VIPR: A Visual Interface Tool for Programming Semantic Web Rules

Kerry-Louise Skillen
De Montfort University, UK
School of Computer Science and Informatics
Leicester, England
kerry-louise.skillen@dmu.ac.uk

Liming Chen
De Montfort University, UK
School of Computer Science and Informatics
Leicester, England
liming.chen@dmu.ac.uk

William Burns
Ulster University, UK
School of Computing and Engineering
Newtownabbey, Northern Ireland
wp.burns@ulster.ac.uk

***Abstract*—Semantic technologies have evolved from the initial purpose of supporting semantic integration and information exchange for the semantic web, towards a generic set of engineering tools for knowledge modelling, representation and inference. However, there is still much work required within the area of Semantic computing and the area highlights a key research challenge involving the complexity in engineering Semantic rules and associated dedicated models. Many existing tools focus on the creation of models, but concentrate on providing support for domain experts, isolating users with no knowledge engineering experience. This paper aims to address this issue by introducing a novel approach to enable the visual creation of Semantic web rules, for use within ontological models and context-aware applications. The developed tool, known as VIPR, aims to provide a user-friendly, interactive approach to aid in the creation of Semantic rules for ontologies. The work describes the design process involved in creating VIPR and presents the results of a comparative user evaluation. The research highlights the extent to which this tool has on improving the usability and intuitiveness of creating rules in an interactive environment and assesses how the tool can improve the learnability level for users with no prior knowledge engineering experience.**

Keywords—semantic technologies, knowledge engineering, ontology, user modelling, rules, programming, interface, HCI, pervasive, SWRL

I. Introduction

With advances in the use of technology for Ambient Assisted Living (AAL), the need for context-aware, personalised services is increasingly prevalent and has been aided through the development of knowledge modelling, representation and inference models in rceent years [1]. The Semantic Web is proposed as an advanced version of the current WWW, offering an enhanced infrastructure that is used to promote knowledge sharing, allowing new research into the areas of mobile computing and pervasive technologies. The use of Semantic technologies have been noted in several research studies, with a large focus on the representation and reasoning of users through profile modelling and context-aware personalisation [2]. For the Semantic Web to function as required, machines must have access and understanding of structured collections of data and models, with the inclusion of rules to allow for reasoning and inference of meaning. Semantic technologies have enabled context-aware applications to link information and allow data exchange across multiple platforms, through the provisioning of formal modelling languages. In particular, knowledge-engineered models (such as the use of ontological models) have been used to support the specification of domain conceptualisations for the purpose of facilitating information sharing, representation of user needs and profiles [1] and knowledge inference [3] in context-aware applications.

This paper presents an interactive web-based tool (VIPR) to allow the visual programming of Semantic rules for ontological user models. In particular, focus is on developing ontology-based rules using the Semantic Web Rule Language (SWRL) [4] and how this can enhance a user's experience in visually developing personalised rule sets for their own ontologies. The remainder of the paper is structured as follows. Section II presents related work within the field of Semantic rule creation, highlighting the use of rule representation languages in user modelling and provides an overview of existing tools that facilitate the management of SWRL rules. Section III introduces the system architecture of VIPR and presents the design methodology adopted to implement the tool. Following this, Section IV presents the results of a comparative user evaluation with 10 computer science researchers on the use of VIPR compared to an industry standard tool for generating SWRL rules. Subsequently, findings are summarised and conclusions are drawn within Section V.

II. Related Work

Currently, there is much work still required within the area of Semantic computing, in order to realise the widespread uptake of Semantic technologies [2], with a key challenge focusing on the complexity involved to achieve this. Research within the area of knowledge representation and modelling has sparked developments in specialised modelling languages. SWRL and RuleML[5] have developed as two popular Semantic rule representation languages for knowledge engineers. A rule can be described as a conditional instruction, which tells a user "*what*" must happen to achieve a desired outcome. RuleML is the predecessor of SWRL and is recognised as a standardised representation language for Semantic rules. The primary purpose of rule representation via these languages is to simplify the process of translating rules into a format that is machine-readable. Semantic rules are typically used by reasoning engines (such as *Jess* [6] and *Pellet* [7]) and as a result need to be converted into XML. Syntactically, SWRL is an extension to OWL with a new conditional format. Its primary purpose is to enable the integration of rules within ontology models, thus extending the existing OWL-DL language (a description logic language used for ontologies). Logical rules are increasingly becoming an important factor within the area of context-aware computing. Rules enable applications to be tailored to suit the needs of the end-user. One of the challenges that users within the domain of knowledge engineering face is the complexity involved in creating models that contain rules [180]. While there are many existing rule editors available to aid in rule construction, many are built into other integrated editor environments (IDE) and are still overly complex for the novice user. By novice user, this refers to any person who is not a knowledge engineer (i.e. no experience in the designing, developing or maintaining ontological models), however, may exhibit some knowledge of the domain.

Recent research has sparked the development of rule-based editor environments to promote the self-management of SWRL rules for knowledge-engineered models. Existing IDE's such as Protégé [8] or SWOOP [9] have been developed, as well as graphical-based editors [10][11] for ontology and SWRL rule visualisation. Protégé is regarded as one of the most widely used ontology editors, but a key issue with this environment is the 'information overload' of the tool's built-in functionalities and 'messy' visualisation techniques [11]. The work in [12] describes the development of Axiomé, an open-source rule editor to support the Protégé ontology editor. This tool made use of rule-base graphing, rule paraphrasing and visualisation of SWRL rules to aid understanding of how SWRL rules are built. Axiomé is developed as a plug-in for Protégé and therefore the user interface theme matches that of the ontology editor. While this could be beneficial to users that are familiar with Protégé, the complexity involved in learning the environment and functionalities of the tool may discourage novice users. One of the key challenges in understanding SWRL is in the initial interpretation of complex rules and learning of the surrounding environment, particularly for novice users.

The Ontology Rule Editor (ORE) [13] is an open-source Java application which is used to manage SWRL rules within ontology models. The tool presents a graphical interface to allow users to create and test Semantic rules. ORE is built for advanced end-users and as a result, the tool limits its audience and fails to provide support for the novice user. In contrast to this, the work in [14] presents a Java Rule Editor (JRE) that aims to help non-specialist users to understand SWRL rules via techniques such as rule visualisation. The work presents a rule management tool, which deconstructs the SWRL rule into its antecedent and consequent allowing the user to add rule variables for each. Table 1 highlights some comparative rule tools available and their existing limitations within their respective domain. Many existing methods rely on the user manually creating the rules, which can be a complex and time-consuming task as the rule set increases in size [14]. Existing rule editors aim to provide interfaces to allow users to visually edit, manage ontology models and/or rules, but are domain specific or present a high level of complexity and assume basic knowledge/understanding of underlying concepts.

TABLE I. TABLE TO OUTLINE VARIOUS COMPARATIVE SWRL RULE MANAGEMENT TOOLS CURRENTLY AVAILABLE, WITH LIMITATIONS PRESENTED

Tool	**Limitation**
JRE (Java Rule Editor) [14]	Portability, accessibility for all users. No categorisation of rules.
ORE (Ontology Rule Editor) [13]	Complex interface, tool is targeted towards advanced users and computer scientists only. Lack of intuitive interface design.
SWOOP [9]	Lack of focus on SWRL rule creation, editing and management. Users must have prior knowledge engineering experience.
Protégé [8]	Both complexity and learnability levels are high. Users must have knowledge engineering experience.
Axiomé [12]	Not independent of a reasoning engine. No support for the translation of SWRL rules into human readable syntax or format.
HomeCI [15]	Only supports rules within a specific domain, it can be difficult to learn the rule syntax and construction.
SWRL Tab (plug-in for Protégé)	As with Protégé, the tab only acts as a visual plug-in but no guidance is given, no visual aids. Users must have some prior knowledge of ontology engineering and SWRL.

Beyond semantic modelling, existing tools such as the HomeCI [15] tool have focused on the development of a visual rule programming environments for specific domains. HomeCI is a visual rule editor that enables the creation of machine-readable rules for use in smart environments. The tool presented an intuitive user interface that can be used within the healthcare domain and aims to provide an interactive system that involves both the patient carer and the patient in creating personalised rules for home monitoring. Similar to this, the work presented within this paper focuses on the visual programming of Semantic rules for use in ontology models, which could be applied to user profiling models within healthcare. The Visual Interface for Programming Semantic Rules (VIPR) supports the interactive creation of Semantic rules using an intuitive design, where the user can easily deconstruct the components of the rule into manageable sections and therefore understand what rule they are creating and why.

III. Design & Implementation

A. *System Architecture*

As presented in Figure 1, VIPR is implemented in a three-tiered architecture, consisting of a bottom *Data* layer, a middle *Logic/Rule Serialisation* layer, and a top *Presentation/User Interface* layer.

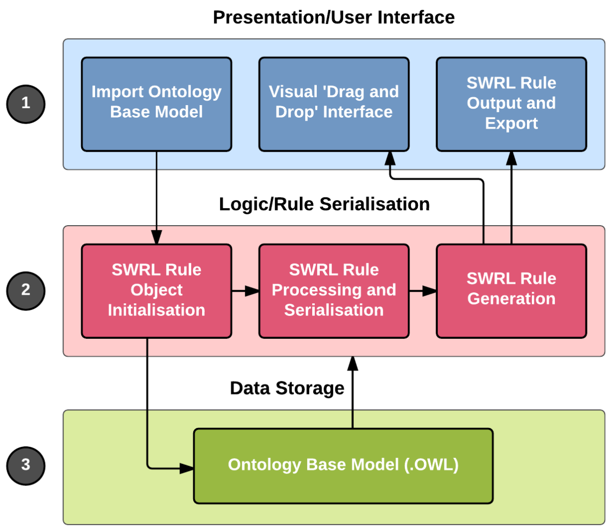

Presentation/User Interface
1
Import Ontology Base Model
Visual 'Drag and Drop' Interface
SWRL Rule Output and Export
Logic/Rule Serialisation
2
SWRL Rule Object Initialisation
SWRL Rule Processing and Serialisation
SWRL Rule Generation
Data Storage
3
Ontology Base Model (.OWL)

Fig. 1. Tiered system architecture for the VIPR system consisting of three separate layers, (1) the data layer where all data is stored, (2) the rule processing, generation and serialisation layer and (3) the top user interface layer.

The *Presentation* layer facilitates the VIPR interface, which contains five components: (1) the ontology URL import, where users load in a URL to initialise the ontology, (2) the rule object initialisation, where class and property concepts are created from the content derived within the loaded ontology, (3) the drag-and-drop visual editor, where the user visually interacts with concepts by dragging them into allocated drop areas for rule creation, (4) the SWRL rule output, which presents a visual output of the rule created on screen and (5) the SWRL rule export, allowing the newly created rule to be appended to the ontology. The *Logic Rule/Serialisation* layer contains the components that enable the core functionality of VIPR. These include the generation of OWL/XML output, the serialisation of JSON/XML and rule translation from JSON-OWL/XML. ***(1)*** The bottom *Data layer* of VIPR is where the ontology model is stored and loaded in via a URL into the web interface.

B. SWRL Rule Syntax

One of the disadvantages of ontological modelling is the model's inability to express complex rules. Within the domain of personalisation, ontologies cannot naturally express all human attributes, needs, wants or characteristics, due to the complex nature and population of people. As a result, this work has implemented the use of designed semantic rules to facilitate the personalisation of user services through SWRL rule sets. Semantic reasoning engines (used to reason about rules) are used to infer logical consequences from a set of asserted facts (i.e. axioms within an ontology model). The use of rules is a powerful way to represent additional concepts that cannot naturally be inferred using OWL or RDF. Ontology languages are unable to express complex formations, such as satisfying more than one condition at the same time.

The SWRL rule language specifies a rule with the following notation as shown in Figure 2. The SWRL format is similar to a simple Horn-like rule structure that is built upon the knowledge base of OWL to increase expressivity in ontology models. Generally, a rule defines a cause-effect relation among a collection of entities that are specific to a domain of knowledge [1]. An OWL ontology written in the abstract SWRL syntax is built using a defined sequence of axioms and facts. An axiom can be defined as 'starting point' of reasoning. Axioms can be class atoms, sub-class atoms or property atoms within an ontology model. SWRL rules (known as rule axioms) are used as an extension to the existing class or property axioms within ontologies, and are used to extend their current reasoning capabilities.

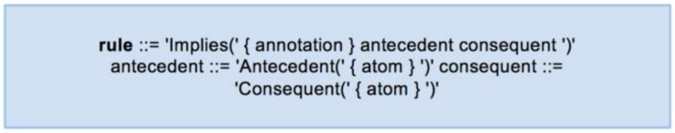

rule ::= 'Implies(' { annotation } antecedent consequent ')'
antecedent ::= 'Antecedent(' { atom } ')' consequent ::=
'Consequent(' { atom } ')'

Fig. 2. The SWRL rule structure, where a consequent rule atom within the rule construct must follow every antecedent rule atom.

Each rule is made up of two parts, the body (known as the *antecedent* of the rule) and the head (known as the *consequent* of the rule. In the domain of context-aware user personalisation, the antecedent of each SWRL rule can be used to represent a conjunction (or union) of user preferences. For example, in the following rule shown in Figure 3, the first line constrains any individuals from the *UserProfile* class that have a health condition (via the *hasHealthCondition* property). In this example, a variable name can be assigned to the *UserProfile* class. If the variable *'?up'* is assigned to the class *UserProfile*. In this rule, the user has a sight impairment resulting in them being *'Blind'*, and the individual name is assigned to the *hasHealthCondition* property. In the rule consequent, it specifies that if the user meets all these aforementioned constraints, then the *HelpDelivery* class will be affected. The *Audio* class within the *HelpDelivery* class will have two major property changes. The *MediaType* will be set to *Audio* as default and the *MediaVolume* will be set to *Vol_High*. The *hasMediaType* property is linked within an ontology model to the *HelpDelivery* class (connected using an object property).

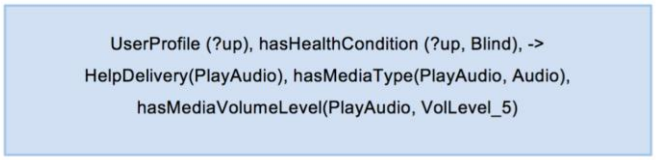

UserProfile (?up), hasHealthCondition (?up, Blind), ->
HelpDelivery(PlayAudio), hasMediaType(PlayAudio, Audio),
hasMediaVolumeLevel(PlayAudio, VolLevel_5)

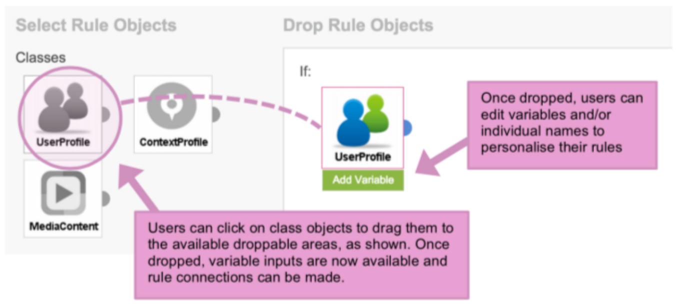
Fig. 3. As illustrated, the SWRL rule shows if a user profile has a health condition of blindness, then various media must be in audio format and played loudly.

The *hasMediaVolumeLevel* is also linked to this class, but its range is set to *Vol_High* and its domain is set to *PlayAudio*. This SWRL rule is quite specific in that individuals from the model are used to restrict the rule. Generic rules can be created in a similar way, which can be used across several application domains. Such rules can be described using variable states (e.g. ?hd for HelpDelivery or ?u for User).

~~B.~~C. *Application Design & Implementation*

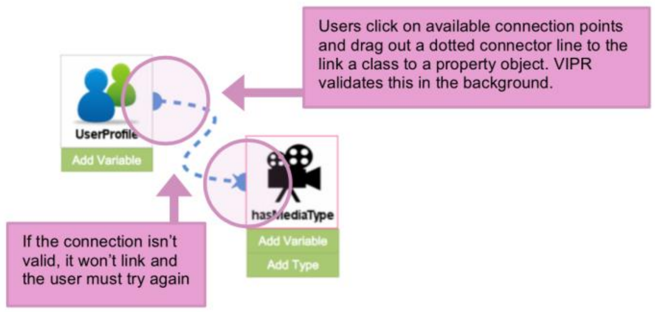
Yan and Guo [16] identified a set of universal web usability guidelines that should be adhered to when designing interactive user interfaces. In particular, emphasis is placed on the idea of a 'user-centric' design approach to human computer interaction (HCI). This approach focuses on the system development being driven by the requirements of the user and not what is technically required. Such an approach was followed through the design and development of VIPR. The user interface design of VIPR was heavily influenced by current web standards on usability and HCI. The general layout consists of the main header bar containing the options to load in the URL for the user's ontology file, a *How to Use* help guide for using the tool and a button to allow the user to log out of VIPR.

The left side of the interface contains several *Rule Objects* that are associated with the ontology that has been loaded by the user. These objects are split into two types, namely the *Class* objects and the *Property* objects. Further to this the *Property* objects are split again into sub-categories of *Data Property* objects and *Object Property* objects. Each type of object is placed under the relevant heading and can be moved to any droppable area. VIPR uses a 'drag and drop' interface where users can select each of the *Rule Objects* and move them across the screen for rule creation. As shown in Figure 4, when an object is selected, dragged and dropped, the icon appears with a 'variables' box underneath, displayed within a green background. The Rule Objects are dragged and then dropped onto each of two specified droppable areas. Once dropped, the user may then begin to create the structure of the rule. VIPR makes use of both jQuery and jsPlumb [17] to provide an interactive view of the rule creation process. Every rule object will have a blue connection point attached to it in the shape of a circle. The user is able to connect two rule objects by clicking on a connection point and pulling a dashed connector line from one rule object to another.


Select Rule Objects
Drop Rule Objects
Classes
UserProfile
ContextProfile
MediaContent
If:
UserProfile
Add Variable
Once dropped, users can edit variables and/or individual names to personalise their rules
Users can click on class objects to drag them to the available droppable areas, as shown. Once dropped, variable inputs are now available and rule connections can be made.

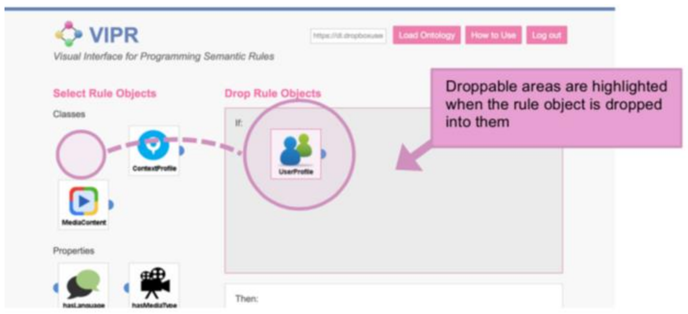
~~Fig. 2.~~Fig. 4. The Rule Objects within VIPR are selectable and draggable, and highlight when selected.

If a connection is not made between the objects, the rule will not output as valid SWRL. VIPR allows the user to design and personalise their SWRL rules via the use of free-text variable and individual names (where the user can enter any name they wish). Upon the initial load of the ontology file the user can view all available rule objects, however, they are unable to specify any variables until they drop that particular object into a droppable area. Once dropped, input boxes appear directly underneath the rule object. For class objects the user can add a variable only, however, for the property objects the user may specify a variable name and an individual name. Both the use of connections and variable naming is presented in Figure 5.


Users click on available connection points and drag out a dotted connector line to the link a class to a property object. VIPR validates this in the background.
UserProfile
Add Variable
hasMediaType
Add Variable
Add Type
If the connection isn't valid, it won't link and the user must try again

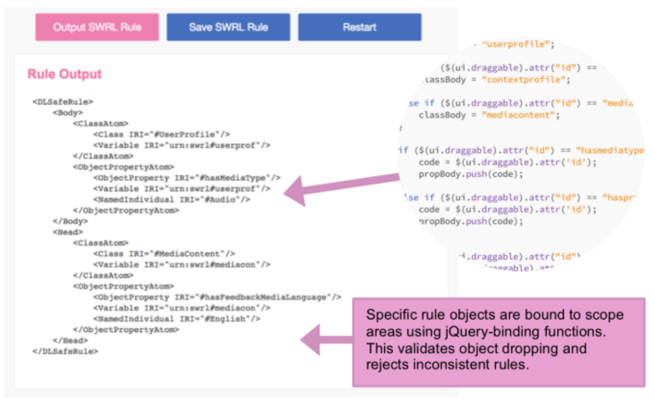
~~Fig. 3.~~Fig. 5. The VIPR interface, allowing users to connect rule objects via jQuery and also specify a variable and/or individual name to the objects.

Located to the right of the *Rule Objects* are the *Droppable Rule* areas. These are split into two distinct areas with the purpose of separating the SWRL rule into manageable sections. The top droppable area is labeled IF and the bottom area is labeled THEN. This technique is used to separate the structure of the SWRL rule for the user to clearly identify where each rule object should be placed. SWRL rules are structured with an antecedent (body) and a consequent (head), similar to that of conditional programming logic via *'IF-THEN'* rules. The rule reads as: *IF* something is specified *THEN* the outcome will be the result of what is specified. The two droppable areas are colour coded and when a rule object is dragged on top of each, the background colour changes to signify that it is a valid droppable zone, as shown in Figure 6.



~~Fig. 4.~~Fig. 6. The droppable areas as displayed within VIPR, with background highlighting to show the valid dropping areas for various rule objects.

The final component of VIPR is regarded as the most significant area for the output of the newly created SWRL rules. The *Rule Output* area is where the SWRL rule is displayed in OWL/XML format and can then be downloaded to the new ontology file, as shown in Figures 7 and 8. This output is displayed after the user selects 'Output SWRL Rule' and allows the user to view what they have created. The user may cancel and begin a new rule clicking on the ‘Restart’ button, where the interface reloads all of the ontology objects.

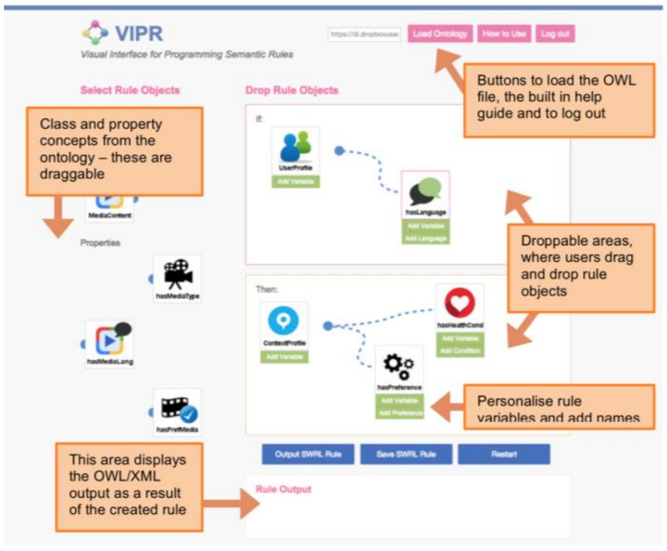

Output SWRL Rule
Save SWRL Rule
Restart
Rule Output
Specific rule objects are bound to scope areas using jQuery-binding functions. This validates object dropping and rejects inconsistent rules.

```
<DLSafeRule>
    <Body>
        <ClassAtom>
            <Class IRI="#UserProfile"/>
            <Variable IRI="urn:swrl#userprof"/>
        </ClassAtom>
        <ObjectPropertyAtom>
            <ObjectProperty IRI="#hasMediaType"/>
            <Variable IRI="urn:swrl#userprof"/>
            <NamedIndividual IRI="#Audio"/>
        </ObjectPropertyAtom>
    </Body>
    <Head>
        <ClassAtom>
            <Class IRI="#MediaContent"/>
            <Variable IRI="urn:swrl#mediacon"/>
        </ClassAtom>
        <ObjectPropertyAtom>
            <ObjectProperty IRI="#hasFeedbackMediaLanguage"/>
            <Variable IRI="urn:swrl#mediacon"/>
            <NamedIndividual IRI="#English"/>
        </ObjectPropertyAtom>
    </Head>
</DLSafeRule>
```

```
 "userprofile";
 ($(ui.draggable).attr("id") ==
 lassBody = "contextprofile";

se if ($(ui.draggable).attr("id") == "media
 classBody = "mediacontent";

if ($(ui.draggable).attr("id") == "hasmediatype
 code = $(ui.draggable).attr('id');
 propBody.push(code);

se if ($(ui.draggable).attr("id") == "haspr
 code = $(ui.draggable).attr('id');
 ropBody.push(code);

 i.draggable).attr("id")
```

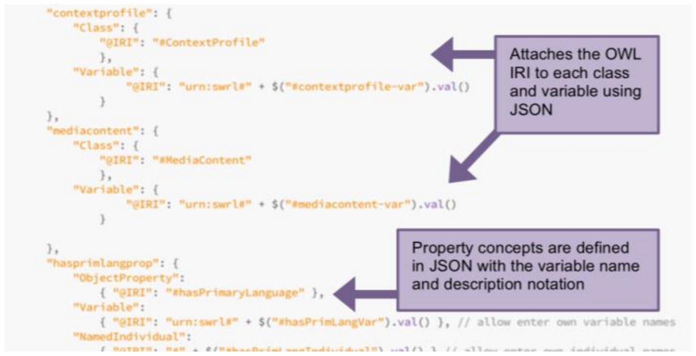
~~Fig. 5.~~Fig. 7. The Rule Output area showing the result of a SWRL rule created using VIPR.

VIPR allows for easy visualisation of ontology classes and sub-properties, via the use of draggable objects on screen (each containing images which relate to the ontology concept). The user drags specific objects to two designated ‘drop’ areas. These areas have been purposely designed to differentiate the two conditions of any SWRL rule. Semantic rule design involves the use of knowledge modelling. This requires the analysis of the domain of knowledge (for example, the ontology model would focus on a specific domain); the identification of application processed and related entities. The purpose is to establish core relationships between specific entities, which are aided via the use of rules.


VIPR
Visual Interface for Programming Semantic Rules
Load Ontology
How to Use
Log out
Select Rule Objects
Drop Rule Objects
Buttons to load the OWL file, the built in help guide and to log out
Class and property concepts from the ontology – these are draggable
MediaContent
Properties
If:
UserProfile
hasLanguage
Droppable areas, where users drag and drop rule objects
hasMediaType
Then:
hasHealthCond
ContextProfile
hasMediaLang
hasPreference
Personalise rule variables and add names
hasPrefMedia
This area displays the OWL/XML output as a result of the created rule
Output SWRL Rule
Save SWRL Rule
Restart
Rule Output

~~Fig. 6.~~Fig. 8. The implemented visual interface for VIPR as displayed within the Google Chrome browser.

Upon dragging and dropping the rule objects to the two designated areas, the user can then view the current output of that rule by clicking on *‘Output Rule’*. The interface displays the output of the rule in an ontology-compatible XML format, known as OWL/RDF. On the client-side, VIPR is written using jQuery (based on JavaScript) and JavaScript Object Notation (JSON). JSON is a lightweight data interchange format that uses human-readable text to transfer data objects that are made up of simple attribute-value pairs. It is used as an alternative language to XML to transmit data between the back-end web server and VIPR. As shown in Figure 9, JSON is made up of attribute-value pairs consisting of the ontology class or property name and the variable(s) linked to this. This JSON list will allow for easy manipulation and generation of the required XML output. In order to generate the output of the rule as XML an external JavaScript library called JSON2XML [18] was used. This library was used to convert the JSON rule snippets to the XML equivalent and jQuery was used to ‘bind’ the variable and individual names to specific class or property concepts.


Attaches the OWL IRI to each class and variable using JSON
Property concepts are defined in JSON with the variable name and description notation

```
"contextprofile": {
    "Class": {
        "@IRI": "#ContextProfile"
        },
    "Variable": {
            "@IRI": "urn:swrl#" + $("#contextprofile-var").val()
        }
},
"mediacontent": {
    "Class": {
        "@IRI": "#MediaContent"
        },
    "Variable": {
            "@IRI": "urn:swrl#" + $("#mediacontent-var").val()
        }

},
"hasprimlangprop": {
    "ObjectProperty":
        { "@IRI": "#hasPrimaryLanguage" },
    "Variable":
        { "@IRI": "urn:swrl#" + $("#hasPrimLangVar").val() }, // allow enter own variable names
    "NamedIndividual":
```

~~Fig. 7.~~Fig. 9. Snippet of JSON notation used to create the ontology classes and properties. Each JSON object contains the class or property name and associated variables.

Once the JSON format is converted to XML, JavaScript is then used to ‘attach’ specific snippets of XML code to each draggable rule object. Each rule object that is shown has a class or property name and an associated variable or individual name with it. Generally, when creating a rule the user is able to specify the variable or individual names linked to each ontology entity. VIPR allows the user to edit the individual or variable name associated with specific rule properties. This ‘binding’ of variable and individual names is completed using jQuery to attach specific naming to different parts of the rule. For example, the user may drop both the class ‘*UserProfile’* and a property ‘*hasLanguage’* in the rule body. They can then click on the ‘*hasLanguage’* property object and type in their own individual name. In this case, the individual name would usually be a language such as ‘English’ or ‘Spanish.’ Once they have entered this, the name is then appended to the JSON in the server-side and serialised into XML for the rule output. This gives the user an added form of flexibility when constructing the rule.

~~C.~~D. *Development Tools*

VIPR was developed as a responsive, cross-browser compatible and interactive web interface. The tool was built upon the basis of existing Web 2.0 technologies such as HTML5 and CSS3 and implemented using jQuery. JavaScript, JSON and XML were combined to facilitate the development of the client-side logic for rule creation. A key feature of VIPR focuses on the use of visual representation of Semantic rules, where the user can connect different ontology concepts and personalise the contents, then convert this layout to display the underlying XML through simple button selections. The tool also allow the user to save the new SWRL rule into their initial *.OWL* file, where the new SWRL rule is appended to the .OWL file that has been initially loaded in and downloaded to user's computer.

IV. User Evaluations

The developed tool VIPR was involved in a comparative user evaluation involving 10 participants, recruited from within the Smart Environments Research Group (SERG) at the Ulster University, and consisted of a combination of computer science PhD students and researchers. Existing research highlights issues with Semantic rule creation and how difficult it can be to learn the processes involved in ontological modelling via rules and reasoning. The outcome of the evaluation was therefore to identify the strength of VIPR regarding three aspects of the *Learnability* of the SWRL rule creation process, the *Usability* of the visual programming interface and the *Efficiency* of SWRL rule creation.

A. Participants

VIPR was evaluated on a total of 10 user participants (male *n=4* and female *n=6*). Nielson suggested that conducting a usability study on very large numbers of users proved wasteful of both resources and time [19]. He stated that the majority (over 2/3) of all usability issues are identified when conducted on just 2-3 users. Participants were both male and female and were aged between 25 and 42 years old (*mean age=28.8*). VIPR was evaluated and benchmarked against the use of Protégé for SWRL rule creation. The evaluation was therefore split into two divisions, focusing on evaluating these factors firstly within VIPR and subsequently within Protégé. Participants were asked to use VIPR first as it contained a help guide (for using VIPR and Protégé). Participants were categorised into two groups. Groups 1 were classed as *novice* users and Group 2 as *experienced* users, although both groups of users were researchers within the field of computing. Participants initially self-rated their experience levels through standardised questionnaire feedback, where Group 1 (*n=5*) exhibited little to no prior experience within the field of knowledge engineering, while participants in Group 2 (*n=5*) exhibited an average to high level of experience.

B. Tasks

At the beginning of each evaluation session, participants were given a participant instruction sheet containing the tasks to be completed using VIPR and Protégé. Such tasks included the completion of a SWRL rule and interpretation of a SWRL rule using both tools. Participants were also given an information sheet detailing the research area and evaluation and the overall purpose of the study, a consent form and a sheet to write down their interpretation of a SWRL rule from within Protégé. Participants were required to work through the instructional sheet and create a Semantic rule based on a sample user profile ontology model [1]. This model provided a pre-defined narrative within the area of user profile personalisation of smart-phone assistive services. The participants were electronically guided through the web interface and were asked to perform the two tasks that were provided on the instruction sheet. They were required to complete the tasks using VIPR, and then replicate the same tasks using Protégé. The tasks focused on creating and interpreting Semantic rules that adjust the media format and deliver mode smart-phone services, based on a user's profile. The evaluation focused on collecting the following parameters:

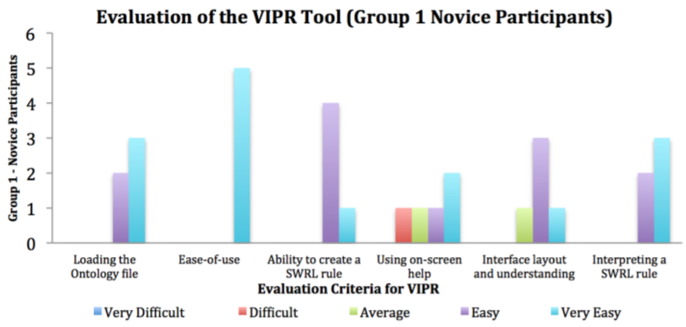
1) The **average time** taken to create and export an SWRL rule using VIPR and Protégé.
2) The **average time** taken to interpret an SWRL rule within VIPR and Protégé.
3) A qualitative measure on the benefits of using a visual programming interface.
4) Any limitations of VIPR as a result of usage.
5) Does VIPR reduce the overall learning curve associated with the management of SWRL rules, when compared to an industry standard package?

Upon completion of the tasks, all participants were asked to complete an online questionnaire that identified their perceptions, thoughts and feedback on the usability, efficiency and learnability of VIPR. Questionnaire design was heavily based on the IBM Computer Usability Satisfaction Questionnaire [20].

C. Results & Discussion

All of the participants completed the tasks correctly, with varying completion times. Participants within both user groups took *less* time to complete the tasks using VIPR compared to Protégé. The longest total time recorded using VIPR was 11 minutes and 59 seconds, with the longest time using Protégé totaling over 14 minutes. Participants took on average a total of 4 minutes and 30 seconds to create a valid SWRL rule within VIPR. Comparatively, participants took on average 2 minutes and 54 seconds longer to complete the same task within Protégé. For the interpretation task, participants took on average a total of 2 minutes and 26 seconds to write down their understanding of a rule within VIPR. In comparison, they took more time to interpret the *same* rule within Protégé, with an average total of 2 minutes and 29 seconds. Within the novice and experienced user groups, all participants were asked to complete the tasks of creating a rule and interpreting a rule using both VIPR and Protégé. The aim was to identify how easily they could do these and what problems occurred.

Evaluating VIPR with Novice Group 1: All participants were asked if they were able to easily load the ontology into VIPR and drag the rule objects, with all responding *'Yes.'* Similarly, Group 1 participants were asked to evaluate the overall ease of use of the VIPR tool, with all responding *as 'Very easy.'* Furthermore, all participants within Group 1 were able to successfully use VIPR to create a valid SWRL rule, despite having little to no prior experience in the area. On a scale of 1- 5, with 1 being *'Very Difficult'* and 5 being *'Very Easy'*, 4 out of 5 participants in Group 1 scored the ability of creating a rule as *'4'*, with 1 participant scoring a *'5.'* All participants made use of the help resources within VIPR and watched the available screencast to learn how to create rules using the tool. The researcher observed each participant as they watched the help screencast. Subsequently, all participants found that the help resources within VIPR were sufficient. Every participant liked VIPR's interface design, particularly commenting on the use of colour and imagery to aid understanding in differentiating rule classes, properties and drop areas. A summary of this is presented in Figure 10.



~~Fig. 8.~~Fig. 10. Overview chart displaying a summary of the VIPR questionnaire responses from participants within the novice user group.

Evaluating Protégé with Novice Group 1: In general, participants felt that the lack of help within Protégé effected their ability to create a valid rule and felt intimidated by the environment overall. However, novice participants were able to identify and interpret rules in Protégé in a similar time to VIPR, only interpreting on average 3 seconds quicker within VIPR. 4 out of 5 participants found the process of creating the same SWRL rule much *more* difficult using Protégé than within VIPR. As shown in Figure 11, all participants were able to successfully load the ontology file into Protégé, but results show that all participants from Group 1 disliked creating and interpreting rules using Protégé, particularly when compared to VIPR. They commented that Protégé's editor was simple, however, ineffective, with no help guides or error control included. Participants struggled with the SWRL syntax and expected help from built-in error messages. When evaluating the use of Protégé to create rules, participants from Group 1 found the process difficult.

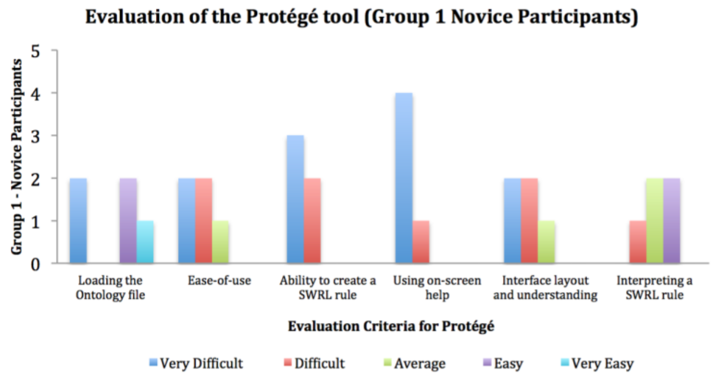
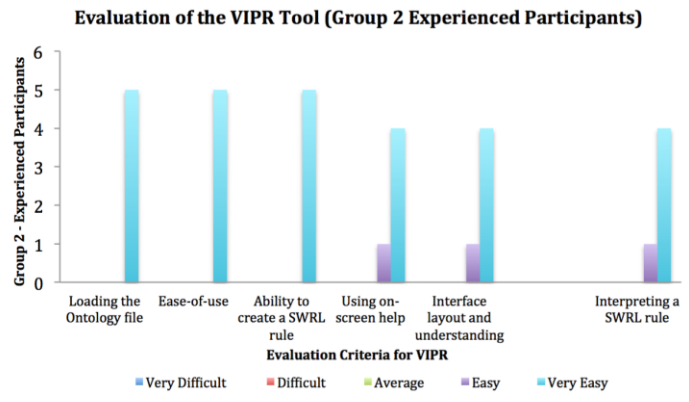


Fig. 11. Overview chart displaying a summary of the Protege evaluation responses from participants within the novice user group.

Participants were able to successfully load the ontology and found this aspect simple, with just 1 participant rating this as *'Difficult.'* The majority of participants (4 out of 5) rated the overall ease-of-use of the tool as either a '4' or a '5' (*'Difficult'* or *'Very Difficult'*). Interestingly, 2 participants noted that they were only able to understand the SWRL syntax within Protégé as they had used VIPR's help guide previously.

Evaluating VIPR with Experienced Group 2: All participants within Group 2 were able to create a SWRL rule in VIPR and interpret the rule shown on-screen with no issues. All participants were able to successfully create the rule, output the OWL/XML on-screen and then download the new rule and open it within Protégé. Due to their previous experience, these tasks took minimal effort and time, and all participants liked the process involved using VIPR's "drag-and-drop" interface. Experienced participants felt the user interface and usability of VIPR were excellent, and commented on the intuitiveness of the entire system. Such results are presented within Figure 12.



~~Fig. 9.~~Fig. 12. Overview chart detailing the questionnaire responses from Group 2 participants when evaluating the VIPR tool.

All participants were able to easily load the ontology model into VIPR and create an SWRL rule with no issues. Participants also rated the ease-of-use of the VIPR tool as a '5' as they found it very easy and straightforward. All participants were also able to successfully interpret a rule as shown in VIPR, with no reported issues. Overall, experienced participants completed both tasks of creating and interpreting SWRL rules in less time than when using Protégé.

Evaluating Protégé with Experienced Group 2: In general, experienced participants found creating a rule within Protégé more difficult and less efficient than when they used VIPR. 3 out of 5 participants rated the process of creating a SWRL rule as a '1' or '2'. The remaining participants found the process quite easy, however, noted that they could not remember the full syntax required, rating it a '3' and '4.' When asked if they liked using Protégé to create rules, 4 out of 5 participants from Group 2 responded with "No." They commented that the help was "insufficient and annoying" (Participant 1, Group 2) and the error messages were *"confusing and complicated"* (Participant 2, Group 2). Figure 13 presents a summary of this feedback.

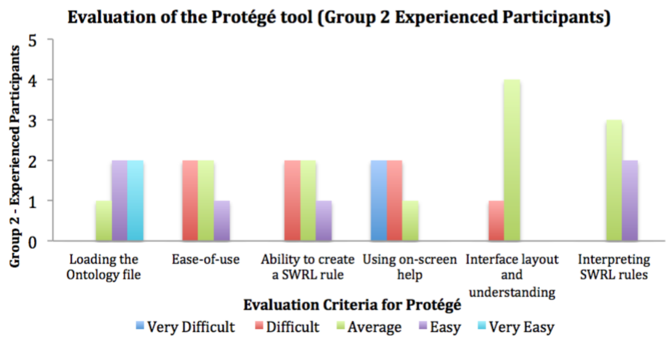


Fig. 13. Overview chart detailing the questionnaire responses from Group 2 participants when evaluating Protégé

As a whole, participants within Group 2 disliked the help built into Protégé and stated that the main problem was the lack of tutorials for writing the syntax. 4 out of 5 participants rated Protégé's help as a '1' or a '2' as a result. All Group 2 participants were able to interpret a rule in Protégé with no problems. 3 out of 5 rated this process as a '3' and the remaining participants scored it a '4.' All participants from Group 2 interpreted the rule quickly and with little help.

D. *Findings*

Overall findings of the work support the hypothesis that there is a clear need for a visually user-friendly tool that is dedicated to allowing users of differing knowledge levels to create their own Semantic rules and import these seamlessly into ontology models. The initial evaluation results demonstrate that both novice and experienced users can undertake the process of creating the rules at a faster rate in VIPR than the universal standard, Protégé. The help guide included within VIPR was a significant advantage over Protégé's built-in documentation and enabled users of any level to create and interpret rules at a faster rate using VIPR. Despite this, the majority of participants felt the inclusion of a more comprehensive help guide would be useful. The use of help points at each step of the rule creation process would distinguish this tool from existing works. The 'drag-and- drop' style visual programming interface was favoured over Protégé's basic rule editor.

V. CONCLUSIONS

This paper presented the design, development and evaluation of a web-based visual interactive interface for the programming of SWRL rules. The need for a dedicated SWRL creator tool was highlighted and the shortcomings in related work was discussed. The overall aim was to evaluate and identify the impact that the tool had on improving the overall usability, efficiency and learnability levels of a user when creating Semantic rules. The evaluation of VIPR was carried out on both a novice and experienced groups of users. The aim was to produce a comparative evaluative study, where VIPR challenged the use of Protégé as a tool to simplify the development of Semantic rules for use in ontological models.

As a result of initial user evaluations, VIPR was found to be intuitive, easy to use and produced valid SWRL for every participant. Errors were highlighted on-screen and participants were able to easily identify the errors and fix these with no issues. However, VIPR currently only supports the loading of one base ontology model, where further work to the tool could include the incorporation of dynamic content over time.

VI. REFERENCES

[1] K.-L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, W. Burns, and I. Solheim, "Ontological user modelling and semantic rule-based reasoning for personalisation of Help-On-Demand services in pervasive environments," *Futur. Gener. Comput. Syst.*, vol. 34, pp. 97–109, 2014.

[2] L. Ding, P. Kolari, Z. Ding, S. Avancha, T. Finin, and A. Joshi, "Using ontologies in the semantic web: A survey," in *Ontologies*, Springer US, 2007, pp. 79–113.

[3] A. K. Kalou, T. Pomonis, D. Koutsomitropoulos, and T. Papatheodorou, "Intelligent Book Mashup: Using Semantic Web Ontologies and Rules for User Personalisation," *2010 IEEE Fourth Int. Conf. Semant. Comput.*, pp. 536–541, Sep. 2010.

[4] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Memb. Submiss.*, vol. 21, p. 79, 2004.

[5] "RuleML: Rule Mark-Up Language." [Online]. Available: http://wiki.ruleml.org/index.php/RuleML_Home. [Accessed: 01-Apr-2016].

[6] E. J. Friedman-Hill, "JESS: The Java Expert System Shell," *http://herzberg.ca.sandia.gov/jess*, 2014. [Online]. Available: http://www.jessrules.com/jess/docs/45/.

[7] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semant. Sci. Serv. agents World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[8] S. C. for B. I. Research, "Protégé - A free, open-source ontology editor and framework for building intelligent systems." [Online]. Available: http://protege.stanford.edu/. [Accessed: 27-Mar-2016].

[9] C. P. University of Maryland, "SWOOP - Semantic Web Ontology Editor," 2014. [Online]. Available: https://code.google.com/p/swoop/. [Accessed: 16-Apr-2016].

[10] J. Bak, M. Nowak, and I. Engineering, "Graph-based Editor for SWRL Rule Based Motivation Graph-based Editor," 2013.

[11] N. Catenazzi, L. Sommaruga, and R. Mazza, "User-Friendly Ontology Editing and Visualization Tools: The OWLeasyViz Approach," *2009 13th Int. Conf. Inf. Vis.*, pp. 283–288, Jul. 2009.

[12] S. Hassanpour, M. O'Connor, and A. Das, "Axiomé: A Tool for the Elicitation and Management of SWRL Rules.," in *OWLED*, 2009.

[13] "ORE: Ontology Rule Editor." [Online]. Available: http://sourceforge.net/projects/ore/. [Accessed: 18-Jan-2016].

[14] R. Guozheng, F. Zhiyong, and J. Biao, "JRE: A Visual Semantic Rule Management Tool," *Softw. Eng. Knowl. Eng. Theory Pract.*, vol. 162, pp. 651–660, 2012.

[15] M. Beattie, J. Hallberg, C. Nugent, K. Synnes, I. Cleland, and S. Lee, “A collaborative patient-carer interface for the self- management of dementia,” in *International Conference on Smart Homes and Health Informatocs*, 2014.

[16] P. Yan and J. Guo, “The research of Web usability design,” *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, vol. 4. Ieee, pp. 480–483, Feb-2010.

[17] “jQuery - Cross Platform JavaScript Library.” [Online]. Available: http://en.wikipedia.org/wiki/JQuery. [Accessed: 23-Jan-2015].

[18] “XML to JSON: A Converter.” [Online]. Available: http://www.thomasfrank.se/xml_to_json.html. [Accessed: 24-Jan-2015].

[19] J. Nielson, “Why You Only Need to Test with 5 Users,” *Nielson Norman Group*, 2015. [Online]. Available: http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. [Accessed: 22-Feb-2016].

[20] J. R. Lewis, “IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use.,” *Int. J. Human-Computer Interact. 7.1*, vol. 1, pp. 57–78, 1995.

VII.