Domain-Specific Model Analysis and Code-Generation Frameworks

George Edwards
USC Center for Systems and Software Engineering
gedwards@usc.edu

Presentation Outline

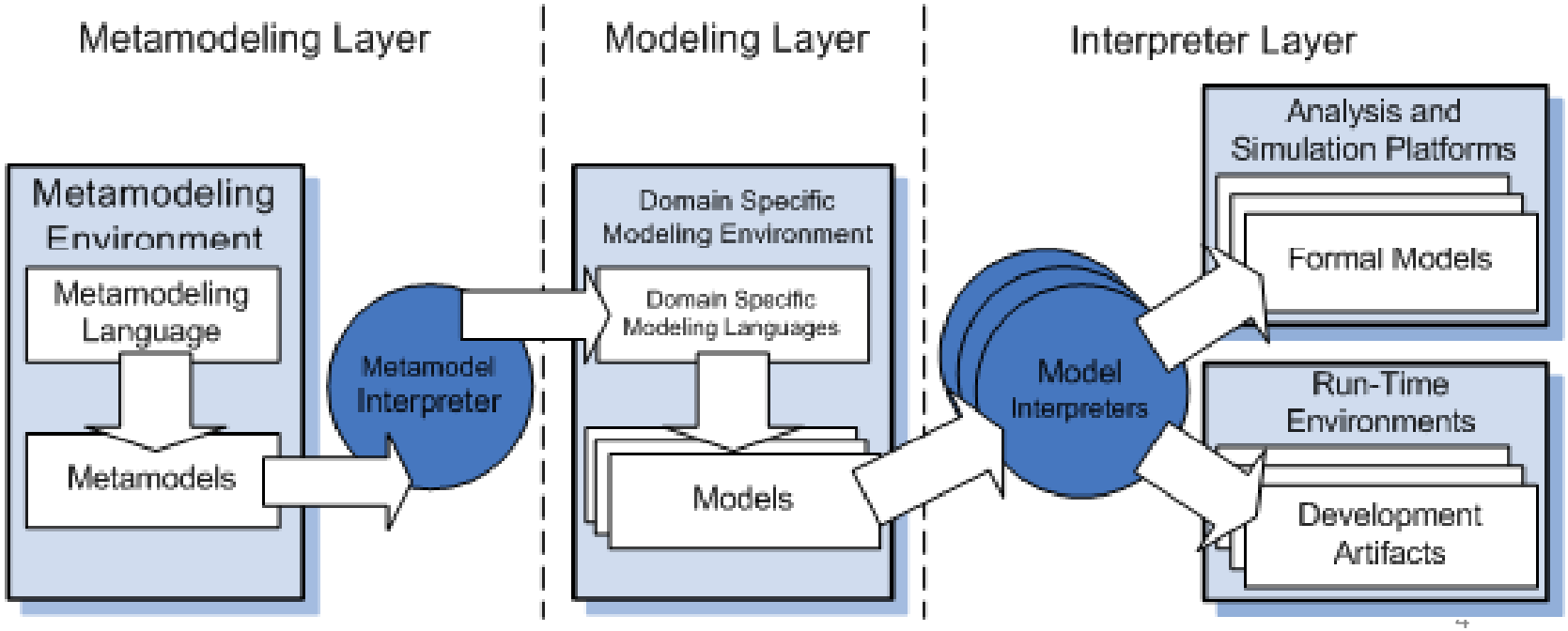
- **Background:** Domain-Specific Languages and Model-Driven Engineering
- **Research Challenge:** Interpretation of Domain-Specific Models
- **Promising Solution:** Model Interpreter Frameworks
- **XTEAM:** Extensible Toolchain for Evaluation of Architectural Models

Why Use DSLs for Ground Systems?

- Heterogeneity of platforms and technologies
  - Examples: multiple types of networks, data models, middleware
  - Result: no “one-size-fits-all” modeling language
- Close and continual interaction with external systems, including electrical and mechanical systems
  - Examples: external science data systems, on-board processing systems
  - Result: models need to capture interfaces to external systems
- Strict operating requirements
  - Examples: real-time data processing, robust fault-tolerance
  - Result: models need to be analyzable, executable

Model-Driven Engineering

- **Model-driven engineering** (MDE) combines domain-specific languages (DSLs) with model interpreters
  - **Metamodels** define elements, relationships, views, and constraints
  - **Model interpreters** leverage domain-specific models for analysis, generation, and transformation



Challenge: Constructing Interpreters

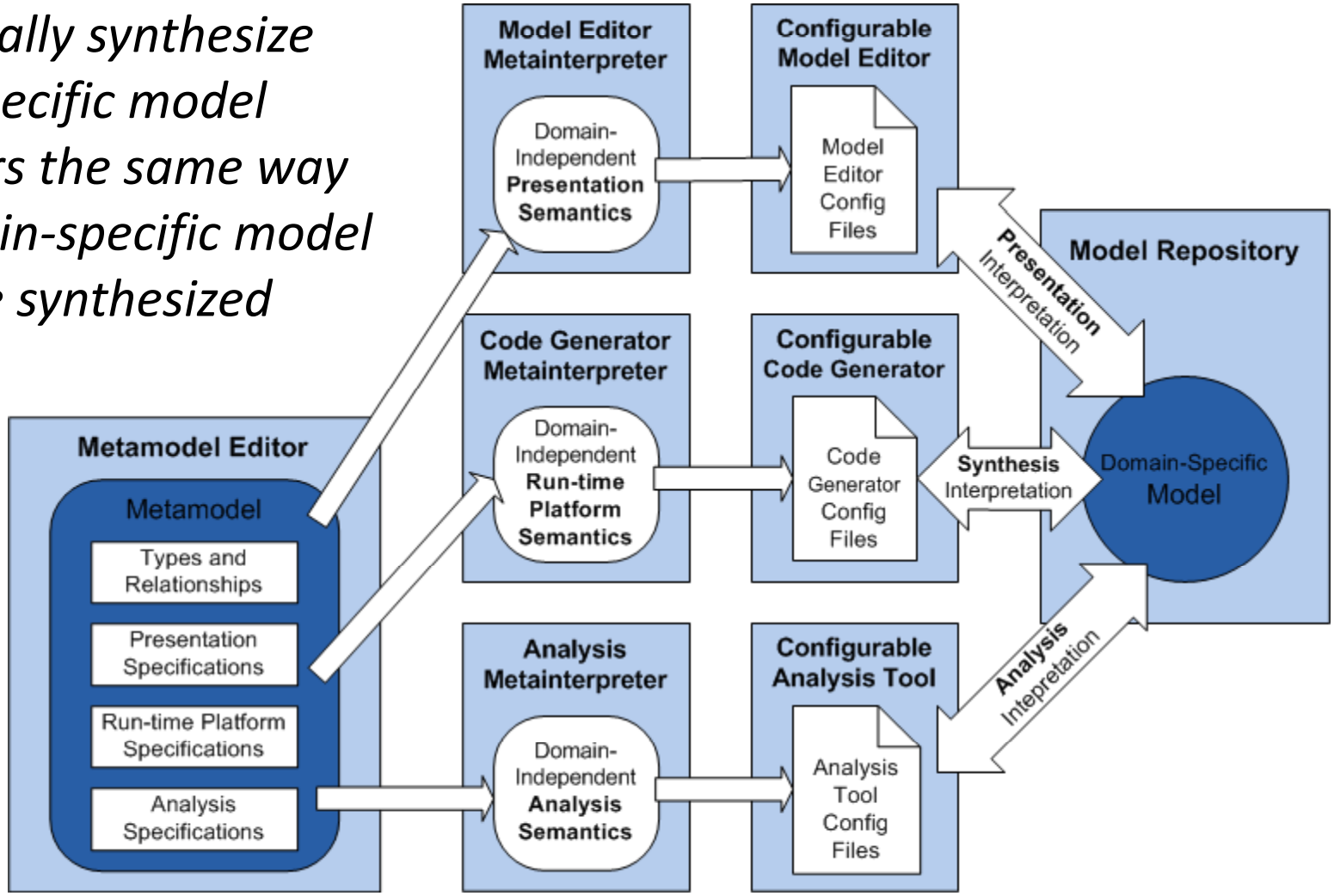
1. Designing, developing, and maintaining DSLs and interpreters is **difficult** and **expensive**
   - A model interpreter must be constructed for each analysis that will be applied to a model
   - Reusing model interpreters for different DSLs is hard
   - Little guidance exists on how to construct DSLs and interpreters
   - The semantics applied to models are opaque (embedded in code)
   - Requires particular types of expertise
   - Common topic of research papers in the modeling community

Model Interpreter Implementation Tasks

1. Find a computational theory that derives the relevant properties
2. Discover the semantic relationships between the constructs present in the architectural models and those present in the analysis models
3. Determine the compatibility between the assumptions and constraints of the architectural models and the analysis models, and resolve conflicts
4. Implement a model interpreter that executes a sequence of operations to transform an architectural model into an analysis model
5. Verify the correctness of the transformation

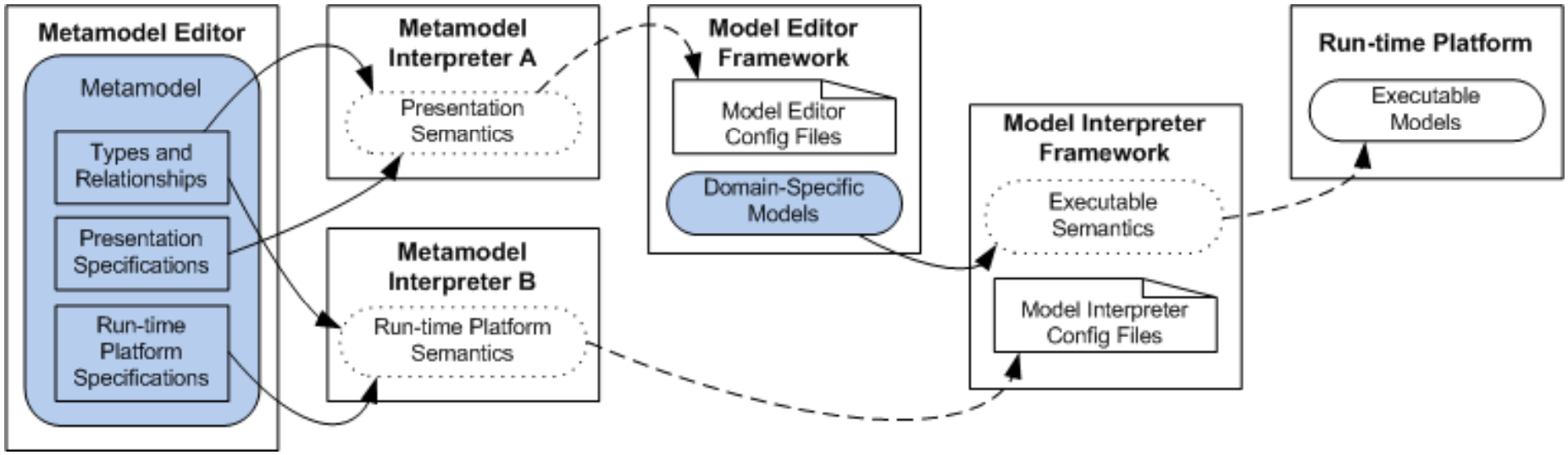
Simplifying Insight

Automatically synthesize domain-specific model interpreters the same way that domain-specific model editors are synthesized



Solution: Model Interpreter Frameworks

- Use a **model interpreter framework** to implement domain-specific analysis
  - Implements a mapping from metamodel types to target platform types
  - Configured via plug-ins generated from a metamodel

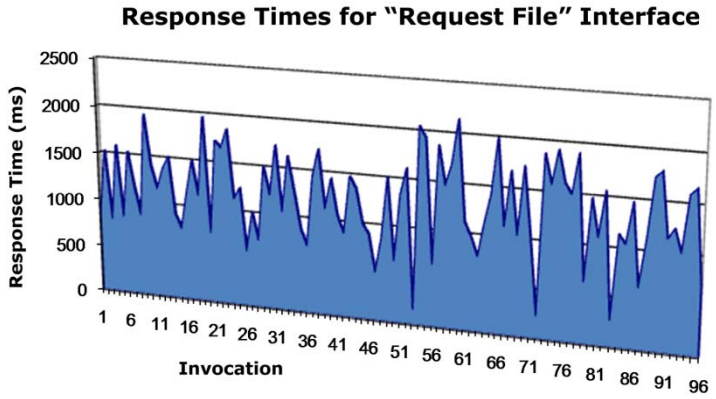


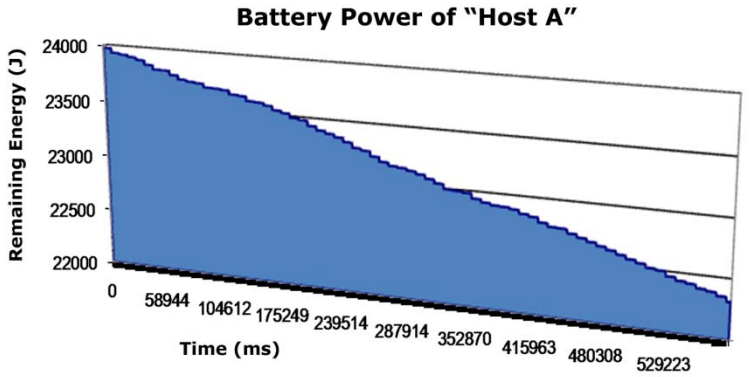
The eXtensible Toolchain for Evaluation of Architectural Models

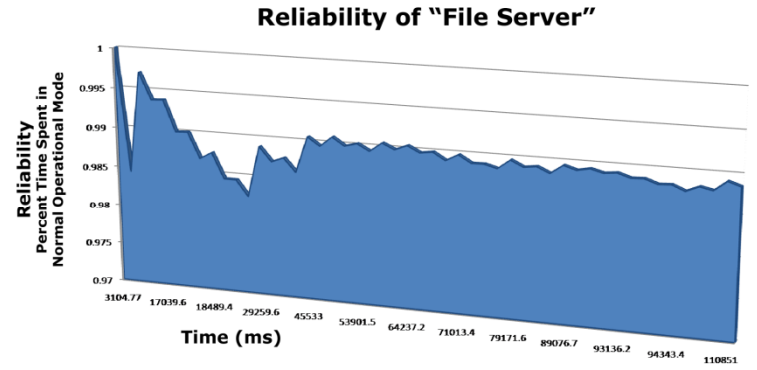
- A modeling environment and accompanying set of model interpreter frameworks for software architectures
- Includes:
  - A specialized metamodeling language
  - A suite of metamodel interpreters and model interpreter frameworks
  - Example extensions targeted towards resource-constrained and mobile computing environments
- Provides the extensibility to easily accommodate both new modeling language features and new architectural analyses

XTEAM Usage

- Providing design rationale
- Weighing architectural trade-offs
- Discovering emergent behavior of component assemblies
- Generating test cases and validating component implementations







Summary

- Building model interpreters to analyze and generate code from domain-specific models is hard

- Our methodology leverages a metamodel and extensible interpreter frameworks to automatically synthesize domain-specific model interpreters

For More Information

Visit the XTEAM website:

http://www-scf.usc.edu/~gedwards/xteam.html

XTEAM Publications:

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.