An approach to increase the usability of Shape Expressions editors

Pablo Menendez[a*], and Jose Emilio Labra-Gayo[a]
[a] *Dept. of Computer Science, University of Oviedo, Spain*
Emails:pmenendz@gmail.com , labra@uniovi.es

Abstract. There is a need to increase the number of tools that support the use of Shape Expressions language (ShEx). In this paper we present YASHE, a ShEx text editor that incorporate new features with respect to the existing ones. It takes the SPARQL text editor YASQE as a starting point and adapts and extends it to the needs of the language. We also present ShExAuthor, a graphical assistant for ShEx schema creation inspired by the WDQS of Wikidata. We have carried out a usability experiment with 16 non-expert users to compare four ShEx editing tools. The results showed no statistically significant differences in terms of time and completeness percentage (CP) between the tested tools. However, our tools obtained better results in CP and YASHE obtained the highest score in terms of precision (time to CP ratio).

Keywords. Shape Expressions, Text Editor, Graphical Editor, Usability

1. Introduction

For a language, to have a good adoption within the community, it is necessary that there are tools that favor its use. Unfortunately, in the case of ShEx[3] (Shape Expressions), there are not many tools. A basic tool for any language is to have a text editor adapted to the syntax and nature of the language. One of the editors that introduced new features in the SPARQL[16] clients ecosystem was the one incorporated by YASGUI[2]. Later extracted as an independent module and known as YASQE[10]. Today it is used by multiple triple stores, publishers, and other applications[14].
In this paper we present a text editor for ShEx known as YASHE. It takes YASQE as a starting point and adapts and extends it to the needs of the language. Apart from incorporating features common to any text editor such as: line numbering, colored syntax, and syntactic error detection, it adds others related to the language domain itself. Some of them are the suggestion of prefixes commonly used in the Semantic Web and the possibility of autocompleting Wikidata identifiers through their name.

On the other hand, the major knowledge sources of the Semantic Web, such as Wikidata[13] or DBPedia[12], require human interaction to obtain a large part of the data they offer. These data belong to very different fields of knowledge. Therefore, the participation of domain experts is necessary. Wikidata makes use of languages such as SPARQL for data querying and has recently incorporated ShEx schemas (known as

[*] Corresponding author: Pablo Menendez, pmenendz@gmail.com

EntitySchemas[1]). One of the problems faced by domain experts when working with these technologies is the fact that they don´t need to be accustomed to the use of computer languages. On the contrary, they may find a graphical interface more comfortable. Wikidata offers, in its SPARQL playground known as Wikidata Query Service (WDQS)[2], a query graphical assistant.

In this paper, we present a tool for ShEx development, known as ShExAuthor, which offers a shapes graphical assistant inspired by that of WDQS. This tool integrates YASHE into its system to visualize the shapes created from the wizard. Both components, editor and assistant, communicate with each other to remain consistent. Thus, allowing the user to always make use of the one that best suits him.

The research questions studied in this paper are the following:

- RQ1: Is YASHE more usable for non-expert users compared to other ShEx development tools?
- RQ2: Does the use of a graphical shape assistant support the creation and use of ShEx schemas by non-expert users?

The rest of the paper is structured as follows. First, we will discuss the related work and compare the features of the ShEx tools that we consider most relevant. Next, the system architecture will be described. Afterwards, the methodology followed for the experiment will be explained, followed by the results and discussions. Finally, we will briefly describe the impact of one of the tools and end with conclusions and future work.

2. Related Work

In this section we discuss the existing text editors for ShEx(2.2) and their features (2.3).

2.1. ShEx text editors and related tools

ShEx2- Simple Online Validator[3] (to abbreviate ShEx2 from now on) is an online tool that allows to perform RDF[15] data validation using Shape Expressions. It incorporates an editor for ShEx where we can write the schemas to conduct the validation. One of the most key features offered by the tool for editing Shape Expressions is the detection of syntactic errors. This feature is not performed automatically but occurs once the validation is performed.
Wikidata has recently incorporated the EntitySchemas[4] in its system. EntitySchemas is Wikidata's way of calling items that are schemas written in ShEx. For the creation of this items, they offer a tool (we will refer to it as wikidata from now on) that provides the user with a plain text editor where he can create write his shapes. This editor does not have any features for the user except the possibility to do and redo. However, once the EntitySchema is created, it offers a colored syntax to visualize it.

[1] https://www.wikidata.org/wiki/Wikidata:WikiProject_Schemas
[2] https://query.wikidata.org/
[3] https://rawgit.com/shexSpec/shex.js/master/packages/shex-webapp/doc/shex-simple.html
[4] https://www.wikidata.org/wiki/Wikidata:WikiProject_Schemas

Ace-shexc-user[5] is a Shape Expressions text editor born from the ace[6] library that provides colored syntax and real-time grammar error detection.
Validata[6] is a tool to help generate RDF documents by validating against schemas written in Shape Expressions. This tool offers us the possibility to create our own Schema through a text editor with some features such as line numbering and the possibility to undo and redo operations.
Shape designer[5] is a graphical tool for the creation of schemas in ShEx and SHACL[11] given an RDF dataset. It incorporates a text editor for both languages with features such as colored syntax and line numbering.
As for more popular editors, such as Sublime Editor[7], there is a plugin for ShEx called **ShEx_Sublime_package**[8] that offers a colored syntax as well as some autocompletion mechanisms mainly focused on Wikidata. There is also a **ShEx extension for Vscode** [9] that offers colored syntax and Wikidata snippets.

2.2. ShEx tools features

Table 1 shows the functionalities available in 9 tools that support the creation and editing of Shape Expressions. Among these features we include those that seem to us basic for any editing tool, as well as others that are more domain specific. Some of the tools are dedicated editors intended entirely for ShEx editing while others have another specific purpose.

YASHE and ShExAuthor have the highest number of functionalities with respect to the other tools (13/16). Some of the most basic ones, such as colored syntax or line numbering are present in 7 of the 9 defined tools. However, there are other functionalities that are only available in some of them. One of these functionalities is validation. Among all the tools with which we compared ourselves, only ShEx2-Simple Online Validator and Shape Designer have this functionality. Using these tools, we can set the data to validate or in the case of ShEx2 run a SPARQL query against endpoints such as Wikidata and automatically validate each of the results against our schema. This is very useful when creating and editing shapes since we can directly check the integrity of our data or those of large sources of knowledge and check if our shapes are well constructed. YASHE and ShExAuthor do not have this functionality, as we consider that our tools focus only on editing tasks, leaving it to other tools to integrate this type of tasks.

Another functionality present in only one tool is the possibility to edit and create EntitySchemas. Wikidata offers a plain text editor to perform this task. This editor only offers 2 of the 17 features defined in Table1. This could facilitate the appearance of grammatical errors in the EntitySchemas. YASHE has a search engine to visualize all the EntitySchemas of Wikidata. However, there is no possibility to save new EntitySchemas or update existing ones directly from YASHE.

Grammatical error detection is present in 4 of the 9 tools compared. In ShEx2, the detection of grammatical errors is not performed in real time, but it is necessary to try to

[5] https://github.com/shexSpec/ace-shexc-user
[6] https://ace.c9.io/
[7] https://www.sublimetext.com/
[8] https://github.com/andrawaag/Shex_Sublime_package
[9] https://github.com/weso/vscode-shex-extensions/tree/master/shex-languaje-extension

	ShEx2	Wikidata	Ace ShEx User	Validata	Shape Designer	Sublime Plugin	Vscode Plugin	YASHE	ShExAuthor
Web Tool	+	+	+	+	-	-	-	+	+
Line Numbers	-	-	+	+	+	+	+	+	+[e]
Syntax highlighting	-	+/- [a]	+	-	+	+	+	+	+[e]
Error Checking	+[g]	-	+[h]	-	-	-	-	+[h]	+[eh]
Autocompleters	-	-	-	-	-	+[b]	+[c]	+[d]	+[e]
Tooltips	-	-	-	-	-	-	-	+[i]	+[ei]
Undo/Redo	+	+	+	+	+	+	+	+	+[e]
Dark mode	-	-	-	-	-	+	+	+	+[e]
Load	-	-	-	-	+	+	+	+	+[e]
Download	-	-	-	-	+	+	+	+	+[e]
Share Schema	-	-	-	-	-	-	-	+	+[e]
Validation	+	-	-	-	+	-	-	-	-
Save Entity Schema	-	+	-	-	-	-	-	-[f]	-
Ghrapic Assistant	-	-	-	-	+	-	-	-	+
Pretty Pritner	-	-	-	-	-		-	+	+

Table 1. ShEx Tools Features Matrix

[a] Color syntax it´s not available while editing. It´s only visible after saving the schema.
[b] Autocompletion of Wikidata prefixes and Wikidata items.
[c] Autocompletion of Wikidata prefixes.
[d] Autocompletion of prefixes, aliases, keywords, defined shapes and Wikidata and Wikibase items.
[e] Feature inherited by YASHE..
[f] EntitySchemas can be searched and displayed.
[g] Not real time error checking.
[h] Real time error checking.
[i] Tooltips for Wikidata and Wikibase items

carry out a validation so that the tool notifies us with the errors made in case there are any. Ace-shexc-user, YASHE and ShExAuthor notify errors in real time, while editing is in progress.

No web tool supports autocompletion mechanisms except the ones presented in this paper. The plugin for Vscode offers the possibility to autocomplete Wikidata prefixes. The plugin for Sublime allows autocompletion of Wikidata prefixes and items.

The graphical assistant feature is only present in 2 tools. On the one hand, Shape Designer offers a graphical interface for the definition of shape patterns together with a node selection query for the automatic creation of schemas. On the other hand, ShExAuthor offers a graphical editor for shape creation using a system of boxes and colors. Both systems offer a text editor to visualize the schema under construction.

3. Description

In this section we will describe the systems that have been developed. Both will be described, and aspects related to their architecture, their graphical interface, their functionalities, and their limitations will be explained.

3.1. YASHE

YASHE is a ShEx text editor that was born as a fork of the YASQE editor, which is based on SPARQL. It offers a working environment suitable for creating and editing schemas in ShEx. It has a Web site where the editor can be used (www.weso.es/YASHE/). However, this tool is intended to be easily integrated into other web applications, as we will see later.

3.1.1. Architecture

This tool runs completely server-side and is mainly developed in HTML5[19] and JavaScript[17], making use of libraries such as jQuery[10] and Codemirror[11]. Codemirror provides a basic text editor for any language, with features such as colored syntax and auto-completion mechanisms. YASHE takes this library as a base, adapts it, and extends it to meet the needs of ShEx. All the functionalities offered by the library are accessible from YASHE, making it highly customizable. FlintSparqlEditor defines a grammar for SPARQL in Prolog that generates a symbol table that Codemirror is able to interpret. This symbol table establishes a relationship between tokens that allows us to know which tokens can precede each other. Our system adapts this library for ShEx and makes use of the symbol table to perform a lexical and syntactic analysis of the editor's content, thus being able to provide a real-time grammatical error detection mechanism. This analysis is performed every time there is a change in the content of the editor. The tokens are checked one by one and when an incorrect token is detected, the user is notified indicating the line where the error occurred.

3.1.2. Features

The most relevant features of the tool are described below:

1. **Syntax highlighting:** This feature is intended to assist in the identification and differentiation of the most important elements of the language.
2. **Syntax Checking:** Syntax errors are reported to the user each time a change is made in the editor. These errors are accompanied by a help message to guide the user on the errors made.
3. **Autocompleters:** The autocompletion mechanisms are one of the most useful features of the tool. They allow us to save time when typing, as well as provide us with information that we might not have been aware of previously. YASHE offers the following auto-completion mechanisms:
    a. **Prefixes:** Within the world of the semantic web and of the languages that form it, the definition of prefixes plays a very important role. The number of prefixes is very high, and it is tedious to have to remember

[10] https://jquery.com/
[11] https://codemirror.net/

each of them. By using Prefix.cc[12], YASHE offers a list with the most used ones every time a prefix is defined . We can search through this list and autocomplete.

   b. ***Keywords, Alias, and Shapes:*** This mechanism offers the possibility to autocomplete language keywords (*prefix, closed, minlength*, etc.), aliases of prefixes that are already defined (*schema: , xsd:*), or Shapes that are also defined (@<*human*>).
   c. **Wikidata Items:** One of the big problems when working with Wikidata and languages like ShEx or RDF is the problem of identifiers. The way wikidata works with elements in a unique way is by assigning them an identifier. This identifier is given by a letter (Q if it is an entity or P if it is a property) followed by a number. This solution makes it possible to treat the elements in a unique way, but it is not very readable for humans when we refer to one of these elements from these languages. In addition, it is necessary for users to learn these identifiers. Currently wikidata has millions of items[13], which makes it impossible for humans to retain such a large amount of information. Therefore, every time we want to reference an element of Wikidata we will have to go to its website and look for its associated identifier. YASHE offers the possibility to perform a search by name from the editor itself, allowing to autocomplete its unique identifier once we have found the desired item.
   d. **Wikibase Items:** We can make use of the above functionality for any instance of Wikibase without the need to configure anything additional in YASHE. It takes care through the defined prefixes to search for possible Wikibase instances and allow us to search and suggest items from those instances in the same way as for Wikidata.

4. **Tooltips:** The problem associated with Wikidata identifiers mentioned above in the section on autocompletion mechanisms, not only affects when writing our schemas but also when reading them. So, when we want to read a schema that makes use of many Wikidata elements the task becomes highly tedious. That is why YASHE offers visual help mechanisms (tooltips) that show us information of the Wikidata element over which we pass our mouse. As for the autocomplete mechanisms, this functionality is available for other Wikibase instances than Wikidata.

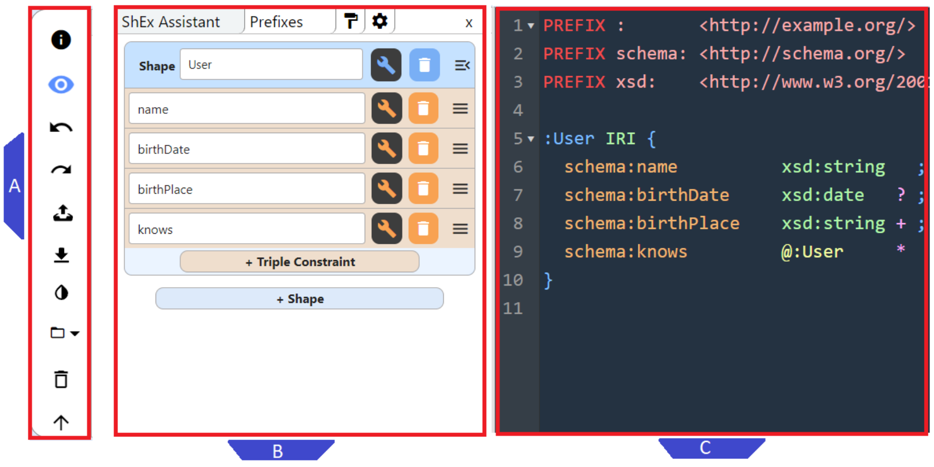
3.2. ShExAuthor

ShExAuthor is a web application that provides the user with a graphical assistant for the creation of Shape Expressions (https://www.weso.es/shex-author/). It integrates YASHE into its system to display the created shapes and to allow the user to interact with both the editor and the assistant according to his preferences. When a change is made in the assistant it is propagated to the editor and vice versa. It runs entirely client-side and is developed mainly in HTML5 and JavaScript making use of the React.js library[14].

[12] https://prefix.cc/

[13] https://www.wikidata.org/wiki/Wikidata:Statistics/en

[14] https://es.reactjs.org/



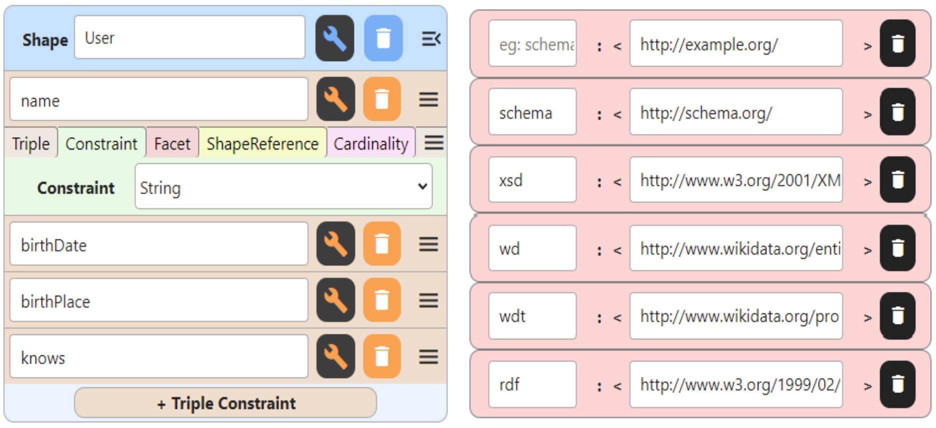
```
PREFIX :       <http://example.org/>
PREFIX schema: <http://schema.org/>
PREFIX xsd:    <http://www.w3.org/200

:User IRI {
  schema:name          xsd:string   ;
  schema:birthDate     xsd:date   ? ;
  schema:birthPlace    xsd:string + ;
  schema:knows         @:User     *
}
```

Figure 1. ShExAuthor user interface which consists of: a tool panel (A), the graphic assistant (B) and a ShEx text editor (YASHE) (C)

ShExAuthor is intended to be used by users not so closely related to the IT world who are more accustomed to using visual tools rather than computer languages, such as, for example, domain experts from other non-IT fields. These play a important role in the Semantic Web, since it needs their knowledge to represent all kinds of information, or as in the case of ShEx, to validate it. If we want to validate information about biological genes, we need to create a Schema that allows us to carry out this task. And for this, it is necessary to have a domain expert who has the necessary knowledge in the field. However, it is very likely that this domain expert does not have any knowledge in ShEx, so he/she will have to transfer his/her knowledge to a person who does. What ShExAuthor aims to achieve is that the domain expert himself creates the schemas through an environment that is comfortable and familiar to him, without the need to delegate this task to someone else.

3.2.1. User Interface

The ShExAuthor user interface consists of three main panels or components: tools panel (Figure 1a), graphical assistant(Figure 1b) and text editor (Figure 1c).
The tools panel allows us to perform do/undo operations, load and unload our schemas, toggle light/dark mode or even load sample examples.
The graphical assistant represents the content of the text editor in a graphical way through a system of boxes for the representation of shapes, triple constraints and prefixes and a color code for each language element. The assistant has a main tab where the shapes are represented and a secondary tab for the representation of prefixes. Each shape is represented by a blue box containing a series of triple constraints. (Figure 2 left). We can configure features of the shape itself (e.g. *prefix*) or we can create new triple constraints or individually configure the existing ones. Each triple constraint is represented by a brown box. We can configure it by clicking on the wrench of its color. Figure 2 left shows the triple constraint that establishes a "name" constraint of type "string". Each one has the following configuration tabs:


Shape User

name

Triple | Constraint | Facet | ShapeReference | Cardinality

Constraint String

birthDate

birthPlace

knows

+ Triple Constraint

eg: schema	: <	http://example.org/ >
schema	: <	http://schema.org/ >
xsd	: <	http://www.w3.org/2001/XM >
wd	: <	http://www.wikidata.org/enti >
wdt	: <	http://www.wikidata.org/pro >
rdf	: <	http://www.w3.org/1999/02/ >

Figure 2. Shape and tiple constraints representation (left) and prefixes representation (right).

- ***Triple:*** In this tab we can configure the type of the triple constraint itself. We can define if it is going to be an IRI (<...>) or if it is going to use a prefix and, in that case, we can set which one it is going to use.
- ***Constraint:*** The constraint tab allows us to set the type of constraints set by the triple constraint. By default, it offers primitive values (string, integer, date and boolean) but it is possible to carry out a more advanced configuration (*valuesets, literal, bnnode*, etc.).
- ***Facet:*** This tab allows us to set length restrictions (*length, minlength, maxlength, totaldigits,* etc.).
- ***ShapeReference:*** In this tab we can reference another Shape already created (*@:User*).
- ***Cardinality:*** The cardinality tab allows us to set all the restrictions related to cardinality (Exactly one, zero or more, one or more, range, etc.).

All these will be represented in the text editor in the same color as they appear in the assistant.

The prefixes are represented by a pink box (Figure 2 right). Each of them has two fields to be filled in by the user: the prefix alias and its IRI. We can create new prefixes, edit existing ones, or delete them. To use the prefixes from the graphical assistant, they must be defined here.

The text editor (YASHE) is a representation of the shapes and triples created in the graphical assistant. In addition, we can interact directly with it so that the changes we make will be updated automatically in the graphical assistant. When we create a new shape or edit an existing one, YASHE notifies the graphical assistant so that it synchronizes with the new content offered in the editor.

3.2.2. Limitations

There are some shapes that cannot be represented in the graphical assistant. One of the existing limitations in ShExAuthor are the nested Shapes. In ShEx we can define an *inlineshape* (Validating RDF Data[18] chapter 4.6.5) inside a triple constraint. Inside this new Shape we will define new triple constraints. This causes that the nesting of

Shapes does not have any kind of limit. In plain text it is not a problem to represent this type of shapes (maybe to read or understand them). However, when representing them in a graphical assistant it is a problem. However, ShExAuthor offers support for this type of shapes up to a nesting of at least 5 levels depending on the monitor size.
Another limitation of the tool is the representation of logical operators. In ShEx we can define a shape with a series of triple constraints and/or others (using the OR/AND operator). To represent this within the graphical assistant would require increasing the number of interface components significantly, thus raising the degree of complexity of the interface and making it less visually appealing. That is why ShExAuthor does not support this type of language operators.
Comments are not represented in the wizard either. We can write comments in the text editor, however, ShExAuthor will remove them once we make use of the assistant.
When ShExAuthor detects any of the elements that represent a limitation, it disables the assistant by displaying an error message to the user specifying that the Shape entered in the text editor is too complex to be represented by the assistant.

4. Methodology

Since 1893, proposals have been made to evaluate text editors [1,7]. The methodology we have followed is based on the one used in [8]. The documents used for the experiment (manuals, examples, tasks, etc.) are available at https://github.com/mistermboy/shex-edit-tools-paper.

4.1. Experiment Design

The experiment aims to make a comparison between 4 of the tools shown in Table 1. On the one hand, the 2 presented in this paper (YASHE and ShExAuthor) and on the other hand the 2 that we consider most important in the current scenario. One of them is ShEx2, which together with RDFShape[4] are the only validation tools recommended on the official ShEx website [https://shex.io/]. We do not compare ourselves with RDFShape as this tool integrates YASHE into its system to support schema creation and editing. The other tool we compare ourselves with is the one provided by Wikidata for the creation of EntitySchemas as this is the largest source of knowledge of the Semantic Web and one of the strongest projects betting on ShEx.
The experiment consisted of two tasks of editing and creation of Shape Expressions. The first task asked for the creation of a schema for the validation of an RDF model. The second task consisted in the creation of a schema where a series of constraints were collected using Wikidata properties.
During the experiment, quantitative measurements were taken using the Mousotron[15] tool, which provided us with measurements such as distance traveled with the mouse, clicks, number of keystrokes, etc. Qualitative measures were also collected through an online Office 365 questionnaire. In it, 12 statements were established where users had to express their agreement based on a Likert scale[9]. The statements collected in the questionnaire were as follows:

[15] https://www.blacksunsoftware.com/mousotron.html

S1. The experience with the tool was satisfactory.
S2. The tool was easy to use
S3. The appearance of the tool seems to me to be adequate.
S4. It was easy to learn how to use the tool.
S5. I consider that the tool could be useful in my job, university, etc.
S6. The tool is intuitive.
S7. The tool leads to commit some errors.
S8. The tool favors the use of ShEx.
S9. The system has a quick response.
S10. The functionalities offered by the tool seem to me sufficient for a shapes creation tool.
S11. The functionalities offered by the tool to work with Wikidata seem useful to me
S12. The error messages were useful to solve problems.

4.2. Conduction

We had 16 students of the Master in Web Engineering of the University of Oviedo (14 boys and 2 girls). All of them had a degree (240 FTE credits) in Computer Science and 12 of them were taking the course New Aspects in Semantic Web, a two-week course (3 hours per day) where they were taught aspects such as RDF or ShEx. The remaining 4 had taken the course the previous year and all of them had passed it. Prior to taking this course they had no knowledge of Semantic Web. All of them were asked for prior consent.

For the students who were taking the course, the experiment was carried out in the usual classroom, so that the students were in a place that was familiar to them and that could not interfere negatively with the results. In addition, it was carried out during the last session of the course, so that the students had a minimum knowledge of the subject.

Students who had already taken the course the previous year were given the same test, but online, where each student could use his or her own computer.

To carry out the experiment, each student was randomly assigned a tool and was provided with instructions and a manual for the use of the assigned tool. They were also provided with a document with examples of schema construction for the validation of RDF data and Wikidata items.

The instructions to be followed by each student were as follows:

1. Open the web page of the assigned tool and delete the given example
2. Open Mousotron, restart it and press the start button.
3. Start with the first task.
4. Once the first task is finished, stop the Mousotron and capture the results.
5. Restart the Mousotron.
6. Start with the second task.
7. Once the second task is finished, stop the Mousotron and capture the results.
8. Fill in the Office 365 questionnaire.

4.3. Analysis

Both qualitative and quantitative data were collected and anonymized. It was necessary to calculate some of the variables using those provided by the students. The completeness percentage (CP) was calculated using three measures: the number of correct prefixes generated 16%, the number of correct shapes generated 13% and the number of well generated and required triple constraints 71%. We have assigned these values by calculating the percentage of each element generated with respect to the total number of elements generated.
Being P the number of prefixes, the calculation of the completness percentage for the prefixes can be calculated as:

$$CPPrefixes = 1 - \frac{Pgenerated - Pcorrect}{Pgenerated}$$

$P_{generated}$ represents the generated prefixes and Pcorrect the number of them generated correctly. Being S the number of shapes, the calculation of the CP for the shapes can be calculated as:

$$CPShapes = 1 - \frac{Sgenerated - Scorrect}{Sgenerated}$$

$S_{generated}$ represents the generated shapes and $S_{correct}$ represents the number of correctly generated shapes. We do not consider that the triple constraints are well generated when determining if the shape is correct or not.TC being the number of triple constraints, the calculation of the CP for the triple constraints can be calculated as:

$$CPTripleConstraints = 1 - \frac{TCneeded - TCcorrect}{TCneeded}$$

TC_{needed} represents the number of triple constraints necessary to perform the exercise correctly. $TC_{correct}$ represents the number of correct triple constraints generated by the user. If the user uses a single triple constraint to represent a number of constraints greater than 1, this triple constraint will have a value equal to the number of constraints it fulfills. Therefore, the total CP can be represented as follows:

$$CP = 0.16 * CPPrefixes + 0.13 * CPShapes + \\ 0.71 * CPTripleConstraints$$

Finally, we calculate the precision. This value establishes a relationship between the time consumed to perform the task and the time of the fastest user, considering the CP obtained. Being T_{un} the elapsed time of user n and CP_{un} the CP of user n, we can calculate the accuracy as follows:

$$Precision = \frac{Tun}{min\ (\{Tu1, \dots, Tun\})} * CPu$$

The other variables used for the analysis were: number of keyboard keys pressed (*KeyStrokes*), number of left mouse button clicks (*LeftButton*), number of right mouse button clicks (*RightButton*), number of double clicks with the left mouse button (*DoubleClicks*) and number of scrolls made with the mouse (*MouseWheel*).
IBM SPSS version 28.0.1.1 was used for statistical analysis. Comparisons between the four groups were performed using a One Way ANOVA where a normal distribution was assumed, and outliers were discarded when necessary.

5. Results

Of the 16 users we had for the experiment, one of them, who had been assigned the ShExAuthor tool, did not send his results or complete the online questionnaire. Another of them, who used the ShEx2, did not complete the online questionnaire either.

Measure	Group	Mean	Std. Deviation	Minimum	Maximum
Time	ShExAuthor	667.0000	253.2252	378.0000	850.0000
	YASHE	691.2500	379.4429	356.0000	1175.0000
	ShEx2	1003.5000	453.8564	447.0000	1480.0000
	Wikidata	384.5000	72.7668	303.0000	480.0000
Distance	ShExAuthor	1707.0000	720.1548	1096.0000	2501.0000
	YASHE	1201.2500	1075.6779	281.0000	2554.0000
	ShEx2	931.2500	781.3289	253.0000	1896.0000
	Wikidata	436.2500	255.5783	164.0000	724.0000
KeyStrokes	ShExAuthor	462.0000	149.9233	339.0000	629.0000
	YASHE	868.5000	307.0228	510.0000	1260.0000
	ShEx2	943.0000	84.2496	818.0000	1001.0000
	Wikidata	797.5000	155.0495	653.0000	1016.0000
LeftButton	ShExAuthor	282.0000	25.4558	264.0000	300.0000
	YASHE	120.2500	62.2756	72.0000	205.0000
	ShEx2	106.0000	86.4214	34.0000	217.0000
	Wikidata	48.0000	20.8966	20.0000	69.0000
RightButton	ShExAuthor	3.5000	2.1213	2.0000	5.0000
	YASHE	1.5000	1.2910	0.0000	3.0000
	ShEx2	0.7500	0.9574	0.0000	2.0000
	Wikidata	0.0000	0.0000	0.0000	0.0000
DoubleClicks	ShExAuthor	23.5000	16.2635	12.0000	35.0000
	YASHE	14.7500	8.0571	3.0000	20.0000
	ShEx2	14.5000	17.5404	2.0000	40.0000
	Wikidata	3.0000	2.8284	1.0000	7.0000
MouseWheel	ShExAuthor	1884.5000	2366.6864	211.0000	3558.0000
	YASHE	620.3333	405.9191	324.0000	1083.0000
	ShEx2	98.2500	74.6654	5.0000	168.0000
	Wikidata	52.6667	47.1628	0.0000	91.0000
Completness Percentage	ShExAuthor	0.8542	0.2190	0.6023	1.0000
	YASHE	0.8509	0.1699	0.6023	0.9602
	ShEx2	0.6262	0.4244	0.0000	0.9403
	Wikidata	0.5722	0.3359	0.0785	0.7933
Precision	ShExAuthor	0.4588	0.2835	0.2147	0.7697
	YASHE	0.4876	0.2799	0.1553	0.7496
	ShEx2	0.2583	0.2120	0.0000	0.5161
	Wikidata	0.4641	0.3009	0.0636	0.7933

Table 2. Descriptive statistics for task 1.

In Table 2 we can observe the descriptive statistics for the quantitative results of task 1. The comparison between the four groups in terms of a One Way ANOVA showed statistically significant differences between the four tools and the number of left clicks made by the users (*LeftButton*) F(3,10)=6.850, p=0.009, ω=0.673. Subsequently, Tukey's test determined that the number of left clicks performed by users using the ShExAuthor tool (282 ± 18 clicks) is higher than the number of clicks performed by users using ShEx2 (106 ± 43.21 clicks), YASHE(120.25 ± 31.138 clicks) and Wikidata (48 ± 10.448 clicks). Statistically significant differences were also found among the four tools and the number of right clicks made by users (*RightButton*) F(3,10)=4.750, p=0.026, ω=0.588. Scheffé's test determined that the number of right clicks performed by users using the ShExAuthor tool (3.5 ± 1.5 clicks) is higher than the number of clicks performed by users using the Wikidata tool (0 ± 0 clicks). Finally, statistically significant differences were found between the four tools and the number of keystrokes made by users (*KeyStrokes*) F(3,11)=3.841, p=0.042, ω=0.512. Tukey's test determined that the number of keystrokes performed by users using the ShExAuthor tool (462 ± 86,558 keystrokes) is lower than the number of clicks performed by users using the ShEx2 tool (943 ± 42,125 keystrokes).

Measure	Group	Mean	Std. Deviation	Minimum	Maximum
Time	ShExAuthor	676.7500	226.7486	493.0000	1007.0000
	YASHE	449.6667	174.7350	248.0000	556.0000
	ShEx2	849.2500	122.5843	754.0000	1015.0000
	Wikidata	713.5000	452.9087	278.0000	1212.0000
Distance	ShExAuthor	1887.5000	523.4721	1382.0000	2472.0000
	YASHE	557.6667	348.6909	197.0000	893.0000
	ShEx2	1228.7500	987.5942	0.0000	2418.0000
	Wikidata	1622.2500	1342.8406	149.0000	3036.0000
KeyStrokes	ShExAuthor	470.5000	209.5272	166.0000	645.0000
	YASHE	683.0000	256.5093	391.0000	872.0000
	ShEx2	696.6667	378.9318	270.0000	994.0000
	Wikidata	871.7500	406.2449	512.0000	1398.0000
LeftButton	ShExAuthor	238.6667	120.9807	141.0000	374.0000
	YASHE	72.3333	37.5810	29.0000	96.0000
	ShEx2	184.3333	80.2579	137.0000	277.0000
	Wikidata	156.2500	125.4469	12.0000	271.0000
RightButton	ShExAuthor	0.6667	1.1547	0.0000	2.0000
	YASHE	0.3333	0.5774	0.0000	1.0000
	ShEx2	0.6667	0.5774	0.0000	1.0000
	Wikidata	0.7500	1.5000	0.0000	3.0000
DoubleClicks	ShExAuthor	25.3333	22.2336	12.0000	51.0000
	YASHE	9.0000	10.1489	0.0000	20.0000
	ShEx2	23.0000	22.2711	3.0000	47.0000
	Wikidata	11.7500	13.5247	0.0000	29.0000
MouseWheel	ShExAuthor	1219.3333	1848.1927	12.0000	3347.0000
	YASHE	1272.0000	1661.7009	97.0000	2447.0000
	ShEx2	1220.0000	1042.2951	155.0000	2238.0000
	Wikidata	427.6667	364.2668	8.0000	662.0000
Completness Percentage	ShExAuthor	0.8861	0.2278	0.5444	1.0000
	YASHE	0.8915	0.1879	0.6746	1.0000
	ShEx2	0.5710	0.5087	0.0000	1.0000
	Wikidata	0.7493	0.4480	0.0785	1.0000
Precision	ShExAuthor	0.3357	0.0886	0.2463	0.4239
	YASHE	0.5252	0.1294	0.4460	0.6746
	ShEx2	0.1768	0.1562	0.0000	0.3289
	Wikidata	0.2867	0.2425	0.0700	0.6320

Table 3. Descriptive statistics for task 2.

For the variables Time, Distance, DoubleClicks, MouseWheel no statistically significant differences were found. In the case of the CP variable, despite there being no statistically significant differences F(3,11)=0.846,p=0.497, ω=0.188, YASHE and ShExAuthor show better results, obtaining a mean of 0.8501 and 0.8541 respectively versus 0.6262 in the case of ShEx2 and 0. 5721 in the case of Wikidata. For the Precision variable, no statistically significant differences were found either F(3,11)=0.615, p=0.619, ω=0.144. However, YASHE obtains a higher mean precision (0.4875) with respect to ShExAuthor (0.45879), ShEx2 (0.2583) and Wikidata (0.4641).

In Table 3 we can see the descriptive statistics for the results of task 2. The comparison between the four groups in terms of a One Way ANOVA showed no significant differences for any of the variables related to the use of the mouse and the keyboard. Nor were statistically significant differences found for the variable Time F(3,11)=1.1652, p=0.368, ω=0.241. Despite this, YASHE and ShExAuthor obtain lower values (449.66s and 676.75s respectively) versus ShEx2 (849.25s) and Wikdiata (713.5s). For the CP variable, no statistically significant differences were found F(3,11)=0.590, p=0.634, ω=0.139. However, our tools obtain higher values (0.8915 and 0.8861) compared to ShEx2 and Wikidata (0.571 and 0.7492 respectively). Moreover, they obtain higher Precision(0.5252 and 0.3356) with respect to ShEx2 (0.1767) and Wikdiata (0.2867) even though the differences are still not statistically significant F(3,11)=2.555, p=0.109, ω=0.411.

The results of the qualitative analysis (Figure 3) showed no statistically significant differences for any of the statements in the questionnaire. Despite this, ShExAuthor achieved the highest scores in 7 statements (of a positive nature) out of the 10 where all the tools participated (in one of them tying with Wikidata and in another with YASHE). It also obtained the lowest score (2.34) in S7, where it was stated that the tool gave rise to certain errors and where YASHE obtained the second lowest score (2.75) and ShEx2 obtained the highest score (4.5). On the other hand, YASHE obtained the highest score in 3 statements (of a positive nature) of the 10 common ones (in one of them tying with ShExAuthor) and the highest score (4.5) in S12, in which all the tools participated except Wikidata for not having error messages and where ShExAuthor obtained the lowest score (1.34).

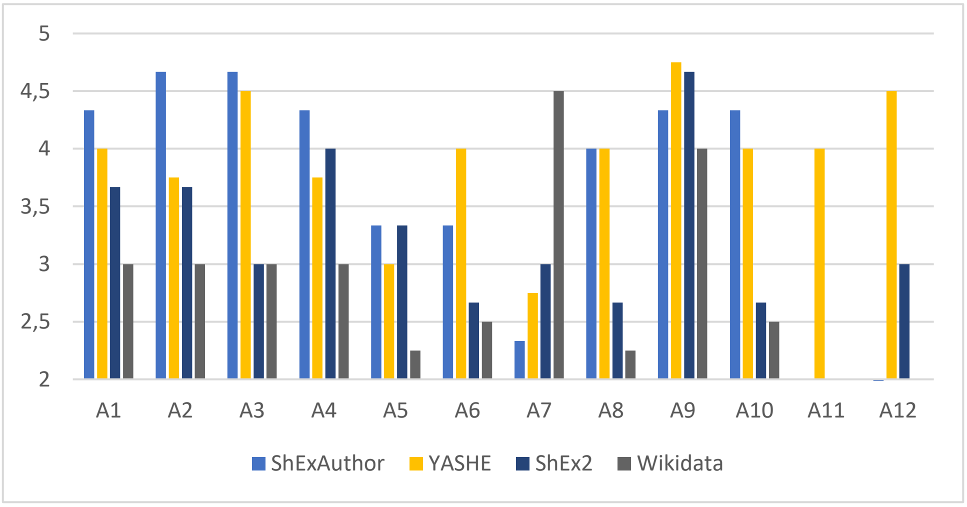


Figure 3. Results for Likert scale questionnaire

6. Discussion

In task 1, the statistically significant differences found in the *KeyStrokes*, *LeftButton* and *RightButton* variables are since the way of working in ShExAuthor. It´s more common to use the mouse buttons to build the shape and where the use of the keyboard is more limited. This difference in the use of the mouse and the keyboard could be related to an increase in the time consumed compared to the other tools. In the case of the number of keystrokes compared to the other tools, it would be expected that there would be no significant difference between ShEx2 and Wikidata since neither of them offers any feature that allows us to save more typing. In the case of YASHE, since it offers autocompletion mechanisms, a lower number of keystrokes could be expected. The fact that no significant differences were obtained with respect to the other two text editors could be since the users did not have enough experience with the tool to make use of the autocomplete features or that the nature of the task did not offer too many scenarios in which to make use of these functionalities. The non-existence of statistically significant differences in the second task for the variables mentioned above could be since it required the representation of a lower number of constraints than the first task and also less restrictive. Thus, even using ShExAuthor did not require a very high number of clicks or keystrokes when using one of the text editors.

The higher Completness Percentage obtained by users who used our tools in both tasks could be related to several reasons. Firstly, by making exclusive use of the ShExAuthor graphical assistant, no syntax errors are made, so if the user establishes all the restrictions required by the task, he/she is assured the maximum score without suffering any type of penalty. Secondly, syntax error detection mechanisms could positively influence the detection and correction of errors by the user, thus decreasing the penalties in obtaining this variable. Wikidata is the only one of the four tools that does not have this functionality. Its score obtained was the lowest with respect to the other tools. On the other hand, although ShEx2 offers a syntactic error detection mechanism, this is not automatic; instead, it is necessary to click on a button on the tool for errors to be reported. This could cause users to make less use of it or forget to use it consistently. YASHE attempts to mitigate this problem by notifying the user of syntactic errors made in real time. In this way, the user might detect a greater number of errors even if he or she is not paying much attention to the task. The usefulness of this functionality is supported by the users in the results obtained in the qualitative analysis, where YASHE obtained a positive score and the highest with respect to the other tools in the statement where this type of mechanism was valued (S12). Finally, ShExAuthor and YASHE are the only tools compared that offer syntax highlighting. This favors the identification of language elements. So, it could contribute to a better identification of errors by the users.

The fact that YASHE obtains a higher accuracy difference with respect to the other tools in the second task compared to the first one, is due to an improvement in the user's times. This improvement could be due to the functionalities offered by YASHE in front of the naturalness of the task itself. As it is focused on the use of Wikidata items, users can make use of the Wikidata autocompletion mechanism. This allows searching directly from the tool itself and avoids having to go to Wikidata itself to search for the desired item. The usefulness and use of this functionality is supported by users in the results obtained in the qualitative analysis, where YASHE obtained a positive score.

One of the reasons why the differences between completness percentage and precision are not significant could be the size of the sample we had to carry out the experiments. We cannot say that YASHE is more usable for non-expert users than the other tools compared. However, we can say that our tool offers a competitive solution to the compared tools. This comes to answer RQ1.

The results obtained for our tools in the qualitative analysis (Figure 3) show a positive response from users. The fact that ShExAuthor obtained the worst score with respect to the other tools in the statement where the usefulness of the error detection mechanisms was evaluated (S12) is because this tool does not allow syntactic errors to be made when only the graphical assistant is used. It is therefore likely that the users who tested this tool did not make use of this functionality at any time. The statement (of a positive nature) where our tools obtained the lowest score was the one that valued the usefulness of the tool at work or at the university (S5). This could be since the users we used for the experiment are not users who habitually use semantic web technologies.

To answer RQ2, we cannot claim that the use of a graphical assistant favors the creation and use of ShEx schemas by non-expert users. We consider that there is a need to further research on this area. However, we have presented a graphical and competitive solution to the compared tools and with which users have had a positive response.

7. Impact

Earlier it was noted that YASHE is intended to be integrated into other applications and projects. The clearest example is ShExAuthor, which integrates YASHE into its system by delegating to it all the text editing functionalities to focus only on the graphical assistant. Currently, the tools that integrate YASHE into your system are the following:

RDFShape
: RDFShape[4] is an RDF playground with a special focus on validation languages such as ShEx and SHACL. It integrates YASHE into its system as the main editor for all the tool's features that involve the use of ShEx.

WikiShape
: WikiShape[16] provides a playground for RDF focused exclusively on Wikidata. Like RDFShape, it integrates YASHE into its system as the main editor.

ShExML
: ShExML[8] is a language based on ShEx for mapping and merging heterogeneous data sources. It´s website[17] integrates YASHE as an editor to visualize the Shapes generated by the language.

Shumlex
: Shumlex[18] is a project that aims to develop a tool to enable integration into ShEx and UML. It has a website where it integrates YASHE for all editing tasks involving Shape Expressions.

[16] https://wikishape.weso.es/
[17] http://shexml.herminiogarcia.com/
[18] https://github.com/fidalgoLXXVI/Shumlex

8. Conclusions and Future Work

A usability study has been carried out for the comparison of four ShEx tools. The text editor we have presented (YASHE), obtained the best results in terms of the relationship time and percentage of task completeness (precision). However, comparisons between tools did not show statistically significant differences for that relationship. This could be due to the sample size we had available to conduct the experiments. Therefore, we cannot claim that YASHE improves usability in non-expert users. On the other hand, we can state that our tool, in addition to introducing new functionalities with respect to the existing ones, offers a competitive solution to the tools compared.

The first (to our knowledge) graphical assistant for shapes creation in ShEx has been introduced. Although this tool did not obtain statistically significant differences with respect to the others, it was the tool that obtained the highest completeness percentage between the two tasks performed in the experiments. In addition, it was the tool that was best received by the users in the survey. We can affirm that the use of a graphical assistant for shapes creation in ShEx offers a competitive solution to the traditional text editors that have been compared.

As future work, an experiment with a larger number of users could be carried out to see if the differences become statistically significant. On the other hand, a larger number of tools could be tested, as well as different user profiles to see if there are differences between them.

References

[1] Roberts, T. L., & Moran, T. P. (1983). The evaluation of text editors: methodology and empirical results. Communications of the ACM, 26(4), 265-283
[2] Rietvelda, L., & Hoekstraa, R. YASGUI: How do we Access Linked Data ?
[3] E. Prud'hommeaux, J.E. Labra Gayo and H. Solbrig, Shape expressions: an RDF validation and transformation language, Proceedings of the 10th International Conference on Semantic Systems - SEM '14 (2014). doi:10.1145/2660517.2660523.
[4] Labra Gayo, J. E., Fernández-Álvarez, D., & Garcıa-González, H. (2018). RDFShape: An RDF playground based on Shapes. In Proceedings of ISWC.
[5] Boneva, I., Dusart, J., Alvarez, D. F., & Gayo, J. E. L. (2019, October). Shape designer for ShEx and SHACL constraints. In ISWC 2019-18th International Semantic Web Conference.
[6] Baungard Hansen, J., Beveridge, A., Farmer, R., Gehrmann, L., Gray, A. J. G., Khutan, S., ... & Splendiani, A. (2015, December). Validata: an online tool for testing RDF data conformance. In Proceedings of the 8th semantic web applications and tools for life sciences international conference, Cambridge UK (Vol. 1546, pp. 157-166).
[7] Borenstein, N. S. (1985). The evaluation of text editors: a critical review of the Roberts and Morgan methodology based on new experiments. ACM SIGCHI Bulletin, 16(4), 99-105.
[8] García-González, H., Boneva, I., Staworko, S., Labra-Gayo, J. E., & Lovelle, J. M. C. (2020). ShExML: improving the usability of heterogeneous data mapping languages for first-time users. PeerJ Computer Science, 6, e318.
[9] Likert, R. (1932). A technique for the measurement of attitudes. Archives of Psychology, 22 140, 55.
[10] Rietveld, L., Hoekstra, R., & Schlobach, S. (2014). Feeling the pulse of linked data. In Proceedings of the Knowledge Engineering and Knowledge Mangement Conference.
[11] H. Knublauch, D. Kontokostas, Shapes constraint language (SHACL), W3C 545 recommendation 20 (07) (2017).
[12] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., Dbpedia–a largescale, multilingual knowledge base extracted from wikipedia, Semantic web 530 6 (2) (2015) 167–195.
[13] Denny Vrandečić and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase". In: Communications of the ACM 57.10 (2014), pp. 78–85.

[14] Rietveld, L., & Hoekstra, R. (2017). The YASGUI family of SPARQL clients 1. Semantic Web, 8(3), 373-383.
[15] R. Cyganiak, D.Wood, M.Lanthaler, RDF 1.1 Concepts and Abstract Syntax (W3C Recommendation 25 February 2014).
[16] E. Prud'hommeaux, A. Seaborne, SPARQL Query Language for RDF (W3C Recommendation 15 January 2008).
[17] ECMA-357, ECMAScript for XML (E4X) Specification, 2nd edition, European Computer Manufacturers Association, Geneva, Switzerland, 2005.
[18] Gayo, J. E. L., Prud'Hommeaux, E., Boneva, I., & Kontokostas, D. (2017). Validating RDF data. Synthesis Lectures on Semantic Web: Theory and Technology, 7(1), 1-328.
[19] Patel, K. (2013). Incremental journey for World Wide Web: introduced with Web 1.0 to recent Web 5.0– a survey paper. International Journal of Advanced Research in Computer Science and Software Engineering, 3(10).