Modeling Cyber-Physical Systems: Challenges and Recent Advances

Gabor Karsai
Institute for Software-Integrated Systems
Vanderbilt University

Seminar at U Conn - 3/3/2015

Acknowledgements

- Personnel
  - Janos Sztipanovits
  - Ted Bapty
  - Sandeep Neema
  - Larry Howard
  - Abhishek Dubey
  - Xenofon Koutsoukos
  - Zsolt Lattmann
  - Tihamer Levendovszky
  - Adam Nagel
  - Joseph Porter
  - Gabor Simko
  - …. and many others at the Institute for Software-Integrated Systems @ Vanderbilt University
- Sponsors
  - DARPA AVM, System F6
  - NSF CPS Program
  - AFRL, AFOSR, ARO
  - NASA
  - Boeing, BAE Systems, General Motors, Google Lockheed-Martin, Microsoft Research, Siemens, UTRC
  - … and many others (see http://www.isis.vanderbilt.edu/sponsors)

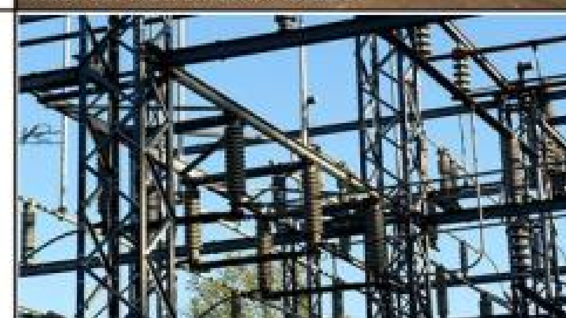
Modeling CPS

- Definition
- Examples
- The three aspects of modeling
  - Modeling the physical system
  - Models of computation and communication
  - Modeling the platform
- Model integration
- Recent results
- Research challenges
- Conclusions

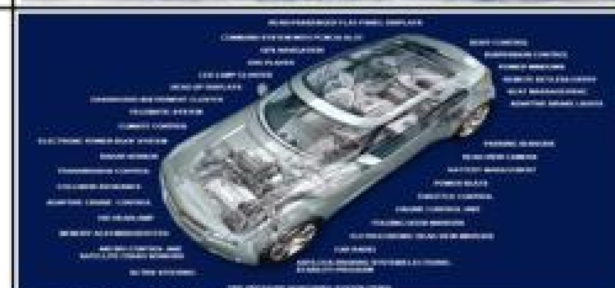
What is a Cyber-Physical System?

- An engineered system that *integrates* physical and cyber components where relevant functions are realized through the *interactions* between the physical and cyber parts.
  - Physical = some tangible, physical device + environment
  - Cyber = computational + communicational

CPS Examples

Sectors	Opportunities	
Health and Biomedical	In-home healthcare delivery. More capable biomedical devices for measuring health. New prosthetics for use within and outside the body. Networked biomedical systems that increase automation and extend the biomedical device beyond the body.	 Goldman: Operating Rooms of the Future
Agriculture	Energy efficient technologies. Increased automation. Closed-loop bioengineering processes. Resource and environmental impact optimization. Improved safety of food products.	 Michael Nørremark: HortiBot
Smart Grid	Highway systems that allow traffic to become denser while also operating more safely. A national power grid that is more reliable and efficient.	

CPS Examples

Sectors	Goals	
Aerospace	• Aircraft that fly faster and further on less energy. • Air traffic control systems that make more efficient use of airspace.	
Automotive	• Automobiles that are more capable and safer but use less energy. • Highways that are safe, higher throughput and energy efficient.	
Defense	• Fleets of autonomous, robotic vehicles • More capable defense systems • Integrated, maneuverable, coordinated, energy efficient • Resilient to cyber attacks	

The Good News…

Networking and computing delivers unique precision and flexibility in **interaction** and **coordination**

Computing/Communication

- Rich time models
- New type of interactions across highly extended spatial/temporal dimensions
- Flexible, dynamic communication mechanisms
- Time-variant, nonlinear behavior
- Introspection, learning, reasoning

Integrated CPS

- Elaborate coordination of physical processes
- Hugely increased system size with controllable, stable behavior
- Dynamic, adaptive architectures
- Adaptive, autonomic systems
- Self monitoring, self-healing system architectures and better safety/security guarantees.

…and the Challenges

Fusing networking and computing with physical processes brings new problems

Computing/Communication

- Cyber vulnerability
- New type of interactions across highly extended spatial/temporal dimensions
- Flexible, dynamic communication mechanisms
- Time-variant, nonlinear behavior
- Introspection, learning, reasoning

Integrated CPS

- Physical behavior of systems can be manipulated
- Lack of composition theories for heterogeneous systems, many unsolved problems
- Vastly increased complexity and emergent behaviors
- Lack of theoretical foundations for CPS dynamics
- Verification, certification, predictability face fundamentally new challenges

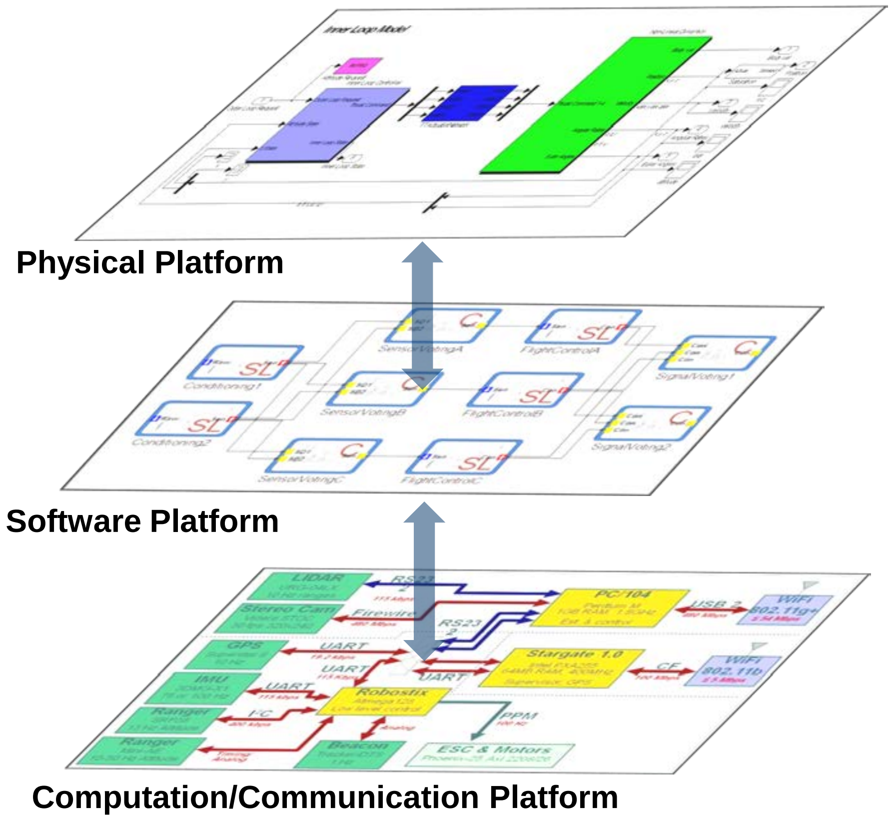
Example for a CPS Approach

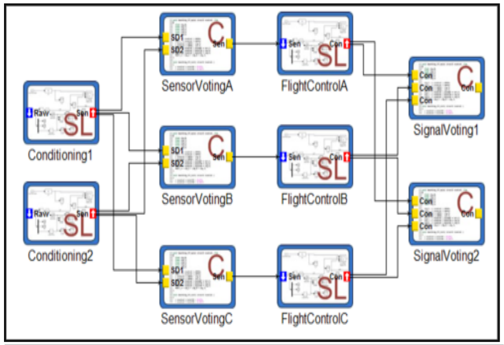
Abstraction layers define platforms.

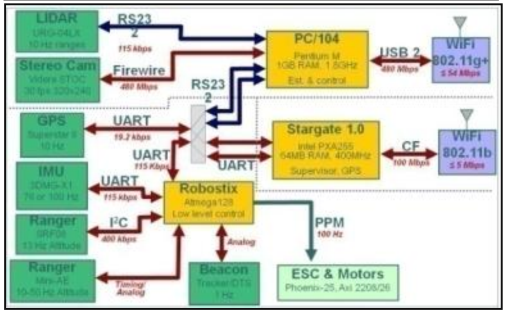
Abstractions are linked through mapping.

Abstraction layers allow the verification of different properties .

Key Idea: Manage design complexity by creating abstraction layers in the design flow.



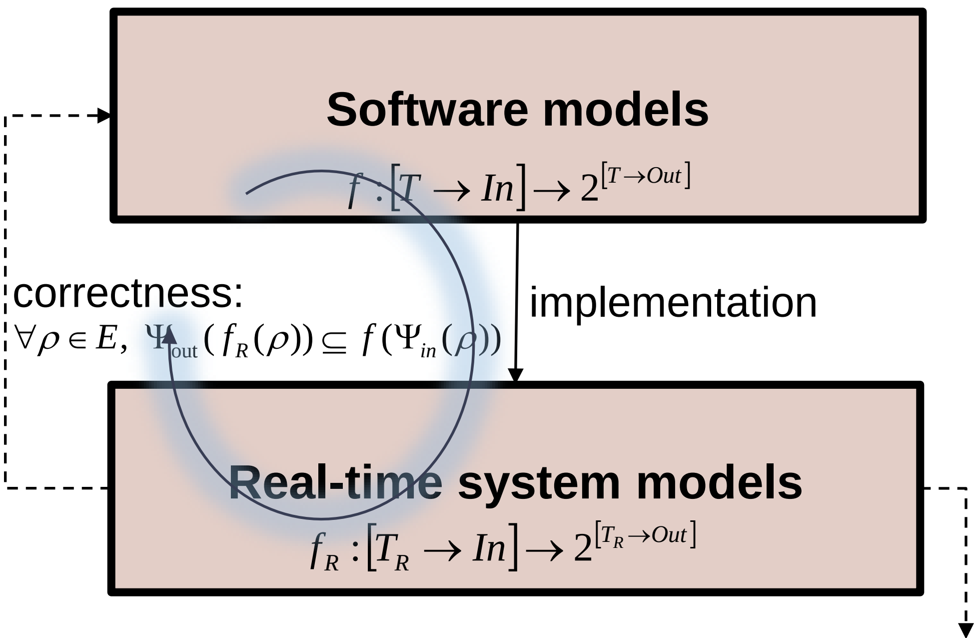




Abstraction layers: SW-RTS

Sifakis at al: "Building Models of Real-Time Systems from Application Software," *Proceedings of the IEEE* Vol. 91, No. 1. pp. 100-111, January 2003

In CPS, essential system properties such as stability, safety, performance are expressed in terms of physical behavior

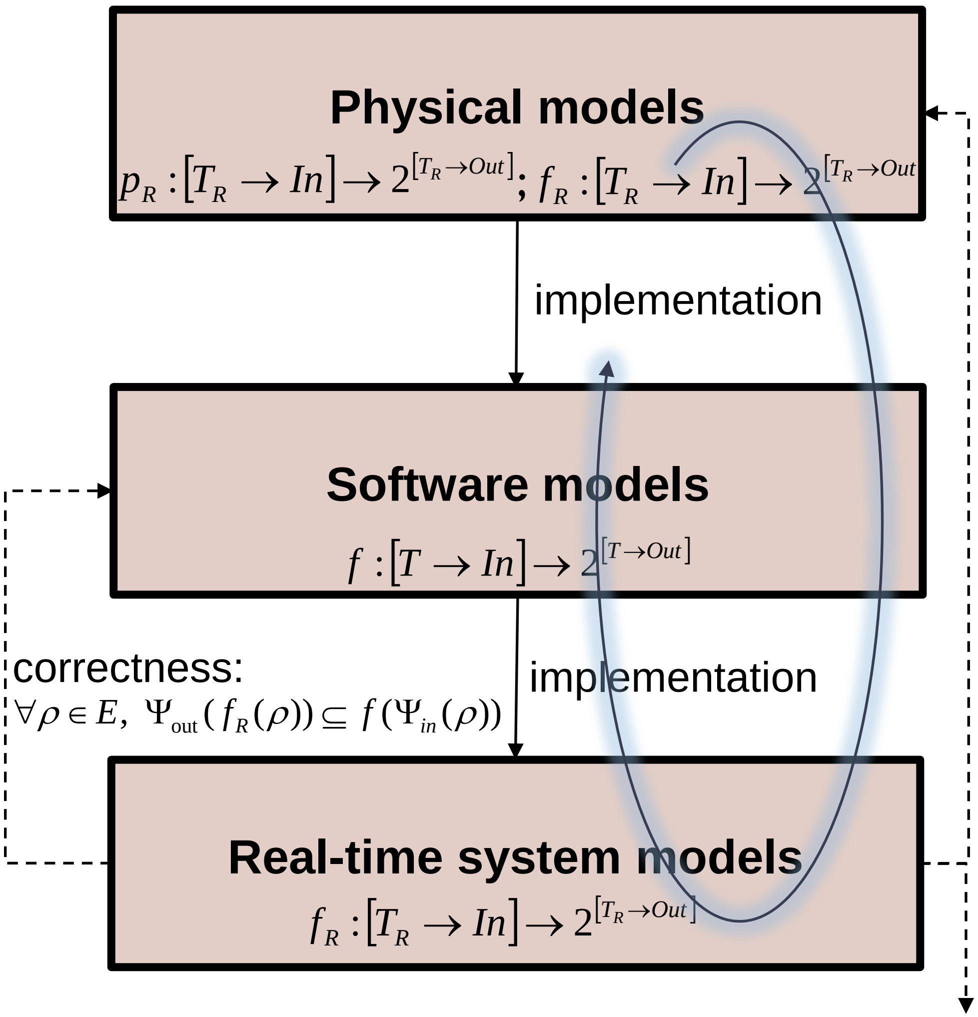


timing analysis (*P*)

$\forall \rho \in E, \forall \pi \in f_R(\rho), (\rho, \pi) \in P$

- f : reactive program. Program execution creates a mapping between logical-time inputs and outputs.
- f_R : real-time system. Programs are packaged into interacting components. Scheduler control access to computational and communicational resources according to time constraints *P*

Abstraction layers: PHY-SW-RTS



timing analysis (*P*)

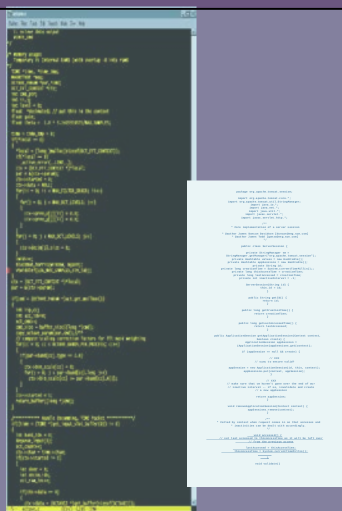
$\forall \rho \in E, \forall \pi \in f_R(\rho), (\rho, \pi) \in P$

<u>Re-defined Goals:</u>

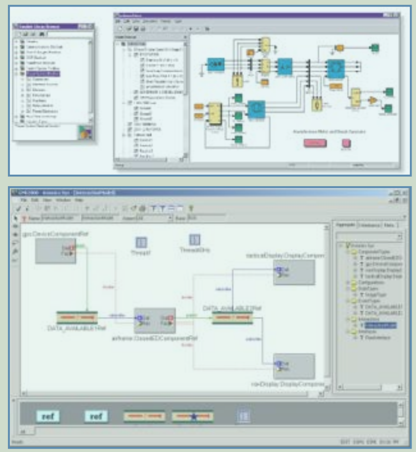
- **Compositional verification of essential dynamic properties**
  - stability
  - safety
- **Derive dynamics - offering robustness against implementation changes and uncertainties caused by faults and cyber attacks**
  - fault/intrusion induced reconfiguration of SW/HW
  - network uncertainties (packet drops, delays)
- **Decrease verification complexity**

Why is CPS Hard?

Software



Control

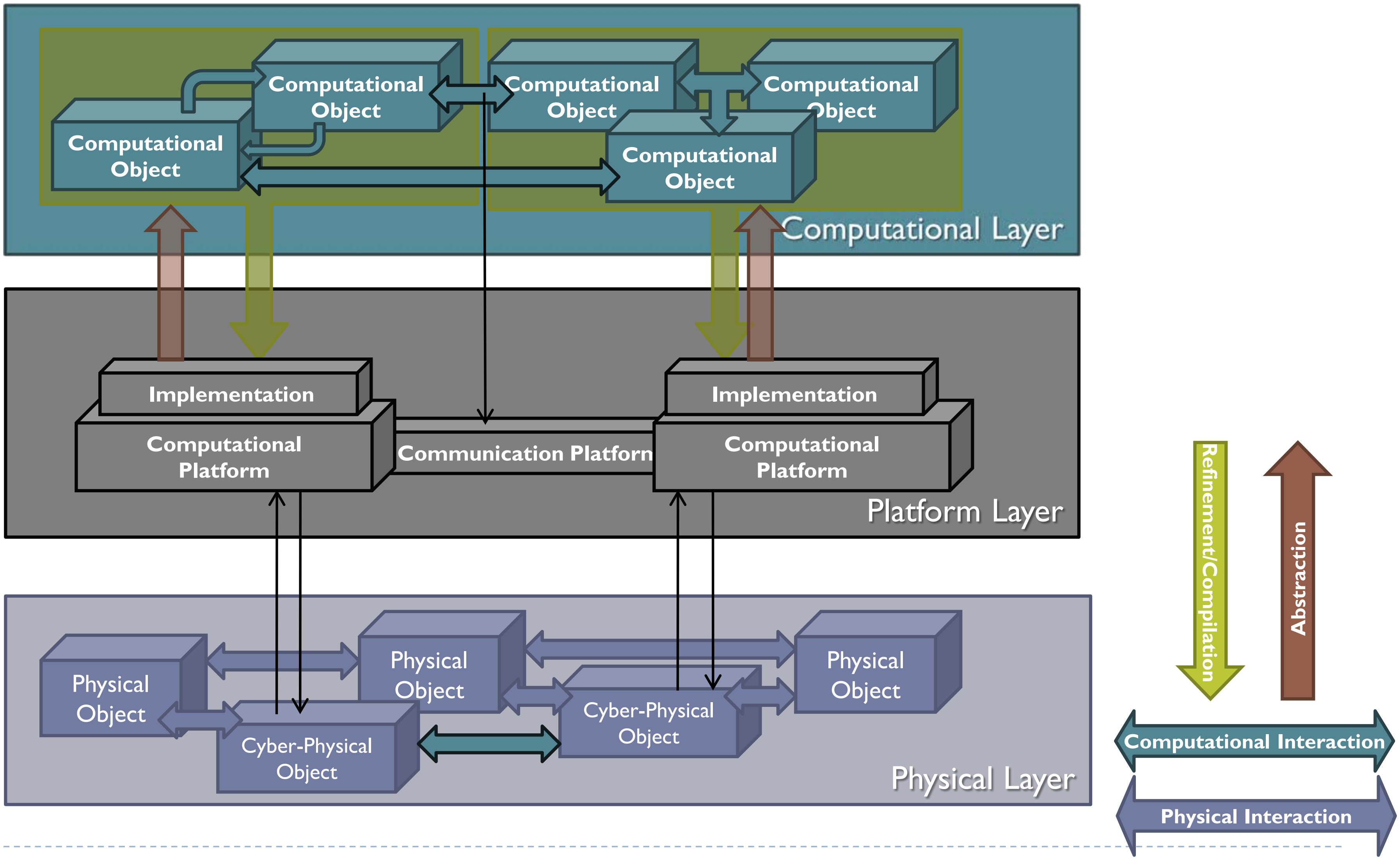


Systems

Crosses Interdisciplinary Boundaries

- Disciplinary boundaries need to be realigned
- New fundamentals need to be created
- New technologies and tools need to be developed
- Education and training need to be restructured

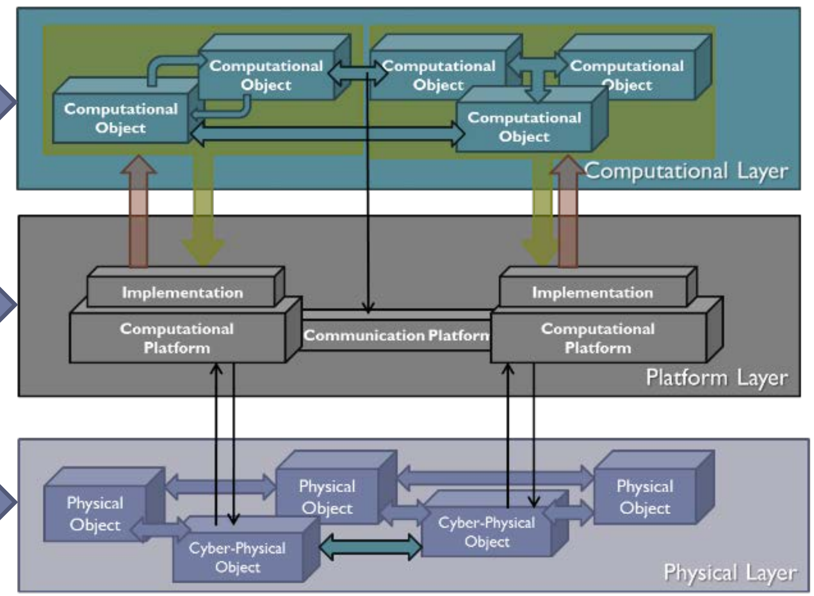
CPS Layers and Interactions



CPS and Model-based Design

Design of CPS layers via MDE

- Software models
- Platform models
- Physical models



Challenge: How to integrate the models so that cross-domain interactions can be understood and managed?

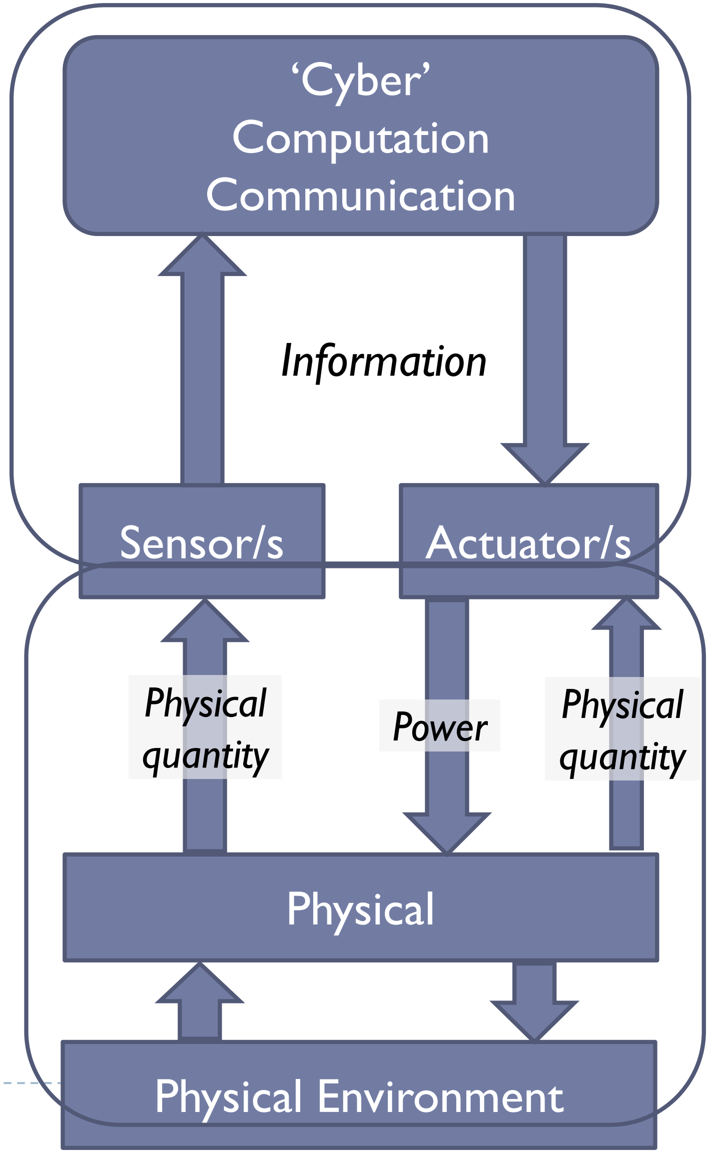
Model Integration for CPS

- Issues
  - Cyber models are insufficient, physical models are insufficient
  - Many modeling paradigms for physical systems (consider engineering or physics!)
  - Many interaction pathways: P2P, P2C, C2C, P2C2P, C2P2P2C
- Universal modeling language with precisely defined semantics?
  - All models are abstractions of reality from a specific point of view for a specific purposes. Universality is not pragmatic.
- Universal modeling language with no/sparse semantics?
  - [SysML] Enabler but not a complete solution – needs content *semantics*

Model Integration for CPS

- Objective: To support the model-based design of CPS
  - Represent the *design*: both physical and cyber, and the *interfaces*
  - Allow analysis of the design
    - Simulation-based evaluation and V&V
    - Discovering unintended interactions
    - Formal verification
  - Drive the implementation of the design
    - Compile to code, drive the fab

Key: understanding cross-domain interfaces and interactions



Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics

DARPA
Adaptive Vehicle Make (AVM) Program

A major DARPA program (a decade after MoBIES):

End-to-end model- and component-based design and integrated manufacturing of a new generation of vehicles; i.e. complex, real-life cyber-physical systems. From infrastructure to manufactured vehicle prototype in five years (2010-2014).

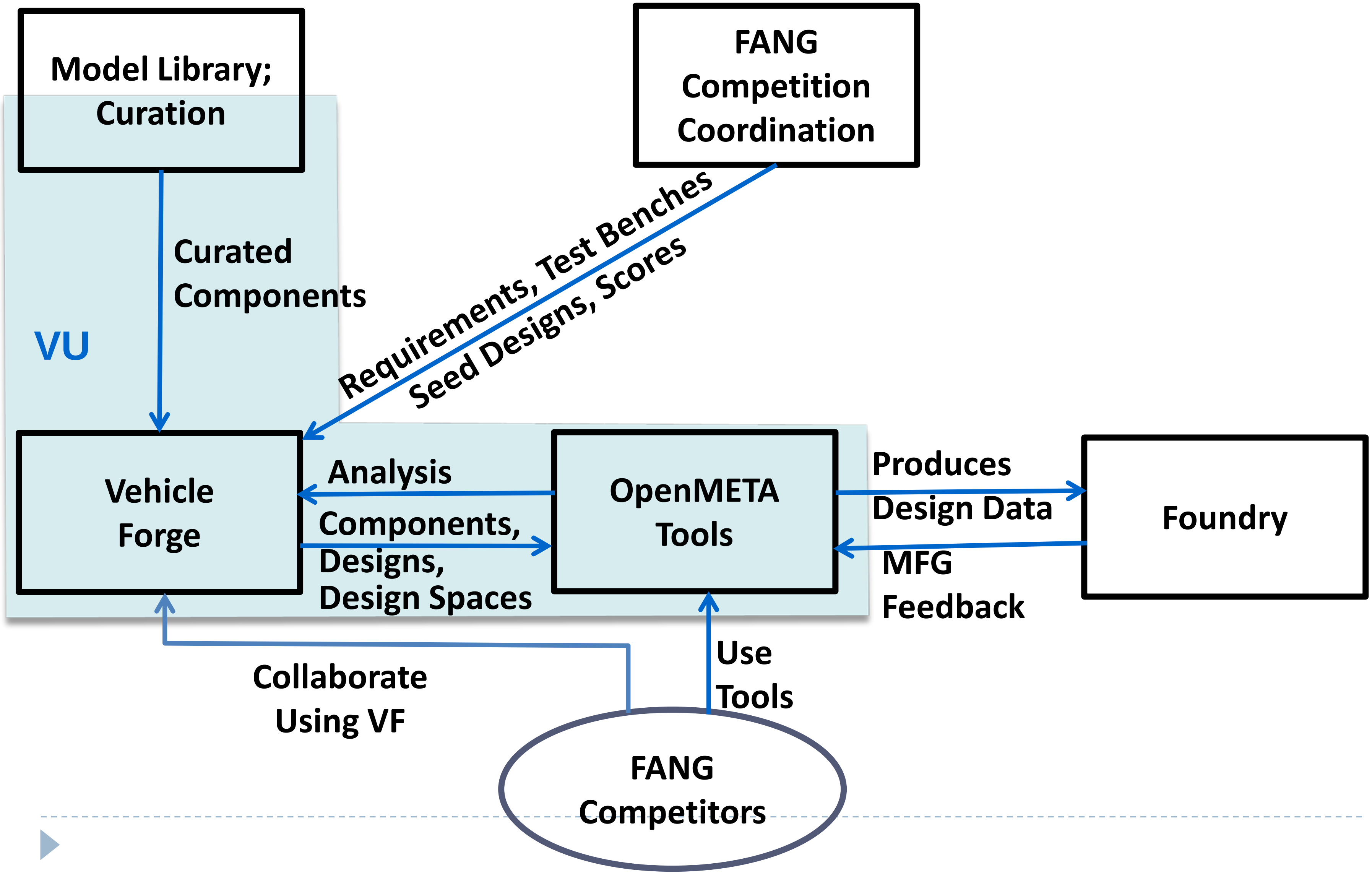
Engineering/economic goals:

- **Decrease development time by 80%** in defense systems (brings productivity consistent with other industries)
- Enable the adoption of **fabless design** and **foundry** concept in CPS
- **"Democratize" design** by open source tool chain, crowed-sourced model library and prize-based design challenges

AVM Scientific Challenge

- **Achieve AVM goals by pushing the limits of "correct-by-construction" design using**
- **Model-based Technologies**
  - **Computational models that predict properties of cyber-physical systems "as designed" and "as built".**
  - **Challenge: Develop domain-specific abstraction layers for complex CPS that are evolvable, heterogeneous, yet semantically sound and supported by tools.**
- **Component-based Technologies**
  - **Reusable units of knowledge (models) and manufactured components.**
  - **Challenge: Go beyond interoperability – find opportunities for composition where system-level properties *can* be computed from the properties of components**

Technical Areas



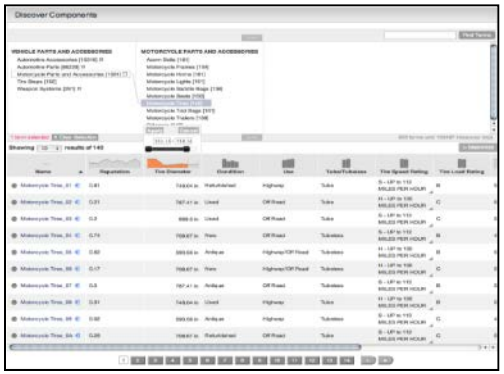
Tools for CPS Design

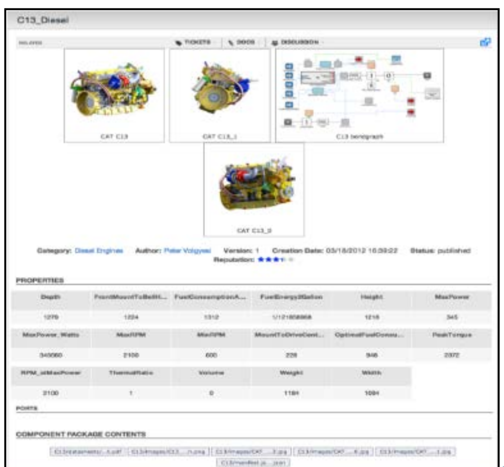
- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Collaborative environment: Vehicle Forge
  - Engineering environment: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics

Interface to OpenMETA: VehicleForge

Components

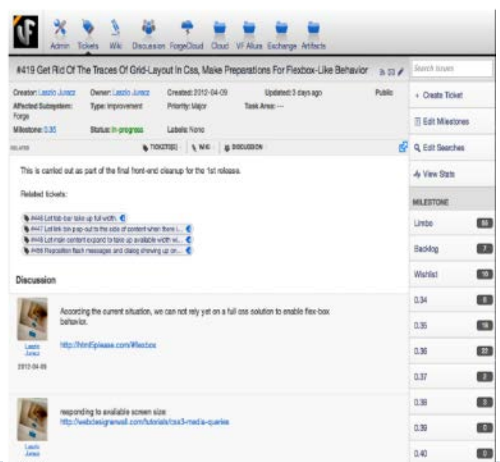
- Component discovery interface based on taxonomical- and faceted search
- Component view/visualization





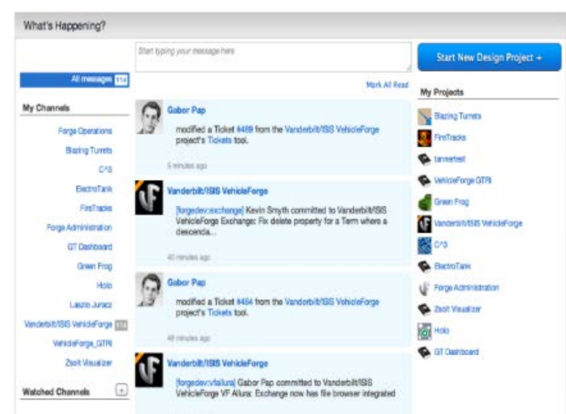
Design Projects

- Self-provisioned collaboration tools
  - Wiki,
  - Discussion Forum,
  - Issue tracking for managing team work.
- Git/SVN repositories for design artifacts
- Project and tool-based permission control
- Notification and Messaging system (in e-mail or as Dashboard messages)
- Set of available tools is extensible

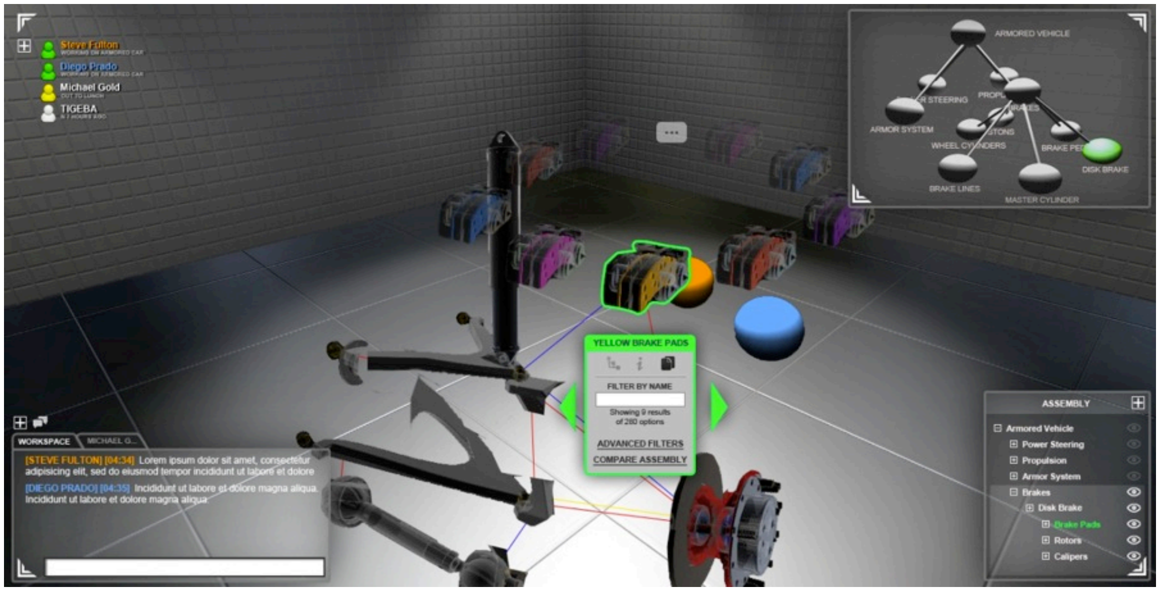


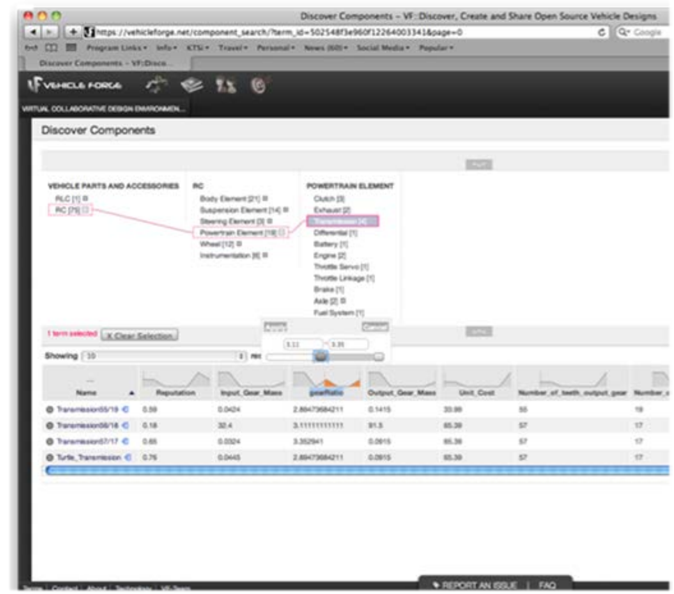
Designers

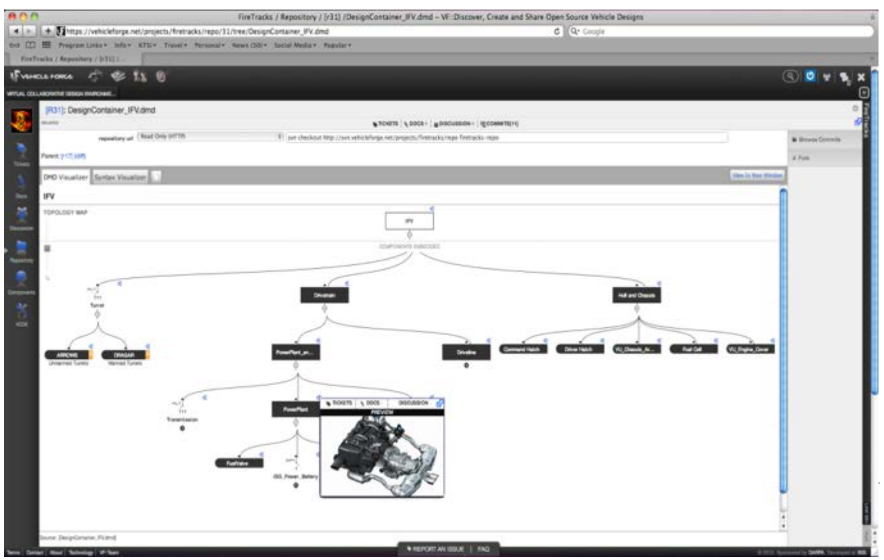
- Public profile to show recent activities and involvement in design projects
- Designer portfolio publishing résumé and for self-promotion
- Find designers based on expertise and résumé
- Private profile for customizing account and notification settings
- User dashboard showing feeds of activities from projects, public/private messages from other users, announcements from forge-message channels

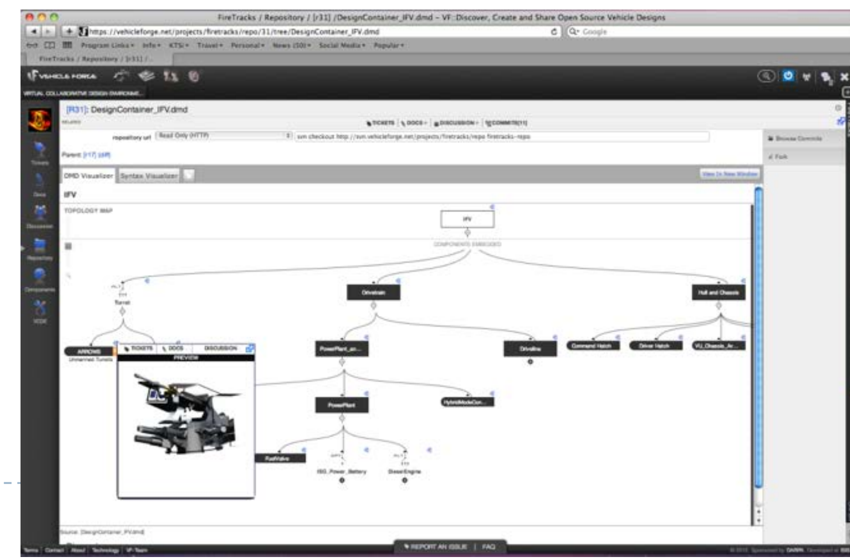


VehicleForge Gateway

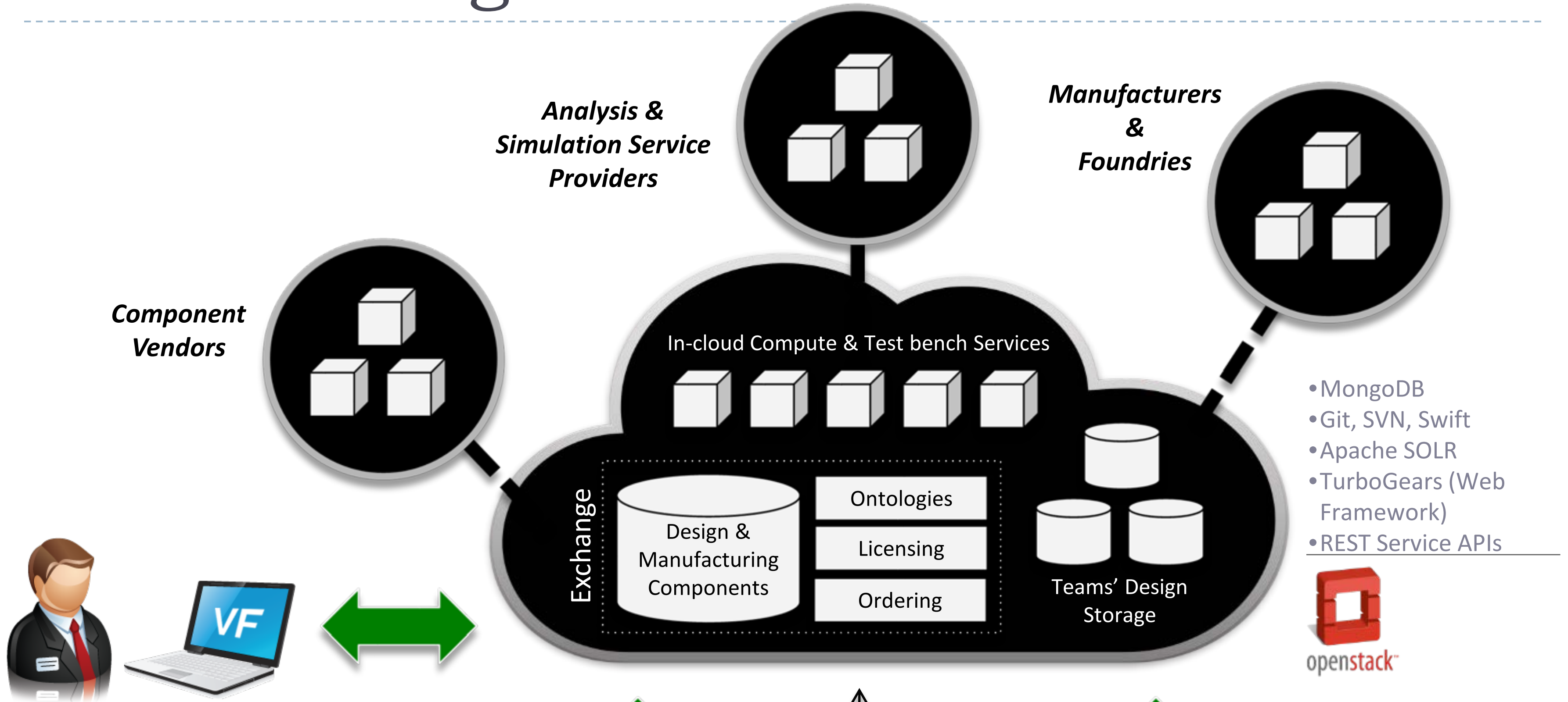






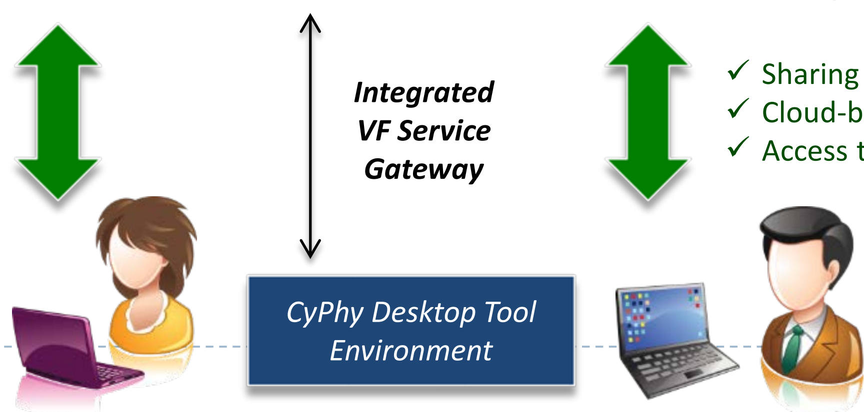


Service Integration Platform



Browser-based

- ✓ Coordination and Monitoring Tools
- ✓ Design-space Evaluation and Visualizers
- ✓ Team-collaboration Tools
- ✓ Component Discovery and Subscription
- ✓ Service and Resource Allocation



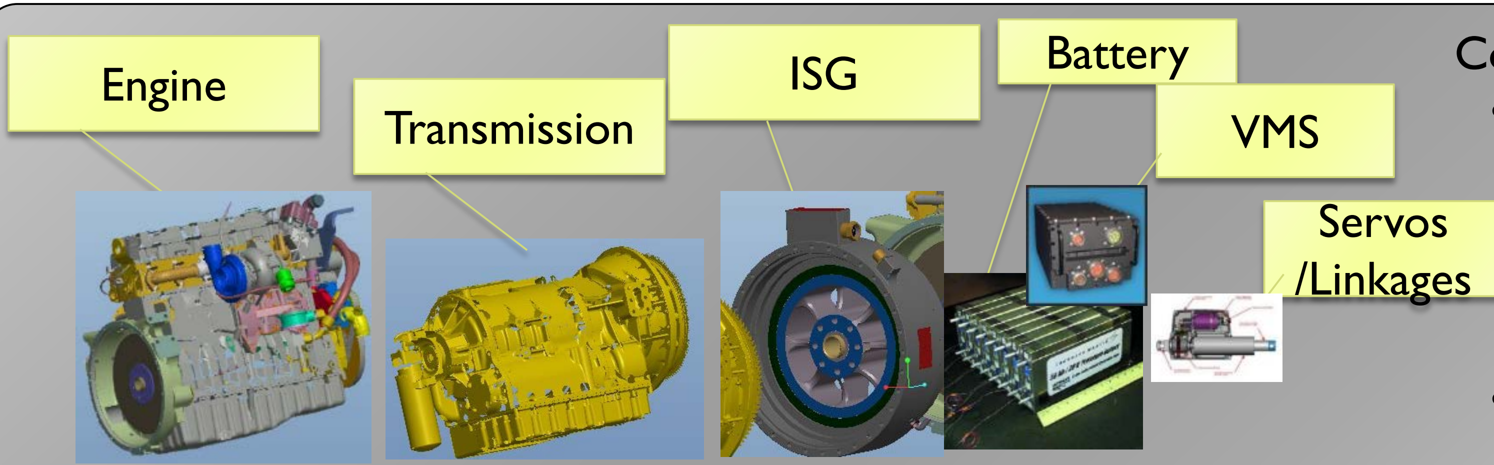
- ✓ Sharing and Collaboration
- ✓ Cloud-based Analysis
- ✓ Access to Remote Resources

- http://vehicleforge.org/

Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics

AVM Components



Components span:
- Multiple physics domains
- Multiple engineering domains
- Multiple tools

Component-based:
- Physical
- Cyber
- Cyber-Physical

Model-based
- Model-Integrated Design and
- Manufacturing Process

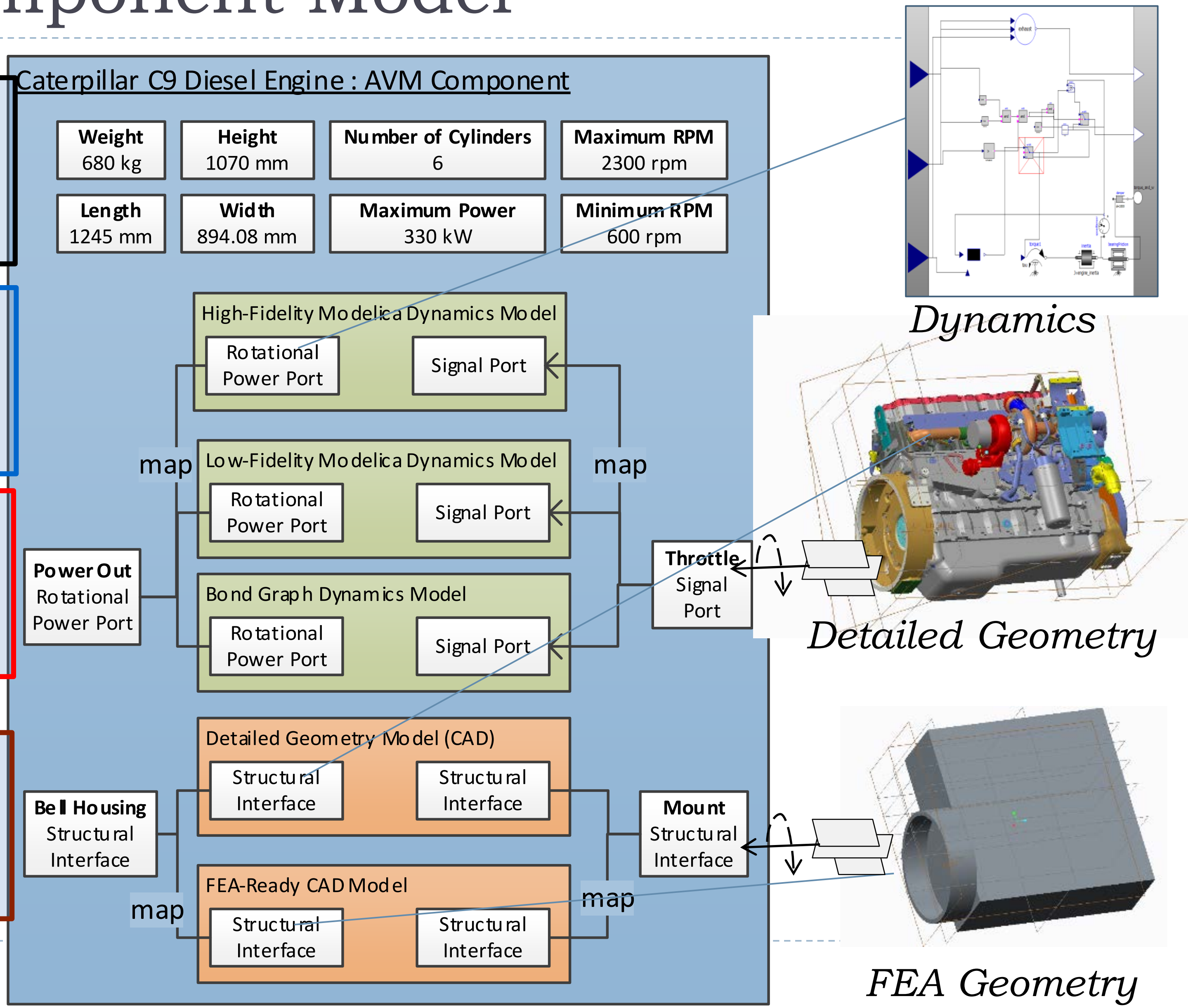
AVM Component Model

Parameter/Property Interfaces
- characterize
- configure

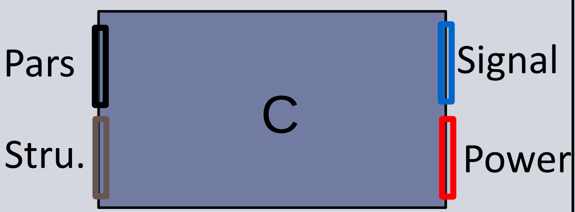
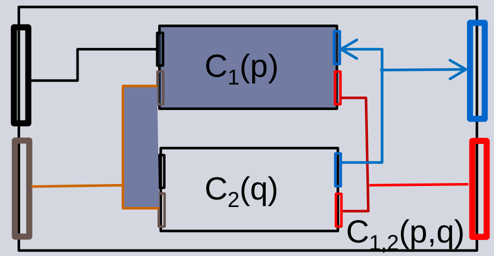
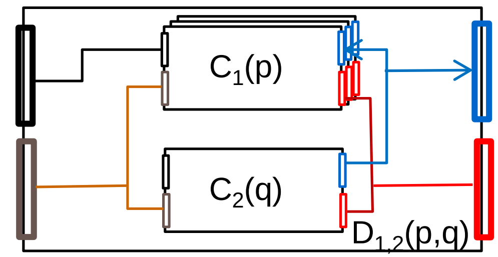
Signal Interfaces
- causal/directional
- logical connect.
- no power transfer

Power Interfaces
- acausal
- physical phen. (torque/angle..)
- power flow

Structural Interfaces
- named datums
- surface/axis/point
- mapped to CAD

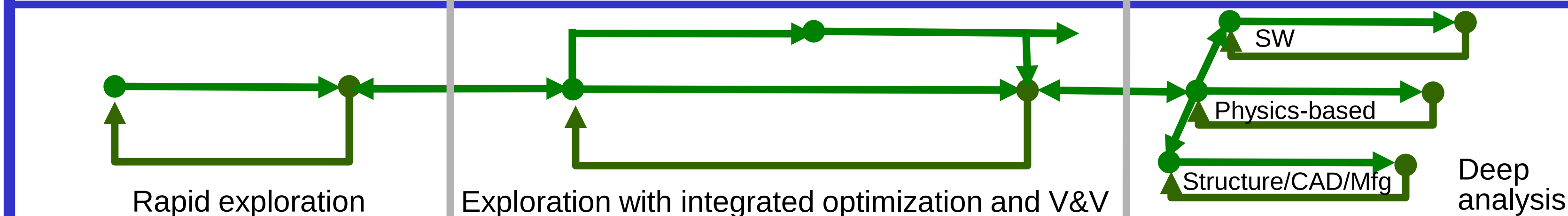


Components, Designs, Design Spaces

Components	Designs	Design Spaces
		
Self-contained building block	**Instantiate** and **connect** components	Sets of **parameterized architectures**
Properties and **Parameters**	Parameters, behaviors, geometry are **composed**	Extended around **seed designs**
Wrapper for **detailed domain models**	Can be wrapped as a **component**	Shaped by **design and manufacturability constraints**
Aggregates the domain interfaces into a **single set of component interfaces**	Aggregates the component interfaces into a **single set of system interfaces**.	Accumulates, evolves **design and manufacturing knowledge**

Design Flow

Architecture Design		Integrated Multi-physics/Cyber Design			Detailed Design	
Modeling	Exploration	Modeling	Simulation	V&V	Modeling	Analysis



Architecture Design

- Design Space + Constraint Modeling
- Architecture Modeling
- Static Component Modeling (multi-physics)

Integrated Multi-physics/Cyber Design

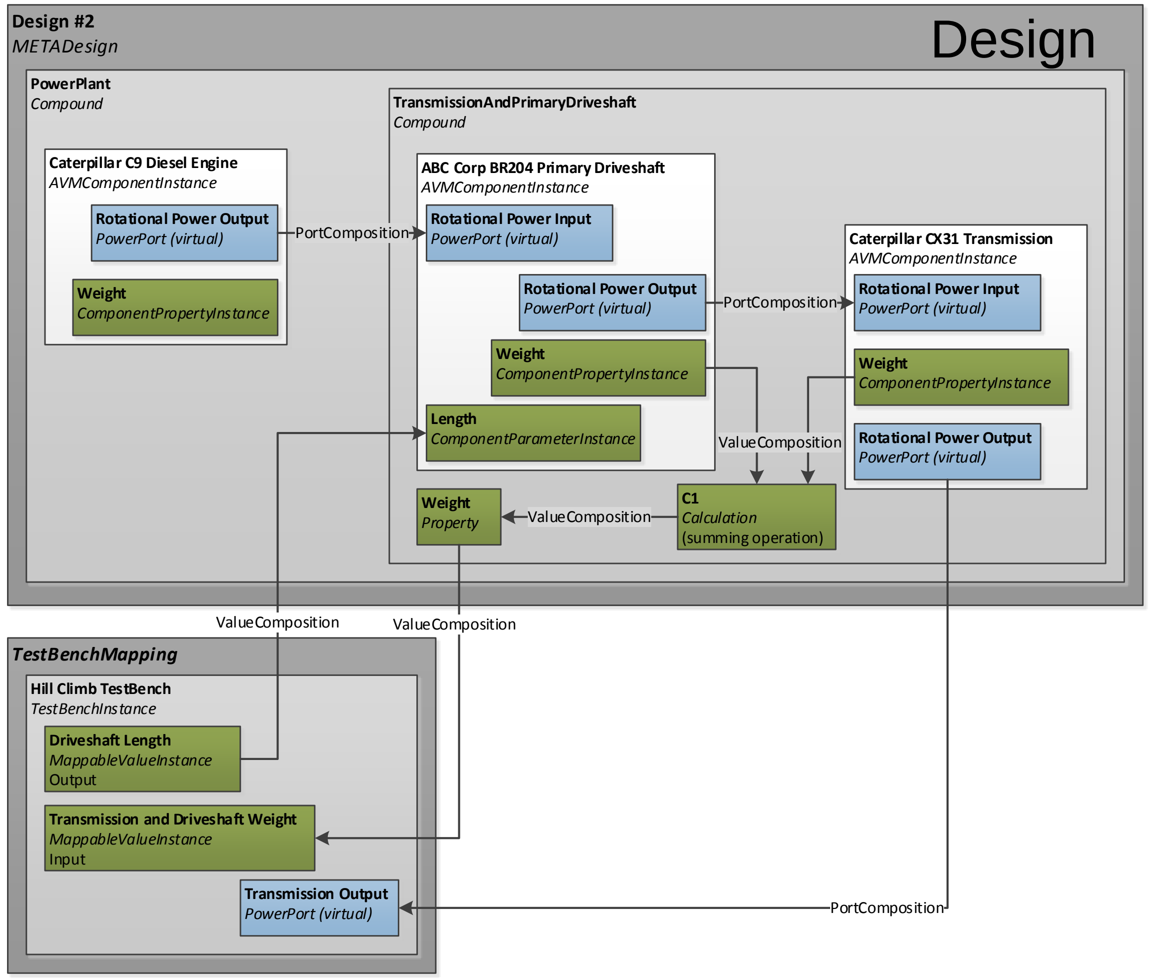
- Design Space + Behavioral Constraint Modeling
- Architecture Modeling
- Dynamics Modeling (multiple abstractions and multi-physics)
- CAD/assembly modeling
- Coarse Manufacturing Constraint Modeling

Detailed Design

- Architecture Modeling
- Detailed Domain Modeling
  - CAD
  - FEA; thermal, fluid…
  - Surrogate gen.
- Detailed Manuf. Modeling
- RT SW modeling

Requirements and Test Benches

- Using each component's mappings to detailed domain models, system-level analyses are automatically composed to verify:
  - Static properties
  - Multi-physics dynamics
  - Geometry
  - FEA properties
- **META Test Benches** provide an analysis context, including stimulus, loading, and monitoring.
- **Test Benches** include algorithms to produce **Metrics,** which are used to evaluate the design against **Requirements.**
- **META Design Models** are mapped to these **Test Benches**.
- **Design Spaces** can also be mapped to **Test Benches**, enabling rapid evaluation of a family of point designs.



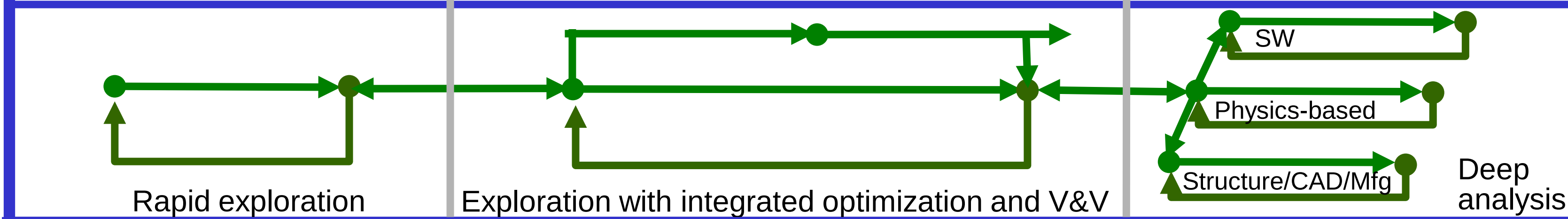
Design and Test Bench Mapping

Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics

Design Flow Spans Heterogeneous Modeling Domains

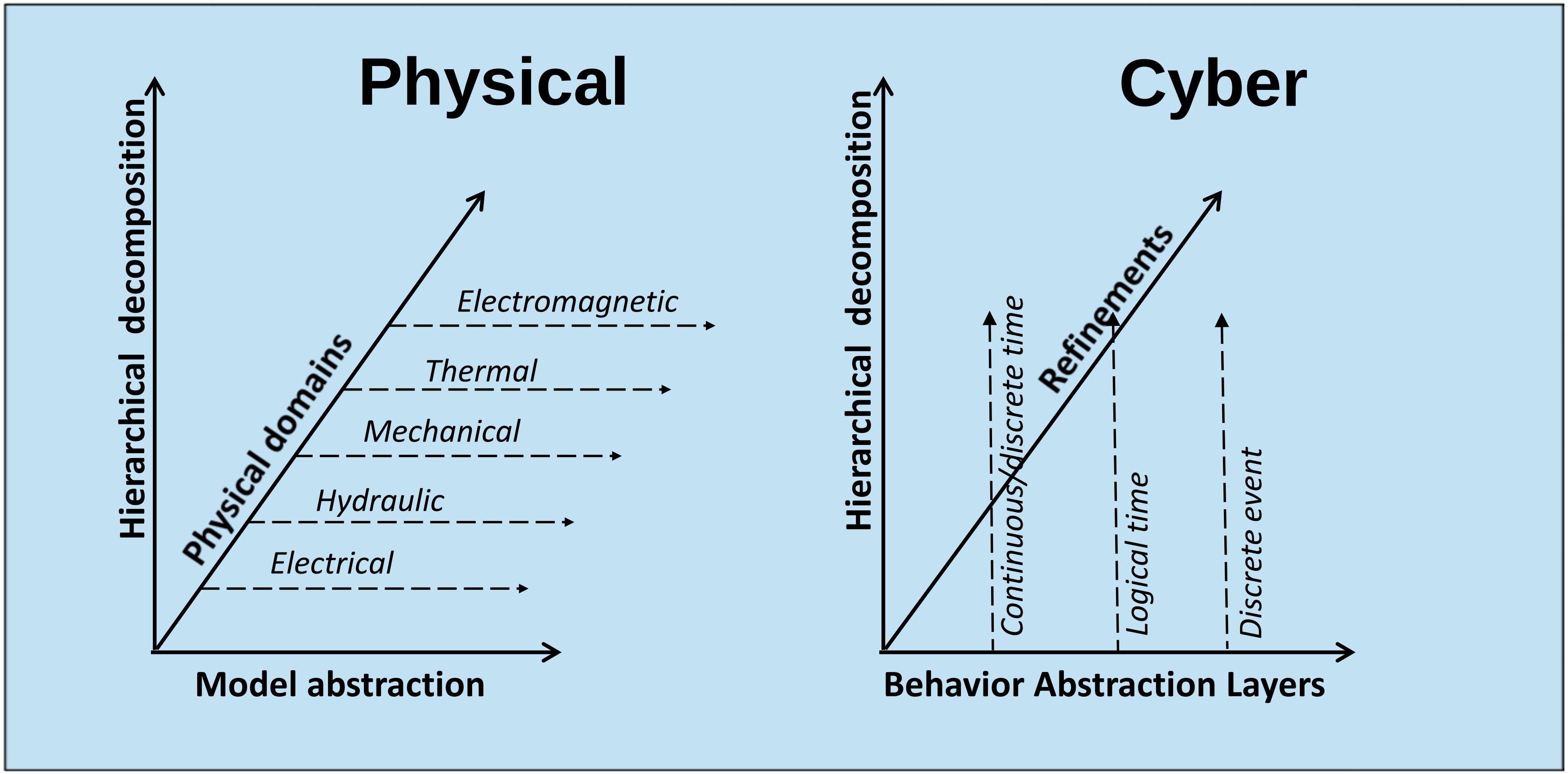
Architecture Design	Integrated Multi-physics/Cyber Design	Detailed Design
Modeling Exploration	Modeling Simulation V&V	Modeling Analysis



Architecture Design	Integrated Multi-physics/Cyber Design	Detailed Design
• Design Space + Constraint Modeling • Architecture Modeling • Static Component Modeling (multi-physics)	• Design Space + Behavioral Constraint Modeling • Architecture Modeling • Dynamics Modeling (multiple abstractions and multi-physics) • CAD/assembly modeling • Coarse Manufacturing Constraint Modeling	• Architecture Modeling • Detailed Domain Modeling – CAD – FEA; thermal, fluid… – Surrogate gen. • Detailed Manuf. Modeling • RT SW modeling

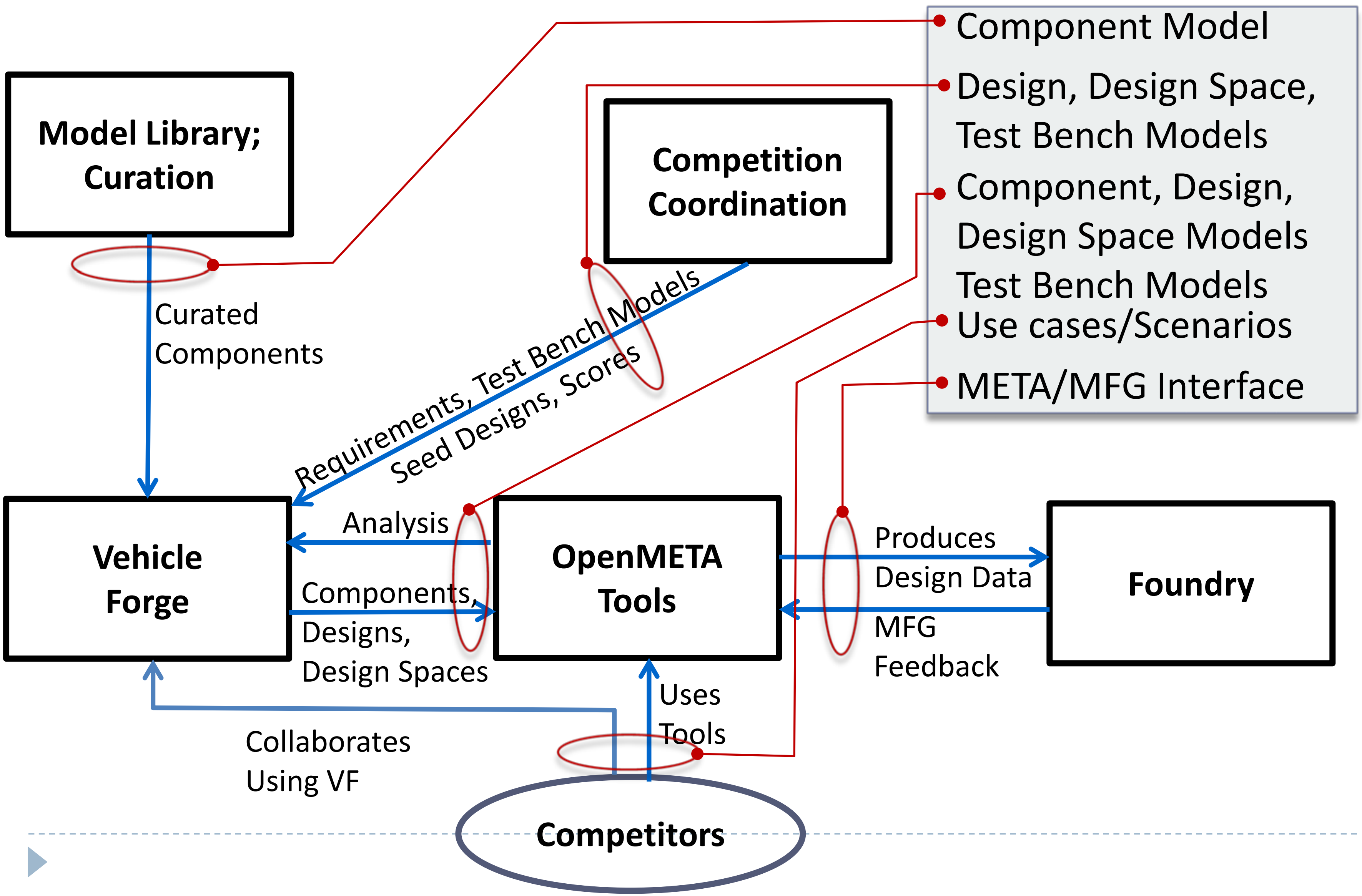
Domain Specific Modeling Languages

Modeling Domains



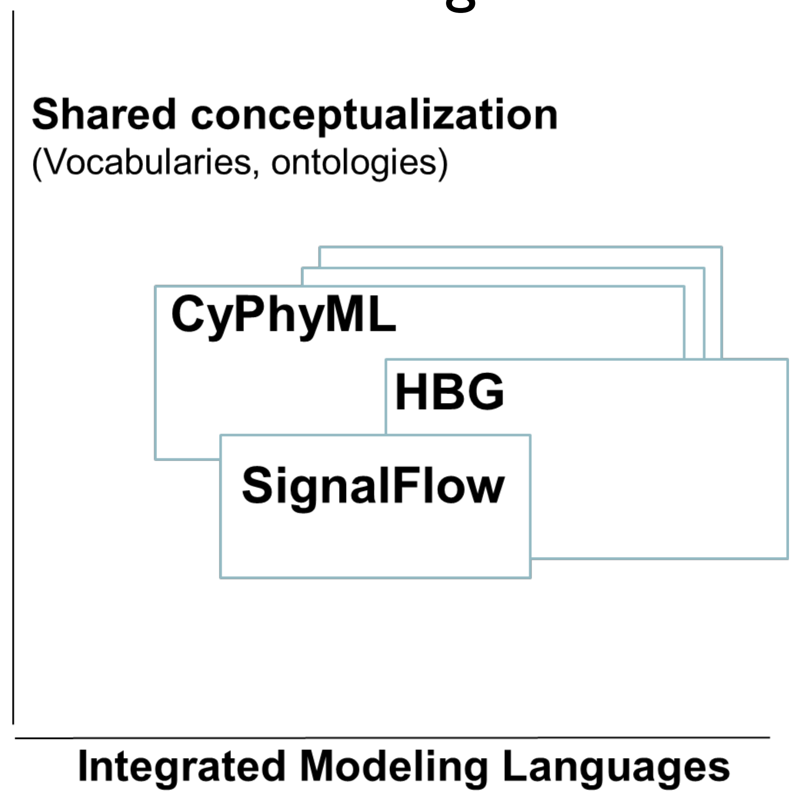
Key META Challenge:
Modeling cross-domain interactions

Information Flows Across Program Components



Information Architecture Challenges

- *Shared* conceptualization
- Semantically *sound* modeling languages
- Integration of *many* tools and their modeling languages



Information Architecture Challenges

How should we choose vocabularies, ontologies?

- Could not find standards covering even smaller part of the AVM domain…
- Grow and evolve vocabularies/ontologies during model library development
- Adopt vocabularies as defined by integrated tools (such as Modelica)

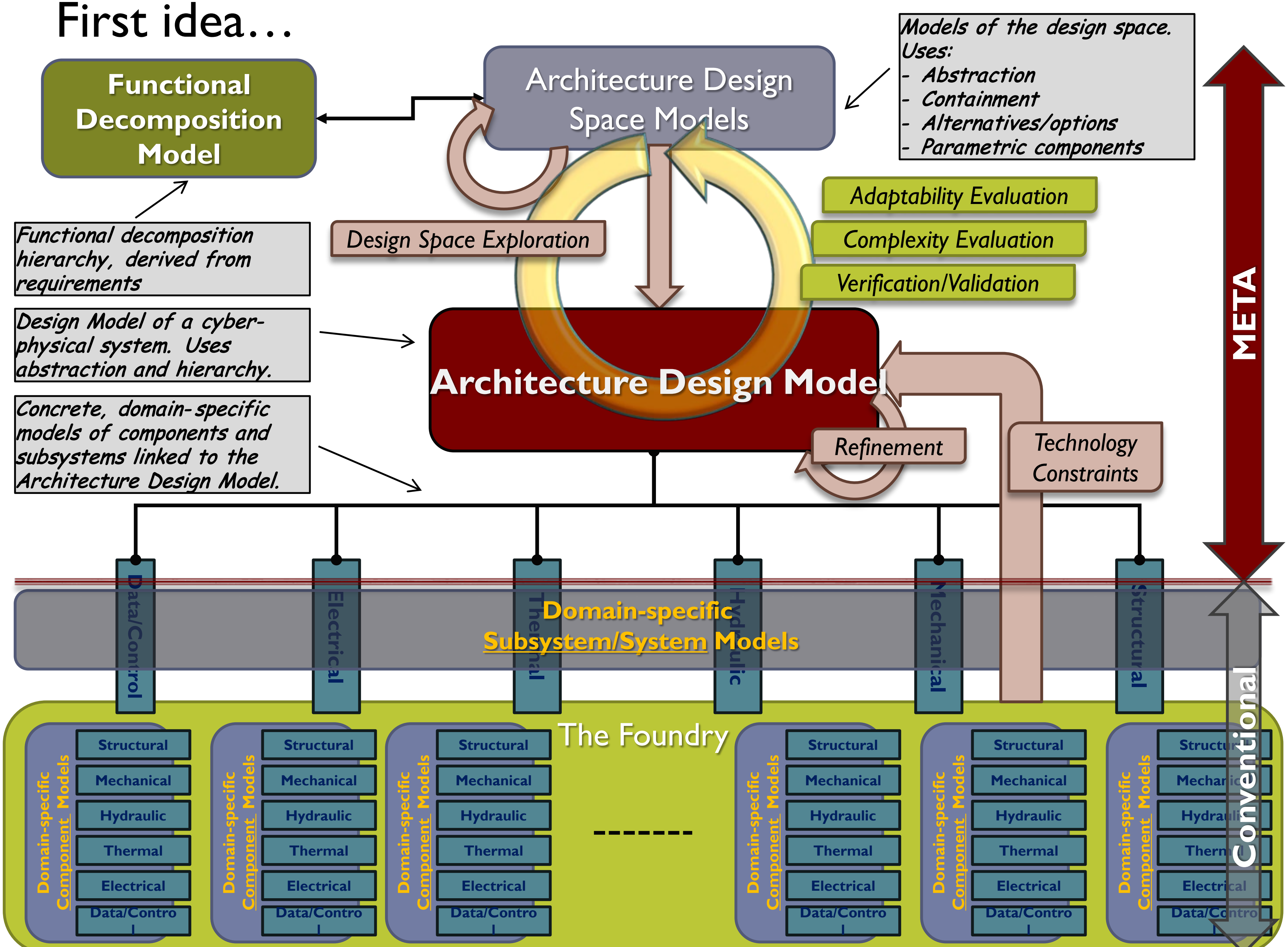
How should we choose modeling language(s)?

- Define yet another modeling language?
- Choose one that already exists and broad enough to cover the design domain?
- Create a new standard or update an old one?

Unintended consequences

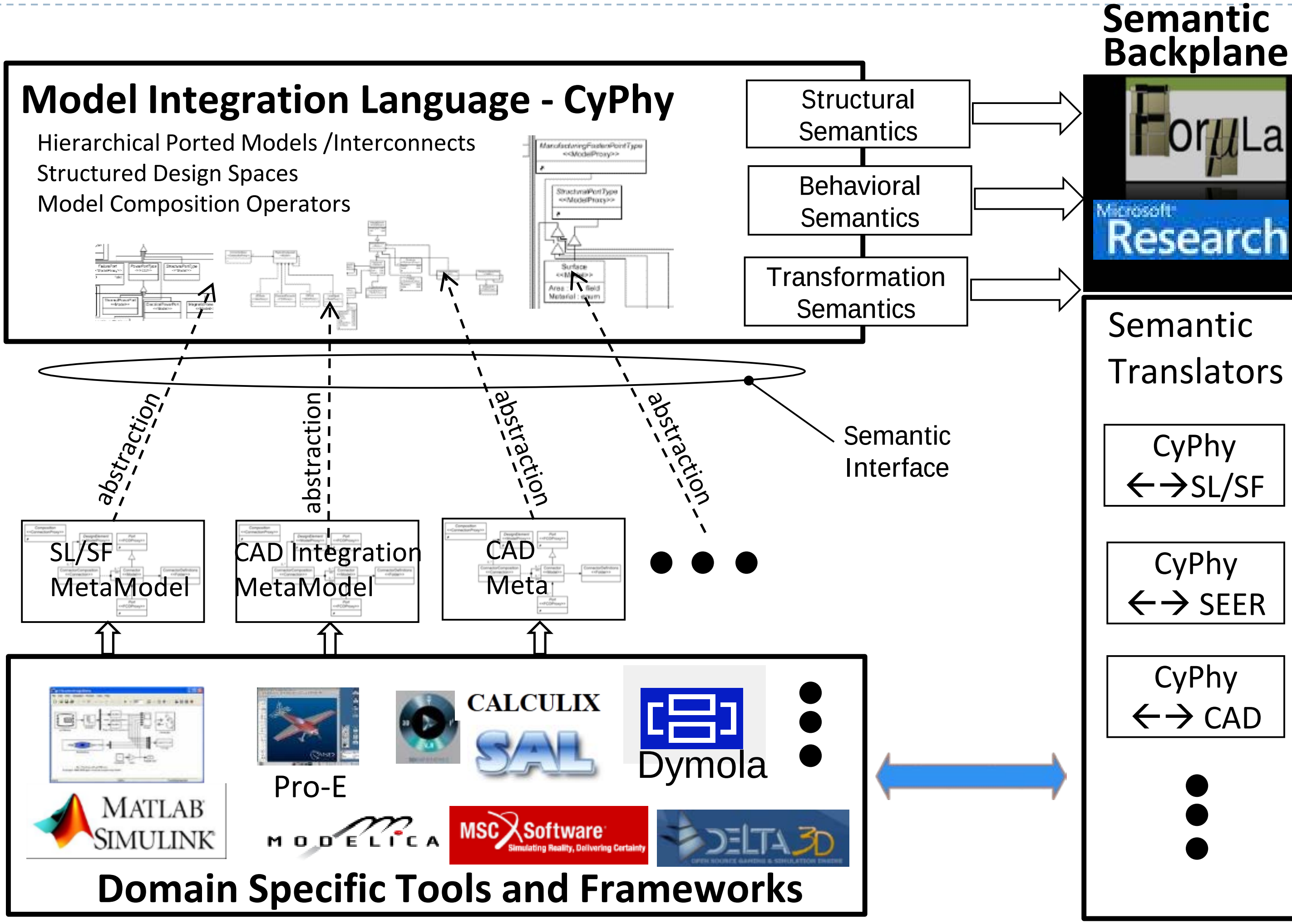
- What are the implications on tools?
- How about "my freedom of abstractions"?
- What is the language evolution trajectory?

First idea…


Functional Decomposition Model
Architecture Design Space Models
Models of the design space. Uses:
- Abstraction
- Containment
- Alternatives/options
- Parametric components
Functional decomposition hierarchy, derived from requirements
Design Space Exploration
Adaptability Evaluation
Complexity Evaluation
Verification/Validation
Design Model of a cyber-physical system. Uses abstraction and hierarchy.
Architecture Design Model
Concrete, domain-specific models of components and subsystems linked to the Architecture Design Model.
Refinement
Technology Constraints
META
Data/Control
Electrical
Thermal
Hydraulic
Mechanical
Structural
Domain-specific Subsystem/System Models
The Foundry
Domain-specific Component Models
Structural
Mechanical
Hydraulic
Thermal
Electrical
Data/Control

Conventional

The Case for Model Integration Languages…



Impact: Open Language Engineering Environment → Adaptability of Process/Design Flow → Accommodate New Tools/Frameworks, Accommodate New Languages

Model-Based Design

Domain Specific Design Automation Environments:

- *Automotive*
- *Avionics*
- *Sensors…*

Tools:

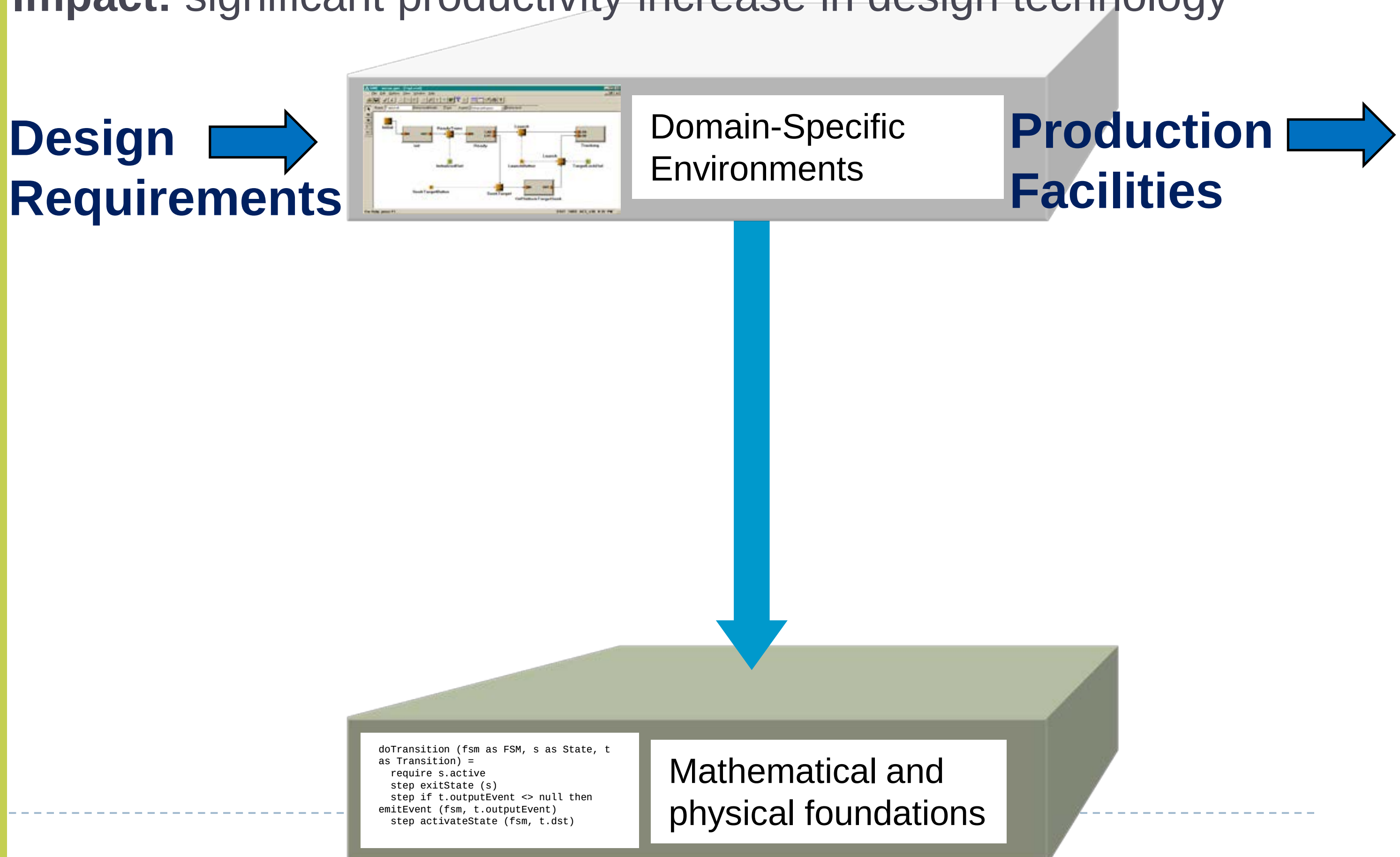
- *Modeling*
- *Analysis*
- *Verification*
- *Synthesis*

Challenges:

- *Cost of tools*
- *Benefit only narrow domains*
- *Islands of Automation*

Key Idea: Use models in domain-specific design flows and ensure that the final design models are rich enough to enable production of artifacts with sufficiently predictable properties.

Impact: significant productivity increase in design technology



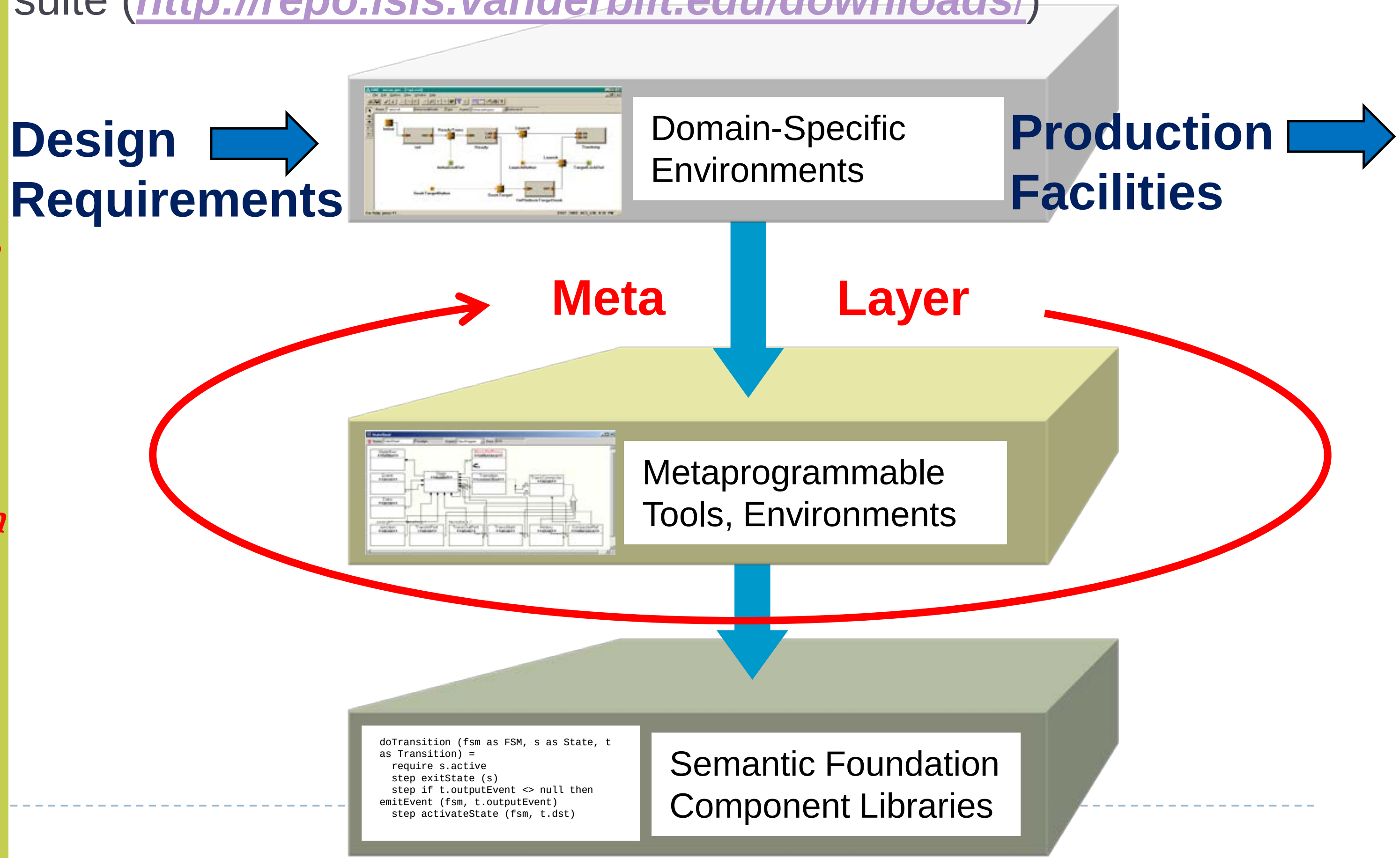
Metaprogrammable Design Tools
“Freedom of Abstractions”

Domain Specific Design Automation Environments:
- *Automotive*
- *Avionics*
- *Sensors…*

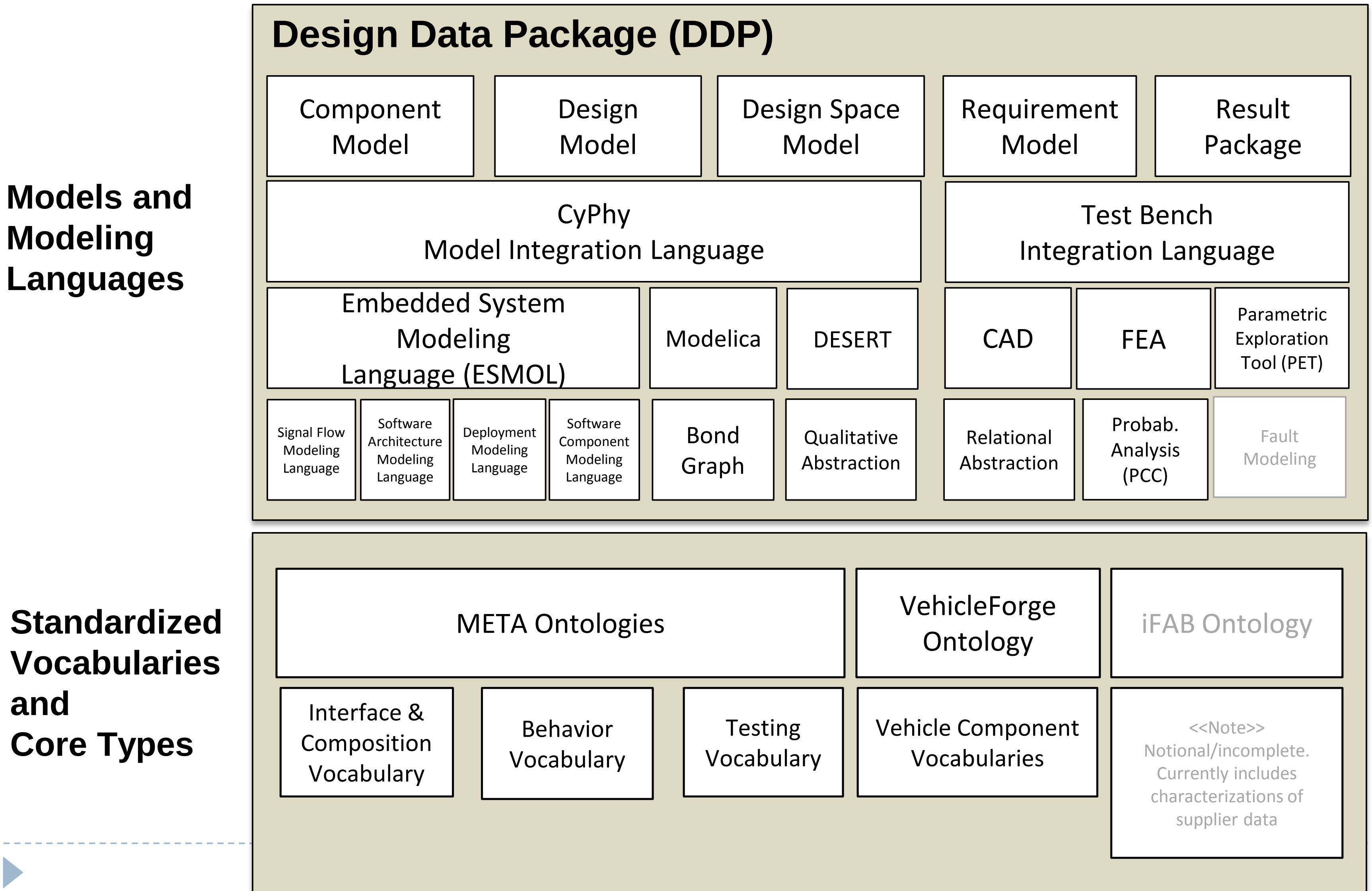
Metaprogrammable Tool Infrastructure
- *Model Building*
- *Model Transf.*
- *Model Mgmt.*
- *Tool Integration*

Explicit Semantic Foundation
- *Structural*
- *Behavioral*

Key Idea: Ensure reuse of high-value tools in domain-specific design flows by introducing a *metaprogrammable* tool infrastructure.
VU/ISIS implementation: Model Integrated Computing (MIC) tool suite (*http://repo.isis.vanderbilt.edu/downloads/*)



OpenMETA
Information Architecture



Summary of OpenMETA – Approach to Information Architecture

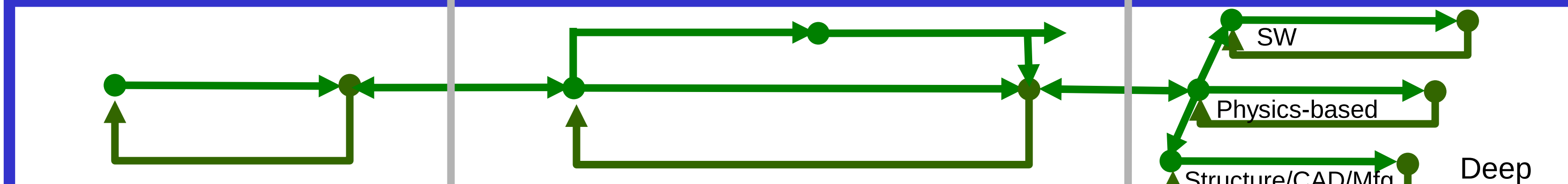
- Model-Integration Language: CyPhyML
- Use of Metaprogrammable tools (MIC Tool Suite of ISIS/Vanderbilt)
- Use of Semantic Integration (see later)

Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics

Design Flow

Architecture Design	Integrated Multi-physics/Cyber Design	Detailed Design
Modeling Exploration	Modeling Simulation V&V	Modeling Analysis



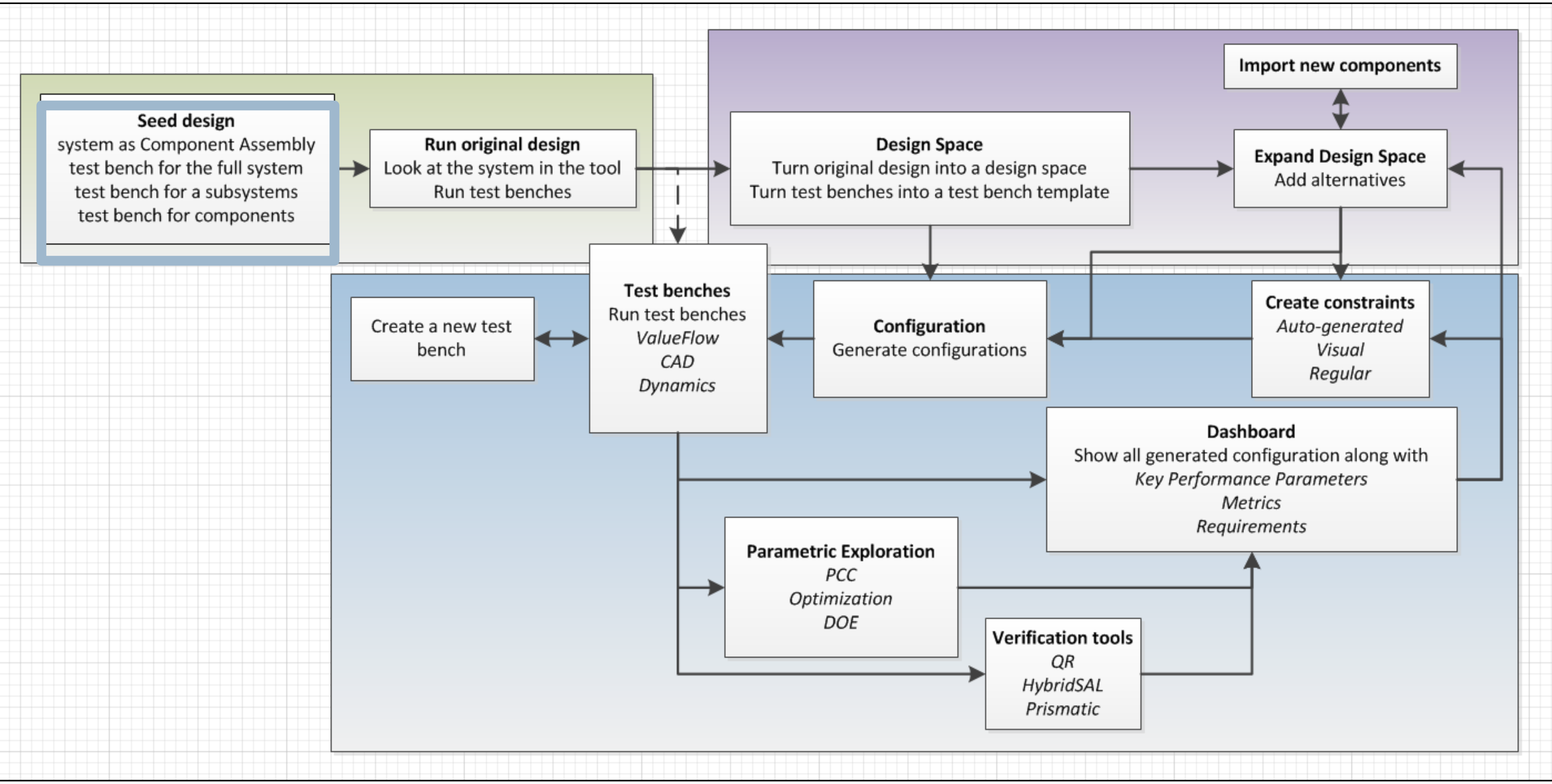
Rapid exploration	Exploration with integrated optimization and V&V	Deep analysis

• Design Space + Constraint Modeling
• Architecture Modeling
• Static Component Modeling (multi-physics)
	• Design Space + Behavioral Constraint Modeling
	• Architecture Modeling
	• Dynamics Modeling (multiple abstractions and multi-physics)
	• CAD/assembly modeling
	• Coarse Manufacturing Constraint Modeling
		• Architecture Modeling
		• Detailed Domain Modeling
		  – CAD
		  – FEA; thermal, fluid…
		  – Surrogate gen.
		• Detailed Manuf. Modeling
		• RT SW modeling

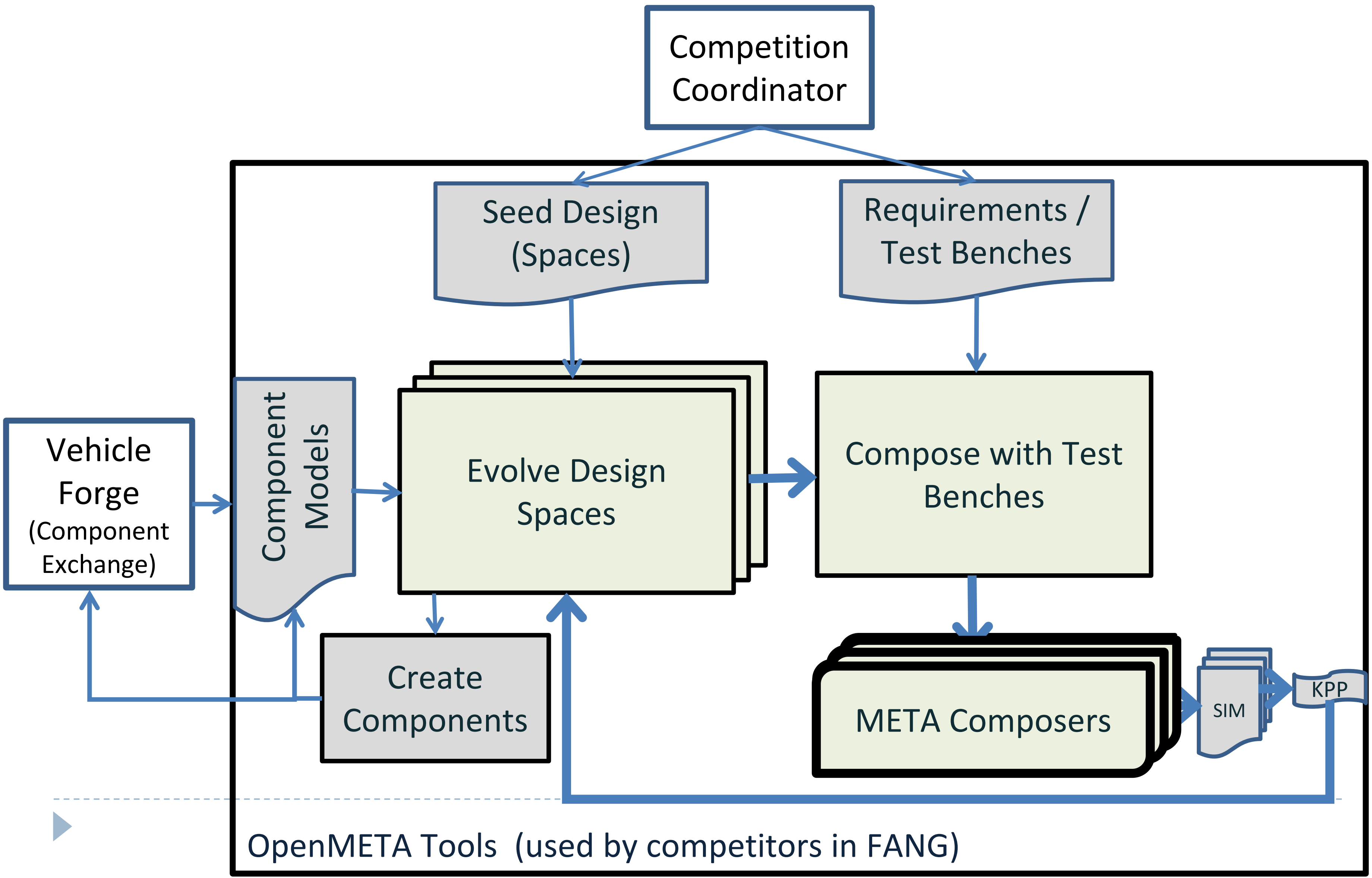
Design Flow Integration Challenges

- How to start the design process?
- How to help its convergence to a “good enough” solution?
- How to link all the tools?

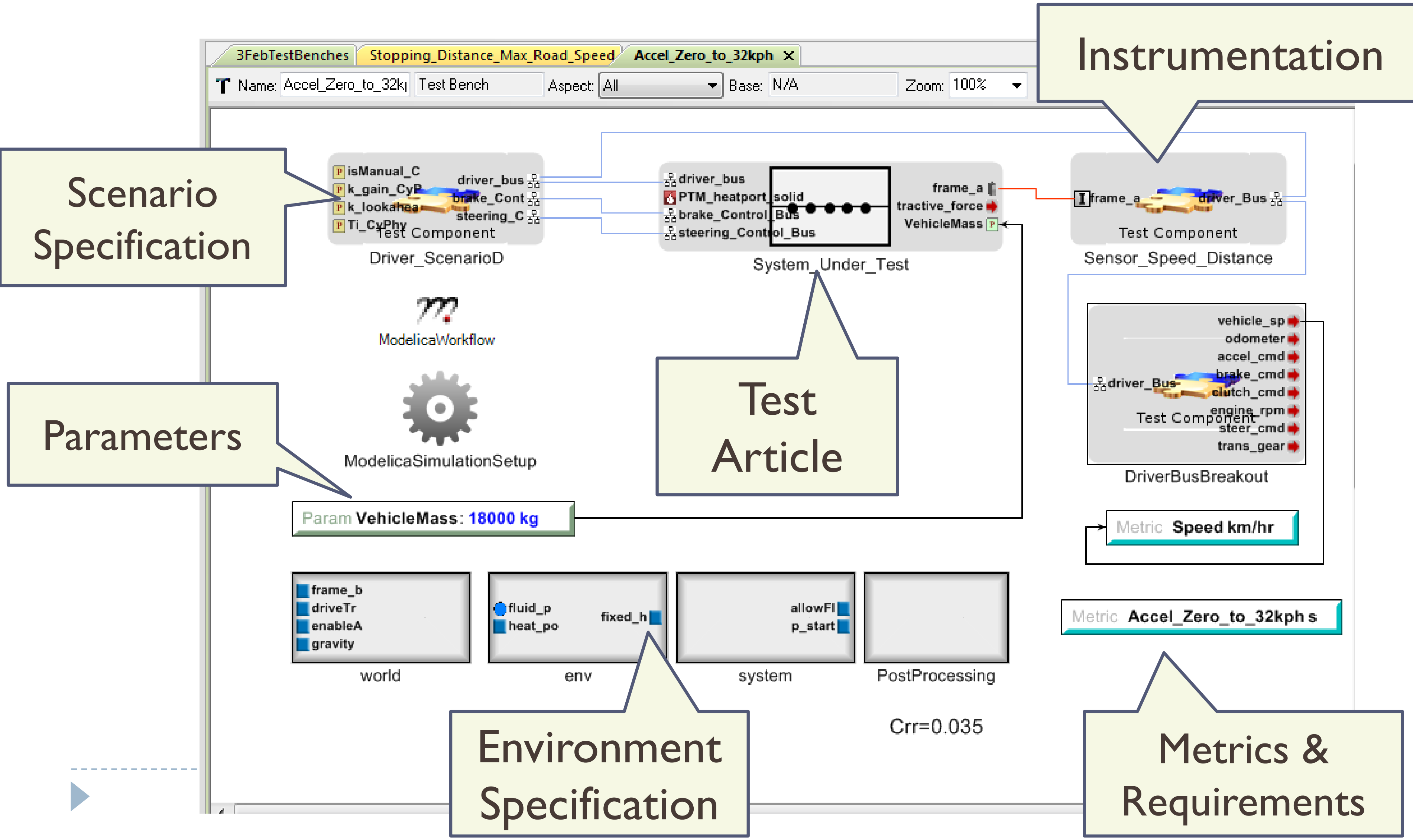
META Design Flow



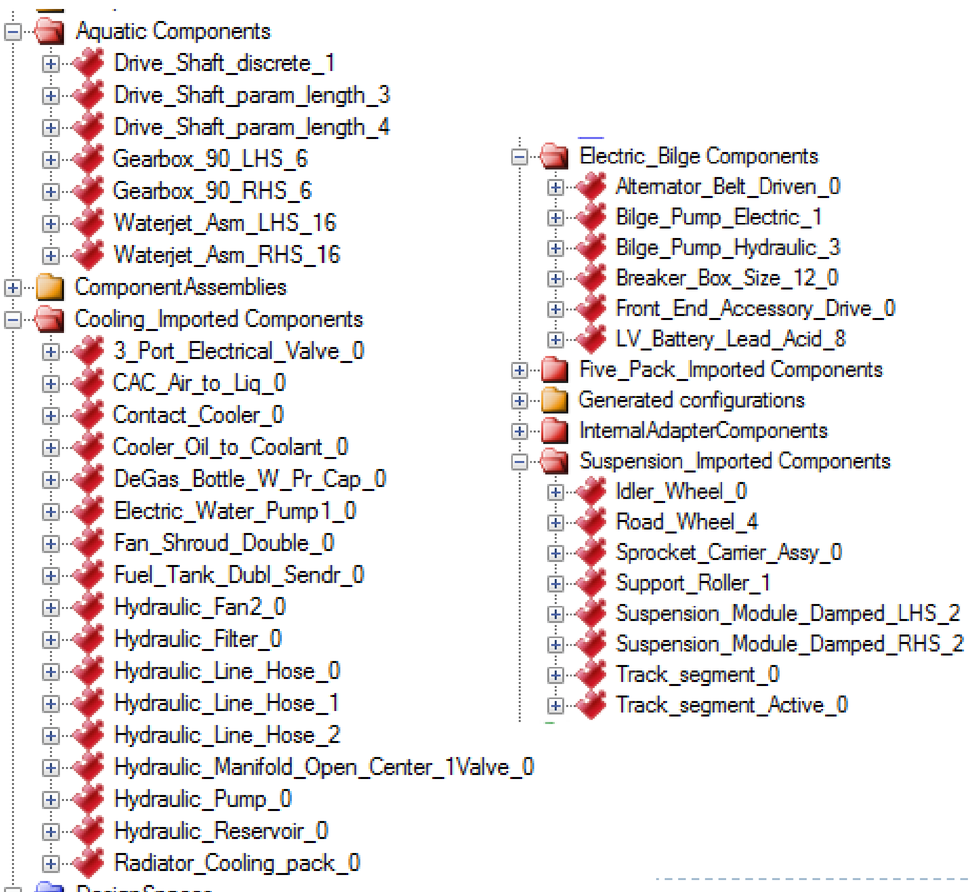
OpenMETA "Composers"

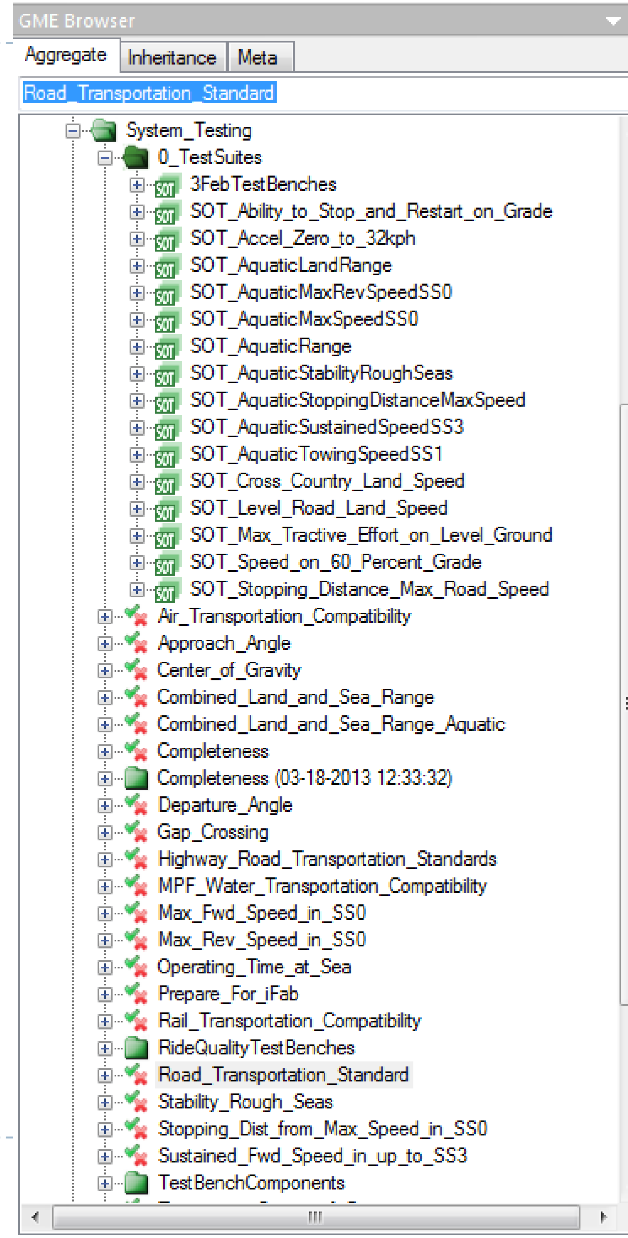


Executable Requirements and Test Bench Concepts

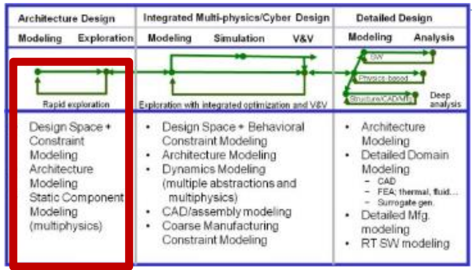


Example for Test Benches to Evaluate FANG Requirements





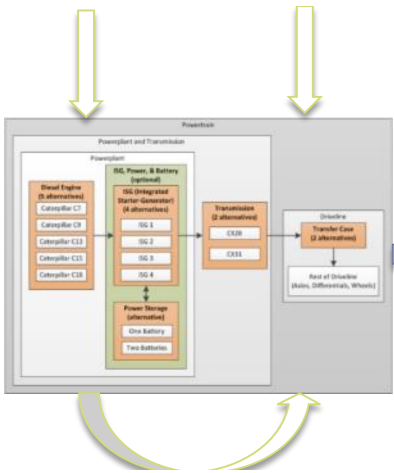
Architecture Exploration Using Interface Abstractions

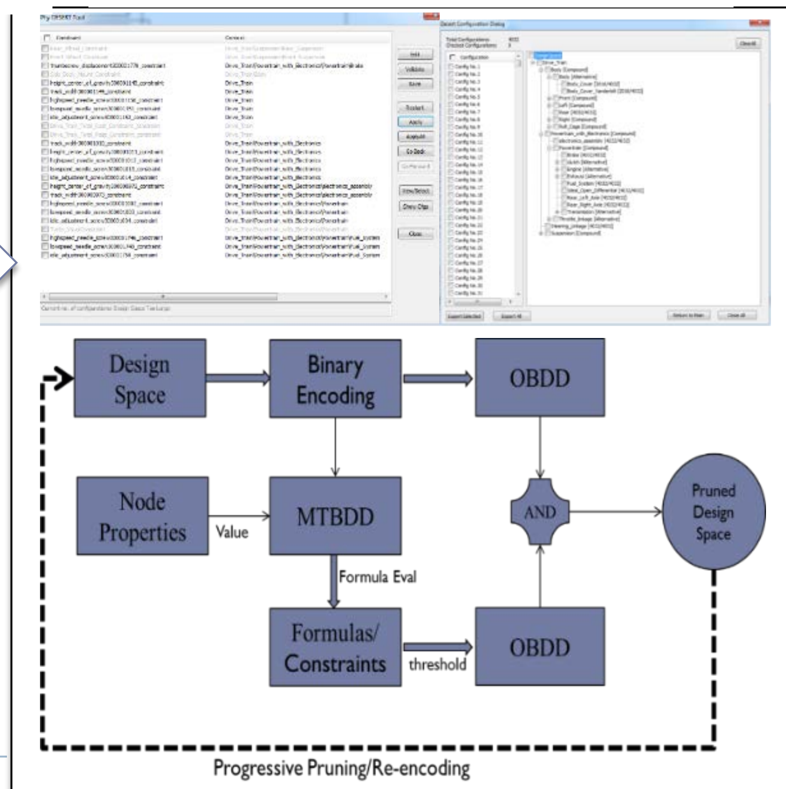


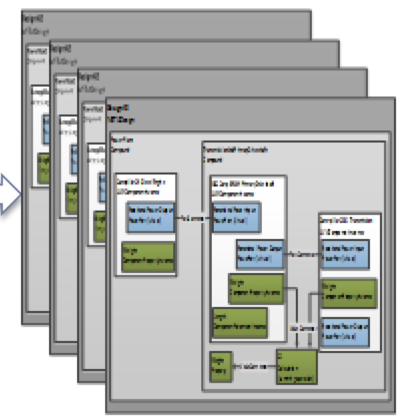
Components

Subsystem Design Spaces

Constraint-guided Design Space Pruning





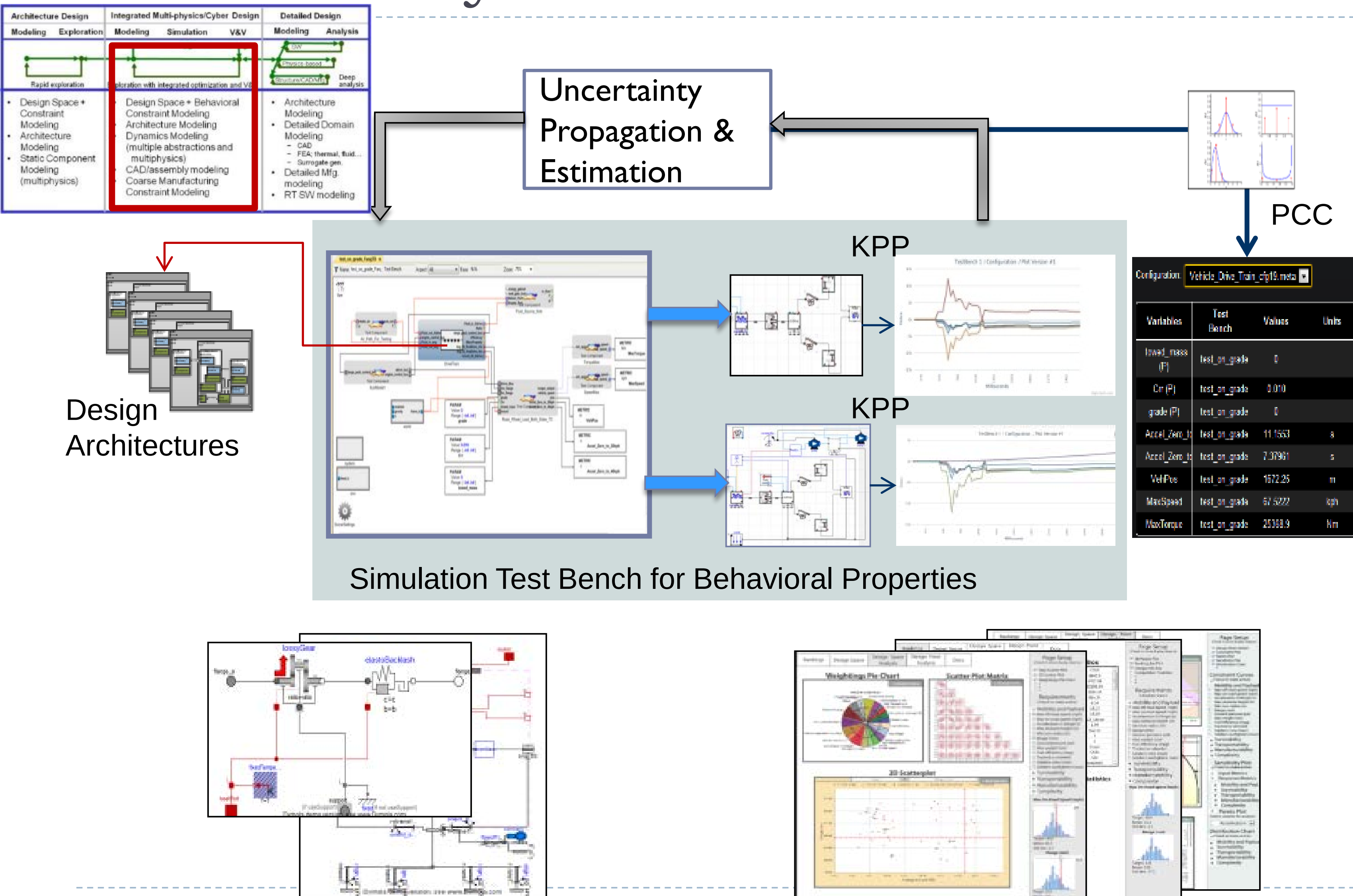


Design Space Evolution

Requirements, Design Rules → Constraints

Design Architectures

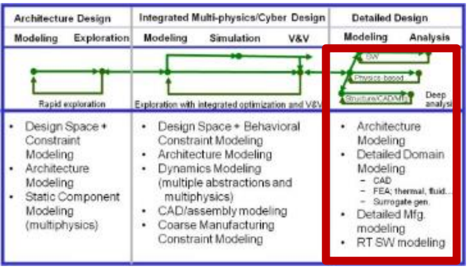
Design Space Exploration Using Multi-Fidelity ODEs

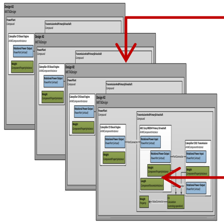


Multiple Fidelity Behavior Models

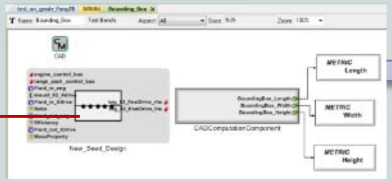
Multiple Physics Domains

Design Space Exploration Using Geometry and FEA





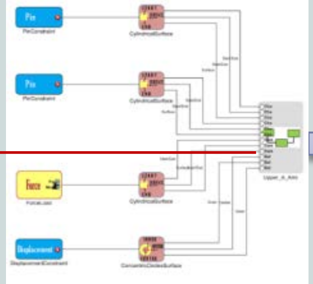
Design Architectures

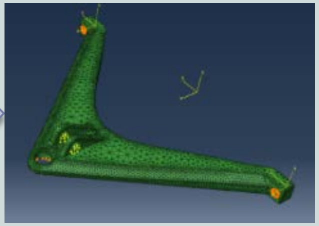


CAD Testbench for Physical Properties

KPP

1) Bounding box
2) Center of Gravity
3) Dimensions





KPP

FEA Testbench for Structural Properties

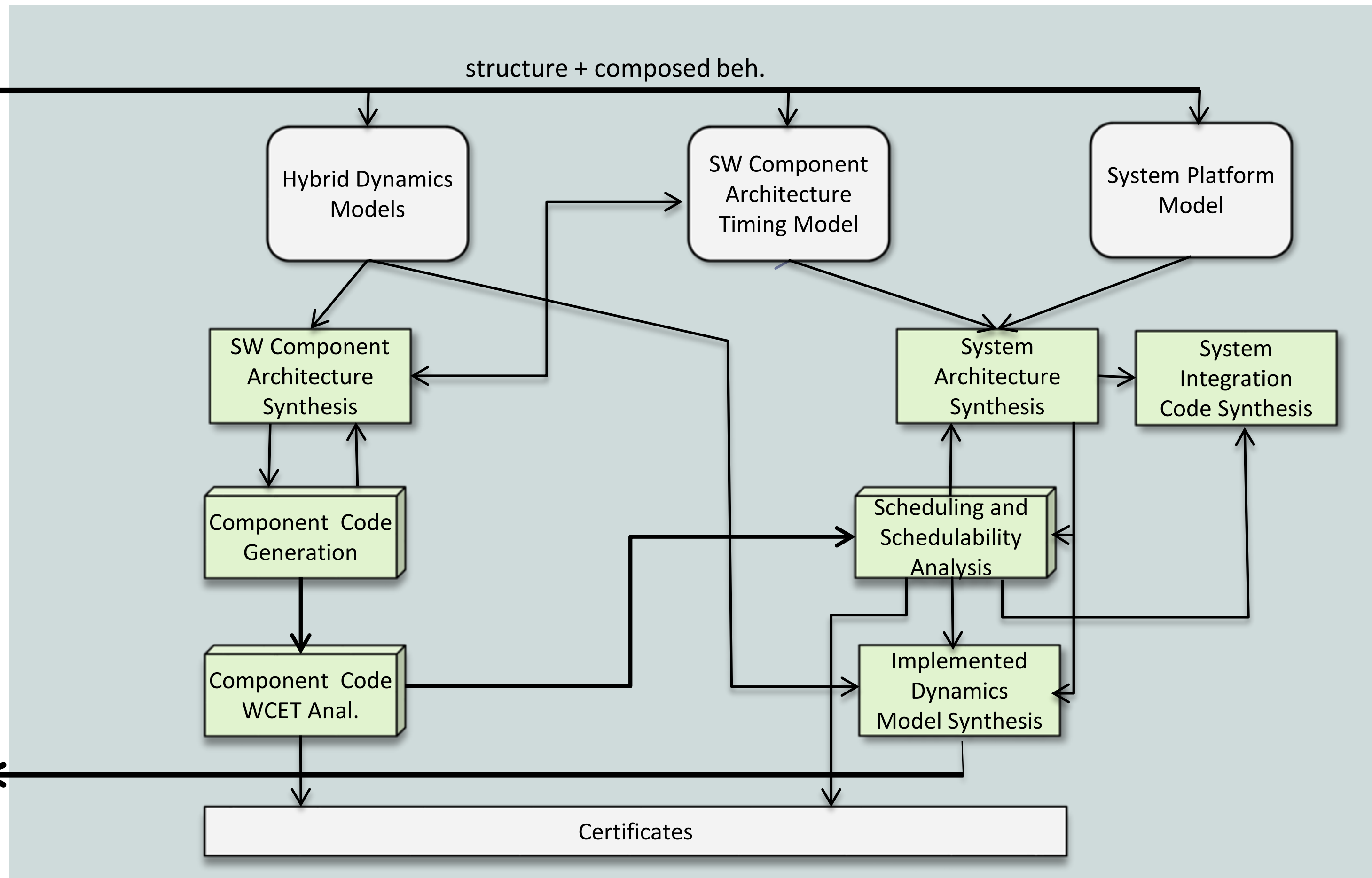
1) maximum shear stress,
2) maximum bearing stress,
3) maximum Von Mises stress
4) factor-of-safety

Configuration: Vehicle_Drive_Train_cfg19.meta

Variables	Test Bench	Values	Units
towed_mass (P)	test_on_grade	0	
Crr (P)	test_on_grade	0.010	
grade (P)	test_on_grade	0	
Accel_Zero_t	test_on_grade	11.1663	s
Accel_Zero_t	test_on_grade	7.37961	s
VehPos	test_on_grade	1672.25	m
MaxSpeed	test_on_grade	67.5222	kph
MaxTorque	test_on_grade	25388.9	Nm

OpenMETA Software Tool Chain

Design Architectures with ideal component dynamics

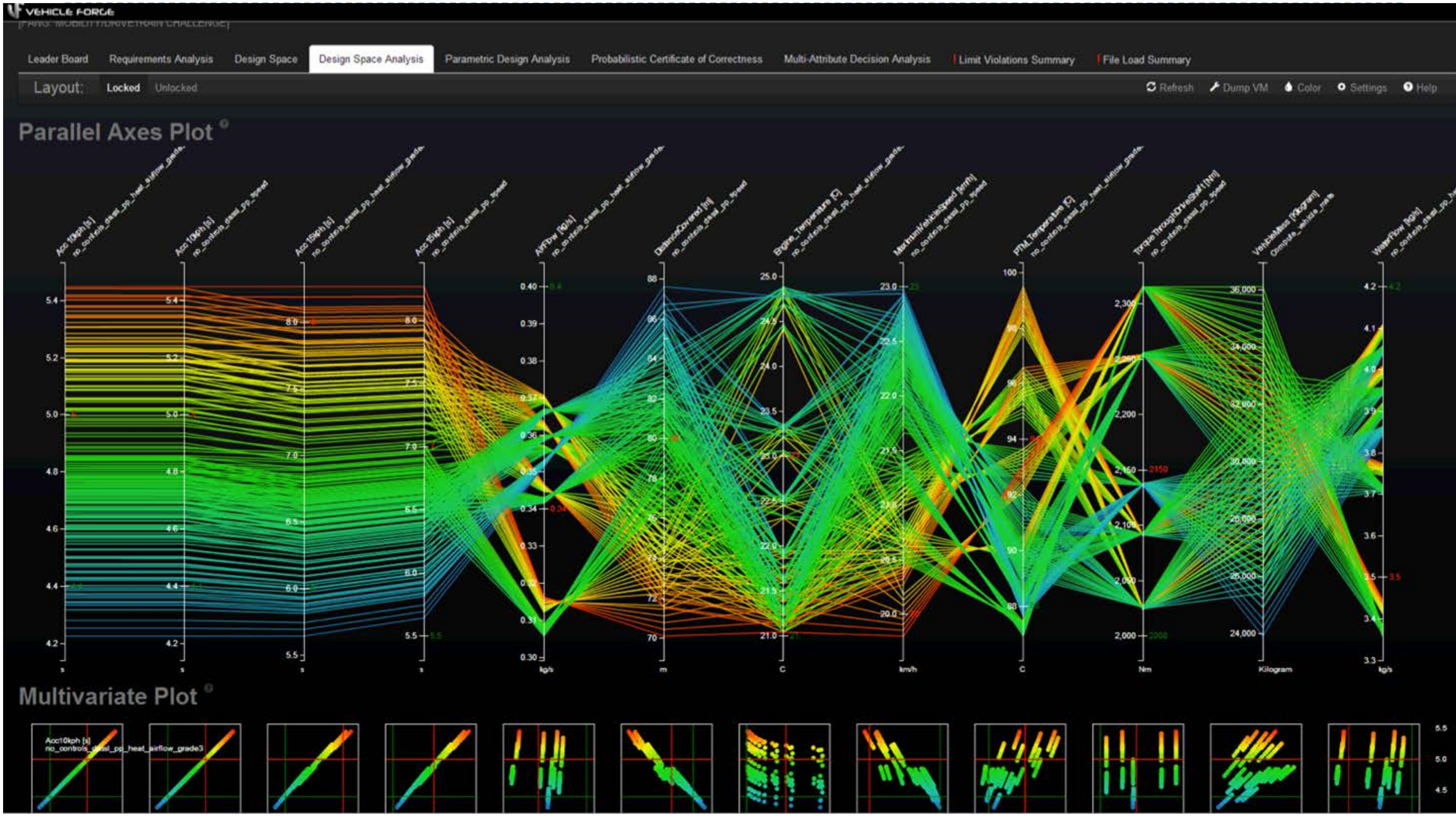


Design Architectures with **deployed component dynamics**

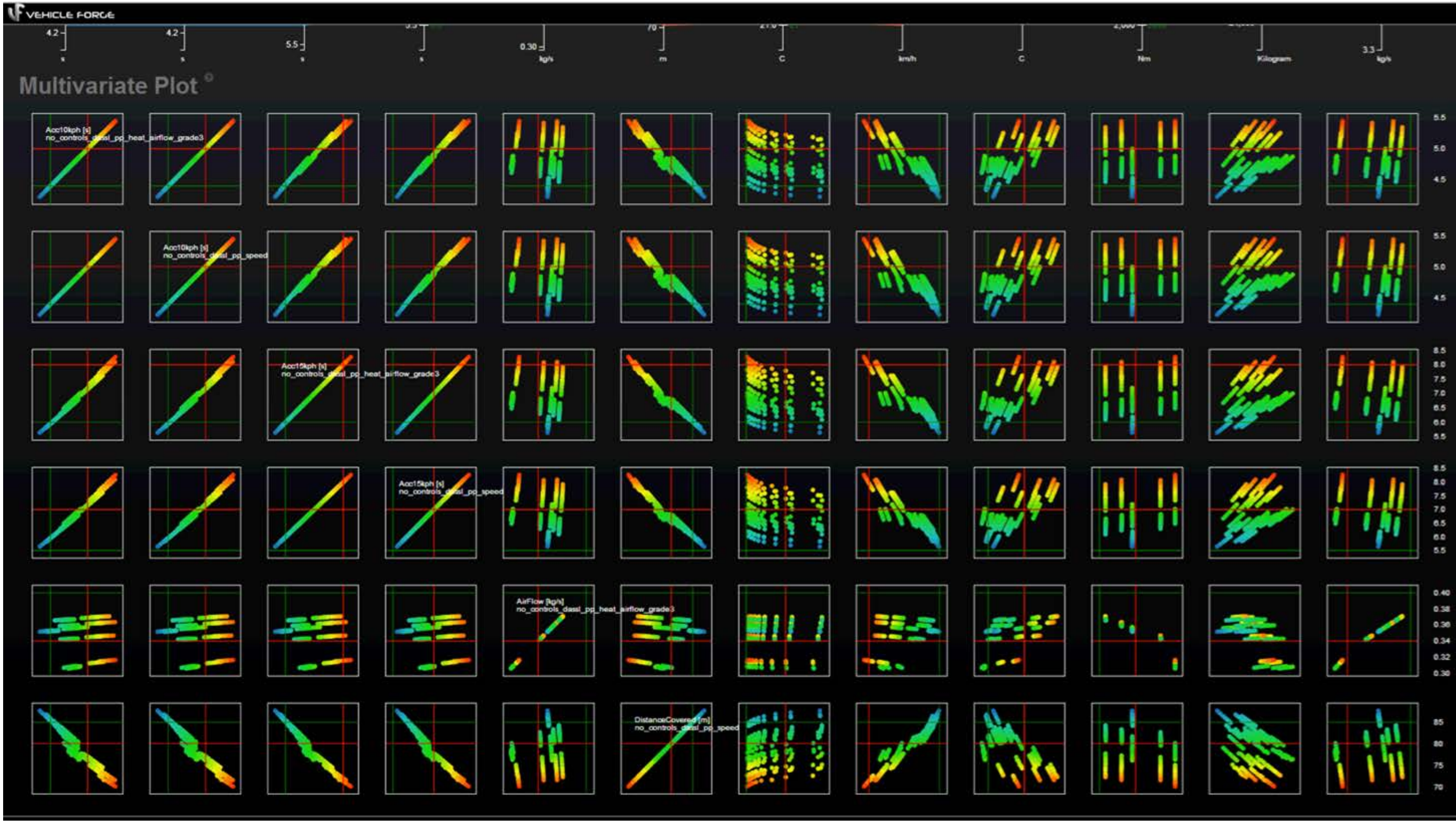
- Time-triggered Model of Computation
- TT bus (or emulated TT bus)

- Event-triggered Model of Computation
- CAN bus

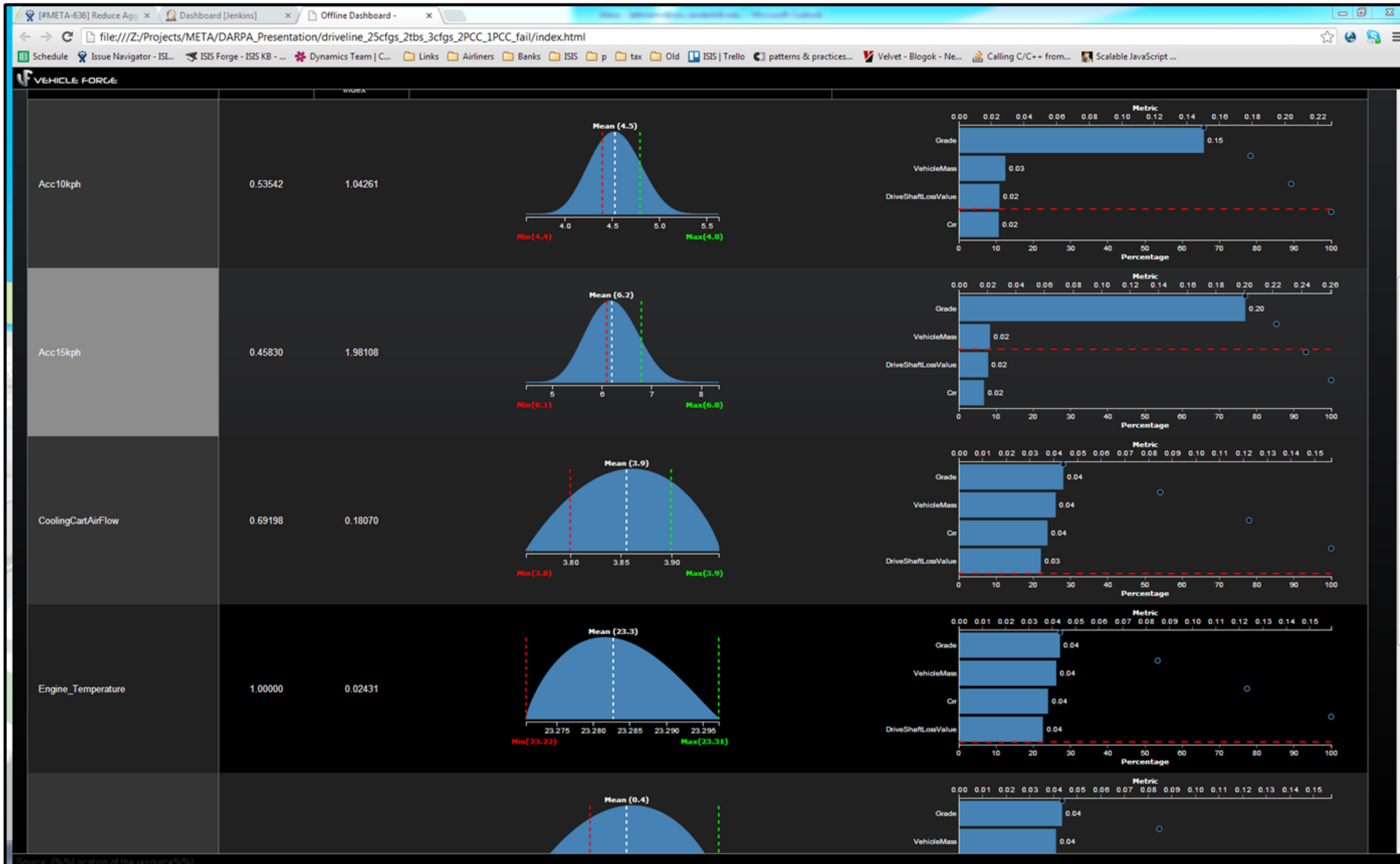
Design Space Evaluation Visualization



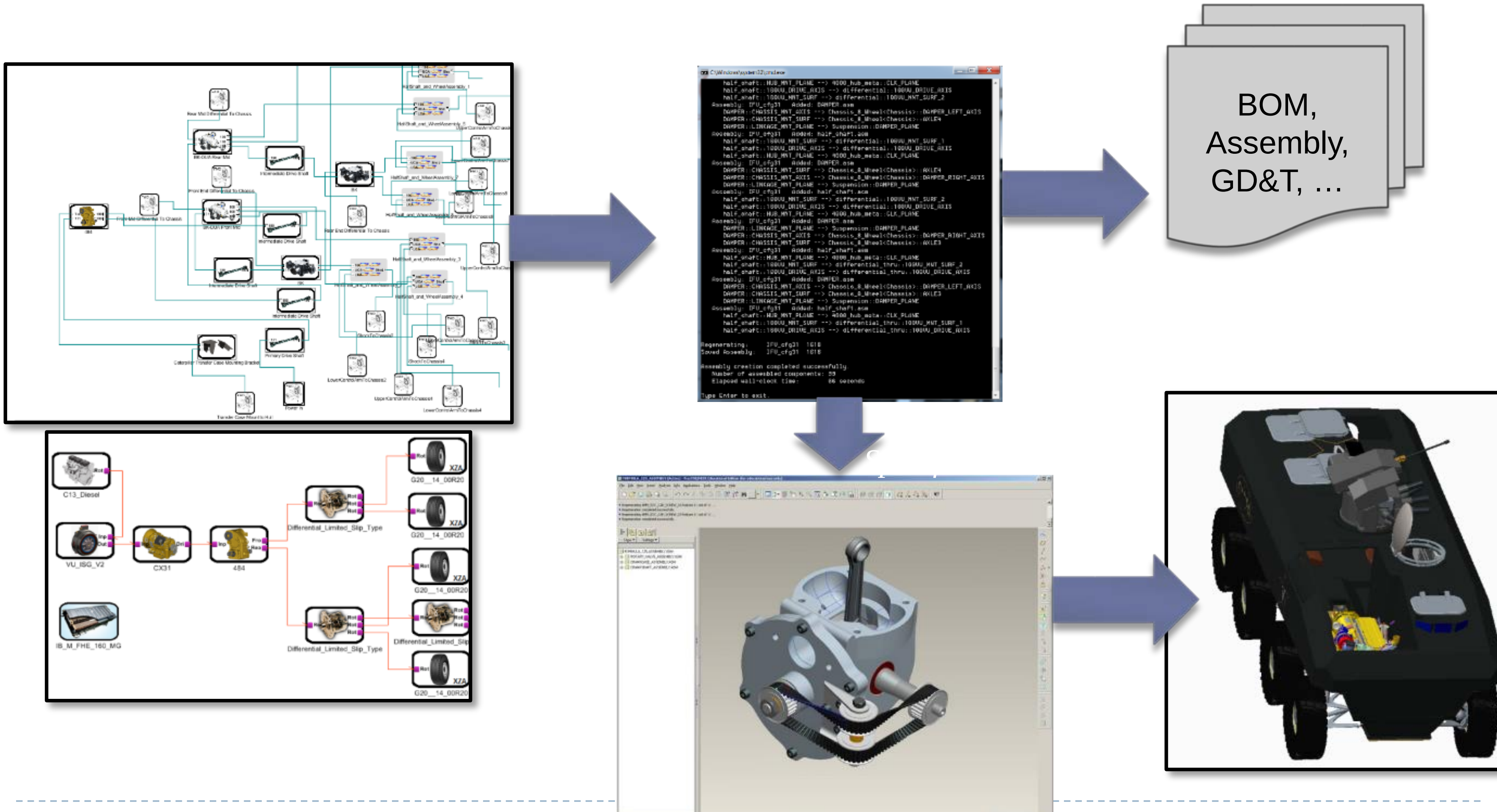
Pairwise Visualization of Metrics



Probabilistic Certificates of Correctness (PCC)



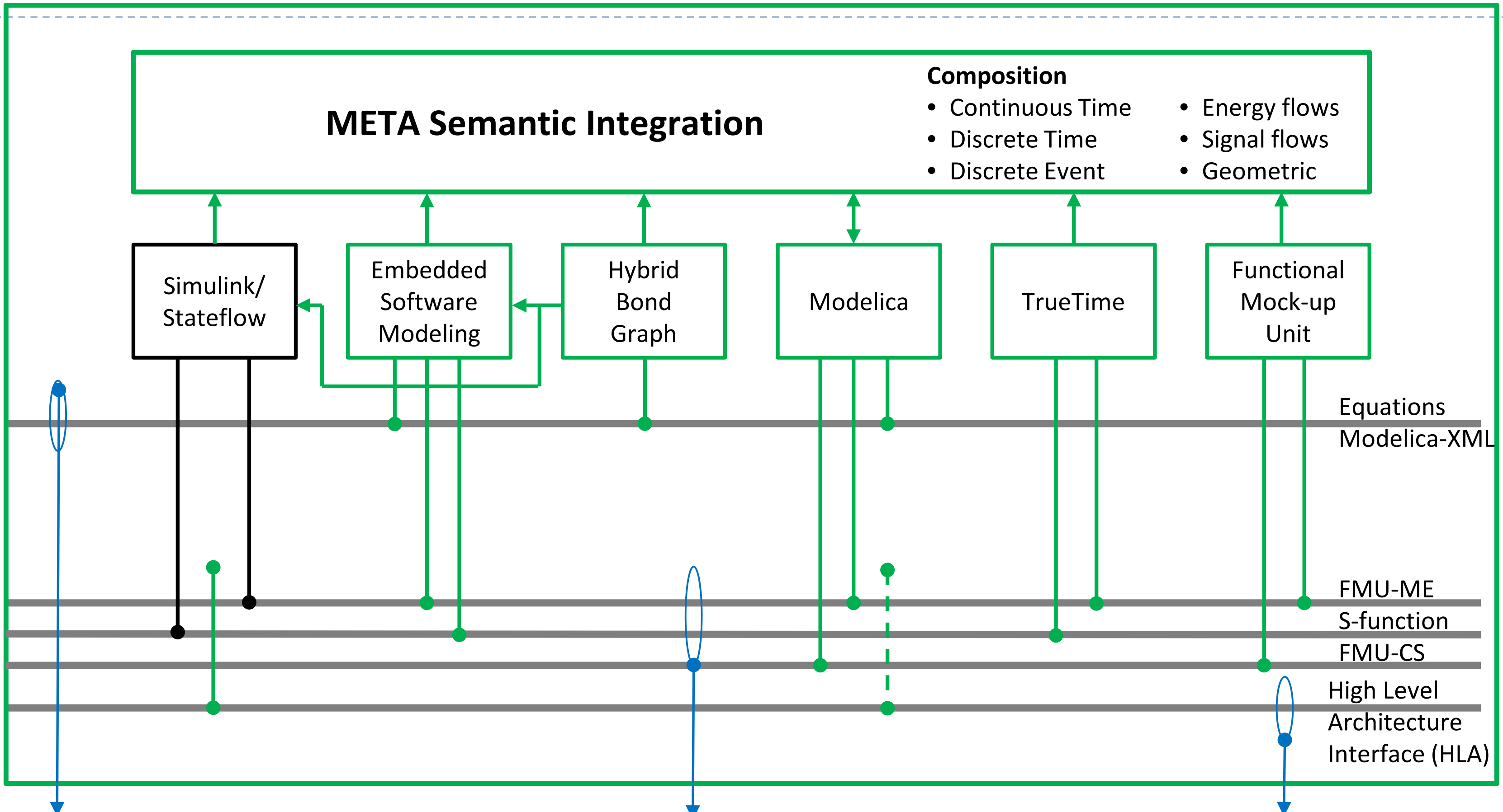
Geometric Reasoning:
CAD Assembly Composition



Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics

The Need for Formal Semantics

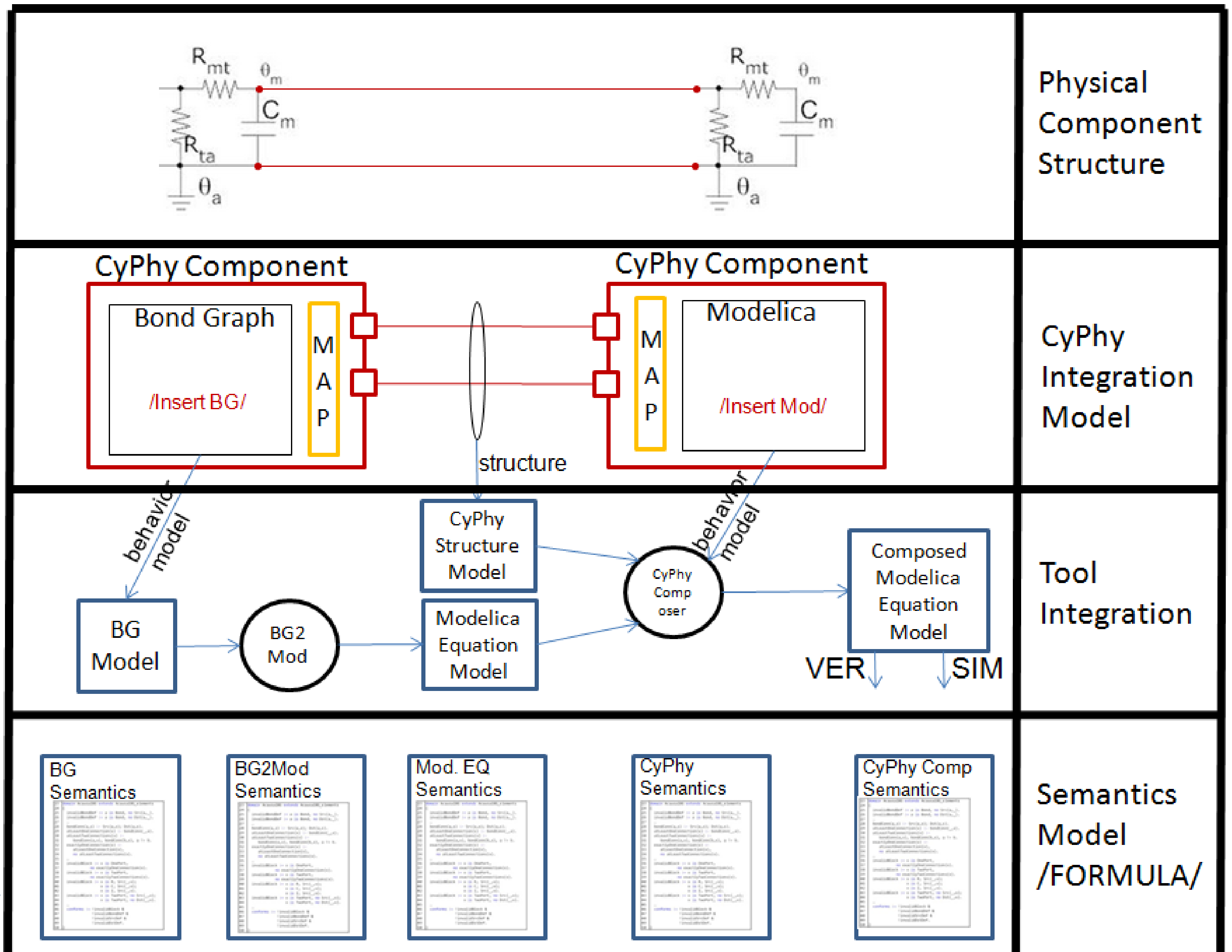


Formal Verification
- Qualitative reasoning
- Relational abstraction
- Model checking
- Bounded model checking

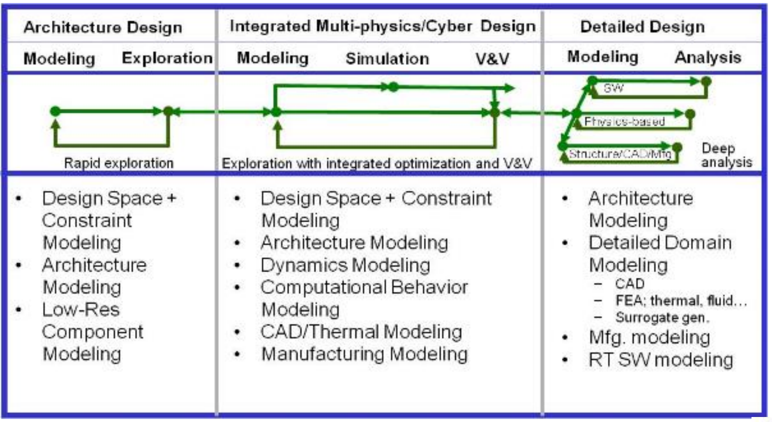
Stochastic Co-Simulation
- Open Modelica
- Dymola

Distributed Simulation
- NS3
- OMNET
- Delta-3D
- CPN

Concept of "Semantic Integration"

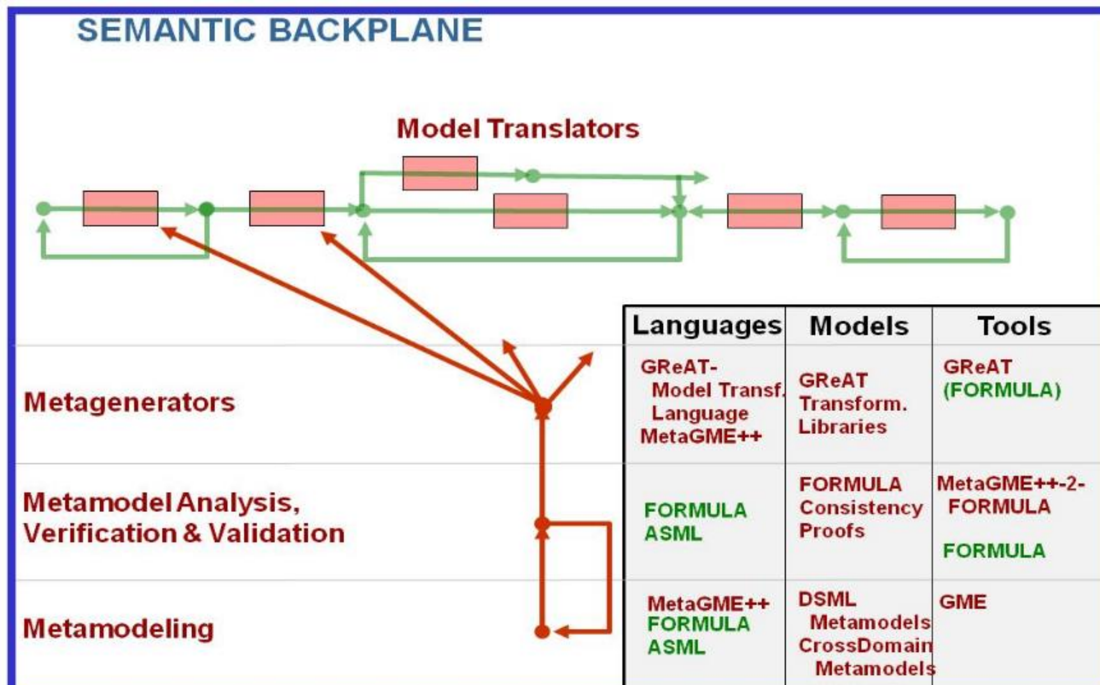


Cost of Model Integration Languages: “Semantic Backplane”



- Tight integration from architecture modeling to physics-based modeling
- Integrated multi-physics modeling
- Bridging gap between computation and physics domains
- Tight integration of structural and behavioral models
- Emphasis is on automation and scaling
- *META tool suite designed for rapid evolution and extensibility*

- Agility is achieved by introducing a ***Semantic Backplane***
- Semantic Backplane is implemented via
  - tools and methods for modeling language specification, validation, and transformations
  - tools and methods for explicit representation of and computation with well-defined structural and behavioral semantics
  - metamodel and transformation libraries
  - metaprogrammable tools



FORMULA: http://research.microsoft.com/formula

Convergence to a Formal Framework: FORMULA

- History: Foundations for Embedded Systems NSF ITR; Ethan Jackson at VU 2005-2008
- Microsoft Research (Bellevue & Aachen); Satisfiability Modulo Theory Solver (Z3); VS distribution

http://research.microsoft.com/formula

- Foundation: Algebraic Data Types (ADT) and First-order logic with fixpoints (FPL)
- Parameterized with background theories (bit vectors, term algebras, etc.
- Semantics is defined by constraint logic programming (CLP)
- Evolving structures; temporal logic

Formalization of Semantics - Structural

<u>**Structural Semantics**</u> defines modeling domains using Algebraic Data Types and First-Order Logic with Fixpoints. Semantics is specified by Constraint Logic Programming.

Use of structural semantics:

- Conformance testing: $x \in D$
- Non-emptiness checking: $D(Y,C) \neq \{nil\}$
- DSML composing: $D_1 * D_2 | D_1 + D_2 | D' \text{ includes } D$
- Model finding: $S = \{s \in D | s| = P\}$
- Transforming: $m' = T(m); m' \in X; m \in Y$

Formalization of Semantics – Behavioral

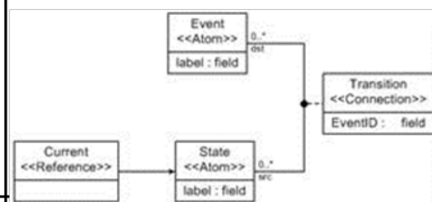
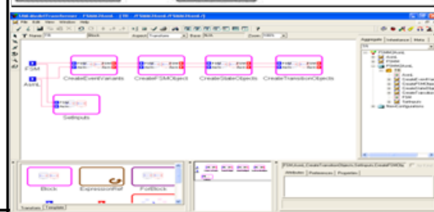
Behavioral Semantics defines exhibited behavior of models by

1. Specifying a translation to a domain with well-understood *operational semantics*
2. Specifying a translation to a mathematical domain defining behaviors *denotationally* (e.g. symbolic DAEs)

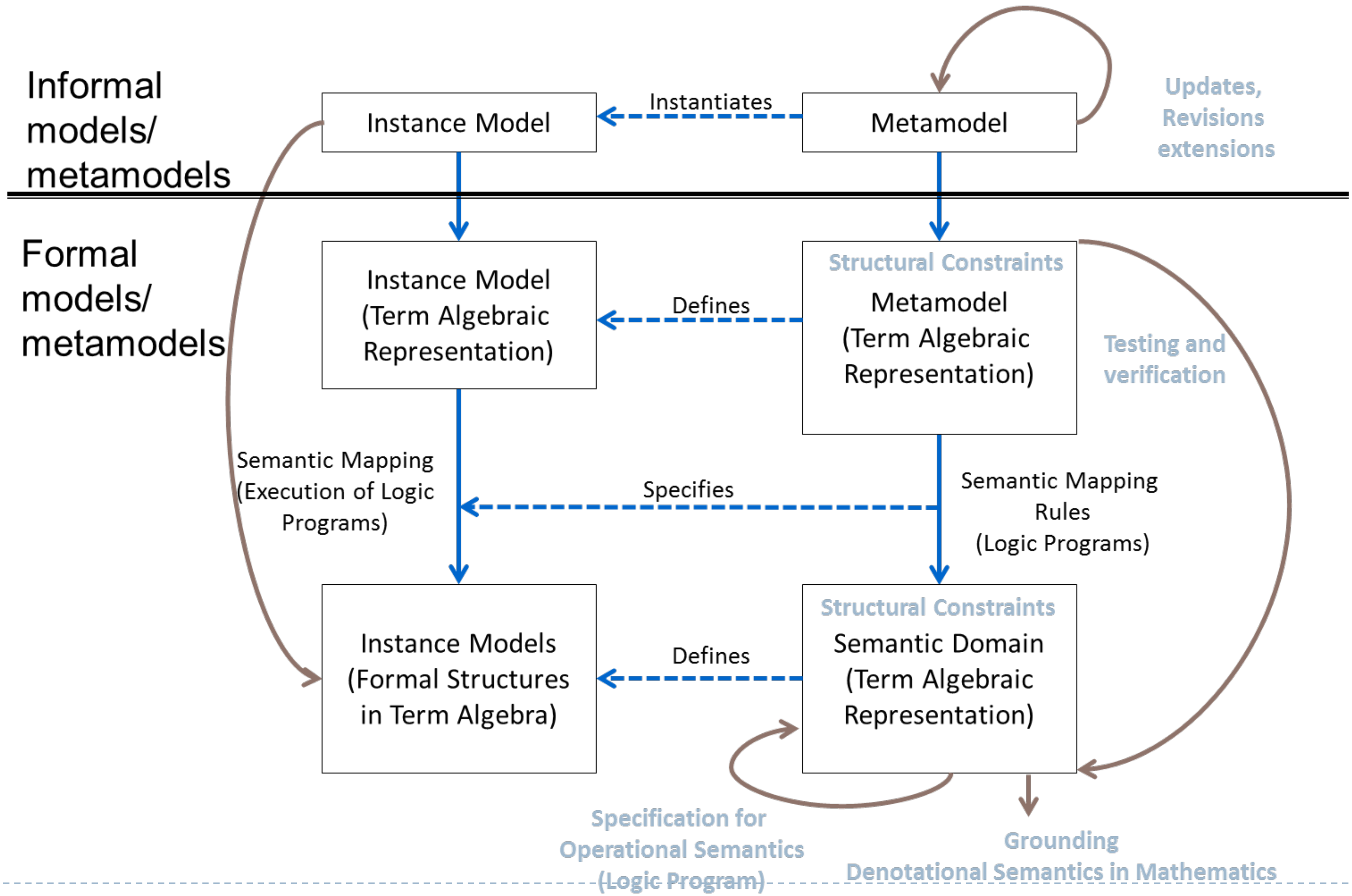
Use of Behavioral Semantics Specifications:

- Validating/understanding behaviors via simulation
- Generating behaviors using "reference semantics" and testing tools w.r.t. reference semantics
- Invariance checking
- Formalization → first steps toward proofs
- Tracking dependences in tool suites

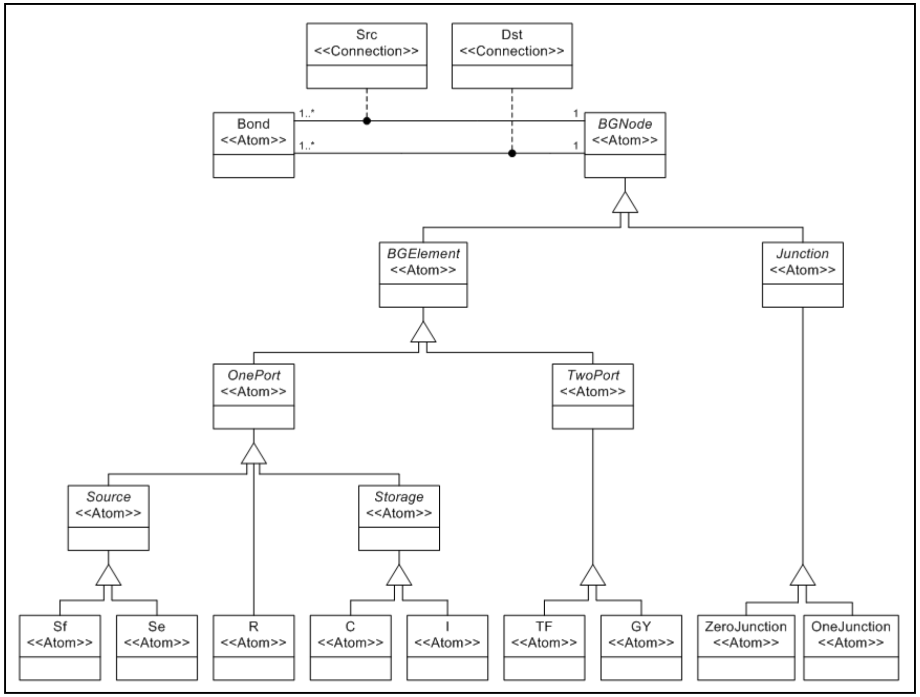
Layers of the Semantic Backplane

Functions	(Meta)Models	Languages	Tools	Role
Metamodeling		MetaGME	• GME • MetaGME-2-Formula	• DSML spec. • Constraint Checking • Metaprog.
Transformation Modeling		UMTL	• GReAT • UDM	• Transf. spec. • Compiling spec to transformer
Formal Metamodeling	domain DFA { primitive Event ::= (lbl: Integer). primitive State ::= (lbl: Integer). [Closed(src, trg, dst)] primitive Transition ::= (src: State, [Closed(st)] primitive Current ::= (st: State).	Formula (MSR)	• Domain Comp. • Trace Gen.	• Metamod. checking • Example gen. • Semantic units
Formal Transformation Modeling	transform Step<fire: in1.Event> from DFA out1.State(x) :- in1.State(x). out1.Event(x) :- in1.Event(x). out1.Transition(s, e, sp) :- in1.Trans out1.Current(sp) :- in1.Current(s), ir out1.Current(s) :- in1.Current(s), fai }		• Semantic Anchoring	• Semantics for complex DSMLs • Composition

Structure of the Semantic Backplane



Metamodel and Formal Metamodel - ADTs



Metamodel of a simplified acausal Bond Graph DSML

```
domain AcausalBG_elements
{
  primitive Sf ::= (id: String).
  primitive Se ::= (id: String).
  primitive R  ::= (id: String).
  primitive C  ::= (id: String).
  primitive I  ::= (id: String).
  primitive TF ::= (id: String).
  primitive GY ::= (id: String).
  primitive ZeroJunction ::= (id: String).
  primitive OneJunction  ::= (id: String).
  Source    ::= Sf + Se.
  Storage   ::= C + I.
  OnePort   ::= Source + R + Storage.
  TwoPort   ::= TF + GY.
  BGElement ::= OnePort + TwoPort.
  Junction  ::= ZeroJunction + OneJunction.
  BGNode    ::= BGElement + Junction.
  primitive Bond ::= (id: String).
  [Closed] primitive Src ::= (Bond,BGNode).
  [Closed] primitive Dst ::= (Bond,BGNode).
}
```

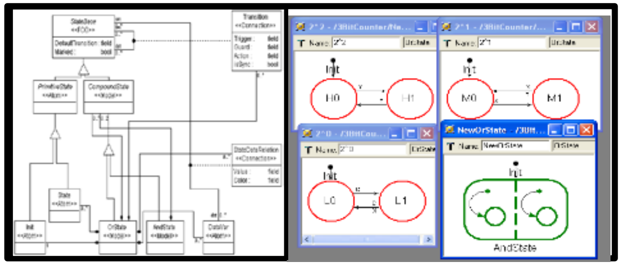
Formal metamodel of a simplified Bond Graph DSML

Part of Structural Semantics for acausal Bond Graphs

```
domain AcausalBG extends AcausalBG_elements
{
  invalidBondDef := a is Bond, no Src(a,_).
  invalidBondDef := a is Bond, no Dst(a,_).
  …
  bondConn(a,x) :- Src(a,x); Dst(a,x).
  atLeastOneConnection(x) :- bondConn(_,x).
  atLeastTwoConnections(x) :-
     bondConn(a,x), bondConn(b,x), a != b.
  exactlyOneConnection(x) :-
     atLeastOneConnection(x),
     no atLeastTwoConnections(x).
  …
  invalidBlock := x is OnePort,
                no exactlyOneConnection(x).
  invalidBlock := x is TwoPort,
                no exactlyTwoConnections(x).
  invalidBlock := x is R, Src(_,x);
                  x is C, Src(_,x);
                  x is I, Src(_,x).
  invalidBlock := x is TwoPort, no Src(_,x);
                  x is TwoPort, no Dst(_,x).
  …
  conforms := !invalidBlock &
              !invalidBondDef &
              !invalidSrcDef &
              !invalidDstDef.
}
```

- Structural semantics is composed of constraints on model structure
- Modeling tools need to check constraints during modeling
- A well-formed model can be mapped into some behavior

Specifying Behavioral Semantics



$$D(Y,C)=\{r \in R_Y \mid r \models C\}$$

```
domain AcausalBG_elements
{
  primitive Sf ::= (id: String).
  primitive Se ::= (id: String).
  primitive R  ::= (id: String).
  //…
primitive TF ::= (id: String).
  primitive GY ::= (id: String).
  primitive ZeroJunction ::= (id: String).
  primitive OneJunction  ::= (id: String).
  Source   ::= Sf + Se.
  //..
}
```

$$[\ ]: R_Y \mapsto R_{Y'}$$

```
transform BG_DenotationalSemantics
  from in1::AcausalBG
  to out1::DAEquations
{
  Eq(e_a,p_x) :- x is Se, Src(a,x).
  Eq(f_a,p_x) :- x is Sf, Src(a,x).
  Eq(e_a, Mul(p_x,f_a)) :- x is R, Dst(a,x).
  DiffEq(e_a, Mul(Inv(p_x),f_a)) :-
    x is C, Dst(a,x).
//…
}
```

$$D(Y',C')=\{r \in R_{Y'} \mid r \models C'\}$$

$$[\ ]: R_{Y'} \mapsto R_{Y''}$$

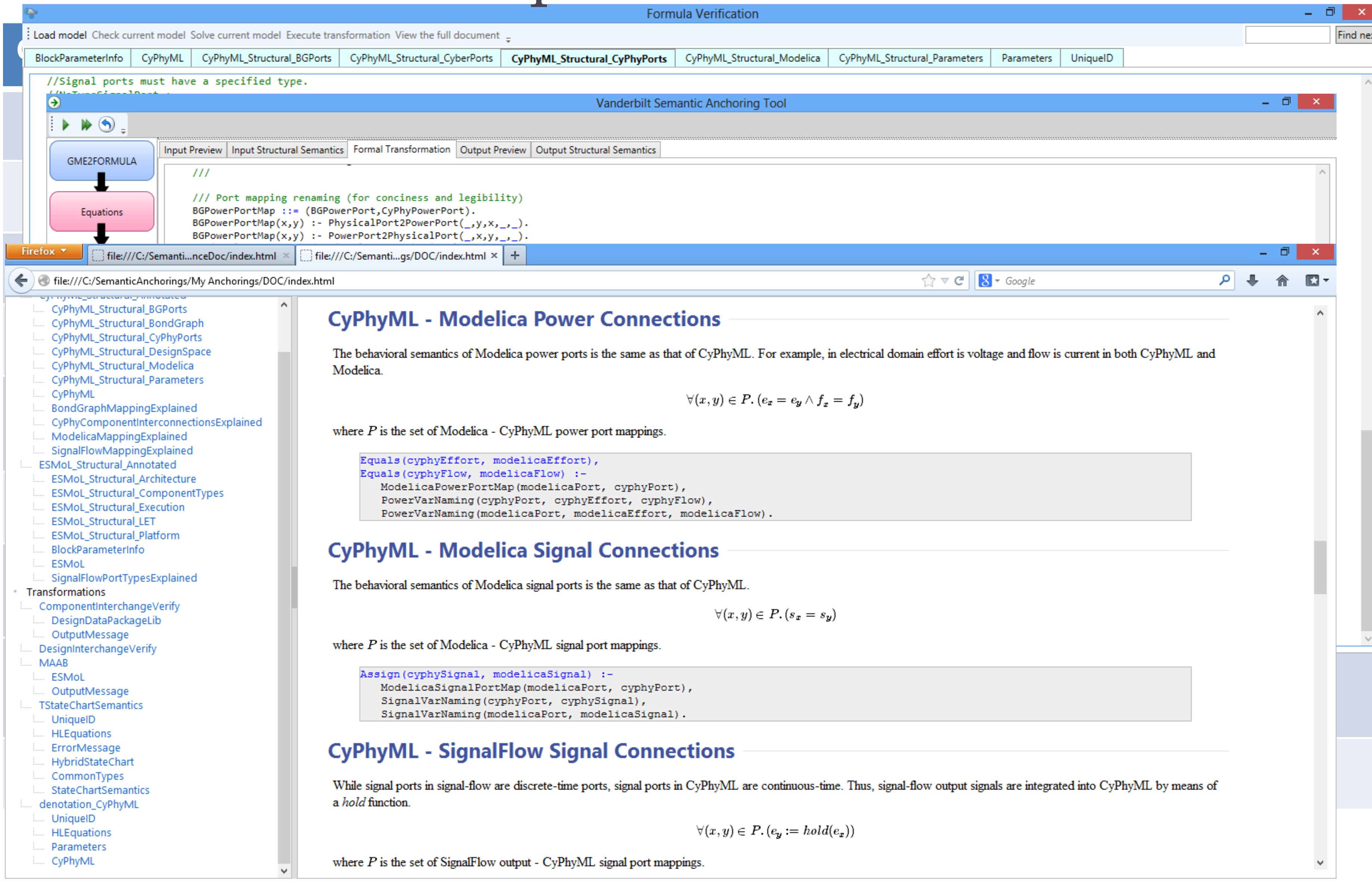
```
domain DAEquations
{
  primitive Variable ::=
    (name: String, id: String).
  primitive Param ::= (id: String).
  primitive Neg ::= (Term).
  primitive Inv ::= (Term).
  //..
  Term ::= Variable + Param + Neg + Inv + Mul + Sum.
  primitive Eq ::= (Variable, Term).
  primitive DiffEq ::= (Variable, Term).
  primitive SumZero ::= (Sum).
  Equation ::= Eq + DiffEq + SumZero.
}
```

Operational Behavioral Semantics for Finite Automata

```
domain DFA {
  primitive Event ::= (lbl: Integer).
  primitive State ::= (lbl: Integer).
  primitive Transition ::= (src: State, trg: Event, dst: State).
  primitive Current    ::= (st: State).
  nonDeterTrans := Transition(s, e, sp), Transition(s, e, tp), sp != tp.
  conforms      := !nonDeterTrans.
}
```

```
transform Step<fire: in1.Event>  from in1::DFA  to out1::DFA
 {
  out1.State(x) :- in1.State(x).
  out1.Event(x) :- in1.Event(x).
  out1.Transition(s, e, sp) :- in1.Transition(s, e, sp).
  out1.Current(sp) :- in1.Current(s), in1.Transition(s, fire, sp).
  out1.Current(s) :- in1.Current(s),
  fail in1.Transition(s, fire, _).
}
```

Semantic Backplane



Summary
Lessons Learned building CPS Tools

- **Understanding the current limits of correct-by-construction design** using model-based verification
  - Significant scalability problems exist even in relatively simple (but real) systems
  - Scalable verification requires strong restrictions on modeling abstractions (e.g. linear hybrid dynamics, order reduction) and has to tolerate low data fidelity
  - The resulting uncertainty is epistemic (systematic, unknown in practice) and cannot be characterized probabilistically

Links

- CPS Virtual Organization: https://cps-vo.org
- AVM Program: http://cps-vo.org/group/avm
- Vehicle Forge: https://vehicleforge.vf.isis.vanderbilt.edu/auth/
- AVM Publications: http://www.isis.vanderbilt.edu/biblio/keyword/183
- AVM Tools: https://vehicleforge.vf.isis.vanderbilt.edu/p/metaresources/home/
- Formula: http://research.microsoft.com/formula