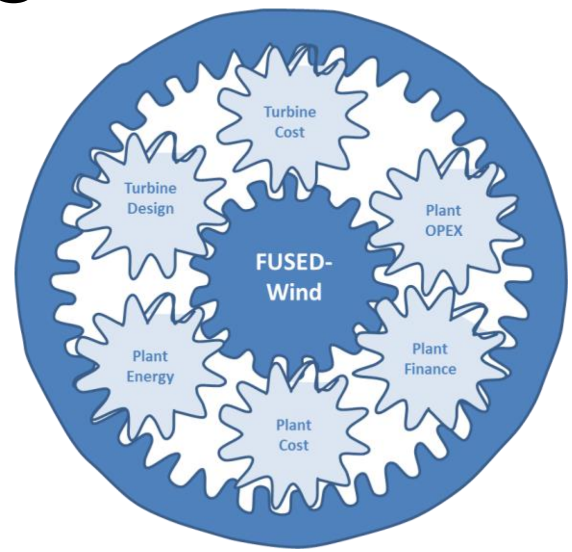
The Framework for Unified Systems Engineering and Design of Wind Plants (FUSED Wind)



Tutorial Presentation
September 15th, 2017

Frederik Zahle & Mike McWilliam, DTU Wind Energy
Katherine Dykes, NREL National Wind Technology Center
Andrew Ning, Brigham Young University

FUSED-Wind

Outline

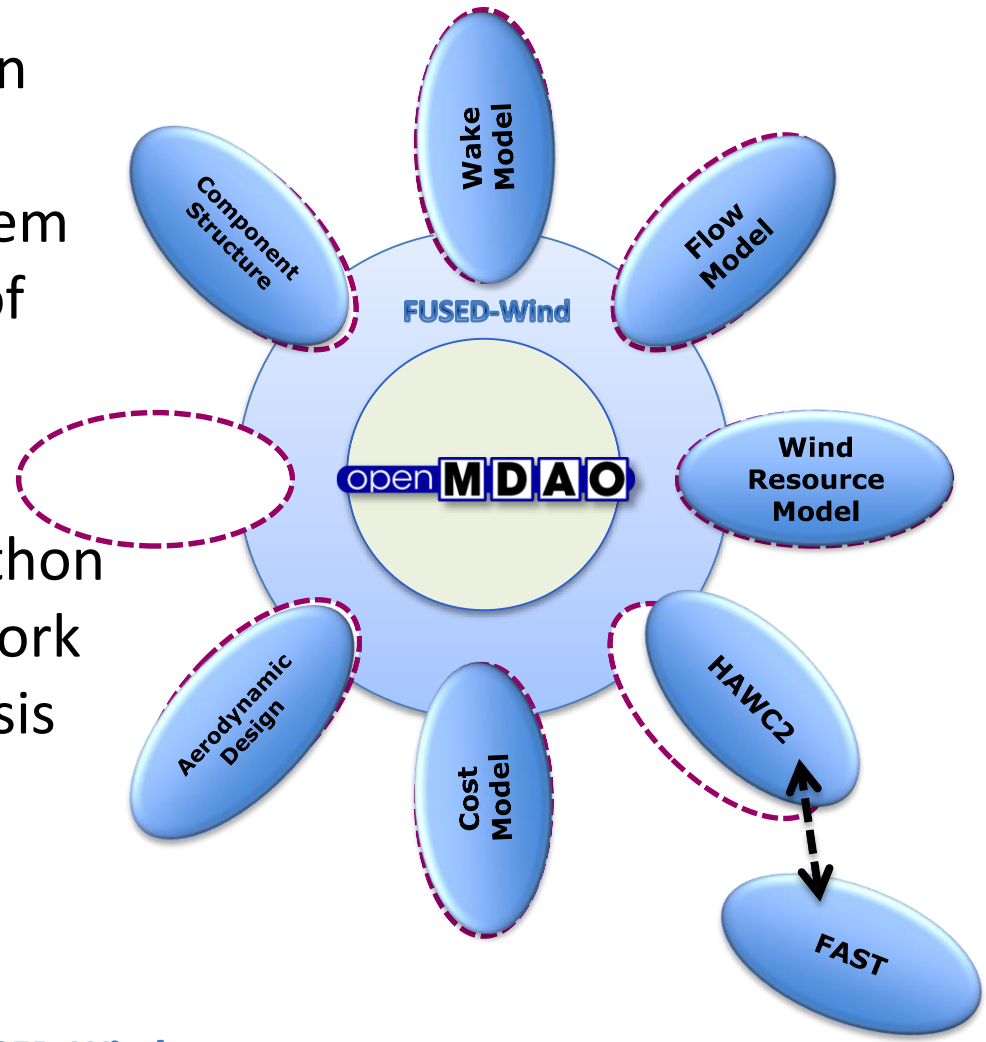
- FUSED-Wind Background and Overview
- OpenMDAO Overview and Example
- FUSED-Wind Structure

FUSED-Wind Background & Overview

FUSED Wind

- Collaborative effort between **DTU** and **NREL** to create a **F**ramework for **U**nified **S**ystem **E**ngineering and **D**esigned of **Wind** energy plants.

- Based on OpenMDAO, a python based **Open** source framework for **M**ulti-**D**isciplinary **A**nalysis and **O**ptimization.



FUSED-Wind

- The FUSED-Wind mission is to establish an open-source collaborative platform for research in wind energy MDAO:
  - Define standard interfaces for wind turbine and plant models
  - Define standard assemblies for common wind energy workflows that are “model agnostic”
  - Establish a library of wind energy model wrappers and utilities to support common analyses
  - Provide standard generic tools to use in wind energy (e.g. standard inputs / outputs files, IEC load calculation, multi-fidelity, UQ)

User scenarios:

"As a wind turbine engineer I would like to …"

- Perform multi-disciplinary optimization/analysis of a wind turbine with my own sub-models
- obtain easily a "second opinion" on my design by swapping the aero-elastic model
- run an optimization with turbine component and aeroelastic models of varying levels of fidelity
- expand the optimization to include 3rd party energy production and cost models

User scenarios:

"As an innovative component technology designer I'd like to…"

- Perform multi-disciplinary optimization/analysis of a wind turbine with my own sub-models for my component technology
- obtain easily a "second opinion" on my design by comparison to conventional technology
- run an optimization with turbine component and aeroelastic models of varying levels of fidelity
- expand the optimization to include 3rd party energy production and cost models

User scenarios:
"As a wind farm planner I would like to …"

- Perform multi-disciplinary optimization/analysis of a wind farm with my own sub-models
- obtain easily a "second opinion" on my design by swapping the wind farm flow model
- run an optimization with wind farm flow models of varying levels of fidelity
- expand the optimization to include 3rd party turbine and cost models

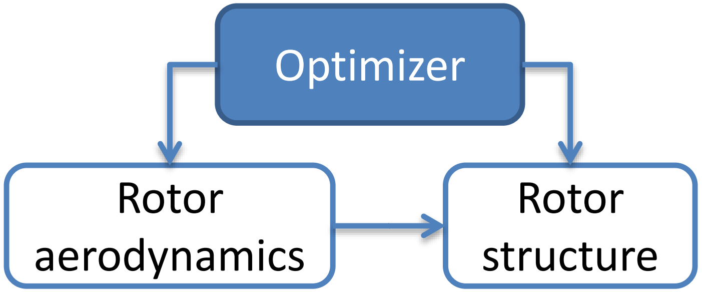
User scenarios:

"As a researcher I would like to …"

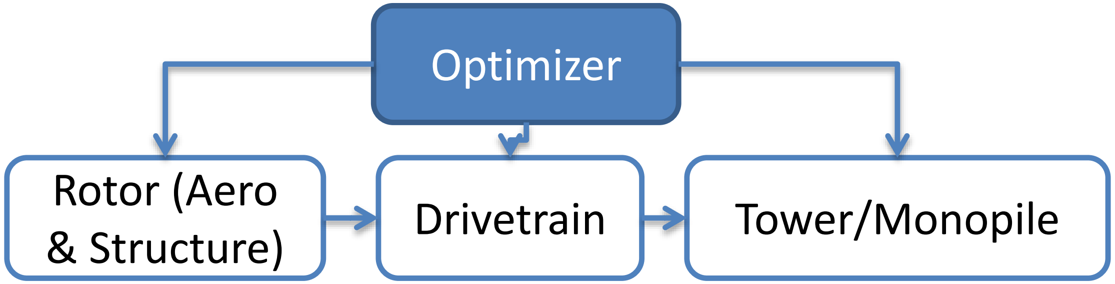
- have a common multi-disciplinary optimization-analysis framework to apply my model in a larger context and to other models
- have standard assemblies to benchmark optimizers on the same problem
- have a platform for promoting and getting feedback on my models
- to make my model as "easy to use" as possible by my collaborators and end-users

Example Applications

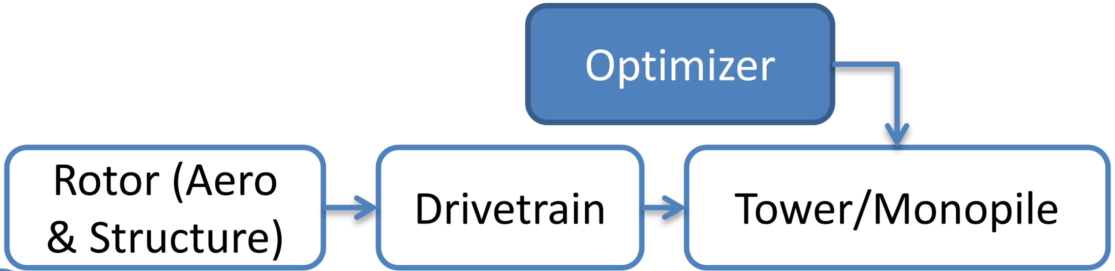
1) Rotor Aero-structural optimization



2) Full turbine redesign



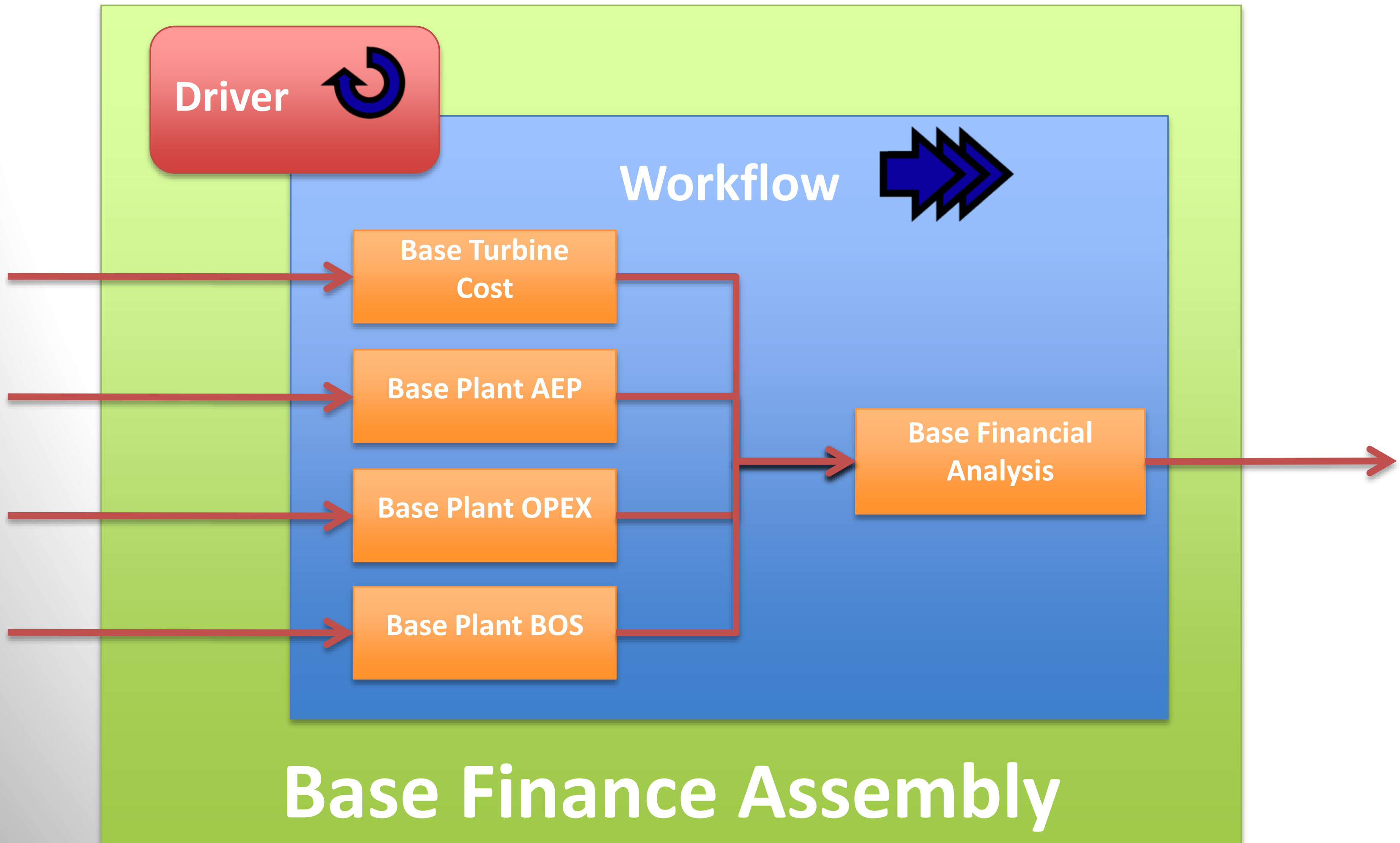
3) Design of substructure for specific offshore sites



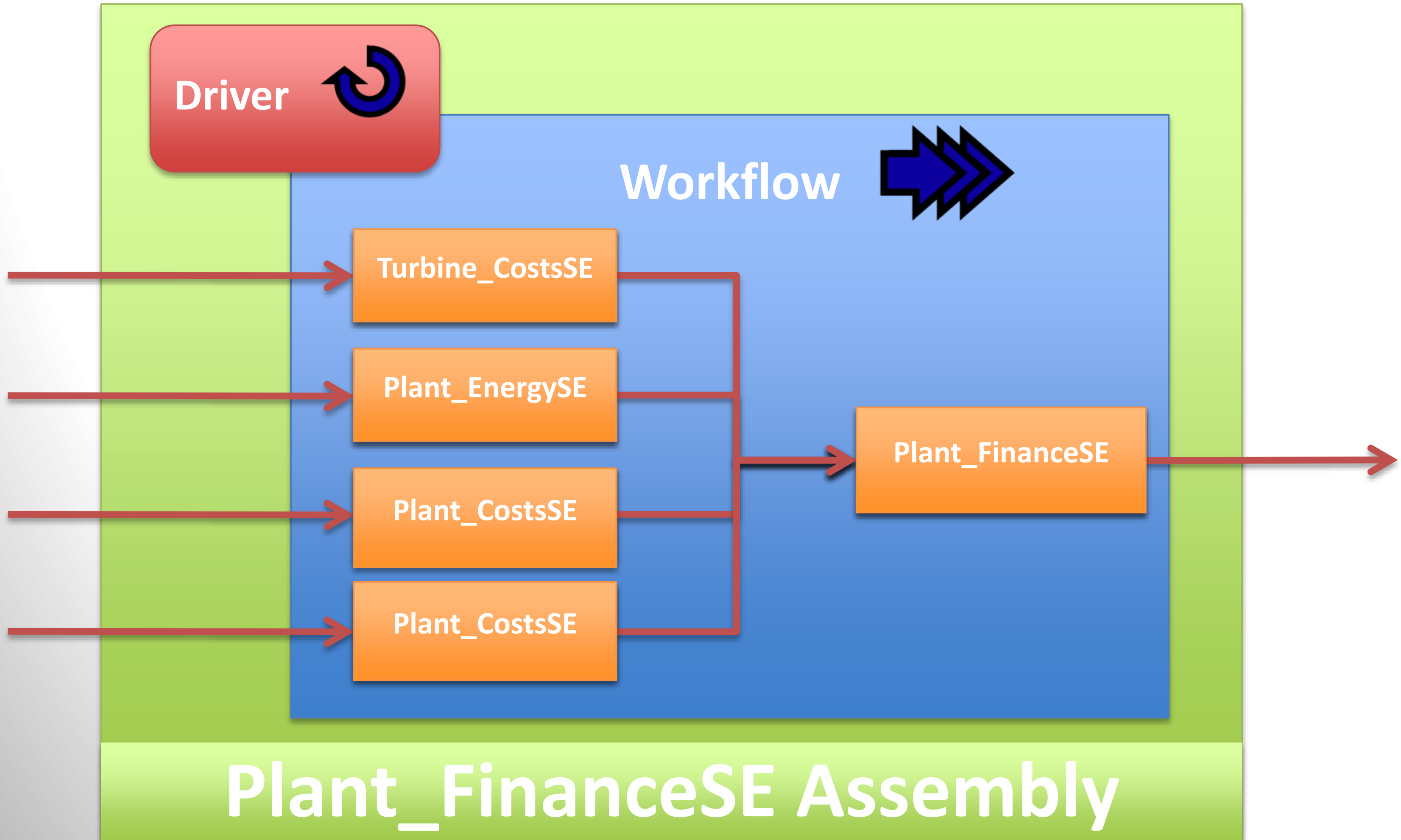
FUSED-Wind Interface Definitions

- Environmental conditions
- Wind turbine
  - Blade geometry and airfoil data
  - Detailed structural description
  - Aerodynamic outputs
  - Solver interfaces
- Wind plant
  - Layout
  - Wake models
  - AEP, Capacity factor
- Wind plant cost
  - Balance of Station
  - Operational Expenditures
  - Financing

FUSED-Wind Interface Example (COE)



FUSED-Wind Interface Example (COE)



Models wrapped to FUSED-Wind

- DTU:
  - AirfoilOpt2:
    - Xfoil
    - EllipSys2D
    - TEnoise
  - HawtOpt2:
    - HAWC2
    - HAWCstab2
    - BECAS
    - CSProps
    - EllipSys3D
  - TOPFARM
    - Wake models:
      - GC Larsen
      - NO Jensen
- NREL:
  - WISDEM
    - Turbine_CostsSE
    - Plant_CostsSE
    - Plant_EnergySE
      - Includes AWS Truepower openWind wrapper
    - Plant_FinanceSE
    - RotorSE
    - DriveSE
    - GeneratorSE
    - Tower/MonopileSE
    - JacketSE
    - FLORISSE
    - WindSE
    - AeroelasticSE (FAST)

Development to Date

- v0.1: Jan 2015
  - Website
  - Public release
  - Standard interfaces for plant cost, energy production and turbine design
  - Gather stakeholder feedback

Current release version of WISDEM uses FUSED-Wind v0.1

http://fusedwind.org

FUSED-Wind   Site ▾   Page ▾   News »   Search

Versions

Development
0.1.dev Github

Stable
v0.1.0

Contents

Overview

Framework for Unified Systems Engineering and Design of Wind Plants (FUSED-Wind) is a free open-source framework for multi-disciplinary optimisation and analysis (MDAO) of wind energy systems, developed jointly by the Wind Energy Department at the Technical University of Denmark (DTU Wind Energy) and the National Renewable Energy Laboratory (NREL). The framework is built as an extension to the NASA developed OpenMDAO, and defines key interfaces, methods and I/O variables necessary for wiring together different simulation codes in order to achieve a system level analysis capability of wind turbine plants with multiple levels of fidelity. NREL and DTU have developed independent interfaces to their respective simulation codes and cost models with the aim of offering an environment where these codes can be used interchangeably. The open source nature of the framework enables third parties to develop interfaces to their own tools, either replacing or extending those offered by DTU and NREL.

GitHub Repository

The project source code is hosted on https://github.com/FUSED-Wind. Along with the FUSED-Wind source code, you can find the code for the examples and tutorials accompanying the documentation on this site. On github.com you can also ask questions, report bugs and request features. For a better overview of all issues and the current progress of the project visit our Waffle page.

Contacts

If you want more information about the platform, please contact the following authors

DTU: Pierre-Elouan Réthoré, Frederik Zahle,

NREL: Katherine Dykes, Peter Graf, Andrew Ning

Created using Sphinx 1.2.3.   Back to top

Development to Date

- From 2015-2017
  - Significant use by NREL and DTU in various projects and also with project partners (in academia, research institutes and industry)
  - Growing pains with OpenMDAO being relatively new (memory issues for parallelization, etc)
  - OpenMDAO 1.x released in 2016 is not backwards compatible (core structure refactored)
  - Other use cases – in particular around uncertainty quantification and analysis – begin to surface

Development to Date

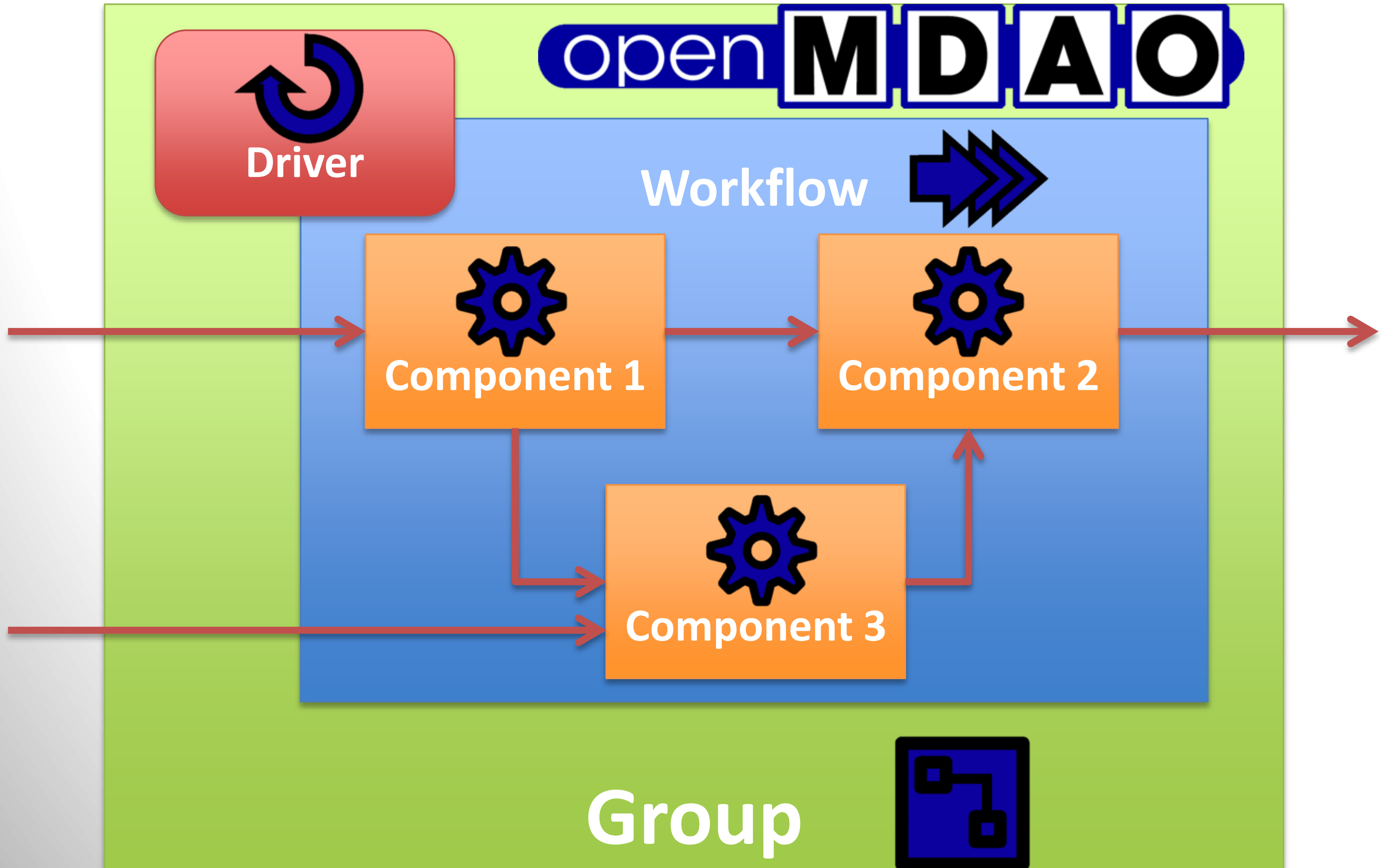
- v1.0 Fall 2017
  - New version of FUSED-Wind created with consideration of three types of software developers:
    1. Wind turbine/plant model developer – no knowledge or use of FUSED-Wind, variable names and API inflexible
    2. FUSED-Wind user – wraps wind turbine/plant models in FUSED-Wind to conform to FUSED-Wind interfaces
    3. FUSED-Wind developer – directly involved in development of used wind; developing helper functions and utilites

Development to Date

- v1.0 Fall 2017
  - Conform to development in IEA Wind Task 37 on framework ontology
    - Adapt to wind turbine and plant ontology as developed in the task
  - Core of FUSED-Wind
    - Interfaces defined through yaml (implementation of IEA Wind Task 37) and imported into Python dictionaries
    - Models wrapped in pure Python as FUSED models
    - OpenMDAO models are created automatically out of a FUSED model but models can also be used in other applications (Chaospy, Scipy, PyOpt, DAKOTA, etc…)

OpenMDAO Overview & Example

What is OpenMDAO?

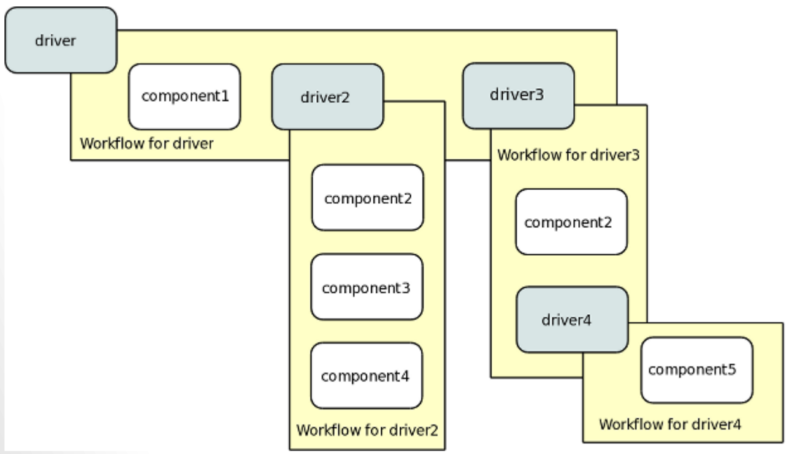


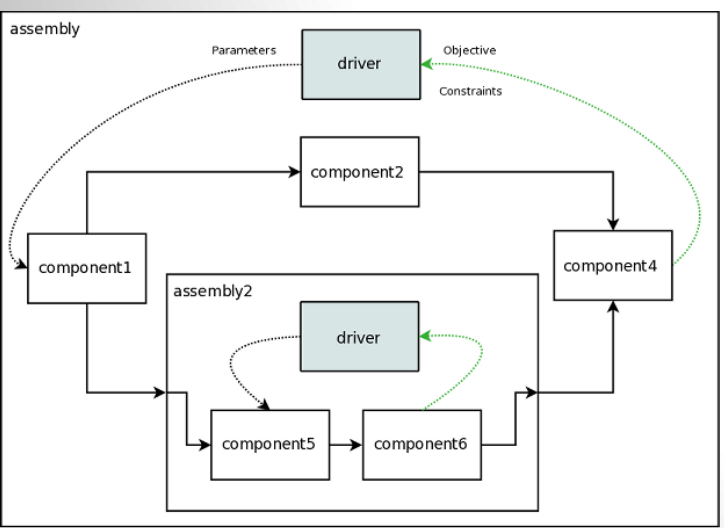
FUSED-Wind http://www.openmdao.org

Reconfiguration of OpenMDAO Assemblies

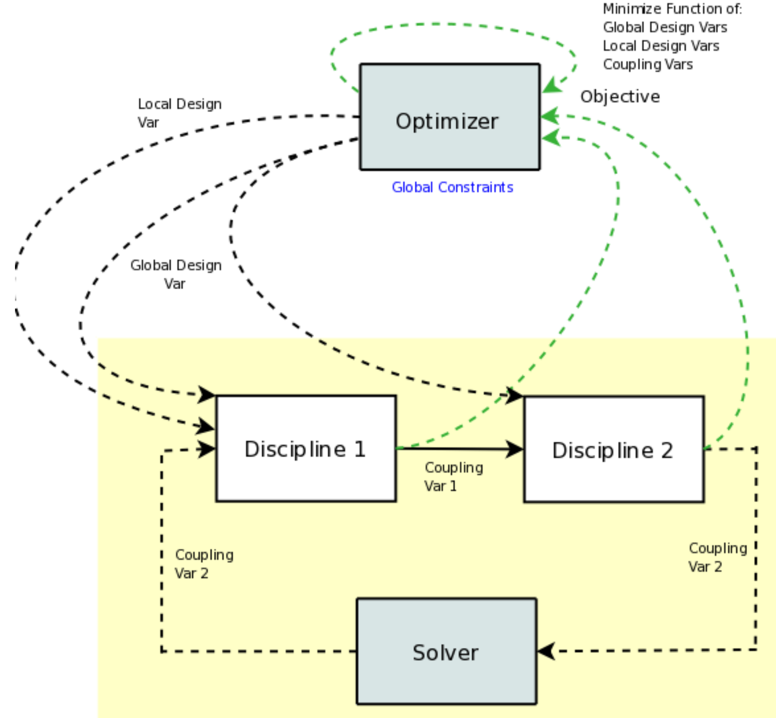
Increases Potential Analysis Complexity

Several nested drivers within one assembly





OpenMDAO handles the local and global analysis dependencies



FUSED-Wind

Different Types of Drivers in OpenMDAO

- **~50 Optimizers**
  - Native optimizers built into OpenMDAO and other available from plug-ins (DAKOTA, pyOpt)
  - Support for analytic gradients
- **Uncertainty Analysis**
  - Native drivers for Design of Experiments, Sensitivity Analysis, Uncertainty Quantification and many others available from plug-ins (DAKOTA, pyOpt)
- **Support for Parallelism and Optimization with High Fidelity Models**

OpenMDAO Example: The Paraboloid Optimization

FUSED-Wind Structure

FUSED-Wind v 1

- Core of FUSED-Wind
  - Interfaces defined through yaml (implementation of IEA Wind Task 37) and imported into Python dictionaries
  - Models wrapped in pure Python as FUSED models
  - OpenMDAO models are created automatically out of a FUSED model but models can also be used in other applications (Chaospy, Scipy, PyOpt, DAKOTA, etc…)

Writing a FUSED-Wind Compatible Model

- System specification (may be a reference turbine or plant represented in one or more models/disciplines with certain levels of fidelity)
- Either implement direction in FUSED-Wind Python variables or import from Yaml file
  - FUSED-Wind will provide scripts for importing FUSED-Wind input/output files into Python


```
input_format_version: 0

planform:
  hub_radius: 2.8
  SParam:
    - 0.00000
    - 0.03057
    - 0.06222
    - 0.09487
    - 0.12841
    - 0.16274
    - 0.19772
    - 0.23321
    - 0.26907
    - 0.30515
    - 0.34128
    - 0.37731
    - 0.41309
```

Writing a FUSED-Wind Compatible Model

- Creating a model involves wrapping the component and ensuring that it uses FUSED-Wind compatible I/O
- Two methods:
  - Explicit enumeration of variables
  - Import variables for interface from FUSED-Wind

Writing a FUSED-Wind Compatible Model

- Method 1: Explicit enumeration

```
from fused_wind import FUSED_Object , FUSED_OpenMDAO , fusedvar

class my_BEM(FUSED_Object):

    def __init__(self, initialization_input):

        super(my_BEM, self).__init__()

        self.add_input(**fusedvar('rotor_diameter', 126.0)

        ...
```

Writing a FUSED-Wind Compatible Model

- Method 2: Interface import

```
from fused_wind import FUSED_Object , FUSED_OpenMDAO , fusedvar
from windio import fifc_BEM

class my_BEM(FUSED_Object):

    def __init__(self, initialization_input):

        super(my_BEM, self).__init__()

        self.implement_fifc(fifc_BEM)

        ...
```

Writing a FUSED-Wind Compatible Model

- Once a model is wrapped, it can be connected easily to other FUSED-Wind models through common i/o

- FUSED-Wind offers helpers for connecting models in OpenMDAO workflows

Writing a FUSED-Wind Compatible Model

- Connecting into an OpenMDAO workflow
  - OpenMDAO components now created automatically and interconnections handled directly through common naming (promote *)

```
root = Group()
root.add('myBEM', FUSED_OpenMDAO(my_BEM()), promotes=['*'])
prob = Problem(root)
prob.setup()

prob['rotor_diameter'] = 200.0

prob.run()
```

Farther Ahead

- Adaptation of FUSED-Wind for outcome of IEA Wind Task 37 framework guidelines
- Incorporation of higher-fidelity models for both turbine and plant applications
  - Enables multi-fidelity modeling with combined analysis with several models
- ….more ideas?