

# **Comparison of Frameworks for Next Generation Multiphase Flow Solver, MFIX: A Group Decision-Making Exercise**

A. Gel, (Aeolus Research, Inc.)

S. Pannala, (Oak Ridge National Laboratory)

M. Syamlal and T. J. O'Brien (National Energy Technology Laboratory)

E. Gel (Arizona State University)

# Outline

---

- Motivation
- Problem statement
- Previous work
- Methodology
  - User and development team survey
  - Multi-criteria decision making
- Findings and conclusions

# Motivation

---

- MFIX (Multiphase Flow with Interphase eXchanges)
  - Developed over the last 20 years at NETL and ORNL
  - Distributed as open source ([www.mfix.org](http://www.mfix.org))
- Since inception of MFIX, many advances have occurred in computational science
- Incorporating these developments and public domain tools offer great potential for robust development of next generation problem solving environment for multiphase flows

## Motivation (cont'd)

---

- Some of the features sought for the next generation Problem Solving Environment (PSE):
  - scripting based interface for numerics & physics
  - components-based design to rapid reuse
  - efficient use of distributed-memory architectures
  - open-source licensing
  - flexible software development environment

# Problem Statement

---

● Multiple alternatives => various number of open source frameworks and PSEs that need to be categorized based on:

- Level of abstraction (features & capabilities)
- Diverse set of user needs
- Performance

⇒ Selection among various packages involves consideration of multiple criteria

# Previous Work

---

- Very limited systematic or quantitative classification of frameworks / PSEs published in open literature
- Application scientists' point-of-view usually not reflected
- No effective guidelines for application scientist who is in the process of adapting a new open source framework or PSE.

# Methodology

---

1. Create User Requirements Document (URD) and Software Requirements Document (SRD)
2. Explore and qualitatively evaluate a set of alternative frameworks and PSEs based on the available documentation, examples, etc.
3. Rate each package w.r.t. features listed in SRD
4. Conduct user survey to determine the relative importance of various features listed in the SRD
5. Compile the results of user survey and package ratings for determining the “optimal solution” (still work in progress)

# Frameworks and PSEs Evaluated

---

- [OpenFOAM](#) ([OpenCFD](#) from UK) Open-source
- [Trilinos](#) ([CCM](#) at SNL) Open-source
- [SAMRAI](#) ([CASC](#) at LLNL) Open-source (noncommercial use)
- [OVERTURE](#) ([CASC](#) at LLNL) Open-source (noncommercial use)
- [AMROC](#) ([CACR](#) at Caltech) Open-source
- [PETSc](#) ([MCS](#) at ANL) Open-source
- [ROCCOM](#) ([CSAR](#) at UIUC) Only DOE users

# Levels of Abstraction in Mathematical Software

---

PETSc v.2  $\Rightarrow$

1. Application-specific interface
  - Programmer manipulates objects associated with the application
2. High-level mathematics interface
  - Programmer manipulates math objects
    - Weak forms, boundary conditions, meshes
3. Algorithmic and discrete math interface
  - Programmer manipulates math objects
    - Sparse matrices, nonlinear equations
  - Programmer manipulates algorithmic objects
    - Solvers
4. Low-level computational kernels
  - BLAS-type operations
  - FFT

# Levels of Abstraction in Mathematical Software

---

Trilinos ⇒

1. Application-specific interface
  - Programmer manipulates objects associated with the application
2. High-level mathematics interface
  - Programmer manipulates math objects
    - Weak forms, boundary conditions, meshes
3. Algorithmic and discrete math interface
  - Programmer manipulates math objects
    - Sparse matrices, nonlinear equations
  - Programmer manipulates algorithmic objects
    - Solvers
4. Low-level computational kernels
  - BLAS-type operations
  - FFT

# Levels of Abstraction in Mathematical Software

---

OpenFOAM  $\Rightarrow$

1. Application-specific interface
  - Programmer manipulates objects associated with the application
2. High-level mathematics interface
  - Programmer manipulates math objects
    - Weak forms, boundary conditions, meshes
3. Algorithmic and discrete math interface
  - Programmer manipulates math objects
    - Sparse matrices, nonlinear equations
  - Programmer manipulates algorithmic objects
    - Solvers
4. Low-level computational kernels
  - BLAS-type operations
  - FFT

# Levels of Abstraction in Mathematical Software

---

SAMRAI  $\Rightarrow$

1. Application-specific interface
  - Programmer manipulates objects associated with the application
2. High-level mathematics interface
  - Programmer manipulates math objects
    - Weak forms, boundary conditions, meshes
3. Algorithmic and discrete math interface
  - Programmer manipulates math objects
    - Sparse matrices, nonlinear equations
  - Programmer manipulates algorithmic objects
    - Solvers
4. Low-level computational kernels
  - BLAS-type operations
  - FFT

# Levels of Abstraction in Mathematical Software

---

OverBlown  $\Rightarrow$

1. Application-specific interface
  - Programmer manipulates objects associated with the application
2. High-level mathematics interface
  - Programmer manipulates math objects
    - Weak forms, boundary conditions, meshes

Overture  $\Rightarrow$

3. Algorithmic and discrete math interface
  - Programmer manipulates math objects
    - Sparse matrices, nonlinear equations
  - Programmer manipulates algorithmic objects
    - Solvers
4. Low-level computational kernels
  - BLAS-type operations
  - FFT

# Levels of Abstraction in Mathematical Software

---

AMROC  $\Rightarrow$

1. Application-specific interface
  - Programmer manipulates objects associated with the application
2. High-level mathematics interface
  - Programmer manipulates math objects
    - Weak forms, boundary conditions, meshes
3. Algorithmic and discrete math interface
  - Programmer manipulates math objects
    - Sparse matrices, nonlinear equations
  - Programmer manipulates algorithmic objects
    - Solvers
4. Low-level computational kernels
  - BLAS-type operations
  - FFT

# Levels of Abstraction in Mathematical Software

---

ROCCOM  $\Rightarrow$

1. Application-specific interface
  - Programmer manipulates objects associated with the application
2. High-level mathematics interface
  - Programmer manipulates math objects
    - Weak forms, boundary conditions, meshes
3. Algorithmic and discrete math interface
  - Programmer manipulates math objects
    - Sparse matrices, nonlinear equations
  - Programmer manipulates algorithmic objects
    - Solvers
4. Low-level computational kernels
  - BLAS-type operations
  - FFT

# Evaluation and Rating of Alternatives

---

- Quantitative as well as qualitative evaluation methodology to determine the “best” alternative
- *Procedure:*
  - Reviewed the documentation
  - Installed the software
  - Rated the available features based on the convention developed (i.e., 1, 0.5, or 0)
  - Sum up the ratings
- For objective evaluation, need more than one person’s ratings, but this requires significant time investment...

## Questions to be addressed :

---

1. **What percent of the desired features are available?**
  - Feature Ratings compared with ideal case software
2. What is the relative importance of each desired feature in the SRD for Next Generation MFIX ?
  - Conduct user survey among MFIX users
3. Weighted ratings: Based on the importance level of desired features, which package seems to offer best set of features ?
  - Multi-criteria decision making

# Evaluations : Phase 1

---

- A worksheet based on the SRD was generated by the core MFIX team to rate different packages.
- SRD worksheet was compiled under 13 top level feature categories:
  1. Geometry
  2. Meshing
  3. Physics Representation
  4. Numerical Solution Scheme
  5. Software Development Environment
  6. Software Maintenance Environment
  7. Testing and Verification
  8. Documentation
  9. Target Hardware and OS
  10. Code Execution
  11. Output Data
  12. Post-Processing and Visualization
  13. Software Distribution Method

# Evaluations : Phase 1 (cont'd)

---

- Determined sublevel features for each category
- Each sublevel feature was rated as follows:
  - = 1 if the sublevel feature is already available,
  - = 0.5 if the sublevel feature is partially available,
  - = 0 if the sublevel feature is not available.

	Frameworks	
E.g.,	# 1	# 2
1. Geometry .....		
2. Meshing		
2.1 .....		
2.2 Mesh Type		
2.2.1 Cartesian Mesh	0.0	1.0
2.2.2 Unstructured Mesh	1.0	0.0
2.2.3 Hybrid Mesh	0.5	

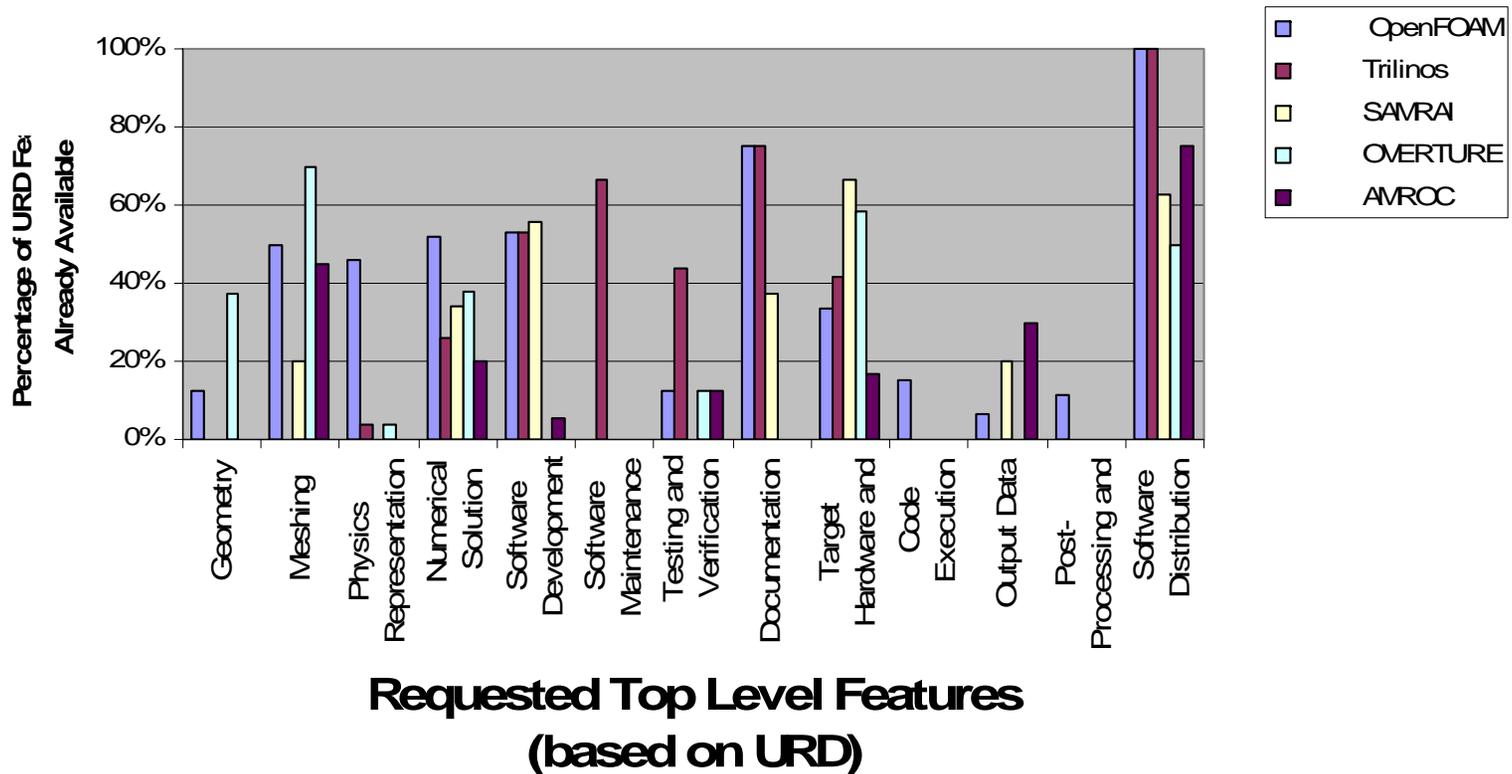
## Evaluations : Phase 1 (cont'd)

---

- Also, a separate risk assessment column was added for each sublevel feature using a similar convention, i.e.,
  - = 1 if there is high risk in acquiring the capability
  - = 0.5 if there is medium risk (partially know how to do it)
- Risk assessment was only applied if the desired feature didn't exist or partially existed
- Ratings presented here were conducted by only one person who evaluated all packages due to limited resources
  - feedback from more users preferable, involves significant time investment to get familiar with the package

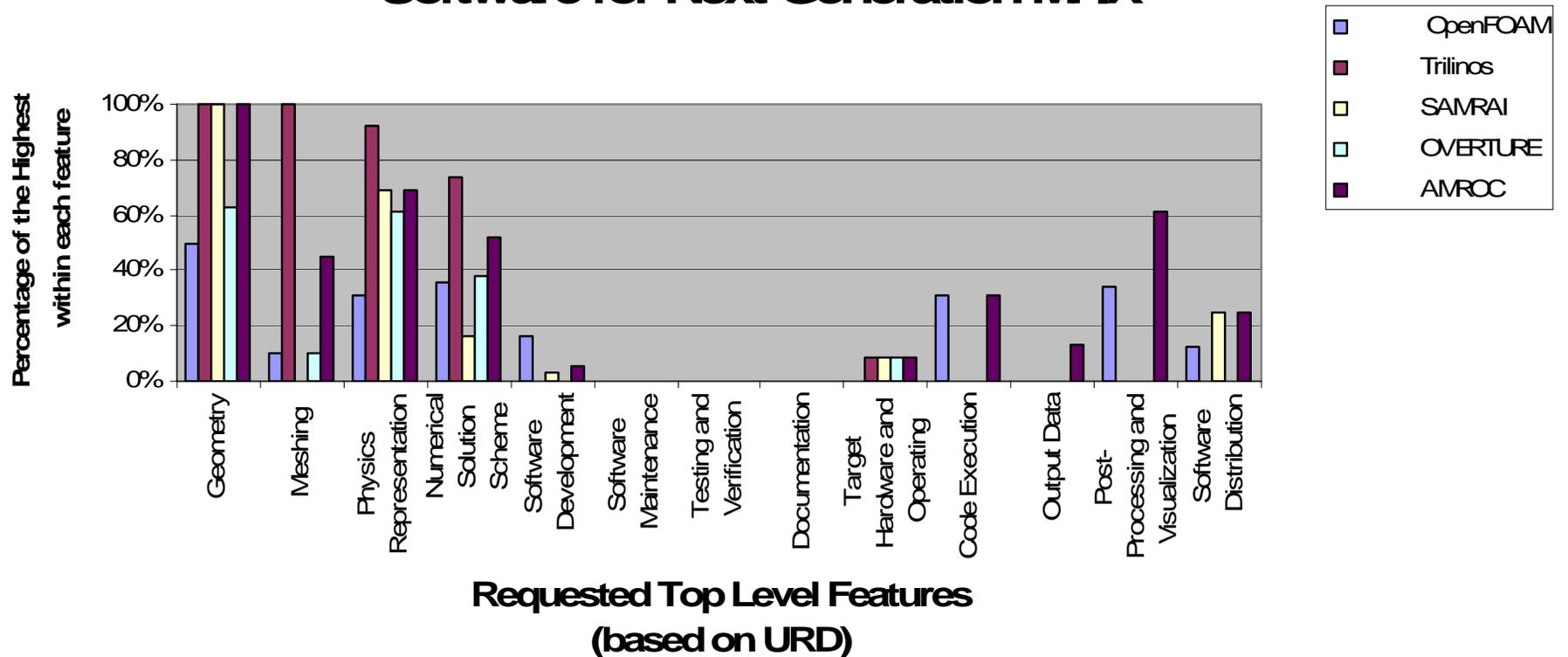
# Evaluations : Phase 1 (cont'd)

## Feature Rating Comparison of Available Open Source Software for Next Generation MFX



# Evaluations : Phase 1 (cont'd)

## Risk Assessment Comparison of Available Open Source Software for Next Generation MFX



## Questions to be addressed :

---

1. What percent of the desired features are available?
  - Feature Ratings compared with ideal case software
2. What is the relative importance of each desired feature (i.e., criterion) in the SRD for Next Generation MFIX ?
  - Conduct user survey among MFIX users
3. Weighted ratings: Based on the importance level of desired features, which package seems to offer best set of features ?
  - Multi-criteria decision making

# Multi-Criteria Decision Making

---

- The alternative to be chosen should maximize a “composite” of the objectives (i.e., criteria)
  - A weighted average of objectives commonly used in decision making
  - The weights associated with each objective reflects the decision maker’s priorities
    - Not straightforward to determine these weights
    - Need to use particular techniques to interview users
  - Furthermore, the choice should be acceptable for all users/decision makers (i.e., group decision making)

## Evaluations : Phase 2

---

- Used an online survey to get user feedback on the relative importance of desired features for next generation MFIX
  - Regrouped features to form 4 criteria
  - Users were asked to make pairwise comparisons on the relative importance
    - Analytical Hierarchy Process –commonly used technique for multi-criteria decision making

# Analytical Hierarchy Process (AHP)

---

- AHP is a systematic method of determining user preferences used commonly in multiple criteria decision making
  - Users are asked to make pairwise comparisons in terms of relative importance, which are then used to determine normalized weights associated with each objective

# AHP: Quick Overview

---

- Form a pairwise comparison matrix  $A$ , where the number in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column gives the relative importance of objective  $i$  as compared with objective  $j$
- Use a 1-9 scale, with
  - $a_{ij} = 1$  if objectives  $i$  and  $j$  are of equal importance
  - $a_{ij} = 3$  if objective  $i$  is weakly more important than objective  $j$
  - $a_{ij} = 5$  if objective  $i$  is moderately more important than objective  $j$
  - $a_{ij} = 7$  if objective  $i$  is strongly more important than objective  $j$
  - $a_{ij} = 9$  if objective  $i$  is absolutely more important than objective  $j$
- Similarly,
  - $a_{ij} = 1/3$  if objective  $j$  is weakly more important than objective  $i$
  - And so on...

# AHP: Quick Overview

---

- To normalize the weights, calculate the sum of each column and divide each column element by the corresponding sum
- The average of each row represents the weight associated with the objective given in that row
- For example, the weights for  $A$  can be calculated as

$$A = \begin{bmatrix} 1 & 1/5 & 1/3 & 1/7 \\ 5 & 1 & 3 & 5 \\ 3 & 1/3 & 1 & 3 \\ 7 & 1/5 & 1/3 & 1 \end{bmatrix} \begin{matrix} \rightarrow 0.066 \\ \rightarrow 0.520 \\ \rightarrow 0.231 \\ \rightarrow 0.183 \end{matrix}$$

# Evaluations : Phase 2 (cont'd)

---

## ● Survey results:

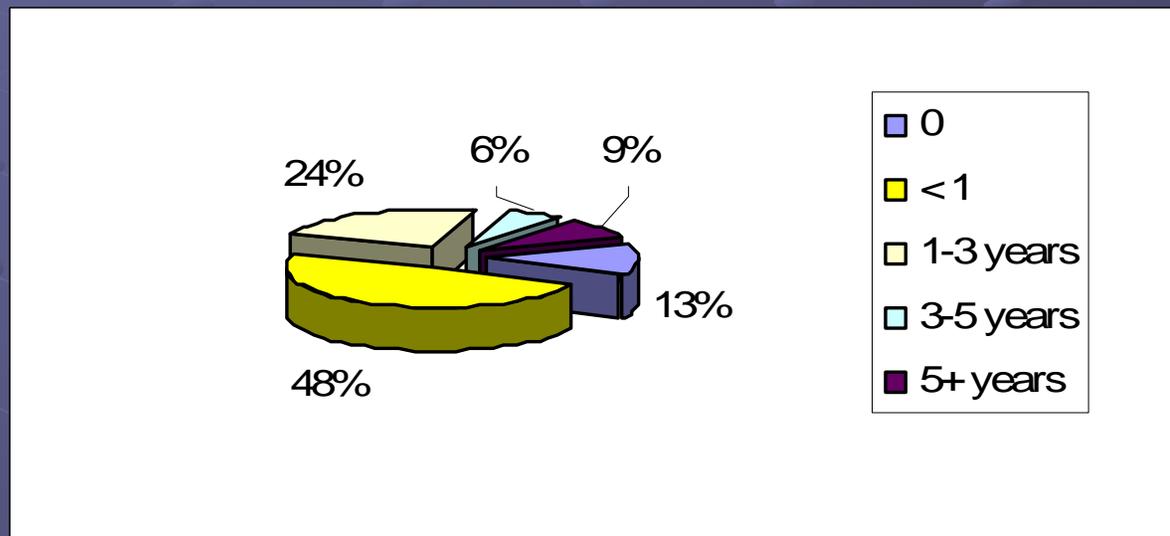
- 70+ responses
- Users from a wide range of backgrounds and application areas
- Reflected in users' answers to the survey...

## Evaluations : Phase 2 (cont'd)

---

### ● Highlights of the survey results:

- Distribution of users – number of years of experience with MFIX

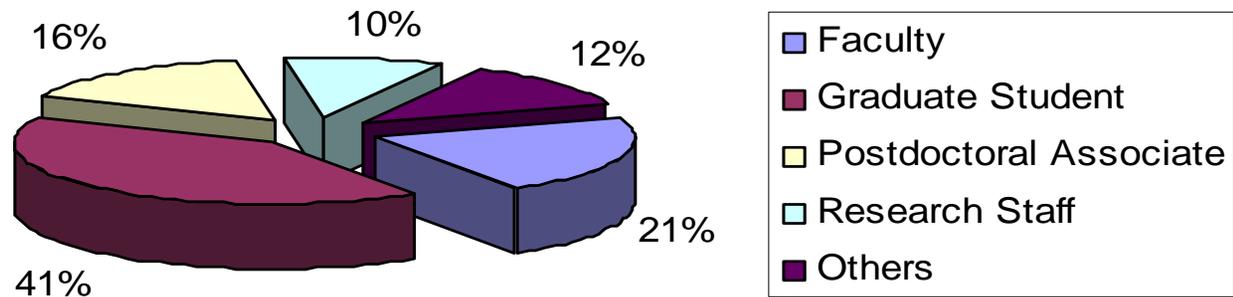


## Evaluations : Phase 2 (cont'd)

---

### ● Survey results:

- Distribution of users—faculty vs. graduate student

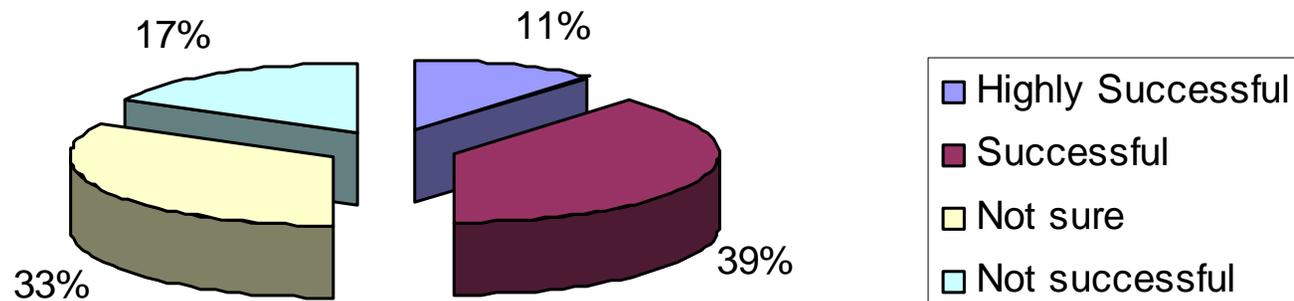


# Evaluations : Phase 2 (cont'd)

---

## ● Survey results:

- Fraction of users indicating highly successful, successful and unsuccessful applications



# Evaluations : Phase 2 (cont'd)

## Survey results:

- Wide range of values for the weights – due to users' diverse backgrounds

	Weight of Obj. 1	Weight of Obj. 2	Weight of Obj. 3	Weight of Obj. 4
Min	0.057	0.059	0.035	0.034
Max	0.645	0.613	0.606	0.716
Average	0.335	0.255	0.181	0.230
Std Dev	0.152	0.129	0.138	0.166

Obj #	Feature
1	Physics Rep.
2	Numerical Soln. Scheme
3	S/W Develop. & Maintenance
4	Open Source S/W Distribution

## Questions to be addressed :

---

1. What percent of the desired features are available?
  - Feature Ratings compared with ideal case software
2. What is the relative importance of each desired feature in the SRD for Next Generation MFIX ?
  - Conduct user survey among MFIX users
3. **Weighted ratings: Based on the importance level of desired features, which package seems to offer best set of features ?**
  - **Multi-criteria decision making**

# Feature Rating of Alternatives:

---

- Feature Ratings tabulated to show % of available w.r.t. ideal case software

	Obj. 1 (%)	Obj. 2 (%)	Obj. 3 (%)	Obj. 4 (%)
OpenFOAM	46.1	52.0	39.6	100.0
Trilinos	3.9	26.0	56.3	100.0
SAMRAI	0.0	34.0	41.7	62.5
OVERTURE	3.9	38.0	0.0	50.0
PETSc	23.1	0.0	0.0	50.0
AMROC	0.0	20.0	0.2	75.0
ROCCOM	7.7	8.0	0.0	0.0

# Weighted Results:

- Using “average” weights (over all users)

	Obj. 1 (%)	Obj. 2 (%)	Obj. 3 (%)	Obj. 4 (%)	Weighted Average
OpenFOAM	46.1	52.0	39.6	100.0	58.8
Trilinos	3.9	26.0	56.3	100.0	41.1
SAMRAI	0.0	34.0	41.7	62.5	30.6
OVERTURE	3.9	38.0	0.0	50.0	22.5
PETSc	23.1	0.0	0.0	50.0	19.2
AMROC	0.0	20.0	0.2	75.0	22.4
ROCCOM	7.7	8.0	0.0	0.0	4.6
<b>WEIGHTS</b> (Avg. of users)	0.335	0.255	0.180	0.230	

OpenFOAM optimal for this set of weights

## Weighted Results: (cont'd)

Using “average” weights (over all users)

	Obj. 1 (%)	Obj. 2 (%)	Obj. 3 (%)	Obj. 4 (%)	Weighted Average
OpenFOAM	46.1	52.0	39.6	100.0	65.8
Trillinos	3.9	26.0	56.3	100.0	71.0
SAMRAI	0.0	34.0	41.7	62.5	53.7
OVERTURE	3.9	38.0	0.0	50.0	21.2
PETSc	23.1	0.0	0.0	50.0	11.7
AMROC	0.0	20.0	0.2	75.0	22.5
ROCCOM	7.7	8.0	0.0	0.0	2.1
<b>WEIGHTS</b>	<b>0.010</b>	0.255	<b>0.735</b>	0.230	

Trillinos optimal for this set of weights

# Conclusions

---

- While trying to decide on a suitable framework, we developed a systematic evaluation approach based on multi-criteria group decision making.
- Effectiveness of our evaluation approach will rely on:
  - how well the software requirements document is drafted.
  - capability to get objective feedback for feature ratings (i.e., multiple users and the developers rate the features w.r.t. SRD instead of one)
  - documentation provided by the framework developers

## Conclusions (cont'd)

---

- As this is a work in progress, we will investigate several other PSEs depending on the resource availability