Building and Deploying Internationalized and Localized Software

Lect.drd. Cristian USCATU
Catedra de Informatică Economică, A.S.E. Bucureşti

With the spread of ICT to every corner of the world users face a problem: usually all interfaces are written in the native tongue of the software authors. For those within the phenomenon, it might not be such a big deal, since this language is usually English and most of them speak it due to the nature of their work. Still, there are cases in which the language is not English and using that particular piece of software may prove to be a challenge. On the other hand, many regular users may not have a clue about English and even less about other languages. Since learning a new language for each new piece of software is not an option, they may have two choices: learn "by heart" the respective interfaces or find a localized version, in their native tongue. Lately, more and more software developers build localized versions for deployment in specific countries. Planning ahead saves a lot of work, time and code in internationalizing and localizing an application.
***Keywords**: Delphi, internationalized application, localized versions, translation tools, integrated translation environment.*

Internaţionalizare şi localizare

Internaţionalizarea unei aplicaţii constă în crearea posibilităţii ca ea să funcţioneze în mai multe seturi de convenţii culturale (numite „locale" în terminologia Windows). Schimbarea setului de convenţii se poate face în funcţie de nevoi, la momentul utilizării aplicaţiei. Dintre aceste convenţii, cea mai vizibilă este limba. În Windows un astfel de set de convenţii este descris printr-o pereche de coduri, reprezentînd ţara şi limba.

Localizarea unei aplicaţii constă în „traducerea" ei astfel încît să corespundă unui anumit set de convenţii culturale. Pe lîngă traducerea elementelor de interfaţă, pot să apară şi alte modificări specifice unei ţări. De exemplu, în cazul aplicaţiilor economice pot fi modificări legate de legislaţia specifică (impozite şi taxe etc.). Localizarea poate fi realizată uşor folosind instrumentele din categoria „Translation Tools", incluse în unele versiuni Delphi.

Internaţionalizarea aplicaţiilor

Internaţionalizarea unei aplicaţii se realizează în trei etape generale:

- Folosirea de cod capabil să manipuleze şiruri de caractere din diferite standarde (ANSI, OEM, UNICODE etc.).
- Proiectarea interfeţei astfel încît să nu fie afectată negativ de schimbările necesare pentru localizare.
- Izolarea resurselor care trebuie localizate (în principal şiruri de caractere, dar pot fi şi imagini sau altele).

În cele ce urmează detaliile se vor referi la mediul de dezvoltare Delphi de la Borland (sînt valabile şi pentru mediul CBuilder al aceluiaşi producător, cu detaliile de implementare specifice).

1. Pregătirea codului. Codul trebuie să fie capabil să trateze toate tipurile de şiruri de caractere ce pot fi întîlnite în diferitele localizări.

1.1. Seturi de caractere. Sistemul de operare Windows este distribuit în mai multe versiuni localizate. Pentru ediţiile occidentale este folosit setul ANSI Latin1 (cod 1252). Pentru ediţiile est-europene se folosesc se folosesc seturile corespunzătoare, pentru ediţia japoneză setul Shift JIS (cod 932) etc. Seturile de caractere se împart în general în 3 categorii: mono-octet, multi-octet şi caracter larg (wide character). Sistemele de operare Windows şi Linux suportă atît seturi monocaracter cît şi multicaracter şi Unicode.

Într-un set monocaracter, fiecare octet reprezintă un caracter. În această categorie intră

setul ANSI, folosit în sistemele occidentale. Într-un set multicaracter, unele caractere sunt reprezentate pe un octet, altele pe mai mulți octeți. Primul octet din fiecare caracter este numit octet conducător. În general primele 128 de caractere ale unui set multicaracter reprezintă caracterele din setul ASCII 7 şi sunt reprezentate pe un octet. Orice octet cu valoarea mai mare de 127 este un octet conducător pentru un caracter reprezentat pe mai mulți octeți. Caracterul nul nu poate fi reprezentat decît de caractere mono-octet. Seturile de caractere multi-octet (de exemplu setul bi-octet DBCS) sunt folosite în special pentru limbile asiatice.

Setul ANSI şi setul OEM. Fiecare utilizator poate specifica un anumit set de caractere care să fie folosit de calculatorul său (prin codul de pagină). Acesta va fi numit setul OEM. Uneori trebuie făcută translația între setul Windows (ANSI) şi setul OEM, ceea ce poate ridica probleme.

Seturi multi-octet. În cazul limbilor asiatice, nu e posibilă o translație unu-la-unu între caracterele limbii respective şi un set mono-octet. Alfabetul pictografic asiatic conține prea multe caractere, depăşind posibilitățile de reprezentare ale unui set mono-octet. Trebuie folosite seturi multi-octet. Tipul de dată *AnsiString* poate conține o combinație de caractere mono-octet şi multi-octet. Într-un set multicaracter, octetul conducător ia valori dintr-un domeniu rezervat (corespunzător setului respectiv). Al doilea octet şi următorii pot avea orice valoare, din set sau din afara lui. Ca urmare, nu se poate spune despre un octet individual dintr-un şir ce reprezintă. Singura posibilitate este parcurgerea întregului şir, de la început şi interpretarea sa, caracter cu caracter. Atunci cînd se lucrează cu seturi multicaracter, trebuie folosite funcții care ştiu să interpreteze în acest fel şirurile. Delphi oferă o multitudine de funcții de bibliotecă adecvate acestui scop.

Tabelul 1. Funcții de bibliotecă pentru manipularea şirurilor de caractere multi-octet

AdjustLineBreaks	AnsiStrLastChar	ByteToCharIndex	ExtractRela-
AnsiCompareFile-	AnsiStrLComp	ByteToCharLen	tivePath
Name	AnsiStrLIComp	ByteType	FileSearch
AnsiExtractQuotedStr	AnsiStrLower	ChangeFileExt	IsDelimiter
AnsiLastChar	AnsiStrPos	CharToByteIndex	IsPathDelimiter
AnsiLowerCase	AnsiStrRScan	CharToByteLen	LastDelimiter
AnsiLowerCaseFile-	AnsiStrScan	ExtractFileDir	StrByteType
Name	AnsiStrUpper	ExtractFileExt	StringReplace
AnsiPos	AnsiUpperCase	ExtractFileName	WrapText
AnsiQuotedStr	AnsiUpperCaseFileName	ExtractFilePath	
AnsiStrComp			
AnsiStrIComp			

În lucrul cu astfel de şiruri trebuie avut în vedere că lungimea în octeți nu corespunde întotdeauna cu lungimea în caractere. Nu trebuie trunchiate la întîmplare aceste şiruri pentru a nu rupe un caracter multi-octet. Neştiind dinainte lungimea unui caracter, nu trebuie transmise caractere ca parametri subprogramelor, ci pointeri către caractere sau şiruri de caractere.

Caractere largi. O altă abordare o constituie reprezentarea tuturor caracterelor pe un număr egal de octeți. Folosirea a numai doi octeți pentru fiecare caracter extinde posibilitățile de reprezentare suficient pentru problemele actuale. Un astfel de set este Unicode, care foloseşte doi octeți pentru reprezentarea fiecărui caracter. Un şir Unicode va conține perechi de octeți care reprezintă cîte un caracter. Caracterele Unicode sunt numite şi caractere largi iar şirurile formate din astfel de caractere se numesc şiruri cu caractere largi. Pentru compatibilitate, primele256 de caractere ale setului Unicode reprezintă caracterele setului ASCII 8. Sistemul Windows suportă setul Unicode UCS-2 iar Linux suportă supesetul UCS-4. Delphi lucrează cu UCS-2 pe

ambele platforme.
Folosire caracterelor largi prezintă unele avantaje. Majoritatea presupunerilor făcute în lucrul cu şiruri de caractere mono-octet pot fi făcute şi aici, ceea ce nu era posibil în cazul seturilor multi-octet. Există o relaţie directă între numărul de caractere din şir şi numărul de octeţi necesari pentru reprezentare. Nu există pericolul ruperii unui caracter în două sau al confundării celui de al doilea octet al unui caracter larg cu octetul conducător al altui caracter. Dezavantajul major îl constituie numărul redus de funcţii API care lucrează cu astfel de şiruri. Din această cauză, componentele VCL reprezintă şirurile de caractere folosind ori seturi mono-octet ori seturi multi-octet. Translaţia între reprezentarea multi-octet şi cea pe caractere largi la fiecare operaţie asupra unui şir ar consuma timp şi cod suplimentar, nefiind mereu justificată, dar ar putea fi o soluţie mai rapidă pentru unele prelucrări, mai ales atunci cînd e nevoie de corespondenţa unu-la-unu între seturile de caractere largi şi partea de început a seturilor mono-octet.

1.2. Includerea funcţionalităţii bidirecţionale în aplicaţii. Unele culturi nu folosesc stilul de scriere citire european, de la stînga la dreapta. În special în culturile Orientului mijlociu textul se scrie/citeşte de la dreapta la stînga iar numerele de la stînga la dreapta. Astfel de limbi sunt numite bidirecţionale (BiDi).
Proprietatea BiDiMode. Pentru implementarea acestor limbi, Delphi pune la dispoziţie proprietăţile *NonBiDiKeyboard* şi *BiDiKeyboard* în obiecte de tipul *TApplication*. Acestea permit specificarea modelului de tastatură utilizat atunci cînd textul trebuie citit de la stînga la dreapta respectiv de la dreapta la sînga. În plus, componentele din biblioteca de componente vizuale prezintă proprietăţile *BiDiMode* şi *ParentBiDiMode*. Proprietatea *BiDiMode* permite unei componente să îşi ajusteze automat aspectul atunci cînd aplicaţia este localizată. Ea modifică aşezarea textului, aşezarea barei de defilare verticale şi alinierea textului. Dacă o componentă afişează sugestii („Hint” în Windows), atunci acestea se vor conforma modului indicat de proprietatea *BiDiMode*. Proprietatea *ParentBiDiMode* arată dacă o componentă foloseşte aceeaşi valoare a proprietăţii *BiDiMode* ca şi părintele său (componenta care o conţine) – valoarea True – sau dacă foloseşte propria valoare. Atunci cînd se doreşte un comportament unitar la schimbarea setului de convenţii locale, e bine ca toate componentele să moştenească de la componenta de cel mai înalt nivel (aplicaţia) această proprietate.
Componentele care au proprietăţi de tip text (de exemplu „Name”) vor afişa în pagina inspectorului de obiecte şi proprietatea *BiDiMode*. Aceasta este de tipul enumerativ *TBiDiMode* definit astfel:

```
type TBiDiMode = (bdLeftToRight,
bdRightToLeft, bdRightToLeftNoAlign,
bdRightToLeftReadingOnly);
```

Valorile au următoarele semnificaţii:

- *bdLeftToRight* – este valoarea implicită. Sub efectul ei, testul este desenat de la stînga la dreapta. Aliniamentul şi bara de defilare nu se modifică. Dacă este setată o tastatură latină, textul e introdus de la stînga la dreapta; dacă este setată o tastatură arabă, ebraică etc. textul va fi introdus de la dreapta la stînga (cursorul intră în modul „împingere”[1]).
- bdRightToLeft – textul este desenat de la dreapta la stînga, alinierea este modificată iar bara de defilare mutată în partea opusă. Textul este introdus de la dreapta la stînga, iar dacă se setează o tastatură latină se introduce de la stînga la dreapta (cursorul intră în modul „împingere”).
- bdRightToLeftNoAlign – textul este desenat conform ordinii de la dreapta la stînga, bara de defilare este mutată, dar nu se schimbă aliniamentul.
- bdRightToLeftReadingOnly – textul este desenat conform ordinii de la dreapta la stînga, dar aliniamentul nu se schimbă şi bara de defilare rămîne pe locul normal.

[1]În modul normal cursorul se deplasează după fiecare caracter introdus (spre dreapta pentru tastaturile latine, spre stînga pentru tastaturile arabe, ebraică etc.). În modul „împingere” cursorul rămîne pe loc iar textul este împins în direcţia inversă la fiecare caracter introdus (spre dreapta pentru tastaturile corespunzătoare ţărilor din Orientul Apropiat, spre stînga pentru tastaturile latine).

Figura 1 prezintă efectul celor patru valori. Am folosit 4 componente de tip *TRichEdit*, fiecare avînd setată proprietatea *BiDiMode* cu una din valorile posibile, în ordinea prezentată mai sus (în ordinea stînga-dreapta, sus-jos în imagine). Convenţiile locale au fost setate pentru o ţară arabă (pentru a putea afişa corect textul corespunzător unei tastaturi arabe). În fiecare componentă a fost introdus textul „acesta este un text acesta este un text", pe mai multe linii. Prima parte a fost introdusă folosind tastatura engleză, iar partea a doua folosind tastatura arabă.

Fig. 1. Efectul diferitelor valori ale proprietăţii *BiDiMode*

Metoda FlipChildren. Schimbarea alinierii şi a direcţiei de desenare a textului nu este suficientă. Ar trebui mutate toate componentele interfeţei şi realiniate din partea opusă. Acest lucru este uşor de realizat folosind metoda *FlipChildren.* Ea este disponibilă pentru componentele de tip container (care pot conţine alte componente – de exemplu *TForm*, *TPanel* şi *TGroupBox*). Metoda poate realinia componentele de pe primul nivel inferior (fiii direcţi) sau poate produce realinierea şi în cadrul fiilor direcţi, pe toate nivelurile. Realinierea se poate face şi în momentul proiectării interfeţei, prin opţiuni ale mediului (din meniul pop-up asociat containerului căruia vrem să îi realiniem fiii, se selectează opţiunea *Flip Children*).
Există şi alte metode ale componentelor vizuale care pot fi folosite în adăugarea funcţionalităţilor bidirecţionale.

1.3. Facilităţi specifice convenţiilor locale. Pentru unele seturi de convenţii locale se poate controla metoda de transformare a secvenţelor de taste apăsate în text (în special pentru limbile asiatice) – „input control method" IME. Controalele care lucrează direct cu text oferă proprietăţile *IMEName* (specifică metoda de control) şi *IMEMode* (specifică în ce mod vor fi convertite caracterele introduse) în acest scop. Variabila globală *Screen* oferă informaţii despre metodele disponibile şi despre setul de convenţii locale activ pe calculatorul utilizatorului în momentul execuţiei programului.

2. Proiectarea interfeţelor. Proiectarea interfeţelor pentru aplicaţii care vor rula în variante localizate trebuie făcută avînd în vedere modificările ce apar la fiecare din aceste variante. Elementele care trebuie avute în vedere sunt: textele, imaginile, formatele, ordinea de sortare, combinaţiile speciale de taste.
Textele îşi modifică dimensiunea prin traducere. În general cele mai scurte versiuni sunt cele din limba engleză. Toate elementele de interfaţă trebuie să fie capabile să afişeze texte mai lungi. Trebuie evitate prescurtările, care nu pot fi traduse în unele limbi (cele cu alfabet pictografic). Trebuie avut în vedere că mărirea dimensiunii se manifestă mai pregnant în cazul textelor scurte. Valorile estimative rezultate din practică sînt:

Lungime în caractere (în limba engleză)	Creştere estimată
1-5	100%
6-12	80%
13-20	60%
21-30	40%
31-50	20%
peste 50	10%

Imagini. În mod ideal imaginile nu trebuie să conţină text, deci nu trebuie traduse. Dacă trebuie să apară text în imagini, ideal ar fi ca el să fie construit ca o etichetă cu fond transparent, suprapusă pe imagine. Astfel este mult mai uşor de tradus şi nu sînt necesare mai multe versiuni ale imaginii. În afară de problema textului, elementele grafice trebuie gîndite avînd în vedere elementele culturale specifice ţărilor unde se va distribui aplicaţia. Nu trebuie folosite elemente sau simboluri specifice unei culturi sau care pot fi ofensatoare pentru altele (de exemplu imaginea unui porc în cazul culturilor musulmane).

Formatele pentru diferite elemente (dată, timp, valori monetare, numere de telefon) trebuie de asemenea traduse pentru fiecare ţară ţintă. În cazul formatelor din Windows nu sînt necesare acţiuni speciale, pentru celelalte este treaba programatorului să asigure traducerea.

Sortarea se face diferit în unele limbi, ceea ce poate influenţa elemente ale interfeţei. Limbile europene conţin diacritice, care diferă de la o limbă la alta şi intervin diferit în ordinea de sortare. Mai mult, în unele limbi există combinaţii de două caractere care sînt considerate un singur caracter şi tratate ca atare la sortare (de exemplu, în spaniolă combinaţia *ch* e considerată un singur caracter, aflat între *c* şi *d*). În alte limbi un caracter e tratat ca fiind o pereche de caractere care intervin fiecare în parte în ordinea de sortare (de exemplu caracterul *eszett* din limba germană).

Combinaţiile speciale de taste pot ridica probleme deoarece unele caractere nu sînt disponibile pe toate tastaturile. Ideal în aceste combinaţii apar tastele funcţionale şi cifrele (disponibile pe toate variantele de tastatură).

3. Izolarea resurselor. Pentru localizarea unei aplicaţii, o sarcină importantă o constituie traducerea textelor folosite de aplicaţie. Pentru a evita modificări extinse asupra codului, textele trebuie separate de acesta. Delphi separă automat resursele pentru dialoguri, meniuri şi bitmap-uri în fişiere de resurse (.dfm). Pentru a separa şi celelalte texte folosite, o metodă convenabilă o constituie folosirea unor constante simbolice. Aceste constante se declară folosind cuvîntul rezervat *resourcestring* (în loc de *const*). Avantajul este că textele respective vor fi salvate ca resurse şi legate de executabil. Modificarea lor se poate face fără a recompila aplicaţia. Ideal toate resursele de tip şir de caractere trebuie incluse într-un singur unit.

Biblioteci de resurse cu legare dinamică. Nivelul următor de separare îl constituie crearea bibliotecilor de resurse cu legare dinamică. Acestea conţin toate resursele unei aplicaţii şi nimic altceva. Aceste biblioteci se pot crea cu ajutorul unui instrument specializat, Expertul pentru biblioteci de resurse cu legare dinamică. În scopul internaţionalizării trebuie creată cîte o astfel de bibliotecă pentru fiecare limbă ţintă. Fişierul care stochează fiecare bibliotecă va avea o extensie în care primele două litere reprezintă codul setului de convenţii locale iar a treia ţara în care se aplică.

Fişierele care compun o aplicaţie conţin toate resursele necesare. Pentru a le înlocui cu resursele traduse, corespunzătoare unei anumite ţări, aplicaţia trebuie să includă şi biblioteca de resurse corespunzătoare. La lansarea aplicaţiei, aceasta detectează setul de convenţii locale de pe maşina pe care rulează apoi caută o bibliotecă de resurse care are acelaşi nume cu aplicaţia şi o extensie care corespunde convenţiilor locale şi limbii. Dacă este găsită o astfel de bibliotecă, resursele din ea

vor înlocui resursele implicite ale aplicației. Dacă nu este găsită, se va căuta o bibliotecă de resurse care să corespundă măcar limbii locale. Dacă nici aceasta nu e găsită se vor folosi resursele implicite ale aplicației (cele din fişierul executabil şi bibliotecile dinamice care compun aplicația).

Se poate forța o aplicație să utilizeze alt set de convenții locale decît cel curent pe maşina pe care rulează. Pentru aceasta trebuie adăugată în registrul Windows o cheie care dă prioritate unui set de convenții menționat explicit. Această metodă e utilă atunci cînd se testează diferite variante de localizare ale unei aplicații fără a schimba setul de convenții locale de pe maşina pe care se lucrează.

Folosind biblioteci de resurse cu legare dinamică, se poate distribui o aplicație care se adaptează automat la convențiile de pe maşina utilizatorului, doar pentru că are biblioteca de resurse corespunzătoare.

În afară de localizarea resurselor la pornirea aplicației, este posibilă schimbarea din mers a limbii aplicației prin încărcarea unui set diferit de resurse, dintr-o altă bibliotecă dinamică de resurse.

Bibliografie

1. *** Borland Delphi Help
2. *** Things to Think About when internationalizing a Program, www.webmasterworld.com/forum47/1616.htm
3. Dan Haught - Internationalizing Access Applications, www.fmsinc.com/tpapers/intaa