April 1997

Building Amaya: Its Architecture

Irène Vatton

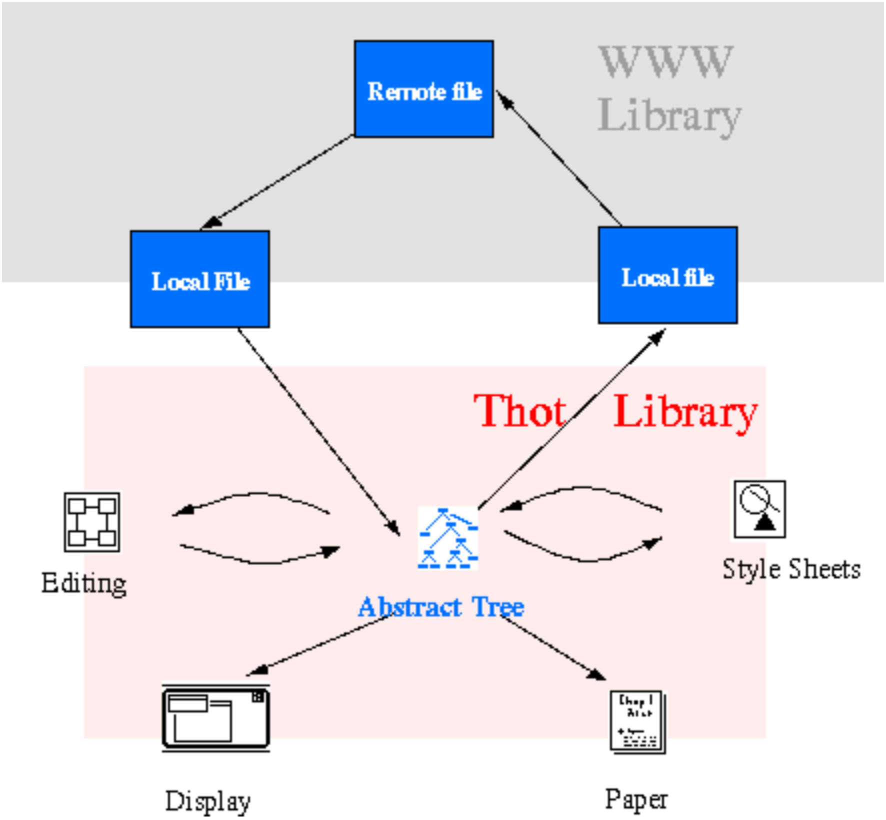
W3C / INRIA

Main Features

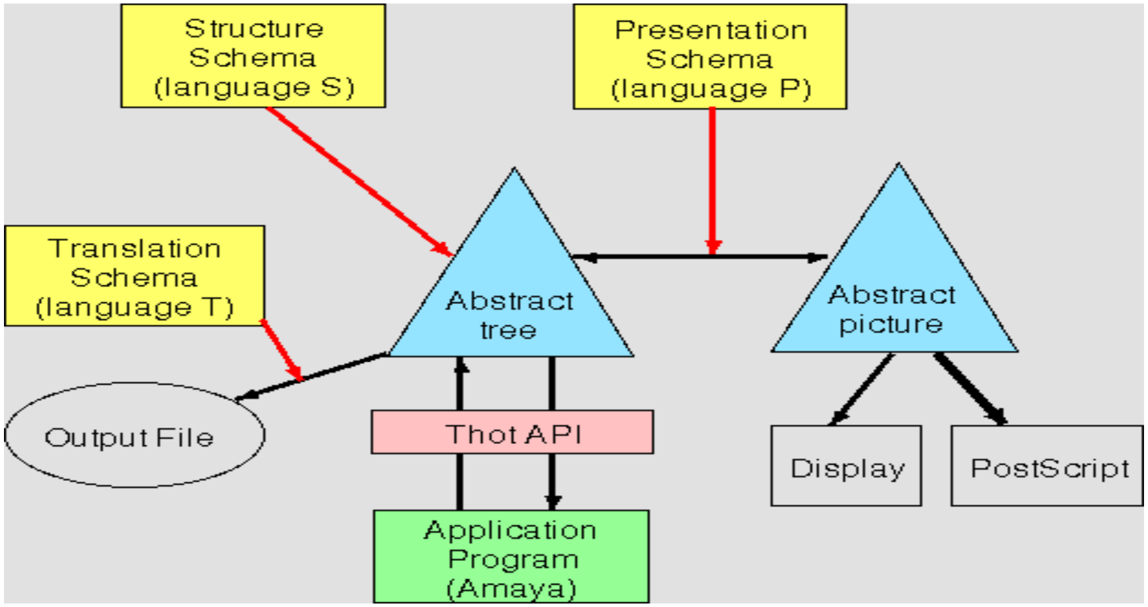
- Combine structure-driven editor and Web client
- Presentation based on logical structure and style
- WYSIWYG user interface with multiple views
- Multiple external formats
- Multilingual dialogue
- Extensible tool

Global Architecture

Two different parts: Web Access and Structure-driven Editor



Thot Architecture


Structure Schema (language S)
Presentation Schema (language P)
Translation Schema (language T)
Abstract tree
Abstract picture
Output File
Thot API
Display
PostScript
Application Program (Amaya)

Structure Schema

- Like a SGML DTD, a structure schema defines the structure of a document type
- Every editing command is controlled by the structure schema

  The logical structure is always correct

- Several structure schemas can be used simultaneously

Presentation Schema

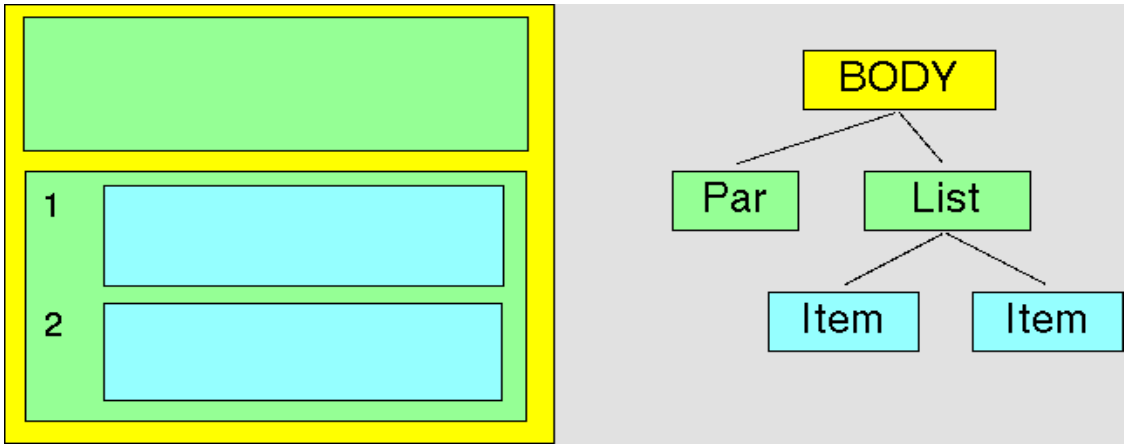
- Like a style sheet, a presentation schema defines the layout and style of a document type
- Documents are formatted by the editor according to
  - their specific logical structure
  - the current presentation schema
- A presentation schema may have extensions with different priorities (cascade)

Box Model

Every element in the document structure has a corresponding box in the abstract picture

Boxes are nested according to the logical structure

Geometrical constraints between boxes define their position and size



Translation Schema

The external syntax of documents is defined by a translation schema

Different translation schemas can be associated with a document type

A translation schema specifies the syntax for each

- element type
- attribute

depending on the structural context

Thot API

- An application program can call all internal functions through an API
- The API accesses only the abstract tree (logical structure)
- The Thot library is in charge of formatting, displaying and printing.

See the document **APIman.html**

Application Generation

Thot provides an application generator:

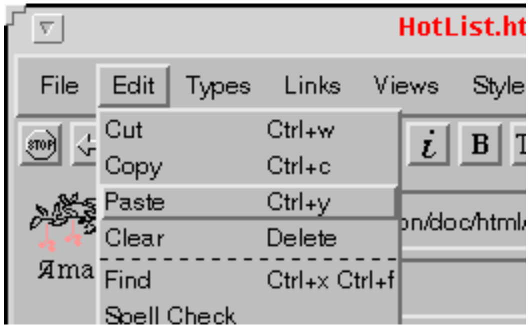
- Declaration of cascading menus (EDITOR.A)
- Declaration of callbacks (HTML.A)

The interface compiler generates the application skeleton and the developer has just to insert specific code (C code or Java code).

Cascading Menus

The Amaya dialogue is defined in a declarative form in File: EDITOR.

```
MENUS
HTML Windows:
BEGIN
Edit button:BCut -> TtcCutSelection;
Edit button:BCopy -> TtcCopySelection;
Edit button:BPaste -> TtcPaste;
Edit button:BClear -> TtcDeleteSelection;
Edit Separator;
Edit button:BFind -> TtcSearchText;
Edit button:BSpellCheck -> SpellCheck;
END;
```



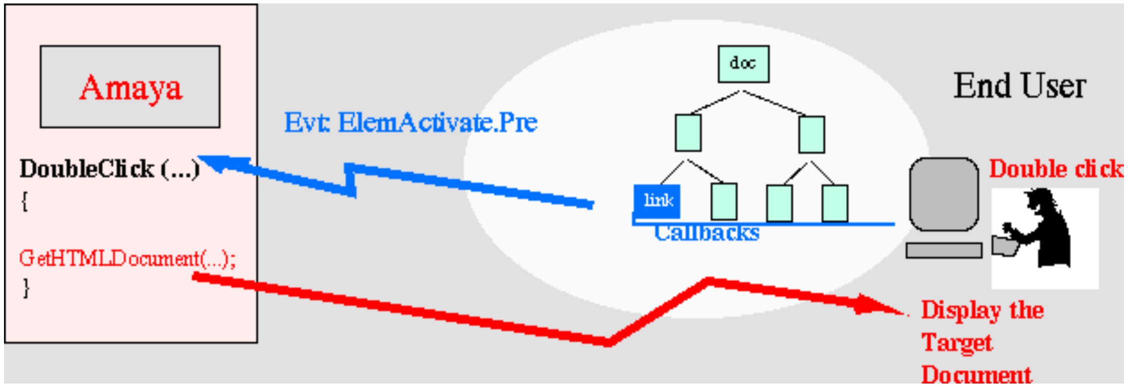
Callbacks

Callbacks allow an application program to be called on any editing action performed by the user

For each document type, a programmer can specify

- the elements and attributes of interest
- the editing actions of interest
- the function to be called

Example of links management:



The Parser

Transforms HTML documents into Thot Abstract Trees

- It reads tags and constructs the Abstract Tree step by step
- It is driven by an internal table and uses the Thot API
- It dynamically corrects structure errors:
  => add missing elements (for example **<Title>** without **<Head>**)
  => move misplaced elements

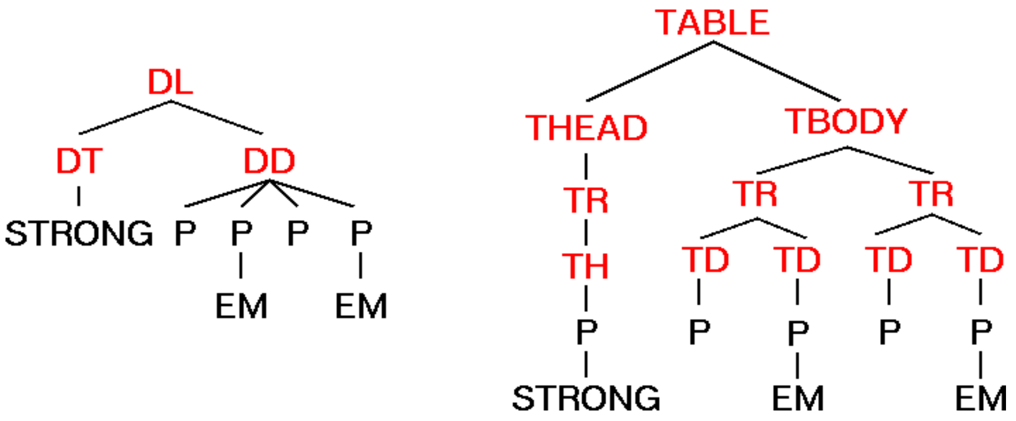
How to Extend ?

- Change the external table
- Update the parser code according to the logical structure

Transformations

Amaya controls the structure conformance It makes tree transformation while editing (cut, paste, transform):

File: HTML.trans



```
Table: DL{(DT|DD)+};
  { DT > <TABLE border=1>.TBODY:TR.TD;
    DD > TABLE.TBODY.TR:TD;}
```

File Management

Document loaded from the Web

- Create a directory for each remote document: **$HOME/.amaya/$docID**
- Store a local copy of the remote document
- Store local copies of remote images
- Add a copy when a new image (remote or local) is added into the document

Local document

- Use local files for local information (document and images)
- Store local copies of remote images