Reasoning on Engineering Knowledge

Applications and Desired Features
Agenda

Application:
- Introduction & Motivation
- Mapping Engineering Data
- Reasoning on Engineering Data

Desired Features:
- Desired Feature 1
- Desired Feature 2
- Desired Feature 3

The presentations’ mission:
Creating awareness in Semantic Web community for further potential of Semantic Web technologies in Engineering of automated production systems
Running Example - Product

- O-Ring
- Spring
- Cylinder
- Piston
- Lid
Running Example – Production System

Logistic module

Assembly

Warehouse

Drilling station
**Introduction & Motivation**

<table>
<thead>
<tr>
<th>Market demand</th>
<th>Resulting issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ Increasing number of product variants</td>
<td>■ Fact: Production systems usually have a longer life-cycle than products have</td>
</tr>
<tr>
<td>■ Decreasing product life-cycles</td>
<td>■ Resulting issue: Total product variance over a production systems’ life-cycle is unknown at design time</td>
</tr>
<tr>
<td>■ Decreasing lot sizes (-&gt; 1pc.)</td>
<td>■ Fact: Adaption to product changes requires to check system functionality against product requirements</td>
</tr>
<tr>
<td>■ Less re-occuring orders</td>
<td>■ Resulting issue: Since lot size is decreasing and orders are less re-occuring, costly manual adaption is not profitably anymore</td>
</tr>
<tr>
<td>■ Low cost and high quality</td>
<td></td>
</tr>
</tbody>
</table>
# Introduction & Motivation

## Approach

- Support production system adaption by providing a skill description
- Skill description represents HW&SW information that eases the check against product requirements
- Skill description contains information on system structure and behavior so that identifying components that need change is eased
- Use of Semantic Web technologies for modeling and reasoning on engineering data
- First step: only focus on kinematic skills

## Requirements

- Production systems are often engineered customer specific:
  - Existing engineering data has to be used (3D-CAD, electrical CAD, PLC code)
  - No manual creation of models
- A production system usually consists of modules of different suppliers (e.g. FESTO, Siemens, Bosch, ...): Same structure and behavior has to result in the same skill
- Approach has to cope with partially incomplete information:
  - Kinematics might not be modeled
  - PLC code might be incomplete
  - ...

Mapping Engineering Data I

- 1472835-001_Layout.iam
- is_CADPartOf(?x, ?y) → consistsOf(?y, ?x)

**Use Case Diagram**

- **Engineer**
  - design()
  - create_file()

- **3D-CAD Tool**
  - import_file()
  - read_Mapping_rules()

- **CAD data file**

- **Mapping Component**
  - map_data()

- **Ontology**

- **Reasoning Engine**
  - reasoning()
3D CAD example

Maped individuals

- RCS_Part1
- RCS_Part2
- Assembly1
- RCS_Assembly1
- MovementRest_Trans1
- Vector_TCP_P2
- Vector_Trans_StartPoint1
- Vector_Trans_Orientation2
- hasReferenceCoordinateSystem
- hasMovementRestriction
- hasToolCenterPoint
- hasRestrictionDirection
Reasoning is necessary, since needed information is only implicitly in engineering data
- Only kinematic of single assemblies
- No combined kinematics

SWRL has been chosen to model knowledge on combination of kinematics

Reasoning Target: Inference of the overall kinematic, based on the combination of „primitive“ kinematics
example on rule combination

<table>
<thead>
<tr>
<th>Rule type</th>
<th>Kinematic</th>
<th>Assemblies</th>
</tr>
</thead>
<tbody>
<tr>
<td>SK2</td>
<td>abcde</td>
<td>Top-Level</td>
</tr>
<tr>
<td>SK1</td>
<td>abc  abcde</td>
<td></td>
</tr>
<tr>
<td>SB1/SB2</td>
<td>a b c d e f</td>
<td>a-f</td>
</tr>
</tbody>
</table>

Red arrows indicate `isComposedOfSkill` relationships.

Blue arrows indicate `isAssembledOn` relationships.
Desired Feature 1

Creating Individuals

- The first desired feature seems easy: creating individuals with SWRL

- Creating individuals becomes necessary at various points, since knowledge is not only connected in another way:
  - A kinematic is not existing in engineering data -> create individual
  - The kinematics’ vectors do not exist in engineering data -> create individual
  - ...

- There already exists a syntax through the SWRL Extensions built-in library

- Unfortunately this extension is not supported by any “of-the-shelf” reasoner

Rule example

Assembly(?a),
hasReferenceCoordinateSystem(?a,?RCSa),
consistsOf(?a,?c),
Component(?c),
hasDegreeOfFreedomTranslational(?c,?DoFT),
hasDegreeOfFreedomRotational(?c,?DoFR),
swrlb:add(?DoFSum,?DoFT,?DoFR),
swrlb:greaterThan(?DoFSum,0),
swrlx:makeOWLThing(?Kin,1) ->

movesRelativeTo(?c,?RCSa),
movementRestrictionDefinesMovementDescr   iption(?c,?Kin),
Kinematic(?Kin)
Knowledge in the manufacturing domain is often based on mathematical descriptions like vectors.

In cases where this knowledge is processed or evaluated in a new context, mathematical operations need to be performed.

Example:
- Every assembly provides a kinematic that has to be calculated
- When combining these kinematics, mathematical operations are necessary in order to identify their type of kinematic (translation, rotation, planes, ...)
- Since every DataProperty of every vector has to be mentioned in a rule, these rules become very complex

An easier way to handle mathematical operations within SWRL rules is needed, that is supported by an “off-the-shelf” reasoner.

Equation example

Extracting a translational kinematic out of CAD data:
- SV = start vector
- O = orientation vector
- RS = restriction start vector
- T = Tool Center Point vector
- EV = end vector
- S = vector pointing on start level
- Alpha = scalar

\[
\overrightarrow{sv} = \overrightarrow{T} + \left( \overrightarrow{O} \cdot \frac{(\overrightarrow{RS} - \overrightarrow{T})}{\overrightarrow{O} \cdot \overrightarrow{O}} \right) \overrightarrow{O}; \ \overrightarrow{ev} = \overrightarrow{S} + \alpha \overrightarrow{O}
\]

Required basic math operations: 16
Defeasible reasoning

- Engineering methods are often based on an iterative procedure, since the overall task needs to be split in smaller tasks.

- Handling iterative procedures in SWRL requires defeasible reasoning, example:
  - Inferring a top level kinematic within one rule results in a too complex rule -> inference of single kinematics and combination of them.
  - This requires defeasible reasoning, since only latest inferred kinematics should be considered for further inference.

- Unfortunately, defeasible reasoning is not supported by any “off-the-shelf” reasoner.

- The need for defeasible reasoning becomes even more apparent, when looking at what has to be done next.
Defeasible reasoning

- After having inferred the kinematic, other data formats have to be mapped in OWL and their restrictions on the skill have to be inferred:
  - E-CAD e.g.: Properties of drives and sensors
  - PLC e.g.: Programmed behavior

- Other data formats than 3D-CAD might restrict the inferred knowledge, example:
  - PLC: Kinematic may show greater movement areal than sensors and drives do allow based on PLC program
  - E-CAD: Chosen drives and sensors restrict possible endpoints due to resulting forces or their structure
Conclusion

- Semantic Web technologies provide a great potential in handling upcoming changes in future manufacturing
- Manufacturing skill descriptions can be extracted from engineering data
- Providing a few features by an “off-the-shelf” reasoner would even enhance the potential

Desired Features

- Creating individuals within SWRL
- Support of complex math operations in SWRL
- Defeasible reasoning

Thank you for your attention