



IEEE Aerospace Conference 2019

# Assuring Correctness, Completeness, and Performance for Model-Based Fault Diagnosis Systems

Allen Nikora, Priyanka Srivastava, Ksenia  
Kolcio, Maurice Prather, Lorraine Fesq, Seung  
Chung

Presented By: Allen Nikora  
Software Assurance



**Jet Propulsion Laboratory**  
California Institute of Technology

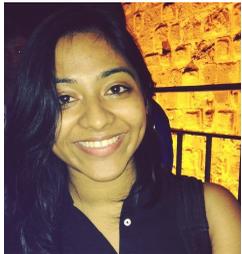
# Who We Are



**Dr. Allen Nikora – Jet Propulsion Laboratory, California Institute of Technology**  
• *Software Reliability Engineer*



**Dr. Seung Chung – Jet Propulsion Laboratory, California Institute of Technology**  
• *PI, manager*



**Priyanka Srivastava – Jet Propulsion Laboratory, California Institute of Technology**  
• *System V&V Engineer*



**Dr. Lorraine Fesq – Jet Propulsion Laboratory, California Institute of Technology**  
• *MBFD Lead*



**Dr. Ksenia Kolcio – Okean Solutions**  
• *Consultant, MBFD supplier*



**Maurice Prather – Okean Solutions**  
• *Consultant, MBFD supplier*

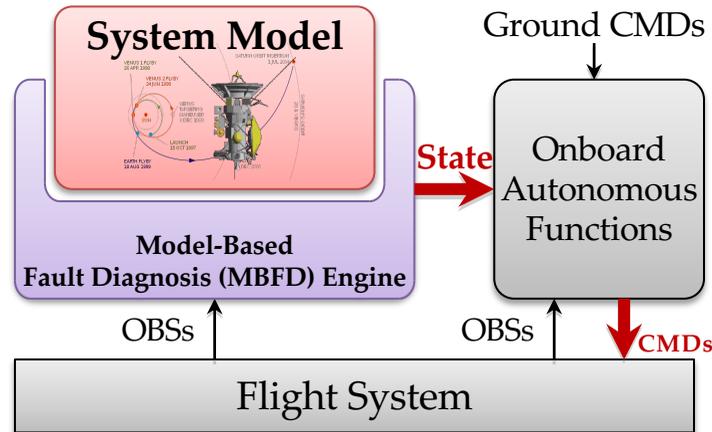
# Assuring Correctness, Completeness, and Performance for Model-Based Fault Diagnosis Systems

IEEE Aerospace Conference 2019

- **Problem Statement**
- **Assurance of Correctness, Completeness, and Performance for MBFD**
  - **Goals**
  - **Approach**
    - Model Correctness and Completeness
    - MBFD Performance (runtime, memory)
    - Static Source Code Analysis and Code Coverage
  - **Results**
    - Model Correctness and Completeness
    - MBFD Performance (runtime, memory)
    - Static Source Code Analysis and Code Coverage
- **Future Work**

# Assurance of Model-Based Fault Diagnosis

## The Need for Reliable Onboard Model-Based Fault Diagnosis

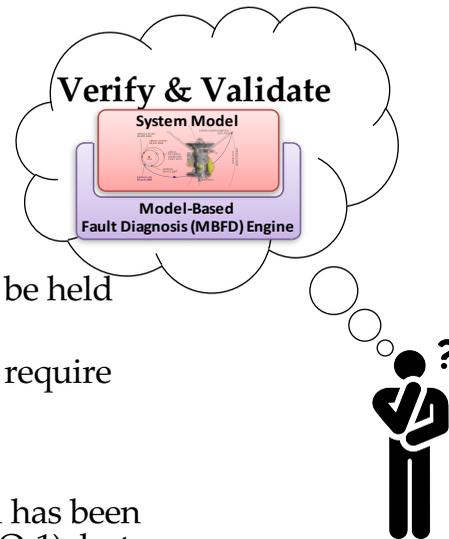


### Our Objective:

- To enable on-board autonomy through model-based fault diagnosis (MBFD) that can continuously verify correct hardware behavior in addition to diagnosing symptoms to estimate the health state. An onboard autonomy capability can then use the determined health states to decide how to react.

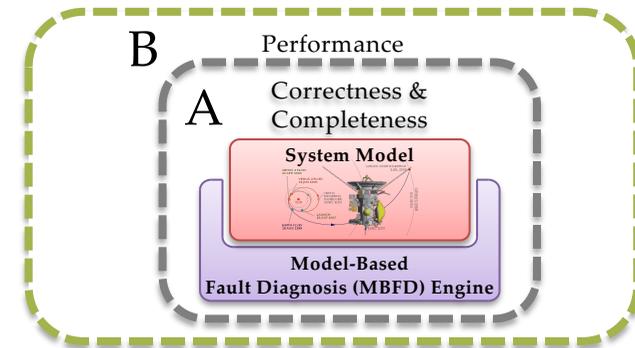
### The Problem:

- Without proper V&V of MBFD...
  - The decisions made by other autonomous systems will be held in **low confidence**
  - The decisions made by other autonomous systems will require review by spacecraft operators
  - The value of onboard autonomy will be diminished
- MBFD has been developed through decades of research and has been demonstrated only twice on flight missions (i.e., DS-1 and EO-1), but
- The techniques for adequately verifying and validating MBFD technologies are not well understood



### Our Solution:

- Establish a concrete methodology to Verify and Validate MBFD technology
- Select an Onboard System to be Modelled
- Apply the V&V Methodology to the chosen system model
- Analyze the V&V test results to ensure the proper functioning of MBFD on the chosen system model



The SMAP GNC Model allowed realistic MBFD and has improved confidence in our V&V approach.

# Assurance of Correctness, Completeness, and Performance for MBFD

## Model Correctness/Completeness Definitions, Performance Assessment

- **Model Completeness and Correctness**
  - Previous work developed operational definitions of model correctness and completeness.
    - Informal – quantitative reasoning about extent of correctness/completeness is difficult.
  - Develop formal definitions of correctness/completeness.
    - Serve as framework to which operational definition can be attached.
    - Facilitate quantitative reasoning.
    - Support development of correctness properties/invariants when applying formal verification methods (e.g., model checking)

# Assurance of Correctness, Completeness, and Performance for MBFD

## Performance Assessment, Definitions of Model Correctness/Completeness (cont'd)

- **Performance Assessment**
  - Performance characteristics highly relevant to flight systems
    - Runtime
    - Memory usage
  - Question to be answered – does the diagnostic engine have good enough performance when querying useful diagnostic models on a flight system?
  - Experimental performance evaluation
    - Performance is related to diagnostic engine and model characteristics
    - Identified model characteristics likely to influence performance
    - Constructed model families based on those characteristics
    - Designed experiments to run instances of model families
  - Analytical performance evaluation
    - Future modification of diagnostic engine/models may decrease accuracy of empirical measurements.
    - Conducted analysis of how engine and model performance vary with observable structural characteristics of the engine and model.
      - number of components.
      - number of inputs.
      - outputs per component.

# Assurance of Correctness, Completeness, and Performance for MBFD Approach – Formal Definition of Correctness and Completeness

- **System and Model Representations**

- Systems and models of systems characterized as sets of logical sentences<sup>1</sup>.
  - System definition (SD) specifying expected behavior of the system or model.
  - Components (COMPONENT) from which system/model is constructed along with their structural properties.
  - Observations (OBS) made of that system or model.
- System and model representations.
  - System:  $S = \{SD_S \cup COMPONENTS_S \cup OBS_S\}$
  - Model:  $M = \{SD_M \cup COMPONENTS_M \cup OBS_M\}$
  - Both S and M are consistent
    - *theorems* of a set G of formulas are those formulas that can be obtained from  $G \cup L$ , where L is a set of axioms, through the finite number of applications of a rule of inference.
    - G is consistent if there is no formula  $\phi$  in the theorems of G such that both  $\phi$  and  $\neg\phi$  can be deduced from G.

1. R. Reiter, "A Theory of Diagnosis from First Principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57-95, 1987.

# Assurance of Correctness, Completeness, and Performance for MBFD Approach – Formal Definition of Correctness and Completeness (cont'd)

## Definition of Model Correctness

- Looks at model correctness in terms of theorems that can be inferred from a set of logical sentences and the preservation of consistency.
- **Definitions**
  - $S = \{SD_S \cup COMPONENTS_S \cup OBS_S\}$ .
  - $M = \{SD_m \cup COMPONENTS_M \cup OBS_M\}$ .
- **Axioms ( $\Lambda$ )**
  - Describe how system's measurable quantities are represented in the model (e.g., discretization of real-valued quantities).
  - Describe relationships between system components and model components.
  - Describe relevant physical laws (e.g., rocket equation for propulsion systems).

# Assurance of Correctness, Completeness, and Performance for MBFD Approach – Formal Definition of Correctness and Completeness (cont'd)

## Definition of Model Correctness (cont'd)

- **Relationship between theorems of S and M**
  - Using axioms  $\Lambda$ , deduce set of theorems  $\mathcal{G}$ .
  - If M is a non-null subset of  $\mathcal{G}$ , and if M is consistent, we can say that the model is consistent with the system.

## Definition of Model Completeness

- **Definitions**
  - $S = \{SD_S \cup COMPONENTS_S \cup OBS_S\}$ .
  - $M = \{SD_m \cup COMPONENTS_M \cup OBS_M\}$ .
  - $m \subseteq M$
- **Relationship between S and M**
  - $m \subseteq M \mid m \models S$
  - If any subset m of model M satisfies the entire set of logical sentences of system S, then M is complete.
  - If  $m \subset M$ ,  $m \neq M$ , and  $m \models S$  implies elements in M may not have been specified in S.

# Assurance of Correctness, Completeness, and Performance for MBFD Approach – Performance Assessment

## Experimental Performance Evaluation

- **Goal – Determine how runtime/memory performance of MONSID varies with structural characteristics of models**
- **Approach**
  - Identify factors likely to affect performance.
    - Number of nodes
    - Number of components
    - Number of connections
  - Identify/acquire performance profiling tools (gprof).
  - Design experiments, construct diagnostic models
  - Conduct experiments
- **Model Families – see next slide for general model form**

### Family A

- Single layer input stage: 1, 2, 4, or 8 inputs per component. 2 components for input layer.
- Common processing stage: 2-in-1-out components, 2 components per layer.
- Output stage: 1 component merging outputs of processing stage.

### Family B

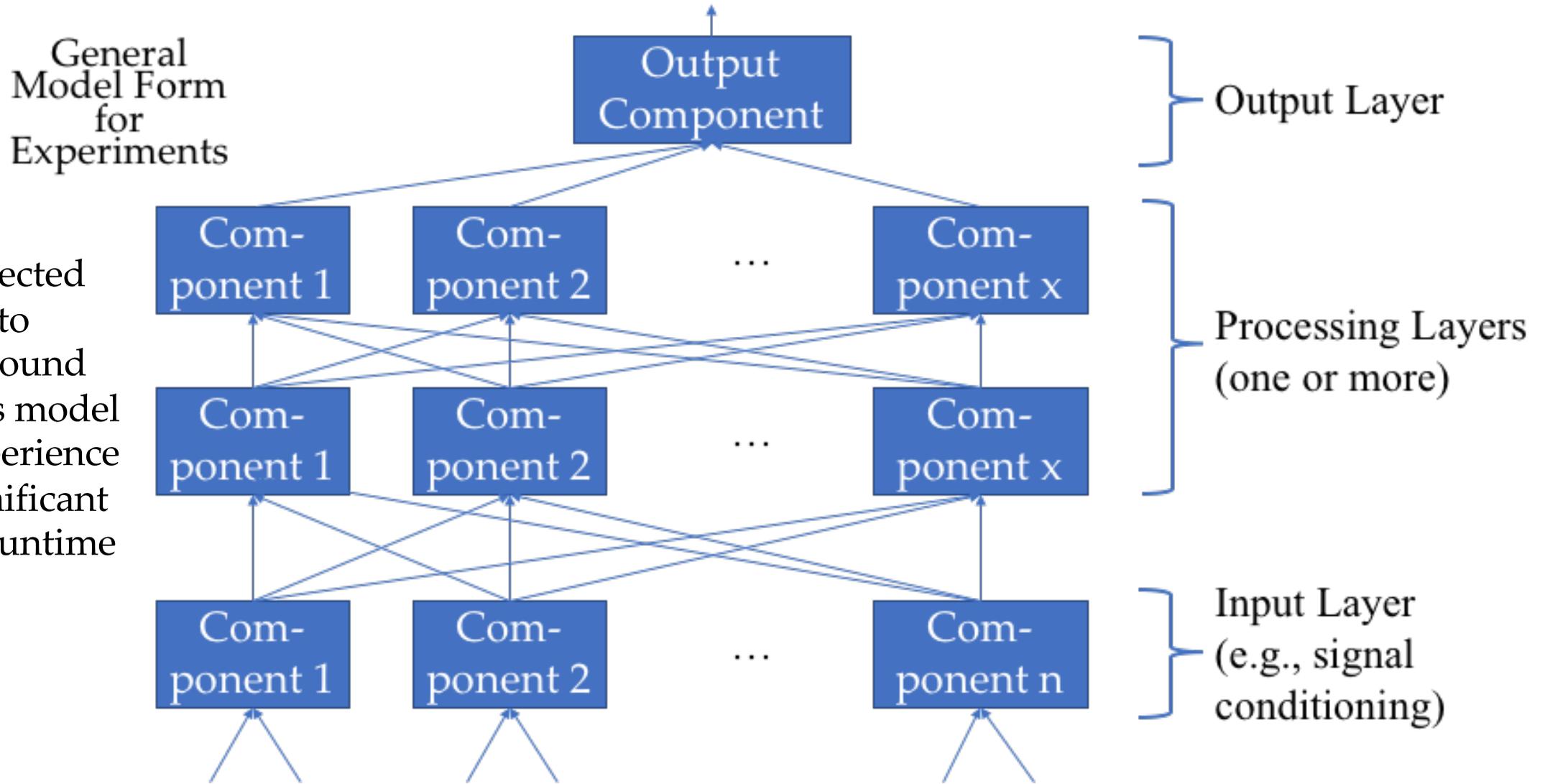
- Single layer input stage: 1, 2, or 4 inputs per component. 4 components for input layer.
- Common processing stage: 4-in-1-out components, 4 components per layer.
- Output stage: 1 component merging outputs of processing stage.

### Family C

- Single layer input stage: 1, 2, or 4 inputs per component. 8 components for input layer.
- Common processing stage: 8-in-1-out components, 8 components per layer.
- Output stage: 1 component merging outputs of processing stage.

# Assurance of Correctness, Completeness, and Performance for MBFD Approach – Performance Assessment (cont'd)

Highly interconnected structure chosen to establish upper bound on what previous model development experience indicates is a significant driver of model runtime performance.



# Assurance of Correctness, Completeness, and Performance for MBFD Approach – Performance Assessment (cont'd)

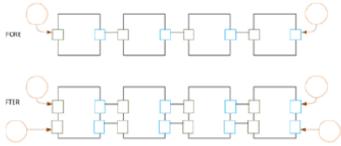
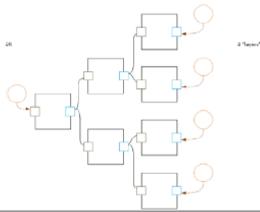
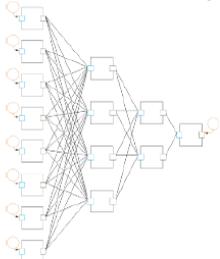
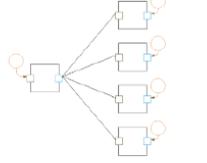
## Analytical Performance Evaluation

- **Goal – Complement experimental performance evaluation. Preserve understanding of diagnostic engine's characteristics if existing diagnostic model(s) are modified or developed anew for future missions.**
- **Approach**
  - Used three methodologies to determine the big O for MONSID's main method
    - Loop counting.
    - Comparisons to graph theory.
    - Empirical validation – used to validate analyses.
  - **Empirical validation**
    - Coupled nature of MONSID elements makes it difficult to clearly extract immediate impact of any element type as they are not independent. However,
    - Able to define topology families that reduced coupling. See next slide.
      - Serial
      - Fan
      - Fan mesh
      - Divergent
    - General parameters for the empirical tests
      - Average time to process nominal data with data sets up to 250,000 rows.
      - Increasing model size up to 10,000 components per the family type.
      - Use of a standard test component in all models.

# Assurance of Correctness, Completeness, and Performance for MBFD Approach – Performance Assessment (cont'd)

## Analytical Performance Evaluation (cont'd)

## Model Family Structure for Analytical Evaluation

Family	Description	Characteristics
<b>Serial</b>		<ul style="list-style-type: none"> <li>As Component count increases, Node and Connections increase in linear fashion.</li> <li>Allows adding additional “channels” to test influence of Nodes and Connections without impacting Component count.</li> <li>Sensor count unchanged as Components are added.</li> </ul>
<b>Fan</b>	<p>Each Component is connected to 2 Components</p> 	<ul style="list-style-type: none"> <li>Component count is doubled with each “layer”.</li> <li>Output nodes can be added without increasing the number of connections (Fan-d)</li> <li>Allows testing of impact of increased Nodes without increasing Connections or Components.</li> </ul>
<b>Fan mesh</b>	<p>Each Component is connected to all downstream Components</p> 	<ul style="list-style-type: none"> <li>Component count is doubled with each “layer”</li> <li>High dependency count as each Component depends on receiving data from all Components in the upstream layer (Forward or Reverse propagation).</li> <li>Any increase in “layer” depth moderately increases Components and Nodes but the Connection count increases significantly.</li> </ul>
<b>Divergent</b>	<p>Simplified 2-layer subset of Fan Mesh</p> 	<ul style="list-style-type: none"> <li>Only 2 “layers”</li> <li>Includes as many Sensors as there are Components</li> <li>Increase in Components results in increase in Sensor, Nodes, and Connections.</li> <li>Allows easy comparison of Forward and Reverse propagation costs</li> </ul>

# Assurance of Correctness, Completeness, and Performance for MBFD

## Approach – Static Source Code Analysis and Code Coverage

### Static Source Code Analysis

- **Goal**
  - Identify errors that desk review of source code may not find, and
  - Ensure that users of MONSID can comply with JPL requirement to apply static source code analysis.
- **Analysis**
  - Analyzed version 2 of MONSID provided by Okean Solutions.
  - Used Semmle’s “Semmle Core” tool (formerly Odasa).
    - Available through JPL’s Computer Aided Engineering (CAE) group.

# Assurance of Correctness, Completeness, and Performance for MBFD

## Approach – Static Source Code Analysis and Code Coverage

### Code Coverage

- Goals
  - Gain additional confidence that testing is improving software quality during development/maintenance
  - Ensure that future projects using MONSID would be able to meet JPL's institutional requirements that software development efforts measure test coverage.
- Applied multiple tools
  - Visual Studio for C# and C++ versions (Windows)
  - gcov tool family for C++ (Linux)
- What was measured?
  - C#                    15 Full Engine tests, 65 Object Model tests
  - C++/Linux        15 Full Engine tests. Object Model tests for C++ under Windows, but not Linux.

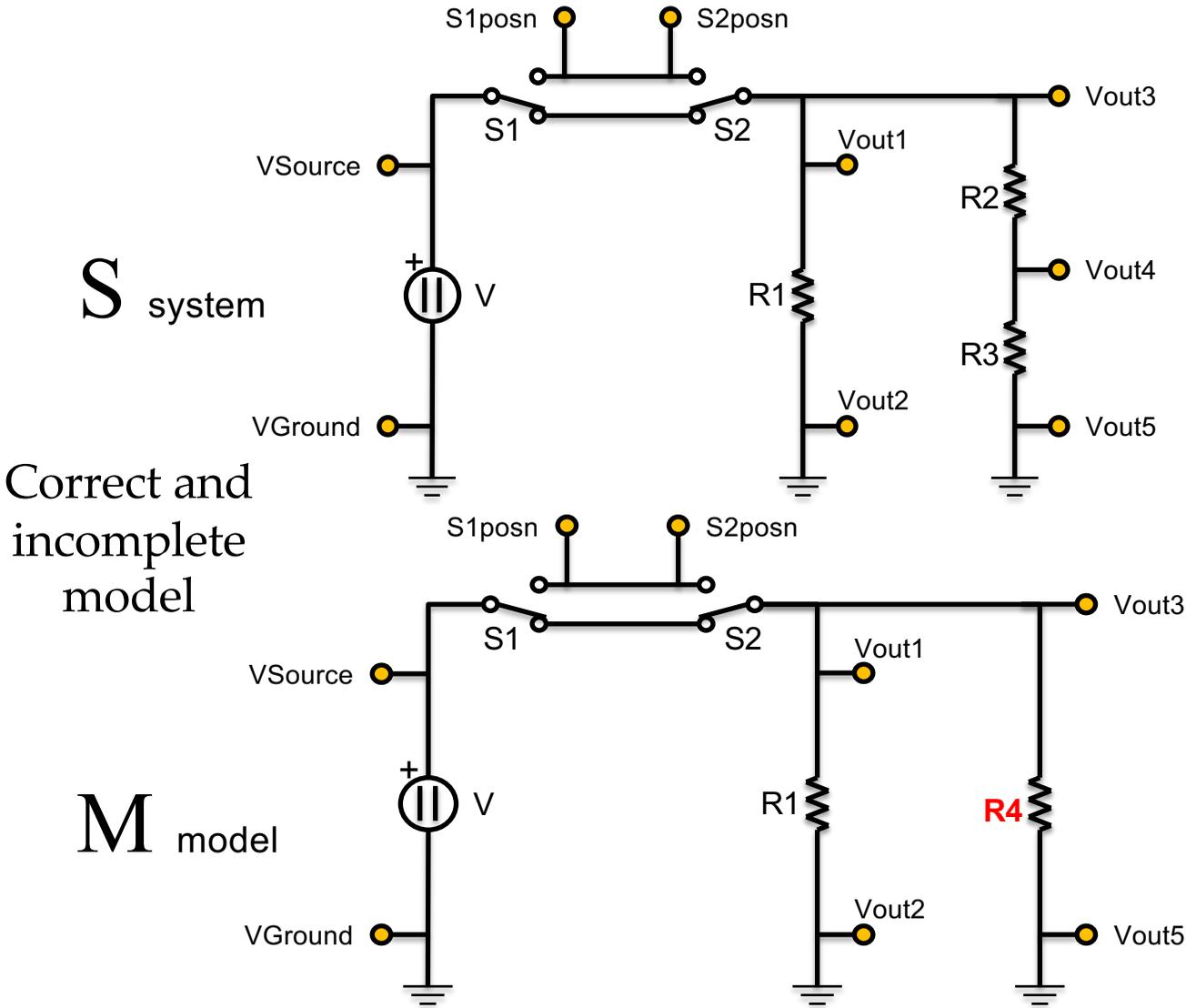
# Assurance of Correctness, Completeness, and Performance for MBFD

## Results

- **Model Correctness and Completeness**
  - Examples on following slides illustrate how correctness and completeness definitions can be used.
    - Correct and incomplete model.
    - Incorrect and complete model.
- **Experimental and Analytical Performance Assessment**
  - Experimental Assessment.
  - Analytical Assessment.
- **Static Source Code Analysis and Code Coverage**

# Assurance of Correctness, Completeness, and Performance for MBFD

## Model Correctness and Completeness Results



$$S = \{SD_S \cup COMPONENTS_S \cup OBS_S\}$$

$SD_S$

$in(S1, 1) = out(V, 1)$   
 $out(S2, 1) = out(S1, 1)$   
 $out(S2, 2) = out(S1, 2)$   
 $in(S2, 1) = in(R1, 1)$   
 $in(S2, 1) = in(R2, 1)$   
 $in(R3, 1) = out(R2, 1)$   
 $out(R1, 1) = GROUND$   
 $out(R3, 1) = GROUND$   
 $in(V, 1) = GROUND$

$COMPONENTS_S$

Components = {V, R1, R2, R3, S1, S2}  
 RESISTOR(R1)  
 RESISTOR(R2)  
 RESISTOR(R3)  
 SPDT\_SWITCH(S1)  
 SPDT\_SWITCH(S2)  
 VOLTAGE\_SOURCE(V)

$COMPONENTS_M$

Components = {V, R1, R4, S1, S2}  
 RESISTOR(R1)  
 RESISTOR(R4)  
 SPDT\_SWITCH(S1)  
 SPDT\_SWITCH(S2)  
 VOLTAGE\_SOURCE(V)

$OBS_S$

RESISTANCE(R1) = 1500  
 RESISTANCE(R2) = 1000  
 RESISTANCE(R3) = 3214.97  
 VGround = 0  
 VSource = 5

$A$  is a set of axioms

$RESISTOR(x) \rightarrow RESISTANCE(x) \in \mathbb{R} \wedge RESISTANCE(x) > 0$   
 $RESISTOR(x) \wedge RESISTOR(y) \rightarrow RESISTOR\_SERIES(x,y) = (in(y, 1) = out(x, 1))$   
 $RESISTOR(x) \wedge RESISTOR(y) \rightarrow RESISTOR\_PARALLEL(x,y) = (in(x, 1) = in(y, 1)) \wedge (out(x, 1) = out(y, 1))$   
 $RESISTOR(x) \wedge RESISTOR(y) \rightarrow SERIES\_RESISTANCE(x, y) = RESISTANCE(x) + RESISTANCE(y)$   
 $RESISTOR(x) \wedge RESISTOR(y) \rightarrow PARALLEL\_RESISTANCE(x, y) = RESISTANCE(x) * RESISTANCE(y) / (RESISTANCE(x) + RESISTANCE(y))$   
 $RESISTOR(x) \wedge RESISTOR(y) \wedge RESISTOR\_SERIES(x,y) \wedge VOLTAGE(in(x, 1)) = V1 \wedge VOLTAGE(out(y)) = V2 \rightarrow VOLTAGE\_DIFF(y) = (V1-V2) * RESISTANCE(y) / (RESISTANCE(x) + RESISTANCE(y))$

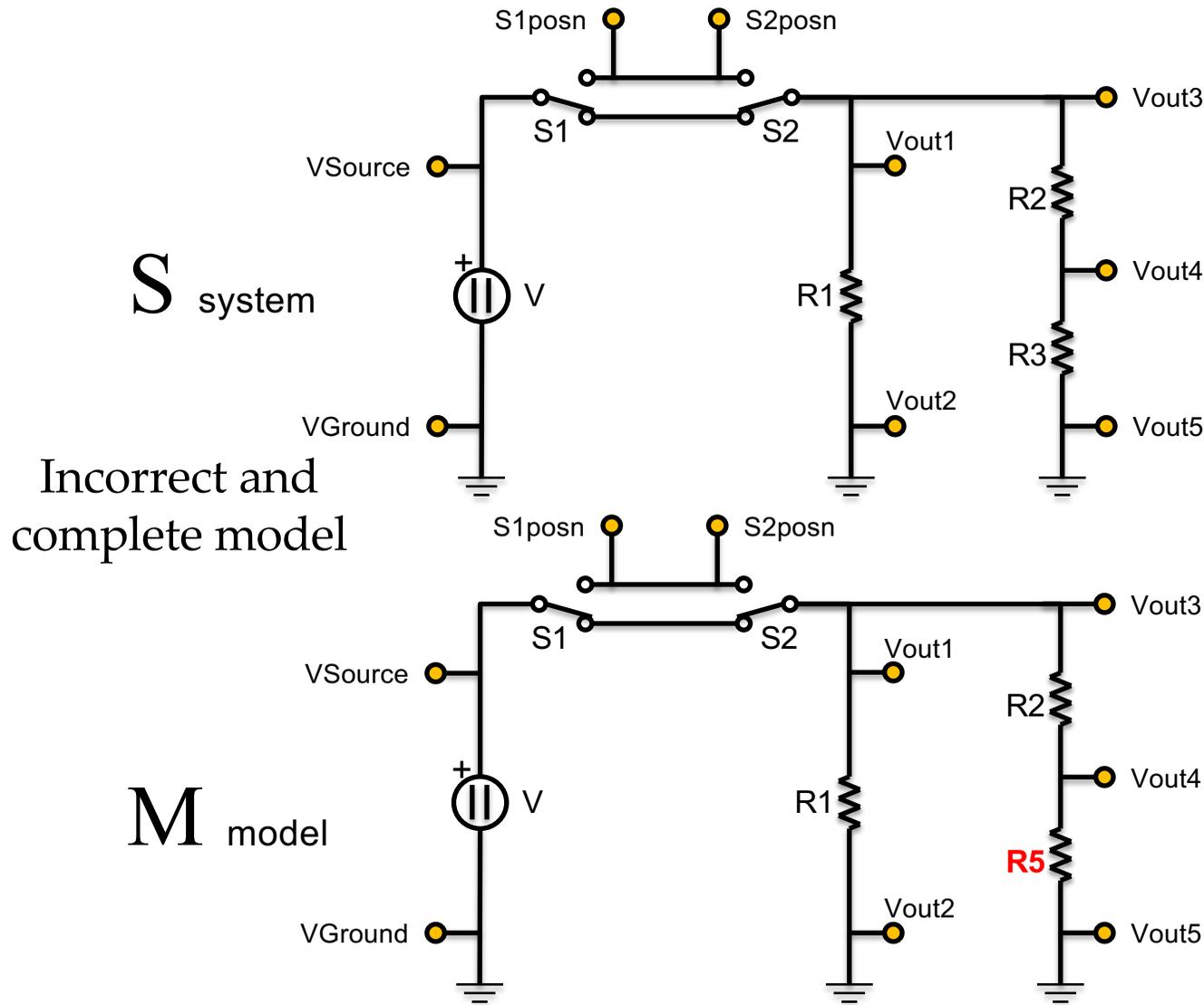
$SPDT\_SWITCH(x) \rightarrow SPDT\_OUT\_SEL(x) = T1 \vee SPDT\_OUT\_SEL(x) = T2$   
 $SPDT\_SWITCH(x) \wedge SPDT\_OUT\_SEL(x) = T1 \rightarrow in(x, 1) = out(x, 1)$   
 $SPDT\_SWITCH(x) \wedge SPDT\_OUT\_SEL(x) = T2 \rightarrow in(x, 1) = out(x, 2)$   
 $SPDT\_SWITCH(x) \wedge SPDT\_SWITCH(y) \wedge out(x, 1) = out(y, 1) \wedge out(x, 2) = out(y, 2) \wedge SPDT\_OUT\_SEL(x) = SPDT\_OUT\_SEL(y) \rightarrow in(x, 1) = in(y, 1)$   
 $SPDT\_SWITCH(x) \wedge SPDT\_SWITCH(y) \wedge out(x, 1) = out(y, 1) \wedge out(x, 2) = out(y, 2) \wedge \neg(SPDT\_OUT\_SEL(x) = SPDT\_OUT\_SEL(y)) \rightarrow \neg(in(x, 1) = in(y, 1))$

$$(S \cup A) \vdash \mathcal{G}$$

$$(M \subseteq \mathcal{G}) \wedge \text{Consistent}(M) \Rightarrow M \text{ is correct wrt } S$$

# Assurance of Correctness, Completeness, and Performance for MBFD

## Model Correctness and Completeness Results



$$S = \{SD_S \cup COMPONENTS_S \cup OBS_S\}$$

$SD_S$

$in(S1, 1) = out(V, 1)$   
 $out(S2, 1) = out(S1, 1)$   
 $out(S2, 2) = out(S1, 2)$   
 $in(S2, 1) = in(R1, 1)$   
 $in(S2, 1) = in(R2, 1)$   
 $in(R3, 1) = out(R2, 1)$   
 $out(R1, 1) = GROUND$   
 $out(R3, 1) = GROUND$   
 $in(V, 1) = GROUND$

$COMPONENTS_S$

Components = {V, R1, R2, R3,  
 S1, S2}  
 RESISTOR(R1)  
 RESISTOR(R2)  
 RESISTOR(R3)  
 SPDT\_SWITCH(S1)  
 SPDT\_SWITCH(S2)  
 VOLTAGE\_SOURCE(V)

$COMPONENTS_M$

Components = {V, R1, R2, **R5**,  
 S1, S2}  
 RESISTOR(R1)  
 RESISTOR(R2)  
**RESISTOR(R5)**  
 SPDT\_SWITCH(S1)  
 SPDT\_SWITCH(S2)  
 VOLTAGE\_SOURCE(V)

$OBS_M$

RESISTANCE(R1) = 1500  
 RESISTANCE(R2) = 1000  
**RESISTANCE(R5) = 4607.2**  
 VGround = 0  
 VSource = 5

$\mathcal{A}$  is a set of axioms

$RESISTOR(x) \rightarrow RESISTANCE(x) \in \mathbb{R} \wedge RESISTANCE(x) > 0$   
 $RESISTOR(x) \wedge RESISTOR(y) \rightarrow RESISTOR\_SERIES(x,y) = (in(y, 1) = out(x, 1))$   
 $RESISTOR(x) \wedge RESISTOR(y) \rightarrow RESISTOR\_PARALLEL(x,y) = (in(x, 1) = in(y, 1)) \wedge (out(x, 1) = out(y, 1))$   
 $RESISTOR(x) \wedge RESISTOR(y) \rightarrow SERIES\_RESISTANCE(x, y) = RESISTANCE(x) + RESISTANCE(y)$   
 $RESISTOR(x) \wedge RESISTOR(y) \rightarrow PARALLEL\_RESISTANCE(x, y) = RESISTANCE(x) * RESISTANCE(y) / (RESISTANCE(x) + RESISTANCE(y))$   
 $RESISTOR(x) \wedge RESISTOR(y) \wedge RESISTOR\_SERIES(x,y) \wedge VOLTAGE(in(x, 1)) = V1 \wedge VOLTAGE(out(y)) = V2 \rightarrow VOLTAGE\_DIFF(y) = (V1-V2) * RESISTANCE(y) / (RESISTANCE(x) + RESISTANCE(y))$

$SPDT\_SWITCH(x) \rightarrow SPDT\_OUT\_SEL(x) = T1 \vee SPDT\_OUT\_SEL(x) = T2$   
 $SPDT\_SWITCH(x) \wedge SPDT\_OUT\_SEL(x) = T1 \rightarrow in(x, 1) = out(x, 1)$   
 $SPDT\_SWITCH(x) \wedge SPDT\_OUT\_SEL(x) = T2 \rightarrow in(x, 1) = out(x, 2)$   
 $SPDT\_SWITCH(x) \wedge SPDT\_SWITCH(y) \wedge out(x, 1) = out(y, 1) \wedge out(x, 2) = out(y, 2) \wedge SPDT\_OUT\_SEL(x) = SPDT\_OUT\_SEL(y) \rightarrow in(x, 1) = in(y, 1)$   
 $SPDT\_SWITCH(x) \wedge SPDT\_SWITCH(y) \wedge out(x, 1) = out(y, 1) \wedge out(x, 2) = out(y, 2) \wedge \neg(SPDT\_OUT\_SEL(x) = SPDT\_OUT\_SEL(y)) \rightarrow \neg(in(x, 1) = in(y, 1))$

$$(S \cup \mathcal{A}) \vdash \mathcal{G}$$

$$\neg(M \subseteq \mathcal{G}) \wedge \text{Consistent}(M) \Rightarrow M \text{ is incorrect wrt } S$$

# Assurance of Correctness, Completeness, and Performance for MBFD

## Performance Assessment Results

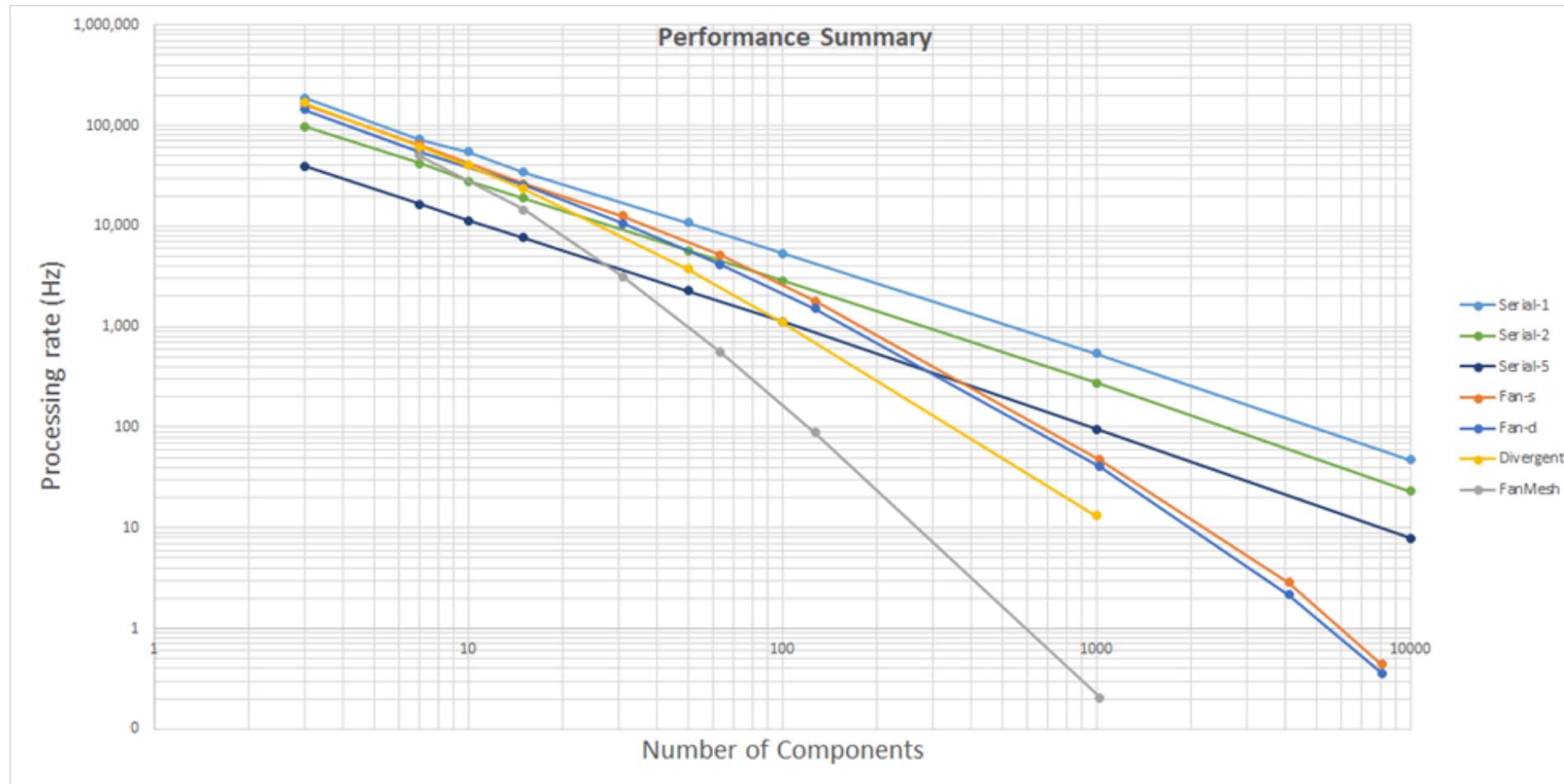
### Experimental Analysis

- Two MONSID runtime performance attributes analyzed.
  - Model processing time
  - Total time
- Attributes vary in a super-linear fashion wrt measured model structural characteristics of the model.
  - Most significant attribute: number of connections.
  - Power law of the form  $\text{time} = c^a$ , where  $c$  is the number of connections and  $a$  is a positive constant  $1 < a$ 
    - Value of “ $a$ ” varies to some extent between model families, ranging between 2 to 5.
  - Results are of the same form as those of analytical performance evaluation.

# Assurance of Correctness, Completeness, and Performance for MBFD

## Performance Assessment Results (cont'd)

### Analytical Assessment



- Serial family offers the fastest operation (Serial 1- channel model can be seen as the baseline).
- Increasing number of Components in any family result in slower performance.
- Connection density has a large impact on performance most notably seen in Fan Mesh but also seen in Serial-2 channel and Serial-5 channel.
- Increasing Nodes minimally impacts performance as evidenced by comparing Fan-S and Fan-D.

Comparison of model family tests in support of complexity and performance analysis

# Assurance of Correctness, Completeness, and Performance for MBFD

## Performance Assessment Results (cont'd)

### Analytical Assessment (cont'd)

- Empirical testing showed the largest influencers to processing time are Components and number of Connections.
  - Analysis shows MONSID complexity is polynomial.
  - In a general sense, MONSID's main method fits within  $O(n^4)$ , as predicted by counting, for nominal data, where  $n$  represents some combination of Components, Nodes, and Connections.
- Corresponds with analysis comparing MONSID Models with traditional graphs where the order is  $O(\text{Components} + \text{Connections})$ .
- MONSID improves on FM techniques for which complexity is exponential – potential reduction in processing time.

# Assurance of Correctness, Completeness, and Performance for MBFD

## Source Code Analysis and Code Coverage

### Source Code Analysis

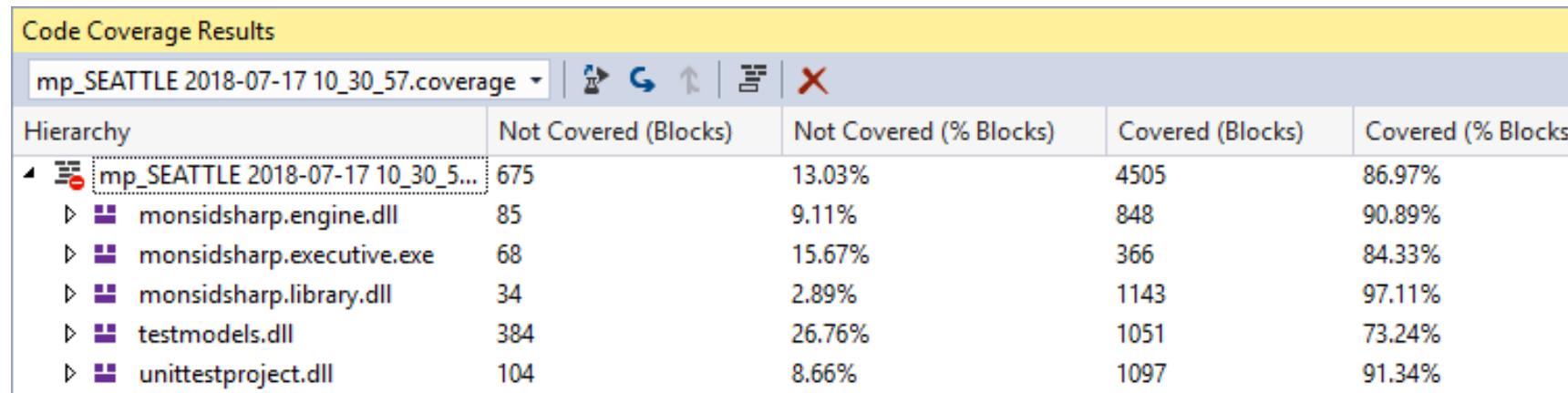
- **Goal**
  - Identify errors that desk review of source code may not find, and
  - Ensure that MONSID users can comply with JPL requirement to apply static source code analysis.
- **Analysis**
  - Analyzed version 2 of MONSID provided by Okean Solutions.
  - Used Semmle's "Semmle Core" tool (formerly Odas).
    - Available through JPL's Computer Aided Engineering (CAE) group.
- **Reviewed findings with Okean Solutions**
  - Nearly all violations were found to be due to Semmle Core not recognizing C++ 11.
  - Finding that Semmle Core does not recognize C++ 11 reported to Semmle via JPL's CAE group.

# Assurance of Correctness, Completeness, and Performance for MBFD

## Source Code Analysis and Code Coverage

### Code Coverage

- Code coverage percentages have increased during MONSID development.
  - Code coverage results for current version of MONSID API with the VS toolset shown below.



Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
mp_SEATTLE 2018-07-17 10_30_57.coverage				
mp_SEATTLE 2018-07-17 10_30_5...	675	13.03%	4505	86.97%
monsidsharp.engine.dll	85	9.11%	848	90.89%
monsidsharp.executive.exe	68	15.67%	366	84.33%
monsidsharp.library.dll	34	2.89%	1143	97.11%
testmodels.dll	384	26.76%	1051	73.24%
unittestproject.dll	104	8.66%	1097	91.34%

- Results are separated into the Engine, Executive, and Core Library.
- Most significant item in the figure is for Core Library (monsidsharp.library.dll)
  - Utilized by the Engine and Executive.
  - Current results show 97% coverage of the MONSID Core Library
    - Represents a 6.5% increase over previous results on earlier versions.

# Assurance of Correctness, Completeness, and Performance for MBFD

Source Code Analysis and Code Coverage (cont'd)

## Code Coverage Findings

- Each tool has its own way of defining and measuring coverage.
- JPL requirements, procedures, and guidelines call for analyzing test coverage, but
- No recommendations found for how coverage is to be measured (e.g., statement, condition, data definition and usage), or what coverage threshold should be<sup>1</sup>.
  - Projects do perform test coverage analysis, but measures and thresholds vary by project.
  - Individual projects contacted have heuristically-determined coverage goals of 80% or more.

1. This is an open research question

# Assurance of Correctness, Completeness, and Performance for MBFD

## Source Code Analysis and Code Coverage

### Code Coverage Recommendations

- Don't rely on test coverage measures alone when deciding whether a testing effort has been completed<sup>1</sup>.
- Use test coverage to supplement information such as test case generation approach and system structure.
- There are many toolsets. Pick a single toolset and understand how that tool measures code coverage.
- There are many coverage measures. Choose at most a small set of those that are understandable and have tool support.
- Test coverage analysis is iterative. Comparing results from run to run will provide the most insight.

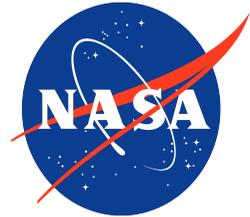
1. Staats, Matt, Gregory Gay, Michael Whalen, and Mats Heimdahl. "On the danger of coverage directed test case generation." In International Conference on Fundamental Approaches to Software Engineering, pp. 409-424. Springer, Berlin, Heidelberg, 2012.

# Future work

Coming up in 2019...

- Elaborate formal definitions of model correctness and completeness
  - Focus on tying the techniques for assuring correctness and completeness previously developed to framework defined by formal definitions.
  - Goal is to help engineers make more informed decisions
    - Which verification and assurance techniques should be used
    - Which aspects of a model require greater attention than others.
  - Also use definitions to support identification/development of invariants to be satisfied when using formal verification techniques (e.g., model checking, theorem proving).
- Demonstrate use of formal verification techniques to determine correctness of diagnostic engine/diagnostic models and the diagnostic engine.
  - Developing correctness properties is key to successful use of these techniques.
- Continue performance analysis, focusing on diagnostic ability (e.g., false-positive/false negative rates).
  - False-positive and false-negative diagnosis rates are functions of system complexity and level of model fidelity.
  - Analyzing false-positive and false-negative diagnosis rates is not trivial - also highly dependent on the design and implementation of the system to be diagnosed.
- Will investigate and develop the techniques for analyzing this aspect of MBFD performance and demonstrate the techniques using MONSID.

# Thank you AeroConf 2019!



**Jet Propulsion Laboratory**  
California Institute of Technology

---

[jpl.nasa.gov](http://jpl.nasa.gov)

**Questions welcome!**  
**(I can say “I don’t know” as well as  
the next person)**