**QuARS**

**User Manual (version 0.5)**

Giuseppe Lami (giuseppe.lami@isti.cnr.it)

Gianluca Trentanni (gianluca.trentanni@isti.cnr.it)

Consiglio Nazionale delle Ricerche - Istituto di Scienza e Tecnologie dell'Informazione A. Faedo - System and Software Evaluation Center - Via Moruzzi 1, 56124 Pisa, Italy

*Abstract.* The QuARS (Quality Analyzer of Requirements Specifications) tool makes it easier to extract structured information and metrics for detecting linguistic inaccuracies and defects in software requirements expressed in Natural language. In the following is presented the user manual.
Introduction

The QuARS (Quality Analyzer for Requirement Specifications) is an automatic tool that supports the analysis of requirements documents written in Natural Language. The functionalities provided by QuARS are:

- **1. Defect identification**
  - **QuARS** performs a linguistic analysis of a requirement document in plain text format and points out defects belonging to the following categories:
    - **Ambiguity**: defects belonging to this category are pointed out when detected sentences:
      - contain terms inherently vague, optional parts or weak verbs,
      - express personal opinions or feelings,
      - have the subject or object generically expressed,
    - **Specification completion**: defects belonging to this category are pointed out when sentences containing generic unspecified objects are detected
    - **Understandability**: defects belonging to this category are pointed out when sentences with multiple subject or main verb are detected

- **2. Requirements clustering**
  - **QuARS** is able to extract subsets of the requirements document composed of sentences dealing with a specific topic.
  - A possible use of this functionality is the extraction of a class of non-functional requirements (e.g. those requirements dealing with a property of the system to be developed).
  - The derived clusters are recorded and made available to the user.

- **3. Metrics derivation**
  - **QuARS** calculates metrics during the analysis of a requirements document.
  - The available metrics are:
    - The Coleman-Liau Formula readability metrics: \((15.89*\text{chars/ws}-0.3*\text{sentences/(100*wsds)}-15.8))\)
    - The reference value of this formula for an easy-to-read technical document is 10, if it is>15 the document is difficult-to-read.
    - The defect rate related to the defects described in 1.

The QuARS's key components are special text files called Dictionaries.
- Two types of Dictionary exist: the defect-related dictionaries and the domain-dictionaries.
  - The defect-related dictionaries contain the terms able to reveal a defect according to what stated in 1.
  - The domain-dictionaries contain the terms belonging to a specific domain and that determine if a sentence is to be included in a cluster or not.

References

- **S.Gnesi, G.Lami, G.Trentanni, F.Fabbrini, M.Fusani**
  - *An Automatic Tool for the Analysis of Natural Language Requirements*

- **G. Lami, G. Trentanni**
  - *An Automatic Tool for Improving the Quality of Software Requirements*

- **A.Fanetti, S.Gnesi, G.Lami, A.Maccari**
  - *Application of Linguistic Techniques for Use Case Analysis*

- **A.Fanetti, S.Gnesi, G.Lami, A.Maccari**
  - *Linguistic Techniques for Use Cases Analysis*

- **F.Fabbrini, M.Fusani, S.Gnesi, G.Lami**
  - *The Linguistic Approach to the Natural Language Requirements Quality: Benefits of the use of an Automatic Tool*

- **F.Fabbrini, M.Fusani, S.Gnesi, G.Lami**
  - *An Automatic Quality Evaluation for Natural Language Requirements*
F. Fabbrini, M. Fusani, S. Gnesi, G. Lami

Quality Evaluation of Software Requirement Specifications

[QuARS HOME] [Contents] [Index]

QuARS License

SOFTWARE LICENSE AGREEMENT

PLEASE READ THIS SOFTWARE LICENSE AGREEMENT CAREFULLY BEFORE DOWNLOADING OR USING THE SOFTWARE.

BY CLICKING ON THE "ACCEPT" BUTTON, OPENING THE PACKAGE, DOWNLOADING THE PRODUCT, OR USING THE EQUIPMENT THAT CONTAINS THIS PRODUCT, YOU ARE CONSENTING TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, CLICK THE "DO NOT ACCEPT" BUTTON AND THE INSTALLATION PROCESS WILL NOT CONTINUE, RETURN THE PRODUCT TO THE PLACE OF PURCHASE FOR A FULL REFUND, OR DO NOT DOWNLOAD THE PRODUCT.

Single User License Grant: ISTI-CNR grant to Customer ("Customer") a nonexclusive and nontransferable license to use the QuARS ("Software") in object code form solely for testing and non-commercial purpose.

Customer may make one (1) archival copy of the Software provided Customer affixes to such copy all copyright, confidentiality, and proprietary notices that appear on the original.

EXCEPT AS EXPRESSLY AUTHORIZED ABOVE, CUSTOMER SHALL NOT: COPY, IN WHOLE OR IN PART, SOFTWARE OR DOCUMENTATION; MODIFY THE SOFTWARE; REVERSE COMPILE OR REVERSE ASSEMBLE ALL OR ANY PORTION OF THE SOFTWARE; OR RENT, LEASE, DISTRIBUTE, SELL, OR CREATE DERIVATIVE WORKS OF THE SOFTWARE.

Customer agrees that aspects of the licensed materials, including the specific design and structure of individual programs, constitute trade secrets and/or copyrighted material of ISTI-CNR. Customer agrees not to disclose, provide, or otherwise make available such trade secrets or copyrighted material in any form to any third party. Customer agrees to implement reasonable security measures to protect such trade secrets and copyrighted material.

Title to Software and documentation shall remain solely with ISTI-CNR.

DISCLAIMER. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE, ARE HEREBY EXCLUDED TO THE EXTENT ALLOWED BY APPLICABLE LAW.

IN NO EVENT WILL ISTI-CNR OR ITS SUPPLIERS BE LIABLE FOR ANY LOST REVENUE, PROFIT, OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL, OR PUNITIVE DAMAGES HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE EVEN IF ISTI-CNR OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The Software QuARS is in a prototyal development phase, it is to be considered a beta software, and is made available for testing or demonstration purpose only. For this it is provided AS IS without any warranty whatsoever.

This License is effective until terminated. Customer may terminate this License at any time by destroying all copies of Software including any documentation. This License will terminate immediately without notice from ISTI-CNR if Customer fails to comply with any provision of this License. Upon termination, Customer must destroy all copies of Software.

Installing QuARS

QuARS comes as a unique self-extracting executable file with all is needed packed inside.
To install QuARS (version 4.1) starting from the QuARS.v41 setup.exe file please follow these simple steps:

1. Navigate your file system to where you downloaded the self-extracting executable file QuARS.v41 setup.exe.
2. Double clicking on the file QuARS.v41_setup.exe a new directory named QuARS.v41 will be created in the same place.
3. If you please, you can now delete the QuARS.v41_setup.exe file.
4. Come inside the new QuARS.v41 directory: QuARS environment is already prepared to start analyzing requirements files.
5. If you prefer you may move the directory QuARS.v41 where you prefer onto your Hard Disk: QuARS will work anyway.

Uninstalling QuARS

Uninstallation Instructions

- Simply remove the installation "QuARS.v41" directory where you installed QuARS.

System Requirements

- Operating Systems*
  - Windows 95
  - Windows 98
  - Windows 98SE
  - Windows ME
  - Windows NT 4.0
  - Windows 2000 (Recommended)
  - Windows XP

- Minimum Hardware
  - Pentium 233 MHz (Recommended: Pentium 500MHz or greater)
  - 64 MB RAM (Recommended: 128 MB RAM or greater)
  - 25 MB hard drive space

- Recommended Screen Resolution: 1024x768
  - Best View: 1152x864

*Note: At this moment QuARS runs only on Microsoft Windows systems.
QuARS: Getting Started

QuARS GUI overview

The QuARS's GUI is composed of three main frames:

- **Input Frame**: by this frame it is possible to load and display the plain text file containing the requirements to be analyzed. The input frame provides the principal functions (buttons) for editing the input requirements.
- **Dictionary Frame**: this frame allows to select, display, modify or create the dictionary corresponding to the type of analysis of interest.
- **Output Frame**: in this frame the results of the analysis are displayed.

Getting Started

In the following the initial steps for starting up the analysis of textual requirements with QuARS are described. Only the main functions of QuARS are described here, for more detailed user guidance see the complete User Manual section of this help.

- **Step 1**: LOAD A FILE
Click the "Load" button and browse the file system to select a plain text file containing the requirements to be analyzed.

The selected file is displayed in the input frame.

- **Step 2: SELECT THE TYPE OF THE ANALYSIS**
  - Three types of analysis are available: Defect Identification using Lexical Analysis ("L" button on the top of the GUI), Defect Identification using Syntax Analysis ("S" button on the top of the GUI) and Views Derivation - Clustering ("V" button on the top of the GUI)

- **L - Lexical Analysis** for defect identification:
  - This function points out the occurrences of the special defect-revealing terms contained in the dictionaries associated to this type of analysis.
  - It is possible to modify the content of the dictionaries associated to this function and also to add new dictionaries.
  - The default set of available lexical analyses is composed of four dictionaries optionality, subjectivity, vagueness, weakness
**S - Syntax Analysis** for defect identification:
This function identifies defective sentences in terms of *implicity, multiplicity* and *underspecification*. For doing that the syntactical structure of each sentence in the input requirement document is derived.
It is possible to modify the content of the dictionaries associated to this function but not add new dictionaries.

**V - Views Derivation** - Requirements clustering:
This function extracts those sentences dealing with a particular topic from the input requirements document and puts them in a special cluster, called View.
The view derivation is based on the existence of domain-specific dictionaries, called View-dictionaries.
It is possible to modify existing and create new View-dictionaries.

⚠️ Since the "View Derivation" is based on the identification of (sub)sections, to perform this kind of analysis it is mandatory that the requirements document is divided into sections according to the following format:

1.
  1.1
  1.1.1
  
  ...
  1.2
  
  ...
  n.
  
  ...

For an explanation of the meaning of the Indicators see "Indicators Explanation"

---

**Step 3: START ANALYSIS**

![QuARS screenshot](image)

Once the requirements document to be analyzed has been loaded and the type of analysis selected, push the Analysis button to start the analysis.

**Step 4: MANAGE RESULTS**

- **Managing results of Lexical analysis for defect identification**
The results of this type of analysis consist of a set of sentences containing defect-revealing words.
This set is displayed on the output frame.
Defective sentences are displayed and the defect-revealing terms are highlighted.

The defective sentences can be traced on the input document by clicking on the identification number. This feature facilitates the correction of the found defect.

- **Managing results of Syntax analysis for defect identification**
  The results of this type of analysis consist of a set of defective sentences according to the selected type of syntax analysis.
  The way the results of syntax analysis for defect identification are displayed and managed is the same as the lexical analysis.

- **Managing results of the Views derivation**
  The result of the View derivation consists of:
  - the cluster of the sentences belonging to the View (i.e., those sentences dealing with the topic the View is related to).
The cluster is displayed in the output frame.
Each sentence of the cluster is displayed preceded by the number of its section in the requirement document.

- the graphic representation of the number of the sentences belonging to a view in each section of the requirements document is displayed.
This is provided as a MS Excel graph.

• Metrics Calculation

During the performance of the analysis QuARS calculates some metrics.
The available metrics are:

- Defect rate: for each Lexical/Syntactical analysis for defect identification performed, the rate between the found defects and the total number of sentences the requirements document is composed of is calculated.
- The Coleman-Liau readability metric is calculated when a file is loaded.
- The metrics above are made available in the following ways:
  - They are displayed on the bottom of the output frame after along with the analysis results.
  - They can be all displayed in isolation selecting the "View Metrics" option from the "Metrics & Log" menu.
  - They can be saved in a special log file selecting the "Save Available Metrics" option from the "Metrics & Log" menu

• Step 5: SAVE LOGS
The results of the performed analysis in the current session can be saved as text files using the function in the "Metrics & Log" menu.

⚠️ QuARS analyses one paragraph at a time, considering a paragraph delimited by the special "end line" char \ independendty of the number of sentences it contains.

**TIP:** in the case of long paragraphs containing several sentences, it's preferable for a more accurate analysis to break them into single-sentence paragraphs.

[QuARS HOME] [Contents] [Index]

**QuARS User Interface**

Menu and Buttonbox

Dictionary Editor Area

QuARS Output Area

Sentences Editor Area

Managing Sentences

The Status-bar

For a detailed description of the QuARS User Interface by mean of a function-based division, please refer the links below:

- Menu and Buttonbox: Menu Reference, Menu Reference Summary, Buttonbox Reference
- Dictionary Editor Area
- Sentences Editor Area
- Output Frame
- The Status-Bar
QuARS Dictionary Editor Reference

The QuARS dictionary frame is indeed a complete editor: you can select text area for cut and paste and exploit the provided drag and drop functionality.

Buttons functionality:

- **New**: pops up the QuARS Dictionary Wizard allowing the creation of a new dictionary that will be displayed among the others
- **Delete**: deletes the file related to the displayed dictionary and the dictionary itself from the current set
- **ReLoad**: loads again the file being displayed if there isn’t actually a file loaded, it behaves like **Load**
- **Save**: saves the editor content without prompting for the name if the name of the file to save is not known (e.g. via a New command) it behaves like **Save As** if the file already exists, an overwrite confirmation will be asked
- **Clear**: clears editor content setting it up for a new blank file
- **Print**: prints the editor content

As you can see in the image above, every kind of analysis (Lexical", "Syntactic" and the "View Derivation") can be performed by means of several Dictionaries. The Dictionaries Area is "notebook" organized:
- every notebook represents a kind of analysis
- every page in a notebook represents a Dictionary
- pages (Dictionaries) can be added!
- switching from a page to another automatically the related analysis will be performed using the "Analysis: [Kind] dict-name" button.
- obviously, notebooks cannot be added: at this moment only three kind of analysis are available

Please refer to this **Keyboard Shortcuts** list for all the others functionalities and here for the Dictionary name format and limitations.

QuARS Output Frame: managing Sentences

Result Tracing

The single defective sentences pointed out in the QuARS output frame as result of the Defect Identification function, can be traced on the input file by clicking the corresponding line number in the output frame. Once the line number has been clicked, the corresponding sentence is highlighted in the input frame, ready for correction.
Hide Results (Tracking False-Positives)

It is possible to hide the false positives (i.e. sentences pointed out by QuARS as defective but actually considered correct by the user) by marking the check-button in the output window corresponding to the sentence to be hidden. Once a sentence has been checked, when the analysis is performed again, this sentence will be not more displayed in the QuARS output window even though still defective.

To see what sentences are hidden, click the Track button in the QuARS output frame: a list of all hidden sentences will be shown. By clicking on the entry "Select All", all hidden sentences will be selected in a brown background color, by clicking on the entry "Deselect All", all sentences will be selected in a brown background color.

To let a hidden sentence be displayed again, click the "Manage" button in the QuARS output frame and deselect from the list the related entry.
To let all the hidden sentences be displayed again, click the "Resume All" button or the "Resume All" entry of the "Manage" button. Use the "Select All" entry of the "Manage" button to hide all the sentences from analysis (useful in high-grade of false-positive documents).

The QuARS Status-Bar shows several information about the current QuARS session. Starting from left to right, the "Status Label" and its related icon show QuARS internal status, the actual activity and possible errors or troubles. The second item shows the current loaded Sentences file ready to be analyzed or the "No Sentences File Loaded" message. The third information is about the Coleman-Liau readability index related to the current file being shown in the QuARSEditor window. The last information is about QuARS itself, version and current build release.

The possible messages shown by the Status Label are:

<table>
<thead>
<tr>
<th>#</th>
<th>Status Label</th>
<th>Status Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>⬤ Ready!</td>
<td>QuARS is ready to start the next analysis</td>
</tr>
<tr>
<td>02</td>
<td>⬤ Starting Computation... Environment</td>
<td>A necessary step preceding any analysis: files and environment variables are ready for being analysed</td>
</tr>
<tr>
<td>03</td>
<td>⬤ QuARS Engine: Analysis...</td>
<td>Analysis lasty starts....</td>
</tr>
<tr>
<td>04</td>
<td>⬤ QuARS Engine: Lexical Analysis...</td>
<td>A Lexical Analysis has been requested</td>
</tr>
<tr>
<td>05</td>
<td>⬤ QuARS Syntactic Analysis: Parsing...</td>
<td>A Syntactical Analysis has been requested: this is the syntactic parsing step</td>
</tr>
<tr>
<td>06</td>
<td>⬤ QuARS Engine: Syntactic Analysis...</td>
<td>A Syntactical Analysis has been requested: this is the syntactic defects check step</td>
</tr>
</tbody>
</table>
### Dictionary Handling

The tool **QuARS** has been designed to be highly tailorable according to particular application domain and different user needs. The main aspect of the **QuARS** tailorability is its capability to handle the dictionaries. In fact, it possible to modify an existing dictionary and make permanent these modifications. It is possible also, for all the types of analysis but the syntax-based one, to create new dictionaries and delete existing ones. For this purpose a set of function buttons are available in the Dictionaries frame.

In the following they are described in detail:

- **New**: this function allows the creation of a new dictionary.
  This function is not allowed for the syntax-based analyses. The capability of adding new dictionaries for lexical-based analysis and View derivation allows the **QuARS** tool to be adapted to particular application domain and user needs.
- **Delete**: this function allows a dictionary to be deleted.
  This function is not allowed for syntax-based analyses.
- **Reload**: this function allows to return to the last saved version of a dictionary.
- **Save**: dictionaries are modifiable by editing their content in the Dictionary frame directly.
After a modification has been made on a dictionary, this function makes permanent the new version of the dictionary.

- **Clear**: this function cancel the content of a dictionary, this cancellation becomes permanent when the Save function is selected.
- **Print**: this function allows a dictionary to be printed out.
- 
- **Drag and Drop**: it is possible to "drag and drop" a text file over the *QuARS* dictionaries frame: automatically, as soon as the left mouse button is released, a new dictionary will be created with the first 15 characters of the name of the dragged file (and with spaces converted to underscore) and with the same content of the dragged file. This function is not allowed for the Syntactical Analysis.

A more detailed guide on "How to create a new Dictionary" is available [here](#).

---

**Input Handling**

---

### Loading input files

*QuARS* analyzes files in plain text (.txt) format. For loading texts for the analysis the following steps have to be followed.

- Click the **Load** button on the Input frame of the *QuARS* GUI.
- Browse the file system and select the textual file containing the requirements to be analyzed.
- Click the **Open** button.

The content of the text file to be analyzed is displayed in the input frame of the *QuARS* GUI. Each line of the input file is numbered on the input frame in order to facilitate the identification of the defective sentences found.

### Editing of the input file

The input file can be edited on the input frame of the *QuARS* GUI. The most common functions of a text editor are available in the Input Frame Toolbar.

Once the text has been edited, it is possible:

- to save the changes made by clicking the **Save** button.
- to save the changed file with a new name by clicking the **Save As** button.

To come back to the last saved version of the input file the **Reload** button has to be clicked. The **Clear** button cancels the contents of the input file. The **Print** button allows to print out the input file.

---

### Converting an input file in the suitable format

If the format of the input file is not plain text, it has to be converted. In the following the procedure to be followed for the most common commercial text formats:

- **MS Word** (.doc):
  - Export the entire file
    - Select from the menubar "File" --> "Save as"
    - Select "text file (.txt)" as saving file format
    - Select where saving the file
    - Click the "Save" button
  - Export an excerpt of the file
    - Select the text of interest
    - Copy the selected text
    - Past the copied text in a plain text file
- **Adobe Acrobat** (.pdf):
The file is now in the suitable format for being analyzed by QuARS.
The editorial format, in case of bullets, tables, pictures, will be lost in the new plain text file.

**Note:** The consequences of that can be relevant.
In particular, it can be difficult to maintain the alignment between the original input file and the new version.

[QuARS HOME] [Contents] [Index]

**Performing Analysis**

> **QuARS** analyses one paragraph at a time, considering a paragraph delimited by the special "end line" char "¶" independently of the number of sentences it contains.

> **TIP:** in the case of long paragraphs containing several sentences, it's preferable for a more accurate analysis to break them into single-sentence paragraphs.

### Defect Identification

#### Lexical analysis

To perform one of the lexical-based analyses it is necessary to select the button on the top tool bar of QuARS (arrow 1 in figure A1).

Once the lexical-based analysis has been selected, in the Dictionaries frame the dictionaries corresponding to all the available lexical-based analyses are shown (arrow 2 in figure A1).

The default set of analyses is composed of four dictionaries: **Optionality, Subjectivity, Vagueness, Weakness**.

Additional dictionaries can be created to enlarge the set of available analyses.

It is possible to select the kind of analysis of interest by selecting the correspondent dictionary book-mark in the Dictionaries frame.

Before starting the analysis the text file containing the requirements has to be loaded.

Once the input file has been selected, its content appears in the Input frame.

The analysis starts when the Analysis button is pushed (arrow 3 in figure A1).

In the Output frame those sentences containing the particular defect we are investigating on are shown along with the indication of the individual term that makes the sentence defective (according to the kind of analysis selected).

The defective sentences can be now corrected.
Syntactic analysis

To perform the syntax-based expressiveness analysis the $button on the top tool-bar of the QuARS GUI (arrow 1 in figure A2).

Once the syntax-based analysis has been selected, in the Dictionaries frame the possible dictionaries of each type of the available syntax-based analyses are shown (arrow 2 in the figure).

It is possible to select the type of analysis of interest by selecting the correspondent dictionary tab in the Dictionaries frame (note that the multiplicity analysis doesn’t need any dictionary, for this reason it is void).

The available analyses are: Implicity, Multiplicity and Underspecification.

Before starting the analysis the text file containing the requirements has to be loaded.

Once the input file has been selected, its content appears in the Input frame.

The analysis starts when the Analysis button is pushed (arrow 3 in figure A2) or [Syntactic] name of the analysis selected from the Analysis menu on the top of the GUI.

In the Output frame those sentences containing the particular defect we are investigating on are shown along with the indication of the individual term that makes the sentence defective (according to the kind of analysis selected).

The defective sentences can be now corrected.
The "View Derivation" relies on dictionaries. The Dictionaries of the Views (V-Dictionaries) contain domain-related terms (instead of defect-related terms). A V-Dictionary is the set of all the terms dealing with a particular View, i.e., those terms that, if contained in a sentence, indicate that this sentence is dealing with the argument the View is referred.

To select the View derivation functionality of the QuARS tool the V button on the top buttonbox of the GUI has to be clicked (arrow 1 in figure A3).

Once the View derivation functionality has been selected, in the Dictionaries frame the available dictionaries are shown (arrow 2 in figure A3). Each dictionary in the Dictionaries frame corresponds to a particular View that can be derived.

It is possible to select the View of interest by selecting the correspondent V-Dictionary. Once the requirements file is loaded, the View derivation can be started.

To start the View derivation push the Analysis button or select [View derivation] name of the view from the Analysis menu on the top of the GUI.

Since the "View Derivation" is based on the identification of (sub)sections, to perform this kind of analysis it is mandatory that the requirements document is divided into sections according to the following format:

```
1.1.1
1.2
```

Note: The way the analysis is performed is the same as the lexical-based analysis. The difference between the two types of analyses is transparent to the user.

In fact, when performing these analyses the tool, first derive the syntactical structure of each sentence in the input file. Anyway, the syntactical structure of the sentences is not displayed.

While for the implicit and underspecification analysis dictionaries are necessary, the multiplicity analysis, even though it relies on the syntax structure of the sentences, for its nature, doesn’t need any dictionary.

You can even refer to a detailed description of the meaning of the Indicators [here](...)
The output of the View derivation is composed of:

- A cluster of the sentences, extracted from the input requirements file, dealing with the specific topic the View is referred to (arrow 3 in figure A3).
- An MS Excel graph, showing the total number of sentences and the number of those sentences belonging to a view contained in each section of the document analyzed (arrow 4 in figure A3).

Results Handling

Result Description

QuARS provides two types of results:

1. Defect Identification
   The Defect Identification is the result of the application of the Lexical and Syntactical analysis.
   The format of the output differs according to the type of analysis performed.
   The results of the following analyses:
   - optionality
   - subjectivity
   - vagueness
   - weakness
   - duplicity
   - underspecification
   are displayed with the following standard format:

      QuARS [type of analysis] ANALYSIS
After the initial line the list of the lines of the document under analysis is provided. Each potentially defective line is displayed according to the following formats.

For the **lexical analysis**, the format is:

> The line number:
> [the number of the potentially defective line in the requirements document is displayed here] [the entire line is displayed here]
>
> is defective because it contains the wording: [the found indicator (single word or multiple words) that causes the defect is displayed here]

---

### QuARS Output

**----------- QuARS [Lexical] optionality ANALYSIS -----------**

**The line number:**

- 205. 1061: the system shall identify and products, if appropriate, all training products, materials, and/or information

is defective because it contains the wording: **and/or**

---

### QuARS Output

**----------- QuARS [Lexical] optionality ANALYSIS -----------**

**The line number:**

- 205. 1061: the system shall identify and products, if appropriate, all training products, materials, and/or information

is defective because it contains the wording: **if appropriate**

---

### QuARS Output

**----------- QuARS [Lexical] optionality ANALYSIS -----------**

**The line number:**

- 201. 2033: the system shall provide security measures to ensure only authorized user access to restricted and/or

is defective because it contains the wording: **and/or**

---

### QuARS Output

**----------- QuARS [Lexical] optionality Statistics (on "requirements.txt" file): -----------**

Number of evaluated sentences: 332
Number of defective sentences: 3
Defect rate: 0%

---

### QuARS Output

**----------- QuARS [Syntactic] implicitness ANALYSIS -----------**

**The line number:**

- 2. 222: each po claim shall identify security objectives and it security requirements statements contained in the at

contains an implicit sentence: **implicit object**

---

### QuARS Output

**----------- QuARS [Syntactic] implicitness ANALYSIS -----------**

**The line number:**

- 115. 2064: the system shall provide the capability to maintain army universal task list (aut) information, this requi

contains an implicit sentence: **implicit determiner**

---

### QuARS Output

**----------- QuARS [Syntactic] underspecification ANALYSIS -----------**

**The line number:**

- 14. 1941: the system shall support implementation of performance based testing

contains a unspecified sentence because the term: **testing**

---

### QuARS Output

**----------- QuARS [Syntactic] underspecification ANALYSIS -----------**

**The line number:**

- 49. 2301.26: the system shall associate a collective task with one or more unit types

contains a unspecified sentence because the term: **unit**
The format of the results of the **multiplicity analysis** is:

- **The line number:**
  - The number of the potentially defective line is displayed here.
  - The entire line is displayed here.
- **Contains a multiple sentence:** More than one main verb

**QuARS Output:**

--- QuARS (Syntactic) multiplicity ANALYSIS ---

**The line number:**

- **5. 463:** The system shall provide the capability to manage and manipulate the education/training catalog.
- **contains a multiple sentence:** More than one main verb

**The line number:**

- **11. 756:** The system shall provide a capability for an organization user to select the soldier training publication [st]
- **contains a multiple sentence:** More than one subject

---

**Result Tracing**

The single defective sentences pointed out in the QuARS output frame as result of the **Defect Identification** function, can be traced on the input file by clicking the corresponding line number in the output frame. Once the line number has been clicked, the corresponding sentence is highlighted in the input frame, ready for correction.

---

**Hide Results (Tracking False-Positives)**

It is possible to hide the false positives (i.e. sentences pointed out by QuARS as defective but actually considered correct by the user) by marking the check-button in the output window corresponding to the sentence to be hidden. Once a sentence has been checked, when the analysis is performed again, this sentence will be not more displayed in the QuARS output window even though still defective.
- To see what sentences are hidden, click the Track button in the QuARS output frame: the list of all actually hidden sentences will be shown.
- By clicking on the entry "Select All", all hidden sentences will be selected in a brown background color
- by clicking on the entry "Deselect All", all sentences will be de-selected
- To let a hidden sentence be displayed again, click the "Manage" button in the QuARS output frame and deselect from the list the related entry

- To let all the hidden sentences be displayed again, click the "Resume All" button or the "Resume All" entry of the "Manage" button
- Use the "Select All" entry of the "Manage" button to hide all the sentences from analysis (useful in high-grade of false-positive documents)

---

**Saving Results**

The log containing the results of the analyses performed with QuARS can be saved in special log files.
To save the current log (i.e. the results of the last analysis performed select.

*Metrics&Logs -> Save Current Log.*

To save the logs of the current analysis session select.

*Metrics&Logs -> Save Available Logs.*
Menu Reference

This menu reference describes each menu item in the QuARS main window menu. You can even refer to the bird-eye summary table here.

- **File**
  - Load a Sentences File
  - Print QuARS Output
  - Page Setup
  - Quit

- **Edit**
  - Copy Selection
  - Copy QuARS Output
  - Clear QuARS Output
  - Select All
  - Deselect

- **View**
  - Wrap/Unwrap Text
  - Increase Text Size
  - Decrease Text Size
  - Style
    - on-line
    - plain
    - tagged
    - tagged 2
    - tagged 3

- **Analysis**
  - Create New Dictionary
  - Switch to Lexical Analysis
  - Switch to Syntactic Analysis
  - Switch to Views Derivation
  - (Dynamic) List of Available Lexical Analysis
  - (Dynamic) List of Available Syntactic Analysis
  - (Dynamic) List of Available Views Derivations

- **Metrics & logs**
  - Save Current Log
  - View "Last Session" Log
  - Save Available Logs
  - View Metrics
  - Save Available Metrics
  - Clear Session

- **Options**
  - Font Settings
  - Default Font

- **? (Help)**
  - QuARS Help Topics
  - QuARS Help: Getting Started
  - QuARS Help: Editors
  - QuARS help-on-Help
  - QuARS License
  - Splashscreen
  - About

---

**File**

**Load Sentences File**

Displays the Open File dialog where you can select a file from your local disk file system. At this moment the supported File Type is plain text only.
Print QuARS Output

Displays the Print dialog, where you can specify the number printed copies, etc.
Press OK to print the QuARS output window content.

Page Setup

Displays a Page Setup dialog where you can specify printing options such as margins, headers/footers and page orientation.

Quit

Closes QuARS window and exits QuARS completely.

[Back to Top]

Edit

Copy Selection

Copies the selected text in the QuARS output window to the clipboard.
You can paste the text somewhere else by using the usual menu entry Paste or keyboard shortcut Control-v.

Copy QuARS output

Copies the whole content of the QuARS output window he selected text to the clipboard.
You can paste the text somewhere else by using the usual menu entry Paste or keyboard shortcut Control-v.

Clear QuARS Output

Clears QuARS output window deleting its content.
This operation doesn’t affect the session that will not be cleaned.

Select All

Selects all text in the QuARS output window.

Deselect

Deselects all selected text in the QuARS output window.

[Back to Top]

View

Wrap/Unwrap Text

This command allows to switch in and out of word-wrapping mode on the text shown in the QuARS output window.

Increase Text Size

Makes the text shown in the QuARS output window larger.
You can also do this pressing Control + to render the text in a larger text font.

Decrease Text Size

Makes the text shown in the QuARS output window smaller.
You can also do this pressing Control - to render the text in a smaller text font.

Style

Each style item organizes the resulting sentences in the QuARS output window in different visual styles.

- on-line: a line for any resulting sentence, without marked text.
  In on-line mode the Track text and hide false-positives (Track) functions are not available.
- plain: several lines for any resulting sentences, without marked text.
  In on-line mode the Trace text and hide false-positives (Track) functions are not available.
- tagged, tagged 2, tagged 3: this are different ways to organize text with several lines (less in "tagged" more in "tagged 3") for any resulting sentences and text marked by color.
Analysis

Create New Dictionary

Pops up the Dictionary Wizard window allowing to create a new dictionary.
The same can be done exploiting Drag and Drop capability of the Dictionary Frame.
It is not allowed to add a new Dictionary for the syntactical analysis.

Switch to Lexical Analysis

Switches the QuARS interface to the Lexical mode.

Switch to Syntactic Analysis

Switches the QuARS interface to the Syntactic mode.

Switch to Views Derivation

Switches the QuARS interface to the Views Derivation mode.

(Dynamic) List of Available Lexical Analysis

This is a dynamic list of Lexical Analyses: an entry related to any single Lexical Dictionary is shown and by means of left-mouse-button-click the related analysis is performed.

(Dynamic) List of Available Syntactic Analysis

This is a dynamic list of Syntactic Analyses: an entry related to any single Syntactic Dictionary is shown and by means of left-mouse-button-click the related analysis is performed.

(Dynamic) List of Available Views Derivations

This is a dynamic list of View Derivations: an entry related to any single View Derivation Dictionary is shown and by means of left-mouse-button-click the related analysis is performed.

Metrics & Logs

Save Current Log

Pops up the "Save File" dialog window allowing to save the QuARS output as a plain text log file.

View "Last Session" Log

Pops up the "Save File" dialog window allowing to save the QuARS output as a plain text log file.

Save Available Logs

Pops up the "Save File" dialog window allowing to save still available past analysis results as a unique plain text log file.
QuARS stores the latest result of any single analysis till the "Menu/Metrics & Logs/Clear Session" entry is pressed.

View metrics

Shows saved available metrics of the latest analyses in the QuARS output window.

Save Available Metrics

Pops up the "Save File" dialog window allowing to save still available past metrics as a unique plain text log file.
QuARS stores the latest result of any single analysis till the "Menu/Metrics & Logs/Clear Session" entry is pressed.

Clear Session
Options

Font Settings
Pops up the chooser allowing to change font type and size for nearly all graphic elements of QuARS graphical interface.

Default Font
Restores default QuARS Interface font.

? (Help)

QuARS Help Topics
Pops up the main QuARS on-line help interface showing QuARS Help topics summary.

QuARS Help: Getting Started
Pops up the QuARS on-line help interface showing QuARS Help Getting Started topic.

QuARS Help: Editors
Pops up the QuARS on-line help interface showing Help topics related to the QuARS Editors.

QuARS Help-on-Help
Pops up the QuARS on-line help interface showing Help about using Help itself.

QuARS License
Show the QuARS License.

Splashscreen
Pos up the QuARS starting Splashscreen.

About
Pops up the QuARS info window.

Menu Reference Summary

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Item entry</th>
<th>Accelerator</th>
<th>Keyb. Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Load a Sentences File</td>
<td>L</td>
<td>Control+I</td>
</tr>
<tr>
<td></td>
<td>Print QuARS Output</td>
<td>P</td>
<td>Control+P</td>
</tr>
<tr>
<td>Page Setup</td>
<td>S</td>
<td>Control+g</td>
<td></td>
</tr>
<tr>
<td>Quit</td>
<td>Q</td>
<td>Control+q</td>
<td></td>
</tr>
<tr>
<td><strong>Edit</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy Selection</td>
<td>C</td>
<td>Control+C</td>
<td></td>
</tr>
<tr>
<td>Copy QuARS Output</td>
<td>O</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Clear QuARS Output</td>
<td>r</td>
<td>Control+r</td>
<td></td>
</tr>
<tr>
<td>Select All</td>
<td>S</td>
<td>Control+/</td>
<td></td>
</tr>
<tr>
<td>Deselect</td>
<td>D</td>
<td>Control+\</td>
<td></td>
</tr>
<tr>
<td><strong>View</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wrap/Unwrap Text</td>
<td>W</td>
<td>Control+w</td>
<td></td>
</tr>
<tr>
<td>Increase Text Size</td>
<td>I</td>
<td>Control+</td>
<td></td>
</tr>
<tr>
<td>Decrease Text Size</td>
<td>D</td>
<td>Control-</td>
<td></td>
</tr>
<tr>
<td>style</td>
<td>s</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create New Dictionary</td>
<td>C</td>
<td>Control+N</td>
<td></td>
</tr>
<tr>
<td>Switch to Lexical Analysis</td>
<td>L</td>
<td>Control+L</td>
<td></td>
</tr>
<tr>
<td>Switch to Syntactic Analysis</td>
<td>S</td>
<td>Control+S</td>
<td></td>
</tr>
<tr>
<td>Switch to Views Derivation</td>
<td>V</td>
<td>Control+V</td>
<td></td>
</tr>
<tr>
<td>(Dynamic) List of Available Lexical Analysis</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>(Dynamic) List of Available Syntactic Analysis</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>(Dynamic) List of Available Views Derivations</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td><strong>Metrics &amp; logs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save Current Log</td>
<td>S</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>View &quot;Last Session&quot; Log</td>
<td>L</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Save Available Logs</td>
<td>A</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>View Metrics</td>
<td>V</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Save Available Metrics</td>
<td>M</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Clear Session</td>
<td>C</td>
<td>F11</td>
<td></td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Font Settings</td>
<td>F</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Default Font</td>
<td>D</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td><strong>? (Help)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QuARS Help: Topics</td>
<td>T</td>
<td>Control+H</td>
<td></td>
</tr>
<tr>
<td>QuARS Help: Getting Started</td>
<td>G</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>QuARS Help: Editors</td>
<td>E</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>QuARS Help-on-Help</td>
<td>H</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>QuARS License</td>
<td>L</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Splashscreen</td>
<td>S</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>About</td>
<td>A</td>
<td>Control+B</td>
<td></td>
</tr>
</tbody>
</table>
1. (Load) pops up the file system browser to open a file in the editor
2. (Save) saves the output frame content to a file
3. (Copy) copy to clipboard selected (in the dictionary editor, in the sentences editor or in the output frame) text
4. (Print) prints the output frame content
5. (View Metrics) displays the available metrics by meaning of the latest analysis session
6. (Next) switches to the Lexical Analysis
7. (Previous) switches to the Syntactic Analysis
8. (Help) pops up the QuARS on-line Help
9. (Clear) clears QuARS Analysis Session
10. (Quit) quits QuARS User Interface

Keyboard Shortcuts

Main Frame QuARS Keyboard Shortcuts

- Control + - increase dictionary text size.
- Control - - decrease dictionary text size.
- Control-L: (Load) pops up the file system browser to open a file in the sentences editor
- Control-P: prints the editor content
- Control-g: pops up the page setup printer interface
- Control-q: quits QuARS User Interface
- Control-W: wraps/unwraps displayed text
- F11: clears the actual QuARS Analysis session
- Control-H: pops up the on-line QuARS Help
- Control-B: pops up the About window with info about QuARS and its authors

For a complete list of Keyboard Shortcuts related to the QuARS editing areas, please refer to:
- QuARS Sentences Editor Shortcuts
- QuARS Dictionary Editor Shortcuts

Dictionary Editor Keyboard Shortcuts

- Control + - increase dictionary text size.
- Control - - decrease dictionary text size.
- Clicking mouse button 1 positions the insertion cursor just before the character underneath the mouse cursor, sets the input focus to this widget, and clears any selection in the widget.
- Dragging with mouse button 1 strokes out a selection between the insertion cursor and the character under the mouse.
- Double-clicking with mouse button 1 selects the word under the mouse and positions the insertion cursor at the beginning of the word.
- Dragging after a double click will stroke out a selection consisting of whole words.
- Triple-clicking with mouse button 1 selects the line under the mouse and positions the insertion cursor at the beginning of the line.
- Dragging after a triple click will stroke out a selection consisting of whole lines.
- The ends of the selection can be adjusted by dragging with mouse button 1 while the Shift key is down; this will adjust the end of the selection that was nearest to the mouse cursor when button 1 was pressed.
- If the button is double-clicked before dragging then the selection will be adjusted in units of whole words; if it is triple-clicked then the selection will be adjusted in units of whole lines.
- Clicking mouse button 1 with the Control key down will reposition the insertion cursor without affecting the selection.
- If any normal printing characters are typed, they are inserted at the point of the insertion cursor.
- The view in the widget can be adjusted by dragging with mouse button 2.
- If mouse button 2 is clicked without moving the mouse, the selection is copied into the text at the position of the mouse cursor.
- The Insert key also inserts the selection, but at the position of the insertion cursor.
- If the mouse is dragged out of the widget while button 1 is pressed, the entry will automatically scroll to make more text visible (if there is more text off-screen on the side where the mouse left the window).
- The Left and Right keys move the insertion cursor one character to the left or right; they also clear any selection in the text.
- If Left or Right is typed with the Shift key down, then the insertion cursor moves and the selection is extended to include the new character.
- Control-Left and Control-Right move the insertion cursor by words.
The QuARS Editor

QuARS Editor User Interface

[QuARS HOME] [Contents] [Index]
The QuARS editor is a specialized editor oriented to the QuARS functions and to the file the formats QuARS uses. You can select text area for cut and paste and exploit the provided drag and drop functionality.

Buttons functionality:

- **Load**: pops up the file system browser to open a file in the editor
- **Reload**: loads again the file being displayed. If there isn’t actually a file loaded, it behaves like **Load**
- **Save**: saves the editor content without prompting for the name.
  - If the name of the file to save is not known (e.g. via a New command) it behaves like **Save As**
  - If the file already exists, an overwrite confirmation will be asked
- **Save As**: pops up the file system browser to choose the file name by which the textual content displayed has to be saved. You can choose an existent file or insert your own name.
  - If the file exists, it will ask an overwrite confirmation
- **Clear**: clears editor content setting it up for a new blank file
- **Print**: prints the editor content

Please refer to the Editor Menu Reference, Editor Menu Reference Summary, Buttonbox Reference and Keyboard Shortcuts for the others functionalities.

[QuARS HOME] [Contents] [Index]

QuARS Editor: Menu Reference

This menu reference describes each menu item in the QuARS Editor window menu. You can even refer to the bird-eye summary table here.

- **File**
  - New
  - Load
  - Reload
  - Save
  - Save As
  - Print
  - Page Setup
  - Quit

- **Edit**
  - Undo
  - Redo
  - Clear Undo/Redo Stack
  - Cut
  - Copy
  - Paste
  - Delete
  - Select All
  - Deselect
  - Time/Date
  - Clear

- **Search**
  - Search Window
  - Replace Window
  - Deselect
  - Help On Search Function
  - Help On Replace Function
  - Close Search/Replace Window

- **View**
  - Wrap/Unwrap Text
  - Hide/Show Sentences Numbers
  - Renumber Sentences
  - Increase Text Size
  - Decrease Text Size

- **Analysis**
File

New
Displays the Open File dialog where you can select a file from your local disk file system. At this moment the supported File Type is plain text only.

Load
Displays the Open File dialog where you can select a file from your local disk file system. At this moment the supported File Type is plain text only.

Reload
Loads again (re-loads) the original file. This is useful to verify that modifications in the shown text are rightly saved in the original file. Nevertheless if modifications have not been previously saved, they will be lost.

Save
Saves the QuARS Editor content with the current file name.

Save As
Pops up the "Save File" dialog window allowing to save the QuARS Editor content as a plain text log file allowing to choose the file name too.

Print
Displays the Print dialog, where you can specify the number printed copies, etc. Press OK to print the QuARS Editor window content.

Page Setup
Displays a Page Setup dialog where you can specify printing options such as margins, headers/footers and page orientation for the QuARS Editor content window.

Quit
Closes QuARS window and exits QuARS completely.

[Back to Top]

Edit

Undo
Cancels (undo) the last edit command.

Redo
Cancels (redo) the last Undocmand.

Clear Undo/Redo Stack
Cuts the Undo/Redo actions memory: after this it will not be possible none past undo/redo action anymore.

Cut
Cuts the selected text in the QuARSEditor window from the document and saves it on the clipboard.
You can paste the text somewhere else by means of menu entry Paste or keyboard shortcut Control-v.

Copy Selection
Copies the selected text in the QuARSEditor window to the clipboard.
You can paste the text somewhere else by means of menu entry Paste or keyboard shortcut Control-v.

Paste
Inserts memorized text on the clipboard in the QuARSEditor window in the current cursor position.

Delete
Deletes selected text in the QuARSEditor window.

Select All
Selects all text in the QuARSEditor window.
(\*Note: The left side line numbers will be not selected, neither copied nor pasted).

Deselect
Deselects all selected text in the QuARSEditor window.

Time/Date
Inserts in the current cursor position the actual time and date.

Clear
Clears the QuARSEditor window deleting its content.

Search

Search Window
Pops Up the Searchwindow.

Replace Window
Pops Up the Search and Replacewindow.

Deselect
Deselects selected text in the QuARSEditor window.

Help on Search Function
Pops up the QuARSEHelp window showing the Help on Search Window topic.

Help on Replace Function
Pops up the QuARSEHelp window showing the Help on Replace Window topic.

Close Search/Replace Window
Closes the Search or Replacewindow.
View

Wrap/Unwrap Text
This command allows to switch in and out of word-wrapping mode on the text shown in the QuARSEditor window.

Hide/Show Sentences Number
Shows/Hides left sentences numbering.

Renumber Sentences
Re-numbers the shown sentences by means of a left-sided column of blue numbers.

Increase Text Size
Makes the text shown in the QuARSEditor window larger.
You can also do this pressing Control + to render the text in a larger text font.

Decrease Text Size
Makes the text shown in the QuARSEditor window smaller.
You can also do this pressing Control – to render the text in a smaller text font.

[Back to Top]

Analysis

Create New Dictionary
Pops up the Dictionary Wizard window allowing to create a new dictionary.
The same can be done exploiting Drag and Drop capability of the Dictionary Frame.
It is not allowed to add a new Dictionary for the syntactical analysis.

Switch to Lexical Analysis
Switches the QuARSE interface to the Lexical mode.

Switch to Syntactic Analysis
Switches the QuARSE interface to the Syntactic mode.

Switch to Views Derivation
Switches the QuARSE interface to the Views Derivation mode.

(Dynamic) List of Available Lexical Analysis
This is a dynamic list of Lexical Analyses: an entry related to any single Lexical Dictionary is shown and by means of left-mouse-button-click the related analysis is performed.

(Dynamic) List of Available Syntactic Analysis
This is a dynamic list of Syntactic Analyses: an entry related to any single Syntactic Dictionary is shown and by means of left-mouse-button-click the related analysis is performed.

(Dynamic) List of Available Views Derivations
This is a dynamic list of View Derivations: an entry related to any single View Derivation Dictionary is shown and by means of left-mouse-button-click the related analysis is performed.

[Back to Top]

Help

QuARSE Editor Help: Topics
Pops up the main QuARS on-line help interface showing QuARS Help topics summary about the QuARS Editor.

**QuARS Editor Help: Search Function**

Pops up the QuARS Help window showing the Help on the Search Window topic.

**QuARS Editor Help: Replace Function**

Pops up the QuARS Help window showing the Help on the Replace Window topic.

---

[QuARS HOME] [Contents] [Index]

### QuARS Editor: Menu Reference Summary

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Item entry</th>
<th>Accelerator</th>
<th>Keyb. Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File</strong></td>
<td>New</td>
<td>N</td>
<td>Control+N</td>
</tr>
<tr>
<td></td>
<td>Load</td>
<td>L</td>
<td>Control+L</td>
</tr>
<tr>
<td></td>
<td>Reload</td>
<td>R</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Save</td>
<td>S</td>
<td>Control+s</td>
</tr>
<tr>
<td></td>
<td>Save As</td>
<td>A</td>
<td>F12</td>
</tr>
<tr>
<td></td>
<td>Print</td>
<td>P</td>
<td>Control+P</td>
</tr>
<tr>
<td></td>
<td>Page Setup</td>
<td>g</td>
<td>Control+g</td>
</tr>
<tr>
<td></td>
<td>Quit</td>
<td>Q</td>
<td>Control+q</td>
</tr>
<tr>
<td><strong>Edit</strong></td>
<td>Undo</td>
<td>U</td>
<td>Control+z</td>
</tr>
<tr>
<td></td>
<td>Redo</td>
<td>R</td>
<td>Control+y</td>
</tr>
<tr>
<td></td>
<td>Clear Undo/Redo Stack</td>
<td>I</td>
<td>F10</td>
</tr>
<tr>
<td></td>
<td>Cut</td>
<td>t</td>
<td>Control+x</td>
</tr>
<tr>
<td></td>
<td>Copy</td>
<td>C</td>
<td>Control+c</td>
</tr>
<tr>
<td></td>
<td>Paste</td>
<td>P</td>
<td>Control+v</td>
</tr>
<tr>
<td></td>
<td>Delete</td>
<td>D</td>
<td>DEL</td>
</tr>
<tr>
<td></td>
<td>Select All</td>
<td>A</td>
<td>Control+/</td>
</tr>
<tr>
<td></td>
<td>Deselect</td>
<td>s</td>
<td>Control+/\</td>
</tr>
<tr>
<td></td>
<td>Time/Date</td>
<td>i</td>
<td>Control+T</td>
</tr>
<tr>
<td></td>
<td>Clear</td>
<td>e</td>
<td>Control+C</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>Search Window</td>
<td>S</td>
<td>Control+f</td>
</tr>
<tr>
<td></td>
<td>Replace Window</td>
<td>R</td>
<td>F8</td>
</tr>
<tr>
<td></td>
<td>Deselect</td>
<td>D</td>
<td>Control+D</td>
</tr>
<tr>
<td></td>
<td>Help On Search Function</td>
<td>e</td>
<td>Control+H</td>
</tr>
<tr>
<td></td>
<td>Help On Replace Function</td>
<td>p</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Close Search/Replace Window</td>
<td>C</td>
<td>Esc</td>
</tr>
<tr>
<td><strong>View</strong></td>
<td>Wrap/Unwrap Text</td>
<td>W</td>
<td>Control+w</td>
</tr>
<tr>
<td></td>
<td>Hide/Show Sentences Numbers</td>
<td>H</td>
<td>F9</td>
</tr>
<tr>
<td></td>
<td>Renumbe Sentences</td>
<td>R</td>
<td>Control+r</td>
</tr>
<tr>
<td></td>
<td>Increase Text Size</td>
<td>I</td>
<td>Control +</td>
</tr>
<tr>
<td></td>
<td>Decrease Text Size</td>
<td>D</td>
<td>Control -</td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td>Create New Dictionary</td>
<td>C</td>
<td>Control+N</td>
</tr>
<tr>
<td></td>
<td>Switch to Lexical Analysis</td>
<td>L</td>
<td>Control+L</td>
</tr>
<tr>
<td></td>
<td>Switch to Syntactic Analysis</td>
<td>S</td>
<td>Control+S</td>
</tr>
<tr>
<td></td>
<td>Switch to Views Derivation</td>
<td>V</td>
<td>Control+V</td>
</tr>
<tr>
<td>Help</td>
<td>QuARS Editor Help: Topics</td>
<td>Q</td>
<td>Control+H</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------</td>
<td>---</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>QuARS Editor Help: Search Function</td>
<td>S</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>QuARS Editor Help: Replace Function</td>
<td>R</td>
<td>--</td>
</tr>
</tbody>
</table>

**QuARS Editor: Keyboard Shortcuts**

Here is a complete list of bindings and keyboard short-key that resume the functionalities provided by the QuARS editor.

**Main Editor Shortcuts**

- **Control +** increase dictionary text size.
- **Control -** decrease dictionary text size.
- **Control-N**: (New) sets a new file in the editor
- **Control-I**: (Load) pops up the file system browser to open a file in the editor
- **Control-S**: (Save) saves the editor content, if the name of the file to save is not known (via a New command) it behaves as Control-I
- **F12**: (Save As) saves the editor content popping up the file system browser allowing to choose a different file name
- **Control-P**: prints the editor content
- **Control-g**: pops up the page setup printer interface
- **Control-q**: quits QuARS User Interface
- **Control-z**: undo latest change
- **Control-y**: redo latest change
- **F10**: clear Undo/Redo stack
- **Control-x**: cuts selected text
- **Control-c**: copies selected text
- **Control-v**: pastes selected text
- **Del**: deletes selected text
- **Control-f**: selects all the text
- **Control-y**: de-selects all the text
- **Control-t**: inserts time and date
- **Control-C**: clears editor content setting it up for a new blank file
- **Control-f**: pops up the QuARS search window
- **F8**: pops up the QuARS Search and Replace window
- **Control-D**: clears the editor content from all the selections
- **Control-H**: pops up the QuARS on-line Help
- **Esc**: closes Search/Replace window
- **Control-w**: wraps/unwraps displayed text
- **F9**: hides/shows sentences number
- **Control-r**: renumbers sentences lines
- **Control-N**: pops up the QuARS Dictionary Wizard
- **Control-L**: switches to the Lexical Analysis
- **Control-S**: switches to the Syntactic Analysis
- **Control-V**: switches to the Views Derivation
- **Control-H**: pops up the on-line QuARS Help

**Other Editor Shortcuts**

- **Clicking mouse button 1** positions the insertion cursor just before the character underneath the mouse cursor, sets the input focus to this widget, and clears any selection in the widget.
- **Dragging with mouse button 1** strokes out a selection between the insertion cursor and the character under the mouse.
- **Double-clicking with mouse button 1** selects the word under the mouse and positions the insertion cursor at the beginning of the word.
- **Dragging after a double click** will stroke out a selection consisting of whole words.
- **Triple-clicking with mouse button 1** selects the line under the mouse and positions the insertion cursor at the beginning of the line.
- The ends of the selection can be adjusted by **dragging with mouse button 1 while the Shift key is down**; this will adjust the end of the selection that was nearest to the mouse cursor when button 1 was pressed.
- If the button is double-clicked before dragging then the selection will be adjusted in units of whole words; if it is triple-clicked then the selection will be adjusted in units of whole lines.
- **Clicking mouse button 1 with the Control key down** will reposition the insertion cursor without affecting the selection.
- If any normal printing characters are typed, they are inserted at the point of the insertion cursor.
- The view in the widget can be adjusted by **dragging with mouse button 2**.
If mouse button 2 is clicked without moving the mouse, the selection is copied into the text at the position of the mouse cursor.
The Insert key also inserts the selection, but at the position of the insertion cursor.
If the mouse is dragged out of the widget while button 1 is pressed, the entry will automatically scroll to make more text visible (if there is more text off-screen on the side where the mouse left the window).
The Left and Right keys move the insertion cursor one character to the left or right; they also clear any selection in the text.
If Left or Right is typed with the Shift key down, then the insertion cursor moves and the selection is extended to include the new character.
Control-Left and Control-Right move the insertion cursor by words.
Control-Shift-Left and Control-Shift-Right move the insertion cursor by words and also extend the selection.
Control-b and Control-f behave the same as Left and Right, respectively.
The Up and Down keys move the insertion cursor one line up or down and clear any selection in the text.
If Up or Right is typed with the Shift key down, then the insertion cursor moves and the selection is extended to include the new character.
Control-Up and Control-Down move the insertion cursor by paragraphs (groups of lines separated by blank lines).
Control-Shift-Up and Control-Shift-Down move the insertion cursor by paragraphs and also extend the selection.
Control-p and Control-n behave the same as Up and Down, respectively.
The Next and Prior keys move the insertion cursor forward or backwards by one screenful and clear any selection in the text.
If the Shift key is held down while Next or Prior is typed, then the selection is extended to include the new character.
Control-v moves the view down one screenful without moving the insertion cursor or adjusting the selection.
Control-Next and Control-Prior scroll the view right or left by one page without moving the insertion cursor or affecting the selection.
Home and Control-a move the insertion cursor to the beginning of its line and clear any selection in the widget.
Shift-Home moves the insertion cursor to the beginning of the line and also extends the selection to that point.
End and Control-e move the insertion cursor to the end of the line and clear any selection in the widget.
Shift-End moves the cursor to the end of the line and extends the selection to that point.
Control-Home move the insertion cursor to the beginning of the text and clear any selection in the widget.
Control-Shift-Home moves the insertion cursor to the beginning of the text and also extends the selection to that point.
Control-End move the insertion cursor to the end of the text and clear any selection in the widget.
Control-Shift-End moves the cursor to the end of the text and extends the selection to that point.
The Select key and Control-Space set the selection anchor to the position of the insertion cursor. They don't affect the current selection.
Shift-Select and Control-Shift-Space adjust the selection to the current position of the insertion cursor, selecting from the anchor to the insertion cursor if there was not any selection previously.
Control-\ selects the entire contents of the widget.
Control-\ clears any selection in the widget.
The F16 key copies the selection in the widget to the clipboard, if there is a selection.
The F18 key inserts the contents of the clipboard at the position of the insertion cursor.
The Delete key deletes the selection, if there is one in the widget. If there is no selection, it deletes the character to the right of the insertion cursor.
Backspace and Control-h delete the selection, if there is one in the widget. If there is no selection, they delete the character to the left of the insertion cursor.
Control-d deletes the character to the right of the insertion cursor.
Control-k deletes from the insertion cursor to the end of its line.
If the insertion cursor is already at the end of a line, then Control-k deletes the newline character.
Control-x deletes whatever is selected in the text widget.
Control-t reverses the order of the two characters to the right of the insertion cursor.
Control-o opens a new line by inserting a newline character in front of the insertion cursor without moving the insertion cursor.

1. (New) sets a new file in the editor
2. (Load) pops up the file system browser to open a file in the editor
3. (Reload) loads again the file being displayed
4. (Save) saves the editor content, if the name of the file to save is not known (via a New command) it behaves as Control-i
5. (Save As) saves the editor content popping up the file system browser allowing to choose a different file name
6. (Print) prints the editor content
7. (Cut) cuts selected text
8. (Copy) copies selected text
9. (Paste) pastes selected text
10. (Delete) deletes selected text
11. (Undo) undo latest change
QuARS Editor: Search & Replace Function

Use this function to find and replace each occurrence of a combination of any characters, including uppercase and lowercase characters, whole words, or parts of words, or regular expression.

- **"Search" box:**
  - this a "combo box" memorizing latest search items you inserted: use this line to specify a text search string.

- **"Replace" box:**
  - this a "combo box" memorizing latest replace items you inserted: use this line to specify a text replace string.

- **Buttons**
  - **Find Next**: finds next occurrence of the searching string and select it in a green background color.
  - **Mark All**: marks all occurrences of the searching string by means of a green selection.
  - **Replace**: replaces the current.
  - **Replace Next**: deselect selected text in the QuARS Editor window.
  - **Replace All**: deselect selected text in the QuARS Editor window.
  - **Deselect**: deselect selected text in the QuARS Editor window.
  - **Close**: closes the Search window.
  - ?: pops up the QuARS Help showing this item.

- **Conditions**
  - **No Case**: this is the default
  - **Exact**: use exact matching. The characters in the matching range must be identical to those in pattern.
  - **Regular Expression**: treat pattern as a regular expression and match it against the text using the rules for regular expressions (see the Regular Expression item for details).

- **Direction**:
  - **Backwards**: the search will proceed backward through the text.
  - **Forwards**: the search will proceed forward through the text.
  - This is the default.

- **Go to the Search Function**:
  - click this button to switch to the Search window
Use this function to find each occurrence of a combination of any characters, including uppercase and lowercase characters, whole words, or parts of words, or regular expression.

- **“Search” box:**
  - this a "combo box" memorizing latest search items you inserted: use this line to specify a text search string.

- **Buttons**
  - **Find Next:** finds next occurrence of the searching string and select it in a green background color.
  - **Mark All:** marks all occurrences of the searching string by means of a green selection
  - **Deselect:** deselect selected text in the QuARS Editor window.
  - **Close:** closes the Search window.
  - ?: pops up the QuARS Help showing this item.

- **Conditions**
  - **No Case:** this is the default
  - **Exact:** use exact matching. The characters in the matching range must be identical to those in pattern.
  - **Regular Expression:** treat pattern as a regular expression and match it against the text using the rules for regular expressions (see the Regular Expression item for details).

- **Direction**
  - **Backwards:** the search will proceed backward through the text.
  - **Forwards:** the search will proceed forward through the text.
  This is the default.

- **Go to the Replace Function:**
  - [click this button to switch to the Replace window](https://www.quars.com/)

**HOW TOs**

_**HOW TOs**_

- **How To Convert an input file in the suitable file format**
- **How To create a new Dictionary**

**How To create a new Dictionary**

To create a new Dictionary you can:

- (1a) drag & drop the text file over the *QuARS Dictionary Area* and skip next points.
The content of the dictionary will be uploaded from the dragged file. 

⚠️ There is no control on the type of the file in that way uploaded, so be sure that its content is **pure plain ascii text**.

- **(1b)** press the "**New**" button in the "**QuARS Dictionary Area**"  
- **(2)** The "**QuARS Dictionary Wizard**" will pop up  
- **(3)** (mandatory) select the kind ("Lexical" or "View Derivation") of the new dictionary  
- **(4)** (mandatory) insert the new dictionary name  
- **(5a)** (optional) you can upload a text file from the file system using the wizard button "Upload a text file" (a file manager will pop up) or choose to fill manually later the dictionary by means of the **QuARS Dictionary Editor**  
- **(5b)** (optional) at any time you can update the content of any dictionary and save it using the **QuARS Dictionary Area** button "Save"  
- **(6)** press the "**Create Dict**" button or the "Enter" key: if a dictionary with the chosen name still exists an error message is shown and the name can be changed  
- **(7)** the new Dictionary will appear in the **QuARS Dictionary Area**  
- **(8)** if you need to create another new Dictionary repeat points 3x-7 or 1a  
- **(9)** close the **QuARS Dictionary Wizard** using the "Close" button
Dictionaries name format and limitations:

- at this moment, the supported file type for the QuARS Dictionaries is only plain text file.
- So, no `.doc`, `.rtf`, `.html`, `.pdf` or whatever file types except for pure plain ascii text files are allowed.
- See here How To Convert an input file in the suitable file format.
- There is no control on the type of the file uploaded, so be sure that its content is pure plain ascii text
- Moreover the name of a new dictionary has to respect a some rules:
  - can be at most 15 chars long (excluding extension)
  - spaces are NOT allowed
  - the QuARS Dictionary Wizard does not allow to insert special characters or spaces
  - if the name of a text file, dropped to create a new dictionary, contains spaces they are automatically converted to underscores
  - the QuARS Dictionary Wizard does not allow to more than 15 characters in the new Dictionary name entry
  - if the name of a text file, dropped to create a new dictionary, is longer than 15 chars, the last chars are automatically cropped to fit the legal 15-chars length

How To Convert an input file in the suitable file format

If the format of the input file is not plain text, it has to be converted.
In the following the procedure to be followed for the most common commercial text formats:

- **MS Word** (.doc):
  - **Export the entire file**
    - Select from the menubar "File" --> "Save as"
    - Select "text file (.txt)" as saving file format
    - Select where saving the file
    - Click the "Save" button
  - **Export an excerpt of the file**
    - Select the text of interest
    - Copy the selected text
    - Past the copied text in a plain text file

- **Adobe Acrobat** (.pdf):
  - **Export the entire file**
    - Select from the menubar "File" --> "Save as Text"
    - Select where saving the file
    - Click the "Save" button
  - **Export an excerpt of the file**
    - Select the "Select Text" tool button in the Adobe Acrobat Reader buttonbox
    - Select the text of interest
    - Copy the selected text
    - Past the copied text in a plain text file
    - Deselect the "Select Text" tool button to come back to the normal Acrobat Reader view

- **HTML** (.htm, .html):
  - **Export the entire file**
    - Select from the menubar "File" --> "Save as"
    - Select "text file (.txt)" as saving file format
    - Select where saving the file
    - Click the "Save" button
  - **Export an excerpt of the file**
    - Select the text of interest
    - Copy the selected text
    - Past the copied text in a plain text file
QuARS Help-On-Help: The User Interface (usage)

The main Help window provides a user-friendly view of the help text, with standard controls (Back, Forward, Index, Contents and etc.). The main window includes a context menu, which is invoked by clicking the right mouse button.

**Keyboard shortcuts:**
- Backspace or Alt-Left or Alt-F5
  Back to previous topic.
- Alt-Right
  Forward to next topic.
- Escape
  Stop loading current topic.
- Control-F
  Find text in current topic.
- Control-S
  Find text in all topics.
- Control-R
  Refresh screen.
- Control-L
  Reload help file(s).
  **Warning:** All history information will be lost.
- Control-plus
  Increase font size.
- Control-minus
  Decrease font size.
- Control-O
  Open another help file. "Open file" dialog will appear.

**Screenshot of "HelpSystem" window:**
The Right-Click menu
The "Right menu button" provides a menu with several useful functions:

- **Back**
  - Back to previous topic.
- **Forward**
  - Forward to next topic.
- **Stop**
  - Stops loading current topic.
- **Find**
  - Pops up the "Find window": the specified item will be sought through the displayed page.
  - **Search in topics**...
    - Pops up the "Find's window": the specified item will be sought all over the pages.
- **Open File**
  - Open another help file. "Open file" dialog will appear.
- **Decrease font size**
  - Decrease displayed font size.
- **Increase font size**
  - Increase displayed font size.
- **Refresh**
  - Refresh screen.
- **Reload**
  - Reload current help file(s).

⚠️ All history information will be lost.
About QuARS Help System

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
Dictionaries name format and limitations

To create a new Dictionary (as seen here) you can "drag and drop" a file over the "QuARS Dictionary Area"

or push the "Upload a File" button in the "QuARS Dictionary Wizard".
In both cases, at this moment, the supported file type for these actions is only plain text file.
So, no .doc, .rtf, .html, .pdf or whatever file types except for simple ascii text files are allowed.
And no control Moreover the name of a new dictionary can be at most 15 chars long and are allowed only numbers, letters (both uppercase and lowercase) and the underscore.
Neither spaces are allowed: if the name of a text file, dropped to create a new dictionary, contains spaces they are automatically converted with underscores.

Indicators Explication

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vagueness</td>
<td>The occurrence of this Indicator is due to the existence of Vagueness-revealing wordings as for example: clear, easy, strong, good, bad, useful, significant, adequate, recent,...</td>
</tr>
<tr>
<td>Subjectivity</td>
<td>The occurrence of this Indicator is due to the existence of Subjectivity-revealing wordings as for example: similar, similarly, having in mind, take into account, as [adjective] as possible,...</td>
</tr>
<tr>
<td>Optionality</td>
<td>The occurrence of this Indicator is due to the existence of Optionality-revealing words as for example: possibly, eventually, if case, if possible, if appropriate, if needed,...</td>
</tr>
<tr>
<td>Implicit</td>
<td>The occurrence of this Indicator is determined by the existence of: Implicit Subject or Object: sentences having the subject or object complements expressed by means of: Demonstrative adjective (this, these, that, those) or Pronouns (it, they...) [examples: those shall run in a safe mode - the system shall display it in the main screen] Implicit Determiner: terms having the determiner expressed by a demonstrative adjective (this, these, that, those) or implicit adjective (as for example previous, next, following, last...) or preposition (as for example above, below...) [example: the system shall produce the previous results]</td>
</tr>
<tr>
<td>Weakness</td>
<td>The verbs that determine the occurrence of this indicator are the Weak verbs: could, might, may.</td>
</tr>
</tbody>
</table>
Under-specification | This indicator occurs when words needing to be instantiated are detected. [e.g. flow instead of data flow, control flow - access instead of write access, remote access, authorized access - testing instead of functional testing, structural testing, unit testing, etc.]

Multiplicity | This indicator occurs when a multiple sentences is found. A multiple sentence has more than one subject [e.g. the mean time to remove a faulty board and the time to restart shall be less than 30 minutes] or more than one main verb. [e.g.: the mean time needed to remove a faulty board and restore service shall be less than 30 minutes]

Readability | It correspond to the actual value of the Coleman-Liau formula.

[QuARS HOME] [Contents] [Index]

Drag & Drop

In some areas of the QuARS GUI is enabled the "drag and drop" function/capability. These areas are:

- **The Dictionary Editor area**: you can drag&drop plain text file onto this area to add a Dictionary to the QuARS Dictionary Set. This capability is enabled only for the Lexical type of analysis and the Views Derivation function.
- **The Sentences Editor Area**: you can drag&drop a plain text file onto this area to load the file to analyze.

The "Coleman-Liau" readability index

The Coleman-Liau Formula usually gives a lower grade than Kincaid, ARI and Flesch when applied to technical documents.

"Coleman-Liau formula" = 5.89 * letters over words - 0.3 * sentences over \( \left\{ \frac{100 \times \text{words}}{\text{sentences}} \right\} - 15.8

- **Reference**:

[QuARS HOME] [Contents] [Index]

Dictionary Wizard

The QuARS Dictionary Wizard allows to create quickly a correct new QuARS dictionary that will be inserted and displayed at once among the
others. Its content can be filled in two ways:
- by uploading - before having created it - a text file clicking on the **Upload a text File** button
- filling it by hands - after having created it - by means of a cut & paste action or typing it word by word

Otherwise you can drag & drop a plain text onto the **QuARS Dictionary Area**.

Follow the "four simple steps" instructions

Choose the new Dictionary name

Correctness check

Choose the content of the new Dictionary (optional)

A detailed "How To" create a new dictionary can be found here

[QuARS HOME] [Contents] [Index]

**QuARS Editor: Regular Expressions in Search/Replace Window**

In the **QuARS Editor** Search and Replace windows you can check the checkbox "Regular Expression" allowing the use of TCL/TK regex in the search/replace functions.

Here is a summary of regular expression grammar and examples of use excerpts from the TCL on-line manual

A regular expression describes strings of characters. It's a pattern that matches certain strings and doesn't match others.

**DIFFERENT FLAVORS OF REs**

Regular expressions ("REs"), as defined by POSIX, come in two flavors: **extended REs ("ERE")s** and **basic REs ("BRE")s**. EREs are roughly those of the traditional egrep, while BREs are roughly those of the traditional ed. This implementation adds a third flavor, **advanced REs ("ARE")s**, basically EREs with some significant extensions.

This manual page primarily describes AREs. BREs mostly exist for backward compatibility in some old programs; they will be discussed at the end. POSIX EREs are almost an exact subset of AREs. Features of AREs that are not present in EREs will be indicated.

**REGULAR EXPRESSION SYNTAX**

Tcl regular expressions are implemented using the package written by Henry Spencer, based on the 1003.2 spec and some (not quite all) of the Perl5 extensions (thanks, Henry!). Much of the description of regular expressions below is copied verbatim from his manual entry.

An **ARE** is one or more **branches**, separated by `|`, matching anything that matches any of the branches.

A branch is zero or more **constraints** or **quantified atoms**, concatenated. It matches a match for the first, followed by a match for the second, etc; an empty branch matches the empty string.

A quantified atom is an **atom** possibly followed by a single **quantifier**. Without a quantifier, it matches a match for the atom. The quantifiers, and what a so-quantified atom matches, are:
*  a sequence of 0 or more matches of the atom
+
?  a sequence of 1 or more matches of the atom
\{m\}  a sequence of exactly m matches of the atom
\{m,n\}  a sequence of m or more matches of the atom
\{m,n\}  a sequence of m through n (inclusive) matches of the atom; m may not exceed n
*?  +? \{m\}?  \{m,n\}?  non-greedy quantifiers, which match the same possibilities, but prefer the smallest number rather than the largest number of matches (see MATCHING)

The forms using { and } are known as bounds. The numbers m and n are unsigned decimal integers with permissible values from 0 to 255 inclusive.

An atom is one of:

(re)  (where re is any regular expression) matches a match for re, with the match noted for possible reporting
(?!re)  as previous, but does no reporting (a \texttt{``non-capturing''} set of parentheses)
()  matches an empty string, noted for possible reporting
(?!:)

[char]
  a \textit{bracket expression}, matching any one of the chars (see BRACKET EXPRESSIONS for more detail)
  .  matches any single character
  \k  (where k is a non-alphanumeric character) matches that character taken as an ordinary character, e.g. \texttt{"\k"} matches a backslash character
  \c  where c is alphanumeric (possibly followed by other characters), an \textit{escape} (AREs only), see ESCAPES below
  \{  when followed by a character other than a digit, matches the left-brace character \texttt{"\{"}; when followed by a digit, it is the beginning of a \textit{bound} (see above)
  x  where x is a single character with no other significance, matches that character.

A \textit{constraint} matches an empty string when specific conditions are met. A constraint may not be followed by a quantifier. The simple constraints are as follows; some more constraints are described later, under ESCAPES.

^  matches at the beginning of a line
$  matches at the end of a line
(?!re)  \textit{positive lookahead} (AREs only), matches at any point where a substring matching re begins
(?!re)  \textit{negative lookahead} (AREs only), matches at any point where no substring matching re begins

The lookahead constraints may not contain back references (see later), and all parentheses within them are considered non-capturing.

An RE may not end with `\texttt{"\:\}';

\textbf{BRACKET EXPRESSIONS}

A \textit{bracket expression} is a list of characters enclosed in `[ ]`. It normally matches any single character from the list (but see below). If the list begins with `\texttt{"^"}`; it matches any single character (but see below) not from the rest of the list.

If two characters in the list are separated by `\texttt{"-"}, this is shorthand for the full \textit{range} of characters between those two (inclusive) in the collating sequence, e.g. `[0-9]` in ASCII matches any decimal digit. Two ranges may not share an endpoint, so e.g. `\texttt{abc-e}` is illegal. Ranges are very collating-sequence-dependent, and portable programs should avoid relying on them.

To include a literal `\texttt{]} or `\texttt{[} in the list, the simplest method is to enclose it in `[ ` and `]` to make it a collating element (see below). Alternatively, make it the first character (following a possible `\texttt{"^"}`), or (AREs only) precede it with `\texttt{"\:\}`. Alternatively, for `\texttt{"-"}, make it the last character, or the second endpoint of a range. To use a literal `\texttt{a} as the first endpoint of a range, make it a collating element (or (AREs only) precede it with `\texttt{"\:\}'.

With the exception of these, some combinations using `[ (see next paragraphs), and escapes, all other special characters lose their special significance within a bracket expression.

Within a bracket expression, a collating element (a character, a multi-character sequence that collates as if it were a single character, or a collating-sequence name for either) enclosed in `[ ` and `]` stands for the sequence of characters of that collating element. The sequence is a single element of the bracket expression's list. A bracket expression in a locale that has multi-character collating elements can thus match more than one character. So (insidiously), a bracket expression that starts with `\texttt{"^"} can match multi-character collating elements even if none of them appear in the bracket expression! (Note: Tcl currently has no multi-character collating elements. This information is only for illustration.)

47
For example, assume the collating sequence includes a `ch` multi-character collating element. Then the RE `[[.ch.]]*c` (zero or more `ch`'s followed by `c`) matches the first five characters of `\chchcc`. Also, the RE `[^c]b` matches all of `\chb` (because `[^c]` matches the multi-character `ch`).

Within a bracket expression, a collating element enclosed in `[` and `]` is an equivalence class, standing for the sequences of characters of all collating elements equivalent to that one, including itself. (If there are no other equivalent collating elements, the treatment is as if the enclosing delimiters were `[^` and `]`.) For example, if `a` and `b` are the members of an equivalence class, then `[[a=a]]`, `[[a=a-]]`, and `[[a]]` are all synonymous. An equivalence class may not be an endpoint of a range. (Note: Tcl currently implements only the Unicode locale. It doesn't define any equivalence classes. The examples above are just illustrations.)

Within a bracket expression, the name of a character class enclosed in [: and :] stands for the list of all characters (not all collating elements!) belonging to that class. Standard character classes are:

```
alpha A letter.
upper An upper-case letter.
lower A lower-case letter.
digit A decimal digit.
dxdigit A hexadecimal digit.
alnum An alphanumeric (letter or digit).
print An alphanumeric (same as alnum).
blank A space or tab character.
space A character producing
white space in displayed text.
punct A punctuation character.
graph A character with a visible representation.
cntrl A control character.
```

A locale may provide others. (Note that the current Tcl implementation has only one locale: the Unicode locale.) A character class may not be used as an endpoint of a range.

There are two special cases of bracket expressions: the bracket expressions `[[<:]]` and `[[>:]]` are constraints, matching empty strings at the beginning and end of a word respectively. A word is defined as a sequence of word characters that is neither preceded nor followed by word characters. A word character is an `alnum` character or an underscore (`_`). These special bracket expressions are deprecated; users of AREs should use constraint escapes instead (see below).

ESCAPES

Escapes (AREs only), which begin with a `\` followed by an alphanumeric character, come in several varieties: character entry, class shorthands, constraint escapes, and back references. A `\` followed by an alphanumeric character but not constituting a valid escape is illegal in AREs. In EREs, there are no escapes: outside a bracket expression, a `\` followed by an alphanumeric character merely stands for that character as an ordinary character, and inside a bracket expression, `\` is an ordinary character. (The latter is the one actual incompatibility between EREs and AREs.)

Character-entry escapes (AREs only) exist to make it easier to specify non-printing and otherwise inconvenient characters in REs:

```
\a  alert (bell) character, as in C
\b  backspace, as in C
\f  formfeed, as in C
\n  newline, as in C
\r  carriage return, as in C
\t  horizontal tab, as in C
\u{wxyz} (where wxyz is exactly four hexadecimal digits) the Unicode character \u{wxyz} in the local byte ordering
\U{stuvwxyz} (where stuvwxyz is exactly eight hexadecimal digits) reserved for a somewhat-hypothetical Unicode extension to 32 bits
\v  vertical tab, as in C
\x{hh} (where hhh is any sequence of hexadecimal digits) the character whose hexadecimal value is \x{hh} (a single character no matter how many hexadecimal digits are used).
\0  the character whose value is 0
\x{xy} (where xy is exactly two octal digits, and is not a back reference (see below)) the character whose octal value is \x{xy}
\xyz (where xyz is exactly three octal digits, and is not a back reference (see below)) the character whose octal value is \xyz
```

Hexadecimal digits are `\0`-`\9`, `\a`-`\f`, and `\A`-`\F`. Octal digits are `\0`-`\7`.

The character-entry escapes are always taken as ordinary characters. For example, `\135` is ] in ASCII, but `\135` does not terminate a bracket expression. Beware, however, that some applications (e.g., C compilers) interpret such sequences themselves before the regular-expression package gets to see them, which may require doubling (quadrupling, etc.) the `\`.

Class-shorthand escapes (AREs only) provide shorthands for certain commonly-used character classes:

```
\d  [[digit:]]
\s  [[space:]]
```
Within bracket expressions, "`\d`, `\s`, and `\w` lose their outer brackets, and `\D`, `\S`, and `\W` are illegal. (So, for example, `[a-c\d]` is equivalent to `[a-c[:digit:]]`. Also, `[a-c\D]`, which is equivalent to `[a-c[:digit:]]`, is illegal.)

A constraint escape (AREs only) is a constraint, matching the empty string if specific conditions are met, written as an escape:

`\A` matches only at the beginning of the string (see MATCHING, below, for how this differs from `\^`)

`\m` matches only at the beginning of a word

`\M` matches only at the end of a word

`\V` matches only at the beginning or end of a word

`\v` matches only at a point that is not the beginning or end of a word

`\Z` matches only at the end of the string (see MATCHING, below, for how this differs from `\$`)

`\m` (where m is a nonzero digit) a back reference, see below

`\mn` (where m is a nonzero digit, and nn is some more digits, and the decimal value mnn is not greater than the number of closing capturing parentheses seen so far) a back reference, see below

A word is defined as in the specification of `[[:<:]]` and `[[:>:]]` above. Constraint escapes are illegal within bracket expressions.

A back reference (AREs only) matches the same string matched by the parenthesized subexpression specified by the number, so that (e.g.) `\1` matches `bb` or `cc` but not `bc`. The subexpression must entirely precede the back reference in the RE. Subexpressions are numbered in the order of their leading parentheses. Non-capturing parentheses do not define subexpressions.

There is an inherent historical ambiguity between octal character-entry escapes and back references, which is resolved by heuristics, as hinted at above. A leading zero always indicates an octal escape. A single non-zero digit, not followed by another digit, is always taken as a back reference. A multi-digit sequence not starting with a zero is taken as a back reference if it comes after a suitable subexpression (i.e. the number is in the legal range for a back reference), and otherwise is taken as octal.

**METASYNTAX**

In addition to the main syntax described above, there are some special forms and miscellaneous syntactic facilities available.

Normally the flavor of RE being used is specified by application-dependent means. However, this can be overridden by a director. If an RE of any flavor begins with `\***`, the rest of the RE is an ARE. If an RE of any flavor begins with `*`, the rest of the RE is taken to be a literal string, with all characters considered ordinary characters.

An ARE may begin with embedded options: a sequence `{xyz}` (where `xyz` is one or more alphabetic characters) specifies options affecting the rest of the RE. These supplement, and can override, any options specified by the application. The available option letters are:

- `b` rest of RE is a BRE
- `c` case-sensitive matching (usual default)
- `e` rest of RE is an ERE
- `i` case-insensitive matching (see MATCHING, below)
- `m` historical synonym for `n`
- `n` newline-sensitive matching (see MATCHING, below)
- `p` partial newline-sensitive matching (see MATCHING, below)
- `q` rest of RE is a literal (`\` quoted") string, all ordinary characters
- `s` non-newline-sensitive matching (usual default)
- `t` tight syntax (usual default; see below)
- `w` inverse partial newline-sensitive (`\` weird") matching (see MATCHING, below)
- `x` expanded syntax (see below)

Embedded options take effect at the ) terminating the sequence. They are available only at the start of an ARE, and may not be used later within it.
In addition to the usual (tight) RE syntax, in which all characters are significant, there is an expanded syntax, available in all flavors of RE with the -expanded switch, or in AREs with the embedded x option. In the expanded syntax, white-space characters are ignored and all characters between a # and the following newline (or the end of the RE) are ignored, permitting paragraphing and commenting a complex RE. There are three exceptions to that basic rule:

a white-space character or `#` preceded by `\` is retained

white space or `#` within a bracket expression is retained

white space and comments are illegal within multi-character symbols like the ARE `(?!` or the BRE `\(`

Expanded-syntax white-space characters are blank, tab, newline, and any character that belongs to the space character class.

Finally, in an ARE, outside bracket expressions, the sequence `(?#ttt)` (where ttt is any text not containing a ` `) is a comment, completely ignored. Again, this is not allowed between the characters of multi-character symbols like `(?!)`. Such comments are more a historical artifact than a useful facility, and their use is deprecated; use the expanded syntax instead.

None of these metasyntax extensions is available if the application (or an initial ***= director) has specified that the user’s input be treated as a literal string rather than as an RE.

MATCHING

In the event that an RE could match more than one substring of a given string, the RE matches the one starting earliest in the string. If the RE could match more than one substring starting at that point, its choice is determined by its preference: either the longest substring, or the shortest.

Most atoms, and all constraints, have no preference. A parenthesized RE has the same preference (possibly none) as the RE. A quantified atom with quantifier `{m}` or `{m}?` has the same preference (possibly none) as the atom itself. A quantified atom with other non-greedy quantifiers (including `{m,n}?` with m equal to n) prefers shortest match. A quantified atom with other non-greedy quantifiers (including `{m,n}` with m equal to n) prefers shortest match. A bracket has the same preference as the first quantified atom in it which has a preference. An RE consisting of two or more branches connected by the | operator prefers longest match.

Subject to the constraints imposed by the rules for matching the whole RE, subexpressions also match the longest or shortest possible substrings, based on their preferences, with subexpressions starting earlier in the RE taking priority over ones starting later. Note that outer subexpressions thus take priority over their component subexpressions.

Note that the quantifiers `{1,1}` and `{1,1}?` can be used to force longest and shortest preference, respectively, on a subexpression or a whole RE.

Match lengths are measured in characters, not collating elements. An empty string is considered longer than no match at all. For example, `bb` matches the three middle characters of `abc`, `(week|week)(night|knight)` matches all ten characters of `weeknights`, when `(.*)`. `*` is matched against abc the parenthesized subexpression matches all three characters, and when `(a*)` is matched against `bc` both the whole RE and the parenthesized subexpression match an empty string.

If case-independent matching is specified, the effect is much as if all case distinctions had vanished from the alphabet. When an alphabetic that exists in multiple cases appears as an ordinary character outside a bracket expression, it is effectively transformed into a bracket expression containing both cases, so that x becomes `[xX]`. When it appears inside a bracket expression, all case counterparts of it are added to the bracket expression, so that `[x]` becomes `[xX]` and `[*x]` becomes `[*Xx]`.

If newline-sensitive matching is specified, `^` will never match the newline character (so that matches will never cross newlines unless the RE explicitly arranges it) and `^` and `$` will match the empty string after and before a newline respectively, in addition to matching at beginning and end of string respectively. ARE `\A` and `\Z` continue to match beginning or end of string only.

If partial newline-sensitive matching is specified, this affects `^` and bracket expressions as with newline-sensitive matching, but not `^` and `$`.

If inverse partial newline-sensitive matching is specified, this affects `^` and `$` as with newline-sensitive matching, but not `^` and bracket expressions. This isn’t very useful but is provided for symmetry.

LIMITS AND COMPATIBILITY

No particular limit is imposed on the length of REs. Programs intended to be highly portable should not employ REs longer than 256 bytes, as a POSIX-compliant implementation can refuse to accept such REs.

The only feature of AREs that is actually incompatible with POSIX REs is that `\` does not lose its special significance inside bracket expressions. All other ARE features use syntax which is illegal or has undefined or unspecified effects in POSIX REs; the *** syntax of directors likewise is outside the POSIX syntax for both SREs and ERs.

Many of the ARE extensions are borrowed from Perl, but some have been changed to clean them up, and a few Perl extensions are not present. Incompleteness of note include `\b`, `\B`, the lack of special treatment for a trailing newline, the addition of complemented bracket expressions to the things affected by newline-sensitive matching, the restrictions on parentheses and back references in lookahead constraints, and the longest/shortest-match (rather than first-match) matching semantics.

The matching rules for REs containing both normal and non-greedy quantifiers have changed since early beta-test versions of this package. (The new rules are much simpler and cleaner, but don’t work as hard at guessing the user’s real intentions.)

Henry Spencer’s original 1986 regexp package, still in widespread use (e.g., in pre-8.1 releases of Tcl), implemented an early version of
today's EREs. There are four incompatibilities between regexp's near-EREs (`REs` for short) and AREs. In roughly increasing order of significance:

In AREs, `\` followed by an alphanumeric character is either an escape or an error, while in RREs, it was just another way of writing the alphanumeric. This should not be a problem because there was no reason to write such a sequence in RREs.

`{` followed by a digit in an ARE is the beginning of a bound, while in RREs, `}` was always an ordinary character. Such sequences should be rare, and will often result in an error because following characters will not look like a valid bound.

In AREs, `\` remains a special character within `[ ]`, so a literal `\` within `[ ]` must be written `\\`. `\` also gives a literal `\` within `[ ]` in RREs, but only truly paranoid programmers routinely doubled the backslash.

AREs report the longest/shortest match for the RE, rather than the first found in a specified search order. This may affect some RREs which were written in the expectation that the first match would be reported. (The careful crafting of RREs to optimize the search order for fast matching is obsolete (AREs examine all possible matches in parallel, and their performance is largely insensitive to their complexity) but cases where the search order was exploited to deliberately find a match which was not the longest/shortest will need rewriting.)

**BASIC REGULAR EXPRESSIONS**

BREs differ from EREs in several respects. `|`, `+`, and `?` are ordinary characters and there is no equivalent for their functionality. The delimiters for bounds are `{` and `}`, with `{` and `}` by themselves ordinary characters. The parentheses for nested subexpressions are `{` and `}`, with `{` and `}` by themselves ordinary characters. `^` is an ordinary character except at the beginning of the RE or the beginning of a parenthesized subexpression, `:` is an ordinary character except at the end of the RE or the end of a parenthesized subexpression, and `*` is an ordinary character if it appears at the beginning of the RE or the beginning of a parenthesized subexpression (after a possible leading `^*`). Finally, single-digit back references are available, and `<` and `>` are synonyms for `[[<:]]` and `[[>:]]` respectively; no other escapes are available.

[QuARS HOME] [Contents] [Index]

**Analysis: Views Derivation**

> Since the "View Derivation" is based on the identification of (sub)sections, to perform this kind of analysis it is mandatory that the requirements document is divided into sections according to the following format:

1. 
   1.1
   1.2
   ...
   n.
   n.1
   ...

The Views Derivation relies again on dictionaries. The Dictionaries of the Views (V-Dictionaries) contain domain-related terms (instead of defect-related terms). A V-Dictionary is the set of all the terms dealing with a particular View, i.e. those terms that, if contained in a sentence, indicate that this sentence is dealing with the argument the View is referred.

To select the View derivation functionality of the QuARS tool the V button on the top buttonbox of the GUI has to be clicked (arrow 1 in figure A3). Once the View derivation functionality has been selected, in the Dictionaries frame the available dictionaries are shown (arrow 2 in figure A3). Each dictionary in the Dictionaries frame corresponds to a particular View that can be derived. It is possible to select the View of interest by selecting the correspondent V-Dictionary. Once the requirements file is loaded, the View derivation can be started. To start the View derivation push the Analysis button or select [View derivation] name of the View form the Analysis menu on the top of the GUI.

The output of the View derivation is composed of:

- A cluster of the sentences, extracted from the input requirements file, dealing with the specific topic the View is referred to (arrow 3 in figure A3).
- A MS Excel graph, showing the total number of sentences and the number of those sentences belonging to a view contained in each section of the document analyzed (arrow 4 in figure A3).
### Lexical analysis

To perform one of the lexical-based analyses it is necessary to select the Lbutton on the top tool bar of QuARS (arrow 1 in figure A1). Once the lexical-based analysis has been selected, in the Dictionaries frame the dictionaries corresponding to all the available lexical-based analyses are shown (arrow 2 in figure A1). The default set of analyses is composed of four dictionaries: **Optionality, Subjectivity, Vagueness, Weakness.** Additional dictionaries can be created to enlarge the set of available analyses. It is possible to select the kind of analysis of interest by selecting the correspondent dictionary book-mark in the Dictionaries frame.

Before starting the analysis the text file containing the requirements has to be loaded. Once the input file has been selected, its content appears in the Input frame. The analysis starts when the Analysis button is pushed (arrow 3 in figure A1).

In the Output frame those sentences containing the particular defect we are investigating on are shown along with the indication of the individual term that makes the sentence defective (according to the kind of analysis selected). The defective sentences can be now corrected.
Syntactic analysis

To perform the syntax-based expressiveness analysis the button on the top tool-bar of the QuARSGUI (arrow 1 in figure A2).

Once the syntax-based analysis has been selected, in the Dictionaries frame the possible dictionaries of each type of the available syntax-based analyses are shown (arrow 2 in the figure).

It is possible to select the type of analysis of interest by selecting the correspondent dictionary tab in the Dictionaries frame (note that the multiplicity analysis doesn't need any dictionary, for this reason it is void).

The available analyses are: Implicit, Multiplicity and Underspecification.

Before starting the analysis the text file containing the requirements has to be loaded.

Once the input file has been selected, its content appears in the Input frame.

The analysis starts when the Analysis button is pushed (arrow 3 in figure A2) or [Syntactic] name of the analysis selected from the Analysis menu on the top of the GUI.

In the Output frame those sentences containing the particular defect we are investigating on are shown along with the indication of the individual term that makes the sentence defective (according to the kind of analysis selected).

The defective sentences can be now corrected.
**Note**: the way the analysis is performed is the same than the lexical-based analysis.
The difference between the two types of analyses is transparent to the user.
In fact, for performing these analyses the tool, first derive the syntactical structure of each sentence in the input file.
Anyway, the syntactical structure of the sentences is not displayed.
While for the implicitity and multiplicity analysis dictionaries are necessary, the multiplicity analysis, even though it relies on the syntax structure of the sentences, for its nature, it doesn’t need any dictionary.

[QuARS HOME] [Contents] [Index]

**QuARS Splashscreen**
Index

- About QuARS
- About QuARS Help System
- Analysis: Defect Identification
- Analysis: Views Derivation
- Buttonbox Reference
- Contents
- Dictionaries name format and limitations
- Dictionary Editor Keyboard Shortcuts
- Dictionary Handling
- Dictionary Wizard
- Drag & Drop
- How To Convert an input file in the suitable file format
- How To create a new Dictionary
- HOW TOs
- Indicators Explication
- Input Handling
- Installing QuARS
- Introducing QuARS
- Introduction
- Keyboard Shortcuts
- Menu Reference
- Menu Reference Summary
- Performing Analysis
- QuARS Dictionary Editor Reference
- QuARS Editor User Interface
- QuARS Editor: Buttonbox Reference
- QuARS Editor: Keyboard Shortcuts
- QuARS Editor: Menu Reference
- QuARS Editor: Menu Reference Summary
- QuARS Editor: Regular Expressions in Search/Replace Window
- QuARS Editor: Search & Replace Function
- QuARS Editor: Search Function
- QuARS Help System
- QuARS Help-On-Help: The User Interface (usage)
- QuARS License
- QuARS Output Frame: managing Sentences
- QuARS Splashscreen
- QuARS Status-Bar
- QuARS User Interface
- QuARS User Manual
- QuARS: Getting Started
- References
- Results Handling
- System Requirements
- The "Coleman-Liau" readability index
- The QuARS Editor
- Uninstalling QuARS

[QuARS HOME] [Contents] [Index]

QuARS Help System
Version 0.5 (build 01.20050621)

Latest Update: 21 June, 2005

QuARS v4.1 Copyright © 2000-2005  SSEC-FM&T (ISTI-CNR)
All Right Reserved.