KNOWLEDGE & INFORMATION MANAGEMENT THROUGH LIFE

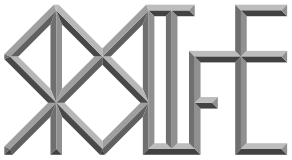
# RRoRIfE User Guide and Registry Reference

Alex Ball

kim12too003ab04.pdf

Access Level: 1

Issue Date: 20 February 2011



## Foreword

This document incorporates and supersedes two previous documents, namely *XML Formats For Engineering-Related Representation Information* (kim12too001ab) and *Ontology of Properties of Engineering-Related File Formats and Software* (kim12too002ab).

## CONTENTS


1 Introduction . . . 1
2 Installation . . . 2
3 Operation . . . 2
3.1 File Format Explorer . . . 2
3.2 File Format RepInfo Editor . . . 2
3.3 File Format Picker . . . 3
3.4 Converter Explorer . . . 3
3.5 Converter RepInfo Editor . . . 4
3.6 Conversion Path Finder . . . 6
3.7 Conversion Path Wizard . . . 6
4 RELAX NG compact schema language . . . 7
5 Processing software: known issues . . . 8
5.1 Commentary . . . 10
5.2 Sample document . . . 12
6 File format characteristics . . . 13
6.1 Commentary . . . 14
6.2 Sample document . . . 14
7 Ontology of properties . . . 15
7.1 Methodology . . . 15
7.2 Interpretation of the hierarchies . . . 15
7.3 Metadata . . . 16
7.4 Construction . . . 16
7.5 Compression and identification . . . 17
7.6 2D geometry . . . 17
7.7 3D geometry . . . 19
8 High-level source code structure . . . 20
9 Acknowledgements . . . 22
10 Licensing . . . 23
References . . . 23


## 1 Introduction

One of the deliverables from the KIM Project is a demonstrator that uses representation information about file formats and software tools to make recommendations about preservation strategies. This tool, the Registry/Repository of Representation Information for Engineering (RRoRIfE), has been written in Java 1.6 and should therefore run on any platform with a recent Java runtime environment installed. Details on how to install and operate RRoRIfE may be found in sections 2 and 3 respectively.

One of the prime uses of the representation information generated by the KIM Project is to judge whether certain characteristics of a piece of data can survive processing by a certain piece

of software; this is a function of the capabilities of both the software and the formats involved. At the time of the project, there did not exist a file format or metadata schema particularly suited to supporting this kind of application, and so two XML schemata were devised to fill this need. The schema for describing the capabilities of and known issues with conversion software is described in section 5. The schema for describing the encoding capabilities of file formats is described in section 6. Both schemata have been defined using the RELAX NG compact schema language, to which a quick introduction may be found at section 4

Supporting both of these schemata is an ontology enumerating the properties by which the various engineering-related file formats and software tools can be judged. In this context, a property is intended to mean a method of processing or recording a semantic expression, such as a way of expressing a NURBS surfaces. Section 7 sets out properties identified so far as being important for interpreting engineering documents.

## 2 Installation

RRoRIfE is supplied as an executable JAR file. It is recommended that it is placed in its own directory, as it stores its repository of XML files in a directory called '`rep`' located in the same directory as the JAR file. It does not have configuration options, and therefore is not dependent on any external files or settings other than the XML files that make up the repository.

## 3 Operation

The RRoRIfE interface consists of two panels. The left-hand panel consists of buttons for selecting the content of the right-hand panel. Initially, the Conversion Path Wizard is displayed. The different screens are described in the following sections.

### *3.1 File Format Explorer*

This screen allows you to look up the stored information about file formats represented in the registry. Selecting from one of these formats brings up the RRoRIfE ID of the format – its identifier within the system – and a set of properties drawn from the ontology (see section 7). These are properties about which the format's level of support is known.

Each property is accompanied by a graphic that indicates the level of support.

- ✔ This format is fully capable of expressing this property.
- (✔) This format is partially capable of expressing this property. The manner in which it falls short of full support will be explained in a comment immediately following the property.
- ✘ This format is incapable of expressing this property.

If a property is not listed, it is not known the extent to which the format supports that property.

### *3.2 File Format RepInfo Editor*

This screen allows you to add new formats to the registry, and to modify the records of the formats already registered. From the drop-down selection box, select the format you wish to edit, or 'New format' if you would like to add a format to the registry. This will bring up the editing controls on the screen

The first two text input boxes allow you to edit the name and version number of the format, respectively. The third input box displays the RRoRIfE ID assigned to the format. It is not currently possible to edit the RRoRIfE ID, though you can, of course, alter the final identifying number manually in the saved XML file.[1]

The next set of inputs allow you to add a comment relating to the format as a whole. The left-hand drop-down selection box allows you to select the comment to edit; the highest number in the list allows you to enter a new comment. The other drop-down selection box allows you to indicate which language the comment is or will be written in. It currently defaults to UK English.[2] There is also a text box for entering the comment itself. Modifications to comments will persist if you move between them. If you add a new comment, simply changing to a different comment will store the changes and provide you with a space for another new comment; after entering a first comment, though, you will need to save the whole record (see below) before the interface will give you space for a second comment. To delete a comment, press the 'Delete' button.

Below the comments on formats is a tabbed tree interface containing all the properties in the ontology grouped in their respective themes. This allows you to specify the extent to which the format supports each property: select a property, and use the radio buttons beneath to specify full support, partial support or no support. The 'Clear' button unsets the support level for the currently selected property, to enable mistaken selections to be removed. Comments can also be added to each property, in order to explain subtleties in the format's support for the property; a comment should always be present if partial support is indicated, but comments are optional otherwise. The interface works in the same way as for comments on the whole format.

Finally, the 'Save' button at the bottom commits the changes to the registry. To discard changes, pick another format (or the 'Please select' message) in the drop-down selection box at the top of the page.

### *3.3 File Format Picker*

This screen allows you to find formats that meet a set of criteria you specify. The screen presents you with a tabbed tree interface containing all the properties in the ontology grouped in their respective themes. On selecting a property, you are able to specify if it is *required* (the format must fully support it), *desired* (the format must support it at least partially), *irrelevant*, or *forbidden* (the format must not support it). If you find you have expressed a preference for the wrong property, you can undo your current selection by pressing 'clear'.

Once you have finished setting your criteria, press 'Search'. RRoRIfE will then search through its database of formats and pick out the ones that match your criteria. The matching formats will be presented beneath the 'Search' button in exactly the same way as on the File Format Explorer screen (see subsection 3.1).

### *3.4 Converter Explorer*

This screen allows you to explore the file format conversion capabilities of the software represented in the registry. To begin, select one of the pieces of software from the drop-down

1. As noted in subsection 6.1, the intention is for RRoRIfE to use IDs already in use by larger registries, but this is not currently supported. Note also that the RRoRIfE ID reflects the location of the respective XML file in the repository folder.
2. It is intended that this default should be modifiable through a configuration option. The same option could be used to switch the language of the interface, or a separate option might be used in its own right or as an override to this behaviour. Needless to say, this is not currently supported.

selection box. This will bring up the RRoRIfE ID of the converter software – its identifier within the system – and two further selection boxes, one for formats that the software can read ('input'), and one for formats that can be saved by the software ('output'). On selecting a pair of formats, the screen displays the RRoRIfE IDs of the two selected formats; if by selecting different options the outcome is different with respect to properties in the ontology, these are presented in different blocks with the options shown that make the difference, but otherwise, only one block is shown. Each block contains a list of properties from the ontology, accompanied by a graphic that signifies the extent to which that property survives the conversion from the input format to the output format.

- ✔ This property is preserved well enough by the conversion to support the reversal of the conversion by other software.
- (✓) This property is not preserved well enough by the conversion to support the reversal of the conversion by other software in all cases, but is more likely to be preserved than it is to be lost or degraded. The way in which the preservation is less than perfect will be explained by a comment immediately following the property.
- ◎ With a file of realistic complexity, this property is at least as likely to be degraded or corrupted as it is to be preserved by the conversion.
- ✘ This property never survives the conversion.

Where properties do not survive intact, it may be that they are degraded rather than corrupted. The message 'degradation is configurable' indicates that the way in which a property is converted to a non-equivalent expression may be configured by the user. An example of this would be a converter that allowed one to specify the extent to which a mesh of tessellating triangles deviates from the NURBS surface it approximates. The message 'degradation is fixed' indicates that the property is always converted to a non-equivalent expression in the same, predictable way. The message 'degradation is unpredictable' means that the degree of success with which the converter manages to re-present the property in a non-equivalent manner varies from case to case.

The 'Reset' button allows you to clear all your selections and start again from scratch.

### *3.5 Converter RepInfo Editor*

This screen allows you to add new conversion software to the registry, and to modify the records of the software already registered. From the drop-down selection box, select the converter you wish to edit, or 'New converter' if you would like to add a converter to the registry. This will bring up the editing controls on the screen

The first two text input boxes allow you to edit the name and version number of the converter, respectively. The third input box displays the RRoRIfE ID assigned to the converter. It is not currently possible to edit the RRoRIfE ID, though you can, of course, alter the final identifying number manually in the saved XML file.[3]

The next two selection boxes allow you to specify a conversion that the converter can perform; the interface only allows you to select registered file formats, so if the converter can handle additional input or output formats, be sure to add them using the File Format RepInfo Editor before using this screen.

3. As noted in subsection 5.1, the intention is for RRoRIfE to use IDs already in use by larger registries, but this is not currently supported. Note also that the RRoRIfE ID reflects the location of the respective XML file in the repository folder.

Having selected a valid pair of formats, you will then be able to add or edit general comments on how the converter handles conversions from one specified format to the other. The left-hand drop-down selection box allows you to select the comment to edit; the highest number in the list allows you to enter a new comment. The other drop-down selection box allows you to indicate which language the comment is or will be written in. It currently defaults to UK English.[4] There is also a text box for entering the comment itself. Modifications to comments will persist if you move between them. If you add a new comment, simply changing to a different comment will store the changes and provide you with a space for another new comment; after entering a first comment, though, you will need to save the whole record (see below) before the interface will give you space for a second comment. To delete a comment, press the 'Delete' button.

The remainder of the form fields on this screen relate to option groups (or 'executions' in the language of the XML schema). The idea behind option groups is to allow you to record how selecting different options affects the extent to which properties are preserved through a conversion. For example, a converter might offer the option to either keep or discard construction histories when converting between two formats; these two cases would be recorded as two different option groups.

Below the comments on conversions is a drop-down selection box, allowing you to choose which option group to edit. Below this is line for entering the relevant options that make up the option group. If the converter does not offer any options for the conversion, or if the options offered do not affect the handling of any of the properties in the ontology, there is no need to enter options. Otherwise, enter only the options that make a difference. (In the above example, it might be an option labelled 'Keep construction history' that is either selected or deselected.) The selection box on this line allows you to enter a new option or edit an existing one, in much the same way as the comments on conversions. To the right of this box are two text fields; the leftmost one is for specifying the name of the option as it appears in the interface or on the command line, while the rightmost one is for specifying the value supplied for that option. In the case of a tickbox, give '0' for unticked and '1' for ticked; otherwise, enter the value exactly as it appears in the interface or as supplied by a user. A 'Delete' button is supplied for removing an option from the option group. Deleting the last option in an option group will delete the entire option group, unless it is the only option group left.

Below this is another line for entering comments, this time relating to the specific option group. The interface works in the same way as for comments on conversions.

Beneath the comments on options is a tabbed tree interface containing all the properties in the ontology grouped in their respective themes. This allows you to specify the extent to which the conversion preserves each property: select a property, and use the radio buttons beneath to specify a good level of preservation, a fair level, a poor level, or total loss. Where preservation is fair or poor, a second row of buttons allows you to indicate how the property degrades, if it does degrade rather than disappear entirely.[5] Both rows of radio buttons have a 'Clear' button that unsets the selections for that row, to enable mistaken selections to be removed. Changing the preservation level to 'good' or 'none' automatically clears any selection in the row relating to degradation. Comments can also be added to each property, in order to explain subtleties in how the converter handles the property; a comment should always be present if fair or poor

4. It is intended that this default should be modifiable through a configuration option. The same option could be used to switch the language of the interface, or a separate option might be used in its own right or as an override to this behaviour. Needless to say, this is not currently supported.
5. By 'degradation' is meant transformation to a non-equivalent expression; for example, a NURBS surface may degrade to a triangular mesh with approximately the same shape. No degradation occurs with good preservation, as the latter implies reversibility. Similarly, where total loss is indicated, no degradation can occur.

preservation is indicated, but comments are optional otherwise. The interface works in the same way as for comments on conversions.

Finally, the 'Save' button at the bottom commits the changes to the registry. To discard changes, pick another converter (or the 'Please select' message) in the drop-down selection box at the top of the page.

### *3.6 Conversion Path Finder*

This screen allows you to find all the possible migration pathways between two formats, based on the information in the registry. First select the two formats: the format a file is currently in, and the format you would like it to be in. Next, select the maximum number of conversions you would allow to get between the two formats; the interface defaults to four conversions, but you may choose fewer if you prefer. To initiate the search, press 'search'.

The sample representation information provided with RRoRIfE concentrates on converting from Solid Edge to PDF, so this combination will give plenty of results.

Once the search is completed, the screen will tell you how many paths it has found. If it found more than one, it will allow you to choose between them using a drop-down selection box. Each path consists of one or more steps. Each step consists of a conversion tool, and input format and an output format; for each of these, a 'details' button is provided, which switches to either the Converter Explorer or the File Format Explorer screen, open at the respective conversion or file format. To return to the results of this screen, press the 'Conversion Path Finder' button on the left-hand side again.

The 'Legend' button clears the results screen and restores the informative message shown initially.

### *3.7 Conversion Path Wizard*

This screen allows you to specify the format a file is currently in, and the properties of the file you would like to preserve or discard. In return, it provides you with suitable migration pathways in which the end result is a file in a different format, but satisfying your criteria. The method of entering these criteria is the same as in the File Format Picker (see subsection 3.3), with the addition of a 'weight' specification relating to the relative importance of the property in question. This weighting affects the order in which the results are presented. As with the Conversion Path Finder, the maximum number of conversions between the original format and the eventual format is initially limited to four, but may be reduced if desired. To initiate the search, press 'search'.

As with the Conversion Path Finder, once the search is completed, the screen will tell you how many paths it has found. If it found more than one, it will allow you to choose between them using a drop-down selection box. Each path consists of one or more steps. Each step consists of a conversion tool, and input format and an output format; for each of these, a 'details' button is provided, which switches to either the Converter Explorer or the File Format Explorer screen, open at the respective conversion or file format.

The 'Legend' button clears the results screen and restores the informative message shown initally.

## 4 RELAX NG compact schema language

RELAX NG is a simple and flexible language for declaring XML schemata, with both an XML and compact mode of expression. The schemata declared in this document use the compact syntax [3, 4]. The remainder of this section is a brief introduction to the language, but is by no means comprehensive.

The basic structure of a RELAX NG compact schema is a hierarchy of elements and attributes, declared using the respective keywords '`element`' and '`attribute`', followed by the name of the element or attribute, then a pair of curly braces containing the allowed contents. Thus:

```
1 element myElement { text }
```

defines a schema in which the element '`myElement`' must occur exactly once, and contain (nothing but) text. The keywords '`text`' and '`empty`' are pre-defined with the obvious meanings; other datatypes may be specified by using the `xsd` namespace to access the datatypes defined by the XML Schema language [1], e.g. '`xsd:gYearMonth`', or by using a similar technique to reference other datatypes. Strings are indicated using inverted commas; `"..."` and `'...'` can be used for short strings (without line breaks), whereas `"""..."""` and `'''...'''` can be used for strings that include line breaks. Strings can be concatenated using a tilde symbol (~).

The closing brace may be followed by a modifier character that determines how many times the element may occur. The declaration:

```
1 element myElement { text }*
```

indicates that the element '`myElement`' may occur zero times, once or many times. The modifier '`+`' indicates that the element may occur once or many times, and the modifier '`?`' indicates that the element may occur zero times or once. Unmodified, an element must occur exactly once.

Sibling elements are declared using either a comma or an ampersand to separate the declarations. The ampersand (&) symbol indicates that the two elements may occur in any order, whereas the comma symbol indicates that the second element must occur after the first. Elements that are alternatives to one another are separated instead by a pipe character (|). Parentheses should be used to mark the scope of each of these binary operators, except where only one type of operator is used in a run of adjacent elements.

The same symbols are used when declaring multiple attributes except that, since XML does not recognise any significance in the order of attributes, the comma takes on the same meaning as the ampersand.

A single hash symbol (#) indicates that the remainder of the line is a comment. A double hash symbol (##) indicates that the comment is a piece of DTD-style documentation, or more formally, a string that forms the content of a '`documentation`' element from the RELAX NG DTD Compatibility namespace.

A RELAX NG compact schema can also be written in a modular form known as a `grammar` pattern. A `grammar` pattern is so called because it occurs within a '`grammar {...}`' construct, although this delimitation can be omitted when the pattern is introduced at the top level. The `grammar` pattern works by making a number of definitions such as:

```
1 Name = element myElement { text }
```

where `Name` is the name of the definition. A definition such as this allows a single declaration or pattern to be used multiple times within the schema. The top level of the schema must be defined using the name `start`.

Patterns can be imported from external files using ‘`external "filename.rnc"`’ in place of a regular pattern, while whole schemata can be imported using ‘`include "filename.rnc"`’ instead of a definition within a `grammar` pattern. In both cases, the referenced file should itself be a `grammar` pattern.

## 5 Processing software: known issues

The following RELAX NG compact schema defines version 0.3 of the XML format [2, 6] for describing known issues with particular migration processes. Documents conforming to this schema can be entered as describing `Processing Software` in the OAIS section of the Registry, or as a piece of `Documentation` relating to a `Software Component` detailed in the PRONOM section of the Registry.

```
# Version 0.3
default namespace = "http://www.ukoln.ac.uk/projects/grand-challenge/conv-issues.rnc"
start =
  element converter {
    attribute toolname { text }?,
    attribute toolid { xsd:anyURI }?,
    attribute version { text }?,
    element conversion {
      attribute source { xsd:anyURI } &
      attribute destination { xsd:anyURI } &
      element execution {
        element options {
          element option {
            attribute key { text } &
            attribute value { text }
          }+
        }? &
        element features {
          element feature {
            attribute property { external "properties.rnc" },
            attribute preservation { "good" | "fair" | "poor" | "none" }?,
            attribute degradation { "configurable" | "fixed" | "unpredictable" }?,
            Comment*
          }+
        }? &
        Comment*
      }+ &
      Comment*
    }+
  }
Comment =
      element comment {
        attribute xml:lang { xsd:language },
        text
      }
```

The referenced file `properties.rnc` (also version 0.3) is as follows.

```
# Version 0.3
start = Properties
```

```
Properties =
  ( "Feature semantics"
  | "Material metadata"
  | "Geometric dimensioning and tolerancing"
  | "Dimensions"
  | "Assembly node metadata"
  | "Assembly hierarchy"
  | "Constructive solid geometry"
  | "Boundary representation"
  | "Trimmed surface"
  | "Parameterized re-use of instances"
  | "Simple re-use of instances"
  | "Construction history modelling"
  | "Multiple alternative representations"
  | "Multiple levels of detail"
  | "Field-wise compression"
  | "Stream-wise compression"
  | "Whole-file compression"
  | "Streaming"
  | "Identification of subassemblies"
  | "Identification of parts"
  | "Identification of surfaces"
  | "Identification of edges"
  | "Identification of vertices"
  | "Analytic 2D geometry"
  | "Point"
  | "Polyline"
  | "Line"
  | "Conic arc"
  | "Elliptical arc"
  | "Circular arc"
  | "Polygon"
  | "Triangle"
  | "Rectangle"
  | "Square"
  | "Ellipse"
  | "Circle"
  | "Open composite curve"
  | "Closed composite curve"
  | "NURBS curve"
  | "Rational Bézier curve"
  | "Non-rational Bézier curve"
  | "Cubic Bézier curve"
  | "Quadratic Bézier curve"
  | "Analytic 3D geometry"
  | "Point cloud"
  | "Helix"
  | "Plane"
  | "Ellipsoid"
  | "Sphere"
  | "Cylinder"
  | "Cone"
  | "Cuboid"
  | "Cube"
```

```
57    | "Torus"
58    | "Mesh of surface segments"
59    | "Mesh of tessellating triangles"
60    | "Lofted surface"
61    | "Ruled surface"
62    | "Translation surface"
63    | "Normal swept surface"
64    | "Polylinear swept surface"
65    | "Extrusion surface"
66    | "Swung surface"
67    | "Rotation surface"
68    | "NURBS surface"
69    | "Rational Bézier surface"
70    | "Non-rational Bézier surface"
71    )
```

The properties listed above are defined in section 7.

### *5.1 Commentary*

```
4     element converter {
5       attribute toolname { text }?,
6       attribute toolid { xsd:anyURI }?,
7       attribute version { text }?,
30    }
```

The top level element is `converter`, representing the software tool being described. The three optional attributes `toolname`, `toolid` and `version` provide a way of specifying the piece of software being described, independent of the representation information held by the Registry. The `toolid` attribute is meant to point to an existing ID, rather then declare one. Support for this has not yet been added to RRoRIfE; in the meantime, it uses local identifiers of the form '`info:rrorife/sw/`*number*'.

```
8       element conversion {
28        Comment*
29      }+
```

Each `converter` is capable of one or more `conversions`. The `conversions` are identified by the source format and destination format.

A comment or several (see below) may be attached to a conversion in order to remove ambiguity, or to record a subtlety not catered for by the remainder of the schema.

```
9         attribute source { xsd:anyURI } &
10        attribute destination { xsd:anyURI } &
```

These two attributes provide IDs that identify the source and destination formats of a particular migration pathway. Globally unique identifiers such as provided by PRONOM[6] are intended. Support for this has not yet been added to RRoRIfE; in the meantime, it uses local identifiers of the form '`info:rrorife/f/`*number*'. These attributes are required.

```
11        element execution {
26          Comment*
```

6. URL: ⟨`http://www.nationalarchives.gov.uk/pronom/`⟩.

```
27        }+ &
```

The same conversion process may be run with different options, resulting in different preservation and degradation properties (described below). If so, each of these variations is recorded as a separate execution. Note that if there is only one way in which a conversion process may be run, the execution element is still required.

A comment or several (see below) may be attached to an execution in order to remove ambiguity, or to record a subtlety not catered for by the remainder of the schema.

```
12          element options {
13            element option {
14              attribute key { text } &
15              attribute value { text }
16            }+
17          }? &
```

Where a conversion has more than one execution, the executions are distinguished by all and only the conversion options that affect the preservation and degradation properties (described below) of the conversion. In other words, if many executions with different options have the same preservation and degradation properties, they are recorded as a single execution, with any options not held in common omitted. The options themselves are recorded using the options element, which must contain at least one option element. Where a conversion has only one execution, the options element is omitted.

The attributes of the option element should reflect the choices presented to a user on using the tool. In the case of a simple tick-box option, the text of the option should be given as the key and 'Y' (for ticked) or 'N' (for unticked) given for the value. In the case of an option group (usually represented by radio buttons), the name of the option group should be given as the key and the name of the individual option chosen should be given as the value. The key string 'command line' is reserved for command line options, which may be given in bulk as the value. Other key-value options are specified in the expected manner.

If no options are presented to the user, or if none of the options that are available impact on the features described under the following element, the options element should be omitted.

```
18          element features {
19            element feature {
24            }+
25          }? &
```

These elements are for recording how a conversion process copes with certain pertinent aspects of a piece of design documentation. The features element may be omitted if no information is yet known about a process; this is to allow the existence of the conversion path to be registered in advanced of any testing being carried out. Nevertheless, the more feature elements that are given, the more useful the document.

```
20              attribute property { external "properties.rnc" },
```

The property attribute specifies one of an authority list of file format characteristics that a) may or may not be requisite for a given purpose, and b) may or may not be supported by a given file format or format conversion. This list will be one of the main points of development as the scheme approaches release.

```
21              attribute preservation { "good" | "fair" | "poor" | "none" }?,
```

This attribute describes how the property fairs under processing. The value 'good' indicates that it is handled and preserved perfectly, and could therefore (given an equally 'good' return path) survive a round-trip conversion intact. The value 'poor' indicates that with any file of realistic complexity, a degradation or corruption of the property is at least as likely as it surviving intact. The value 'fair' describes an intermediate state between 'good' and 'poor'; it is intended to cover cases where there are known circumstances in which the property does not survive intact, but that this is the exception rather than the rule. Where a value of 'fair' is used, a comment element should be used to explain the meaning. The value 'none' indicates that the property never survives intact.

```
22              attribute degradation { "configurable" | "fixed" | "unpredictable" }?,
```

In some cases a property is degraded rather than corrupted. A prime example of this is where a boundary representation is approximated by tessellating polygons. The value 'configurable' indicates that the user of the tool can configure how the degradation occurs, e.g. by specifying the maximum deviation of a set of straight lines from the curve they approximate. The value 'fixed' indicates that the degradation process cannot be configured, but occurs in a known/documented fashion. The value 'unpredictable' indicates that the process by which the property is degraded is unknown/undocumented, making it hard to predict how any given instance will degrade.

```
23              Comment*
31  Comment =
32        element comment {
33          attribute xml:lang { xsd:language },
34          text
35        }
```

Any ambiguities (such as with the use of the preservation value 'fair') can be cleared up for the human reader in free text, using the comment field. The language in which the comment is written should be specified. Each feature element, may contain zero, one or multiple comment elements. Note that the comment elements attached to conversion or execution elements follow the same format as those attached to feature elements.

### *5.2 Sample document*

Given an identifying URI for this schema of ⟨http://www.ukoln.ac.uk/projects/grand-challenge/conv-issues.rnc⟩, a document using the schema would look as follows.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <converter xmlns="http://www.ukoln.ac.uk/projects/grand-challenge/conv-issues.rnc"
3             toolname="Freedy Bacrohat"
4             toolid="info:rrorife/sw/38560"
5             version="8.0">
6    <conversion source="info:rrorife/f/240" destination="info:rrorife/f/245">
7      <execution>
8        <options>
9          <option key="Preserve precise geometry" value="1" />
10       </options>
11       <features>
12         <feature property="NURBS surface" preservation="fair"
13                  degradation="unpredictable" />
14         <feature property="Geometric dimensioning and tolerancing"
```

```
                preservation="good" />
      </features>
      <comment xml:lang="en-GB">The tool has difficulty with multiply-bounded NURBS
        surfaces</comment>
    </execution>
    <execution>
      <options>
        <option key="Preserve precise geometry" value="0" />
      </options>
      <features>
        <feature property="NURBS surface" preservation="none"
                 degradation="configurable" />
        <feature property="Geometric dimensioning and tolerancing"
                 preservation="good" />
      </features>
    </execution>
  </conversion>
</converter>
```

## 6 File format characteristics

A similar schema (version 0.1) is declared for recording the characteristics of individual file formats. This is intended to supplement rather than duplicate representation information recorded in other ways in the RI RegRep, e.g. in specification documents. Documents conforming to this schema may be entered into the RI RegRep as a `Digital File Format Description`, with the schema itself entered as a `Digital File Format Description Language`, in the OAIS section of the Registry, or as a piece of `Documentation` relating to a `File Format` detailed in the PRONOM section of the Registry.

```
# Version 0.1
default namespace = "http://www.ukoln.ac.uk/projects/grand-challenge/ff-chars.rnc"
start =
  element format {
    attribute formatname { text }? &
    attribute formatid { xsd:anyURI }? &
    attribute version { text }? &
    element features {
      element feature {
        attribute property { external "properties.rnc" },
        attribute support { "full" | "partial" | "none" }?,
        Comment*
      }+
    } &
    Comment*
  }
Comment =
  element comment {
    attribute xml:lang { xsd:language },
    text
  }
```

Note that this schema uses the same list of properties as the schema for processing software.

### 6.1 Commentary

```
4   element format {
5     attribute formatname { text }? &
6     attribute formatid { xsd:anyURI }? &
7     attribute version { text }? &
16  }
```

The top level element is `format`, representing a particular version of a file format. The three optional attributes `formatname`, `formatid` and `version` provide a way of specifying the file format being described, independent of the representation information held by the Registry. The `formatid` attribute is meant to point to an existing ID, rather then declare one. Support for this has not yet been added to RRoRIfE; in the meantime, it uses local identifiers of the form '`info:rrorife/f/`*number*'.

```
8     element features {
9       element feature {
10        attribute property { external "properties.rnc" },
11        attribute support { "full" | "partial" | "none" }?,
13      }+
14    } &
```

The `property` attribute specifies one of an authority list of file format characteristics that a) may or may not be requisite for a given purpose, and b) may or may not be supported by a given file format or format conversion. This list will be one of the main points of development as the scheme approaches release. It is the same list used in the schema for known issues with processing software.

The `support` attribute sums up whether or not a `property` is supported in the file. The value `partial` indicates that support for the attribute is incomplete in some way — for example, restricted to a limited range of values.

Exactly one `features` element must be present in the document, containing one or more `feature` elements.

```
12        Comment*
15    Comment*
17  Comment =
18    element comment {
19      attribute xml:lang { xsd:language },
20      text
21    }
```

Any ambiguity or other pertinent information not able to be codified as a `feature` element can be added as human-readable text in one or more `comment` elements. The language in which the comment is written should be specified.

### 6.2 Sample document

Given an identifying URI for this schema of ⟨`http://www.ukoln.ac.uk/projects/grand-challenge/ff-chars.rnc`⟩, a document using the schema would look as follows.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <format xmlns="http://www.ukoln.ac.uk/projects/grand-challenge/ff-chars.rnc"
3          formatname="portable three dimensional format"
```

```
        formatid="info:rrorife/f/245"
        version="1.7">
  <features>
    <feature property="NURBS surface" support="partial">
      <comment xml:lang="en-GB">The format can only encode NURBS surfaces with 256
        control points or fewer</comment>
    </feature>
    <feature property="Geometric dimensioning and tolerancing" support="full" />
  </features>
</format>
```

# 7 Ontology of properties

## 7.1 Methodology

This ontology is based on the synthesis of properties either defined or identified by the following documents.


[1] M Coyne et al. *The Significant Properties of Vector Images*. JISC Digital Preservation Programme Study Report. Version 2. 2007. URL: http://www.jisc.ac.uk/media/documents/programmes/preservation/vector_images.pdf (2010-01-19).

[2] ISO/IEC 19775:2004. *Information technology – Computer graphics and image processing – Extensible 3D (X3D)*. International Organization for Standardization. URL: http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/ (2008-02-01).

[3] CK Shene. *A User Guide to the Curve Subsystem of DesignMentor*. 1997. URL: http://www.cs.mtu.edu/~shene/COURSES/cs3621/LAB/curve/curve.html (2008-02-01).

[4] CK Shene. *A User Guide to the Surface Subsystem of DesignMentor*. 1997. URL: http://www.cs.mtu.edu/~shene/COURSES/cs3621/LAB/surface/surface.html (2008-02-01).


## 7.2 Interpretation of the hierarchies

The expressed aim of this list is to identify properties that can reasonably be expected to be semantically equivalent across many different formats and software tools. For example, if two file formats each have a specific syntax for expressing a square, then one can reasonably expect to be able to construct a one-to-one mapping between the two syntaxes, and hence construct a software tool that can convert between the two.

This list expresses some pairs of properties in a parent-child relationship. In such cases, the child property is a special case of the parent property, which is to say that while any information expressed using the child property has an equivalent expression in terms of the parent property, the reverse is not true. Any values attached to a parent property are assumed to hold for all its child properties as well, unless otherwise specified. Some properties are offered as a convenience on the basis of this behaviour, rather than being true properties in their own right (e.g. 'analytic 2D geometry', 'analytic 3D geometry').

The ontology has been divided into categories for ease of comprehension: metadata, construction (i.e. techniques for expressing geometry), compression and identification, two-dimensional geometry, and three-dimensional geometry. These categories are offered as a way of presenting the properties to users – indeed, the RRoRIfE tool uses them in selection forms – but do not form part of the ontology and are not normative.

### 7.3 Metadata

*Feature semantics*
Metadata explaining the function, meaning, etc. of a shape may be attached to the definition of the shape.

*Material metadata*
Metadata specifying from what material(s) a shape should be made.

*Geometric dimensioning and tolerancing*
The specification of a shape's real world dimensions, and its allowable variations in dimensions, form, orientation, and position with respect to other shapes that may be associated with it.

- *Geometric dimensioning*
  The real world dimensions of a shape, attached to its definition.

*Assembly hierarchy*
Structuring allowing geometric instances to be grouped into parts and (multiple levels of) subassemblies, before being grouped into an assembly.

*Assembly node metadata*
Arbitrary metadata, attached to geometric instances or groups thereof

### 7.4 Construction

*Constructive solid geometry*
Complex solids are defined by adding or subtracting simpler solids.

*Boundary representation*
Complex solids are defined in terms of their surfaces.

*Trimmed surface*
A surface defined in terms of a larger surface, possibly infinite, and a boundary function that specifies its extent (e.g. two-dimensional shape, intersection with another surface).

*Parameterized re-use of instances*
Shapes can be defined once but instantiated more than once within the model, with each instance adjusted by means of parameters

- *Simple re-use of instances*
  Shapes can be defined once but instantiated more than once within the model.

*Construction history modelling*
Metadata attached to a shape that indicate how it was constructed, used by some modelling systems to allow a shape to be quickly reconstructed with slightly different parameters.

*Multiple alternative representations*
A number of different representations or descriptions may be provided for all or part of the model.

- *Multiple levels of detail*
  A number of different levels of detail may be provided for all or part of the model.

### *7.5 Compression and identification*

*Field-wise compression*
At least one data field within the file can be optionally compressed.

*Stream-wise compression*
At least one section greater than a data field within the file can be optionally compressed.

*Whole-file compression*
Both a text and a binary form of the format exist; the binary format is derived from the text format by means of compression; and at least one software tool can process the binary form directly (i.e. without recourse to writing out the text format to disc first).

*Streaming*
The file format is constructed so that a visualization of the model, albeit incomplete or at a low level of detail, may be presented using only a portion of the file.

*Identification of subassemblies*
Subassemblies may be labelled with an identifier unique within the model.

*Identification of parts*
Parts may be labelled with an identifier unique within the model. ('Part' here means a 3D shape, possibly constructed from several simpler 3D shapes and/or surfaces.)

*Identification of surfaces*
Surfaces may be labelled with an identifier unique within the model.

*Identification of edges*
Edges (lines) may be labelled with an identifier unique within the model.

*Identification of vertices*
Vertices (points) may be labelled with an identifier unique within the model.

### *7.6 2D geometry*

*Analytic 2D geometry*
Simple 2D shapes (specifically points, polylines, conic arcs, polygons and ellipses) may be defined.

- *Point*
  Point within the co-ordinate system of the graphic.
- *Polyline*
  Sequence of straight lines connecting end to end.
  - *Line*
    Single straight line.

*Conic arc*
Arc formed from a portion of a parabola, a hyperbola, or from a portion of the circumference of an ellipse or circle.

*Elliptical arc*
A portion of the circumference of an ellipse.

*Circular arc*
A portion of the circumference of a circle.

*Polygon*
Closed polyline where none of the line segments cross another.

*Triangle*
Polygon made up of three straight lines.

*Rectangle*
Polygon made up of four straight lines.

*Square*
Polygon made up of four equal straight lines.

*Ellipse*
Single, closed elliptical curve.

*Circle*
Single, closed circular curve.

*Open composite curve*
Sequence of curves, however defined, connecting end to end without looping.

*Closed composite curve*
Sequence of curves, however defined, connecting end to end and explicitly looped so that first curve continues from the last. Such curves also bound an area.

*NURBS curve*
A non-uniform rational B-spline evolving in one parametric direction. Given $(m+1)$ knots $t_i$ with $t_0 \le t_1 \le \ldots \le t_m$, a B-spline of degree $n$ is defined by the function $\mathbf{S}(t) = \sum_{i=0}^{m} \mathbf{p}_i b_{i,n}(t)$, $t \in [t_n, t_{m-n}]$, where the $\mathbf{p}_i$ are control points, $b_{j,0}(t) = 1$ if $t_j \le t < t_{j+1}$, 0 otherwise, and $b_{j,n}(t) = \frac{t-t_j}{t_{j+n}-t_j} b_{j,n-1}(t) + \frac{t_{j+n+1}-t}{t_{j+n+1}-t_{j+1}} b_{j+1,n-1}(t)$. Non-uniform means that the knots are not (necessarily) equidistant. Rational means that the effect of each knot is given a scalar weight.

*Rational Bézier curve*
Special case of a NURBS curve where the knots are uniform (equidistant).

*Non-rational Bézier curve*
Special case of a rational Bézier curve where the effect of each knot is given equal weight.

*Cubic Bézier curve*
The curve traced between $\mathbf{p}_{t=0}$ and $\mathbf{p}_{t=1}$ by the parametric function $(1-t)^3\mathbf{p}_{t=0} + 3t(1-t)^2\mathbf{p}_c + 3t^2(1-t)\mathbf{p}_d + t^3\mathbf{p}_{t=1}$, where $\mathbf{p}_c$ and $\mathbf{p}_d$ are control points.

*Quadratic Bézier curve*
The curve traced between $\mathbf{p}_{t=0}$ and $\mathbf{p}_{t=1}$ by the parametric function $(1-t)^2\mathbf{p}_{t=0} + 2t(1-t)\mathbf{p}_c + t^2\mathbf{p}_{t=1}$, where $\mathbf{p}_c$ is a control point.

### *7.7 3D geometry*

*Analytic 3D geometry*
Simple 3D shapes (specifically point clouds, helices, planes, ellipsoids, cylinders, cones, cuboids and tori) may be defined.

*Point cloud*
Set of points, usually indicating corners, edges and/or surfaces.

*Helix*
Helical curve.

*Plane*
2D surface orientated in 3D space.

*Ellipsoid*
Solid ellipsoid.

*Spheres*
Solid sphere.

*Cylinder*
Solid cylinder.

*Cone*
Solid cone.

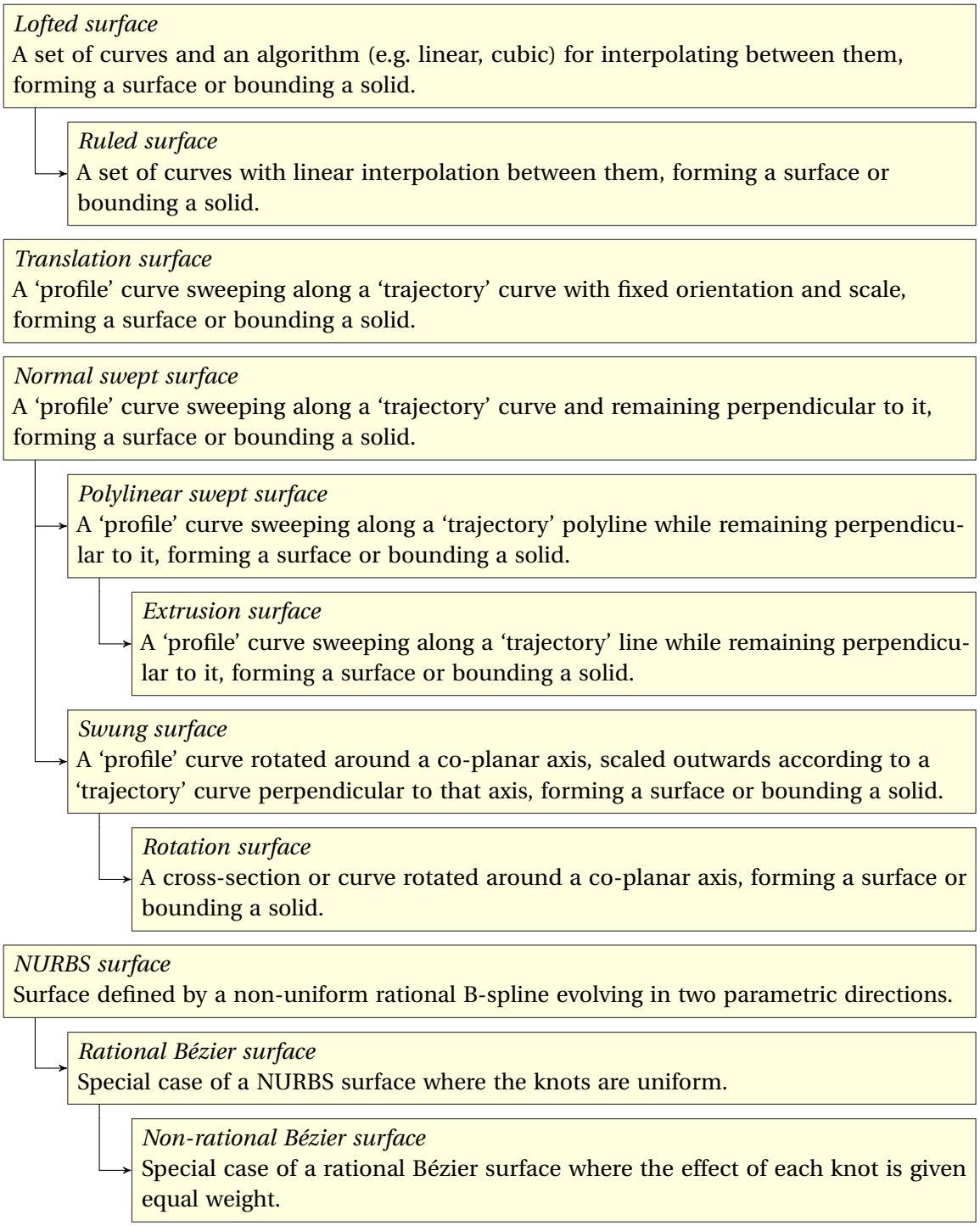
*Cuboid*
Solid cuboid.

*Cube*
Solid cube.

*Torus*
Solid torus.

*Mesh of surface segments*
Mesh composed of arbitrary surface segments, forming a surface or bounding a solid.

*Mesh of tessellating triangles*
Mesh composed of tessellating triangles, forming a surface or bounding a solid.




## 8 High-level source code structure

The source code for RRoRIfE is made up of four packages in the `uk.ac.ukoln.rrorife` family. Two of them – `conv` and `ff` – are generated automatically from the schemata that define the two XML formats used in the registry, using JAXB [5]. A further package, `schema`, implements the RRoRIfE ontology of properties as a type of `JTree`. The remainder of the code is in the `gui` package.

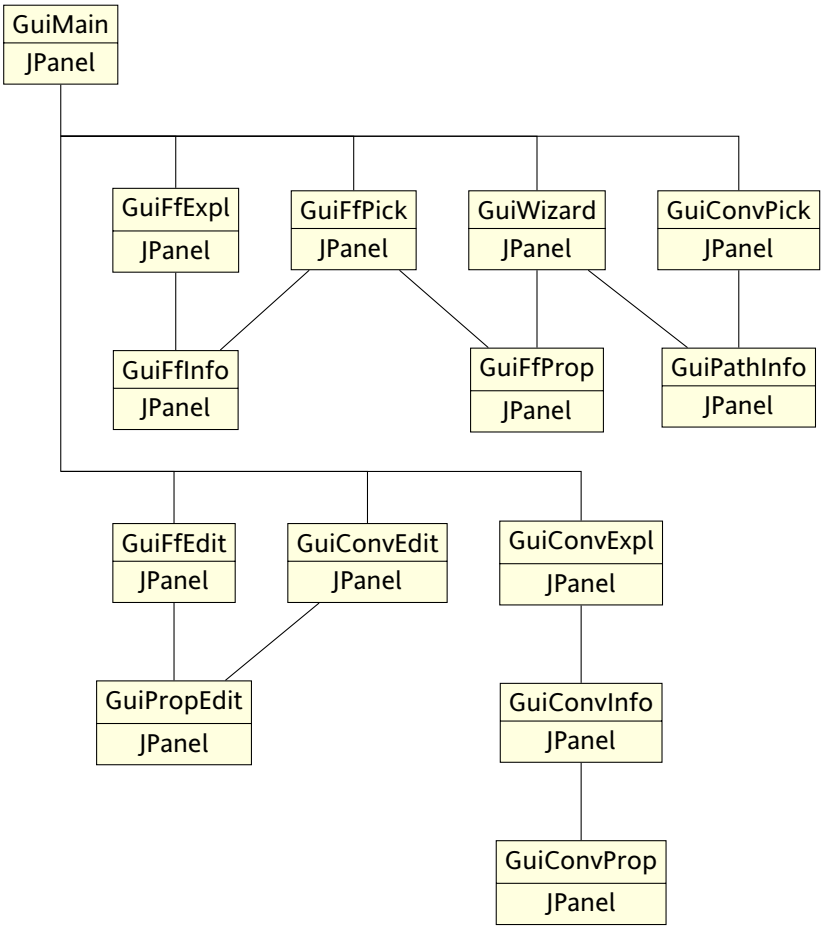



**Figure 1: Class structure of the user interface**

The actual user-facing elements are in classes with names beginning 'Gui'. They are all extension of the JPanel class. GuiMain is the class that is actually run. It contains seven functional panels presented as a deck of cards:

1. GuiFfExpl: File Format Explorer
2. GuiFfEdit: File Format RepInfo Editor
3. GuiFfPick: File Format Picker
4. GuiConvExpl: Converter Explorer
5. GuiConvEdit: Converter RepInfo
6. GuiConvPick: Converter Path Finder
7. GuiWizard: Conversion Path Wizard

Each of these functional panels relies on further reusable panels that specialize in presenting a particular form of information.

- GuiFfInfo: displays the characteristics of a file format.
- GuiFfProp: provides an interface for prioritizing file format characteristics.
- GuiConvInfo: provides an interface for selecting an input format–converter–output format triple.

- GuiConvProp: displays the different ways in which a single converter can migrate information between two specified file formats, and the impact each has on file format characteristics.
- GuiPathInfo: displays a migration pathway between two file formats.
- GuiPropEdit: provides an interface for editing the properties of a file format or an input format–converter–output format triple.

These relationships are mapped out in Figure 1. In addition, the JLabelWrap class is defined, providing a version of JLabel which has a dynamic width up to a specified maximum.

The `gui` package provides several classes for storing data.

- Registry: this loads information from the XML files in the registry, writes information to the XML files in the registry, and provides utility functions for working with the loaded information.
- RrorifeID: superclass for FormatID and SoftwareID; provides tools for working with the literal (string) identifiers for formats and converters.
- FormatID: surrogate class for a format in the registry; provides some additional functions not present in the auto-generated classes derived from the XML schema.
- SoftwareID: surrogate class for a converter in the registry; provides some additional functions not present in the auto-generated classes derived from the XML schema.
- Pathway: this stores a sequence of format migration stages, along with a map of scores for how well the pathway preserves file format characteristics (used, along with a map of weightings/priorities, to compare performance).

Some additional classes are defined to help work with the data.

- LanguageCode: enumeration providing a controlled vocabulary for specifying the language in which comments are written.
- Requirement: enumeration providing a controlled vocabulary for expressing the significance or otherwise of a file format characteristic in a particular circumstance.
- NumStringSort: sorts strings consisting of arabic digits as if they were numbers, falling back to regular string sorting for all other strings.
- FormatScoreSort: sorts the (FormatID) keys of a map according to their (float) values; used for ordering formats according to how well they suit a set of requirements.

There are also classes representing background processes.

- ProPanelLoader: loads a functional panel into GuiMain; used to improve startup time.
- ProFormat: searches the registry for formats matching a profile of requirements.
- ProPath: searches the registry for pathways that exist between two formats, in a specified number of migration stages or fewer.
- ProWizard: calls ProFormat to retrieve formats matching a profile of requirements; for each format calls ProPath to retrieve pathways that exist to it from the given starting format, scoring or discarding each one according to how well it fulfils the requirements; then for all remaining paths, ranks them according to score and length.

## 9 Acknowledgements

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) and the Economic and Social Research Council (ESRC) under Grant Numbers EP/C534220/1 and RES-331-27-0006.

## 10 Licensing

RRoRIfE itself is licensed under the MIT Licence.



RRoRIfE uses and is distributed with Brunch Boy Design's RelativeLayout constraint-based layout manager. The copyright statement and licence for RelativeLayout is as follows.

## References

[1] PV Biron & A Malhotra( eds.). *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation. World Wide Web Consortium. 2004. URL: http://www.w3.org/TR/xmlschema-2/ (2010-01-19).

[2] T Bray et al.( eds.). *Extensible Markup Language (XML) 1.1 (Second Edition)*. W3C Recommendation. World Wide Web Consortium. 2006. URL: http://www.w3.org/TR/xml11/ (2010-01-19).

[3] J Clark( ed.). *RELAX NG Compact Syntax*. Committee Specification. Organization for the Advancement of Structured Information Standards. 2002. URL: http://relaxng.org/compact.html (2010-01-19).

[4] J Clark, J Cowan & M Murata( eds.). *RELAX NG Compact Syntax Tutorial*. Working Draft. Organization for the Advancement of Structured Information Standards. 2003. URL: http://relaxng.org/compact-tutorial.html (2010-01-19).

[5] *JAXB Reference Implementation Project*. 2011-02-11. URL: http://jaxb.java.net/ (2011-02-19).

[6] W3Schools. *XML Tutorial*. 2006. URL: http://www.w3schools.com/xml/ (2010-01-19).