CADIXE Editor User Manual[1]

(Preliminary version)

From CADERIGE Project : http://caderige.imag.fr/

Editor version 2.0a4, Built 05032005)

Contact : Gilles.Bisson@imag.fr

Goal of the Editor

The CADIXE XML Annotation Editor has been developed in the frame of the CADERIGE Project (http://caderige.imag.fr) for doing easy and interactive annotation of text documents using a tag set described by an external DTD. Contrary to many other classical XML editors, the end-user using CADIXE does not have to structure directly all the document as an XML tree. New tags can be introduced one by one, in any order. Text markup is displayed by using a CSS style sheet that can be freely modified by the end-user.

Content of this Manual

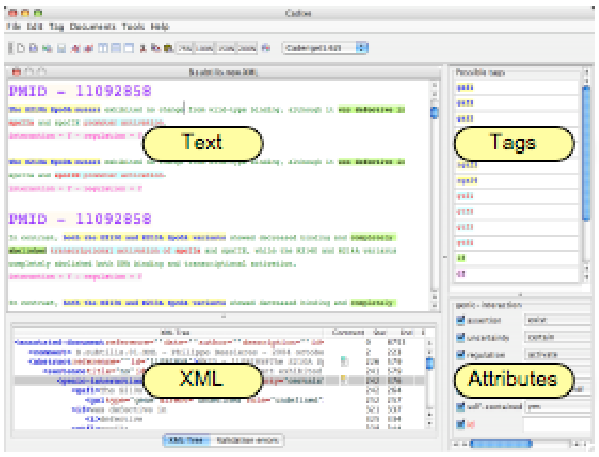
[1] Many thanks to Thierry Poibeau (poibeau@lipn.univ-paris13.fr) who wrote the first release of this manual.

CADIXE in a Nutshell: Why and What !

Currently, there exists a lot of implementation of XML editors but most of them are oriented toward the *creation* or the *modification* of XML documents and are poorly adapted to the *annotation task*. In such process, the user wants to insert progressively some XML tags within an existing text, to express some semantic information described through an external DTD. The CADIXE editor has been developed to fulfil this needs and to allow the user :

- To insert the tags with very few constraints on the insertion order
- To visualize the document not as a tree or an XML document, but just as a text
- To eases the collaboration between several annotators
- To avoid a long and complex learning curve

The first requirement is of course crucial since during the annotation task the user interprets "in real time" the meaning of the document. So it is not possible to force him/her to enter the tags in the way they are organized in the DTD, because the first element identified by the user is not always (never in fact) the root of the DTD. So, an annotation editor must provide some freedom to the annotator in the way he/she inserts the information.



The interface of CADIXE editor we developed in the Caderige project (http://caderige.imag.fr) is mainly composed of four main parts. The text zone where the user sees the text he/she want to annotate, the attributes zone allowing to enter or to modify the values associated to the current tag (i.e. the tag in which the text cursor has been moved), the XML code currently generated and the list of the tags that can are allowed at a given time. Thus, the tagging process is simple : first the user highlights with the cursor the part of the text to annotate. The tags list shows all the relevant tags. The user selects the tag he/she wants to apply and then enters the values in the attributes editor. There is also a mode in which the editor forced the user to enter the values of the attributes. This mode is relevant when the user is not familiar with the DTD. Moreover, it is possible to associate to each tag a "comment" to explain the annotation choices to the other member of a project.

Finally, a "style" is associated to each tag in order to express the way this tag is displayed in the document. Several style sheets (based on the CSS format) can be loaded in a document to allow different "point of view". For instance, in a biological domain, an annotated sentence could look like this one: each color corresponding to a specific information :

A **low level** of GerE **activated** transcription of CotD by GerE RNA polymerase but in vitro

The CADIXE editor has been implemented in JAVA 1.42 and it can be used on the following platforms : Windows, Linux, MacOs X. It is used in our project but also at the SIB (Swiss Institute of Bioinformatics) with another DTD, in the frame of the European project "BioMint". The software can be (freely) obtained by any researcher, which is involved in a document annotation task, if you are interested just contact : gilles.bisson@imag.fr.

Installation and Start-up

Minimal configuration

The software is written in java. It can thus be launched on a large number of operating systems (it has been tested on Linux, MacOS X and Windows). A Java Virtual Machine (JVM) v1.4.2 or higher is required and the system should have at least 256 Mo RAM.

Software installation

The application is named CADIXE.jar (**this name should not be changed** !) and you just have to drag it in the folder of your choice. During the first run, the editor builds automatically the hierarchy of files that it needs. They are structured as follows :

- CADIXE.jar : runtime
- /resources : resources (icons, messages, …) used by the application
- /CADIXE-preferences : preferences folders
  - CADIXE.properties : default preferences file
- /formats
  - /models : document models
  - /dtd : DTD files
  - /styles : CSS style sheets
- /documents : user documents
- /scripts : scripts folder (not use in this release)

Caution : when you install a new release of CADIXE over a previous one, you can either put it in a new folder or, more simply, move it to the current location. In the later you **must** delete the existing "resources" folder before the first run to update resources.

Positions and names of the folders written in green can be modified by the user (cf. "Preferences settings" and "Command line" descriptions in the chapter "CADIXE settings"). Therefore, the application can be customized such that common knowledge (DTD, CSS, …) stays in the application folder and are shared between the users, while specific knowledge (documents, preferences, …) are stored in user's folders.

How to launch the editor …

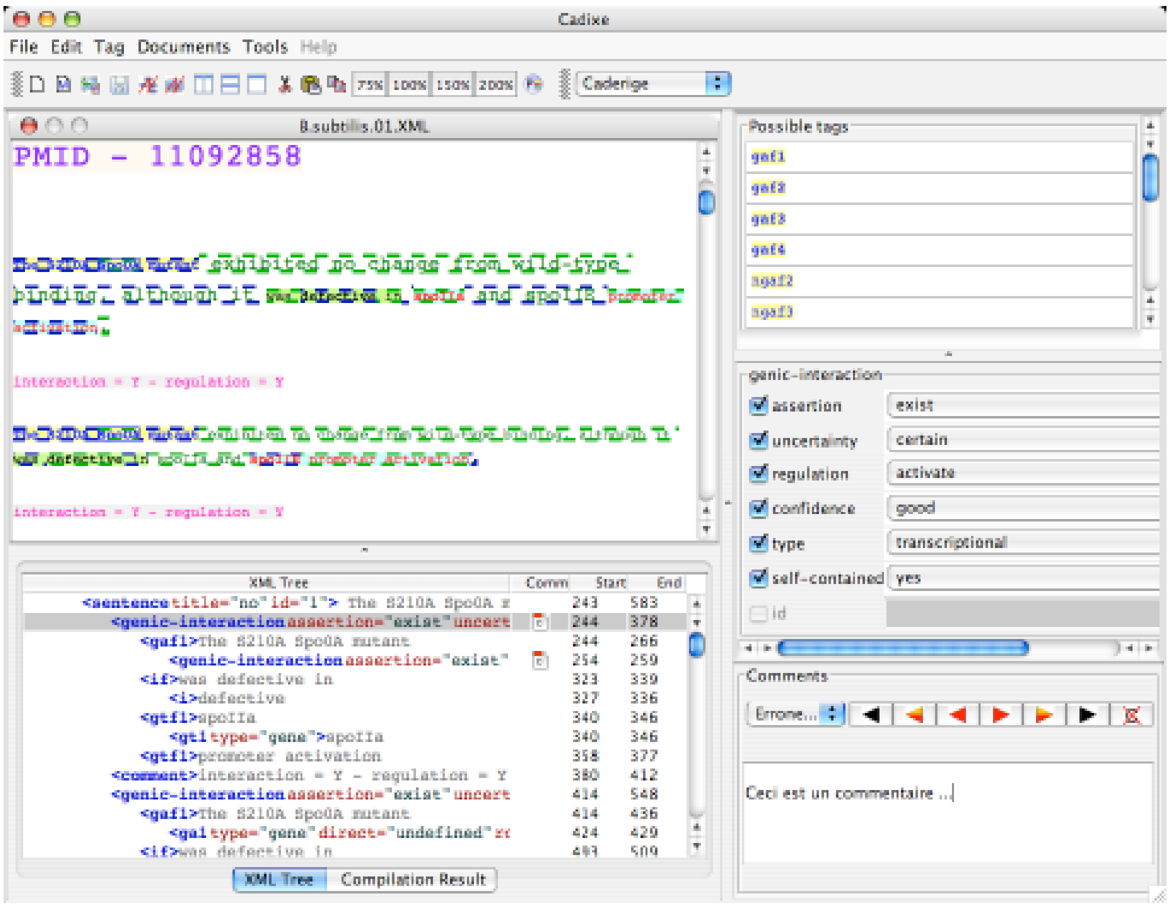
The annotation editor can be launched in a console (xterm or equivalent), by using the following command line :

```
java  -jar CADIXE.jar
```

If you are not in the right path, you can provide it in the command. For instance :

```
java  -jar  /Volumes/Memnos/Caderige/CADIXE/version 2/CADIXE.jar
```

However, with the Operating System having a "file browser" (MacOS X or Windows for example), the annotation editor can also be launched with a double-click on the *CADIXE.jar* icon. As this application has been written with "Swing", the aspect of the User Interface is adapted according to the OS. For instance, with MacOs X you should obtain this window :



PMID - 11092858

interaction = Y - regulation = Y

interaction = Y - regulation = Y

XML Tree	Comm	Start	End
<sentence title="no" id="1"> The S210A SpoOA m		243	583
<genic-interaction assertion="exist" uncert	✓	244	378
<gaf1>The S210A SpoOA mutant		244	266
<genic-interaction assertion="exist"	✓	254	259
<if>was defective in		323	339
<i>defective		327	336
<gtf1>spoIIa		340	346
<gt1 type="gene">spoIIa		340	346
<gtf1>promoter activation		358	377
<comment>interaction = Y - regulation = Y		380	412
<genic-interaction assertion="exist" uncert		414	548
<gaf1>The S210A SpoOA mutant		414	436
<ga1 type="gene" direct="undefined" m		424	429
<if>was defective in		493	509

Possible tags: gaf1, gaf2, gaf3, gaf4, ngaf2, ngaf3

genic-interaction

✓ assertion	exist
✓ uncertainty	certain
✓ regulation	activate
✓ confidence	good
✓ type	transcriptional
✓ self-contained	yes
id	

Comments

Ceci est un commentaire ...

What's new and History

What's new …

New features in the release 2.0a4 with respect to the release 2.0a3

The release 2.0a4 is the fourth "alpha" release of CADIXE 2. Here are the new features.

- EMPTY tags are implemented. These tags are displayed in the document as icons (GIF images) that can be defined by the user in the style files.
- New commands allowing to load/remove styles sheets and to create models.
- We add the line : <?xml-stylesheet type="text/css" href="XXX.CSS"?> in the header of annotated documents, where XXX.CSS is the name of the current style. In this way by putting the XML file and the style in the same folder, one can visualize XML annotations through a simple Web browser (FireFox for instance).

Known missing features of the release 2.0a4

- No Undo/Redo possibilities.
- No documentation on-line
- Commands to modify the range of an existing tag.
- Call to external scripts to automate some annotations.
- Real HTML exports.
- HTTP server allowing to externally drive the editor (load, close, …).
- Automatic completion of tags (as in the release 1.xx of CADIXE)

Known bugs

- When comments are written in the tool placed at the right part of the editor, modifications are taken into account at the next click in another part of the interface.
- When the editor window is small and that several tools are simultaneously displayed, some parts of these tools can be hidden.
- Deletion of the tags can be time consuming on a large document.

History of the software

New features in the release 2.0a3 with respect to the release 2.0a2

- XML code validation using Xerces.
- Several bugs of the release 2.0a2 have been fixed, concerning the following areas :
    - Search and Replace functions
    - Dialog box to fill the value of the attributes
    - Cut and paste has been fixed and generalized
    - Display of the tags after inserting a more general tag on a zone.
    - The keystroke "suppr" is now correctly handled
    - Improvement done to the current manual

New features in the release 2.0a2 with respect to 2.0a1

- Search functions allowing to find either in the text or in the tags (new to V 2.0). The user can search for a specific tag, value of attribute or comment.
- The user can associate a comment to each tag of the document (new to V 2.0). A comment is made of two parts : a "status" among : {validated, unvalidated, erroneous} and a free text area.
- Implementation of "parametric styles" (new to V 2.0). They allow to associate several styles to a given tag. The style to use during display process is chosen according to the current tag AND the value taken by one or several of its attributes.
- Preference dialog box has been implemented.
- Ease the reuse of the document annotated with the release 1.xx of CADIXE. Style sheets and DTDs must be put in the corresponding folder of the release 2.0a2
- Users can express where the different resources are stored. In such way it is possible either to share files between users (DTD, styles,…) or to put them in a user specific location (Documents, preferences, …).
- Now we save the document settings (zoom, spacing, cursor position, …) and we allow to attach some "meta-information" (annotator, history, …) to the files.
- New command line arguments : -load, -name, -model, -userhome.
- We fixed several bugs of the release 2.0a1

New features in the release 2.0a1 with respect to 1.17 (and more generally 1.xx)

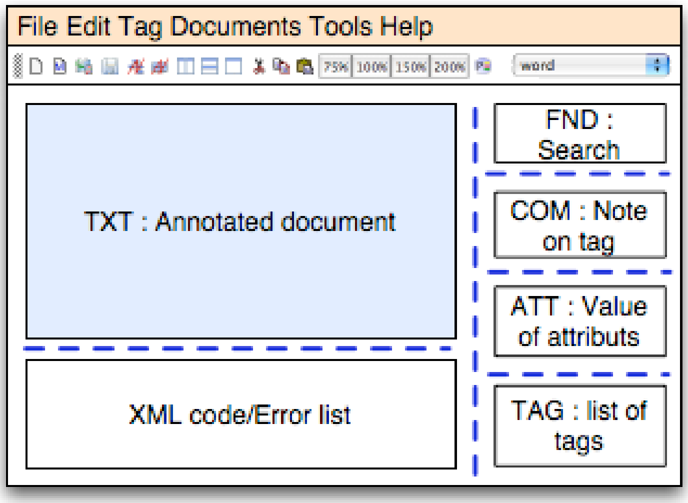
- This release 2.0 is a totally new implementation of CADIXE editor with some dramatic improvements in terms of features, speed and stability.
- Possibility to simultaneously open and to display several documents.
- Cut/Copy/Paste of tagged parts of the document. Copied text is stored in the clipboard as an XML string so it can be exchanged with any textual application.
- Better control of tag insertion : now the editor verify if the inserted tag is more general in the DTD that those occurring in the selected area of the documents.
- New graphical annotations (boxes) and style sheets are coded using CSS format.
- Implementation of a zoom.
- Resources are self-installed at the first run.

Interface Organisation and Main Principles

Overall structure of the User Interface

Besides the classical Menu bar and the Toolbar containing icons of the most common functions, the main window of the application is separated in several zones that can be resized by using the separations bar :

- TXT : original document being annotated (upper left)
- XML : XML tags introduced in the text (lower left)
- FND : search tool to find a fragment of text or a specific tag
- COM : comment tool allowing to associate a note to the current tag
- ATT : value of the XML attributes of the current tag
- TAG : list of available tags from the current context



Display of the following areas TOOLBAR, ATT, TAG, FND and COM can be switched on/off with the commands occurring in the menu TOOLS. Moreover the relative position of the four tools at the right can be modified (cf “Preferences setting”).

Text annotation process using CADIXE editor

The annotation process mainly includes three different steps :

1) First, the user selects with the mouse the chunk of text (words, paragraph) to annotate, by default the selection is automatically extended to the beginning/ending of the next/previous words (cf. “preference setting” to change this behavior).
2) Then he/she applies an annotation tag on this chunk by clicking on the corresponding tag in the right side of the window. The corresponding chunk of text will then appear by using the display format associated with the tag in the current style sheet. Moreover, the new tag appears in the XML area in the lower part of the editor.
3) Finally, the user enters the values of the tag’s attributes (if any) in the ATT area.

The “current tag” is always defined by the cursor position. Thus, to check a specific annotation, the end-user only needs to click on the relevant chunk of text. Tag name, list of attributes and their values automatically appears in the ATT area of annotation editor. These values can be modified by the end-user at any moment. If values have been defined in the DTD by an enumerated list, they can be directly chosen from a menu. Otherwise, a text field is provided to enter the value.

The list of available tags (TAG area) is dynamically modified according to the position of the cursor in the text. Only tags that are (according to the DTD) *more specific* than the current tag AND *more general* that the tags occurring in the selected area can be used. Other tags are either greyed or made invisible (cf. "preferences settings"). This behaviour has been defined to avoid tagging errors and to guide the end-user in the annotation process. When a new tag is inserted, a dialog box optionally appears (cf "preferences settings") to enter values of the different attributes. Clicking on the "Cancel" button stops the current insertion process. The REQUIRED attributes are written in red.

The XML part of the main window shows the list of all the tags inserted within the document. To each tag is associated four information : the positions of the first and last character of the tag, the fact there is a comment associated with this tag (the color of the icon expressing the comment status) and finally the fact there is a validation error associated to this tag (cf. menu TOOLS). The navigation between the annotated document and the XML part is fully synchronized, that is to say that clicking on one area automatically position the other area to the corresponding text chunk

Modifying a text and its annotations

As in a classical editor, the end-user is free to insert, delete or copy-and-paste text in a document. To delete an annotation tag, the user must put the cursor on the corresponding text area and select the command "Delete current tag" in menu TAG. To suppress *all the annotations* in an area, the user must apply the command "Delete tag group". In that case, after a confirmation query, the current tag will be deleted, together with all the subsequent tags (XML sub-tree).

When the user selects a chunk of text then hits the key "Suppr" (or "Delete") the character are deleted and the associated annotations are deleted. However, to avoid the erroneous keystrokes, the editor asks for a validation when the delete zone contains more than one annotation (as with the command "Delete tag group")

In some cases, due to the DTD structure, several different tags can be applied on the same chunk of text (same start and stop position). In such a case, to select the relevant tag, it is possible to click on the corresponding tag in the XML zone.

How to write a DTD

Annotation of text documents is done through a "tag set" whose "grammar" is defined in an external, domain specific, DTD (Document Type Definition). Below are two websites (among many others) concerning XML and explaining the way to write a DTD :

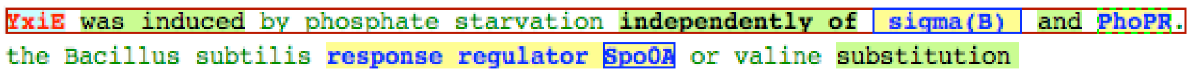
- XML description : http://www.brics.dk/~amoeller/XML/overview.html
- How to write a DTD : http://www.w3schools.com/dtd/dtd_intro.asp

How to write a style sheet

The style sheets used by CADIXE to display the tags, are based on the classical CSS format (Cascading Style Sheets) that must be written out of the editor. To create or to edit CSS files the user can either used a text editor or, far better, to use a specific CSS editor. Freeware and shareware versions of such applications exist for all the OS (Linux, MacOs X, Windows), here are some of them :

- Multi-OS:
  - http://www.nvu.com/
  - http://www.westciv.com/style_master/
  - http://www.ucware.com/juststyle/index.htm
- Linux :
  - http://cssed.sourceforge.net/
- MacOs X :
  - http://www.macrabbit.com/cssedit/

By default, styles sheets are placed in the directory "formats/styles". Here is the list of the CSS attributes that are understood by CADIXE. Other attributes are currently ignored but we hope to decrease these restrictions in the next releases. Here is an example of display :


YxiE was induced by phosphate starvation independently of sigma(B) and PhoPR.
the Bacillus subtilis response regulator Spo0A or valine substitution

Hereunder is the list of the attribute handled by CADIXE. Specific attributes of the editor (not in the CSS 2.0 reference) begin with the prefix "cadixe-" :

Standard attributes to set fonts and colors :

- background-color : *color ;*
- color : *color ;*
- font-family : *font name ;*
- font-style: italic ;
- font-size : *number ;*
- font-weight : bold;
- text-decoration : {underline, line-through} ;

Attributes to draw a "box" around tags :

- border-color : *color ;*
- border-width : *number* ;
- border-style : {solid, dotted, dashed} ;
- cadixe-border-shape : {rectangle} ;

Attributes to declare "parametric styles" (see chapter "Special Features"):

- cadixe -index-attribute : *attribut1, attribut2, …* ;

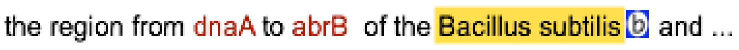
Attributes to chose the icon of an "EMPTY" tag (see chapter "Special Features"):

- cadixe -empty-tag : *path* ; - Relative path to a GIF picture (~ 12pt), the root folder is (momentarily!!) the folder "resources/icons".

Special Features of CADIXE

The “EMPTY” tags

EMPTY tags (for instance a tag such as </BR> in XHTML) are special tags of the DTD whose goal is not to annotate a fragment of text but to introduce some local information in the document. These tags are displayed as icons. The icon (GIF image) to use for a given tag is given in the tag style (cf “Creation of Style Sheets”). Here is an example of display :


the region from dnaA to abrB of the Bacillus subtilis b and ...

Of course to insert this kind of tag it is not necessary to select a fragment of text as with the other tags. However, if a fragment is selected the EMPTY tag is inserted at the end.

Notion of “Comments”

With CADIXE, the user can associate a *comment* to each tag inserted in the document. This comment allows one to express the meaning of an annotation and/or his/her doubts and queries about it. This feature is interesting as soon as the annotation task can spend a long time and/or can involved several different annotators. A comment is organized in two parts :

- A text field in which the user can provide explanations.
- A “status” whose value is among : {validated, unvalidated, erroneous}.

Tags having a comment are displayed in the XML panel with a specific icon whose color mirrors the status. These notes are stored in the XML file as a comment “<-- ... -->“ (in the XML meaning of the term) that are interpreted by CADIXE during loading process. This information therefore doesn’t change the congruity of the document with respect to the DTD. To add (or modify) a comment associated to the current tag, there are three ways :

- To call the command “Add comment” from TAG menu.
- To do a double click in the column “comment” of the tag in XML zone.
- To activate the display of the “comment tool” in the right part of the editor. In that case the command “Add comment” is automatically inactivated.

When the dialog box (or the tool) is displayed, the user can navigate between the comments thanks to a set of three colored pairs of buttons. “Black” ones allow to go to the next/previous comment, “Orange/red” one allow to go to the next/previous comment having a status either “unvalidated” or “erroneous”, and “Red” ones allow to navigate between the “erroneous” comments only.

Finally, the Reset button (the crossed “C”) suppresses the current comment : the status becomes “validated” and the text field is emptied.

Parametric styles

With CADIXE it is possible to associate several style definitions to a given tag, the selection of the style being dynamically done according to the value of one or several attributes. This feature allows to graphically telling apart a set of information having the same semantic without the need of multiplying the number of tags described in the DTD.

We are going to illustrate this feature with a simple example. Let us imagine we have in a (biological) DTD a tag named <INTERACTION> with an attribute ROLE that can take two different values : ACTIVATE and INHIBIT. If the user wants to have two different display styles for this tag according to the value of ROLE, he/she just have to put the following lines in the CSS styles (see also “Creation of style sheets”) :

1) First we need to point out that the style associated to the tag INTERACTION is “specialized” by the value of ROLE. This is achieve by adding the attribute *CADIXE-index-attribute* in the definition of this style :
   - *.interaction { … CADIXE-index-attribute : role ; … }*
2) Then we create in the CSS file two new styles by concatenating the name of the tag with each of the value taken by ROLE. Tag name and value identifier must be separated by an underscore (“_”). In the hereunder example the variation concerns the color of the text but any other modification is feasible.
   - *.interaction_activate { … color : red ; … }*
   - *.interaction_inhibit { … color: blue ; … }*

The two new styles will be automatically used by CADIXE according to the value filled in ROLE, the display will be the following :

Tag in the annotated document	Display
`<interaction role=“activate”> Gene Abc </interaction>`	Gene Abc
`<interaction role=“inhibit”> Gene Abc </interaction>`	Gene Abc

If the user would like to specialize the display with several attributes he/she just have to put a list of attributes in *CADIXE-index-attribute* (separated by a comma) and to create has many new styles that the number of possible combination of values.

Looking at the annotated document on the WEB

It is possible to show an annotated document with a “standard” WEB browser as soon as it is able to deal with the CSS style sheets in a XML document (that’s the case for instance of FireFox, Safari, …). Indeed, when a document is saved in CADIXE the following line is automatically inserted in the file header :

```
<?xml-stylesheet type="text/css" href="XXX.CSS"?>
```

The name XXX.CSS is the one of the current style sheet. Therefore by copying this file in the same folder than the document, any user can load and display the document on the WEB.

Caution: as the annotated document are “pure XML”, to display a “Carriage Return” between some of the tags, one need to add the attribute “`display : block;`” in their style descriptions. Moreover, to improve the aspect of the document, it is interesting to insert some attributes such as “`padding : `*`number`*`px;`” and “`vertical-align : `*`number`*`px;`”. Adding these attributes is not a problem for CADIXE since there are currently ignored.

Menubar Description

FILE menu

This menu gathers the commands to create, to load and to save annotated documents :

- New Document : CADIXE initializes a new document using the current “default model” (cf. “preference settings”). A model is a file containing the name of a DTD and a list of style sheets that will be load in the new document.

- New Document from Model : the editor asks the user for the model he/she wants to use and then builds a new document. The path to the folder containing the model files is defined in the “preference settings”.

- Load Document : loads a document (annotated or not). When the document is a plain text file, the root tag of the DTD defined by the default model is automatically inserted. When the document is already annotated, the editor searches for the DTD and the style sheets in the directories defined in the “preference settings”.

  If the DTD used by the document is missing an error is fired. However, if there is no style sheet a new one is automatically created by CADIXE by using the name of the current DTD. Of course, this style sheet is “empty”: there is not specific graphical style associated to the tags.

- Load recent : list of recent files loaded within the editor.

- Insert Text … : insert a text file at the current position of the cursor.

- Save Document : save the current annotated file with the names of the DTD and the style sheets. The meta-information (cf. menu “Documents”), cursor position, line spacing and level of zoom are also saved in the header.

- Save As … : asks the user for the name of the annotated file to save.

- Save Document as Model : the editor creates a new “model” file (Cf the command “new document”) from the current document. The name of this file is the DTDs name of the current document and it is stored in the folder containing the models.

- Quit … : exits from the editor after a confirmation dialog box.

EDIT menu

In this release of CADIXE, it is possible to copy/paste an annotated area. In that case, the annotated text is stored in the clipboard as a XML string. Therefore it is possible to exchange the annotations between CADIXE and any application dealing with text.

Caution : in this release the editor doesn’t validate the content of the clipboard before pasting the tags in the document, thus if the external application sends a XML string which is wrong with respect to the DTD, the editor will fail.

- Cut : cuts the selected text and the associated annotations and comments.

- Copy : copies the selected text and the associated annotations and comments.

- Paste : pastes the content of the clipboard. However, if it contains some tags, some conditions must be fulfilled in order this command succeed : the current tag in which the clipboard is pasted must be more general or identical (in that case the tags are merged) to the “root” tag in the clipboard.

- Paste as text : pastes the plain text contained in the clipboard in the current tag.

- Find … : searches for a string or a regular expression (when the associated check-box is activated) in the annotated document. The syntax of RG is described in the following site (among many) and here are some classical commands.
    - http://www.regular-expressions.info/reference.html
    - http://java.sun.com/docs/books/tutorial/extra/regex/index.html

.	: the “dot” corresponds to any character in the search.
^ and &	: indicates the beginning and the ending of a line.
\d, \w, \s	: corresponds to a number, a letter and a space character.
[ ]	: list of possible characters (ie : [0-9] ; [a-z] ; [abcdef0123] …).
?	: sequence of characters repeated 0 or 1 time.
+	: sequence of characters repeated 0 or several time.
*	: sequence of characters repeated 1 or several time.
|	: alternative (if we seek for “this” or “that”put “this|that”).
{*num*}	: sequence of characters repeated *num* time exactly.
{*min,max*}	: sequence of characters repeated between *min* and *max* time.
( )	: to delimit the sub-expression (i.e.: “[0-9\s(protein|gene)+”).

  Besides, the 4 buttons placed at the bottom of the dialog : “First”, “Previous”, “Next” and “Last” allows to navigate between the occurrences.

- Replace … : allows to replace a substring by another one in the document. To be coherent the replacement is feasible only is this substring is completely included in one or several tags. If that is not the case the replacement is impossible since the editor doesn’t know at what positions the different tags must be applied on the final substring.

- Search Tag … : this command retrieves a tag in the document by using one or several criteria : its name, comment text and status and the value attribute. For instance the user can seeking for the <Abstract> tags which are not validated (unvalidated) and that are containing a missing attribute named ID …

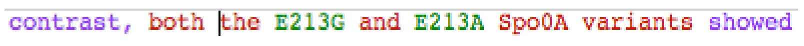
- Next Find/Search : Applies again the last find/search command performed.

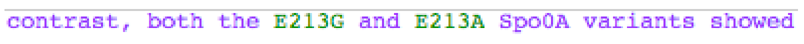
- Preferences : displays the preferences dialog box allowing to customize the software. The options are described in another section of this document.

TAG menu

This menu gathers the commands to manage the tags. In the current release some options are still missing to extend/shorten the range of a tag.

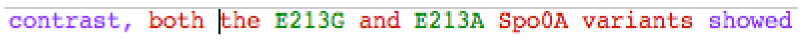
- Delete current Tag: suppresses the current tag. This command let the included tags unchanged, for instance if we suppress the tag with red color :

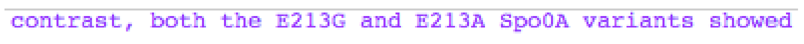
  Before :
  
  contrast, both the E213G and E213A Spo0A variants showed

  After :
  
  contrast, both the E213G and E213A Spo0A variants showed

  It is impossible to delete the tag corresponding to the "root" of the DTD since this deletion has no meaning in the frame of CADIXE use.

- Delete Tree Tags : suppresses recursively all the tags contained in the current tag. As this command could be dangerous (you can delete a large amount of tags) a confirmation dialog box is displayed before the deletion. If we tag back the previous example, here is the effect of this command :

  Before :
  
  contrast, both the E213G and E213A Spo0A variants showed

  After :
  
  contrast, both the E213G and E213A Spo0A variants showed

- Add comment … : adds a comment to the current tag, the notion of "comment" is describe in a previous section of this document. We can also add a new comment by double-clicking in the column "comment" of the XML zone.

DOCUMENTS menu

This menu gathers the commands allowing to manage the documents that have been loaded in the editor.

- Document Information … : this command allows to display some information about the document (DTD used, saving date) and to edit some "meta information" associated to the document. These information are stored in the header of the document as XML comments <-- … -->. We can provide :
  - The (long) title of the document,
  - The names of the annotators,
  - The references to the genuine document,
  - Some comments about the history of the annotation …

- Zoom : to modify the zoom level, this information is stored (by default 100%).

- Spacing : spacing (in number of points) between the lines of the document.

- Window organisation : this command allows to switch between the mono/multi-documents modes and to set the way these documents are organized : horizontally or vertically. The maximum number of documents that can be displayed can be modified in the CADIXE preference settings. Of course these possibilities are interesting to use in order to compare two or several annotated documents.

- Load Style … : loads a new styles file in the current document. This new style becomes the current one.
- Reload Style : reloads the current style from its file. This command is very useful when the user is "finalizing" a style sheet with an external CSS editor (cf. "Style sheet") to display rapidly the effect of a modification. In this way he/she doesn't have to close and reload the document to take into account the new style.
- Remove Style : removes the current style from the document (the file is not deleted). If there is just one style in the document this command do nothing.
- Styles ▸ : this sub-menu contains the list of styles loaded in the current document and allows to select a given style.
- Close document : close the current document (the document having the focus). If it has not been already saved a confirmation dialog box is displayed.
- *Document list* : the menu DOCUMENTS finishes by the list of all the documents currently loaded in CADIXE. By selecting one of these items the user can change the document associated to the current window.

TOOLS menu

- The five first items provide the list and the state ("show/hide") of the palette tools that can be displayed at the top/right part of the interface. When the user changes this display his/her choices are automatically stored in the preferences of the editor and reused at the next run. Let's notice that the relative positions of the palettes can be changed thanks to the preference dialog box. The tools are the following :
  - The toolbar containing the icons of the most common commands
  - The list of possible tags (with respect to the cursor position)
  - The attributes of the current tag
  - The comment tool (*can be also a dialog box, menu EDIT*)
  - The search tool. (*can be also a dialog box, menu TAG*)

  According the current task, the user can use one or several palettes. For instance, during the annotation (s)he mainly uses the tools "tags" and "attributes", but when verifying an existing annotation (s)he mainly uses the tools "search" and "comments".

- XML Validation of the Document : this command runs the XML validation of the annotated document. The errors are displayed in the panel "Validation result".

  When the user clicks on an error, CADIXE automatically sets the cursor on the corresponding part in the document. Furthermore, by double-clicking on an error the editor also selects the panel "XML code". In the XML code, the faulty tags have a red bullet. By double-clicking on this bullet the editor switches back to the list of error (panel "Validation result"). There are three kind of validation error :
  - Warning : these errors generally point out a problem in the attributes of the tag : missing value, erroneous value, … If this error frequently occurs the user should activate the flag "*Display a dialog to fill the attributes* " (cf in

preferences, Attribute options) which forces the user to enter a value in the attributes of the inserted tag.

- Simple error : this is the most frequent error corresponding in CADIXE to a missing tag in the document with respect to the DTD. If this error frequently occurs the user should activate the mode *"Allows to insert the most specific tag"* (cf in preferences, Tag options) which forces the user to enter the tags in the same order they are defined in the DTD. In this way it becomes difficult to forget the intermediary level of tagging.
- Fatal error : this kind of error stops the validation process. It can occur for instance when an ending tag is missing. In principle, this kind of error should not appear in CADIXE due to the controls made during tag insertion.

HELP menu

In this release, this menu just allows to display the presentation screen of the application with the current release number.

Editor Settings

The command line

When CADIXE is launched via a terminal, the command line can include the following options to modify the default behaviour of the application :

-userhome *path* : by default the editor stores its working directories at the same place that the runtime. This argument allows to indicate the path containing the "preferences file" and according to its content (Cf. "preferences settings") to change the behavior of the system. For instance, with Unix, the user could put the variable $HOME in the path if he/she wants that data management is done with respect to its login place.

At the runtime, the missing files are automatically copied from the application to the current "userhome". In this way, it is convenient to distribute a given configuration to all the users. Here is an example (be cautious, the folder CADIXE must exist in this case)

java -jar /CADIXE.jar –userhome $HOME/CADIXE

-prefs*name* : this argument allows to provide explicitly the name of the preferences file to load in the "CADIXE-preferences" folder. By default the name used by the system is "CADIXE.properties".

-load *filename* : to load a given file that can be either a text file or an annotated one. When it is a text file, the editor uses the DTD and style sheets indicated by the default model file (cf. "preferences settings").

-name *string* : open a new document (based on the default model) with this name.

-model filename : to provide the name of the default model to use in the parameters –load and –name instead of the one given in the preferences file.

Preference settings

The preferences dialog box (EDIT menu) allows to set some general behavior of the editor and to adapt it to the user. Here are the possibilities founded in each topic :

- *Default paths* : allows to indicates the directories in which the editor must find (and store) the different resources such as :
  - The name of the default model file uses to create a new document
  - The path to the model folder
  - The path to the DTD folder
  - The path to the style (CSS) folder
  - The path to the document folder
  - The path to the script folder (not used in this release)

  When a path is *relative*, the reference folder can be either the application (the default) or the path that has been provided through the argument *–userhome* of the command line. The selection of this behavior is done thanks to the radio-button.

In this way, it is easy to set-up the editor such that shared resources (i.e. DTD, models, styles, …) would be stored in the application folder, while specific resources (i.e. documents, preferences, …) would be stored in user folder.

- *Editor* : in this topic the user can set the tools' positions in the right part of the interface (see also the menu TOOLS to turn on and off the display). The other options concerns : the facts that a new document is automatically create when the editor is launched, the maximum number of documents simultaneously displayed in the mode "multi-documents" (see DOCUMENTS menu) and the number of files that are memorized in the list of "Load Recent" (see FILE menu).

- *Annotation* : in this topic the user can select if the selection in the document must be automatically aligned on the boundary of the closest word (default). Indeed, the tagging is generally done at the level of the word rather than the characters.

- *XML zone* : this topic allows to select the font and the size of the characters used to display the XML tag. It is also possible to define the color and the style (plain, bold, italic) associated to the tags, attribute, value and excerpt at the right of the tag.

- *Attributes* : the user can customize the way the attributes' values are entered and controlled when a new tag is inserted in the current document. Here are the options :

  - *Display a dialog to fill the attributes:*
  Allows to choose if a dialog box must appears to force the user to fill the attributes, there are three options in the pop-up menu :

    - "Never" : the attributes are classically set through the ATT tools.
    - "Always" : a dialog box always appears (if needed) to set the values.
    - "When REQUIRED": the dialog box appears when there is at least one attribute of type REQUIRED in the current tag.

  - *Verify if REQUIRED attributes are set :*
  When the display of a dialog box is set (see above) this option forces the user to put a value in the REQUIRED attributes. The dialog box can be closed only if the corresponding fields have been filled.

  - *Allow to suppress the attributes :*
  When this option is activated a "check-box" is displayed at the left part of each attribute : it enables to (des)activate the generation of a given attribute in the XML code. This option allows to simplify the annotation when some attributes are locally irrelevant. When the option "*Always include required attributes"* is set, the REQUIRED attributes cannot be removed (default).

- *Tags* : this topic contains the options allowing to control the display of the list of tags that can be inserted in the document. Here are the possibilities :

  - *Allows to insert all the specific tags :*
  When this check-box is deactivated (the default) the editor only displays the tags that are *immediately* more specific (i.e. the direct children in the DTD) than the current tag. Instead, if it is checked all the possible tags are displayed

The first mode provides few freedom to the user but is well adapted to guide those who are unfamiliar with the current DTD (or more generally not aware about the constraints of XML) and allows to reduce dramatically the number of validation errors (see TOOLS menu). The second mode is less restrictive and is interesting for the "power users" that want to insert the tags in any order.

▪ *Display relevant tags only:*
When this option is activated (the default) only the suitable tags (with respect to the current selected fragment) are displayed in the list. If it is deactivated, the unrelevant tags are greyed. The second option could be interesting to use when the number of tag described in the DTD is small.

▪ *The tags are validated by default :*
This option is not already used in the current release.

- *Toolbar* : this topic allows the user to customize the content of the toolbar. There are two columns : the left one contains the list of commands displayed in the toolbar, the right one list all the commands existing in the editor. The button "Delete " suppresses the selected command from the leftmost list, the button "Add" add the selected command from the rightmost list to leftmost one, and the button "Separator" inserts a separator in the leftmost list. The arrangement of the commands in the toolbar can be changed by "drag & drop" in the left list.

Acknowledgments and Contact

Acknowledgements

The CADIXE XML Annotation Editor has been initially developed in the frame of the CADERIGE project (http://caderige.imag.fr/) founded by the French government through the "ACI Bioinformatique". The goal of this project was to extract the Gene Interaction Network described in the Medline abstract by combining Natural Language Processing (NLP) and Machine Learning (ML) approaches. The software has been written by :

- Version 1 : G. Bisson, P-E Gros et A. Morin.
- Version 2 : G. Bisson et H. Zidelkheir.

Many tanks to the people of MIG group from INRA (Institut National de la Recherche Agronomique) for their active support and also to the people from SIB (Institut Suisse de Bioinformatique) who collaborate to this development.

Contact

For any further information (and to download the software), please contact Gilles Bisson (gilles.bisson@imag.fr). If you want to report a bug, please include as far as possible your data (annotated documents, DTD, style sheet, …) and a small textual description of the behaviour of the software, so that we can reproduce (and perhaps fix ☺) this bug. Thanks !