Inconsistency Management in Model-Based Systems Engineering

Sebastian J. I. Herzig
Ahsan Qamar
Christiaan J. J. Paredis
Georgia Institute of Technology
Model-Based Systems Engineering Center

GLOBAL PRODUCT DATA INTEROPERABILITY SUMMIT 2014
Overlapping Concerns

Global Product Data Interoperability Summit | 2014

- Software Engineering
- System Architecture
- Thermal Analysis
- Project Management
- Manufacturing System Design
- Product Design

• Controller Design
• Hydraulics
• ...

Terms:
- cost
- profit
- mass
- reliability
- security
- maneuverability
- policy compliance
- demand
- lifespan
- geometry
- standards
- comfort
- production rate
- robustness
- detectability
- security
- reliability
- cost
- standards
- maneuverability
- policy compliance
- demand
- lifespan
- geometry
- comfort
- production rate
- robustness
- detectability
Situation

Global Product Data Interoperability Summit | 2014

Major processes supported

Source: GPDIS 2013
Used with permission from Rainer Romatka (Boeing, Seattle)
Typical Scenario

Global Product Data Interoperability Summit | 2014

Set of workstations inconsistent with those defined in Teamcenter or SysML?

Heuristics followed (e.g., DFMA)?

Predicted assembly time inconsistent with desired (or required) value?

Company-wide naming conventions for part numbers followed?

Sequence of assembly steps the same as in Teamcenter and/or SysML?
Another Scenario: CM in Design & Development

- Change in requirements or design: how do changes propagate? Which models are inconsistent as a result?

- Example: changing *full holes* to *piloted holes* in a pylon assembly
  - Affects manufacturing process, optimal shop floor layout for workstations
  - Affects ergonomics models
  - Affects cost models
  - ...

Adapted from a scenario provided by Michael Christian (Boeing, St. Louis)
Relationships Between Models

Software Engineering

System Architecture

Thermal Analysis

Project Management

Manufacturing System Design

Product Design

• Controller Design
• Hydraulics
• …
Relationships Between Models

Global Product Data Interoperability Summit | 2014

How can information from multiple sources and in different formats be handled automatically?

How can conflicts, i.e. *inconsistencies*, be detected and resolved?

How can macro-level questions spanning multiple models be answered?
Overview

• Context & Motivation
• How do we Think About Inconsistency Management?
• Technology Demonstrator
• Current & Future Work
• Conclusions
What is an Inconsistency?

An inconsistency is a contradiction

- There exists no possible world that can represent all of the given information
- Mathematically speaking, it is possible to derive a statement $S$ and its negation $\neg S$

Some more examples:

"The aircraft has 3 landing gears and the aircraft has 5 landing gears" (logical contradiction)

"On earth, the velocity of a free-falling object follows a sinusoidal curve" (inconsistency w.r.t. observations and/or accepted theories/laws of nature)

"The probability of the engine failing is 0.1%, and I think it will not fail with a probability of 98%" (mathematical inconsistency)

"I prefer apples over oranges, oranges over bananas, and bananas over apples" (intransitive preferences)
A Basic Process for Inconsistency Management

Retrieve Current State of Models

Check for Inconsistencies

Inconsistency identified?  

yes  

no  

Resolve Inconsistencies

Inconsistency Mgmt. Knowledge

Inconsistency detection knowledge

Inconsistency resolution knowledge

Knowledge
Managing Inconsistencies – A Simple Example...

Evolve: add property with name “Diameter”

Check: the property's name is “Diameter” (M) ⇔ The property's name must be “diameter” (P)

⇒ But, on a symbolic level, “diameter” and “Diameter” are distinctly different
⇒ Hence, a contradiction exists - i.e., inconsistency can be derived fairly easily

Resolve: resolving is trivial in this case: change “Diameter” to “diameter” ...

Company policy: All named attributes / properties must have camelCase names
Example 2: Managing Inconsistencies Across Models

- Here, obvious overlap exists
  - A property / attribute is semantically equivalent ➔ a relation across models
  - Inconsistent if constraints on property are not compatible
  - Semantically equivalent properties with incompatible constraints are inconsistent ➔ pattern can identify such inconsistencies

F86 Sabre CAD Data – Credit to ASDL, Georgia Tech
Example 2: Managing Inconsistencies Across Models

- Here, obvious overlap exists
  - A property / attribute is semantically equivalent \(\rightarrow\) a relation across models
  - Inconsistent if constraints on property are not compatible
  - Semantically equivalent properties with incompatible constraints \(\rightarrow\) a pattern that identifies such inconsistencies

1) How and where to capture relations across models?
2) How can patterns going across models be defined (and queried)?
   \(\Rightarrow\) Need a common representational formalism

F86 Sabre CAD Data – Credit to ASDL, Georgia Tech
Graphs as a Common Representational Formalism

Physical Architecture

Mfg. Simulation Model

CAD Geometry Model

Implicit (inferable)

Pattern

Variable Bindings

Cross-Model Relations

Transformation
**Hypothesis:** any inconsistency that is *derivable* from the information and knowledge represented by the graph can also be identified by querying an appropriate pattern

- Use graph transformation rules for inference
- Query graph patterns to identify inconsistencies
Identifying Inconsistencies by Querying Patterns

Some observations:

- Terminology used in different modeling languages can be very different: e.g., attribute vs. value property
  - Rules / patterns need to refer to these
  - Requires rule variants for same type of inconsistency: a nightmare to maintain!

- Also: someone needs to define relations across models (can we define rules for this purpose? → more later)
Pattern Vocabulary - Varying Semantic Precision

- **Base Vocabulary**
  - Domain Voc. #2
  - Domain Voc. #3
  - Domain Voc. #n
  - Domain Voc. #m

- **Language Vocabulary**
  - Language Voc. #1
  - Language Voc. #2
  - Language Voc. #3
  - Language Voc. #4

- **Semantic Precision**
  - e.g., Rule 1
  - e.g., Rule 2
  - e.g., Requirements
  - e.g., DOORS
  - e.g., SysML

- **Inference Paths**
  - ↔
Pattern Vocabulary - Varying Semantic Precision

- Concepts (Base): Object, property, relation, type, constraint, value, ...
- Individuals (Base): Name, equivalentTo, contains, ...

Language Voc. #2

Concepts (SysML): ValueProperty, PartProperty, Block, Requirement, ...

Language Voc. #1

Domain Voc. #1

e.g., Rule 1

Domain Voc. #3

e.g., Rule 2

e.g., Requirements

e.g., SysML

Language Voc. #3

Language Voc. #4

e.g., DOORS

Domain Voc. #3

Domain Voc. #n

Domain Voc. #m

Semantic Precision

Inference Paths
Inconsistency Management

1. Retrieve Current State of Models
2. Check for Inconsistencies
   - Inconsistency identified?
     - yes
       - Resolve Inconsistencies
     - no
       - Inconsistency identified?
3. Inconsistency Mgmt. Knowledge
   - Inconsistency detection knowledge
   - Inconsistency resolution knowledge

- Common representational formalism
- Manageable, unifying node & edge label vocabularies
How Do We Resolve These Inconsistencies?

- Three different representations of the value for (semantically) the same property
- Can we resolve all of these inconsistencies automatically?
**Which Model Should We Trust?**

The SysML model, of course!

The information in the PLM / PDM system, of course!

\[ D = 0.9 \]

\[ \text{hubDiameter} = 0.85 \]
Resolving Inconsistencies

Global Product Data Interoperability Summit | 2014

- Three different representations of the value for (semantically) the same property
- Can we resolve all of these inconsistencies automatically??
Resolving Inconsistencies

Inconsistency 2

Great, just negated our previous fix… why did this happen?

- Often more than one way to fix a particular (type of) inconsistency
- Multiple related inconsistencies
- Need some strategy to find optimal sequence of resolution actions:
  e.g., here, source of authority

Oooops....

We’re good here now
Resolving Inconsistencies: a Decision-Making Problem

Given:
- Exhaustive list of possible fixes to particular kinds of inconsistencies
- List of inconsistencies identified by a corresponding pattern
- We can use this information to generate a list of alternative sequences of fix operations
  - Some may lead to a better outcome than others → Analyze
  - Which one of these is optimal depends on preferences and beliefs… → Evaluate & select most preferable

Alternatives → Outcomes → Selection Criterion: E[u] → Most Preferred Alternative

Figure Adapted from G. Hazelrigg
Summary: Makeup of an Inconsistency Rule

Identification Pattern & Context

Resolution Alternatives

1. Tolerate / Do Nothing

2

3

Applies: always (invariant)
Owner: sherzig
Created: 2014-08-18

...
Overview

Global Product Data Interoperability Summit | 2014

- Context & Motivation
- How do we Think About Inconsistency Management?
- Technology Demonstrator
- Current & Future Work
- Conclusions
Technology Demonstrator: ConSystent

• Use of W3C semantic web standards (RDF, HTTP, REST)
• Utilizes OSLC recs. and Framework

• Adapters are specific to tools / models and can:
  • Access model data (r/w)
  • Generate RDF data
  • Run as servlets

Inconsistency Rules
Model Cache
Adapters
Intra-/Internet

ConSystent

• Adapters are specific to tools / models and can:
  • Access model data (r/w)
  • Generate RDF data
  • Run as servlets

Use of W3C semantic web standards (RDF, HTTP, REST)
Utilizes OSLC recs. and Framework
• Statements about resources in the form of subject-predicate-object expressions (triples) ➔ put together, these form a graph

Subject = URI

http://.../blocks/WheelHub

Predicate = URI

http://.../property

Object = URI or literal

http://.../WheelHub/properties/diameter

• World Wide Web Consortium (W3C) standard for data interchange
• Used in Semantic Web applications ➔ stronger semantics for websites, enabling next generation search engines
• Variety of syntax notations and data serialization formats (e.g., XML)

Adapted and Used with Permission from Axel Reichwein, Koneksys LLC
### Open Services for Lifecycle Collaboration (OSLC)

**Domain / Language Vocabularies**

<table>
<thead>
<tr>
<th>Domain / Language Vocabularies</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALM/PLM Interoperability</td>
<td>Draft</td>
</tr>
<tr>
<td>Architecture Management</td>
<td>2.0</td>
</tr>
<tr>
<td>Asset Management</td>
<td>2.0</td>
</tr>
<tr>
<td>Automation</td>
<td>2.0</td>
</tr>
<tr>
<td>Change Management</td>
<td>2.0</td>
</tr>
<tr>
<td>Estimation and Measurement</td>
<td>Draft</td>
</tr>
<tr>
<td>Performance Monitoring</td>
<td>2.0</td>
</tr>
<tr>
<td>Quality Management</td>
<td>2.0</td>
</tr>
<tr>
<td>Reconciliation</td>
<td>2.0</td>
</tr>
<tr>
<td>Requirements Management</td>
<td>2.0</td>
</tr>
<tr>
<td>MBSE / SysML</td>
<td>Draft</td>
</tr>
</tbody>
</table>

**OSLC Core Vocabulary**

- OSLC adds additional semantics for expressing structural constraints on RDF data (e.g., cardinality)
- Domain vocabularies introduce standard terminology across domains

**OSLC resource types for various domains and languages**

**OSLC Core Vocabulary**

- OSLC adds additional semantics for expressing structural constraints on RDF data (e.g., cardinality)
- Domain vocabularies introduce standard terminology across domains
Example - Inconsistency Detection

硫酸：

<sysml:valueProperty rdf:about="http://.../sparLength">
  <sysml:defaultValue ...>2.3</sysml:defaultValue>
  ...
  <sem:equivalentTo rdf:resource="http://.../length"/>
</sysml:valueProperty>
Inconsistency patterns (and, generally, inconsistency rules) are represented in a language such as SPARQL and executed using a corresponding engine.
Status of OSLC Adapters Developed @ MBSEC

**Open Source**
- NoMagic MagicDraw SysML (now / soon part of Eclipse Lyo)
- Mathworks Simulink (now / soon part of Eclipse Lyo)

**Have also developed adapters for:**
- PTC Integrity (Requirements Management)
- LMS Imagine Lab AMESim (Multi-Domain Systems Simulation)

**Examples of adapters developed outside of MBSEC:**
- Siemens Teamcenter (General Motors)
- Bugzilla (part of Eclipse Lyo)
- NinaCRM
- ...
Overview

• Context & Motivation
• How do we Think About Inconsistency Management?
• Technology Demonstrator
• Current & Future Work
• Conclusions
Current Work – Probabilistic Inconsistency Reasoning

Context may provide additional evidence

Prior Belief + Identified Features (“Evidence”) = Updated Belief

... and in the same spirit: P(Inconsistent) = ?
Future Work – Visualization

• Visual analytics tools to generate interactive visualizations using data from various system modeling sources
• Explore global patterns in data
• Seamless transition between models from various domains

Future work with Dr. Rahul Basole (CS), Dr. Leon McGinnis (ISyE)
Future Work – Visualization

- Visual analytics tools to generate interactive visualizations using data from various system modeling sources
- Explore global patterns in data
- Seamless transition between models from various domains

Future work with Dr. Rahul Basole (CS), Dr. Leon McGinnis (ISyE)
Conclusions & Key Takeaways

- Inconsistency management = continuous V&V

- Directed, attributed, typed multi-graphs can be used to represent a wide variety of engineering models

- Graph patterns can be used to model types of inconsistencies and query for these

- Resolving inconsistencies is a decision-making problem, where the most preferred alternative sequence of "fix" operations must be determined

- Semantic web technologies are a powerful and practical basis for a scalable model integration framework
Some Open Questions…

- **Access Control**
  - Who can access which (select parts of) models, and with which rights?
  - How do we deal with restricted access to some data (e.g., ITAR)?

- **Rule & Inter-Model Relations Ownership**
  - Who manages the rules for which portion of the system?
  - Who should manage relations between models?

- **Automated Inconsistency Resolution**
  - Is it ever a good idea to autonomously resolve an inconsistency?

Inaccessible information (due to, e.g., proprietary nature of data or tool limitations)
Assume the following axiom holds in our world (verbally):

- “Any Aircraft has exactly one kind of Turbojet Engine”
• An inconsistency exists if we can deduce that (in our world) there exists an Aircraft that has two (different) types of TurbojetEngines

• Facts explicitly and implicitly encoded in graph:
  • “F-86 type Aircraft”
  • “GE_J47 type TurbojetEngine”
  • “F-86E type TurbojetEngine”
  • “GE_J47 differentFrom F-86E”
  • “F-86 has GE_J47”
  • “F-86 has F-86E”
  • “F-86 has two TurbojetEngines”

Inconsistent, because:
(F-86 has two TurbojetEngines) ∧ (F-86 has one Turbojet Engine) → ⊥
• Inconsistency (as pattern / negative graph constraint):

“An inconsistency is present if there is an element $x$ of type Aircraft that is associated through the relation ‘has’ to at least two elements ‘$y$’ of type TurbojetEngine”
ConSystent – Overview of Architecture

Inconsistency Management Infrastructure

Inconsistency Detector (Jena)
- Inference Engine
- Inference Rules
- Pattern Matching
- Identification Patterns

RDF Database (Apache Fuseki)
- RDF Crawler
- GUI

Inconsistency Resolver
- Decision Analyzer
- Resolution Alternatives

OSLC Adapter 1
- Web Server 1
- RDF over HTTP

OSLC Adapter n
- Web Server n
- Tool n
OSLC Tool Adapters in Java in a Nutshell

OSLC Adapter for Tool A

Tool API / Custom Parser

Resource Shape(s)
(Java Object)

Serialization / Deserialization
(JAX-RS, Lyo)

Web / App. Server
(Tomcat, Jetty, …)

RDF over HTTP

Resource Shape Def.
(Java Class with OSLC-Specific Annotations)

instance of

Work with Axel Reichwein, Koneksys LLC
OSLC Tool Adapters in Java in a Nutshell

Automatic generation of RS descriptions from Ecore model

Work with Axel Reichwein, Koneksys LLC
OSLC Tool Adapters in Java in a Nutshell

Serialization of resource shape in various formats (e.g., RDF/XML, TTL, HTML)

Work with Axel Reichwein, Koneksys LLC