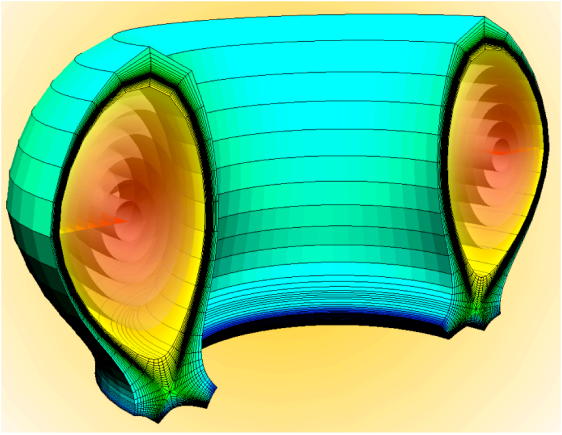
Framework efforts for the Fusion Simulation Project Planning

John R. Cary

Lead PI, FACETS

cary@txcorp.com



The FSP Framework Planning Work Product should be a convincing plan for developing a comprehensive predictive capability that can address the physics problems important to achieving fusion as an energy source.

- **Ability to predict requires incorporability and, ultimately, incorporation of the highest fidelity components**
- **Confidence in prediction comes from V&V**

The highest fidelity computations require

- Mastery of detail
  - Not just the zoo of plasma phenomena
  - But also that in conjunction with
    - Atomic physics
    - Controls
    - Materials
- Parallel computations
  - Message passing, distributed memory
  - Delegation to accelerators (e.g., GPUs) that require massive use of threads
- Flexibility
  - We already see new computational architectures on the horizon
  - We will have to incorporate new components as they become available

Framework (in the broadest sense): how will we build this?

- What methodologies will we use?
- How will we assure that the software fits together?
- How will we avoid the maintenance trap?
- How can we achieve sufficient abstraction to have minimal code while having sufficient specificity to have a useful product?

This is a broader definition that generally used by the Computer Science (CS) community.

What is a software framework?

Wikipedia: A software framework, in computer programming, is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality. Frameworks are similar to software libraries in that they are reusable abstractions of code wrapped in a well-defined API.

Software frameworks have these distinguishing features that separates them from libraries or normal user applications:

1. **inversion of control - Unlike libraries or normal user applications, in a framework the overall program's flow of control is not dictated by the caller, but by the framework.[1]**
2. **default behavior - A framework has a default behavior. This default behavior must actually be some useful behavior and not a series of no-ops.**
3. **extensibility - A framework can be extended by the user usually by selective overriding or specialized by user code providing specific functionality**
4. **non-modifiable framework code - The framework code, in general, is not allowed to be modified. Users can extended the framework, but not modify its code.**

What is a software framework?

http://www.codeproject.com/KB/architecture/WhatIsAFramework.aspx

- it makes it easier to work with complex technologies
- it ties together a bunch of discrete objects/components into something more useful
- it forces the team (or just me) to implement code in a way that promotes consistent coding, fewer bugs, and more flexible applications
- everyone can easily test and debug the code, even code that they didn't write

A Framework has

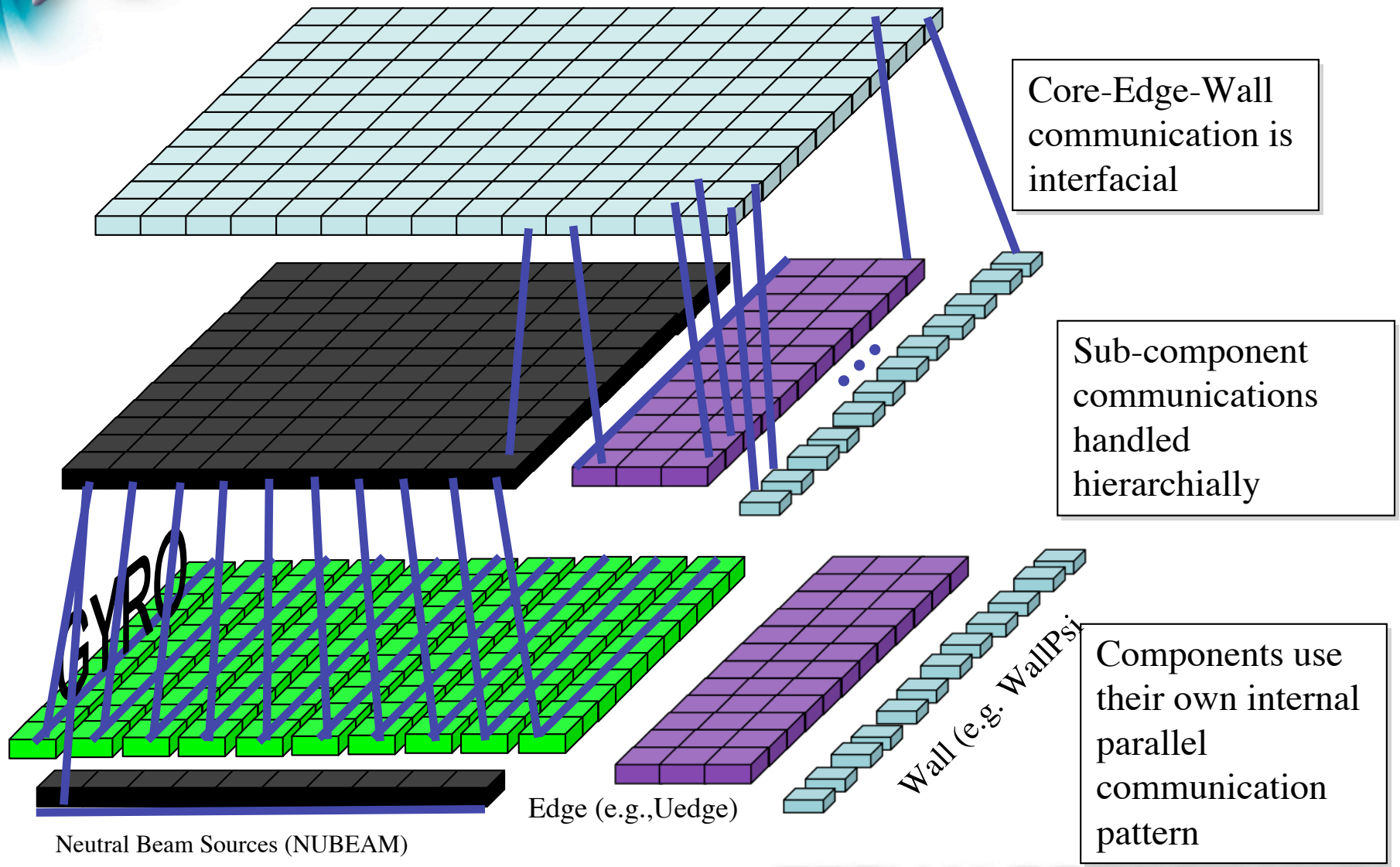
- wrappers. A wrapper:
  - o simplifies an interface to a technology
  - o reduces/eliminates repetitive tasks
  - o increases application flexibility through abstraction
  - o are often re-usable regardless of high level design considerations
- architectures. An architecture:
  - o manages a collection of discrete objects
  - o implements a set of specific design elements
- methodologies: A methodology:
  - o enforces the adherence to a consistent design approach
  - o decouples object dependencies
  - o are often re-usable regardless application requirements

Framework: reminiscent of the 9 blind men and the elephant

- "Little-f" framework = Infrastructure
  - Methodologies
  - Build systems
  - Workflow tools
  - Examples: CORBA, ESMF
- "Big-F" Framework = Superstructure
  - Code for bringing components into a combined application
  - Standards that allow components to be brought into a Framework
  - Examples: FACETS, CCSM, ESMF

Autotools (autoconf, automake, libtool) is a **Framework** for cross-platform code building, but it can be part of a **framework**

Goal: a Framework capable of running highest fidelity components



The framework must accept components of multiple languages

- http://en.wikiquote.org/wiki/Larry_Wall#.22Present_Continuous_-_Future_Perfect.22
- And C was good at something I like to call manipulexity, that is the manipulation of complex things. While shell was good at something else which I call whipuptitude, the aptitude for whipping things up.
- "We've got to start over from scratch" - Well, that's almost any academic language you find.
- "English phrases" - Well, that's Cobol. You know, cargo cult English.
- "Text processing doesn't matter much" - Fortran.
- "Simple languages produce simple solutions" - C.
- …
- "This is a very high level language, who cares about bits?" - The entire scope of fourth generation languages fell into this... problem.
- …
- "Let's make this easy for the computer" - Lisp.
- "Most programs are designed top-down" - Pascal.
- "Everything is a vector" - APL.
- "Everything is an object" - Smalltalk and its children. (whispered:) Ruby.
- "Everything is a hypothesis" - Prolog.
- "Everything is a function" - Haskell.
- "Programmers should never have been given free will" - Obviously, Python.

A Framework contains a workflow, it is not a workflow

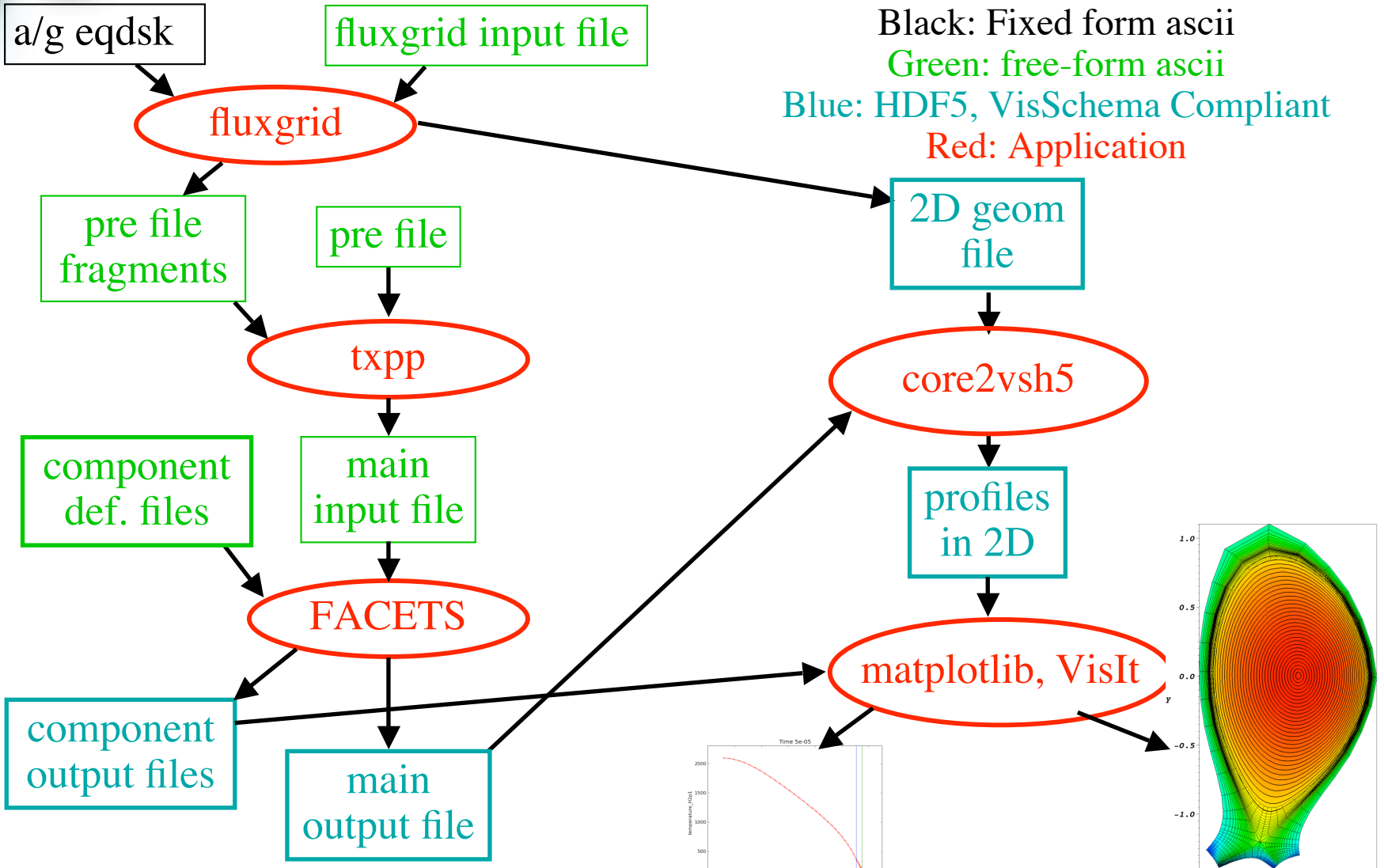
- **Workflow: a series of separable actions to get from problem definition to human interpretable data**
- **Framework: one or both of**
  - **Superstructure by which one couples together components (objects, modules, …)**
  - **Infrastructure on which one builds components**

(not often cleanly separable)

that allows components to be executed conditionally and composed abstractly.

FACETS workflow: buckets of bits from eqdsk to viz

Black: Fixed form ascii
Green: free-form ascii
Blue: HDF5, VisSchema Compliant
Red: Application



THE FSP Planning Proposal has the following items

- Assess past integrated modeling efforts
- Assess existing proto-FSPs
- Determine needed couplings
- Determine execution models
- Determine needed algorithms
- Design verification plan
- Determine software standards
- Assess existing components for use within a framework
- Technology and structure adoption
- Determine workflow needs, settle on workflow tools

Framework efforts broken down

- Management (1 unit, Cary)
  - Sets goals, oversight, decision framework
- Framework evaluation and design (2 units, Siegel, others at Tech-X)
- Assessment of user needs (1 unit, R Cohen)
  - Interviews, reports
  - Must assess all stakeholders (ASCR, users, …
- Software engineering (1 unit, Van Straalen): evaluate tools

Management sets overall decision framework

- What is allowed or not?
- Provide vision
- Set agendas (includes goals)
- Who needs to be involved in decisions?
- Monitor progress
- Present results

Will assess user needs (R Cohen)

- Work with Physics needs (Kritz) to prioritize physics needs
  - What is the minimal tool that the community will find useful?
  - What is the timeline for tool inclusion
- Determine user needs
  - How do they work? (Involves where as well)
  - What do they need? (physics, problem setup tools, postprocessing, visualization)
  - How does this vary depending on the level of user?
    - User
    - Builder (expert)
    - Knows 1 system, all OS's

Framework 1: evaluate previous frameworks, set goals for framework, ultimately design

- Integrated modeling has a long history
  - BALDUR, pTRANSP
- Fusion proto-FSPs: CPES, FACETS, SWIM
  - Each currently publishing papers on their frameworks
- Non fusion frameworks
- Determine needed couplings
- Algorithmic needs

Framework 2: assess components, efforts to make

- Work with X. Tang
- Components from legacy code
  - https://www.facetsproject.org/facets/wiki/ComponentsFromLegacyCode
  - What components do we need?
  - How long to make a component from legacy code?
  -
- New components
  - How long to create? (Concept to first physics)

Software engineering: tools and methodologies

- Technology and structure adoption
  - Build (autotools? Cmake?)
  - Output (HDF5, schema)
  - Workflow
  - Viz
- Determine software standards
  - Formatting
  - “No calls to exit!”
- Verification plan (work with Greenwald)
  - Data access methodology

Deliverable of the Framework Effort

- What to build
- How to build
- What to build first

  No funds yet in Boulder, but
- Discussion with team subgroup leaders (Cary, Siegel, Cohen, Van Straalen)
- Request input on their tasks
- Progress reports, reassess, move ahead

Initial tasks

- Review with team. Initial meeting tonight.
- Begin assessment of user needs
  - What early deliverables could make the project a success in the eyes of modelers?
- Define what we want from other teams
  - Multiple candidates for near-term capabilities from science drivers
  - What is the status of components?
- At the same time, we can move ahead with cross cutting assessments
  - What are other frameworks doing?
  - What is the status of components with regard to inclusion in framework