Using Drool Rule-Platform for the Optical CAD Web-Application Development

Dmitry Mouromtsev, Maxim Kolchin
Saint-Petersburg National University of Information Technologies
Mechanics and Optics
Saint-Petersburg, Russia
{d.muromtsev, kolchinmax}@gmail.com

Abstract

This paper describes a development of rule-based web application to the problem of starting point selection in optical design. The system architecture is shortly discussed. A formalization of the optical system representation based on formal syntax is given. Three examples of rules in natural language and Drools Rule Language syntax illustrate the work of the solution.

Index terms: web application, rule-based system, Drools, optical design

I. INTRODUCTION

The design of optical systems (OS) is a sequential process including stages of the analysis of a technical specification, a structural synthesis, a parametric optimization etc. Currently there exists a lot of computer-aided design systems (CAD) that can effectively solve the problem of an optical scheme optimization (also known as parametric synthesis), calculating the exact parameters of an optical system. But for successful calculation of an optical system's parameters the initial optical scheme or "starting point" for the whole design process must be specified correctly. Until now this problem is solved by an optical designer relying on his or her experience. At the same time, a special kind of software - expert systems (ES) are used successfully to solve design problems for several decades [1].

The selection of suitable starting point is one of the most important steps in the optical design and it in more than 80% determines the success of the whole project. A formal rule-based design method of structural synthesis shortly described in [2] was proposed by prof. M. Russinov and developed by I. Livshits. This makes it possible to create an expert system for the optical design.

II. MAIN PART

Due to the progress of expert system's technology, there are plenty of rule platforms, which can be reused in a new development of expert system. When the complexity of the development is high, the right choice of effective tools acquires importance. Therefore we decided to choose the most effective platform that will suit to our purposes.

A. Selection of a rule platform

Before we started searching a suitable rule-platform we had defined general requirements of the platform to be complying with. The requirements are the following:

- advanced knowledge representation language (KRL);
- the support of traditional programming languages (PL);

- the support of the forward chaining;
- the presence of tools for knowledge engineering and knowledge base development, such as a rule repository, a rule editor etc.;
- an Open Source license and an active developer's community.

We reviewed a lot of platforms, but only the following comply with the requirements: OpenRules [6], Drools [7], OpenL Tablets [8] and CLIPS [9]. The results of the review are presented in Table I.

TABLE I
THE COMPARISON OF THE RULE PLATFORMS

Name	KRL	The support of traditional PL	The type of inference	The presence of tools for knowledge engineering	License
OpenRules	Tables (Excel, Rational DB), XML, POJOs	Java	the activation of rules depends on its order in the repository	rule editor, rule repository, version control	GPLv2, Commercial
OpenL Tablets	Tables (Excel, Word), Decision trees	Java	the activation of rules depends on its order in the repository	rule editor, rule repository, version control, decision tables	LGPL
Drools	DRL (Drools Rule Language), RuleML, CLIPS	Java	the forward and backward	rule editor, rule repository, version control, decision tables	ASLv2, Commercial
CLIPS	CLIPS	Python, C/C++, Java	only the forward	rule editor, version control	Public Domain

As it's seen from the table, the all platforms have many ways to write rules (tables, own language etc.), tools for knowledge engineering and Open Source licenses, but only Drools Platform has both an advanced knowledge representation language and an powerful inference engine that can perform the forward and backward inference, whereby more complex expert system can be built.

B. Overview of the system architecture

By selecting the most appropriate rule platform, we have got the all needed tools for the development. Drools platform provides a powerful tool for knowledge engineering - Guvnor, containing different kinds of editors (such as a guided editor, a decision table editor), an integration testing framework, a storage of knowledge bases supporting versioning of rules, models etc. and a web-service giving an access to knowledge bases.

On Fig. 1 a conceptual diagram of the system architecture is present. As it's seen from the diagram a user can work with the system either using a plain browser or via some external software communicating with the system through a web-service. All versions of knowledge bases are kept in the repository that is why the system has a separate module called knowledge agent connecting to the repository at the bootstrap stage of the system work and fetching the necessary version of the knowledge base to be used by inference engine.

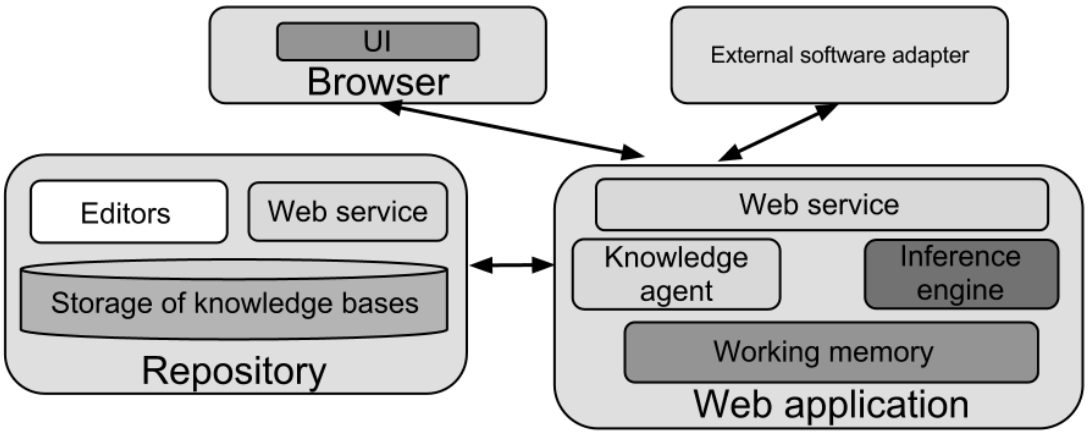

Fig. 1. Conceptual diagram of the system architecture

C. Optical design data formalization

A simple and effective formalization of the optical system representation based on formal syntax allows to build a description from the basic notions to the complex description, formulate consistent rules for the knowledge base, as well as expand the range of the problems of optical design, by introducing elements of the new features avoiding conflicts with an already used in the system.

The initial data for the starting point selection process are the technical requirements measured in physical units: Aperture speed, Angular field, Focal length, Angular units, Spectral range, Image quality, Wave units, Back focal distance, Entrance pupil position. However, for the rules of the knowledge base system it is more convenient to use generalized qualitative characteristics “0”, “1” or “2” and truth-value variables, inferred by the system. The conversion of the technical requirements to generalized values is present in Table II.

TABLE II
CONVERSION OF THE TECHNICAL REQUIREMENTS TO GENERALIZED VALUES (GVal)

Class	Characteristic	GVal “0”	GVal “1”	GVal “2”
J	Aperture speed	OS is not fast < 2.8	OS is fast 2.8 ... 1.5	OS is super fast > 1.5
W	Angular field	OS with small angular field < 15∘	OS with average angular field 15∘ ... 60∘	wide angular OS > 60∘
F	Focal length	short focal length OS < 50 mm	average focal length OS 50 … 100 mm	long focal length OS > 100 mm
L	Spectral range	monochromatic OS < 5 nm	ordinary polychromatic 5nm ... 200nm	super polychromatic correction > 200 nm
Q	Image quality	“geometrical“ image quality > 5	“intermediate” image quality 2 … 5	“diffraction” image quality < 2
S	Back focal distance	OS with short back focal length s < ½f	OS with average back focal length 1/2f ≤ s ≤ f	OS with long back focal length s > f
D	Entrance pupil position	entrance pupil located inside OS	removed back entrance pupil	removed forward entrance pupil

A superposition notation is used for a formal description of the optical system scheme as a sequence of elements «OE1 + OE2 + OE3 + ...». Each element is represented by a type symbol and a list of surfaces. A generic notation of an element is nexsxs, where n, x - numbers, e - type of element, s - type of surfaces.

A type of an element can be “B” - Basic Elements, “C” - Correction Elements, “T” - Fast Elements or “Y” - Wide-angular Elements. A type of surfaces is selected from”O” - a flat surface,”P” - a surface concentrated the center of the input pupil,”A” - an aplanatic surface,”F” - a surface concentrated the focus,”I” - a surface located near the focal zone,”V” - a variable surface. The variety of number of surfaces depends on a construction of the element. For example for lenses the list includes two records. But for other types of optical elements the list may consist of one record, if the element is a mirror, or 3 or more for elements with a complex structure.

Finally a description of the optical surface consists of specifying the zone where it is located and the type of surface. Location of the surfaces can be “1“- before aperture, “2”- near aperture, “3” - near image.

Formal representation syntax for optical system scheme in Backus–Naur Form is give below:

```
OS ::= <Y>” + “<B>” + “<T>” + “<C>
<Y> ::= ”” | <Y_expr> | <Y_expr><C>
<Y_expr> ::= <numbers_elements>”Y”<surface_list>
<B> ::= <B_expr> | <B_expr><C>
<B_expr> ::= <numbers_elements>”B”<surface_list>
<T> ::= ”” | <T_expr> | <T_expr><C>
<T_expr> ::= <numbers_elements>”T”<surface_list>
<C> ::= ”” | <C_expr>
<C_expr> ::= <numbers_elements>”C”<surface_list>
<numbers_elements> ::= ”” | ”2” | ”3” | ”4” | ”5”
<surface_list> ::= <surface><surface>
<surface> ::= <zone><surface_type>
<zone> ::= ”1” | ”2” | ”3”
<surface_type> ::= ”O” | ”P” | ”A” | ”F” | ”I” | ”V”
```

An example of a starting point description according with proposed syntax is “Y3P3P + B3A3P + C3I3I”.

D. Examples of rules

The rules have the following format: IF <conditions> THEN <actions>. Below are examples of three rules from the knowledge base. The first one is an example of conversion of the technical requirement of an aperture speed to a generalized value “OS is fast”.

The rule in the natural language:

```
IF
    the aperture speed of the OS between 2.8 and 1.5
THEN
    classify the OS as a fast and set the J class with the value
equals “1”
```

And the same rule in the DRL (Drools Rule Language) that is used in our knowledge base as a knowledge representation language:

```
declare Classification
    J : int
    W : int
```

```
    ...
end

declare Requirements
    apertureSpeed : float
focalDistance : float
    ...
end

rule "Classify as fast"
    when
      $class := Classification()
      Requirements( apertureSpeed < 2.8 && > 1.5 )
    then
      modify( $class ){
            setJ( 1 )
      }
end
```

Here we declare two new types of data. The first with name "Requirements" represents the technical requirements; the second "Classification" represents the classification of the OS and a rule with name "Classify as fast". The Requirements type has fields apertureSpeed, focalDistance etc. And the Classification type has fields J, W etc. representing the classes.

The second is an example of reasoning on a meta level as we exclude from the starting point inference all rules concerning fast optical elements. The rule in the natural language:

```
IF
    the optical system is not fast
THEN
    a fast element isn't needed at the system
```

And the same meta-rule in the DRL:

```
declare SystemConditions
    isFastNeeded : boolean
    isWideAngleNeeded : boolean
    ...
end

rule "No fast"
    when
          Classification( J == 1 )
          $conds := SystemConditions()
    then
          modify( $conds ) {
                setIsFastNeeded( false )
          }
end
```

Here we declare one more type of data. It's "SystemConditions" representing inferred conditions for another rules and a rule with name "No fast". The SystemConditions type has fields isFastNeeded, isWideAngleNeeded etc. The "No fast" rule will be activated when the working memory contains an instance of the Classification type with the value of the field J equals '1' and then the value of the isFastNeeded field of an SystemConditions' instance will be set to 'false'.

The last rule is a plain rule as we select only the ‘B3A3P’ base element.
The rule in natural language:

```
IF
    the entrance pupil of an optical system is removed forward
THEN
    exclude the all base elements with exception of B3A3P element
```

The same but only in the DRL syntax is:

```
declare OpticalElement
    type : String
    surfaces: String
    ...
end

rule "The B3A3P element"
    when
          Classification( D == 2 )
          $elements := ArrayList() from collect( OpticalElement()  )
    then
          modify( $elements ) {
                clear()
                add( new OpticalElement( type == "B",
                                         surfaces == "3A3P" ))
    }
end
```

At this example we declare a new type “OpticalElement” representing an optical element and the “The B3A3P element” rule that will be fired when an instance of Classification type with the value of the D field equals to `2`.

III. Conclusion

In the optical design process there is a specific stage - a starting point selection or structural synthesis of an optical system scheme. There are no appropriate software solutions for this stage. But it is well known that a rule based approach has proved its effectiveness for such a task. Authors describe an approach to development of rule-based web application for the optical design. As a platform Drools is used. The selection of the platform is based on rule-engines’ review. Also a knowledge formalization of optical design data is made. This formalization make it possible to formulate rues in user-friendly manner. But in fact the proposed approach is opened for extension and rules for other types of optical systems can be added into the knowledge base. We also plan to develop a special user interface to be used in designers’ training and integrate our system with optical optimization software.

References

[1] T. Gavrilova, D. Muromtsev Intelligent technologies in management. St. Petersburg: Izd. GSOM, p. 488, 2008.
[2] M. Russinov Technical Optics. Mashinostroenie. Leningrad, p. 488, 1979 (in russian).
[3] Livshits, A. Salnikov, I. Bronchtein, U. Cho, Database of optical elements for lens CAD, 5th International Conference on Optics-Photonics Design & Fabrication, pp. 31-32, 2006.
[4] L. Livshits, I. G. Bronchtein, V. N. Vasiliev, Information technologies in CAD system for lens design, Proc. SPIE.
[5] 7506, 2009. doi:10.1117/12.837544.
[6] Irina L. Livshits, Vladimir N. Vasiliev, Optical systems classification as an instrument for getting knowledge from optical expert, 7th Int'l Conf. on Optics-Photonics Design & Fabrication, pp. 13-14, 2010.
[7] Robert R. Shannon, "The Art and Science of Optical Design", Cambridge University Press 1997, ISBN 0-521-58868-5.

[8] Open Rules (http://openrules.com).
[9] Drools (http://www.jboss.org/drools).
[10] OpenL Tablets (http://openl-tablets.sourceforge.net).
[11] CLIPS (http://clipsrules.sourceforge.net/).