DELT/A Tutorial

Tutorial for the
Document Exploration and Linking Tool

2.5.2

Peter Votruba, Katharina Kaiser, Silvia Miksch

Vienna University of Technology
Institute of Software Technology and Interactive Systems
Information & Software Engineering Group

Dec-2005

Table of Content

Table of Figures

This tutorial covers two topics:

1. The basic features, illustrated by using DELT/A in a multi-step transformation process of a clinical guideline
2. Advanced features regarding visualization

For a complete reference of all features, menus and toolbars, see the DELT/A user manual (available at the DELT/A project site http://ieg.ifs.tuwien.ac.at/projects/delta/).

1 Introduction

DELT/A has been developed to support knowledge engineers in transforming plain-text clinical guidelines (available as HTML files) to an XML-based guideline representation language by providing two main features:

1. **Macros**: a macro combines one or more XML elements together with their attributes. When inserting a macro into an XML file, its content is inserted instead. Using an appropriate macro file containing well-sorted macros based on often used patterns will facilitate building up a new XML document.
2. **Links**: DELT/A enables to create and maintain connections between two documents to relate corresponding parts. Such links can be easily inserted using special macros.

Parts of DELT/A user interface:

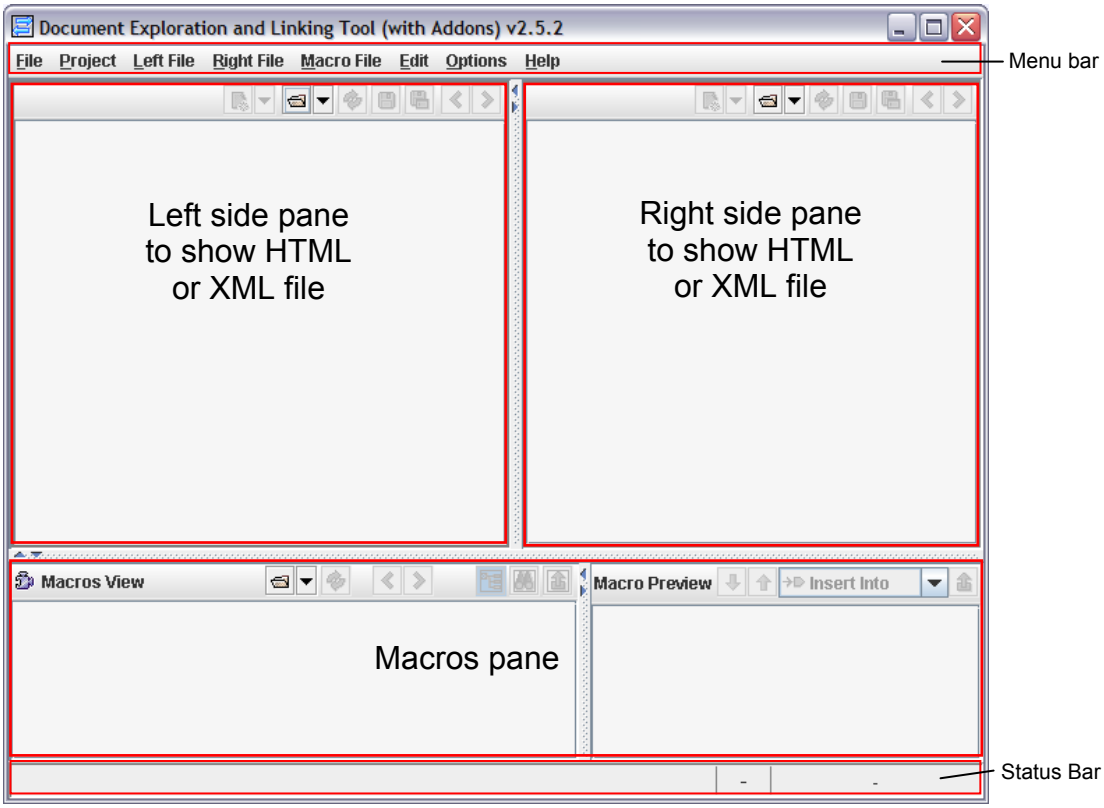

Figure 1: User interface of DELT/A.

The user interface of DELT/A consists of three main parts: in the upper half of the window are two equal panes side by side, both capable of showing HTML or XML documents. Below is the macros pane that consists of the Macros View, which shows the available macros, and the Macro Preview that shows the content of the currently selected macro.

2 DELT/A Supporting Multi-step Transformation Process of Clinical Guidelines

Due to its two main features – links and macros – DELT/A is well suited to be used for transforming a guideline (available in HTML) to a formal representation (defined as an XML language). This part of the tutorial describes step-by-step the usage of DELT/A in a two-step transformation process. The first transformation step consists in transforming the guideline to an intermediate representation and within the second step the intermediate document is further formalized resulting in a document of the target representation.

Transformation Step 1

Transformation of the guideline to an intermediate representation

1 Setting Up Files

To accomplish this first step we need three files:

1. Open HTML file at the left pane of DELT/A:

   To open an HTML file choose Left File → Open. A standard file chooser dialog (see Figure 2) will appear. Make sure that 'HTML Files' is selected in the 'Files of Type' combo box and select the HTML file that should be the source of the transformation process.

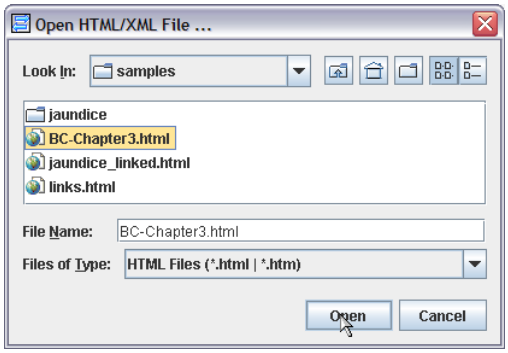

Figure 2: File chooser dialog for opening HTML files.

2. Create a new intermediate XML file in the right pane of DELT/A:

   If there is already an XML file of the intermediate representation's XML language opened in the right pane, choose Right File → New empty XML file.

   If you want to create a new file of another XML language, choose Right File → New from other XML-Lang to select the desired XML language.

   Note: the name of the XML language of the current XML document can be seen on the toolbar on the left next to the file name (see Figure 3).

Figure 3: Toolbar showing XML language and file name

   In both cases, a file chooser dialog appears where you can choose the location and enter the name of the new XML file (see Figure 4).

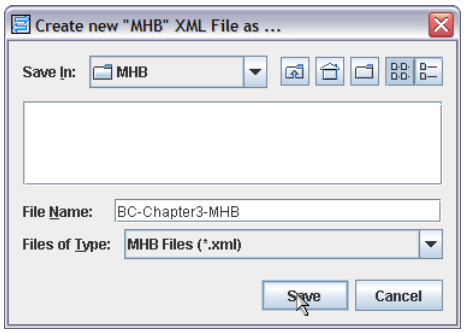

Figure 4: File chooser dialog for creating a new XML file of the actual XML language.

3. Open appropriate Macro file:

   To open a macro file for the intermediate XML language, select Macro File → Open. A file chooser dialog (see Figure 5) appears where you can select an appropriate Macro file.

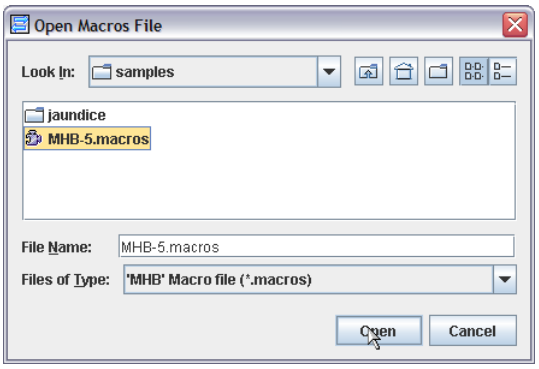

Figure 5: File chooser dialog for opening a macro file.

→ At this point, we see the original HTML file opened in the left side pane and the just created empty intermediate XML file opened in the right side pane and the macro file for the intermediate XML language (see Figure 7).

2 Use Macros to Construct an Intermediate XML File

Macro files contain macros in a hierarchical structure with macros as leaves.

We can differentiate between two types of macros: (1) macros containing a link ( ) and (2) macros without a link ( ).

If a macro contains a link ( ), the other link endpoint has to be selected in the left pane before inserting the macro (as can be seen in Figure 7). Macros, which are not allowed to be inserted into the current XML element, are marked by ■ (i.e., instead of and instead of).

To proceed select an appropriate macro (or) in the macros view (see Figure 6) and click on the macro-insert button ( ) on the right side of the macro pane as shown in Figure 7 (before inserting) and Figure 8 (after inserting). Alternatively, you can change the insert mode (combo box left to the insert button) from "Insert Into" to "Insert After", "Insert Before" or "Replace".
Navigating in the macros view is similar to navigating in a tree view: the items with the icon (or) are folders that can contain macros and sub-folders; if such a folder is selected, its content is shown in the list box to the right.

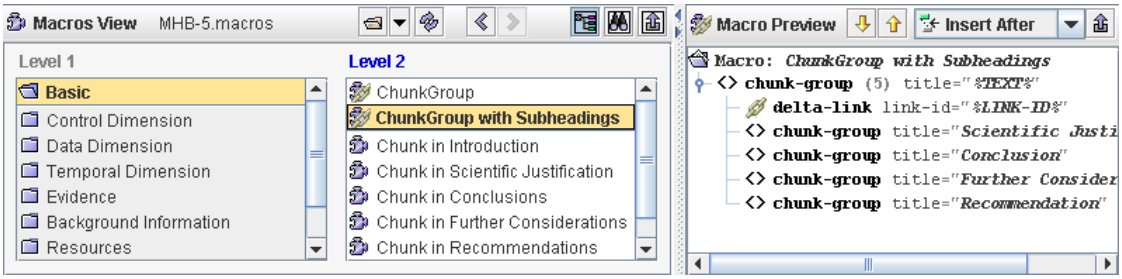

Figure 6: Macros View showing macros for an intermediate representation.

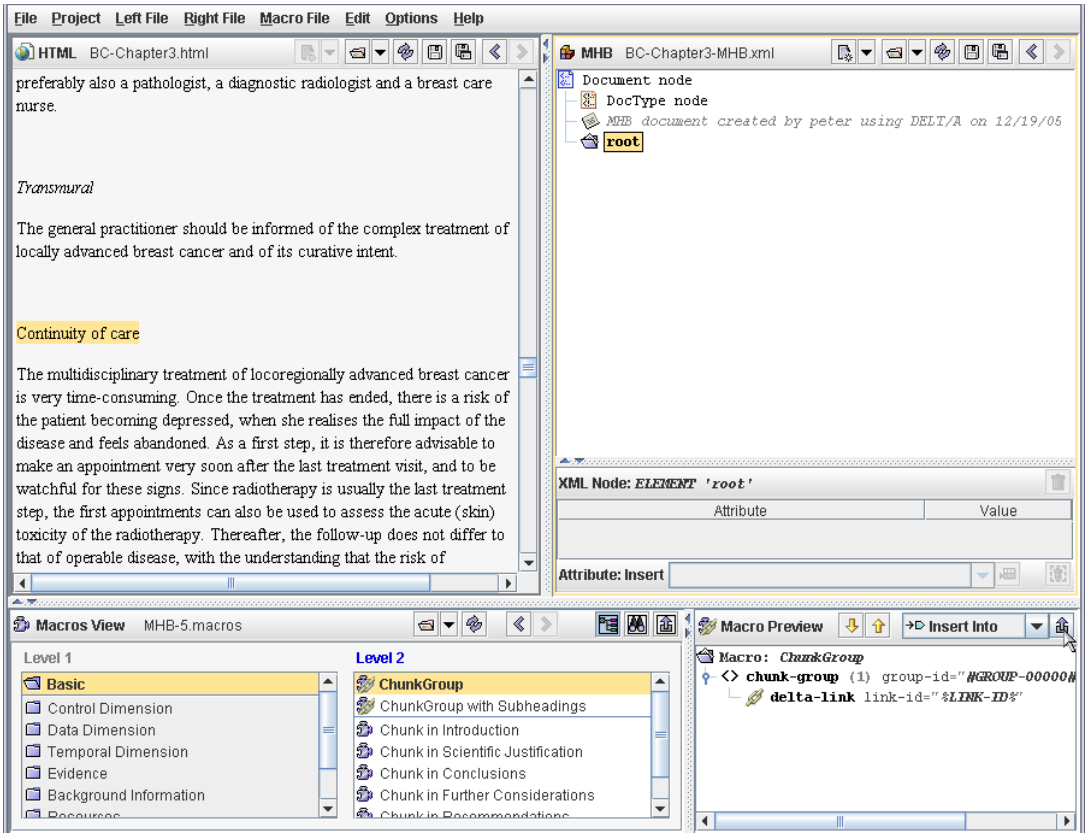

Figure 7: Inserting a macro containing a link. In the left pane text is selected, in the ‚Macros View' the macro 'ChunkGroup' is selected which will be inserted at the actual position in the right pane.

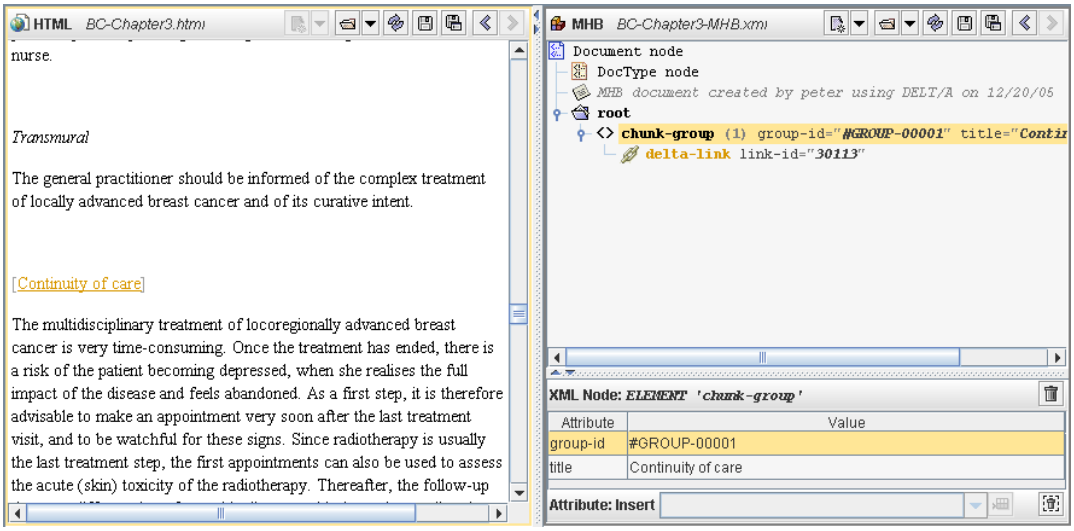

Figure 8: HTML and XML documents after inserting a macro.

Since some macros require user input, an input dialog may appear where one or more attribute values can be entered (see Figure 9).

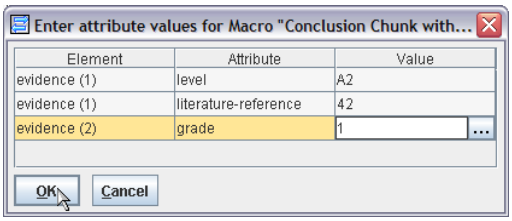

Figure 9: Attribute input dialog when inserting a macro.

Keep inserting macros (and creating links) until all relevant knowledge of the source document is linked and thus modeled in the target document in the right pane.

Note: *feel free to play around with the macros – every action you take (e.g., inserting macros) can be easily undone using Edit → Undo.*

3 Save Resulting Files

To save both files – the HTML file enriched with DELT/A links and the new XML file – just use File → Save all.

→ This is the end of the first transformation step: the result of this step is the guideline in the intermediate representation format linked to the original guideline. Now we can start our second transformation step. Note that the source document of this transformation step will be the just created intermediate XML file.

Transformation Step 2

Starting from the intermediate representation we will now show how to proceed with the second transformation step.

1 Setting Up Files

Again, we need three files for accomplishing the second transformation step:

1. If the intermediate XML file is open in the right side pane, you may choose File → Swap sides to get it on the left side of DELT/A – to retain the left-to-right concept. Otherwise, you may choose Left File → Open to choose the source file of this transformation step.
2. Create new, empty target XML file at the right side.
3. Open appropriate macro file for target XML language.

→ At this point, we see the previously constructed intermediate XML file opened in the left side pane, the just created empty target XML file opened in the right side pane and the macro file for the target XML language in the Macro View pane (see Figure 10).

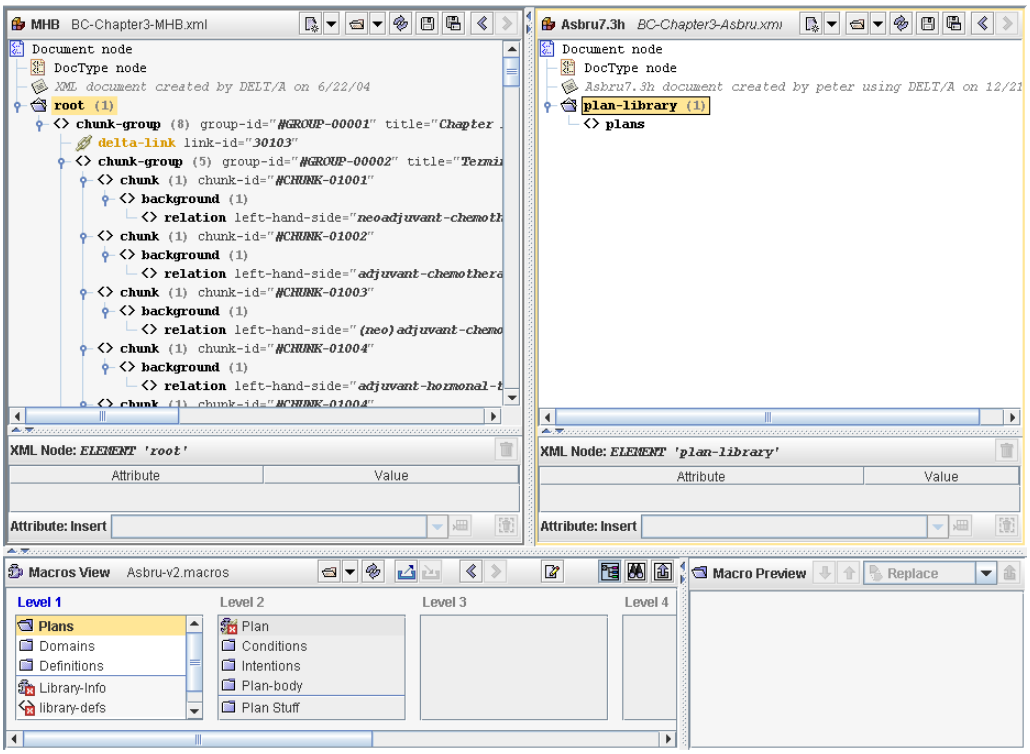

Figure 10: DELT/A window at the beginning of the second transformation step.

2 Use Macros to Build the Target XML File

Proceed like in Transformation Step 1.

Keep in mind that this XML file should be linked to the two other files. To accomplish this, select an already existing link (`delta-link` element with icon) in the left pane when inserting a link macro. Thereby, the new link endpoint in the right pane will be connected to the existing link in the left pane.

3 Save Resulting Files

Proceed like in Transformation Step 1.

→ This is the end of the second transformation step and thus also the end of the entire transformation process: the result of this step is the guideline in the target representation format linked to the intermediate file and to the original guideline.

Working With All Documents

Now you can start using the links between any two of the three HTML/XML documents.

To use the links between two files, just click on a link – dark-orange underlined text in HTML view or a `delta-link` element in the same color in XML view – and all the corresponding links in the other side will be highlighted (see Figure 11 for an example).

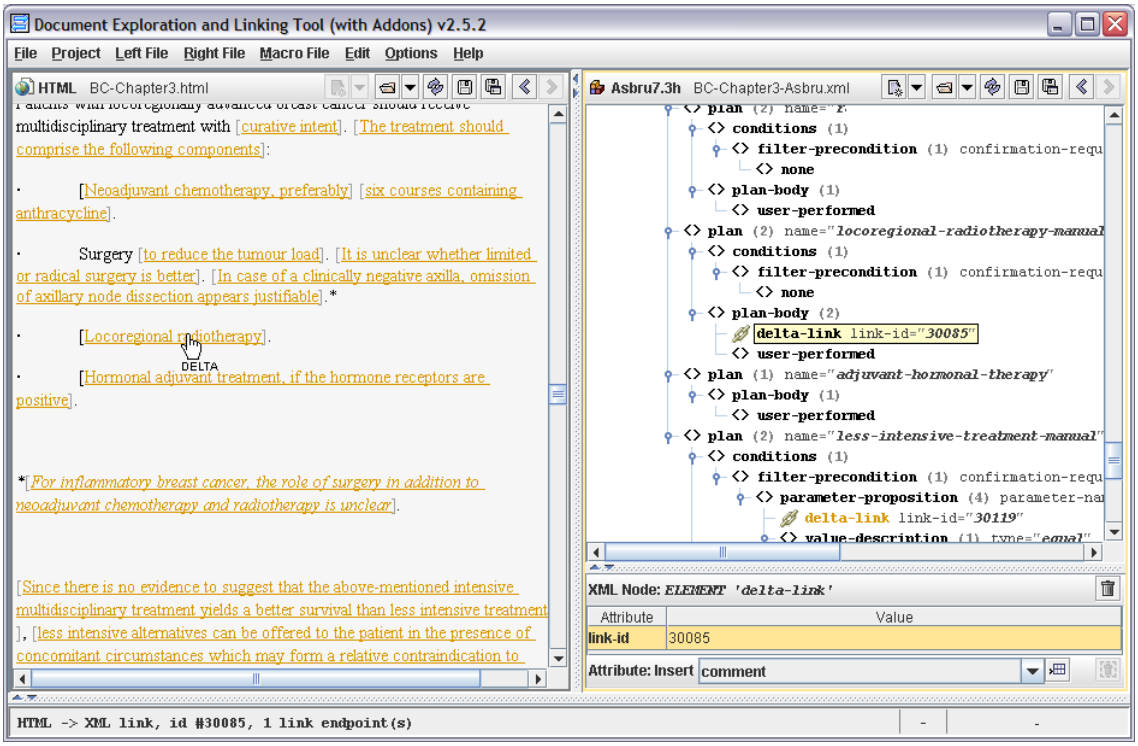

Figure 11: Activating DELT/A link pointing from HTML to XML file.

3 Usage of DELT/A Within the Framework of a Visualization Task

In addition to the basic features described in the previous section, DELT/A provides several advanced features whereas some of them can be seen as visualization features in the sense that they provide a different view of an XML document.

Note: all these features are only available, if DELT/A is currently running in the "Advanced Mode". To ensure this, go to the Preferences dialog (Options → Preferences) and check the "Advanced Mode" checkbox. After changing the mode, DELT/A has to be restarted.

---

1. Highlighting – mark certain XML elements:

This feature allows highlighting a specified set of XML elements. The rules that define which elements should be highlighted are expressed using XPath. Such XPath-Expressions can be pre-defined for each XML language to be easily invoked as shown in Figure 12.

Figure 12: Invoking pre-defined highlighting.

Figure 13 shows the result of invoking the highlight definition selected in Figure 12.

Figure 13: Highlighting all *plan* elements.

2. *Filtering* – hiding or showing certain XML elements:

This feature – accessible through the Applicable "Views" → Filter sub-menu of the context menu – can be used to show or hide specific nodes. Aside from standard filters to toggle the visibility of all types of XML nodes (except elements), XPath expressions can be used to show or hide any arbitrary set of XML nodes. Additionally, custom XPath filters can be defined for each XML language and recalled later.

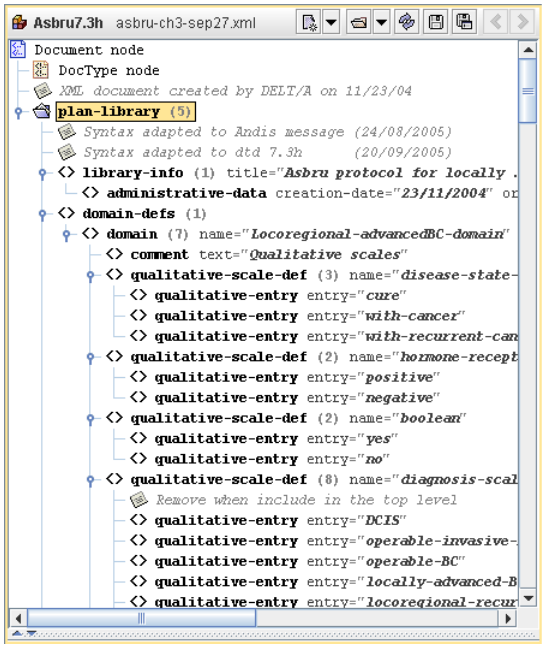

Figure 14: XML view with no filter applied (all elements are shown).

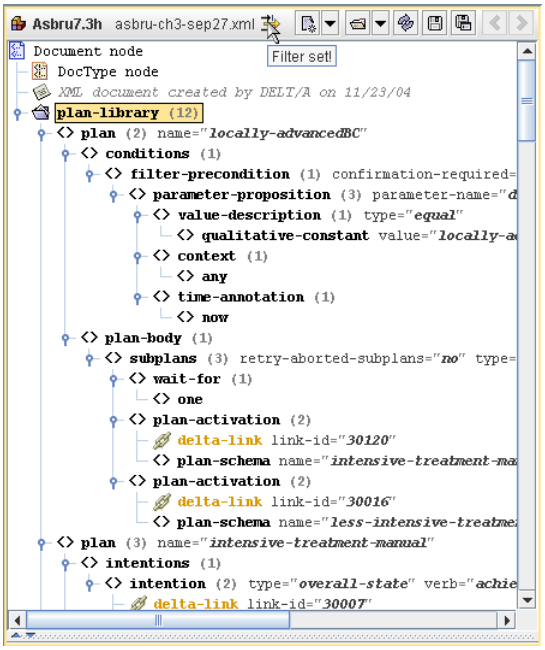

Figure 15: XML view after applying a filter (only "plan" elements are shown).

3. *Folding* – collapsing or expanding certain XML elements:

The term "Folding" means expanding or collapsing one or more XML tree nodes. There are actions for expanding or collapsing all nodes of the current document, expanding or collapsing only the nodes of the current sub-tree in the Applicable "Views" → Folding sub-menu of the context menu. Additionally, there are two menu items that can be used to

expand or collapse only specific nodes using XPath. Such folding XPath expressions can be pre-defined for each XML language.

4. Summarize – shows summary of content of collapsed elements:

The idea behind this feature is that if an element tree-node is collapsed, the user has apparently no information about its content (i.e., its sub-nodes). The solution is to display short information ("summary") about the sub-nodes instead of the attributes in the tree-node label of certain elements. XSL files are used to store the rules which information should be shown for which collapsed element. A list of such XSL files can be pre-defined for each XML language and easily invoked through the Applicable "Views" → Summarize sub-menu of the context menu.

Figure 16 and Figure 17 shows an example of this feature: when the element is expanded the attributes are shown beside the element name as usual (see Figure 16), whereas when the same element is collapsed summed up information about the content of the element is shown instead (see Figure 17).

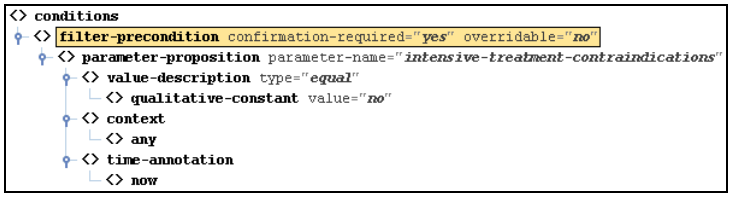

Figure 16: Summarize example with expanded element.

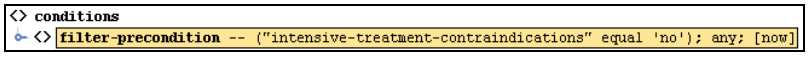

Figure 17: Summarize example with collapsed element.

5. Generating reports using XSLT scripts

DELT/A provides the possibility to transform the current XML document to an HTML report by applying an XSLT script. Such XSLT scripts can be pre-defined for each XML language and then invoked through the Applicable "Views" → Apply Transformation sub-menu of the context menu. Figure 18 shows invoking one of the XSLT scripts for the *Asbru* XML language that are included in the DELT/A package.

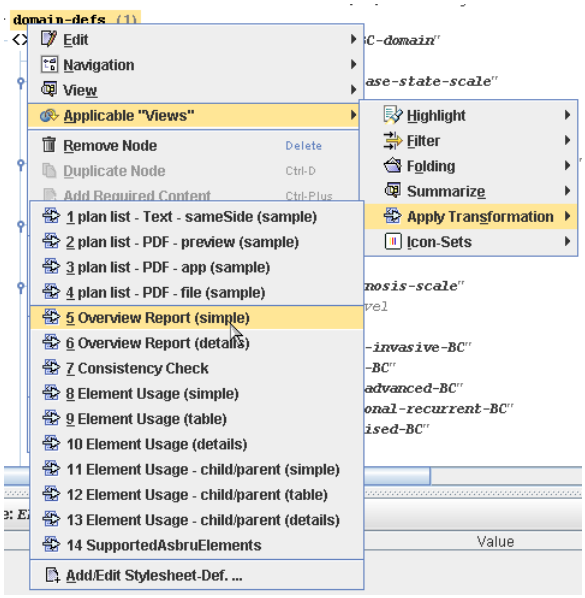

Figure 18: Applying an XSL transformation.

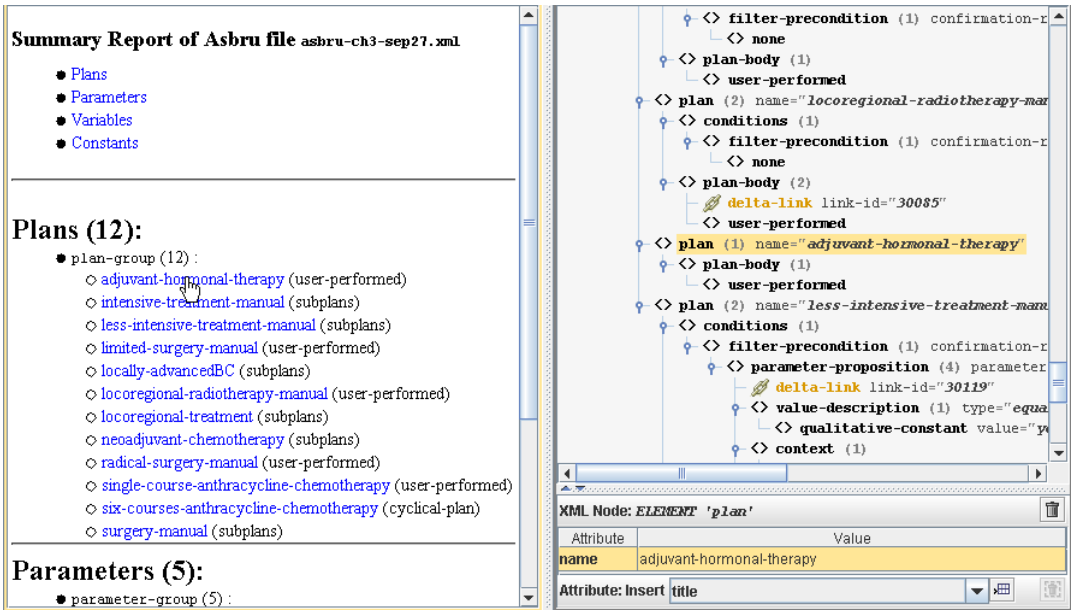

Figure 19: Resulting HTML report (left side) shown after applying XSLT script.

An XSLT script that generates an HTML report may incorporate special links that would induce the XML view to select certain elements. See Figure 19 for an example: after clicking on the list-entry “adjuvant-hormonal-therapy”, the XML View selects the plan element with the same name.

DELT/A comes already with several useful XSLT scripts that generate HTML output, which is shown in the other side pane than the source XML document.
The following six general XSLT scripts can be used for XML documents of any XML language (e.g. *MHB* or *Asbru*):

- Element Usage (simple): generates an HTML report that provides a simple list of names of all used elements in this document with the count of their occurrences, see Figure 20 for an example
- Element Usage (table): same content as "Element Usage (simple)" but formatted as an Excel-compatible HTML table, see Figure 24
- Element Usage (details): similar to "Element Usage (simple)" but with links to each occurrence of every element plus an index of all elements, see Figure 22
- Element Usage - child/parent (simple): same as "Element Usage (simple)" plus showing for each element where it occurs, see Figure 21
- Element Usage - child/parent (table): same content as “Element Usage - child/parent (simple)” but formatted as an Excel-compatible HTML table, see Figure 25
- Element Usage - child/parent (details): similar to "Element Usage - child/parent (simple)" but with links to each occurrence of every element plus an index of all elements, see Figure 23

For the XML language *Asbru*, there are 3 additional XSLT scripts included:

- Asbru Overview Report (simple): generates an HTML report that provides an overview of all plans, parameters, variables and constants defined in the current Asbru document (all listed items are linked to their occurrence in the XML view), see Figure 19
- Asbru Overview Report (details): same as "Asbru Overview Report (simple)" but showing also where each plan, parameter, variable and constant is used, see Figure 27
- Asbru Consistency Check: generates an HTML report showing many different kinds of (possible) problems found in the current Asbru file, see Figure 26

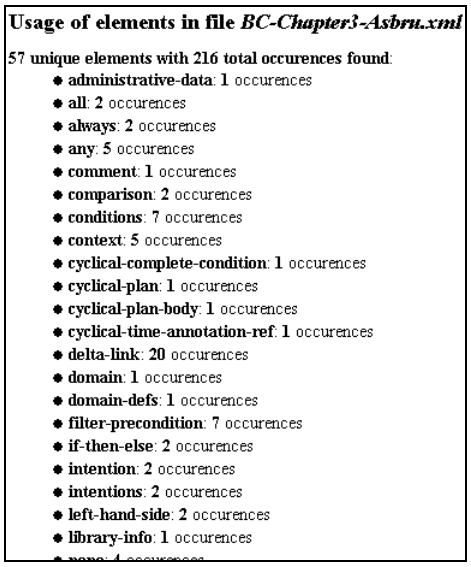

Figure 20: Result of "Element Usage (simple)" XSLT script

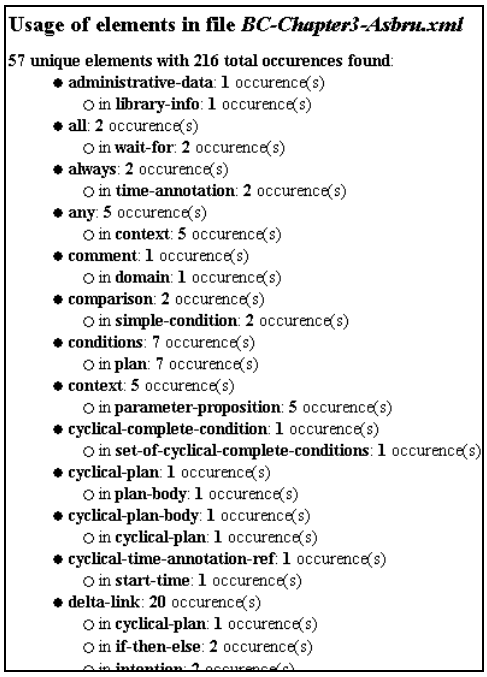

Figure 21: Result of "Element Usage - child/parent (simple)" XSLT script

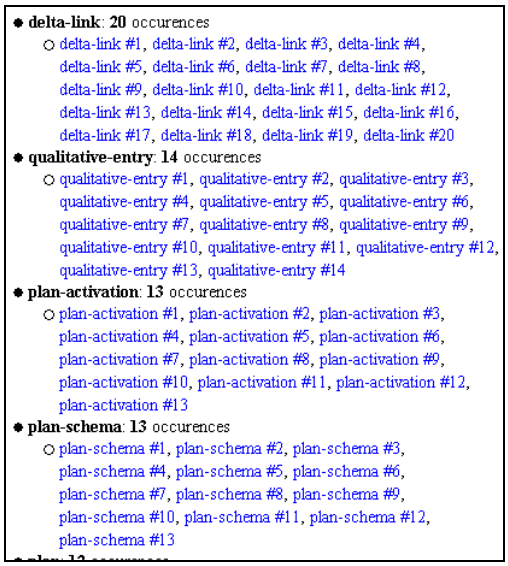

Figure 22: Result of "Element Usage (details)" XSLT script

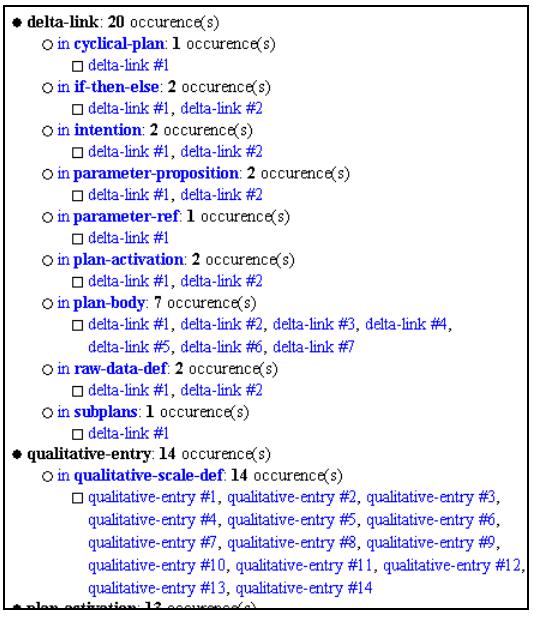

Figure 23: Result of "Element Usage - child/parent (details)" XSLT script

Usage of elements in file BC-Chapter3-Asbru.xml	
Element	**Occurences**
administrative-data	1
all	2
always	2
any	5
comment	1
comparison	2
conditions	7
context	5
cyclical-complete-condition	1
cyclical-plan	1
cyclical-plan-body	1
cyclical-time-annotation-ref	1
delta-link	20
domain	1

Figure 24: Result of "Element Usage (table)" XSLT script

Usage of elements in file BC-Chapter3-Asbru.xml			
Child-Element	**Occurences (total)**	**Parent-Element**	**Occurences**
administrative-data	1	library-info	1
all	2	wait-for	2
always	2	time-annotation	2
any	5	context	5
comment	1	domain	1
comparison	2	simple-condition	2
conditions	7	plan	7
context	5	parameter-proposition	5
cyclical-complete-condition	1	set-of-cyclical-complete-conditions	1
cyclical-plan	1	plan-body	1
cyclical-plan-body	1	cyclical-plan	1
cyclical-time-annotation-ref	1	start-time	1
delta-link	20	cyclical-plan	1
delta-link	20	if-then-else	2

Figure 25: Result of "Element Usage - child/parent (table)" XSLT script

Consistency Check of Asbru file

BC-Chapter3-Asbru.xml

Content: Plans, Parameters, Variables, Arguments, Return-Values, Constants, Types, Units, Contexts, Misc. Stuff

Plans:

- Undefined Plans : **none**
- Uncalled Plans - if 0 or more than one: **one**
- Duplicate Plans : **none**

Parameters:

- Undefined Parameters : **none**
- Circular References in domain part : **none**
- Unused Parameters - no parameter-ref's or parameter-proposition's: **none**
- Duplicate Parameters : **none**

Variables:

- Undefined Variables : **none**
- Referenced, but never assigned variables (1):
  - tumour-size-history
- Assigned, but never referenced variables : **none**
- Unused Variables - defined, but never referenced or assigned: **none**
- Duplicate Variables : **none**

Figure 26: Result of "Asbru Consistency Check" XSLT script

Summary Report of Asbru file BC-Chapter3-Asbru.xml

- Plans
- Parameters
- Variables
- Constants

Plans (12):

- plan-group (12) :
  - adjuvant-hormonal-therapy referenced by
    1. plan-schema, inside plan intensive-treatment-manual
    2. plan-schema, inside plan less-intensive-treatment-manual
  - intensive-treatment-manual referenced by
    1. plan-schema, inside plan locally-advancedBC
  - less-intensive-treatment-manual referenced by
    1. plan-schema, inside plan locally-advancedBC
  - limited-surgery-manual referenced by
    1. plan-schema, inside plan surgery-manual
  - locally-advancedBC **not** used
  - locoregional-radiotherapy-manual referenced by
    1. plan-schema, inside plan locoregional-treatment
  - locoregional-treatment referenced by
    1. static-plan-pointer, inside plan intensive-treatment-manual
    2. static-plan-pointer, inside plan less-intensive-treatment-manual
    3. plan-schema, inside plan intensive-treatment-manual
    4. plan-schema, inside plan less-intensive-treatment-manual

Figure 27: Result of "Asbru Overview Report (details)" XSLT script