Quality Criteria for ISO15926-8 Compliant Installation Descriptions

Martin Giese

8 September 2011

Department of Informatics

University of Oslo

Outline

- IOHN
- RDF Basics
- ISO15926-8
- Why do we need quality criteria?
- Sample Criteria

Integrated Operations in the High North

- Design, implement, and test a Digital Platform for 2nd generation “Integrated Operations”

Integrated Operations in the High North

- Design, implement, and test a Digital Platform for 2nd generation “Integrated Operations”
  - 1st gen: integrate onshore-offshore

Integrated Operations in the High North

- Design, implement, and test a Digital Platform for 2nd generation “Integrated Operations”
  - 1st gen: integrate onshore-offshore
  - 2nd gen: integrate oil company, suppliers, service companies,. . .

Integrated Operations in the High North

- Design, implement, and test a Digital Platform for 2nd generation “Integrated Operations”
  - 1st gen: integrate onshore-offshore
  - 2nd gen: integrate oil company, suppliers, service companies,. . .
- 2008–2011

Integrated Operations in the High North

- Design, implement, and test a Digital Platform for 2nd generation “Integrated Operations”
  - 1st gen: integrate onshore-offshore
  - 2nd gen: integrate oil company, suppliers, service companies,. . .
- 2008–2011
- Lead by Det Norske Veritas (DNV)

Integrated Operations in the High North

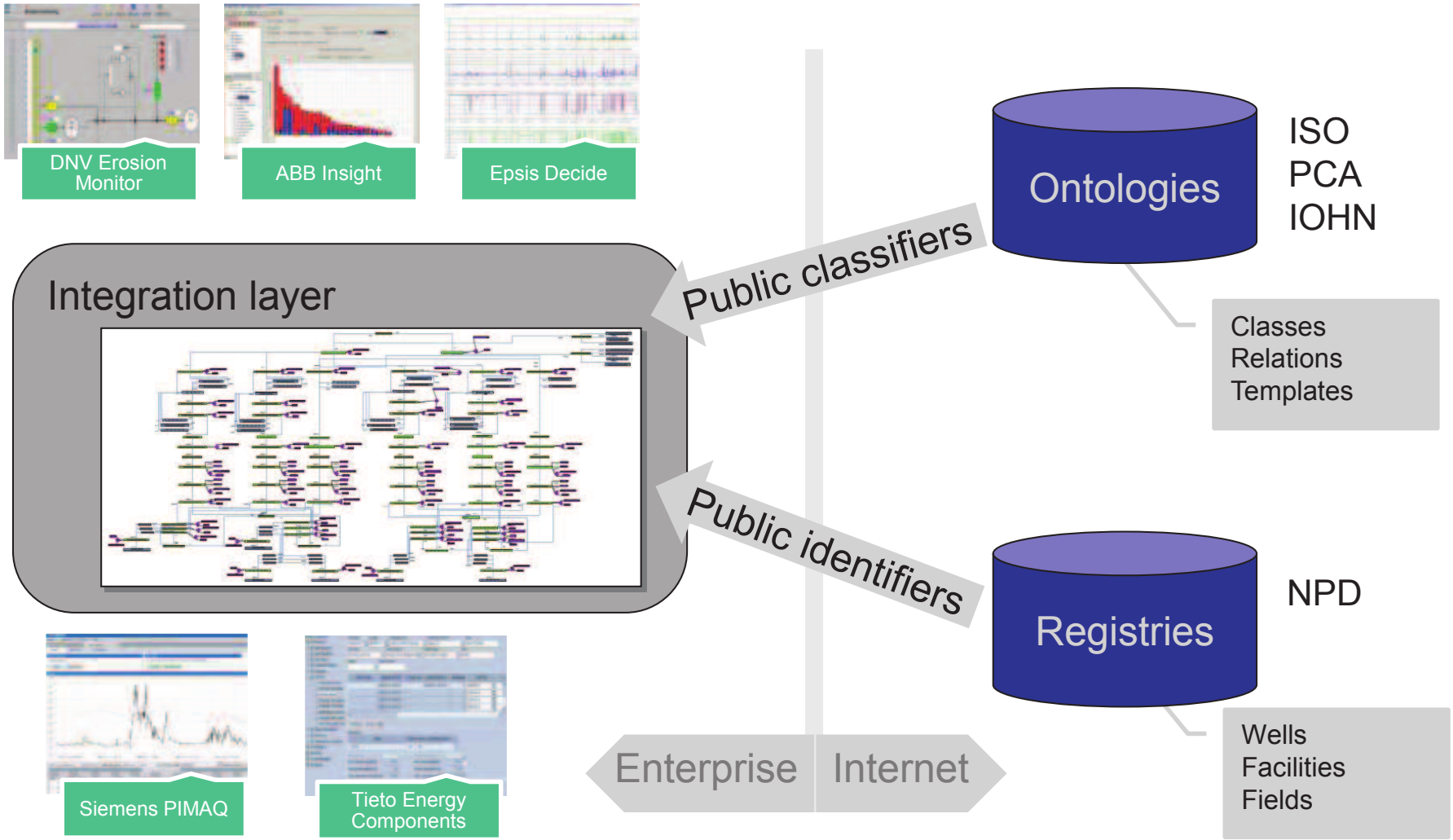
- Design, implement, and test a Digital Platform for 2nd generation “Integrated Operations”
  - 1st gen: integrate onshore-offshore
  - 2nd gen: integrate oil company, suppliers, service companies,. . .
- 2008–2011
- Lead by Det Norske Veritas (DNV)
- Partially financed by the Research Council of Norway

Integrated Operations in the High North

- Design, implement, and test a Digital Platform for 2nd generation “Integrated Operations”
  - 1st gen: integrate onshore-offshore
  - 2nd gen: integrate oil company, suppliers, service companies,. . .

- 2008–2011
- Lead by Det Norske Veritas (DNV)
- Partially financed by the Research Council of Norway
- Participants: ABB, Abelia, Baker Hughes, Cisco, Computas, Det Norske Veritas, ENI, Epsis, FMC Technologies, FSI, IBM, IO Center, IRIS, National Oilwell Varco, NTNU, OLF, POSC Caesar Association, Petroleum Safety Authority Norway, Siemens, Statoil, Norwegian Defence, University of Oslo, University of Stavanger

IOHN prototype



©Johan W. Klüwer, DNV

RDF

- Resource Description Framework

RDF

- Resource Description Framework
- a data model for knowledge representation

RDF

- Resource Description Framework
- a data model for knowledge representation
- standardised by the World Wide Web Consortium (W3C)

RDF

- Resource Description Framework
- a data model for knowledge representation
- standardised by the World Wide Web Consortium (W3C)
- all information represented as “statements” consisting of

RDF

- Resource Description Framework
- a data model for knowledge representation
- standardised by the World Wide Web Consortium (W3C)
- all information represented as “statements” consisting of
  - Subject

RDF

- Resource Description Framework
- a data model for knowledge representation
- standardised by the World Wide Web Consortium (W3C)
- all information represented as “statements” consisting of
  - Subject
  - Predicate

RDF

- Resource Description Framework
- a data model for knowledge representation
- standardised by the World Wide Web Consortium (W3C)
- all information represented as “statements” consisting of
  - Subject
  - Predicate
  - Object

RDF

- Resource Description Framework
- a data model for knowledge representation
- standardised by the World Wide Web Consortium (W3C)
- all information represented as “statements” consisting of
  - Subject
  - Predicate
  - Object
- S,P,O are “resources” identified by URIs

RDF

- Resource Description Framework
- a data model for knowledge representation
- standardised by the World Wide Web Consortium (W3C)
- all information represented as “statements” consisting of
  - Subject
  - Predicate
  - Object
- S,P,O are “resources” identified by URIs
- objects can also be “literals”

RDF

- Resource Description Framework
- a data model for knowledge representation
- standardised by the World Wide Web Consortium (W3C)
- all information represented as “statements” consisting of
  - Subject
  - Predicate
  - Object
- S,P,O are “resources” identified by URIs
- objects can also be “literals”
  - Like strings, but can carry indication of language or data type.

Types

- RDF has a standardised predicate `rdf:type` to assign a type to a resource

Types

- RDF has a standardised predicate `rdf:type` to assign a type to a resource
- `:s4711 rdf:type :PressureSensor`

Types

- RDF has a standardised predicate `rdf:type` to assign a type to a resource
- `:s4711 rdf:type :PressureSensor`
- Possible to declare type hierarchies:

```
:PressureSensor rdfs:subClassOf :Sensor
:TemperatureSensor rdfs:subClassOf :Sensor
:Sensor rdfs:subClassOf :Instrument
```

Types

- RDF has a standardised predicate `rdf:type` to assign a type to a resource
- `:s4711 rdf:type :PressureSensor`
- Possible to declare type hierarchies:

```
:PressureSensor rdfs:subClassOf :Sensor
:TemperatureSensor rdfs:subClassOf :Sensor
:Sensor rdfs:subClassOf :Instrument
```

- using OWL, more complex relations between predicates and classes can be expressed (Modelling)

Types

- RDF has a standardised predicate `rdf:type` to assign a type to a resource
- `:s4711 rdf:type :PressureSensor`
- Possible to declare type hierarchies:

```
:PressureSensor rdfs:subClassOf :Sensor
:TemperatureSensor rdfs:subClassOf :Sensor
:Sensor rdfs:subClassOf :Instrument
```

- using OWL, more complex relations between predicates and classes can be expressed (Modelling)
- Models can allow to compute new types (inferred types)

```
:s4711 rdf:type :Sensor
:s4711 rdf:type :Instrument
```

Types

- RDF has a standardised predicate `rdf:type` to assign a type to a resource
- `:s4711 rdf:type :PressureSensor`
- Possible to declare type hierarchies:

```
:PressureSensor rdfs:subClassOf :Sensor
:TemperatureSensor rdfs:subClassOf :Sensor
:Sensor rdfs:subClassOf :Instrument
```

- using OWL, more complex relations between predicates and classes can be expressed (Modelling)
- Models can allow to compute new types (inferred types)

```
:s4711 rdf:type :Sensor
:s4711 rdf:type :Instrument
```

- Rich models can derive more unexpected inferred types

ISO15926 part 7

- Only binary relations in RDF. Often need for n-ary relations.

ISO15926 part 7

- Only binary relations in RDF. Often need for n-ary relations.
- Use of RDF encoding for ISO15926 vocabulary complicated.

ISO15926 part 7

- Only binary relations in RDF. Often need for n-ary relations.
- Use of RDF encoding for ISO15926 vocabulary complicated.
- ISO15926 part 7 introduces *templates*

ISO15926 part 7

- Only binary relations in RDF. Often need for n-ary relations.
- Use of RDF encoding for ISO15926 vocabulary complicated.
- ISO15926 part 7 introduces *templates*
- Template *instance*:

$$T(a_1, a_2, \ldots a_n)$$

ISO15926 part 7

- Only binary relations in RDF. Often need for n-ary relations.
- Use of RDF encoding for ISO15926 vocabulary complicated.
- ISO15926 part 7 introduces *templates*
- Template *instance*:

$$T(a_1, a_2, \ldots a_n)$$

- Template *definitions* $\approx$ macro expansion

$$T(x, y, z) \leftrightarrow C(x) \wedge R(x, y) \wedge S(y, z) \wedge D(z)$$

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

- Representation:

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

- Representation:

$$\begin{aligned} &\texttt{\_:x rdf:type } T\texttt{;} \\ &\qquad R_{T,1}\ a_1\texttt{;} \\ &\texttt{...} \\ &\qquad R_{T,n}\ a_n\texttt{.} \end{aligned}$$

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

- Representation:

$$\begin{aligned} &\texttt{\_:x rdf:type } T\texttt{;} \\ &\qquad R_{T,1}\ a_1\texttt{;} \\ &\texttt{...} \\ &\qquad R_{T,n}\ a_n\texttt{.} \end{aligned}$$

- RDF-predicates $R_{T,i}$ for each T also described in RDF

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

- Representation:

$$\begin{aligned} &\texttt{\_:x rdf:type } T\texttt{;} \\ &\qquad R_{T,1}\ a_1\texttt{;} \\ &\texttt{...} \\ &\qquad R_{T,n}\ a_n\texttt{.} \end{aligned}$$

- RDF-predicates $R_{T,i}$ for each T also described in RDF
- Final RDF model for part 8 representation contains:

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

- Representation:

$$\begin{aligned} &\texttt{\_:x rdf:type } T\texttt{;} \\ &\qquad R_{T,1}\ a_1\texttt{;} \\ &\texttt{...} \\ &\qquad R_{T,n}\ a_n\texttt{.} \end{aligned}$$

- RDF-predicates $R_{T,i}$ for each T also described in RDF
- Final RDF model for part 8 representation contains:
  - ISO15926 part 2 reference model

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

- Representation:

$$\begin{aligned} &\texttt{\_:x rdf:type } T\texttt{;} \\ &\qquad R_{T,1}\ a_1\texttt{;} \\ &\texttt{...} \\ &\qquad R_{T,n}\ a_n\texttt{.} \end{aligned}$$

- RDF-predicates $R_{T,i}$ for each T also described in RDF
- Final RDF model for part 8 representation contains:
  - ISO15926 part 2 reference model
  - Declaration of vocabulary for template declarations

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

- Representation:

$$\begin{aligned} &\texttt{\_:x rdf:type } T\texttt{;} \\ &\qquad R_{T,1}\ a_1\texttt{;} \\ &\texttt{...} \\ &\qquad R_{T,n}\ a_n\texttt{.} \end{aligned}$$

- RDF-predicates $R_{T,i}$ for each T also described in RDF
- Final RDF model for part 8 representation contains:
  - ISO15926 part 2 reference model
  - Declaration of vocabulary for template declarations
  - Declaration of concrete templates used

ISO15926 part 8

- Part 8: Representation of *template instances* as RDF.
- Template instance

$$T(a_1, a_2, \ldots a_n)$$

- Representation:

$$\begin{aligned} &\texttt{\_:x rdf:type } T\texttt{;} \\ &\qquad R_{T,1}\ a_1\texttt{;} \\ &\texttt{...} \\ &\qquad R_{T,n}\ a_n\texttt{.} \end{aligned}$$

- RDF-predicates $R_{T,i}$ for each T also described in RDF
- Final RDF model for part 8 representation contains:
  - ISO15926 part 2 reference model
  - Declaration of vocabulary for template declarations
  - Declaration of concrete templates used
  - Representation of template instances

Why Quality Criteria?

- RDF is a very generic data model

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration
  - In particular for a “semantic web” scenario!

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration
  - In particular for a “semantic web” scenario!
- In a description of a concrete installation:

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration
  - In particular for a “semantic web” scenario!
- In a description of a concrete installation:
  - not needed or useful

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration
  - In particular for a “semantic web” scenario!
- In a description of a concrete installation:
  - not needed or useful
  - make further processing more difficult than necessary

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration
  - In particular for a “semantic web” scenario!
- In a description of a concrete installation:
  - not needed or useful
  - make further processing more difficult than necessary
- Design quality criteria to ensure that representation is:

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration
  - In particular for a “semantic web” scenario!
- In a description of a concrete installation:
  - not needed or useful
  - make further processing more difficult than necessary
- Design quality criteria to ensure that representation is:
  - precise

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration
  - In particular for a “semantic web” scenario!
- In a description of a concrete installation:
  - not needed or useful
  - make further processing more difficult than necessary
- Design quality criteria to ensure that representation is:
  - precise
  - homogeneous

Why Quality Criteria?

- RDF is a very generic data model
- Could say within one document:
  - Parts A and B are connected by some “Connector”
  - The type “Connector” has been defined by John Smith in 2009.
  - “Screw” and “Nail” are subclasses of “Connector”
- Vagueness and inhomogeneity useful e.g. for data integration
  - In particular for a “semantic web” scenario!
- In a description of a concrete installation:
  - not needed or useful
  - make further processing more difficult than necessary
- Design quality criteria to ensure that representation is:
  - precise
  - homogeneous
  - . . . keeping the *required* generality

Models and Instance Data

- Information in RDF can be divided in:

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)
  - instance data

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)
  - instance data
- Nice to be able to mix this, e.g. for semantic web applications

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)
  - instance data
- Nice to be able to mix this, e.g. for semantic web applications
  - Inhomogeneity!

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)
  - instance data
- Nice to be able to mix this, e.g. for semantic web applications
  - Inhomogeneity!
- Different representation norms

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)
  - instance data
- Nice to be able to mix this, e.g. for semantic web applications
  - Inhomogeneity!
- Different representation norms
  - ISO15926-8 applies to instance data, not to vocabulary description

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)
  - instance data
- Nice to be able to mix this, e.g. for semantic web applications
  - Inhomogeneity!
- Different representation norms
  - ISO15926-8 applies to instance data, not to vocabulary description
- Different quality requirements

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)
  - instance data
- Nice to be able to mix this, e.g. for semantic web applications
  - Inhomogeneity!
- Different representation norms
  - ISO15926-8 applies to instance data, not to vocabulary description
- Different quality requirements
  - One thing to talk about a particular screw, another to talk about all “Connectors”.

Models and Instance Data

- Information in RDF can be divided in:
  - modelling information, talking about types and relations (vocabulary description)
  - instance data
- Nice to be able to mix this, e.g. for semantic web applications
  - Inhomogeneity!
- Different representation norms
  - ISO15926-8 applies to instance data, not to vocabulary description
- Different quality requirements
  - One thing to talk about a particular screw, another to talk about all “Connectors”.
- Best to separate them for installation descriptions.

A Sample Requirement

A Sample Requirement

Requirement 2.1.1

There is a clear separation of the set of RDF statements into

A Sample Requirement

Requirement 2.1.1

There is a clear separation of the set of RDF statements into

- a vocabulary description, and

A Sample Requirement

Requirement 2.1.1

There is a clear separation of the set of RDF statements into

- a vocabulary description, and
- instance data.

A Sample Requirement

Requirement 2.1.1

There is a clear separation of the set of RDF statements into

- a vocabulary description, and
- instance data.

A Sample Requirement

Requirement 2.1.1

There is a clear separation of the set of RDF statements into

- a vocabulary description, and
- instance data.

This separation might be effected through the storage in different files, in different graphs underlying a SPARQL endpoint, etc. In any case, it must be possible to say which part a given triple belongs to.

- Rationale

A Sample Requirement

Requirement 2.1.1

There is a clear separation of the set of RDF statements into

- a vocabulary description, and
- instance data.

This separation might be effected through the storage in different files, in different graphs underlying a SPARQL endpoint, etc. In any case, it must be possible to say which part a given triple belongs to.

- Rationale
  - different rules apply to vocab. description and instance data

A Sample Requirement

Requirement 2.1.1

There is a clear separation of the set of RDF statements into

- a vocabulary description, and
- instance data.

This separation might be effected through the storage in different files, in different graphs underlying a SPARQL endpoint, etc. In any case, it must be possible to say which part a given triple belongs to.

- Rationale
  - different rules apply to vocab. description and instance data
- Implementation

A Sample Requirement

Requirement 2.1.1

There is a clear separation of the set of RDF statements into

- a vocabulary description, and
- instance data.

This separation might be effected through the storage in different files, in different graphs underlying a SPARQL endpoint, etc. In any case, it must be possible to say which part a given triple belongs to.

- Rationale
  - different rules apply to vocab. description and instance data
- Implementation
  - RDF data will have to be presented in separate parts to tools testing other criteria

Generic vs. Specific Types

- RDF is a general knowledge representation mechanism

Generic vs. Specific Types

- RDF is a general knowledge representation mechanism
- Useful to be able to say that `:s4711` is a `PhysicalObject` when nothing more is known.

Generic vs. Specific Types

- RDF is a general knowledge representation mechanism
- Useful to be able to say that `:s4711` is a `PhysicalObject` when nothing more is known.
  - Vagueness!

Generic vs. Specific Types

- RDF is a general knowledge representation mechanism
- Useful to be able to say that `:s4711` is a `PhysicalObject` when nothing more is known.
  - Vagueness!
- When describing a concrete installation, more *is* known and should be represented.

Generic vs. Specific Types

- RDF is a general knowledge representation mechanism
- Useful to be able to say that `:s4711` is a `PhysicalObject` when nothing more is known.
  - Vagueness!
- When describing a concrete installation, more *is* known and should be represented.
- Tools extracting information about e.g. all pressure sensors in the description need to rely on specific type information

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept
- The divide between the generic and the specific is dependent on the context of the description.

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept
- The divide between the generic and the specific is dependent on the context of the description.

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept
- The divide between the generic and the specific is dependent on the context of the description.

Requirement 2.1.2

There is a clear separation of the vocabulary definition into

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept
- The divide between the generic and the specific is dependent on the context of the description.

Requirement 2.1.2

There is a clear separation of the vocabulary definition into

- an application specific vocabulary

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept
- The divide between the generic and the specific is dependent on the context of the description.

Requirement 2.1.2

There is a clear separation of the vocabulary definition into

- an application specific vocabulary
- a generic vocabulary

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept
- The divide between the generic and the specific is dependent on the context of the description.

Requirement 2.1.2

There is a clear separation of the vocabulary definition into

- an application specific vocabulary
- a generic vocabulary

- E.g.:

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept
- The divide between the generic and the specific is dependent on the context of the description.

Requirement 2.1.2

There is a clear separation of the vocabulary definition into

- an application specific vocabulary
- a generic vocabulary

- E.g.:
  - Generic vocabulary: “Physical Thing”, “Connector”

What is Specific?

- A type `PressureSensor` might be concrete enough in one situation, while in another, one needs to have e.g. `AirPressureSensor`.
- This dichotomy can be observed for almost any concept
- The divide between the generic and the specific is dependent on the context of the description.

Requirement 2.1.2

There is a clear separation of the vocabulary definition into

- an application specific vocabulary
- a generic vocabulary

- E.g.:
  - Generic vocabulary: “Physical Thing”, “Connector”
  - Specific vocabulary: “air pressure sensor”, “M8 wood screw”

Requiring Specific Types

Requiring Specific Types

Requirement 2.3.1

Any resource mentioned in the instance data should have a type, either explicitly given with `rdf:type`, or inferable, that is declared in the specific vocabulary.

Requiring Specific Types

Requirement 2.3.1

Any resource mentioned in the instance data should have a type, either explicitly given with `rdf:type`, or inferable, that is declared in the specific vocabulary.

- May require inference to find a specific type.

Requiring Specific Types

Requirement 2.3.1

Any resource mentioned in the instance data should have a type, either explicitly given with `rdf:type`, or inferable, that is declared in the specific vocabulary.

- May require inference to find a specific type.
- Given: `:s4711 rdf:type :Sensor`
  Given: `:s4711 rdf:type :PneumaticInstrument`
  Infer: `:s4711 rdf:type :AirPressureSensor`

Requiring Specific Types

Requirement 2.3.1

Any resource mentioned in the instance data should have a type, either explicitly given with `rdf:type`, or inferable, that is declared in the specific vocabulary.

- May require inference to find a specific type.
- Given: `:s4711 rdf:type :Sensor`
  Given: `:s4711 rdf:type :PneumaticInstrument`
  Infer: `:s4711 rdf:type :AirPressureSensor`
- Reasoning can be expensive

Requiring Specific Types

Requirement 2.3.1

Any resource mentioned in the instance data should have a type, either explicitly given with `rdf:type`, or inferable, that is declared in the specific vocabulary.

- May require inference to find a specific type.
- Given: `:s4711 rdf:type :Sensor`
  Given: `:s4711 rdf:type :PneumaticInstrument`
  Infer: `:s4711 rdf:type :AirPressureSensor`
- Reasoning can be expensive
- Required when

Requiring Specific Types

Requirement 2.3.1

Any resource mentioned in the instance data should have a type, either explicitly given with `rdf:type`, or inferable, that is declared in the specific vocabulary.

- May require inference to find a specific type.
- Given: `:s4711 rdf:type :Sensor`
  Given: `:s4711 rdf:type :PneumaticInstrument`
  Infer: `:s4711 rdf:type :AirPressureSensor`
- Reasoning can be expensive
- Required when
  - checking the requirement

Requiring Specific Types

Requirement 2.3.1

Any resource mentioned in the instance data should have a type, either explicitly given with `rdf:type`, or inferable, that is declared in the specific vocabulary.

- May require inference to find a specific type.
- Given: `:s4711 rdf:type :Sensor`
  Given: `:s4711 rdf:type :PneumaticInstrument`
  Infer: `:s4711 rdf:type :AirPressureSensor`
- Reasoning can be expensive
- Required when
  - checking the requirement
  - processing the data

Types and Reasoning

- Vary amount of inferred triples in representation...

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types
  - large model

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types
  - large model
  - in general expensive to check requirements

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types
  - large model
  - in general expensive to check requirements
  - convenient for further processing

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types
  - large model
  - in general expensive to check requirements
  - convenient for further processing
- Can restrict to simpler, less expensive reasoning.

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types
  - large model
  - in general expensive to check requirements
  - convenient for further processing
- Can restrict to simpler, less expensive reasoning.
  - E.g. restrict expressivity of models.

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types
  - large model
  - in general expensive to check requirements
  - convenient for further processing
- Can restrict to simpler, less expensive reasoning.
  - E.g. restrict expressivity of models.
  - smaller model

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types
  - large model
  - in general expensive to check requirements
  - convenient for further processing
- Can restrict to simpler, less expensive reasoning.
  - E.g. restrict expressivity of models.
  - smaller model
  - checking requirements affordable

Types and Reasoning

- Vary amount of inferred triples in representation...
- Can require representation to include all inferred types
  - large model
  - in general expensive to check requirements
  - convenient for further processing
- Can restrict to simpler, less expensive reasoning.
  - E.g. restrict expressivity of models.
  - smaller model
  - checking requirements affordable
  - affordable inference needed for further processing

Types and Reality

Types and Reality

Requirement 2.3.6

All types any resource *belongs to* are explicitly given or can be inferred.

Types and Reality

Requirement 2.3.6

All types any resource *belongs to* are explicitly given or can be inferred.

- What types does an object actually belong to?

Types and Reality

Requirement 2.3.6

All types any resource *belongs to* are explicitly given or can be inferred.

- What types does an object actually belong to?
  - `:s4711` may be an “air pressure sensor” even if that is not stated or inferable.

Types and Reality

Requirement 2.3.6

All types any resource *belongs to* are explicitly given or can be inferred.

- What types does an object actually belong to?
  - `:s4711` may be an “air pressure sensor” even if that is not stated or inferable.
- Can’t check mechanically that all relevant aspects of reality were accurately represented.

Types and Reality

Requirement 2.3.6

All types any resource *belongs to* are explicitly given or can be inferred.

- What types does an object actually belong to?
  - `:s4711` may be an “air pressure sensor” even if that is not stated or inferable.
- Can’t check mechanically that all relevant aspects of reality were accurately represented.
- Checking requires human action like e.g. code review

Further Criteria

- Literals: typed or with a language

Further Criteria

- Literals: typed or with a language
- Consistency: description must not contain contradictions

Further Criteria

- Literals: typed or with a language
- Consistency: description must not contain contradictions
- ISO15926 Part 8 compliance

⋮

Further Criteria

- Literals: typed or with a language
- Consistency: description must not contain contradictions
- ISO15926 Part 8 compliance

  ⋮

- Currently, 14 formulated requirements

Implementability of Criteria

- Different implementability status of criteria:

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us
  - algorithm known but non-standard

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us
  - algorithm known but non-standard
  - implemented by standard reasoning tools

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us
  - algorithm known but non-standard
  - implemented by standard reasoning tools
  - implementable as a set of SPARQL queries that can be computed from the domain ontology

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us
  - algorithm known but non-standard
  - implemented by standard reasoning tools
  - implementable as a set of SPARQL queries that can be computed from the domain ontology
  - implementable as static SPARQL queries, independent of the domain vocabulary.

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us
  - algorithm known but non-standard
  - implemented by standard reasoning tools
  - implementable as a set of SPARQL queries that can be computed from the domain ontology
  - implementable as static SPARQL queries, independent of the domain vocabulary.
- choice of requirements to apply will depend on

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us
  - algorithm known but non-standard
  - implemented by standard reasoning tools
  - implementable as a set of SPARQL queries that can be computed from the domain ontology
  - implementable as static SPARQL queries, independent of the domain vocabulary.
- choice of requirements to apply will depend on
  - application context

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us
  - algorithm known but non-standard
  - implemented by standard reasoning tools
  - implementable as a set of SPARQL queries that can be computed from the domain ontology
  - implementable as static SPARQL queries, independent of the domain vocabulary.
- choice of requirements to apply will depend on
  - application context
  - available resources

Implementability of Criteria

- Different implementability status of criteria:
  - undecidable
  - algorithm not known to us
  - algorithm known but non-standard
  - implemented by standard reasoning tools
  - implementable as a set of SPARQL queries that can be computed from the domain ontology
  - implementable as static SPARQL queries, independent of the domain vocabulary.
- choice of requirements to apply will depend on
  - application context
  - available resources
- Implementation of a validation tool is underway at DNV and UiO

Conclusion

- Investigated use of RDF for installation descriptions

Conclusion

- Investigated use of RDF for installation descriptions
- Features of RDF not required in this context:

Conclusion

- Investigated use of RDF for installation descriptions
- Features of RDF not required in this context:
  - Inhomogeneity (e.g. mixing model and data)

Conclusion

- Investigated use of RDF for installation descriptions
- Features of RDF not required in this context:
  - Inhomogeneity (e.g. mixing model and data)
  - Vagueness (e.g. generic type information)

Conclusion

- Investigated use of RDF for installation descriptions
- Features of RDF not required in this context:
  - Inhomogeneity (e.g. mixing model and data)
  - Vagueness (e.g. generic type information)
- Identified 14 requirements for RDF used in this context

Conclusion

- Investigated use of RDF for installation descriptions
- Features of RDF not required in this context:
  - Inhomogeneity (e.g. mixing model and data)
  - Vagueness (e.g. generic type information)
- Identified 14 requirements for RDF used in this context
- Requirements have widely differing computational status

Conclusion

- Investigated use of RDF for installation descriptions
- Features of RDF not required in this context:
  - Inhomogeneity (e.g. mixing model and data)
  - Vagueness (e.g. generic type information)
- Identified 14 requirements for RDF used in this context
- Requirements have widely differing computational status
- Implementation of requirement checking tool is ongoing