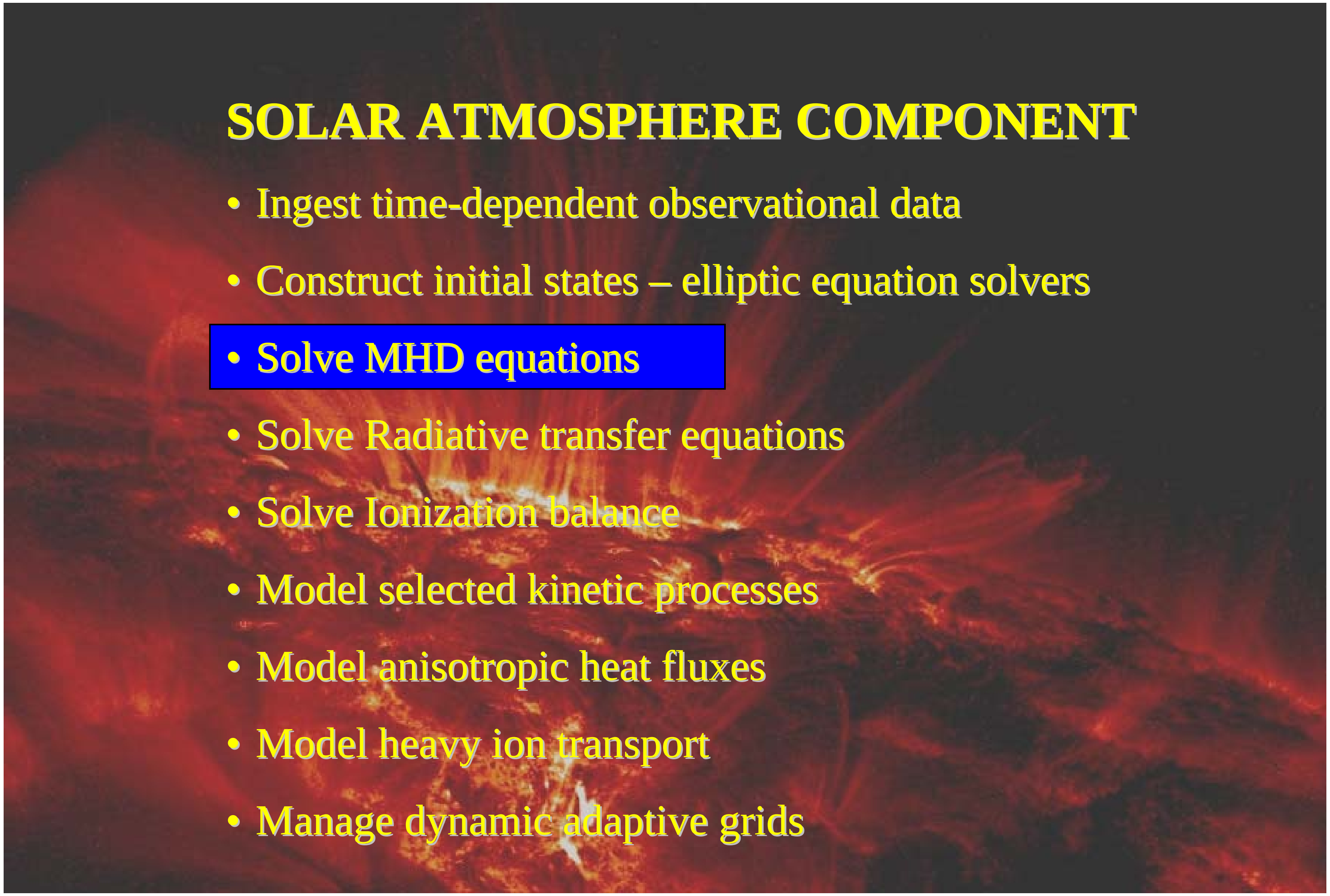
Wikipedia

What Forecasters Will Need



patch-panel architecture

What Forecasters Will Need



SOLAR ATMOSPHERE COMPONENT

- Ingest time-dependent observational data
- Construct initial states – elliptic equation solvers
- Solve MHD equations
- Solve Radiative transfer equations
- Solve Ionization balance
- Model selected kinetic processes
- Model anisotropic heat fluxes
- Model heavy ion transport
- Manage dynamic adaptive grids

Candidate Coronal MHD Models

List (guaranteed incomplete) of 3D MHD codes used in coronal modeling literature since 2002.

1. MAS-SAIC
2. SWMF component (Hi order Godunov)
3. ARMS (FCT)
4. CRUNCH3D (Dahlburg – Spectral)
5. Wu et al
6. Nordlund (Mellor et al)
7. Zeus 3D MHD (Ledvina et al)
8. Fan,Y. (Zeus-like 3D MHD?)
9. Kleinmann et al (3rd order CWENO)
10. Arber et al (Lare3d – Lagrangian remap + shock capturing - cartesian)
11. Ofman (NLRAT - modified Lax-Wendroff)
12. Torok and Kliem (p=0 approx. – modified Lax-Wendroff)
13. Podgornyi and Podgornyi
14. Kuwabara, Uchida, Cameron – Code ?
15. Ye et al or Feng et al ??(TVD Lax-Friedrich with MacCormack II ??)

Software Requirements

The ultimate 'suite' of space weather forecasting tools MUST achieve

- Accuracy
- Robustness
- Efficiency (both cpu and memory)

Software Requirements

- Portability
- Faster than real time
- Ease of component replacement
- Ease of component addition
- Ease of component development
- Flexible data assimilation support
- <u>Maximum software reuse</u>

Software Reuse Opportunities

- Inter panel (and intra-panel) ‘glue’ – coupling interfaces and execution scheduler
- Model state descriptors
- Grid management
- Data interpolation and transformation
- Equation solvers
- I/O
- Visualization support

CCMC

Software Requirements

- Reuse by standardizing those aspects of the overall software design.
- Additional standardizations relevant to CCMC
  - Datafile formats
  - Documentation
  - Feature extraction tools
  - Archival and database interfaces
- FRAMEWORK and COMPONENT programming are the favored approach to achieving this reuse – origins in OOP.
- Building on legacy codes, not from scratch.
- Tension between generality and efficiency

Theoretical Definitions of a Framework

"A framework is a set of prefabricated software building blocks that programmers can use, extend, or customize for specific computing solutions."

Taligent white paper

"A framework provides architectural guidance by partitioning the design into abstract classes and defining their responsibilities and collaborations. A developer customizes the framework to a particular application by subclassing and composing instances of framework classes."

NASA CT CAN

"Frameworks are reusable designs of all or part of a software system described by a set of abstract classes and the way instances of those classes collaborate."

Roberts and Johnson.

Frameworks at CCMC

- SWMF (*Gombosi et al*, Univ. of Mich.) – already playing with this!

  SC, IH, GM – BATS-R-US

  SP – Kota's SEP or FLAMPA

  IM – Rice Convection Model (RCM)

  RB – Rice RBM

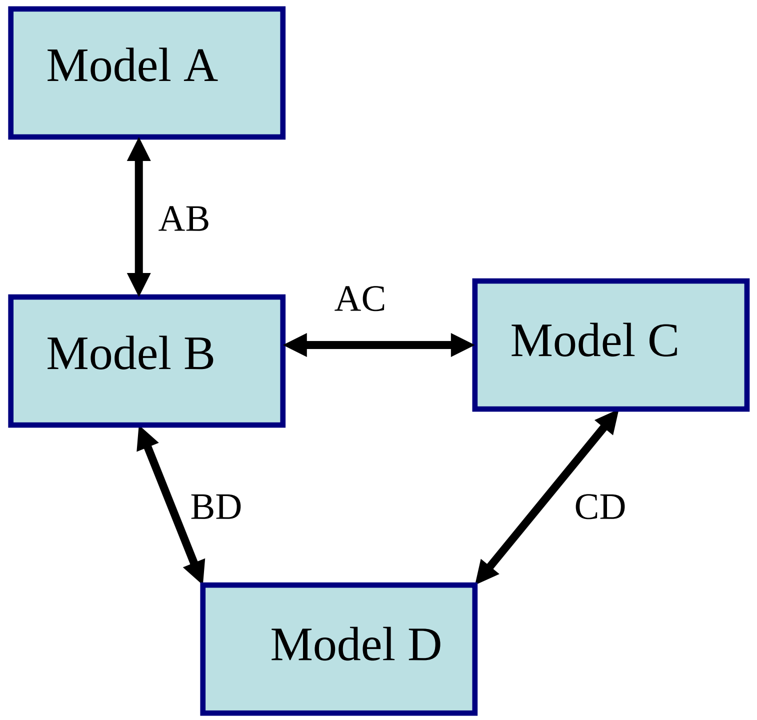
  IE – Ridley Ionosphere

  UA – Global Iono. Thermosp. Model (GITM)

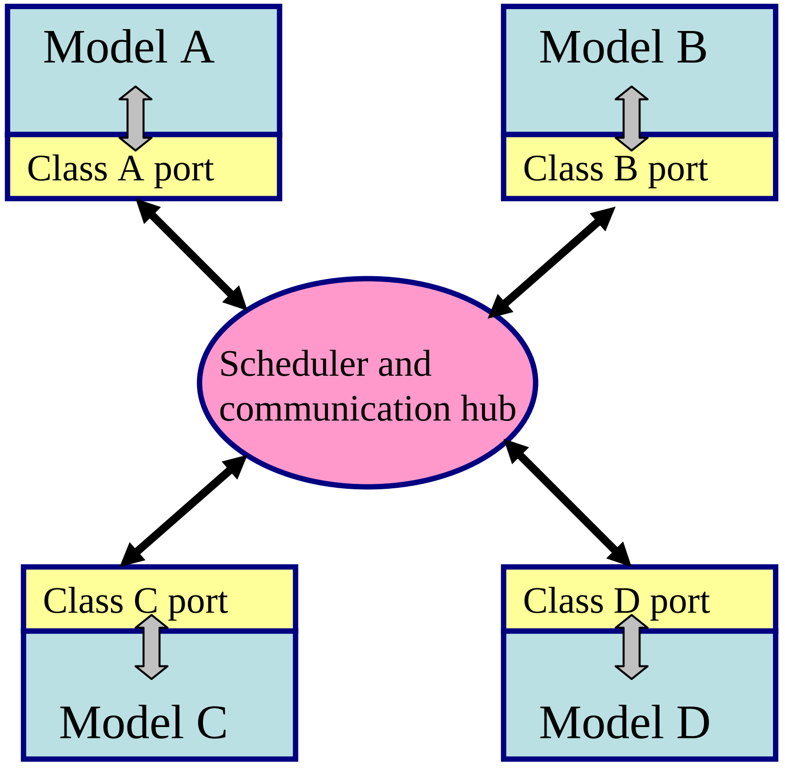
- CISM – MOU in place. Expect to begin experimenting with couplings later this year.

A successful framework is a critical need for the long-term role of the CCMC !

A Framework?

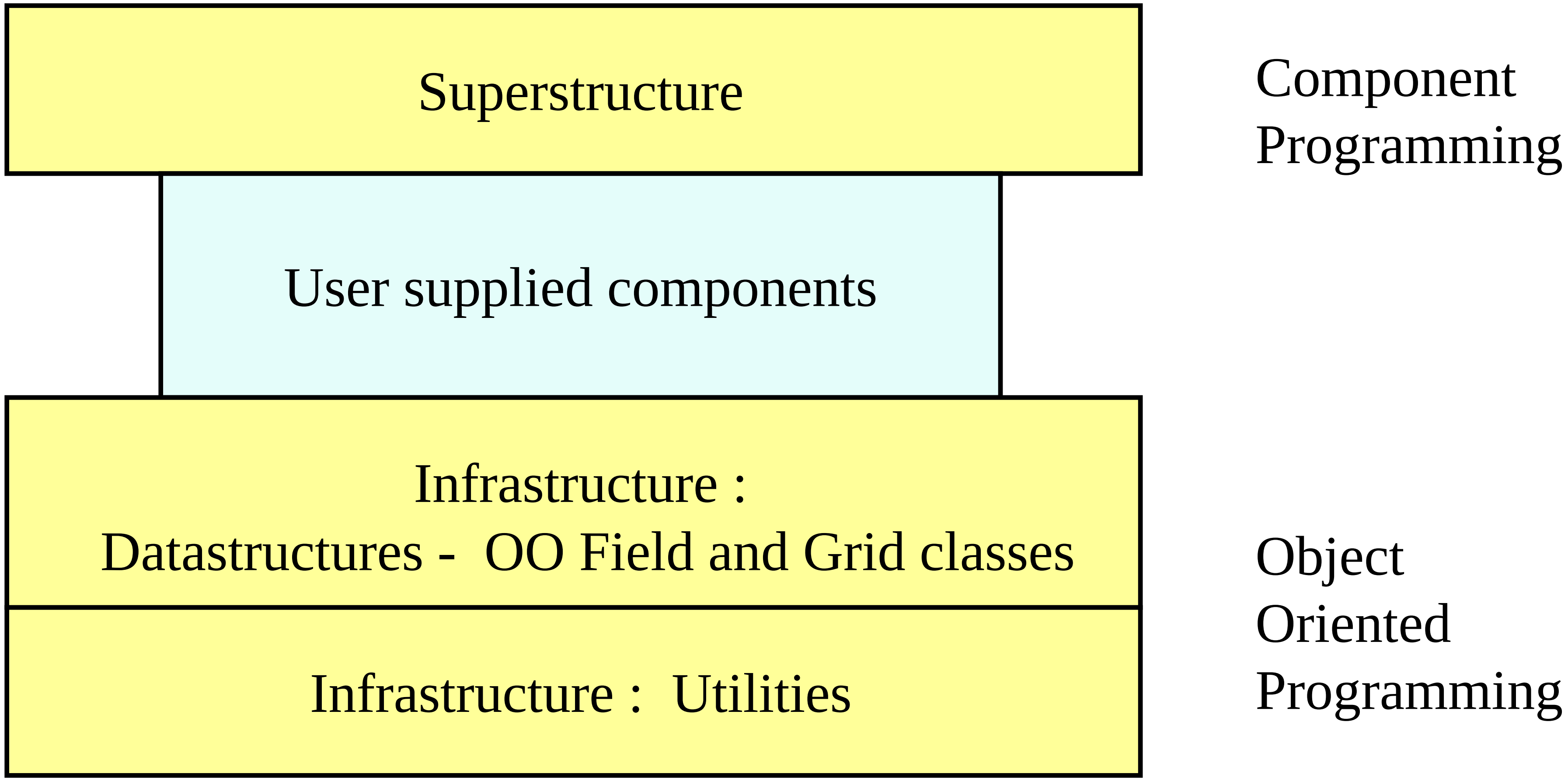


Ad hoc coupling – Not a Framework!



Elementary Framework!

Example – ESMF


Superstructure
User supplied components
Infrastructure :
Datastructures - OO Field and Grid classes
Infrastructure : Utilities
Component Programming
Object Oriented Programming

Example – ESMF Reorganise Original Code

For each model :

- Separate gridded code from coupler code
- Driver component is a third type of component
- For each component craft create, initialize, run and finalize sections (or methods)
- Each component must have import and export state arguments
- Data for model coupling passed through import/export arguments

Example – ESMF Rearrange Original

Sub modelA

⋮

mpi_send(X)

⋮

mpi_get(Y)

⋮

End modelA

modelA_create

modelA_init

modelA_run(import_state, export_state)

⋮

modelA_finalize

Gridded component

coupleAB_create

coupleAB_init

coupleAB_run(import_state, export_state)

⋮

coupleAB_finalize

Coupler components

⋮

Call ESMF_**Grid**Comp**Run**(**modelA**,import1,**export1**,…)

Call ESMF_**Cpl**Comp**Run**(**coupleAB**,**export1**,**import2**,…)

Call ESMF_**Grid**Comp**Run**(**modelB**,**import2**,export2,…)

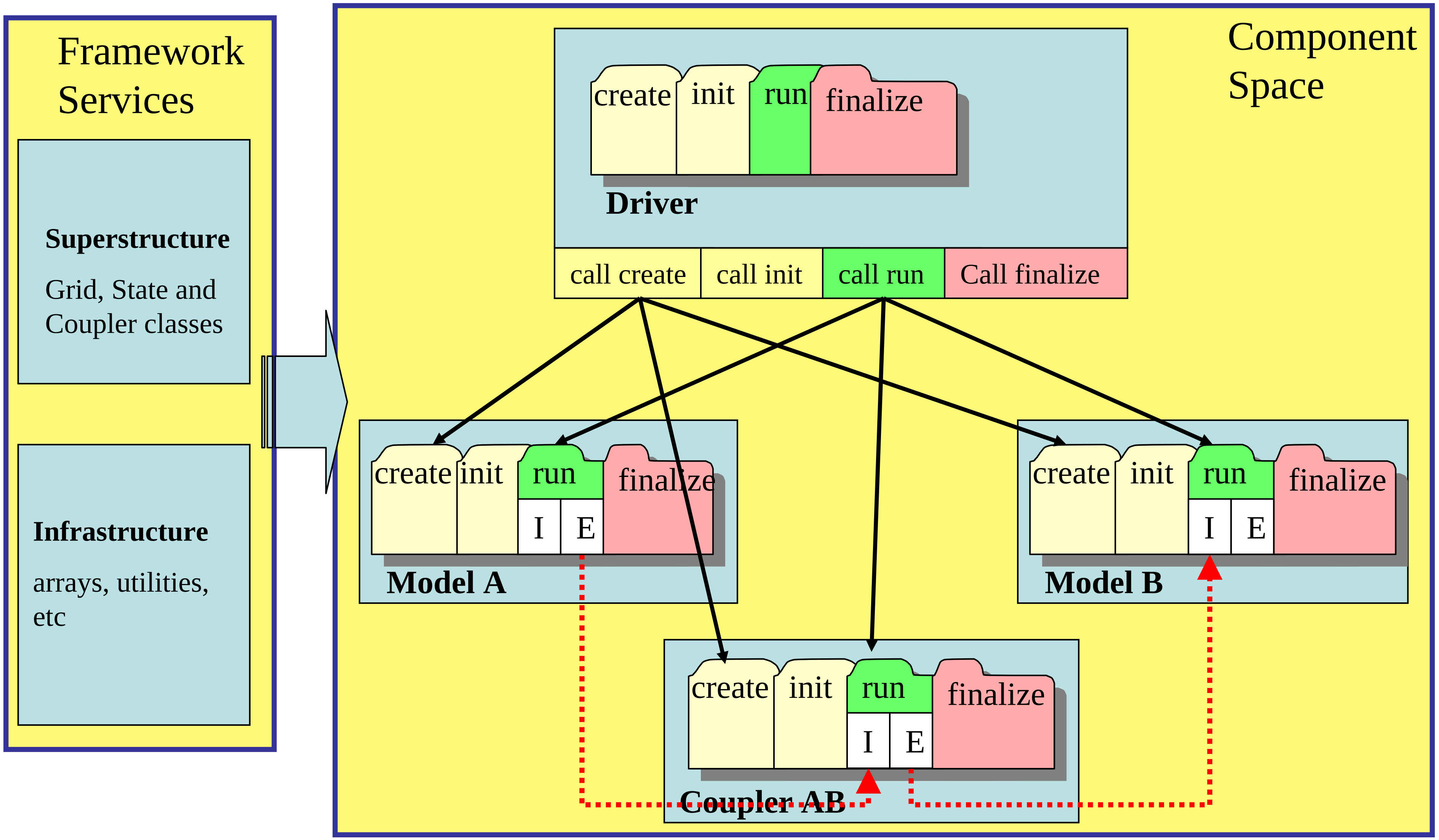
⋮

Driver Component

Example – ESMF

Create Phase

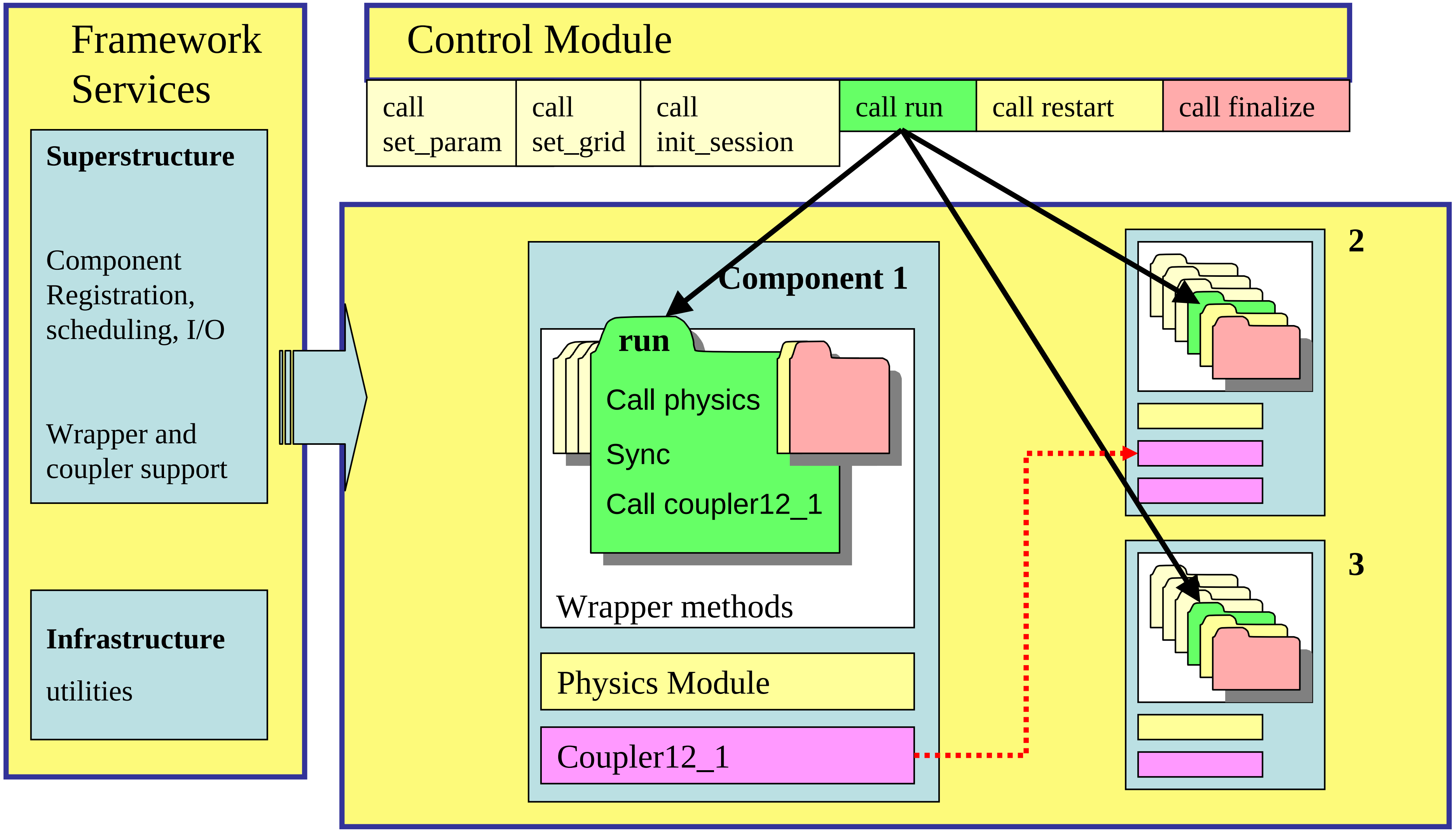
Run Phase



PM6

Execution Phase

Example - SWMF



Slide 19

PM6 What is actual mechanism of communication? Is it an MPI call embedded within the component source code, or an expression of what to do on entry and exit from the component?

Peter MacNeice, 10/3/2005

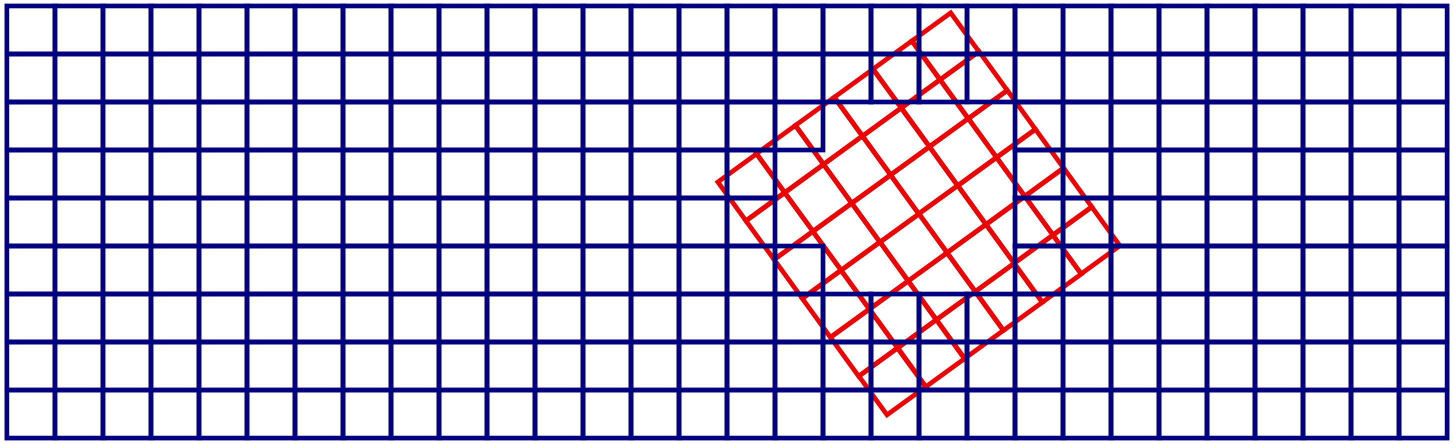
Compare and Contrast ESMF and SWMF

- Very similar in overall design - built into a single executable
- Principal architectural Differences
  - ESMF treats driver as a component
  - ESMF couplers are separate components
  - Component encapsulation - ESMF components have private states, only accessible through argument lists.
- Other differences
  - Support functionality - particularly types of grids supported
  - Language constraints

PM2

Example - CISM

- More loosely coupled - components may be separate asynchronously executing processes.



- Models 1 and 2 have overlapping grids
- Model 1 provides X to model 2, but 2 needs the interpolated form Y
- Model 2 returns Y to 1, but 1 needs it in original form to update X

Slide 21

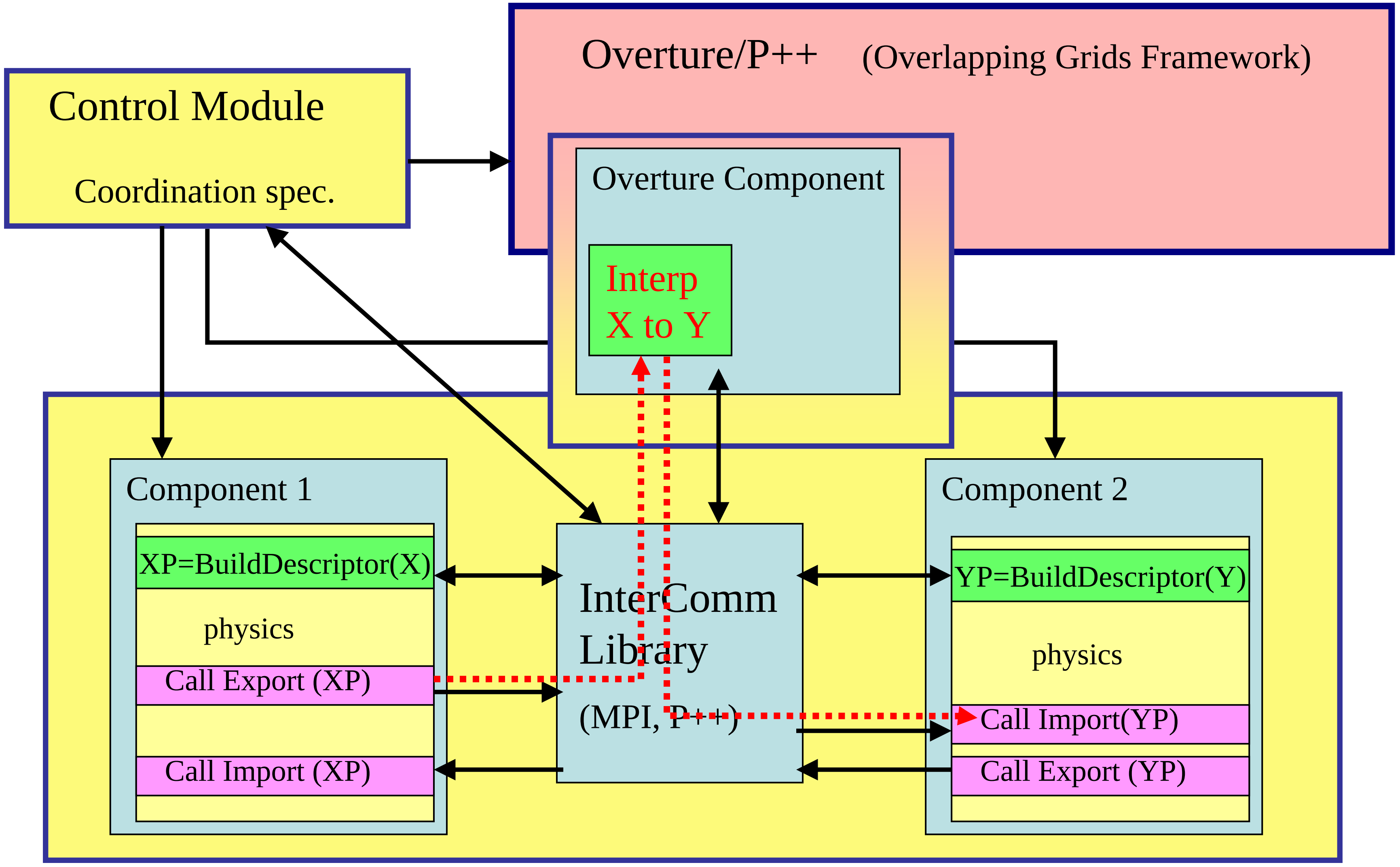
PM2 What is actual mechanism of communication? Is it an MPI call embedded within the component source code, or an expression of what to do on entry and exit from the component?
Peter MacNeice, 10/3/2005

Example - CISM

Composition Phase

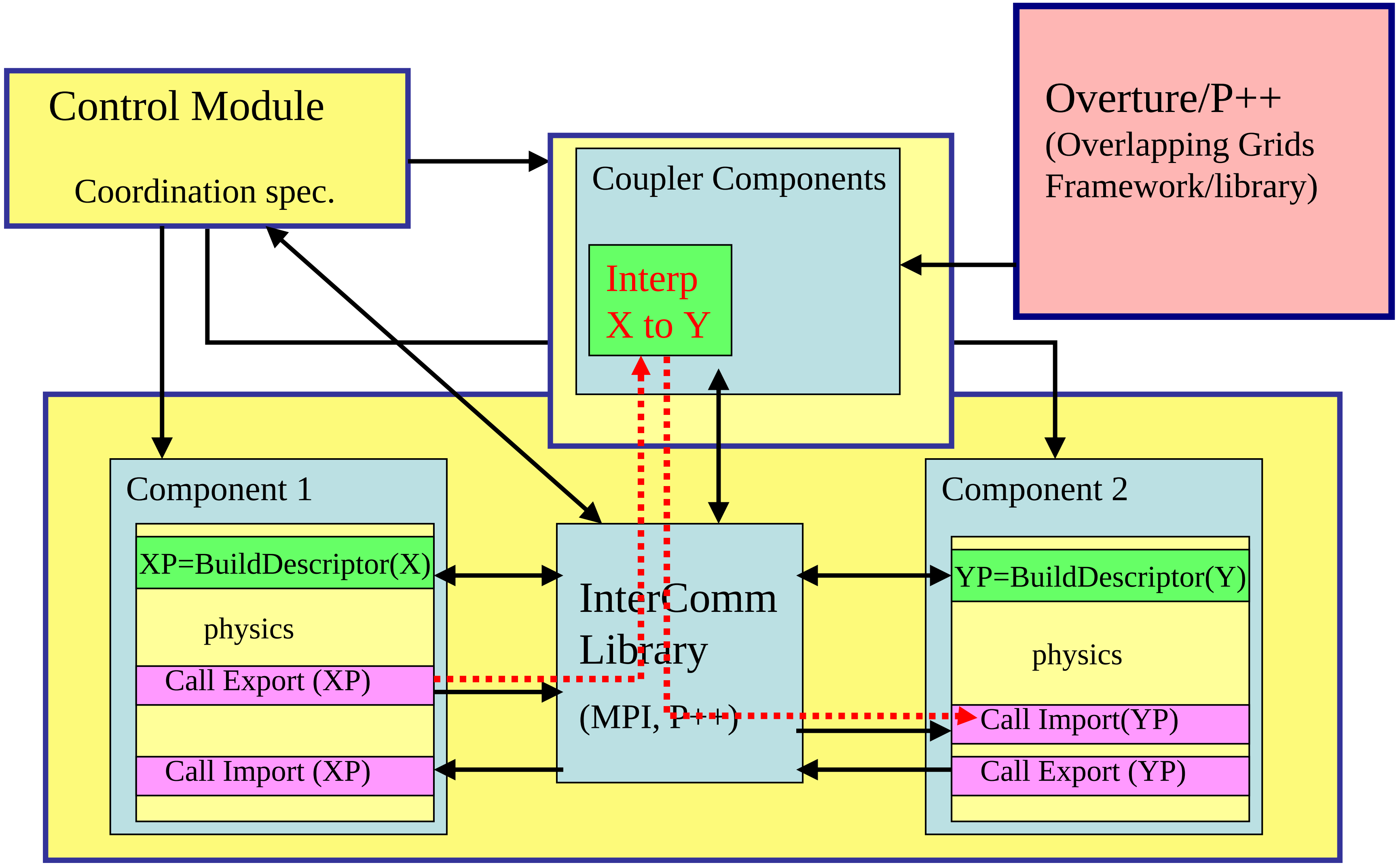
Initialization Phase

Execution Phase



PM4	What is actual mechanism of communication? Is it an MPI call embedded within the component source code, or an expression of what to do on entry and exit from the component? Peter MacNeice, 10/3/2005

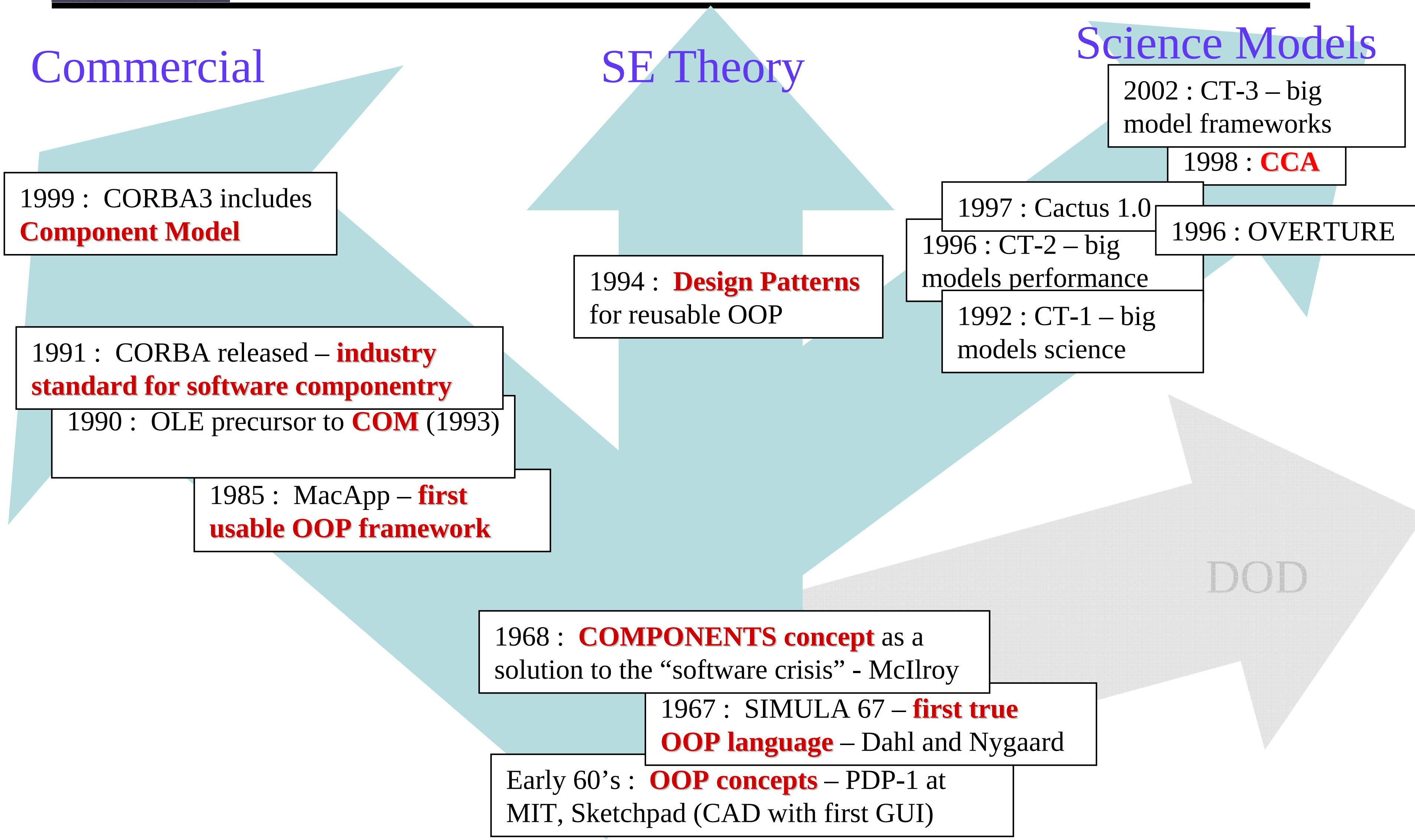
Example - CISM



Framework Domains

- The patch panel architecture is a ‘Domain Framework’.
- The domain is the family of space weather forecasting models – the ‘domain’ reflects the ways we will need to generalize the application to identify the software reuse opportunities of most value to us.
- Framework domains often overlap.
- The domain of MHD models using adaptively refined meshes(AMR) is relevant to 4 of the panels (Solar Interior, Solar atmosphere, Heliosphere, Magnetosphere). eg CHOMBO and OVERTURE frameworks.
- How can our framework make use of parts of the AMR frameworks?
- One answer is COMPONENT technology through use of a ‘component architecture’.

Historical Aside - 1



2002 : CT-3 – big model frameworks

1998 : **CCA**

1997 : Cactus 1.0

1996 : OVERTURE

1996 : CT-2 – big models performance

1992 : CT-1 – big models science

1999 : CORBA3 includes **Component Model**

1994 : **Design Patterns** for reusable OOP

1991 : CORBA released – **industry standard for software componentry**

1990 : OLE precursor to **COM** (1993)

1985 : MacApp – **first usable OOP framework**

1968 : **COMPONENTS concept** as a solution to the “software crisis” - McIlroy

1967 : SIMULA 67 – **first true OOP language** – Dahl and Nygaard

Early 60’s : **OOP concepts** – PDP-1 at MIT, Sketchpad (CAD with first GUI)

Components

Definition is fuzzy and has evolved.

- Past -- source code modules, as in a fortran library.
- Now -- developing software systems by assembling a set of larger components off-the-shelf(COTS) in a ‘plug and play’ manner. Electronics industry analogy.

Key generic properties of a COMPONENT:

- Information hiding – user only sees the external interface
- Context independence – means innards must be self-contained, independent of any other components
- Components cannot address each other directly – need a registry

Components vs Objects

- Total encapsulation – black box vs white box (reuse thru class inheritance)
- Granularity – typically components are larger than objects

CCMC

Component Models

COMPONENT MODELS - To work together components must share a common architecture for their interface, registry and composition rules.

Benefits – component use independent of original framework, and perhaps original language and platform.

"If one day CBSD really get established in a large scale, <u>it will bring fundamental changes in the way software is developed</u>. Through CBSD, the process of programming components gets separated from that of composing components to applications. The hope is that components can be plugged together as easily as Lego bricks without the need for writing source code. <u>Thus, even domain experts with rather few technical skills can assemble applications</u>."

Wikipedia again!

Common Component Architectures

- Component models well established in the commercial world (Corba, DCOM, Enterprise JavaBeans)
- CCA – A component architecture for high performance computing.
- Enables framework components to be accessed outside their original framework.
- Some ESMF components have been expressed as CCA components and composed into an application.
- CCA applications can be composed and executed using GUI.

THE END