User Interfaces Development in openDIEL

Argens Ng (HKU), Tanner Curren (Maryville College)

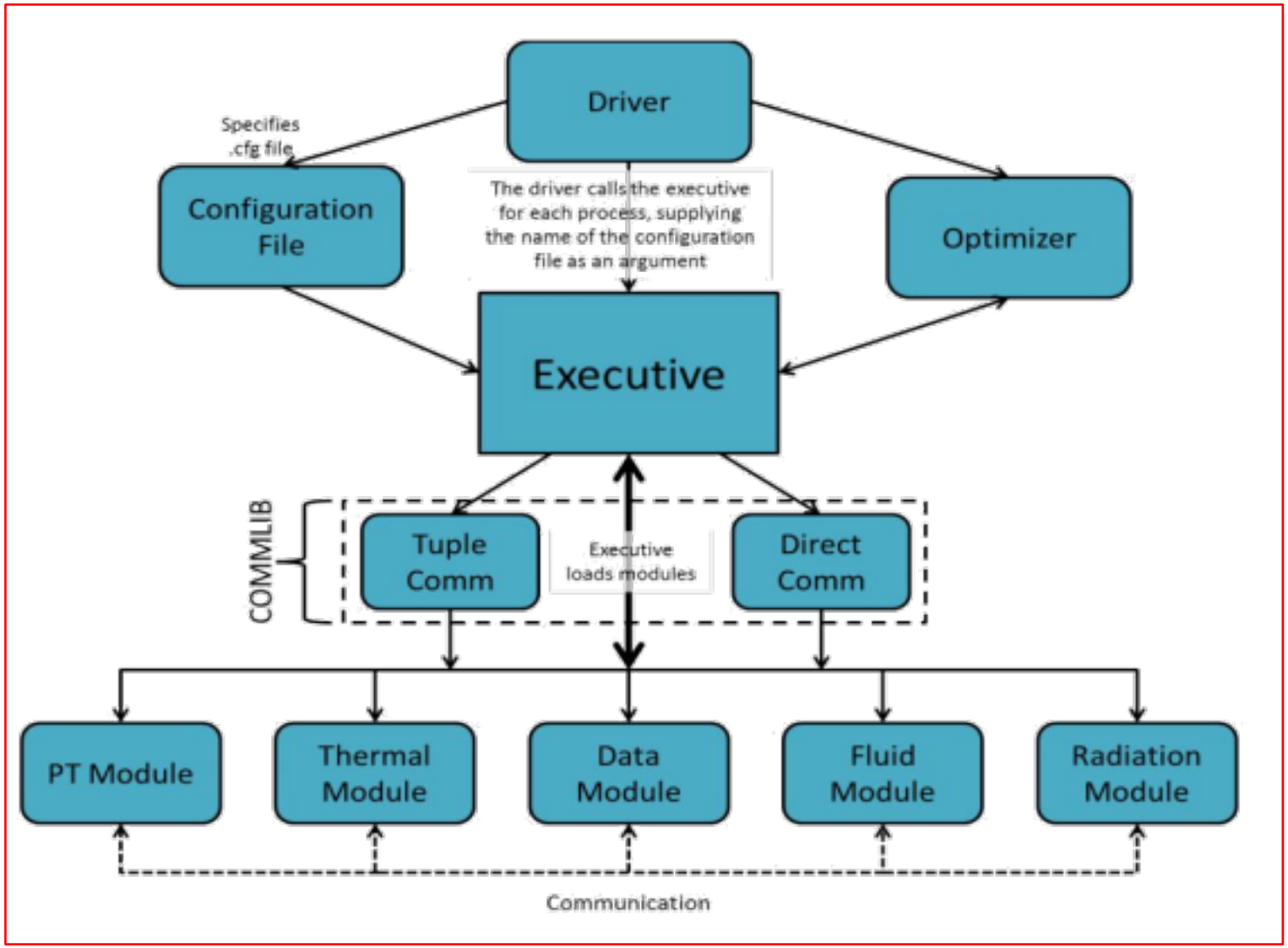
Mentor: Kwai Wong (UTK)

OpenDIEL – Introduction

- Open Distributive Interoperable Executable Library
- Uses MPI (Message Passing Interface)
- Cooperation between loosely coupled modules
- Outputs single executable
- Crucial in combining multiple simulation scenario !!

OpenDIEL – Manual

- User has to provide:
  - Modules (Executable, C module, Fortran Module, Executable)
  - Configuration File (LibConfig)
  - Driver (C - File)

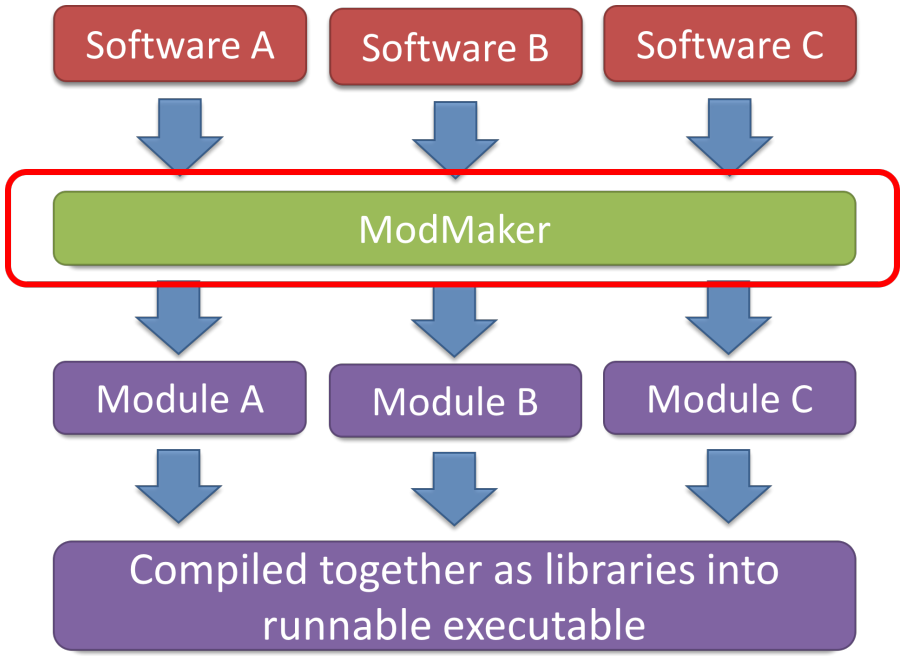

Driver
Specifies
.cfg file
Configuration File
The driver calls the executive for each process, supplying the name of the configuration file as an argument
Optimizer
Executive
COMMLIB
Tuple Comm
Executive loads modules
Direct Comm
PT Module
Thermal Module
Data Module
Fluid Module
Radiation Module
Communication

OpenDIEL – Enhancements

- ModMaker
- Timer
- I/O Improvements
- Fortran Support

ModMaker – Introduction

- Transform at source code level
- No longer compiles into standalone executable
- Compiles into a library



ModMaker – Workflow

ModMaker – Workflow

1

```
#include "mpi.h”
#include "stdio.h”
#include "test.h”
int main(int argc,char**argv)
{
        int rank;
        int size;
        int rc;

        rc |= MPI_Init (&argc, &argv);
        MPI_Comm_rank (MPI_COMM_WORLD, &rank);
        MPI_Comm_size (MPI_COMM_WORLD, &size);
        printf ("I am process %d out of all %d processes\n", rank, size);
        MPI_Barrier (MPI_COMM_WORLD);
        MPI_Finalize();

        return 0;
}
```

ModMaker – Workflow

2

Identifying Feature

```
int main(int argc,char**argv)
{
        int rank;
        int size;
        int rc;

        rc |= MPI_Init (&argc, &argv);
        MPI_Comm_rank (MPI_COMM_WORLD, &rank);
        MPI_Comm_size (MPI_COMM_WORLD, &size);
        printf ("I am process %d out of all %d processes\n", rank, size);
        MPI_Barrier (MPI_COMM_WORLD);
        MPI_Finalize();

        return 0;
}
```

ModMaker – Workflow

3

```
Transforming file "test.c" into module "first"
Old files will be put into directory Archive_modMaker
3 MPI_COMM_WORLD have been replaced.

We have found something that may need to be changed in your code for the followi
ng reason:

        Only one Main can exist and it belongs in the driver.c

5)      int main(int argc,char**argv)
        ----------------------------

If you agree on this change press [Enter] or key in [y/Y]. If it is incorrect, k
ey in [n/N]
```

ModMaker – User Participation

```
Transforming file "test.c" into module "first"
Old files will be put into directory Archive_modMaker

Press Enter to Continue or type "exit" to quit:
We are about to conduct transformation on test.c.
Press [Enter] to begin or key-in "n/N" to skip this file.
```

```
Transforming file "test.c" into module "first"
Old files will be put into directory Archive_modMaker
3 MPI_COMM_WORLD have been replaced.
Suspected Main remained unchanged
Suspected MPI_Init remained unchanged
Suspected MPI_Finalize remained unchanged
```

ModMaker – User Participation

```
Transforming file "test.c" into module "first"
Old files will be put into directory Archive_modMaker

13)             MPI_Comm_rank (MPI_COMM_WORLD, &rank);
                              ---------------

14)             MPI_Comm_size (MPI_COMM_WORLD, &size);
                              ---------------

18)             MPI_Barrier (MPI_COMM_WORLD);
                             ---------------

We have found 3 occurances of MPI_COMM_WORLD.
Press [Enter] to authorize every change or key-in "n/N" to authorize one-by-one.
```

ModMaker – User Participation

```
What file types should we focus on? Please optionally enter extensions one by on
e and end with a blank line.
(Include "." and use small letters. Example: ".c")

Extension:
```

```
What file types should we focus on? Please optionally enter extensions one by on
e and end with a blank line.
(Include "." and use small letters. Example: ".c")

Extension: .c
Extension:

Press Enter to Continue or type "exit" to quit:
```

ModMaker – User Participation

```
=====In file 1_SOURCE/a30x0e1d.c=====

=====In file 1_SOURCE/a3k0ke1d.c=====

=====In file 1_SOURCE/a3x00e1d.c=====

=====In file 1_SOURCE/an10e1d.c=====

=====In file 1_SOURCE/an200e1d.c=====

=====In file 1_SOURCE/an3000e1d.c=====

=====In file 1_SOURCE/appdbcon1d.c=====

We are about to conduct transformation on 1_SOURCE/appdbcon1d.c.
Press [Enter] to begin or key-in "n/N" to skip this file.
```

ModMaker – Pattern Matching

- Two options were considered
  - Matching by character
  - Matching by string

- Matching by string is used
  - Simplicity
  - Similar performance

$$O(n, s) = ns$$

n = string size, s = document size (measured in character)

Text	/	*	T	h	i	s		i	s
Pattern 1	i	n	t		m	a	i	n	
Pattern 2	M	P	I	_	I	n	i	t	
No match x 2	/	*	T	h	i	s		i	
No match x 2		*	T	h	i	s		i	s

Text	/	*	T	h	i	s		i	s
Pattern 1	i	n	t		m	a	i	n	
No match	/	*	T	h	i	s		i	
No match		*	T	h	i	s		i	s
Pattern 2	M	P	I	_	I	n	i	t	
No match	/	*	T	h	i	s		i	
No match		*	T	h	i	s		i	s

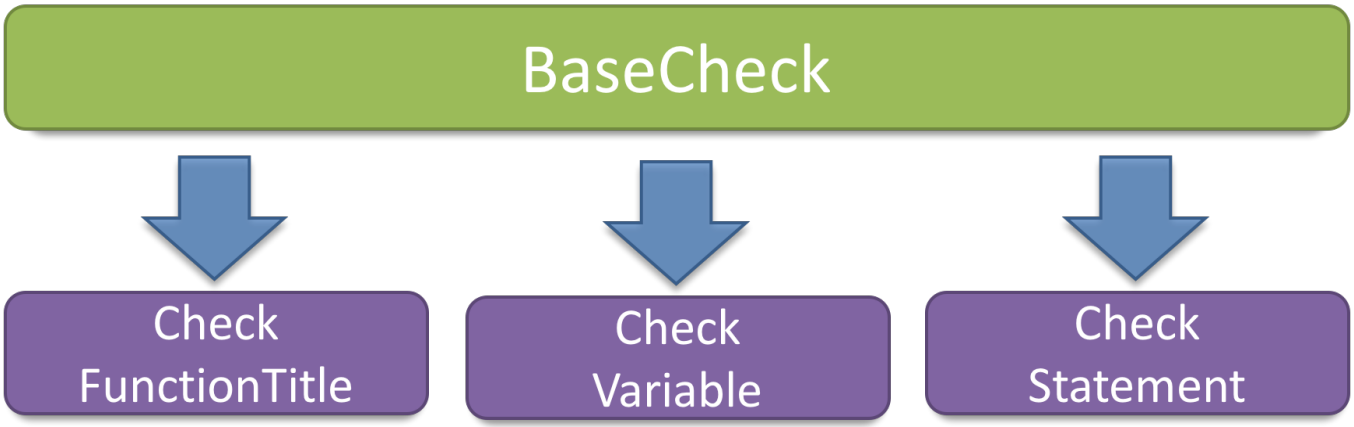
ModMaker – Syntactic Freedom

```
1) int main(int argc,char**argv){\n
2) MPI_Init(&argc, &argv);
3) rc=MPI_Init(&argc,&argv);
4) int rc=MPI_Init(&argc,&argv);
5) int main(intargc,char**argv) {\n
```

- Locating
- Spaces
- Variable Declaration

ModMaker – Modularization

- Function Title
- Statement
- Variable

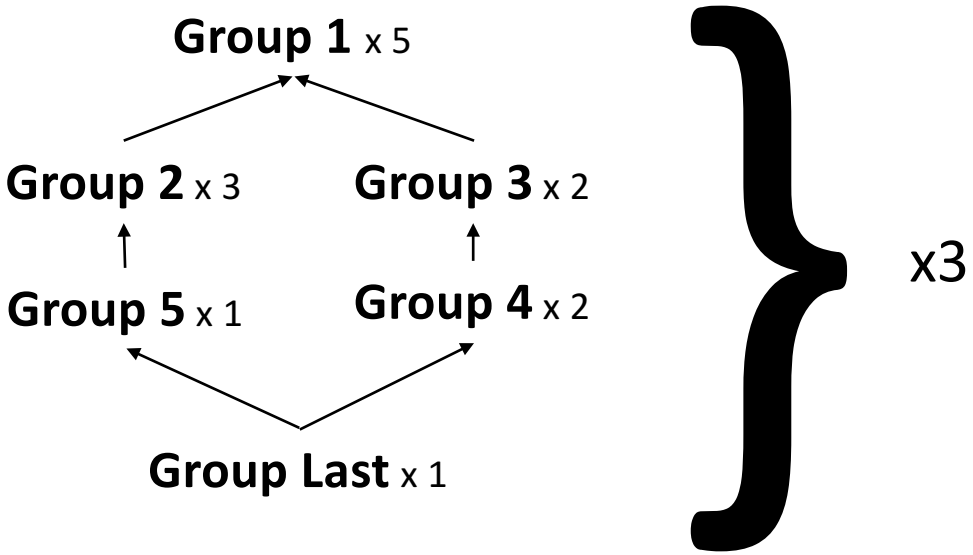


- BaseCheck (line, tokenList, prevSep, nextSep, msgBundle, replacement, returnValue)

ModMaker – Testing

- Tester.c
- Dummy “IEL.h”

Timer – Background




```
num_set_runs=3

tuple:
{
  order=("ielTupleServer")
  iterations=1
}

# This group has no dependencies, so it should run first
group1:
{
  order=("first")
  iterations=5
}
# group2 and group3 will both begin running at the same time after
# group1 has finished
group2:
{
  order=("second")
  iterations=3
  depends=("group1")
}
group3:
{
  order=("third")
  iterations=2
  depends=("group1")
}
group4:
{
  order=("fourth")
  iterations=2
  # Will only run after group3 has finished
  depends=("group3")
}
group5:
{
  order=("fifth")
  iterations=1
  depends=("group2")
},
grouplast:
{
  order=("last")
  iterations=1
  # Will only run after all other groups have finished
  depends=("group1", "group2", "group3", "group4", "group5")
}
```

Timer – Results

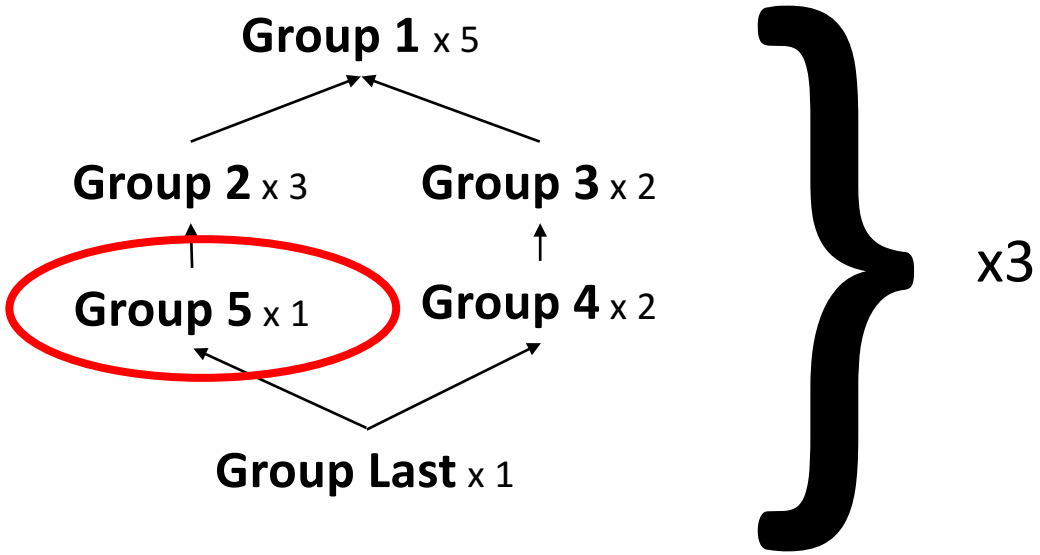
```
-------------------------------------------------------
Most Idle Time:        Process 11      30.195744 seconds (90.631661%).
Earliest End Time:     Process 1       time = 27.261870 seconds.
Latest End Time:       Process 0       time = 33.316978 seconds.
-------------------------------------------------------
```

```
0.048955 (-0.023994)                              [Begin] Info Unpack
0.048968 ( 0.000013)                              [ End ] Info Unpack
0.048968 (-0.000000)                              [Begin] Direct Communication Setup
0.096939 ( 0.047971)                              [ End ] Direct Communication Setup
0.096940 (-0.000000)                              [Begin] Workflow Setup
0.145935 ( 0.048995)                              [ End ] Workflow Setup
8.150633 (-8.004699) [-24.025%]                   [Begin] Function_5
9.150695 ( 1.000062)                              [ End ] Function_5
19.25096 (-10.10027) [-30.314%]                   [Begin] Function_5
20.25102 ( 1.000057)                              [ End ] Function_5
30.27751 (-10.02648) [-30.093%]                   [Begin] Function_5
31.27764 ( 1.000133)                              [ End ] Function_5
33.31793 (-2.040292)                              [Begin] IEL Finalize
33.31794 ( 0.000005)                              [ End ] IEL Finalize
33.31794 ( 33.31794)                        [ End ] driver.c

Total Idle Time = 30.195744 seconds. (90.629081%)
End Time = 31.277643.
```

Process 11

Timer – Background




```
num_set_runs=3

tuple:
{
  order=("ielTupleServer")
  iterations=1
}

# This group has no dependencies, so it should run first
group1:
{
  order=("first")
  iterations=5
}
# group2 and group3 will both begin running at the same time after
# group1 has finished
group2:
{
  order=("second")
  iterations=3
  depends=("group1")
}
group3:
{
  order=("third")
  iterations=2
  depends=("group1")
}
group4:
{
  order=("fourth")
  iterations=2
  # Will only run after group3 has finished
  depends=("group3")
}
group5:
{
  order=("fifth")
  iterations=1
  depends=("group2")
},
grouplast:
{
  order=("last")
  iterations=1
  # Will only run after all other groups have finished
  depends=("group1", "group2", "group3", "group4", "group5")
}
```

Timer – Modularity

- void timestamp (char* tag, char* function, int level_change)
- void timer_finalize (int rank)

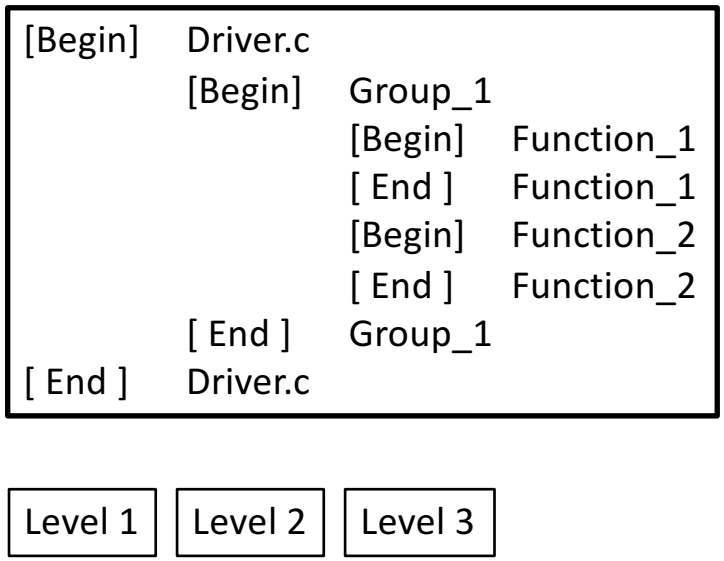
Timer – Modularity

- Driver.c
  - Group_1
    - Function_1
    - Function_2

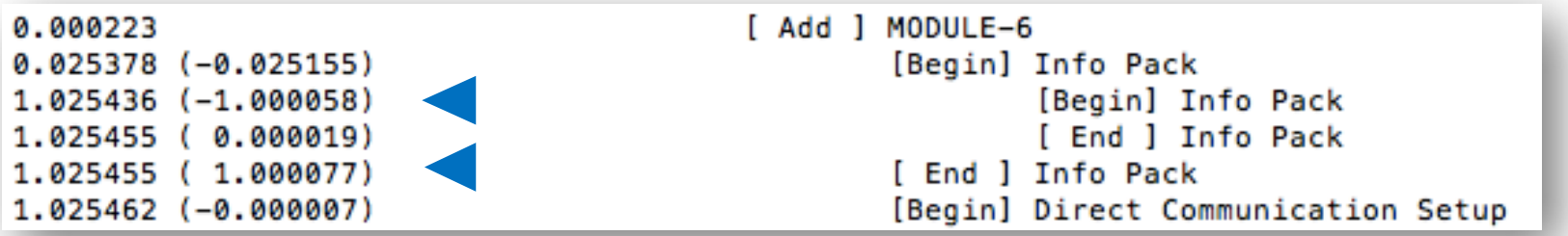
Level 1

Level 2

Level 3



Timer – Example



- -ve means: possible overhead
- +ve means: process runtime

Future Work

- Multiple I/O
- ModMaker – Fortran Support

Reference

- openDIEL
- http://cfdlab.utk.edu/openDIEL/opendiel.php

- Stack Overflow
- http://stackoverflow.com

Acknowledgement

This project is made possible only with the support of our mentor Dr. K. Wong, the NSF, the University of Tennessee and Oak Ridge National Laboratory