Internationalization of a Distance Exam Web Environment

Radouane Mrabet
Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes (Rabat, Morocco). Tele-Teaching Laboratory.
Email: mrabet@ensias.ma
Jamal Bentahar
Laval University. Computer Science Department (Québec, Canada). Cognitive Informatics Laboratory.
Email: jamal.bentahar@ift.ulaval.ca

Abstract

This paper describes an architecture to provide multilingual support for an exam Web environment with an emphasis on Arabic localization (or Arabization). Developing software products for populations with different cultures is a two-step process: internationalization followed by localization. Proper Arabization is particularly complex with many interesting challenges.

In this context, we developed an architecture to provide multilingual support for EDILE (Exam DIistance Learning Environment). EDILE covers all steps of the evaluation process, from the exams edition to the grades publication. The architecture is based on Java resource bundles and separates a program's code from the resources it uses. Of necessity, this architecture also includes a component for proper language and cultural presentation. EDILE uses the same source code for different language versions. The architecture uses parameters to indicate, for example, the language and the font. Hence, the accessed resources are specified only at run time. This configuration makes EDILE a dynamic environment.

For Arabic localization, we use the Unicode bi-directional algorithm. We solved the problems of the contextual analysis and ligatures by using finite states automata. Our solution also makes use of the Unicode character-glyph model. We also use a virtual keyboard implementation based on the Unicode table to manage the Arabic keyboard.

In the paper, we first briefly review relevant aspects of internationalization and Arabization. Then we show the EDILE environment and its internationalized architecture. We conclude by describing details on the Arabic localization of EDILE.

1. Introduction

Currently, having software account for culture has become a necessity. The best solution to develop such products is based on internationalization and localization processes.

Internationalization is the process of designing software to support different languages and cultures without having to change the program. On the other hand, localization is the process of modifying an internationalized program so that it behaves correctly in a given language and culture. In this context, Arabic localization or Arabization (A9n) requires complex processing to solve contextual analysis and bi-directional layout issues.

For web applications, internationalization is a significant element. Moreover, for applications such as Web-based learning and distance exam, multilingual aspects become measure of the quality of the implementation.

In this paper, we describe an architecture to provide multilingual support for a distance evaluation Web environment called EDILE (Exam Distance Learning Environment). EDILE is a client/server application which covers all steps of an exam process. The architecture is based on Java resource bundles and allows us to use the same source code for various linguistic versions. Parameters specify, for example, the language and the font used and a component for proper presentation of language and culture. We also present the Arabic localization and our solution of contextual analysis and ligatures. The solution is presented in the form of a state machine. EDILE uses the Unicode bi-directional algorithm [Unicode 00] and an Arabic virtual keyboard is implemented by using the Unicode table.

We start by reviewing relevant aspects of internationalization and Arabization. Then we show the EDILE environment and its internationalized architecture. We conclude by describing details on the Arabic localization of EDILE.

2. Internationalization and Arabization

2.1. Internationalization

Internationalization of software at the time of its design facilitates the management of these linguistic versions. It minimizes the cost of its maintenance and makes possible a simple and fast localization. Internationalization is related not only to applications but also to the World Wide Web. Although the Internet removed the distance barrier, language barriers still remain.

Internationalization typically requires a modification of source code. If internationalization is not integrated into the software development process, adding it later can be much more expensive. Internationalization requirements must be studied at the requirements analysis and definition phase, or at the latest at the system and software design phase.

Unicode character-glyph model

In the internationalization context, Unicode describes abstract character codes but does not offer methods for rendering text. Indeed, within the framework of automatic text processing, the distinction between a character's meaning (the character) and its shape (the glyph) is significant [Hart, TR 15285]. Character and glyph are two closely related but different concepts. Rendering text is nothing more than the transposition of a set of characters into a set of glyphs. In general, the relationship between coded characters and glyph identifiers is an M-to-N mapping,

where $M > 0$ and $N \geq 0$. Note that where this relationship is one-to-one ($M = N = 1$), the glyph selection process is unnecessary. On the other hand, when the relationship is M-to-N where $M \neq N$ (like for the Arabic script, for example), a sophisticated process of glyph selection is required.

2.2. Arabization

The problem of localizing software for Arabic can be treated in the multilingual context. Two approaches are proposed. The first one consists in radically changing the system. The second one aims at extending the monolingual system to make them multilingual plate-forms. Hence, specific Arabization support must be added either to the application itself, or to the operating system interface [Henda 97]. The main Arabic language support problems are the following: character codeset and standard encoding, character shaping and text direction algorithms, character fonts and dual keyboard management [Amara 96].

Arabic encoding

Before the advent of Unicode, Arabic automatic processing faced coding obstacles. Indeed, in 1976, a unified code, similar to ISO/IEC 646, created the first Arabic code (CODAR1) [Romerio 76]. In 1982, ALESCO (Arabic counterpart of UNISCO) and ASMO (Arabic Standards and Measurements organization) defined the Unified Arabic Code / Final Form (ASMO 449). Since then, other Arabic code standards were defined and implemented, for example, ISO 9036: 1987 and ISO/IEC 8859-6: 1999.

Arabic script and character-glyph model

The Arabic script presents major and specific processing problems that are not encountered in Latin-based languages. Three complex phenomena exist for rendering text in the Arabic script: right to left direction, contextual forms, and ligatures. The order of the presentation of glyphs may not follow the same logical order of characters in storage. In addition, the Arabic script uses cursive (or joined) forms of the letters. Hence, each character may have several presentation forms. Some characters can be linked to another character on either side, and some characters can be linked only on their right side. In the first case, each character has four possible glyphs (Figure 1-a). In the second one, each character (called an exceptional character) has only two possible glyphs (Figure 1-b). Finally, Arabic topography makes extensive use of ligatures. These ligatures associate one specific glyph to the joining of two or more Arabic characters. Ligatures are necessary for well-rendered Arabic text. Like the glyphs for Arabic letters, ligatures can have glyph forms that depend on the position of the letters in a word (Figure 2). Because of ligatures and the cursive nature of the Arabic script, the character-to-glyph relationship is not one-to-one.

Arabic script and font model

A font for Arabic characters has particular requirements. Because Arabic is written in cursive form, the font needs to be adapted to this style. The glyph selection process must be able to select glyphs to join the Arabic characters into words to avoid blank columns between characters [Amara 96].

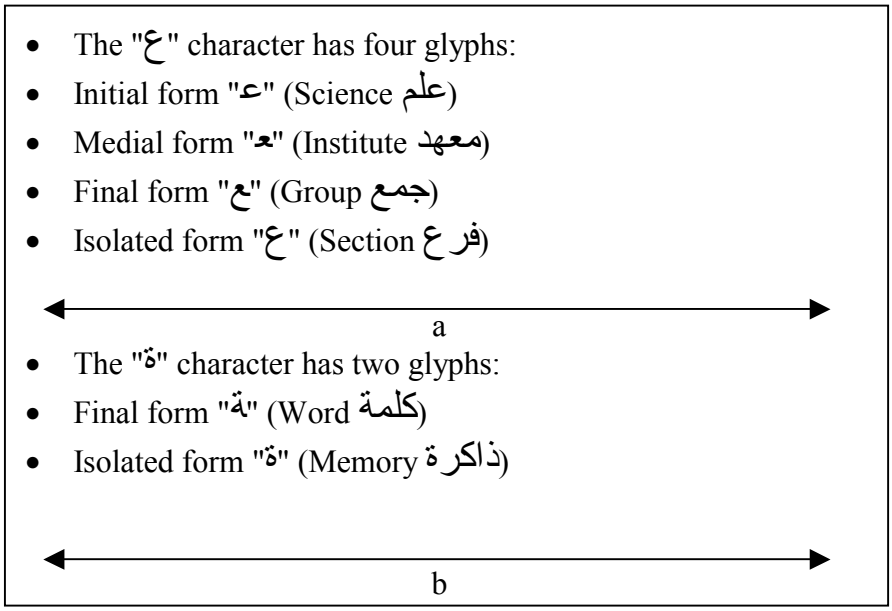


Figure 1: The relationship between characters and glyphs in Arabic script.
a : 1 to 4 mapping. b : 1 to 2 mapping.

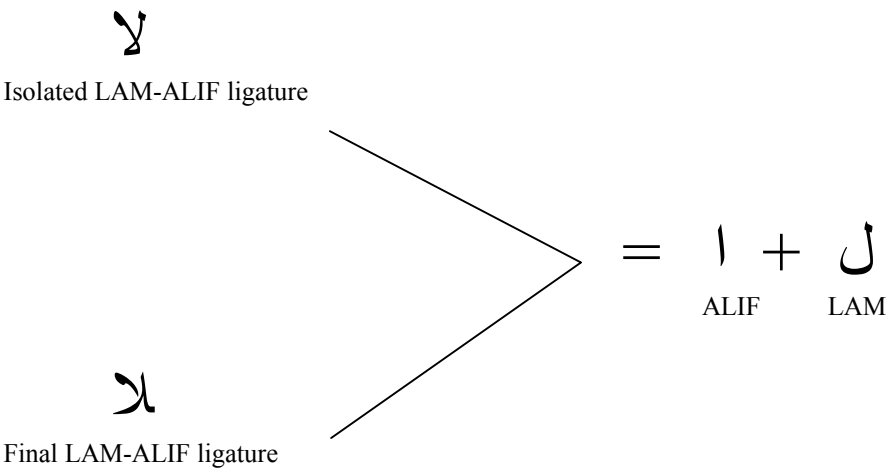


Figure 2: LAM-ALIF ligature.

The Unicode character-glyph model [Hart 99, TR 15285] describes three models for rendering characters into glyphs: the coded font model, the font resource model, and the intelligent font model.

In coded font model, the general layout process uses the character code to locate a corresponding glyph. This model requires that the relationship between characters and glyphs is one-to-one. Consequently, to support the more complex glyph mapping in the Arabic script, the

coded font model requires additional processing. Thus, this model is not appropriate for the Arabic script.

On the other hand, the font resource model and the intelligent font model can support it easily. The reason is that these two models can deal correctly with the M-to-N mapping required by the Arabic script. Because the intelligent font model includes layout transformation information to process bi-directional text, it should be more appropriate than the font resource model.

In addition to the character rendering issues, which we discussed earlier, an Arabic system must manage the Arabic keyboard. Moreover, in the Latin systems (under Windows for example) when the user presses a key, the keyboard input method of the operating system receives the corresponding keyboard scan code, uses the keyboard state and scan code to convert it to a character code, and sends the character code to the display process for glyph selection and rasterizing the glyph for presentation to the user[Microsoft 97]. In an Arabic system, keyboard processing is more elaborate. An Arabic input method may need to map sequences of typed keyboard scan codes to sequences of character codes.

In the following parts, we show the EDILE system and its internationalized architecture. Finally, we present the Arabization process in EDILE and our contextual analysis algorithm.

3. EDILE: a Web environment of distance evaluation

Traditional teaching systems lack flexibility for people who are not local full-time students who are able to participate in classes. Limits imposed by time, place, and rhythm make classical teaching extremely limited. Tele-teaching offers more interesting alternatives and challenges. For example, tele-teaching applications must support methods of distance knowledge assessment.

EDILE (Exam Distance Learning Environment) [Mrabet 98] is an environment that supports a distributed evaluation system. It was defined in the TELESUN (TELE-teaching System for UNiversities) project that aims at implementing a worldwide multimedia tele-teaching system using pedagogical process [Nasri 97].

EDILE topology consists of several exam centers connected by the Internet. Each center has a fast Ethernet, 100 Mb/s local area network, which connects local client stations to a Windows NT server station.

The exam process requires several people. In EDILE these are: students, professors, proctors and administrators. EDILE covers all steps of the evaluation process from creating the exams to publishing the grades. These steps are:

1. Exam preparation.
2. Exam registration.
3. Exam distribution.
4. Exam release.
5. Exam copies retrieval.
6. Grade publication.

EDILE is a client-server application implemented in Java / HTML [Tazi 99]. Three applets are used in the client computers: the first one for editing the exam, the second one for delivering the exam to the students, and the third one for retrieving the answers to the question on the exams. Communication between the client applets and the server application uses a TCP (Transmission Control Protocol) connection.

4. Internationalization of EDILE

Like any Web application, the objective of EDILE internationalization is to develop a world wide multilingual environment. In addition, in a significant field such as Web-based learning, internationalization is a prerequisite to building a complete education system.

EDILE multilingual version is implemented in Java. This choice is justified by: 1) the power of this language in the Web development field and its inclusion of functions to perform internationalization, and 2) its native support of the Unicode character set.

In accordance with the definition and with the roles of internationalization [Gillam 99], EDILE uses the same source program for the various language versions it supports. All user interface elements (labels, messages, buttons, etc.) are separated from applet codes. So depending on the language and the culture, these elements, also called resources, may be easily configured.

4.1. Architecture

The internationalized architecture of EDILE includes a component for proper language and cultural presentation. Particularly, the component deals with Arabic language aspects such as contextual analysis and ligature selection. The component functions are specified outside the applet programs; this makes EDILE more flexible and easy to maintain.

Illustrated by Figure 3, this architecture is designed to support multilingual aspects. For this reason, EDILE applets use parameters to indicate, for example, the language and the font. In this context, these applets are parameterized and the accessed resources are specified dynamically at run time.

4.2. Localization process

Since EDILE is an interactive environment, we had to carefully design the user interface to ensure successful internationalization. First, the user interface arranges its various elements to respect the language writing direction. In addition, we paid particular attention to the text of the labels and message to make them clear and avoid abbreviations.

The architecture that we defined for EDILE enormously facilitates its localization process. For each language, it is only necessary to define the resources that it uses. Indeed, EDILE is based on Java Resource bundles to take advantage of inheritance and to avoid repetition [Gillam 99]. EDILE currently supports three languages: Arabic, English and French. These languages correspond respectively to the classes Resources_ar, Resource_en, and Resource_fr. The

linguistic versions are intended for three countries: Morocco, the United States and France. EDILE is an extensible environment and other languages and countries can be easily added in the future versions. This operation does not require recompilation of applet source codes, but only the addition of a new resource file.

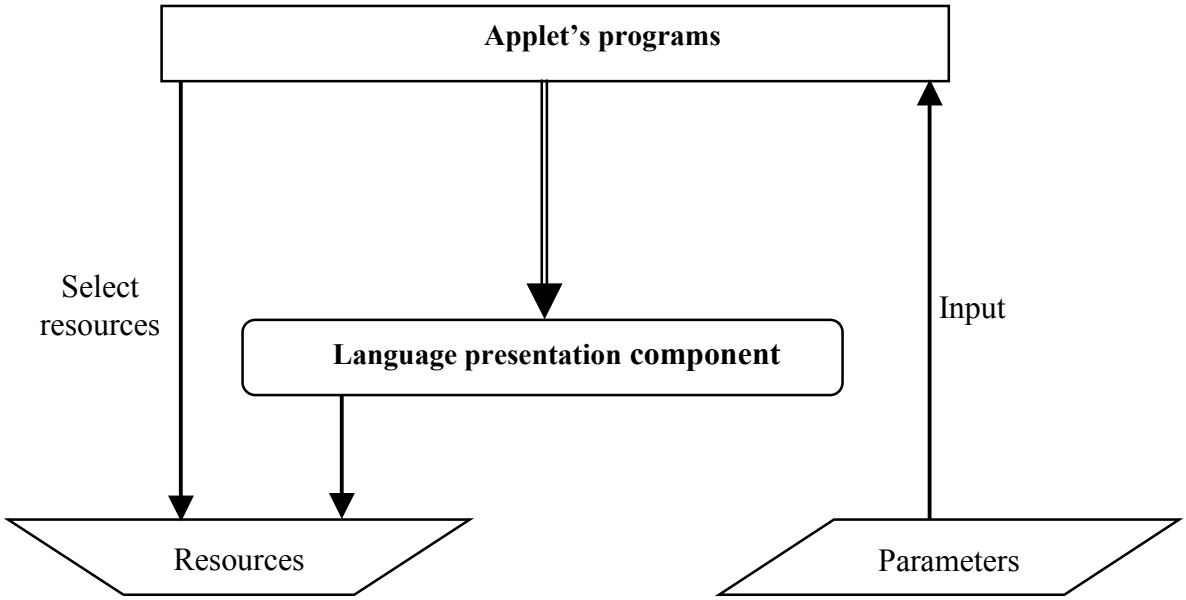


Figure 3: Multilingual EDILE architecture.

Users start at the EDILE home page to choose the language. This is specified only at run time. The Applet's HTML files make use of three localization parameters: the language, the country, and the font to select the information and present it to the user. Figure 4 shows the HTML file associated with the edition applet for Arabic localization.

```
<html>
<head>
<title>EDILE Environment</title>
</head>
<body LINK="#0000FF" VLINK="#800080" background="bleu.jpg">
<p>&amp;nbsp;&amp;nbsp; &amp;nbsp;
<applet CODE="Edition.class" WIDTH="540" HEIGHT="392" align="texttop" VIEWASTEXT>
<PARAM NAME = "Language" VALUE = "ar">
<PARAM NAME = "Country" VALUE = "MA">
<PARAM NAME = "Font" VALUE = "Arabic Transparent">
<PARAM NAME = "Direction" VALUE = "R_to_L">
loading images</applet>
</p>
</body>
</html>
```

Figure 4 : The HTML file of the edition applet (Arabic version)

4.3. Arabic localization

To handle multilingual input from the same keyboard, EDILE implements a virtual keyboard. Client computers are configured to use the default (US English) keyboard driver. This physical keyboard outputs Unicode values of the typed English (Latin) characters. The EDILE virtual keyboard uses keyboard mapping tables to support multilingual input. Figure 5 shows the EDILE keyboard management. EDILE defines as many keyboard mapping tables as supported languages (and keyboards). For example with an Arabic user, when the "a" key is pressed, the code generated should be an "ض" (U+0636, for the ARABIC LETTER DAD) rather than an "a" (U+0061 for the LATIN SMALL LETTER A). When a key is pressed, the keyboard driver of the operating system provides the Unicode character code for an English keyboard. EDILE selects the correct character code for the language by looking in the active language table. This solution enables us to manage not only the Arabic keyboard but also keyboards of all supported languages that use keyboards compliant with ISO/IEC 9995 [ISO 9995].

While Arabic text is written from right to left, text containing both Arabic and Latin strings must be presented on the same line so that the Latin strings are displayed from left to right and the Arabic strings are displayed from right to left. Text is stored in sequential order (the order in which the text is typed) in the backing store. EDILE uses the Unicode bi-directional algorithm to correctly display text in the backing store.

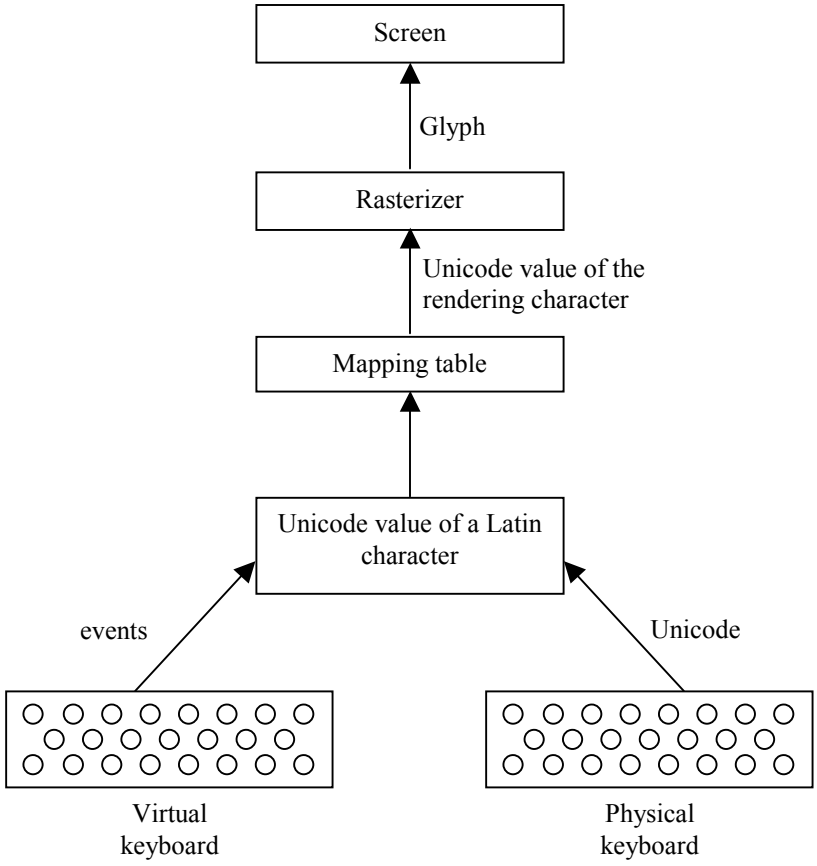


Figure 5: Keyboard management in EDILE.

Arabic version

Like the English and French versions, the EDILE Arabic version consists of the same three applets. To publish an exam, a professor must initially fill in a form (Figure 6) that includes general information such as exam identifier, location, subject, etc. Then, the professor uses an editor to build the exam questions. The professor then makes the exam available to the students. The student selects the exam, and EDILE displays questions in the proper language. The student then answers the questions. Finally, the professor receives the results from each student and grades the exam. EDILE permits the professor to make notes on the answers while grading an exam.

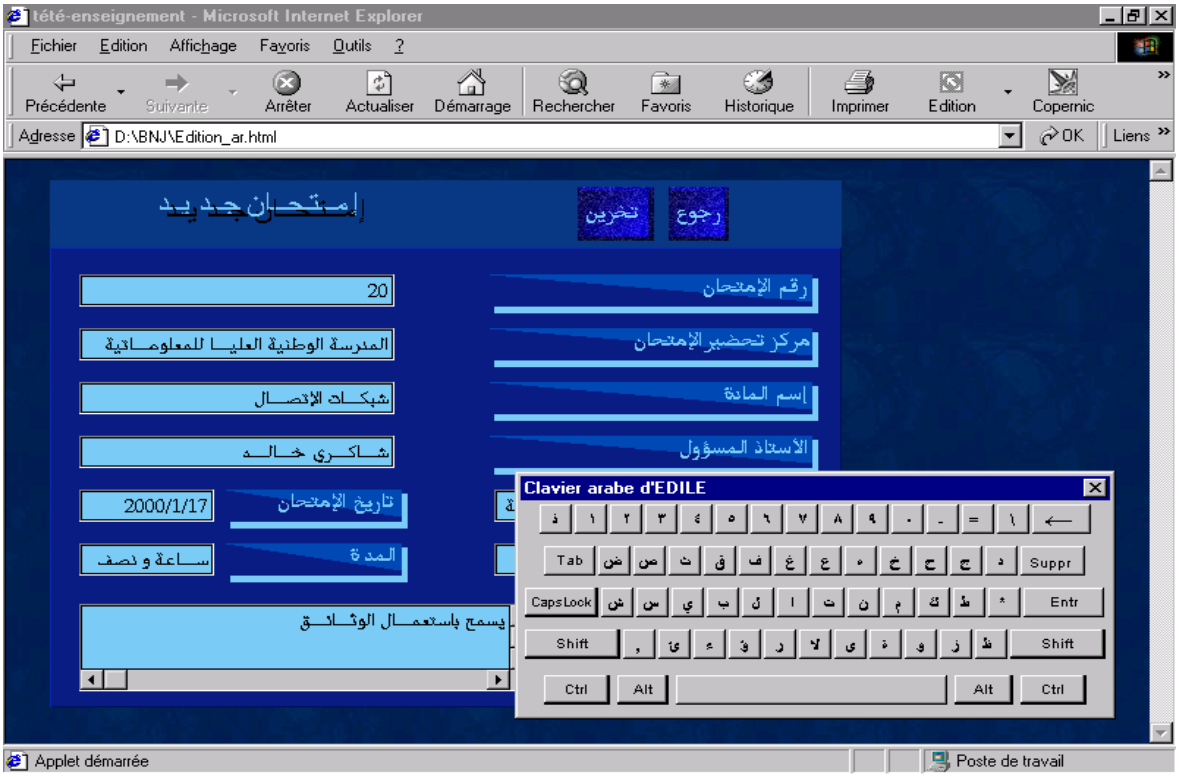


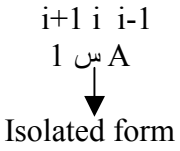
Figure 6: An exam's general information.

5. Contextual analysis

Contextual analysis is necessary for many scripts to present the correctly shaped and combined glyphs. For Arabic script, to select a character's proper presentation form, it is necessary to take into consideration the previous and the next character if they exist. Our contextual analysis algorithm implements the following rules:

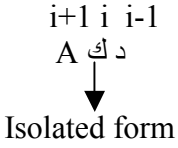
- If the character (i-1) (i being the index of the current character) is not an Arabic character (or it does not exist) and the character (i+1) is not an Arabic character (or it does not exist), then the character (i) takes the isolated form.

  Example:



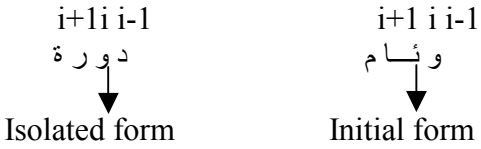
- If the character (i-1) is an exceptional Arabic character and the character (i+1) is not an Arabic character (or it does not exist), then the character (i) takes the isolated form.

  Example:



- If the character (i-1) is an exceptional Arabic character, and the character (i+1) is an Arabic character, then the character (i) takes the initial form if it is not exceptional and the isolated form if it is exceptional.

  Example:



- If the character (i-1) is an unexceptional Arabic character and the character (i+1) is an Arabic character, then the character (i) takes the medial form if it is unexceptional and the final form if it is exceptional.

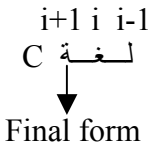
  Example:

- If the character (i-1) is not an Arabic character (or it does not exist) and the character (i+1) is an Arabic character, then the character (i) takes the initial form if it is unexceptional and the isolated form if it is exceptional.

  Example:

- If the character (i-1) is an unexceptional Arabic character and the character (i+1) is not an Arabic character (or it does not exist), then the character (i) takes the final form.

  Example:



To check whether a character is an Arabic character we use the range 06 of Unicode which describes the basic Arabic characters and the fourth part of the range FE which describes the presentation forms of basic Arabic characters. We also use the range FE to render Arabic characters.

EDILE Makes use of true-type fonts. Therefore, it is based on font resource model. We use six tables to determine which glyph must be displayed : Isolated_Form, Initial_Form Medial_Form, Final_Form, Exceptional_Isolated_Form, and Exceptional_Final_Form tables. Each table includes Unicode values according to the glyph form. These tables offers a mapping between character codes and glyph identifiers.

In Arabic, letters of a string progressively change forms with the use of the contextual analyzer. Indeed, only the two last letters of this string change. We use finite states automata to implement our algorithm. States are the forms of the two latest letters and transitions are labeled by the previous character type. Previous character indicates the one that precedes the last character read by the analyzer. Only two actions are possible: "Exceptional character (E)" which means that the previous character belongs to the "Exceptional_Form" table and "Unexceptional character (U)" which means the contrary. Figure 7 shows the graph associated with the automata.

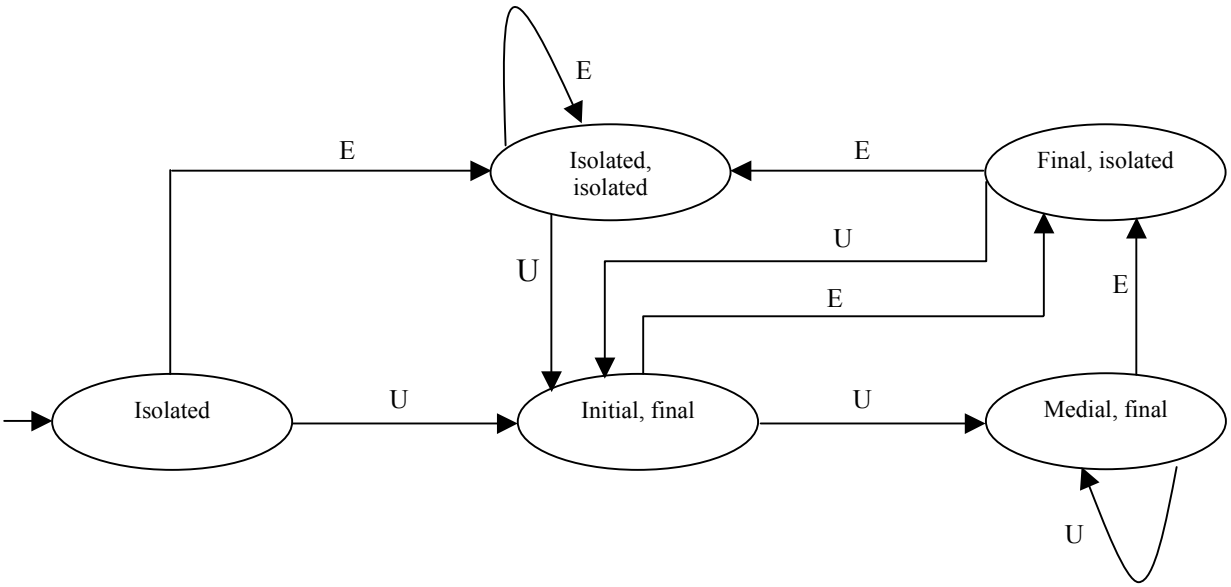


Figure 7: Contextual analysis automata.

The analyzer receives a character having isolated form. So the initial state is made up only of the first letter shape. There is transition from a state to another after reading a new character. The automata stops when the string finishes. Thus, all states can be viewed as final states. Figure 8 shows an example of use of this automata.

Rendering word: نظام

- Read character = ن
  Rendering = ن ("isolated" state = the initial state)
- Read character = ظ
  The previous character (ن) is unexceptional
  The transition "U" is selected
  Rendering = نظ ("initial, final" state)
- Read character = ا
  The previous character (ظ) is unexceptional
  The transition "U" is selected
  Rendering = نظا ("medial, final" state)
- Read character = م
  The previous character (ا) is exceptional
  The transition "E" is selected
  Rendering = نظام ("final, isolated" state)

Figure8: An example of the contextual analyzer.

Ligature resolution

Here, we only deal with the LAM_ALIF ligature. The other variants follow exactly the same logic. Our approach solves rendering ligatures according to the two ligature's letters. For LAM_ALIF ligature, it is necessary to take into account various LAM forms. Two ligature's forms can be produced according to the rules below:

- If the character (i-1) = initial LAM or isolated LAM and the character (i) = ALIF (arbitrary form), then we must replace the character (i-1) and the character (i) by the LAM_ALIF isolated ligature.

- If the character (i-1) = final LAM or medial LAM and the character (i) = ALIF (arbitrary form), then we must replace the character (i-1) and the character (i) by the LAM_ALIF final ligature.

This method can be viewed as a particular case of the contextual analysis. Arabic rendering requires a collaboration between contextual analyzer and ligature resolution process. Figure 9 shows the diagram of the automata associated with the contextual analysis supplemented by the ligatures resolution. The added states describe LAM_ALIF ligature shape. We treat ligatures as being exceptional characters since they have only two forms: isolated and final.

Transitions are also modified to consider at the same time the previous character and the current character. So, the transition "(E, x)" means that the previous character is exceptional and the read character is of unspecified type (i.e., exceptional or not). The transition "(LAM, ALIF)" indicates that the previous character is a LAM (its form is determined by the second variable of the output state) and the current character is an ALIF (isolated form). Finally, the transition "(U, x)-{LAM_ALIF}" means that the previous and the current characters do not form a LAM_ALIF sequence and that the previous character is unexceptional.

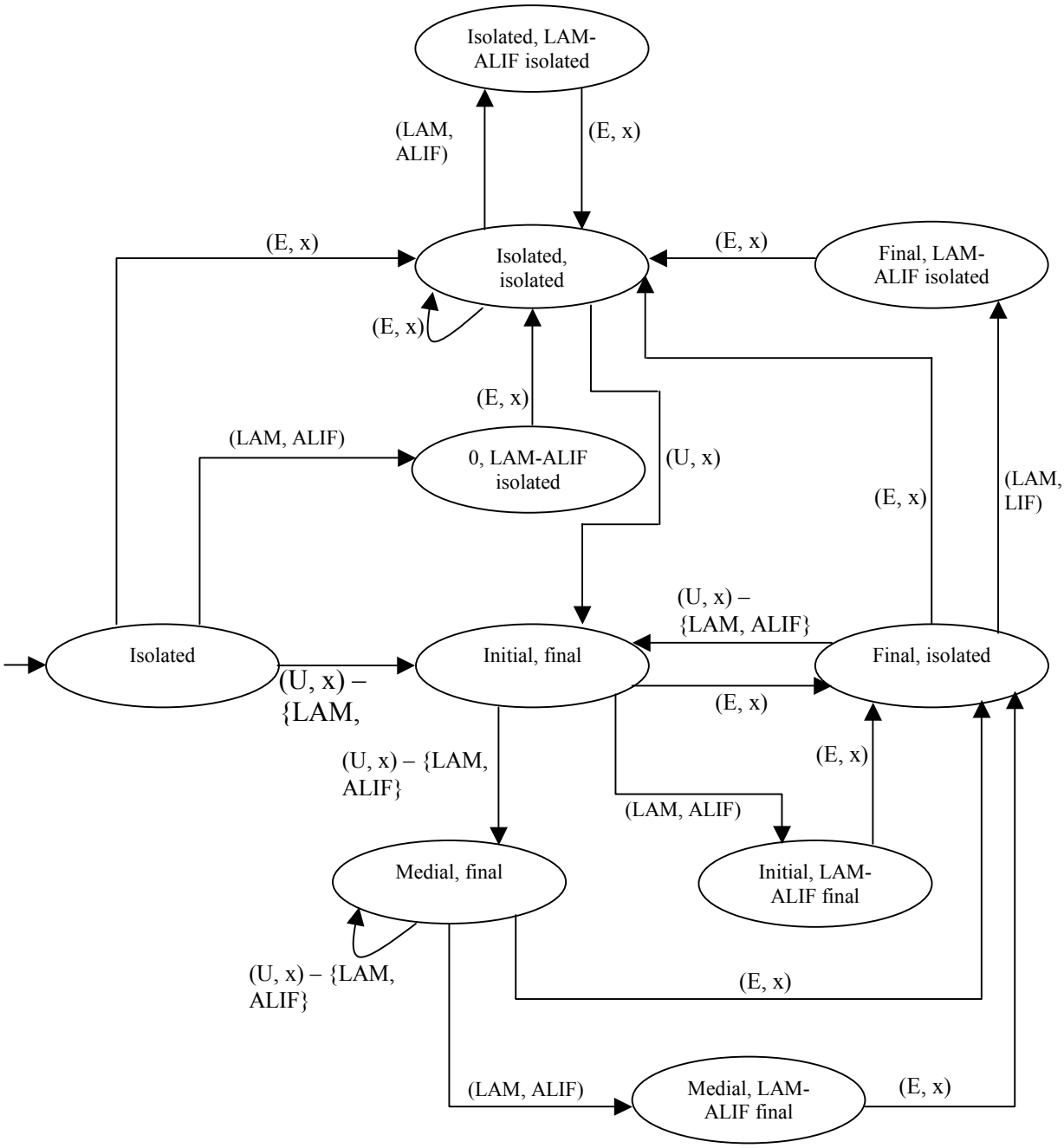


Figure 8: Contextual analysis completed by the ligatures resolution.

6. Conclusion

In this paper we discussed relevant aspects of internationalization and we presented an overview over Arabic localization for the EDILE web application for student examinations. We also introduced an architecture to provide multilingual support for EDILE. Based on Java

resource bundles, this architecture separates the source code from the user interface. It describes the EDILE components used for proper language and cultural presentation, including virtual keyboard management. EDILE is a dynamic environment where the multilingual and other resources are specified at run time.

In the discussion of Arabic localization, we introduced an algorithm for the contextual analysis and ligature resolution for the Arabic script. This algorithm is presented in the form of a state machine. Arabic keyboard management is provided by implementing a virtual keyboard.

Acknowledgements

We would like to thank Mr. Edwin Hart of the "Johns Hopkins University" (Applied Physics Laboratory), USA for his availability, his interesting advise, and his relevant updates which enabled us to write this paper. We would also like to thank the anonymous referees of this paper for their interesting and very helpful comments and suggestions.

References

[Amara 96] F. Amara and F. Portaneri, "Arabization of Graphical User Interfaces", *International User Interfaces*, Elisa Del Galdo and Jakob Nielsen, John Wiley & Sons, ISBN: 0-471-14965-9 (hardcover). New York, 1996.

[Gillam 99] R. Gillam, "Developing Global Applications in Java", *15th International Unicode Conference*, San Jose, California, August 30-September 2, 1999.

[Hart 99] E. Hart, "The Unicode Character-Glyph Model: What you Need to Know about Processing and Rendering Multilingual Text", *15th International Unicode Conference*, San Jose, California, August30-September 2, 1999.

[Henda 97] M. B. Henda, "Vers une normalisation des pratiques de communication dans le contexte d'un multilinguisme intégral (arabe/latin) : entre la diversité des usages et les contraintes technologiques", Acte du séminaire international "Penser les usages", France, 27-29 May, 1997.

[ISO 9995] ISO/IEC 9995-1:1994, "Information Technology – Keyboard layouts for text and office systems, Part 1: General principles governing keyboard layouts".

[Microsoft 97] Microsoft Typography, "The TrueType rasterizer", June 30, 1997 http://www.microsoft.com/typography/what/raster.htm.

[Mrabet 98] R. Mrabet and M. D. El Kettani, "EDILE, Exam DIstance Learning Environment", *7th World Conference on Continuing Engineering Education*, pp. 282-286, Turing, Italy, May 10-14, 1998.

[Nasri 97] S. Nasri, "TELESUN: A world wide multimedia TELEteaching System for Universities. Vers l'Université du futur : une université agéographique", *Regional Symposium on The Arab World and The Information Society*, Tunis 4 - 8 May 1997.

[Romerio 76] G. F. Romerio, "L'arabe voyellé en informatique", Project Report of UNESCO/PNUD MOR/73/024, Rabat, IERA, 1976.

[Tazi 99] M. Tazi, "Le télé-enseignement : classification et évaluation", Master's thesis of diplôme d'études supérieures approfondies, UFR ASIC, Mohammed V-Souissi University, ENSIAS, Rabat, Morroco, December, 1999.

[TR 15285] ISO/IEC TR 15285:1998, "Information Technology – An operational model for characters and glyphs".

[Unicode 00] The Unicode Consortium, *The Unicode Standard, Version 3.0*, Addison-Wesley, 2000.