

Towards a Uniform User Interface for Editing Data Shapes

Ben De Meester, ✉ ben.demeester@ugent.be, 🐦 @Ben_DM
Pieter Heyvaert, Anastasia Dimou, and Ruben Verborgh

*IDLab, Department of Electronics and Information Systems,
Ghent University – imec, Ghent, Belgium*

Garbage In, Garbage Out?

Data shapes

Easy editing is important

“Fitness for use”

Constraint languages:

declaration and implementation are decoupled

SHACL (W3C Recommendation)

ShEx

...

Machine-processability in mind

**What are the necessary
features for *visually*
editing data shapes?**

Outline

- 1 SOTA
- 2 Features
- 3 PoC: unSHACLed
 UI
 Features
- 4 Conclusions

Outline

- 1 **SOTA**
- 2 Features
- 3 PoC: unSHACLed
UI
Features
- 4 Conclusions

State of the Art

Data shapes

Validation based on: OWL | SPARQL | SHACL, ShEx

Data shape editors

Depend on the (constraint) language or
enforce a linear workflow

Editors

Data editors: text-based, form-based, use-case specific

Ontology editors: graph-based, indented-tree-based, UML-based

SPARQL editors: text-based

Linked Data generation rule editors: form-based, graph-based

Outline

1 SOTA

2 **Features**

3 PoC: unSHACLed

UI

Features

4 Conclusions

Desired Features for Data Shape Editing

- 1 Independence of constraint language
- 2 Support multiple data sources
- 3 Support different serializations
- 4 Support multiple ontologies
- 5 Multiple alternative editing approaches
- 6 Non-linear workflows
- 7 Independence of execution

Outline

1 SOTA

2 Features

3 **PoC: unSHACLed**

UI

Features

4 Conclusions

unSHACLed

Visual Data Shapes editor as Web application

Drag-and-drop loaded data graphs and data shapes

Add data shapes using templates

Get visual feedback on conformance

Export shape

<https://w3id.org/imec/unshacled/app>

The unSHACLED UI, consisting of an *Overview Sidebar* (left), an *Action Toolbar* (top), and an *Editing Area* (middle-right)

The screenshot displays the unSHACLED user interface. On the left is the **Overview Sidebar** (labeled **a)**, which includes a search bar and a list of components under the 'Template' section: SHACL, Shape, Node Shape, Property Shape, and Template. A button 'Add template from selection' is highlighted. At the top is the **Action Toolbar** (labeled **b)**, featuring a menu, search, navigation, and utility icons, along with 'Save Graph' and 'Conformance errors' buttons. The main **Editing Area** (labeled **c)** shows a grid of components: 'ex:Person Shape', 'ex:ssn', 'ex:worksFor', and '._b2'. Each component has a list of properties and values. Blue arrows (labeled **d)** indicate connections from the 'ex:ssn' component to the 'ex:Person Shape' and '._b2' components. On the right, a red box labeled **e)** represents an instance 'ex:Bob' with properties: 'rdf:type: ex:Person', 'ex:ssn: "123-45-6789"', and 'ex:ssn: "124-35-6789"'. The background is a light gray grid.

Different elements enable the different features

Features	UI Element		
	<i>Overview Sidebar</i>	<i>Action Toolbar</i>	<i>Editing Area</i>
F1. Independence of constraint language	✓		✓
F2. Support multiple data sources		✓	✓
F3. Support different serializations		✓	
F4. Support multiple ontologies		✓	
F5. Multiple alternative modeling approaches			✓
F6. Non-linear workflows	✓	✓	✓
F7. Independence of execution		✓	

Outline

1 SOTA

2 Features

3 PoC: unSHACLed
UI
Features

4 **Conclusions**

Conclusions

No need to write RDF / SHACL / ShEx

Use-case independent

Open Issues

- (Map)VOWL or UML or ... ?

- User evaluation graphical representation

- Representation large data shapes

 - Workspaces

 - Detail levels

Features as starting point for visual data shape editors

Towards a Uniform User Interface for Editing Data Shapes

Ben De Meester, ✉ ben.demeester@ugent.be, 🐦 @Ben_DM
Pieter Heyvaert, Anastasia Dimou, and Ruben Verborgh

*IDLab, Department of Electronics and Information Systems,
Ghent University – imec, Ghent, Belgium*