Fuzzy Rules for HTML Transcoding

Robbie Schaefer, Andreas Dangberg, Wolfgang Mueller
Paderborn University/C-LAB
Fuerstenallee 11
Paderborn, Germany
Robbie.Schaefer@c-lab.de, Andreas.Dangberg@c-lab.de, Wolfgang.Mueller@c-lab.de

Abstract

With the increasing availability of Web-enabled mobile devices, we are facing the problem to effectively adapt Web content for those devices. For adaptation, Web page structures require rearrangement, i.e., transcoding, to better fit on small displays when layout-specific HTML structures like tables and frames are used. In this article, we present a Fuzzy rule-based language, i.e., Fuzzy-RDL/TT. Fuzzy-RDL/TT (Fuzzy Rule Description Language for Tree Transformation) specifies sets of transcoding functions and defines their selection with respect to user and hardware profiles. The definition of profile selection is defined by the means of fuzzy rule descriptions where transcoding functions are specified by a Java-oriented language which operate on the DOM (Document Object Model) tree representation of an XML-based document.

1. Introduction

In contrast to the initial idea of HTML (Hyper Text Markup Language), layout-specific structures like frames and tables are meanwhile heavily used with the increasing application of authoring tools for Web page creation. However, Web pages with layout or pointing device-specific structures cannot be properly rendered for mobile devices like PDAs (Personal Digital Assistants) or smartphones with limited display sizes. Thus, these structures have to be rearranged or translated, i.e., transcoded, to different structures or to other markup languages like WML (Wireless Markup Language) which have been developed for those devices. Transcoding mainly depends on device profiles like screen size as well as user preferences and abilities. Table 1 gives a listing of some mobile devices with their categories and display resolution. It shows that each phone basically has an individual display size where in most cases sizes even vary between phone families. This makes it extremely difficult to find a general, multi-purpose transcoding scheme suitable for all devices which are available on the market.

Device	Category	Resolution (Pixel x Pixel)
Motorola Timeport P7389	Small Phone	96x40
Nokia 6185	SmallPhone	78x30
Sanyo C3045A	SmartPhone	128x112
NeoPoint 1000	SmartPhone	120x160
Sony CLIE	PDA	160x160
Psion Revo Plus	PDA	480x160
Psion Series 7	PDA	640x480
Siemens Gigaset CL4 SIMpad	Webpad	800x600

Table 1. Display Sizes and Categories of Some Mobile Devices

For determining which transformation fits best for which device, we will demonstrate in this article that fuzzy-based technology perfectly applies for this purpose. This is, since on the one hand, we have to determine which device approximately matches which category based on crisp values such as display sizes. On the other hand, we have fuzzy user requirements which have to be combined with other values to select transcoding rules. We present Fuzzy-RDL/TT (Rule Description Language for Tree Transformation). Fuzzy-RDL/TT is an extension of RDL/TT [9] by means of fuzzy set-based transcoding rule selection. RDL/TT is a language for specification of DOM (Document Object Model) tree manipulation which basically specifies conditional transformation rules which are assigned to HTML or XML tags. In Fuzzy-RDL/TT, conditions range over fuzzy sets. Crisp hardware properties like display sizes are subject to fuzzification where fuzzy user requirements can be directly integrated without transformation.

The remainder of this paper is organized as follows. The next two sections briefly discuss related works and outline

HTML applications for limited devices. Thereafter, the main section is dedicated to our transcoding approach. It outlines the role of user preferences and device profiles and introduces Fuzzy-RDL/TT. Section 5 gives some examples for an HTML to Compact HTML transcoding. After briefly sketching the implementation of our transcoding system, this article closes with conclusions and an outlook to future work.

2. Related Work

Existing transcoding approaches are limited to specific applications and are not well suited to define general transcoding rules between two languages. Currently, there exist two main efforts for the generic transcoding of XML-based documents. The first one considers transformations defined by XSLT templates. The second one is known as IBM's system for the annotation of HTML files.

The Extensible Stylesheet Language (XSL) [1] has been defined by the World Wide Web Consortium (W3C) for transformation of XML documents. XSL consists of two parts. The tree transformation (XSLT [2]) and the formatting of the transformed XML document. In our scope, those tree transformations are of specific interest since hereby XML documents can be converted into other languages which may differ in syntax and structure. A major condition is that the target language can be represented by a tree structure. The transformation from a source tree to the target tree is described by means of a stylesheet which consists of templates and transformation rules. By use of the XPath language [3], templates determine on which elements transformation rules are to be applied. XPath offers many possibilities for element selection. It is possible not only to select element types but also elements which appear in a specific context. Templates are computed recursively and are embedded in transformation rules which give the structure of the target document. Though, XSLT provides powerful means through recursive template definitions, stylesheets tend to get quite complex when they are used to describe complete transformation like transforming all possible HTML constructs to WML. So, XSL/XSLT is mainly applied to and suited best for matching and transforming of specific XML-based documents.

IBM's approach for annotation of Web content [5] defines the transcoding of specific files by annotating them with respect to given device profiles and user preferences by use of CC/PP (Composite Capability/Preference Profile) [10]. That approach is used to supply an HTML document with descriptions on how elements have to be changed in order to make this document renderable on other devices. Since these descriptions cannot be integrated into HTML documents (e.g., by means of additional tags or attributs) without violating the HTML standard, an external annotation file based on RDF [8] is used. An external annotation file gives the definition of matching templates by means of XPointer descriptions [7], which is an extension of the XPath language. Due to XPointers, annotations can be applied only to those files they are individually defined for. To use the annotation concept for generic transcoding, additional tools to generate the annotation files have to be employed.

The IBM approach employs the transcoding with respect to different user and hardware profiles given by CC/PP. CC/PP is an RDF-based data format which enables the exchange of device profiles and user preferences, which are related to the presentation of content. These profiles are stored in a tree structure with "profile" as root and the different components like hardware and operating system as child elements. With these parameters, a transcoding system can determine which modifications should be applied to the source document. CC/PP descriptions can be extended with a proxy component. This is useful when the transcoder is integrated into a proxy server rather than into the main content server. Since CC/PP is not bound to a specific transcoding method, it is an ideal exchange format for client and server information. We thus apply CC/PP in the context of our transcoding system.

Beside these two general efforts, there also exists an approach for the transcoding of XML-based user interface descriptions. VTT has implemented a system for the transcoding of user interfaces based on UIML (User Interface Markup Language) [6]. For transcoding description, they have defined an extension of the UIML peers section.

3. HTML for Limited Devices

Full HTML 4.0 is not well suited for application on mobile devices like PDAs and mobile phones are limited, e.g., in screen resolution. Moreover, since most HTML descriptions are designed for large displays and contain elements like framesets and tables which require at least the space of a PC monitor. So, other markup languages were developed to enable the use of Internet services on mobile limited devices: C-HTML (Compact HTML), WML, and MML (Mobile Markup Language).

Compact HTML is an HTML 4.0 subset which is dedicated to the limitations of mobile devices. Elements that do not meet the mobile device requirements like small display and low memory are excluded. C-HTML also excludes image maps, tables, and framesets. So the art of transcoding HTML to C-HTML considers not just replacing or deleting unsupported elements but also to restructure the document so that no information is being lost. One of the main advantages of C-HTML is that it is 100 % compatible with complete HTML so that it can be processed by any HTML compatible Web browser (but not necessarily every HTML ele-

ment can be rendered by a C-HTML browser). This made C-HTML very popular for PDAs. One example is the application within the iMode service in Japan.

WML is an XML-based language defined by the WAP (Wireless Application Protocol) Forum for microbrowsers on mobile devices with limited resources. WML is based on a combination of a HTML 4.0 subset and HDML (Handheld Device Markup Language) plus some extensions. In addition to HTML structures, WML is composed of a set of cards. Specific hyperlinks provide means to navigate between cards of one WML specification. Compared to C-HTML, WML differs in a couple of details, like in the support of bold character sets and subheadings (h1, h2, ...). Unfortunately, WML is currently not widely accepted and not well supported as an Internet Markup language. On PC, only very few browsers such as Opera support WML.

In addition to C-HTML and WML, the Japanese provider J-Phone has defined the MML (Mobile Markup Language) family of languages for their J-Sky service: S-MML (Small MML), M-MML (Medium MML), F-MML (Full MML). S-MML is compliant to a HTML 4.0 subset defined for small displays with 4 lines with 12 characters each. M-MML is defined for mobile phones with display sizes 160x100 pixel. F-MML stands for full compatibility to HTML 4.0.

In most cases transcoding systems are components of a proxy server which takes requests from a client, forwards them to a content server, and receives HTML documents in reply (cf. Figure 1). With a request, the client can also submit its profiles, for example, in CC/PP format. To reduce the amount of traffic, the client can send only a client- and user ID and profiles can be loaded from a different database where the proxy has access to. Having obtained the profiles, the transcoder converts an HTML file from the content server into the target language format. The processing takes two steps. In the first step, transformation specification is loaded. In a second step, the transcoding is applied to the HTML or XML file. Advanced transcoding systems compute the transcoding based on profiles, mainly considering user and hardware profiles.

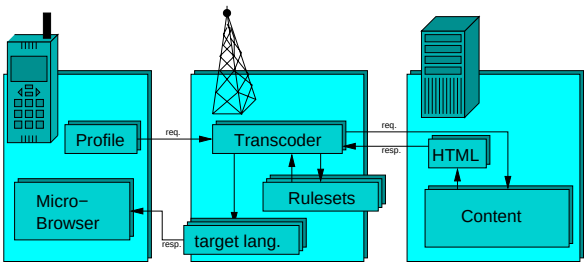

Figure 1. Transcoding System Architecture

4. A Fuzzy-Based Transcoding System

We introduce a fuzzy-based highly generic transcoding system which takes user and hardware profiles, fuzzy rule definitions, sets of transcoding specifications, and an XML-based document as input and generates a different XML-based document as output. The generated file format is due to the specific set of selected transcoding rules and can be an HTML subset, WML, VoiceML etc. The transcoding functions are generally defined, i.e., they are not due to a specific Web page structure and do not presume specific content in specific substructures such as it is done by other approaches given in Section 2.

Before introducing our transcoding approach, we first discuss the role of user requirements and abilities. Thereafter, we present our system by its individual components: fuzzy set definitions, fuzzy rule definitions, and transcoding definition.

4.1. Hardware Properties, User Requirements, and User Abilities

To get the best transformation results for a transcoding process, a transcoder needs sufficient information about the user as well as the used hardware.

In the wireless world there are numerous different devices available which all come with different properties. Unfortunately, there are neither standards in pixel sizes and shapes nor in display sizes, display resolution, characters per line etc. Considering all the currently available variations in mobile hardware, we can roughly distinguish them into four different display categories: small phones, smart phones, PDAs, and Webpads where, e.g., a PDA is considered to have a higher resolution than a smart phone and smart phones have higher resolutions than small phones etc. (cf. Table 1). It has to be noted here, that there are many cases where a device cannot be uniquely assigned to such a category. Considering user preferences, users often give very inprecise specifications of their requirements. For example, a user might object to frequently use horizontal scrollbars. Then, when a page is too large for a display, a transcoding algorithm has to split a page into several subpages. Questions to be solved here are: what does *frequent* mean in this context and what is the size of a piece when splitting the content? Table 2 determines a first solution and tries to define when to split a document page with respect to different device categories.

If we additionally take accessibility aspects into account, we have, for example, to deal with various visual abilities of users. So, if a user has visual impairments, the font has to be enlarged to keep text readable. That, of course, affects the available space on the display. So, an PDA display might have only the comparable capacity of the display of

contentSplitting	smallPhone	smartPhone	PDA	Webpad
paging	always	always	always	always
someScrolling	often	often	sometimes	sometimes
moreScrolling	sometimes	sometimes	onDemand	onDemand
noPaging	never	never	never	never

Table 2. Content Splitting for Different Devices

a small phone when displaying larger characters. We thus denote this as the size of the virtual display since enlarging the font size virtually has the same effect as decreasing the display size. If a person has significant visual impairments, we sometimes cannot use traditional output devices. In that case a transcoder also has to produce output for a voice enabled interface or VoiceML instead. Table 3 gives the relationship between different visual abilities and display sizes of mobile devices where, e.g., vPDA stands for virtual PDA.

4.2. Fuzzy-RDL/TT

For the definition of the transcoding process by fuzzy sets and rules we have defined Fuzzy-RDL/TT (Rule Description Language for Tree Transformation). Fuzzy-RDL/TT is a fuzzy set-based language to describe rules over fuzzy sets triggering tree transformations. We apply tree transformation since each XML-based document can be represented by a tree structure, the DOM (Document Object Model) representation [4]. This representation enables to easily locate and manipulate elements and other objects. Fuzzy concepts within Fuzzy-RDL/TT are used for the selection of tree transformation function which implement the transcoding.
A Fuzzy-RDL/TT description is composed of three parts

- fuzzy set definitions
- rule definitions
- transcoding definitions

The first two parts are for the definition of fuzzy sets and rules over fuzzy sets. The third part gives a definition of transcoding functions separated into categories which are given by previously defined fuzzy values. Details of the three parts are outlined in the remainder of this section.

4.2.1. Fuzzy Set Definitions

The definition of the fuzzy sets are given by associating an identifier with values of a membership function for fuzzification. For this we define each function with four coordinates $(x_1, 0), (x_2, 1), (x_3, 1), (x_4, 0)$. For matter of abbreviation we omit the y-coordinates in the specification and only enumerate the four x-values. Table 2 depicts the function graph for the fuzzy set defined by $50, 70, 100, 120$ where the y-axis denotes the membership to the fuzzy set by a value between 0.0 and 1.0 which is denoted as Degree of Match (DoM).

Given the different mobile device categories in Table 1, the corresponding fuzzy set specifications for *display* can be defined as follows.

```
SET display:
    smallPhone = 50,70,100,120;
    smartPhone = 100,120,160,180;
    PDA        = 140,160,640,660;
    WebPAD     = 600,640,800,1200;
```

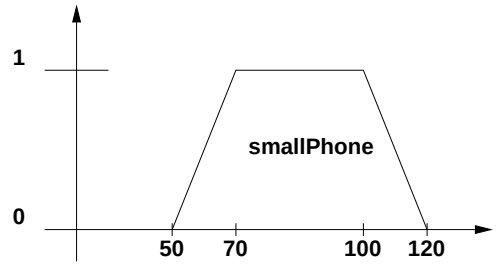

Figure 2. Fuzzy Set for Small Phone

Note here, that for our purpose we only consider the display height for computing the value. In our investigations it has been shown that this is a good approximation for the display size since display's width is typically linearly dependent on its height. With crisp input the defined functions are matched and the value for the degree of Match is returned. That means, for instance, when given the function in Figure 2 and a display height of 60 pixel the DoM is approximately 0.54 for category *smallPhone*.

Additionally, we have to declare fuzzy sets for non crisp values, e.g., for user preferences. In contrast to hardware profiles, user preferences are directly selected by a user. In Fuzzy-RDL/TT, those sets are enumerated after the set identifier for the purpose of declaring them for their usage in the later rules definitions as given by the following example.

```
SET userPrefs: paging;someScrolling;
               moreScrolling;noPaging;
```

visualAbilities	SmallPhone	SmartPhone	PDA	Webpad
blind	audio	audio	audio	audio
very weak	audio	audio	vSmallPhone	vSmartPhone
weak	audio	vSmallPhone	vSmartPhone	vPDA
good	vSmallPhone	vSmartPhone	vPDA	vWebpad
eagle	vSmartPhone	vPDA	vWebpad	vWebpad

Table 3. Virtual Display Sizes for Devices and Abilities

For those sets we do not define a specific membership function but implicitly assign a constant function of 1.0 to them.

4.2.2. Fuzzy Rule Definitions

Rule definitions directly correspond to decision tables introduced in Subsection 4.1. Rules are combined to sets preceded by the keyword *RULE* and an identifier. Each individual rule follows the pattern

```
IF <conditions> THEN <value>
```

where *conditions* combine variables from fuzzy sets and *value* is the return value when matching the rule. The following example gives the rule specification for the first column of Table 2.

```
RULE contentSplitting:
   IF display == smallPhone &
                 userPrefs == paging
      THEN always;
   IF display == smallPhone &
                 userPrefs == someScrolling
      THEN often;
   IF display == smallPhone &
                 userPrefs == moreScrolling
      THEN sometimes;
   IF display == smallPhone &
                 userPrefs == noPaging
      THEN never;
```

Executing the rules, the return values of all matched rules are combined. In details, rules are computed as follows. Firstly, for all conditions the Degree of Match is computed. Recall that non crisp values like user preferences are considered with a DoM of 1.0. For each rule, all conditions are combined by computing the mean average. Thereafter, the rule with the highest mean average is selected and the corresponding value returned. For each value there exists a set of transcoding rules which are being executed in the final step. The definitions of the transcoding rules and functions are outlined in the next paragraphs.

4.2.3. Transcoding Definitions

A transcoding definition is composed of the global settings and the definition of the transcoding rules.

```
<global settings>
<transcoding rules>
```

The global settings are for the retrieval of values from given hardware, software, and user profiles. Retrieved values can be assigned to internal variables, which can be applied as parameters to transcoding functions. Additional variables can be declared for internal applications like counters etc. Variables are typed with respect to the expression on the right hand side: integer, float, string etc.

To outline the basic concepts, we we give a simple example which first retrieves values from the display profile (witdth, number of characters in a row) and the user profile, and assigns their values to local variables. Additional integer variables are defined, which can be used as row and column counters in the later transcoding rules.

```
resX =
  getProperty("display.Width");
noOfChar =
  getProperty("display.TotalChars");
userScroll =
  getProperty("user.Scrolling");
row = 0;
column = 0;
```

After the global settings global and value specific transcoding rules are defined.[1] Global transcoding rules are always executed before the execution of the specific rules. In the context of a HTML to C-HTML transcoding, this is useful for the insertion of the HTML-tag attribute "version=C-HTML 1.0", for instance. Value specific transcoding rules are separated by a label which has to correspond to a rule identifier and a return value. The following definition gives the structure of the rule definitions with labels from the previously introduced example.

[1] RDL/TT as it was introduced in [9] only covered global settings and global transcoding rules.

```
<global transcoding rules>

contentSplitting.always:
     {<transcoding rules>}
contentSplitting.often:
     {<transcoding rules>}
contentSplitting.sometimes:
     {<transcoding rules>}
contentSplitting.onDemand:
     {<transcoding rules>}
contentSplitting.never:
     {<transcoding rules>}
```

A transcoding rule definition consists of a collection of search patterns, each followed by a rule description. A search pattern is an XML (or HTML) tag such as `<body>` which uniquely denotes an XML element. Alternatively, a list of tags can be given as a shortcut definition, e.g., `<h1>, <h2>, <h3>, <h4>, <h5>`. The transcoding definitions given after the list of tags are then applied to each element of that list.

The functions specified after the tag list are executed on the DOM (Document Object Model) tree representation of the underlying XML document. More precisely, the given functions manipulate the subtrees of all elements that match the tag. Since not only tags are represented as nodes in the DOM tree almost all of the transcoding functions can be applied to attributes and text as well. For manipulation, RDL/TT supports the following operations:

- deletion:
  An HTML element is deleted, i.e., it is removed with all sub-elements. In the tree representation this relates to the deletion of a node with all its children. RDL/TT uses the function *remove(pathname)* for this operation.

- insertion:
  With insertion of a new element, we distinguish between inserting or appending with respect to the position of the current node. This is supported by the functions *insert(name)* and *append(name)*. *insertText(text)* and *appendText(text)* inserts/appends textnodes.

- replacement:
  The replacement function moves a node including its subtree from its original position given by the first parameter to the position given by the second pathname. This function can also be used to remove a pair of tags instead of the whole element as the element will be replaced by its content: (*replace(this, this.content)*).

- renaming:
  Assigns a new name to the matched element.

- replacement with links:
  An element can be replaced by a combination of the previous deletion, insertion, and replacement functions. RDL/TT additionally supports replacement of an element including its subtree by the function *replaceWithLink*. The replacement function generates a link (for example, using the <a> tag). Parameters are the name of the link, an URL to the replacing file, and the original element name.

- outsourcing:
  A very important concept is the feature to take elements including its subtrees, create a new document, put elements in that document, and replace them with a hyperlink in the original document. This is supported by the RDL/TT function *replaceWithDocument*. For application consider a transcoding rule defining that tables are to be displayed on a separate page. In HTML, this is only possible if a table is outsourced to a separate file.

Syntactically, RDL/TT rule definitions are based on Java also inheriting concepts from XSL and XPath. An RDL/TT statement can be either an assignment to a variable or a function call. RDL/TT provides a built-in data type for pathnames denoting object locations in the given tree. Paths are specified by separating their components by the dot operator. Components are HTML elements or attribute identifiers or one of the following components:

- "this" denotes the current node.

- "parent" points to the parent node.

- "$*$" means all neighbouring nodes.

- "attribute" denotes an attribute node. The preceding path component has to be an element and the following component has to be the attribute name.

- "content" points to the content of the element specified by the preceding components.

Since a path component can refer to several nodes when the parent node has several children of same type, we introduce aggregations on those components to support a clear selection of subelements. For example, "node[0]" refers to the first node, "node[0..2]" refers the first three nodes, and "node[$*$]" stands for all nodes. Without an aggregation specification, the first node is taken by default. For example, "this.p.content" points to the content of the first paragraph child of the current node.

RDL/TT additionally provides two control structures: "if/else"-branches and loops, where loops iterate over path patterns. The latter concept is inherited from UNIX C-Shell programming constructs. For example, "foreach var in this.p" iterates over every paragraph which is a child of

the current node and variable "var" takes each paragraph as value.

The following example sketches the previously introduced concepts of RDL/TT where a more complex example is given in the next section. We take the previously introduced defined fuzzy rules returning `always`, `often` etc. as an example and consider a transcoding from HTML to C-HTML.

```
<html>
   insertAttribute("version","C-HTML 1.0");
<head>
   remove(this.isindex);
<applet>,<param>,<basefont>,
<script>,<style>,<link>
   remove(this);

contentSplitting.always:
  <.>
      split(noTotalChars);

contentSplitting.often:
  <p>
      replaceWithDocument("Page" +
              stringValue(pageCounter),
              this.content, "Page" +
              stringValue(pageCounter));
```

The global transcoding rules first add an extra attribute to the <html>-tag. The attribute *isindex* is being deleted from tags <head>, and tags <applet>, <param> etc. are being removed including all its enclosed elements, i.e., subtrees. When evaluating to *always* and *often* additional transcoding operations are executed. In the case of *always*, a line break is inserted by the function which applies to all text nodes. The function is called with the maximum character width `noTotalChars` to which the corresponding value from the hardware profile was previously assigned to. In the case of *often*, each paragraph is replaced by a hyperlink which points to a file with paragraph content.

5. Examples

We now elaborate on a more complex example. Figure 3 shows a Web page which is composed of a table to set up a frame-like screen structure. Note, that in this case the table is purely used for layout purpose. The thick lines in the screenshot are virtual lines which indicate the positions of rows and columns of that table.

In contrast to the small example in the last section which dealt with general page breaking, we show here how this page is transcoded to different virtual devices according to user abilities and real device properties.

In this part we focus on the actual transcoding rules and assume that we already have computed the virtual display

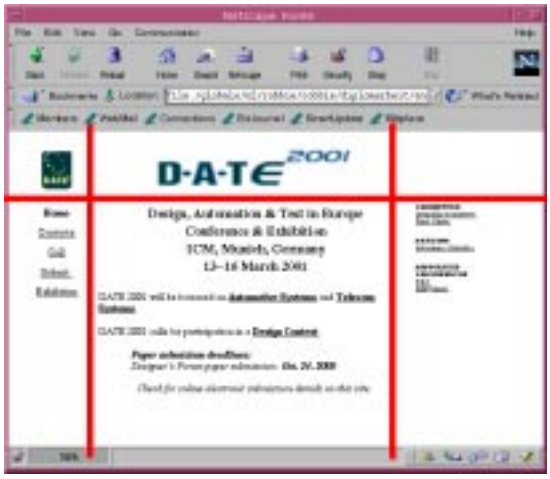

Figure 3. Example Web page

size applying the table in Subsection 4.1. An excerpt of the used ruleset is shown hereafter:

```
virtualDisplay.vWebpad:;
// no modification

virtualDisplay.vPDA:
// keep all images and sequentialize tables
  rowcounter = 0;
  headcounter = 0;
  <img>;
  <table> remove(this);
  // remove the tag
  <th>
  // store table headers for later use in th
     th[headcounter] = this.content;
     headcounter = headcounter + 1;
  <td>
  // break after each table data cell
     appendNode("br");
  <tr>
  // each Row will be marked
  // either with a heading
  // or with a counter
     if (th[rowcounter] != "")
       insertText(th[rowcounter] + ":");
     else
       insertText("Row" + rowcounter + ":");

virtualDisplay.vSmallPhone:
// remove images and write table content to
// extra file
  tablecounter = 0;
  <img>
  // create a abbreviation for the image name
     src = getSecondLastToken(
           valueOf(this.attribute.src));
  // use long description as name for hyperlink,
  // if name exists
     if (exists(this.attribute.longdesc))
       if (exists(this.attribute.alt))
         replaceWithLink("[" +
               valueOf(this.attribute.alt) + "]",
```

```
            valueOf(this.attribute.longdesc),
            "image");
      else
        replaceWithLink("[IMG:" + src + "]",
            valueOf(this.attribute.longdesc),
            "image");
    else {
      insertText("[IMG:" + src + "]");
      remove(this);}

<table>
// write table content to extra file
// use for the hyperlink name either
// the  title or the table counter
  if (exists(this.attribute.title))
    replaceWithDocument("TBL:" +
          valueOf(this.attribute.title),
          this.content, doctitle + ">TBL:" +
          valueOf(this.attribute.title));
  else
    replaceWithDocument("TBL:" +
          stringValue(tablecounter),
          this.content, doctitle + ">TBL:" +
          stringValue(tablecounter));
  tablecounter = tablecounter + 1;
...
```

Let us assume that a PDA is used by a user with visual impairments. So instead of selecting a PDA as a virtualDisplay, the system selects a small phone as virtual display and applies the corresponding rules. The small phone is being selected since the font size has to be increased, which means that the PDA can only display the information amount of a small phone.

The transcoding rules for such a vSmallPhone define that images are replaced by hyperlinks to textual descriptions, if possible, and tables are outsourced to separate files. For the table-elements (<th>, <tr>, <td>) the same operations as in the PDA part are applied (not explicitly shown in the above listing), or in other words, the table is sequentialized.

The result of the previously defined transcoding is shown in Figure 4. Here, part of the transcoded lower right table cell is rendered in a browser which emulates the "Philips Nino 500" device with a resolution of 320x240 pixel.

Figure 4. Transcoded Example

6. Implementation

The transcoding system has been implemented as outlined in Figure 5 in Java 1.3. The fuzzification and inference component has been implemented based on the RDL/TT component. In order to normalize the HTML input file we use the Tidy preprocessor from W3C and "Xerces" from Apache for DOM. HTML-Tidy provides a filter for HTML files which removes errors and generates XHTML. The XHTML file is translated into a DOM tree representation by Xerces. Our RDL/TT interpreter finally computes the tree transformation based on that DOM tree with fuzzy sets, rules, and user and hardware profiles as input.

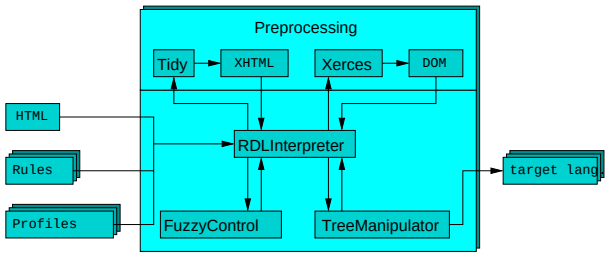

Figure 5. Transcoding System Architecture

7. Conclusion and Future Work

In this article, we have introduced Fuzzy-RDL/TT for the transcoding of XML-based documents for mobile devices. We have demonstrated our approach by the example of an HTML to C-HTML transcoding. However, since our approach is based on DOM tree representation and manipulation it also applies to general transcoding for any XML-based document that can be represented as a DOM tree. Fuzzy-RDL/TT provides means to capture user and hardware properties as fuzzy sets and to evaluate them in order to select the adequate set of transcoding functions. Thus, the fuzzy evaluation component in this process is actually a preprocessor for selecting those functions.

We presently have implemented a complete version of the RDL/TT interpreter and a first version of the fuzzy rule evaluation. This gives us a stable prototype through which we are able to evaluate parameters of this highly configurable system. First evaluations demonstrate promising results. However, the quality of the result depends on the quality and quantity of the transcoding rules. At the moment we have transcoding definition for complete HTML to C-HTML and to WML for PDA-oriented devices which already gives us a considerably complex transcoding system. We currently have defined only a few rules for small phones and smart phones. Nevertheless, at the moment

we are clearly at an experimentation phase mainly targeting towards a complete system for transcoding HTML to C-HTML. In order to support small phones and/or visually disabled or elderly people, we are currently also investigating a combined transcoding to C-HTML and VoiceML with speech synthesis.

Acknowledgements

The work described herein is funded by the German Ministry for Education and Research (BMBF) under the ITEA VHE Middleware project.

References

[1] S. Adler et al. *Extensible Stylesheet Language (XSL) Version 1.0, W3C Working Draft*. World Wide Web Consortium, October 2000. URL: www.w3.org/TR/xsl.

[2] J. Clark. *XSL Transformations (XSLT) Version 1.0, W3C Recommendation*. World Wide Web Consortium, November 1999. URL: www.w3.org/TR/xslt.

[3] J. Clark and S. de Rose. *XML Path Language (XPath) Version 1.0, W3C Recommendation*. World Wide Web Consortium, November 1999. URL: www.w3.org/TR/xpath.

[4] M. Davies et al. *Document Object Model (DOM) Level 2 Core Specification Version 1.0 W3C Proposed Recommendation*. World Wide Web Consortium, September 2000. URL: www.w3.org/TR/DOM-Level-2-Core.

[5] M. Hori et al. *Annotation of Web Content for Transcoding, W3C Note*. World Wide Web Consortium, Juli 1999. URL: www.w3.org/TR/annot.

[6] J. Plomp, R. Schaefer, W. Mueller, H. Yli-Nikkola. *Comparing Transcoding Tools for Use with a generic User Interface Format*. Oulu/Finland, Paderborn/Germany, October 2001. (submitted for publication)

[7] R. Daniel Jr. et al. *XML Pointer Language (XPointer) Version 1.0, W3C Recommendation*. World Wide Web Consortium, Juni 2000. URL: www.w3.org/TR/xptr.

[8] R. Swick and O. Lassila. *Resource Description Framework, (RDF) Model and Syntax Specification, W3C Recommendation*. World Wide Web Consortium, Februar 1999. URL: www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[9] R. Schaefer, A. Dangberg, W. Mueller. *A Generic Language for the Transcoding of HTML Pages*. C-LAB Report 31/2001, Paderborn, Germany, February 2001. (in German)

[10] F. Reynolds et al. *Composite Capability/Preference Profiles (CC/PP): Structure, W3C Working Draft*. World Wide Web Consortium, Juli 2000. URL: www.w3.org/TR/2000/WD-CCPP-struct-20000721.