Metamodeling-Enabled Model-Checking for Complex Systems

Prof. Nenad Medvidovic
neno@usc.edu

Dr. George Edwards
gedwards@usc.edu

Center for Systems and Software Engineering
Computer Science Department
University of Southern California

Presentation Outline

1. Project Background and Completed Work
   1. DSLs and MDE
   2. The XTEAM Project
      Metamodeling-Enabled Discrete Event Simulation

2. Proposed Future Work
   1. Planned Research Approach
   2. Potential Benefits

XTEAM Project Overview

- Processes, notations, tools, and designs that enable automated synthesis of end-to-end domain-specific toolsets for architecture modeling, analysis, and code generation

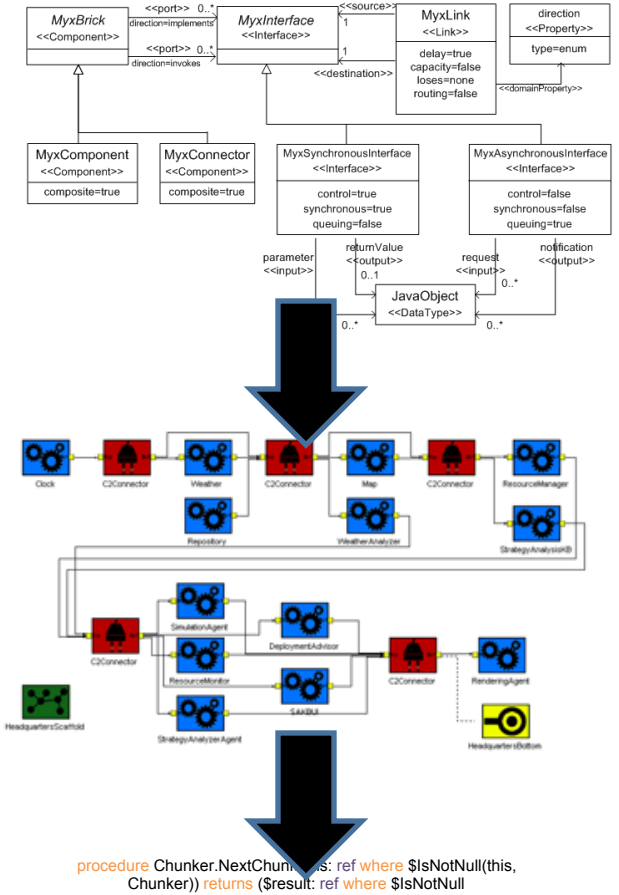
- Example application areas:
  - Embedded and real-time systems
  - Safety- and mission-critical systems
  - Cloud and grid systems

XTEAM Capabilities

- **Metamodeling**
  Domain-specific language definition

- **Architecture modeling**
  System design and requirements definition

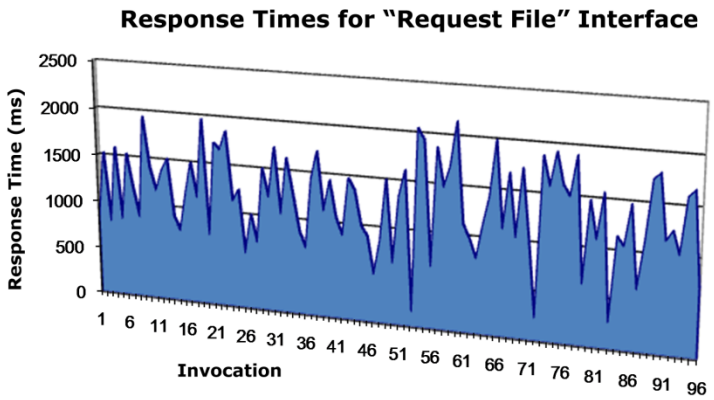
- **System simulation and code generation**
  Design and requirements analysis

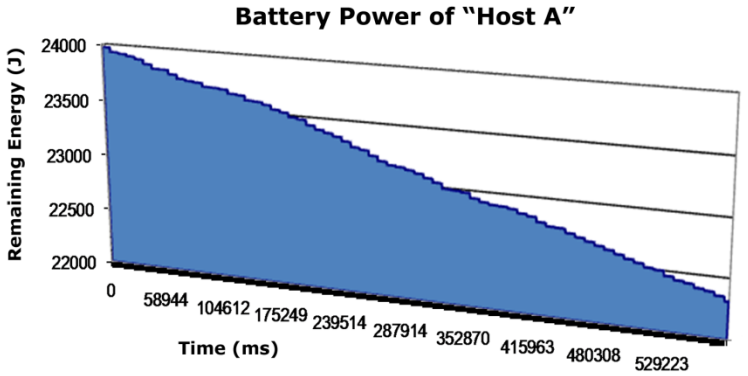


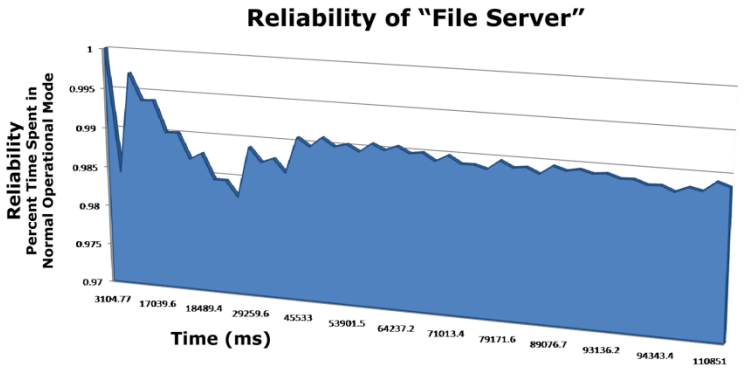

```
procedure Chunker.NextChunk(this: ref where $IsNotNull(this,
        Chunker)) returns ($result: ref where $IsNotNull
        ($result, System.String));
  // in-parameter:  target object
  free requires $Heap[this, $allocated];
  requires ($Heap[this, $ownerFrame] ==
        $PeerGroupPlaceholder || !($Heap[$Heap[this,
        $ownerRef], $inv] <: $Heap[this, $ownerFrame]) ||
        $Heap[$Heap[this, $ownerRef], $localinv] ==
        $BaseClass($Heap[this, $ownerFrame])) && (forall
        $pc: ref :: $pc != null && $Heap[$pc, $allocated] &&
        $Heap[$pc, $ownerRef] == $Heap[this, $ownerRef] &&
        $Heap[$pc, $ownerFrame] == $Heap[this,
        $ownerFrame] ==> $Heap[$pc, $inv] == $typeof($pc)
        && $Heap[$pc, $localinv]
  // out-parameter:  return value
  free ensures $Heap[$result, $allocated];
```

XTEAM Use Cases

- Providing design rationale
- Weighing architectural trade-offs
- Discovering emergent behavior of component assemblies
- Validating component implementations

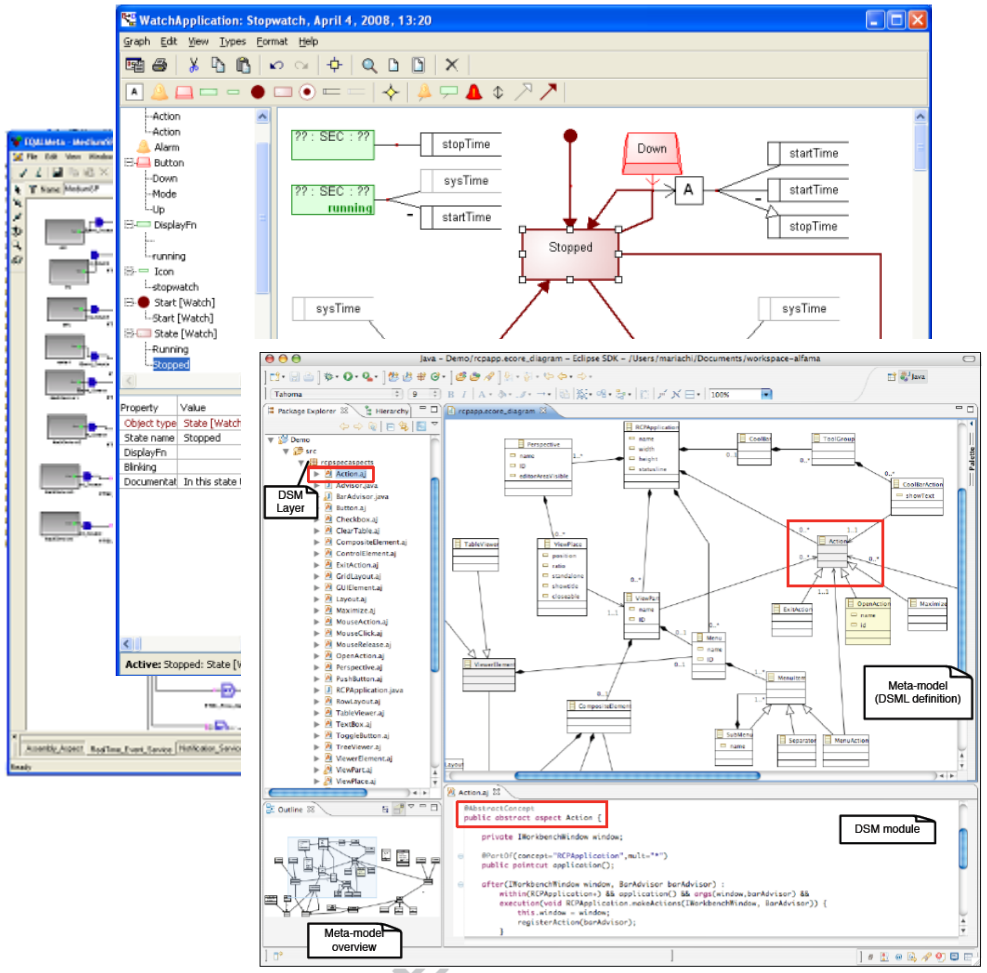






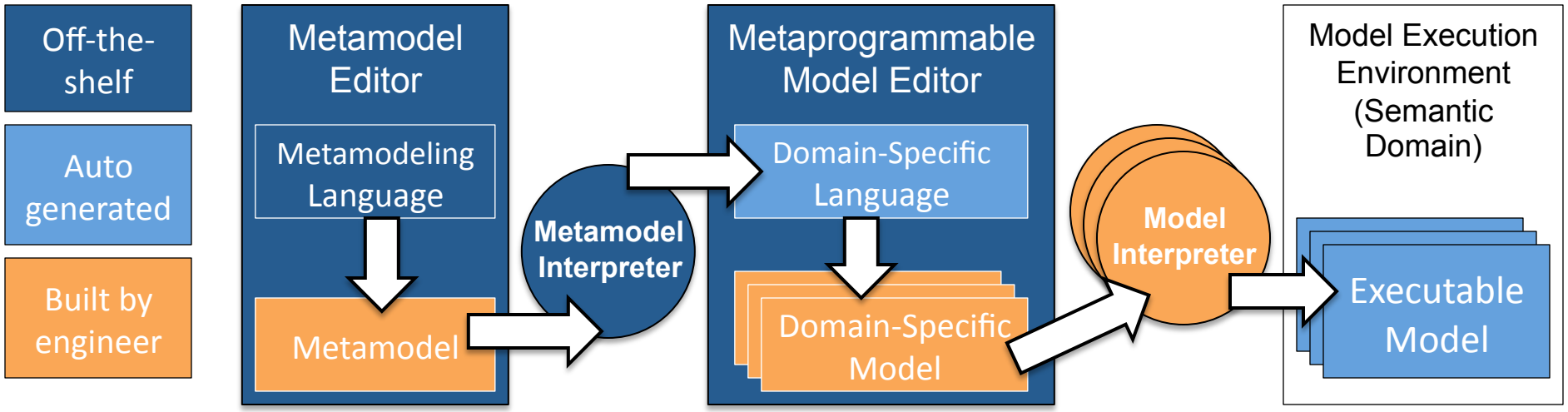
Domain-Specific Languages (DSLs)

- Customized for a particular family of problems (the **domain**)
- Concise and intuitive
  - No missing or extra features
  - Capture patterns
  - Enforce constraints
  - Use native symbols and terms
- Can be modified, evolved, and composed



Model-Driven Engineering (MDE)

- MDE leverages DSLs for architecture modeling
  - **Metamodels** define DSL syntax (types, properties, views, and constraints)
  - **Model interpreters** define DSL semantics (analysis, code generation, etc.)
- MDE platforms provide (some) tool support
  - **Metamodel editor** with built-in **metamodeling language**
  - **Metamodel interpreter** that configures a **metaprogrammable model editor**



Problems with MDE

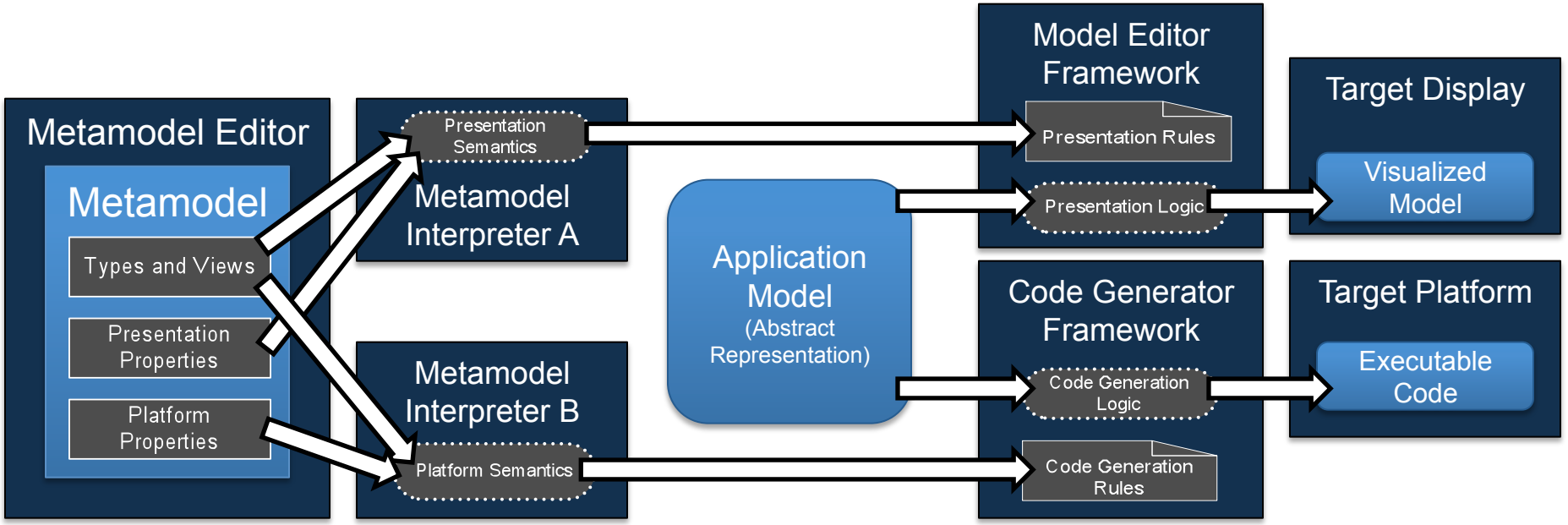
- Building and maintaining code generators for DSLs is *inherently difficult*
  - High design complexity
  - Disproportionate maintenance and evolution costs
  - Hard to verify correctness
  - Redundant development effort
  - Opaque semantics embedded in source code

"Writing translators by hand... in addition to being inefficient, has yet another serious drawback: the semantic mapping between the input and the output is vaguely specified...[Building model interpreters] is the most time consuming and error prone phase of the MIC approach..."

G. Karsai, A. Agrawal, F. Shi, J. Sprinkle. On the Use of Graph Transformation in the Formal Specification of Model Interpreters. Journal of Universal Computer Science, 2003.

XTEAM Solution Approach

Synthesize domain-specific code generators using the same mechanisms that have proven successful for synthesizing domain-specific model editors.

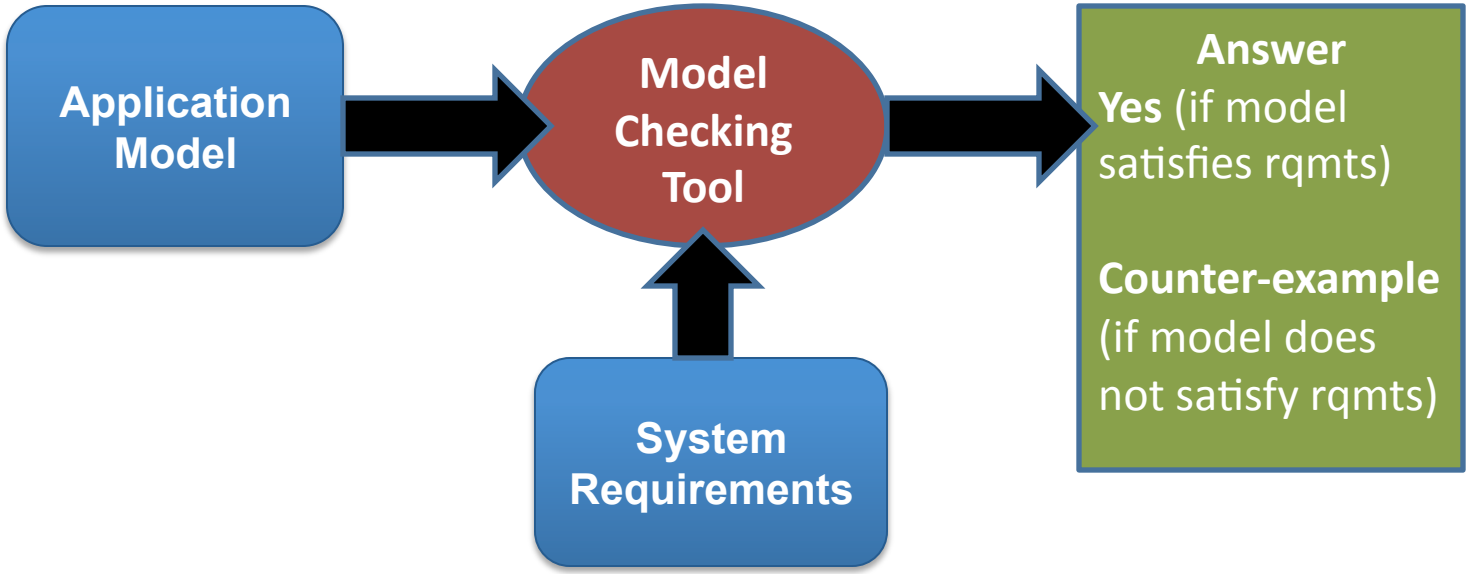


Selected Applications of XTEAM

- MIDAS: wireless sensor network (WSN) applications for building monitoring and control
- Software Mobility Framework: analysis, implementation, deployment, and monitoring framework for mobile robotics systems
- SASSY: automated run-time generation of service-oriented architectures
- RESIST: dynamic reliability estimation and proactive adaptation in situated software systems
- FUSION: self-tuning self-adaptive software systems

Proposed Next Step: Model Checking

- Used to verify requirements and design
  - Safety, security, and other properties
  - Successfully used in real-time and embedded systems

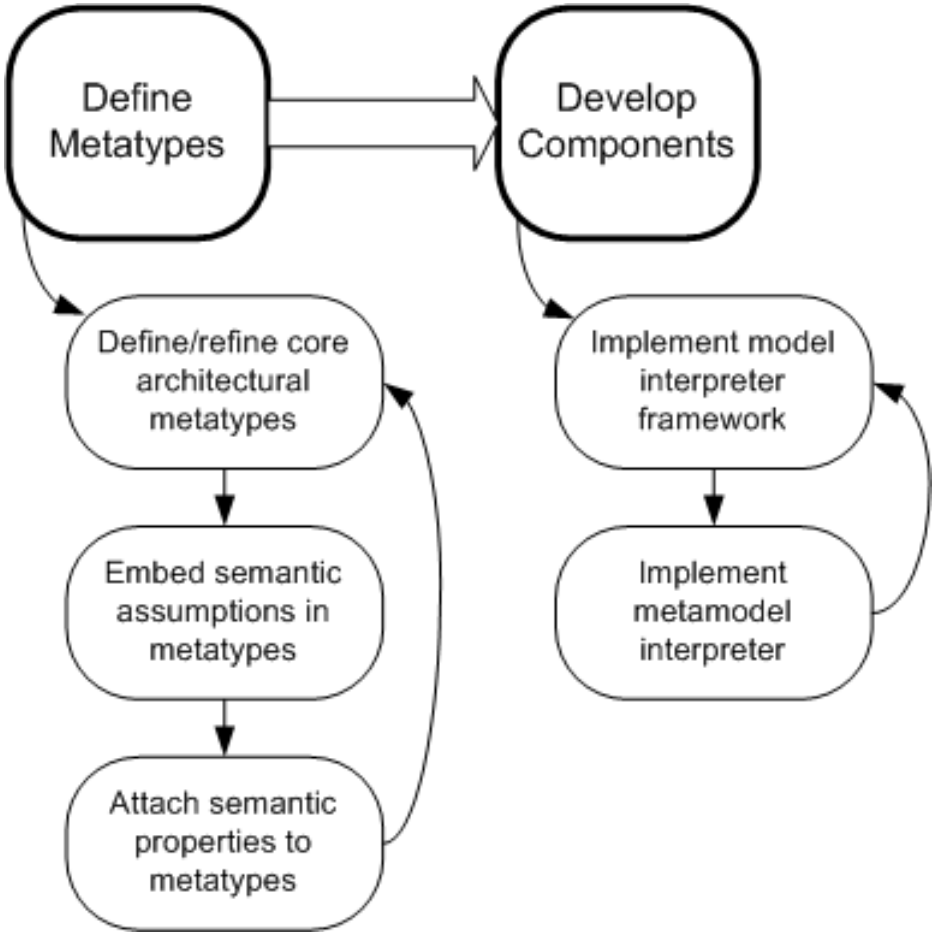


- Well-known tools: SPIN/dSPIN, SMV, Java Pathfinder/ JPF

Existing model checkers do not provide metamodeling

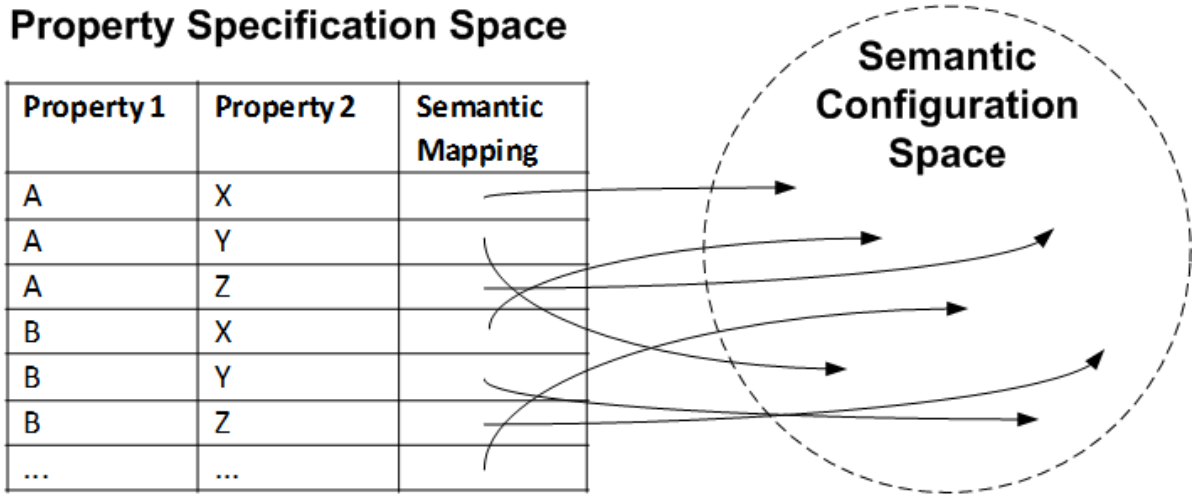
Planned Research Approach

- Refine the XTEAM metamodeling language
  - Include sufficient semantic information for mapping to model checking input
- Implement metaprogrammable model checker
  - Metamodel interpreter
  - Model interpreter framework



Semantic Definitions

- Embedded semantic assumptions
  - Independent of the metamodel
  - Capabilities
    - Behaviors that metatypes exhibit by default
  - Responsibilities
    - Information required to map metatypes to the semantic domain
- Semantic properties
  - Capture semantic variability among metamodels
  - Values map to a semantic configuration



Outcome and Benefits

- End deliverable: A tool chain with model checking AND metamodeling capabilities
  - DSLs: customizable, concise, intuitive modeling
  - Model checking: formal verification of requirements and design
  - No model interpreter development
    - Generally 4 person-months to 24 person-months of effort
  - Predictable cost and timeline for tool development
    - Less risk than a conventional DSL tool chain

Proposed Schedule and Personnel

- Co-PIs:
  - Nenad Medvidovic
  - George Edwards
- Two year project
  - One month per year of support for Medvidovic
  - 50% support for Edwards
  - 50% support for two graduate research assistants

Additional Information

- George Edwards and Nenad Medvidovic, Model Interpreter Frameworks, Technical Report USC-CSSE-2009-514, Center for Software and Systems Engineering, Univ. of Southern California, July 2009.
- George Edwards and Nenad Medvidovic, A Highly Extensible Simulation Framework for Domain-Specific Architectures, Technical Report USC-CSSE-2009-511, Center for Software and Systems Engineering, University of Southern California, May 2009.
- George Edwards and Nenad Medvidovic, A Methodology and Framework for Creating Domain-Specific Development Infrastructures, Proceedings of the 23rd IEEE ACM International Conference on Automated Software Engineering (ASE), September 2008.
- George Edwards, Chiyoung Seo, and Nenad Medvidovic, Model Interpreter Frameworks: A Foundation for the Analysis of Domain-Specific Software Architectures, Journal of Universal Computer Science (JUCS), Special Issue on Software Components, Architectures and Reuse, 2008.
- George Edwards, Chiyoung Seo, and Nenad Medvidovic, Construction of Analytic Frameworks for Component-Based Architectures, Proceedings of the Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), August 2007.
- George Edwards, Sam Malek, and Nenad Medvidovic, Scenario-Driven Dynamic Analysis of Distributed Architectures, Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering (FASE), March 2007.