# Engineered Resilient Systems

## Large Scale Tradespace Capabilities

**18th NDIA Systems Engineering Conference**
**26-29 October 2015**

**Tommer R. Ender, PhD**
Principal Research Engineer
Chief, Systems Engineering Research Division
Georgia Tech Research Institute
tommer.ender@gtri.gatech.edu

**Simon R. Goerger, PhD**
Director, Institute for Systems Engineering Research
Engineer Research & Development Center
US Army Corps of Engineers
simon.r.goerger@erdc.dren.mil

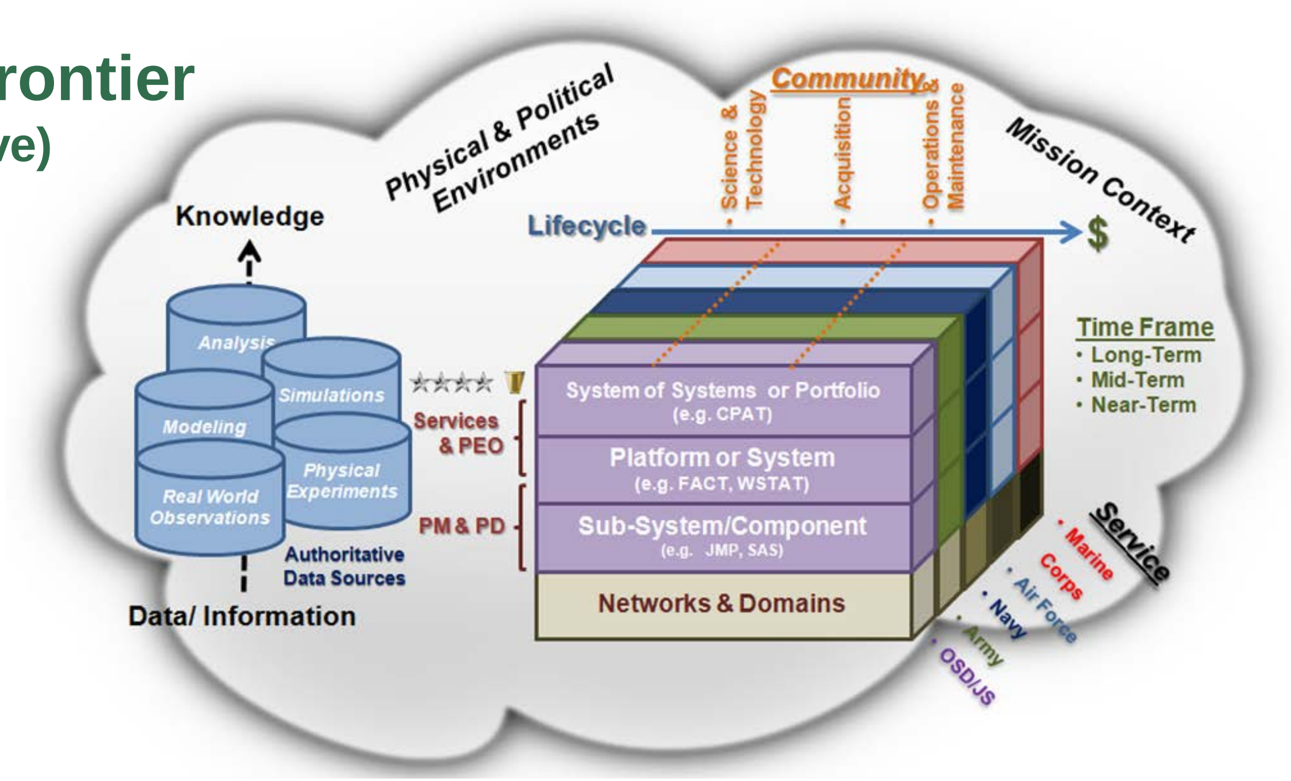
# *What is a “Tradespace”?*

- …the space spanned by completely enumerated design variables – it is the potential solution space
- …the set of program and system parameters, attributes, and characteristics required to satisfy performance standards

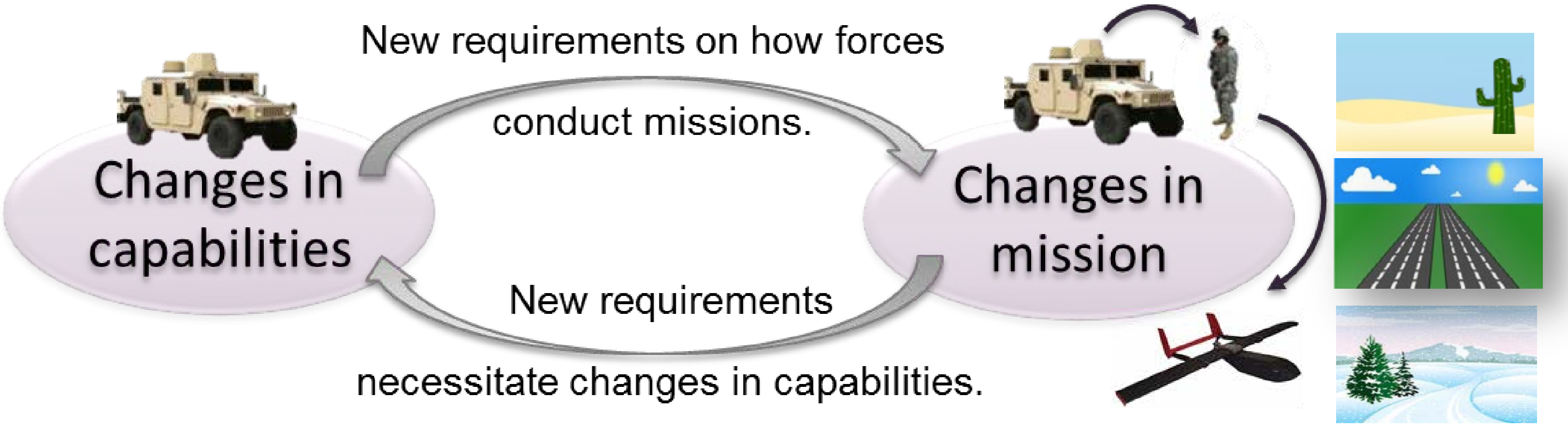
**The enumeration of a large *__tradespace__* helps prevent designers from committing to limited point designs and allows them to recognize better design solutions**

## ERS Tradespace Frontier

## (A DoD Perspective)



# *Analytical Constructs for Resiliency Evaluation*



Changes in capabilities

New requirements on how forces conduct missions.

Changes in mission

New requirements necessitate changes in capabilities.

***"A Resilient System…***

- **is trusted and effective in a wide range of contexts,**

***Operational Context***

- **How do we compare system design concepts in the face of competing or expected changes in requirements?**

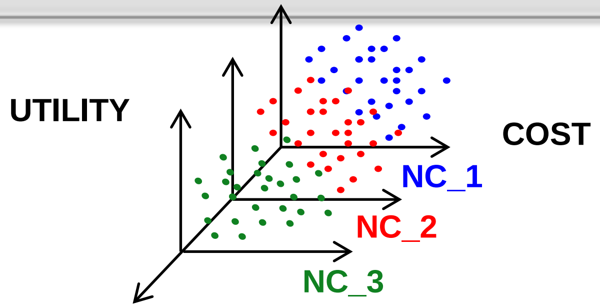
- **is easily adapted to many others through reconfiguration or replacement, and**

***Change Flexibility***

- **How do we evaluate the impact for various system designs if a key component must be replaced?**

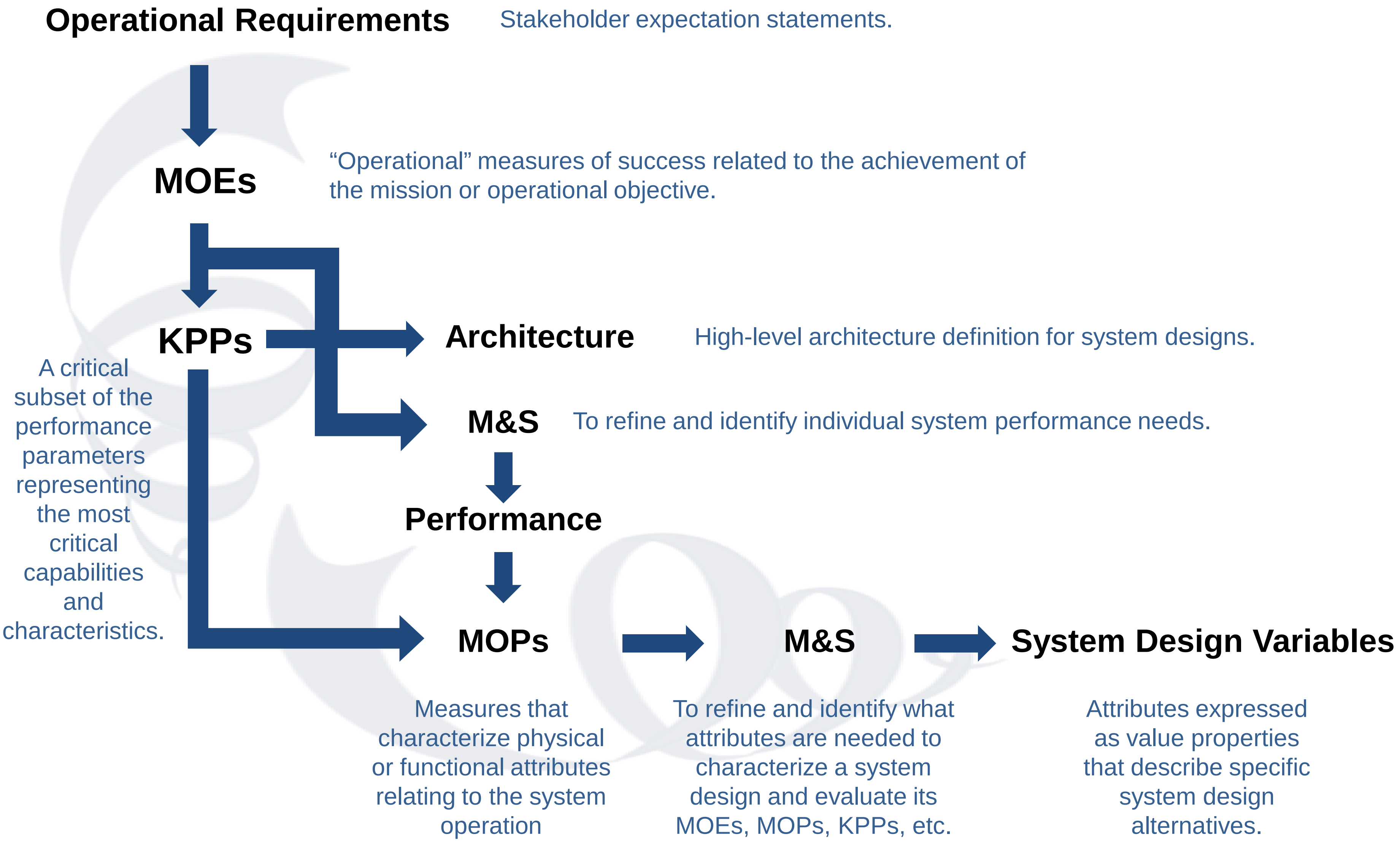
- **has predictable degradation of function."**

~ ***Holland, ERS Overview Dec 2013***



***Multi Attribute Utility Theory***

# The “Big Picture” Process



**Operational Requirements**

Stakeholder expectation statements.

**MOEs**

“Operational” measures of success related to the achievement of the mission or operational objective.

**KPPs**

A critical subset of the performance parameters representing the most critical capabilities and characteristics.

**Architecture**

High-level architecture definition for system designs.

**M&S**

To refine and identify individual system performance needs.

**Performance**

**MOPs**

Measures that characterize physical or functional attributes relating to the system operation

**M&S**

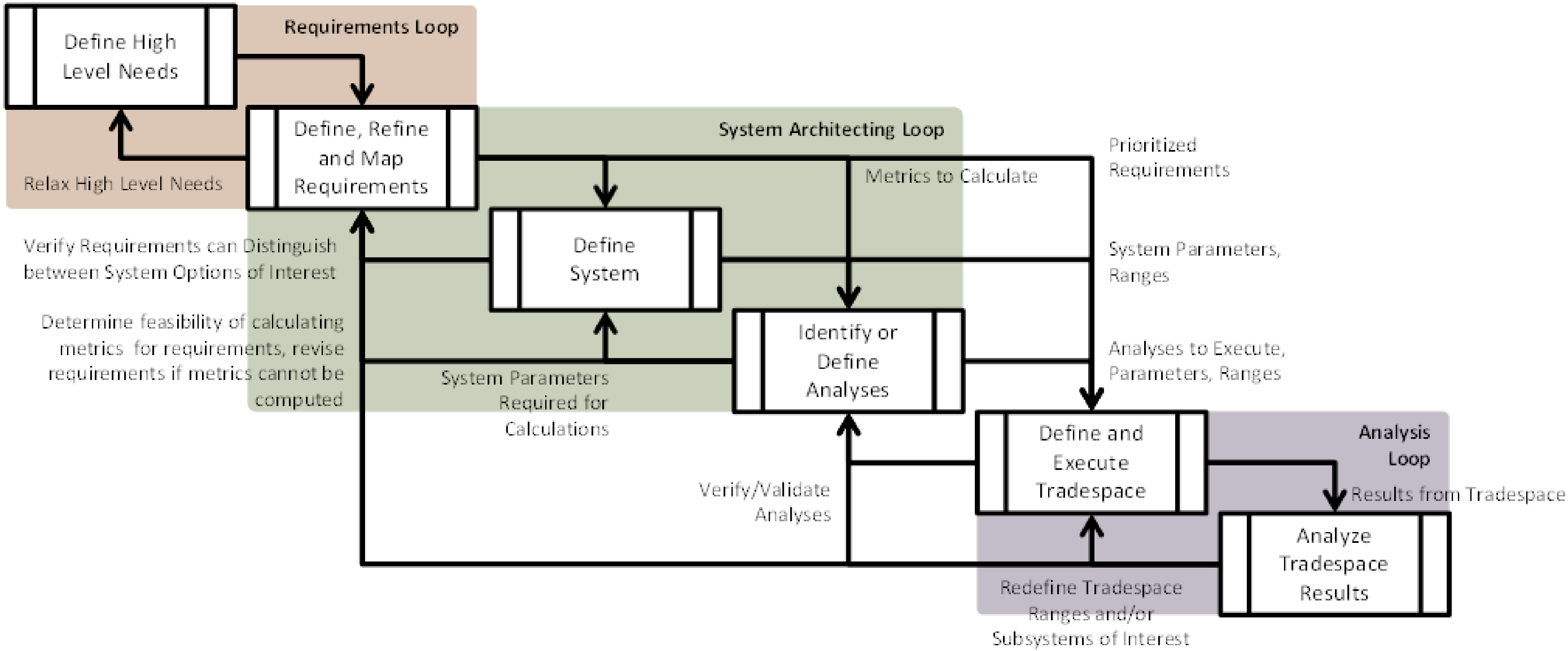
To refine and identify what attributes are needed to characterize a system design and evaluate its MOEs, MOPs, KPPs, etc.

**System Design Variables**

Attributes expressed as value properties that describe specific system design alternatives.

# Networked Workflow through a Design Space Environment

## Generalized Systems Engineering Workflow showing the set of all Systems Engineering Use Cases


Define High Level Needs
Requirements Loop
Define, Refine and Map Requirements
Relax High Level Needs
System Architecting Loop
Metrics to Calculate
Prioritized Requirements
Define System
Verify Requirements can Distinguish between System Options of Interest
System Parameters, Ranges
Identify or Define Analyses
Determine feasibility of calculating metrics for requirements, revise requirements if metrics cannot be computed
System Parameters Required for Calculations
Analyses to Execute, Parameters, Ranges
Define and Execute Tradespace
Analysis Loop
Results from Tradespace
Verify/Validate Analyses
Analyze Tradespace Results
Redefine Tradespace Ranges and/or Subsystems of Interest

**A use case has a specific path through the networked workflow. Driving the tool development with the generalized workflow helps ensure we can meet the requirements of *future* use cases.**

# *Process Steps*

- **Define**

  Users describe the needs, the analyses to assess whether or not the needs are met, and the system(s) being designed to satisfy those needs

- **Execute**

  Users set conditions for and manage/monitor the execution of the integrated engineering models

- **Analyze**

  Users assess the information generated by the execution of the models to improve their mental models of the problem and the system of interest

**MBSE**

Model Based Systems Engineering

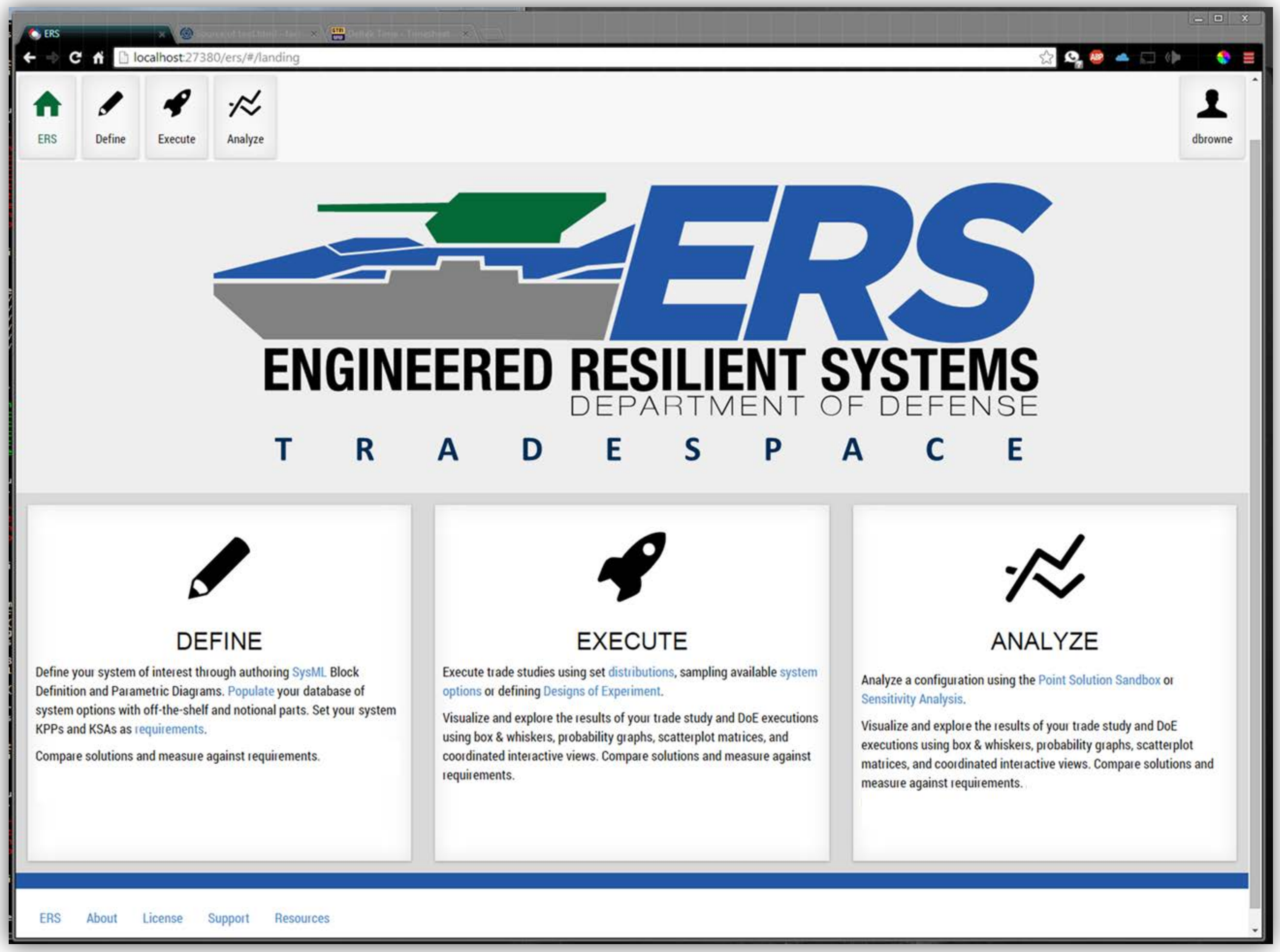
**MDAO**

Multidisciplinary Design Analysis and Optimization

**MCDM**

Multi Criteria Decision Making

# *Web-enabled Collaborative Tradestudies*



DEFINE

Define your system of interest through authoring SysML Block Definition and Parametric Diagrams. Populate your database of system options with off-the-shelf and notional parts. Set your system KPPs and KSAs as requirements.

Compare solutions and measure against requirements.

EXECUTE

Execute trade studies using set distributions, sampling available system options or defining Designs of Experiment.

Visualize and explore the results of your trade study and DoE executions using box & whiskers, probability graphs, scatterplot matrices, and coordinated interactive views. Compare solutions and measure against requirements.

ANALYZE

Analyze a configuration using the Point Solution Sandbox or Sensitivity Analysis.

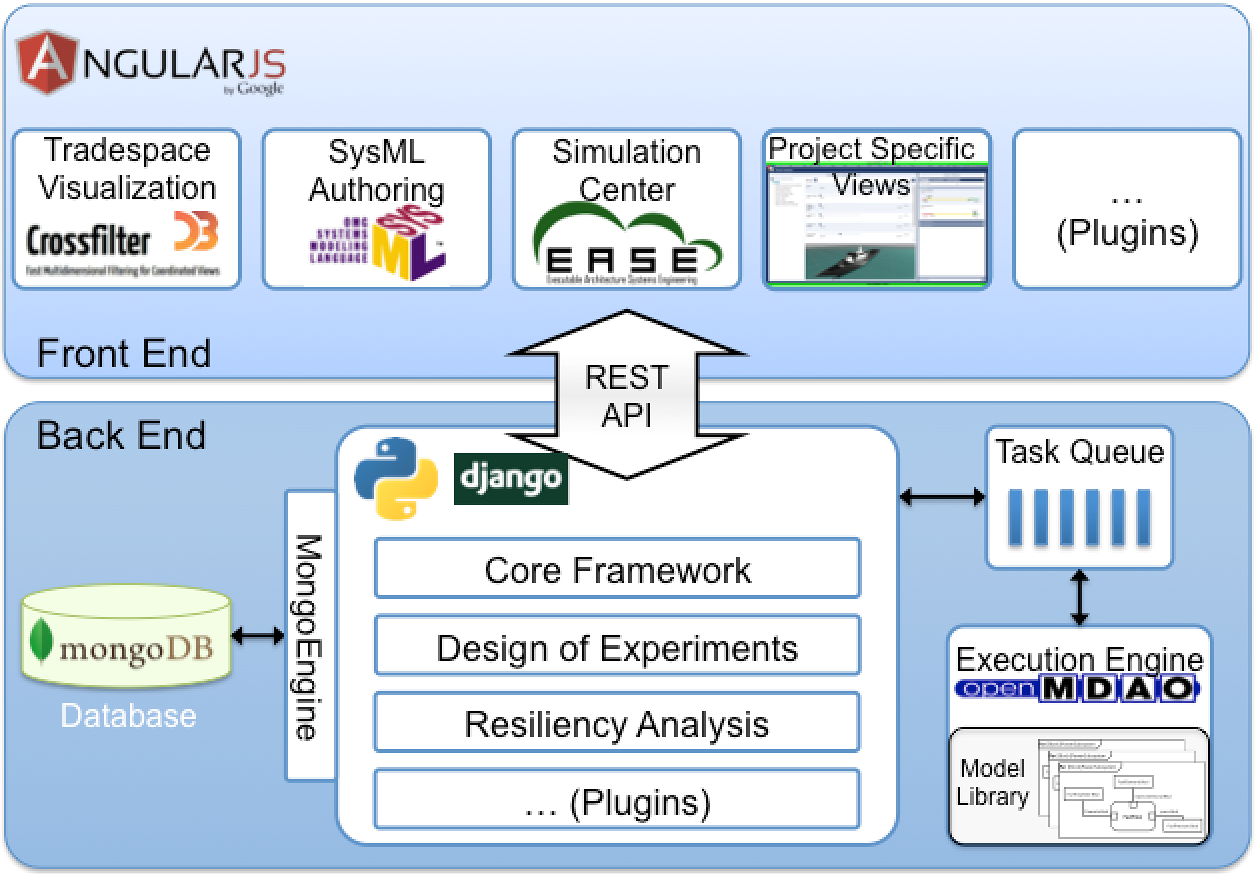
Visualize and explore the results of your trade study and DoE executions using box & whiskers, probability graphs, scatterplot matrices, and coordinated interactive views. Compare solutions and measure against requirements.

**Developed by ERDC and GTRI as part of a larger ERS software program development effort**

# *Software Architecture*

- **Front End**
  - Composed from a collection of Angular Modules and Javascript libraries

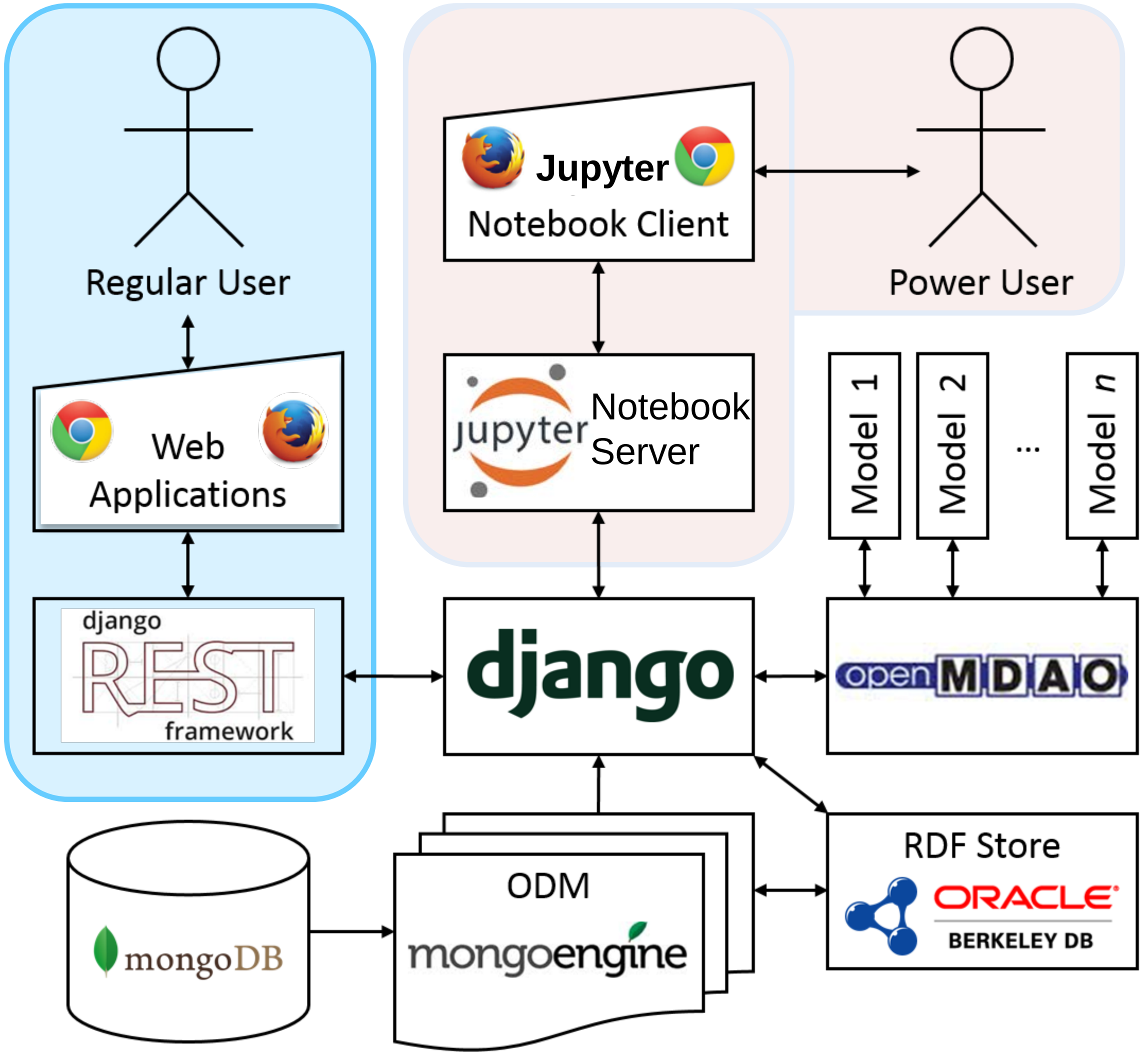
- **Backend**
  - Composed from Django apps and other python libraries
  - Using OpenMDAO to orchestrate the execution of linked constraints



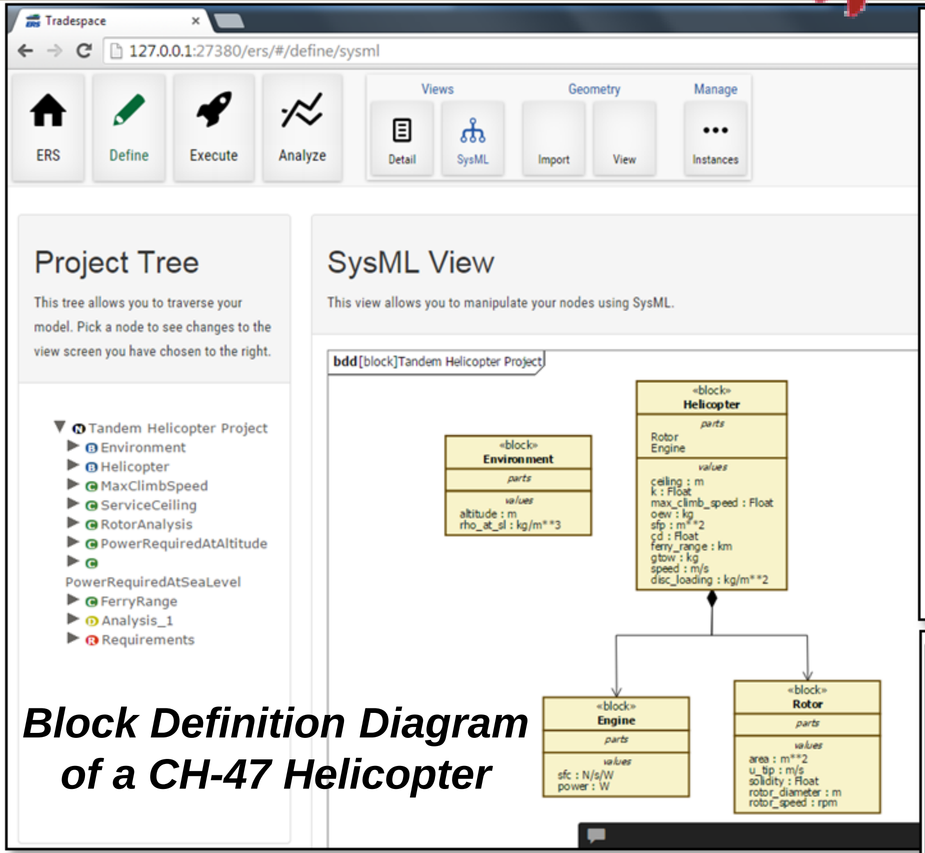
**Modular approach to progressively layer in analysis capabilities and help to make code testable by focusing modules on a particular task**

# *Support Two Disparate Users*

- **Power Users**
  - Are experts that frequently do not need a GUI to quickly build a model
  - Can interact directly with the data through scripting environment (Jupyter notebooks)
- **Regular Users**
  - Are typically consuming views of the data (e.g., SysML diagrams, tradespace analysis visualizations)
  - Can interact via views customized for the type of actions needed



# Collaborative SysML in the Browser



## Project Tree

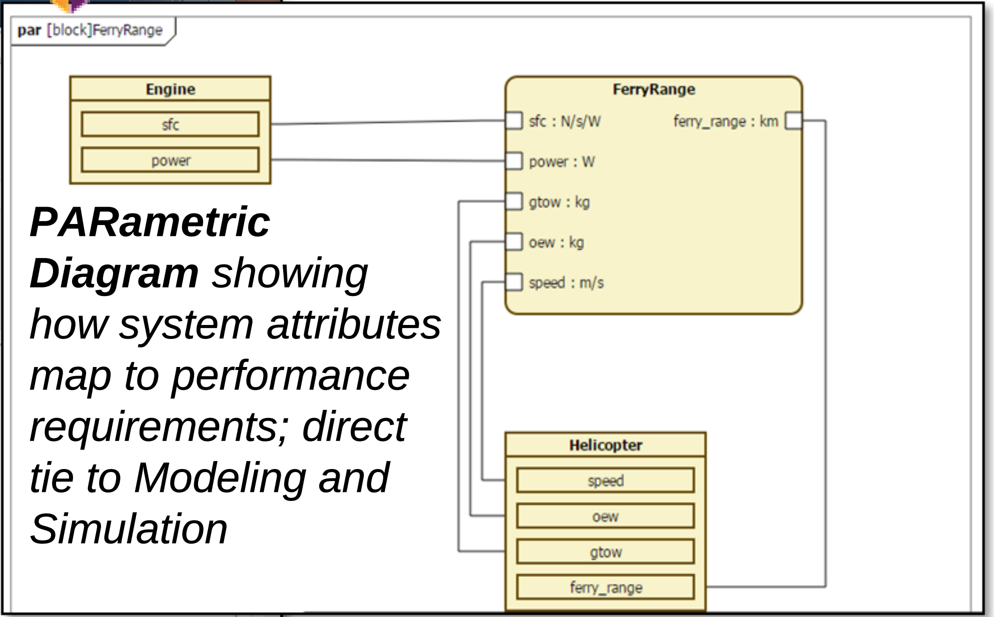
This tree allows you to traverse your model. Pick a node to see changes to the view screen you have chosen to the right.

- Tandem Helicopter Project
  - Environment
  - Helicopter
  - MaxClimbSpeed
  - ServiceCeiling
  - RotorAnalysis
  - PowerRequiredAtAltitude
  - PowerRequiredAtSeaLevel
  - FerryRange
  - Analysis_1
  - Requirements

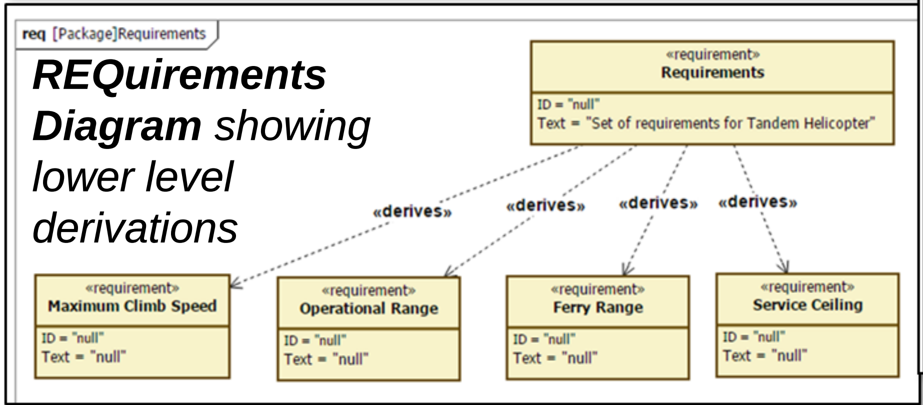
## SysML View

This view allows you to manipulate your nodes using SysML.

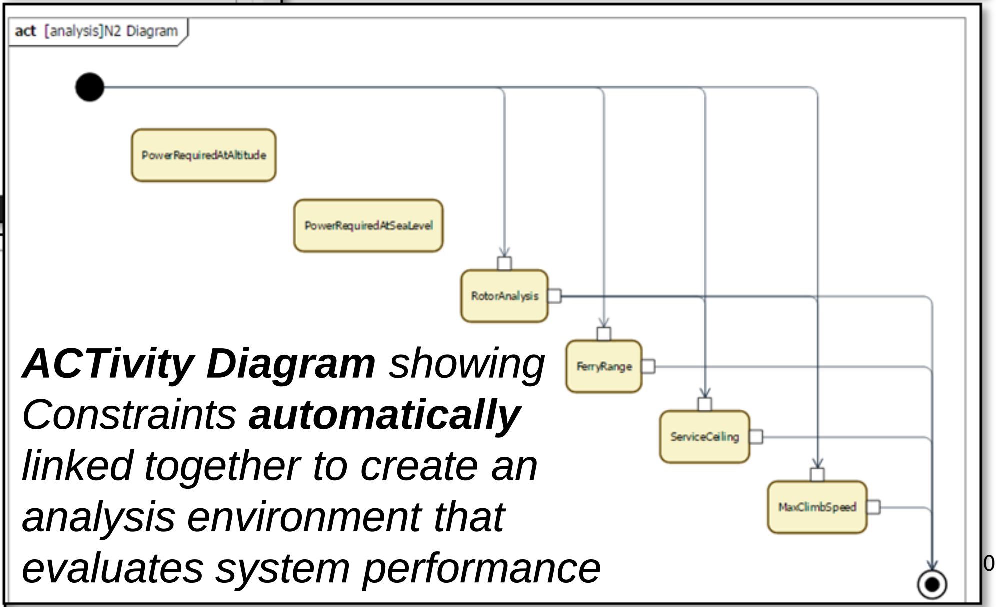
**Block Definition Diagram of a CH-47 Helicopter**



***PARametric Diagram** showing how system attributes map to performance requirements; direct tie to Modeling and Simulation*



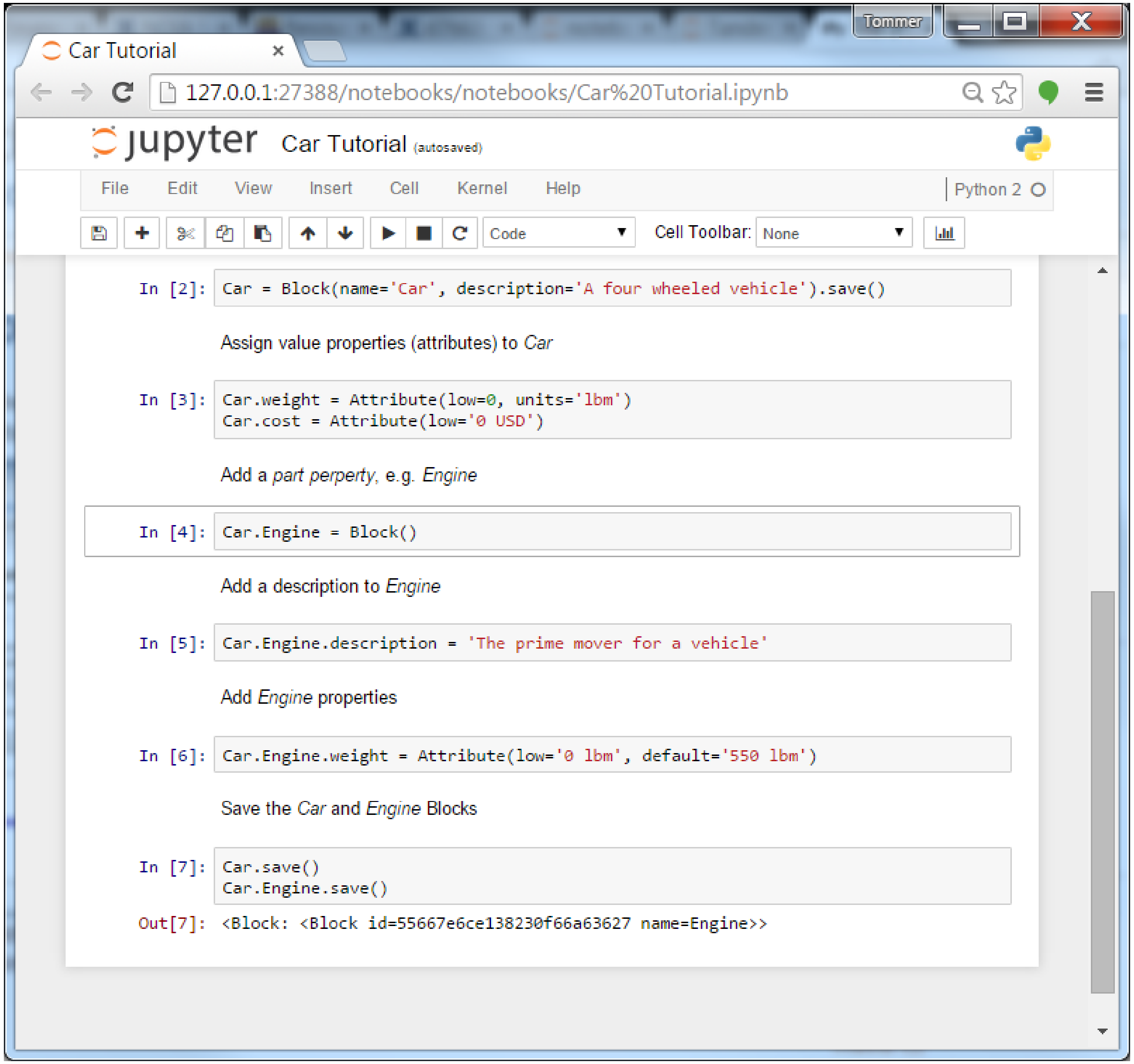
***REQuirements Diagram** showing lower level derivations*



***ACTivity Diagram** showing Constraints **automatically** linked together to create an analysis environment that evaluates system performance*

# For the “Power User”: Jupyter* Notebook Interface

- ***Jupyter* Notebooks*** allow user through the browser to directly execute python code on the back end
- Example declaratively builds up a simple model of a ***Car***
- The *Car* has value properties and includes a *Engine* as a part property



```
In [2]: Car = Block(name='Car', description='A four wheeled vehicle').save()
```

Assign value properties (attributes) to *Car*

```
In [3]: Car.weight = Attribute(low=0, units='lbm')
        Car.cost = Attribute(low='0 USD')
```

Add a *part perperty*, e.g. *Engine*

```
In [4]: Car.Engine = Block()
```

Add a description to *Engine*

```
In [5]: Car.Engine.description = 'The prime mover for a vehicle'
```

Add *Engine* properties

```
In [6]: Car.Engine.weight = Attribute(low='0 lbm', default='550 lbm')
```

Save the *Car* and *Engine* Blocks

```
In [7]: Car.save()
        Car.Engine.save()
Out[7]: <Block: <Block id=55667e6ce138230f66a63627 name=Engine>>
```

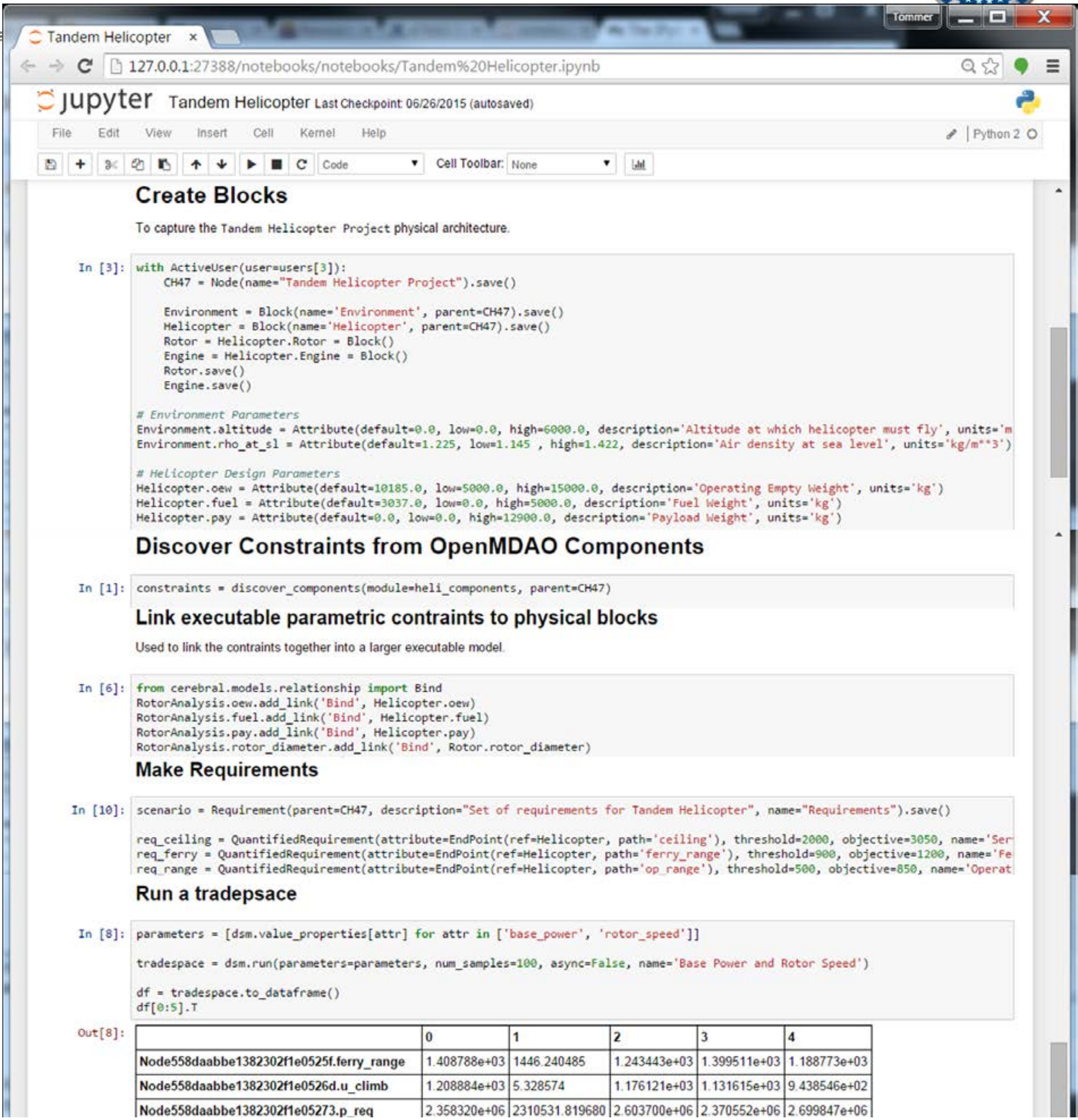
*formerly iPython Notebook

# *For the “Power User”: Jupyter* Notebook Interface*

- ***Jupyter* Notebooks*** allow user through the browser to directly execute python code on the back end

- **Tandem helicopter** example shows how SysML block are created

- Tradespace can be executed directly



Tandem Helicopter
Tommer
127.0.0.1:27388/notebooks/notebooks/Tandem%20Helicopter.ipynb
jupyter
Tandem Helicopter Last Checkpoint 06/26/2015 (autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Python 2
Code
Cell Toolbar:
None

### Create Blocks

To capture the `Tandem Helicopter Project` physical architecture.

```
In [3]: with ActiveUser(user=users[3]):
            CH47 = Node(name="Tandem Helicopter Project").save()

            Environment = Block(name='Environment', parent=CH47).save()
            Helicopter = Block(name='Helicopter', parent=CH47).save()
            Rotor = Helicopter.Rotor = Block()
            Engine = Helicopter.Engine = Block()
            Rotor.save()
            Engine.save()

        # Environment Parameters
        Environment.altitude = Attribute(default=0.0, low=0.0, high=6000.0, description='Altitude at which helicopter must fly', units='m
        Environment.rho_at_sl = Attribute(default=1.225, low=1.145 , high=1.422, description='Air density at sea level', units='kg/m**3')

        # Helicopter Design Parameters
        Helicopter.oew = Attribute(default=10185.0, low=5000.0, high=15000.0, description='Operating Empty Weight', units='kg')
        Helicopter.fuel = Attribute(default=3037.0, low=0.0, high=5000.0, description='Fuel Weight', units='kg')
        Helicopter.pay = Attribute(default=0.0, low=0.0, high=12900.0, description='Payload Weight', units='kg')
```

### Discover Constraints from OpenMDAO Components

```
In [1]: constraints = discover_components(module=heli_components, parent=CH47)
```

### Link executable parametric contraints to physical blocks

Used to link the contraints together into a larger executable model.

```
In [6]: from cerebral.models.relationship import Bind
        RotorAnalysis.oew.add_link('Bind', Helicopter.oew)
        RotorAnalysis.fuel.add_link('Bind', Helicopter.fuel)
        RotorAnalysis.pay.add_link('Bind', Helicopter.pay)
        RotorAnalysis.rotor_diameter.add_link('Bind', Rotor.rotor_diameter)
```

### Make Requirements

```
In [10]: scenario = Requirement(parent=CH47, description="Set of requirements for Tandem Helicopter", name="Requirements").save()

         req_ceiling = QuantifiedRequirement(attribute=EndPoint(ref=Helicopter, path='ceiling'), threshold=2000, objective=3050, name='Ser
         req_ferry = QuantifiedRequirement(attribute=EndPoint(ref=Helicopter, path='ferry_range'), threshold=900, objective=1200, name='Fe
         req_range = QuantifiedRequirement(attribute=EndPoint(ref=Helicopter, path='op_range'), threshold=500, objective=850, name='Operat
```

### Run a tradepsace

```
In [8]: parameters = [dsm.value_properties[attr] for attr in ['base_power', 'rotor_speed']]

        tradespace = dsm.run(parameters=parameters, num_samples=100, async=False, name='Base Power and Rotor Speed')

        df = tradespace.to_dataframe()
        df[0:5].T
```

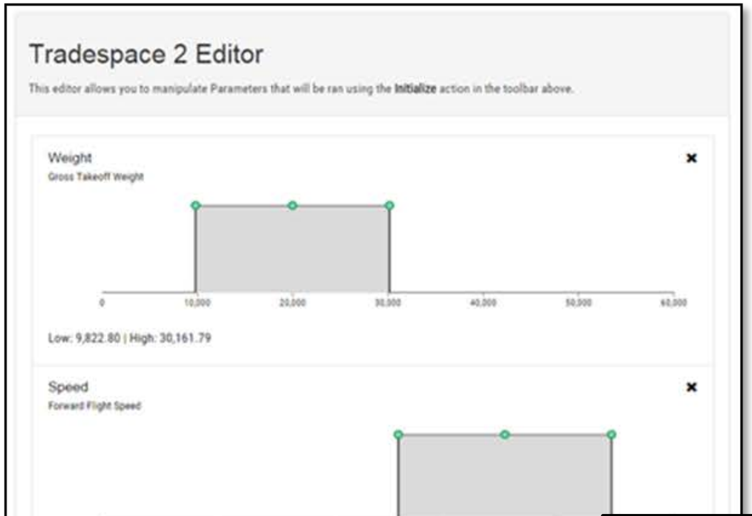
Out[8]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **Node558daabbe1382302f1e0525f.ferry_range** | 1.408788e+03 | 1446.240485 | 1.243443e+03 | 1.399511e+03 | 1.188773e+03 |
| **Node558daabbe1382302f1e0526d.u_climb** | 1.208884e+03 | 5.328574 | 1.176121e+03 | 1.131615e+03 | 9.438546e+02 |
| **Node558daabbe1382302f1e05273.p_req** | 2.358320e+06 | 2310531.819680 | 2.603700e+06 | 2.370552e+06 | 2.699847e+06 |

*formerly iPython Notebook

# *Tradespace Execution and Visualization*

**A tradespace can be generated by varying the independent parameters in the analysis**



Tradespace 2 Editor

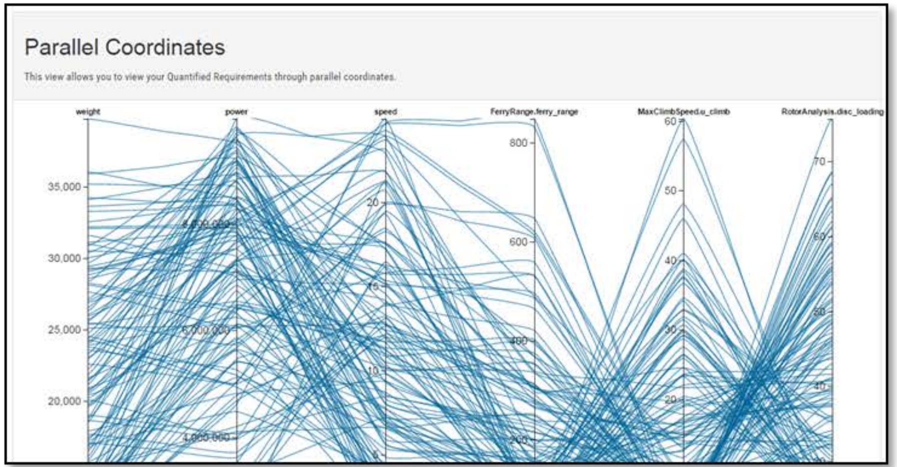
This editor allows you to manipulate Parameters that will be ran using the **Initialize** action in the toolbar above.

Weight
Gross Takeoff Weight

Low: 9,822.80 | High: 30,161.79

Speed
Forward Flight Speed

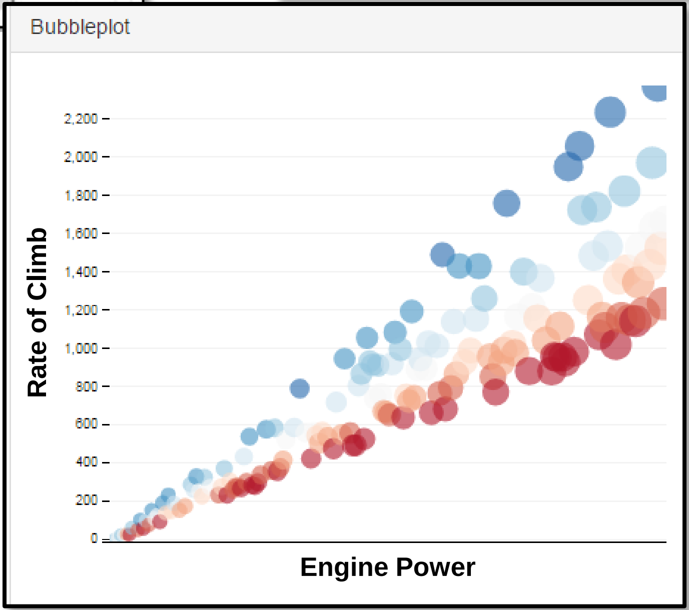
**The results from a tradespace can then be evaluated using an interactive visualization (in this case a *brushable* parallel coordinates plot)**



Parallel Coordinates

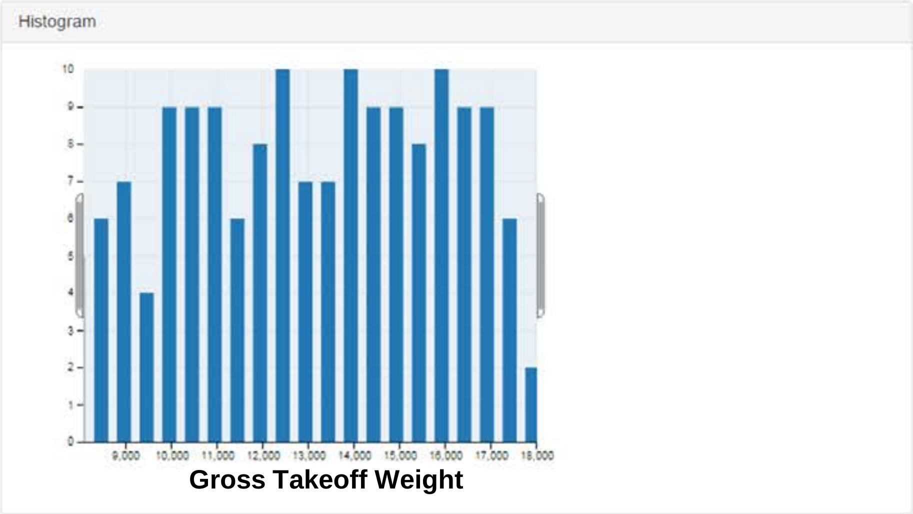
This view allows you to view your Quantified Requirements through parallel coordinates.

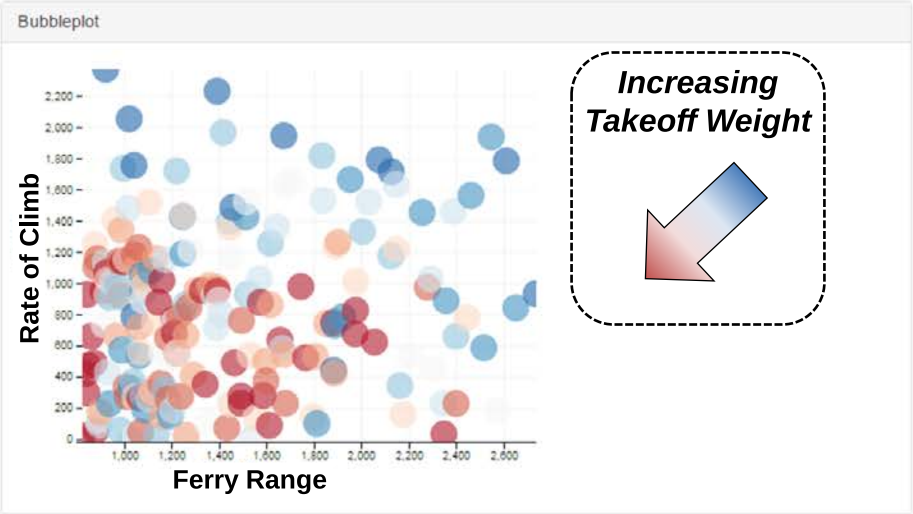
**Scatter/Bubble plots can be used to show multidimensional trades**



***Increasing Payload***

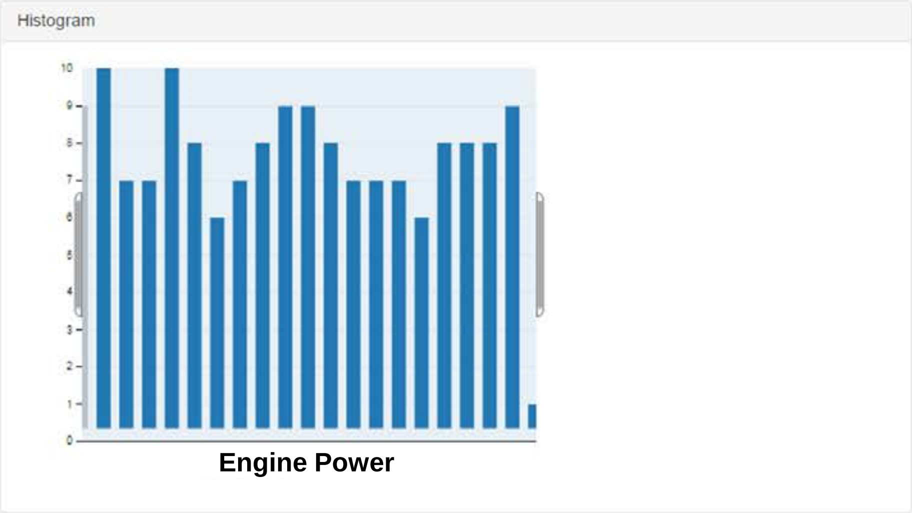
# *Dynamic Analysis*

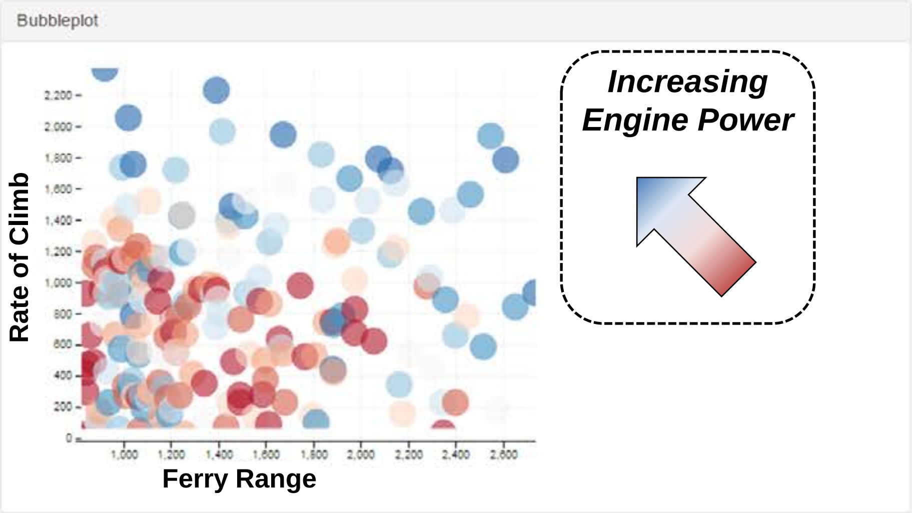




- Able to select regions of interest within a tradespace
- Note that as ***Takeoff Weight*** is increased, a tradeoff emerges between ***Rate of Climb*** and ***Ferry Range***

# *Dynamic Analysis*



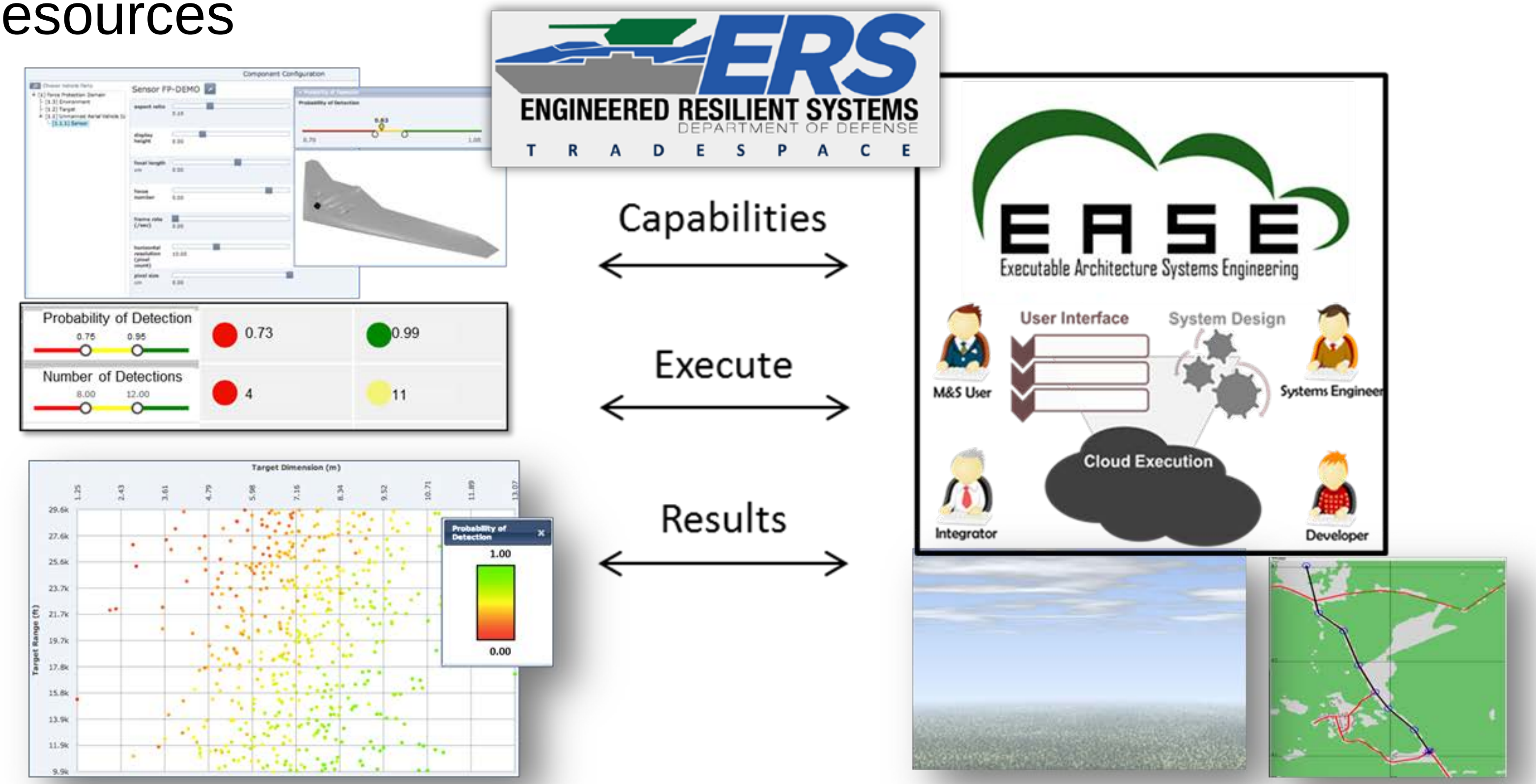


***Increasing Engine Power***

- Able to select regions of interest within a tradespace
- Note that as ***Engine Power*** is increased, a tradeoff emerges between ***Rate of Climb*** and ***Ferry Range***

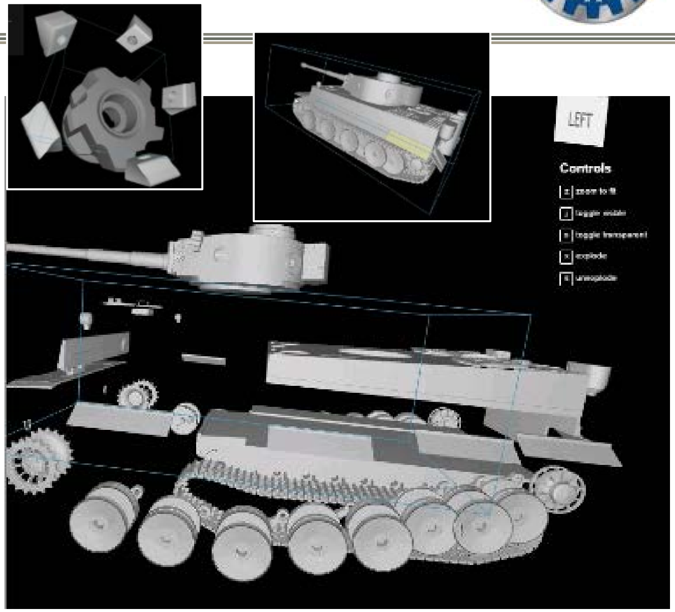
# *Interfacing Tradestudies with Simulation Operational Scenarios*

- **Executable Architecture Systems Engineering (EASE)**
  - Links analytical, experimental and training objectives with Modeling and Simulation
  - Explore **operational aspects** of the analytical questions in simulation
- ERS effort develops interface between MBSE/Tradestudies and Army Research Lab investment in executable, cloud-computing resources



# *Next Steps*

- Extend a limited "**CAD in the browser**" capability
- Integration with **High Performance Computing** assets at DoD HPC Centers
- Application to **DoD acquisition programs**



http://itl.erdc.usace.army.mil/featurecenter

# *Parting Thoughts…*

- Tradespace exploration supports DoD leadership by helping identify the impacts of decisions across a system's acquisition lifecycle
- Critical program decisions are often made based on the outcomes of trades defined by multiple types and quantities of data and information
- Tradespace exploration for ERS is grounded on big data and information analyzed and presented in a holistic view
- Trades data and information must present the perspectives of multiple decision makers across numerous time steps
- ERS tradespace exploration is using a process to identify requirements and attributes that define appropriate trades

# *Acknowledgements*

The authors wish to thank the following for their valuable contributions to the content of this work

Georgia Tech Research Institute

- Dr. Valerie Sitterle
- Dr. Santiago Balestrini-Robinson
- Daniel Browne
- Dr. Dane Freeman
- L. Drew Pihera

ERDC Engineer Research & Development Center

- Dr. Owen Eslinger
- Alex Baylot
- Daniel Chausse
- David Stuart
- Dr. Andrew Strelzoff

**Portions of this material are based upon work supported, in whole or in part, by the United States DoD through the Systems Engineering Research Center (SERC) under Contract HQ0034-13-D-0004. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. The views and conclusions are those of the individual authors and participants, and should not be interpreted as necessarily representing official policies, either expressed or implied, of the DoD, any specific US Government agency, or the US Government in general.**