MODEL-BASED SYSTEMS ENGINEERING DESIGN AND TRADE-OFF ANALYSIS WITH RDF GRAPHS

Nefretiti Nassar and Mark Austin

Institute of Systems Research, University of Maryland, College Park, MD 20742.

CSER 2013 Presentation, Atlanta, GA, March 21, 2013
INTRODUCTION

Project Motivation

- Complexity of engineering systems is on the rise.
- Strategic approaches to design will employ semantic descriptions of application domains and use ontologies and rule-based reasoning to enable validation of requirements, automated synthesis of potentially good design solutions, and communication among multiple disciplines.

Tenet of our Work

- Semantic Web concepts and technologies can provide assistance in the model-based system engineering and design of modern-day systems. But how?
MOTIVATING DESIGN PROBLEM

Prototype implementation: Satisfaction of requirements & components selection for a Home Theater Design Problem

Outcomes:
1st Search procedure will find combinations of components that satisfy requirements
2nd Design requirements stated in such a way that no feasible designs exist
OUTLINE

Questions

• What is the Semantic Web?
• What technologies are provided by the Semantic Web?
• Which technologies in the Semantic Web will be useful for design?
• Can Semantic Web Technologies be used to create a chain of transformations for the synthesis of design alternatives?
• What parts of the design process can be handled by Semantic Web?
• What parts of the design process cannot be handled by Semantic Web?
• Can the limitations of Semantic Web be overcome through the use of Java/Python software?
• To what extent is it possible to simplify the design process?
WHAT IS THE SEMANTIC WEB?

Goals of the Semantic Web

• Facilitate communication of knowledge
• Automated discovery of new knowledge

How can Semantic Web help design?

• Validation of requirements
• Automated synthesis of design solutions
• Formal design representations
WHAT IS THE SEMANTIC WEB?

Resource Description Framework

• Graph-based data model for describing relationships between objects and class in simple, but general, way.

RDF Triples and Graphs
**Simplified Design with RDF and Python**

**Using Semantic Web ...**

- Basic Reasoning Capability
  - OWL
    - RDF
      - Jena
        - Protege Ontology Editor
  - Reasoning with Ontologies and Rules
    - OWL
      - Ontology
      - Knowledge base
    - SWRL
      - Rules
        - Jess
          - Jess Rule Engine
            - Inferred knowledge base
    - Save
      - OWL format

**No Ontologies ...**

- RDF
  - Python
    - Reasoning

- Straightforward and uncomplicated
- Smaller graph size
- Practical design solutions can be obtained
DESIGN METHODOLOGY

Synthesis of design solutions from RDF graph representations of requirements and design components

Explore: RDF graphs for representation of requirements and design component properties. Python for implementation and sequencing logical reasoning and inference mechanisms.
CASE STUDY: HOME THEATER DESIGN PROBLEM

- **Requirements**
  - I need a home theatre system.
  - The total cost must be less than US $2,100

- **System Design / Architecture**
  - Design Space
    - **TV** | **Amplifier** | **Speaker**
    - 3 | 3 | 3
    - Potential System-Level Designs: 27

- **Interface requirements**
  - Cost of Speakers + Cost of Amplifier + Cost of TV < US $2,100
RDF GRAPH MODELS

Modeling Requirements & Design Components as RDF Graphs

RDF Triple:
- Subject = Requirement
- Predicate = Property
- Object = Value
SYNTHESIS OF FEASIBLE SYSTEM CONFIGURATIONS

System Architecture Rules

Component Compatibility Rules

Potential System-Level Designs: 27  18
QUANTITATIVE EVALUATION OF DESIGN REQUIREMENTS

--- Level 1 Requirements

--- Level 2 Requirements

--- Level 3 Requirements

Flat screen TV

Amplifier Speakers

Component Specification involves identifying key inequalities constraints
Satisfaction of equality occurs through satisfaction of inclusive and dependency requirements
SYNTHESIS OF SYSTEM-LEVEL DESIGN ALTERNATIVES

Feasible System Configurations → Requirement Verification

Design Space
System-Level Designs: 27 18 9
TRACKING RDF GRAPH SIZE

**Requirement & Component Graph**
- **Edges**: 130
- **Vertices**: 42, 47, 52, 57, 62
- **Component Compatibility Rule**: 148
- **System Architecture Rule**: 133

**Merged Graph**
- **Edges**: 278
- **Vertices**: 98.5, 99, 99.5, 100, 100.5
- **Level 1 Requirement Rules**: 321
- **Level 2 Requirement Rules**: 319, 310
- **Level 3 Requirement Rules**: 308

Legend:
- Requirement Graph
- Component Graph
- Merged Graph

Diagram showing the tracking of RDF graph size with graphs indicating requirement, component, and merged graphs. The graphs display the number of edges and vertices for each graph type, with specific values highlighted for rule applications.
Python: Systematic comparison of Feasible System Designs wrt cost, performance, and reliability

Trade-off Analysis: Maximize Reliability & Minimize Cost

Cost ($)

Reliability (%)
CONCLUSIONS AND FUTURE WORK

Benefits and Limitations

• Satisfy requirements and acquire good design solutions in a straightforward and uncomplicated manner.
• RDF graph representations provide desirable balance of expressiveness and flexibility.
• Not scalable: BUT during the early stages of development, design solutions for component selection are usually based upon smaller numbers of requirements and component options.

So what about Jena, OWL and SWRL?

• RDF graphs are smaller – a lot smaller -- than OWL counterparts
• RDF graph storage can be simple – Strings. This works well with Python.
• Jena and OWL can represent and reason with physical quantities.
Questions?