

## Breakout Session Fusion

~ Internal and External, Yin and Yang, Holistic Unity... ☺  
(The “un-scientific” ones...)

What are the barriers to reuse?

What are the broad categories of things we should be focusing on?  
→ What to standardize?

How do we move forward?

### Barriers to Reuse

1. Incompatible Data Models!
2. Incompatible Control Models
  - a. Every tool/framework wants to be main(), imply its own control structure.
3. Monolithic Infrastructure
4. Limited Interface Exposure
  - a. Documentation...?
5. Availability / incentives for code sharing
  - a. No reward system for open sourcing
6. Performance
7. Unique Requirements
  - a. Specialized needs preclude “full” reuse of existing tools/technology
8. Parallelism
  - a. Startup
  - b. Parallel Data Models
  - c. Orchestration of Serial Components
9. Programming Languages / Interoperability
10. Licensing...
  - a. Recent progress in DOE...
11. Tight Coupling to GUI
12. Lack of a Public Forum / Community
  - a. No Code Repository
13. Combinatoric Variations on individual functional blocks
  - a. Grids, data types, parallelism
14. Lack of Orthogonal Decompositions
15. Incompatibility of Input and Output (Display) Devices
16. Data Transport Mechanisms
17. Testing, verification and quality of code
  - a. Unit and regression testing...
18. Portability

## Strategies for Alleviating these Barriers:

1. [Polly](Data Models) Adopt a Hub/Spoke Scheme for Data Model Interoperability
  - a. Utilize Generalized Adaptor/Accessor Wrappers to Translate
2. [Steve](Control Models) Target multiple “F”-frameworks with a disparate variety of control models (how many will it take?)
  - a. “F”-frameworks imply a specific execution model
  - b. “F”-frameworks are a means for organizing/integrating independent tools
3. [Andrew](Monolithic Infrastructure) Encourage fine-grained reuse
4. [Sam](Interface Exposure)
  - a. Identify interfaces to be standardized
  - b. Create “committees” to develop and agree on those standards
    - i. The simplest and most productive committee is N=1. ☺
  - c. Document and publish standard viz interfaces
  - d. Encourage use of those standards
5. [Mike](Sharing)
  - a. Big Stick / Money Approach ~ Require Office of Science project deliverables be compatible with the blessed F/f-framework(s)...
    - i. Programmatic support, encouragement or requirements for sharing viz tools / interfaces...
    - ii. Support for creating “best” solutions, attractive tools and interfaces for community adoption...
    - iii. Yet need to avoid constraining revolutionary developments...!
  - b. Culture a publication venue for visualization tools
6. [Mark](Performance) Maintain/improve the performance of existing technology
  - a. Design components and interfaces with performance in mind
  - b. F/f-framework cannot *preclude* performance
  - c. Quantify performance impacts of standardized interfaces
  - d. Provide performance monitoring and analysis capabilities
    - i. Develop performance model interfaces to viz components
7. [Mark](Unique Requirements)
  - a. Big problem will not go away, we can only minimize & manage it...
    - i. Reduce the granularity of components to improve degree of reuse
    - ii. Design hierarchical/extensible interfaces for specialized capabilities
    - iii. Rapid prototyping / composition
  - b. Open source helps, too... (Steal & “customize” existing code)
8. [Jeeem]([doc](#))(Parallelism) Needs to be implicit in the whole process.
  - a. Every interface must address parallelism (or justify why not)
  - b. Thread safety is important (at least identify safe or not!)
  - c. Utilize fine-grained serial components that can be composed in parallel
    - i. Not always possible... Not always efficient enough...

- d. Port existing tools to parallel / multi-threaded environments
  - i. Ensure that F/f-ramework compositional mode supports parallelism
  - ii. Interfaces are designed with parallelism in mind
  - iii. Categorize the parallelism constraints of components and interfaces
- 9. [Andrew](Language Interoperability)
  - a. Two Options: Pick ONE Language, or Use something like SIDL/BABEL
    - i. True language interoperability limits performance
      - 1. Restricts sharing of complex constructs across languages
    - ii. Rely on “experts” to help save us!?
  - b. Can we simply pick a common subset of languages (C/C++ and python) for our purposes, and use SIDL/BABEL for integration with applications...?
    - i. We will make recommendations for languages and interoperability.
    - ii. We will decide on generic or language-specific interfaces
- 10. [Wes](Licensing) Promote DOE Open Source Licensing!
  - a. Form licensing requirements for component repository
    - i. Avoid infecting / poisoning code with “evil” licenses...
  - b. DOE cannot enforce a single all-encompassing license
    - i. Must support a range of possibilities...
    - ii. A given component *can* be multiply licensed (yuk)...
    - iii. Need for DOE-wide solution to expedite integrated software release?
  - c. Decide on licensing for common F/f-ramework code
- 11. [Polly](Coupling to GUI)
  - a. “Good” design decouples front-end GUI from back-end components.
    - i. Event-based, etc.
  - b. Develop or adopt generalized UI descriptions and intermediate protocol/meta-language (e.g. XML) for interfacing to front-end GUIs
    - i. Web-based
    - ii. Wimp
    - iii. VR
  - c. Remain “agnostic” – don’t pick a single GUI environment...
- 12. [Mike](Public Community)
  - a. Create a viz interface and component web site and repository.
    - i. Easy to set up, harder to get approval for postings...
    - ii. Create a SourceForge-based community site?
    - iii. Promote continued viz frameworks workshops like this one! ☺
- 13. [Steve](Combinatoric Variations)
  - a. Design interfaces/components that handle arbitrary grids, data types, etc...
  - b. Foster creation of flexible viz interfaces that accommodate various combinations, e.g. hub & spoke.
- 14. [Mark](Orthogonal Decompositions)
  - a. Develop set of “best practices” for designing viz tools
- 15. [Mike](Devices)

- a. Develop or adopt API for input device events
    - i. E.g. SNL's "Dors" system for multiplexing to "virtual displays"...
  - b. Is Chromium the answer for display devices...?
    - i. Doesn't handle every scenario... (A la DMX)
  - c. Develop or adopt multi-pipe API for displays
  - d. "Raw hand-to-hand combat with display technology"
    - i. Would seem an easy target for cooperation...?
    - ii. Have a workshop on multi-display technology
16. [Wes](Data Transport Mechanisms) Yes. ☺
- a. Need the "ftp" equivalent for moving images, pixels, etc...
  - b. Create serialization interfaces/capabilities for viz-specific objects/data...
    - i. Wire protocols...
  - c. Adopt other community standards where appropriate (Globus?)
17. [Nagiza](Testing) Require test suites for code repository approval
18. [Polly](Portability) Develop "best practices"
- a. Consider portability in language recommendations and interface design...

Definitions:

- f-framework == "community interoperability strategy"?
- F-framework == "framework" ☺

<BREAK>