



# Internationalization Challenges & Solutions

Xoriant

<http://www.xoriant.com/>

Copyright © (2002) Xoriant Solutions, Inc. All rights reserved.

The contents of this document are confidential and proprietary to Xoriant and no part of this document should be reproduced, published in any form by any means, electronic or mechanical including photocopy or any information storage or retrieval system nor should the information in this document be disclosed to third parties without the express written authorization of Xoriant.

## Table Of Contents

<b>1. SYNOPSIS.....</b>	<b>3</b>
1.1. KNOWING XORIENT.....	3
1.2. XORIENT’S TECHNICAL COMPETENCE .....	3
<b>2. XORIENT INTERNATIONALIZATION SERVICES .....</b>	<b>5</b>
<b>3. OVERVIEW AND CONCEPT.....</b>	<b>5</b>
<b>4. DESIGN FACTORS IN INTERNATIONALIZATION .....</b>	<b>6</b>
4.1. GEOGRAPHIC FACTORS .....	6
4.2. FUNCTIONAL FACTORS.....	7
4.3. REGULATORY FACTORS .....	8
4.4. CULTURAL FACTORS .....	8
4.5. ECONOMIC FACTORS .....	9
<b>5. DESIGN GUIDELINES FOR INTERNATIONALIZATION.....</b>	<b>9</b>
5.1. PRESENTATION LAYER .....	9
5.2. APPLICATION LAYER.....	10
5.3. DATABASE.....	10
<b>6. CHALLENGES IN DEVELOPMENT FOR INTERNATIONALIZATION... </b>	<b>10</b>
<b>7. CHALLENGES IN TESTING FOR INTERNATIONALIZATION.....</b>	<b>12</b>
<b>8. INTERNATIONALIZATION USING JAVA/ORACLE .....</b>	<b>12</b>
<b>9. BENEFITS OF XORIENT’S SERVICES.....</b>	<b>18</b>
<b>10. WORK CULTURE AT XORIENT .....</b>	<b>20</b>
<b>11. CASE STUDIES.....</b>	<b>21</b>
<b>12. APPENDIX:.....</b>	<b>22</b>



# 1. Synopsis

## 1.1. Knowing Xoriant

**Xoriant** is an Information Technology (IT) Services company, based in Silicon Valley, California and having operations / offices in New Jersey and India (Mumbai, Pune and Kolkata). A **SEI CMM Level-4** certified company; Xoriant specializes in providing **Software Engineering Services** to its clients in the United States, Europe, Asia and South America.

Xoriant Corporation offers a continuum of tactical software and IT services to both creators and users of information technology. Its highly flexible and customized support models focus on three primary vertical markets: Banking Financial Services and Insurance (BFSI), Telecom and High Technology Supply Chain Management.

Xoriant has the qualities of an ideal software/IT partner – flexibility, versatility, agility and value. Xoriant builds world-class products, yet reducing total costs, leveraging cutting-edge skills, reducing time-to-market and optimizing end-to-end product development.

Xoriant's software talent comprises developers of cutting-edge products on one side and business-facing solutions implementers on the other side. This gives Xoriant the versatility to tackle complex environments as well as to achieve healthy crossbreeding between technology and user experience.

Xoriant has been selected by many of the world's leading companies to help them optimize their customer relationships and make the most of their technology investments. Xoriant's **clients** include leading Hi-tech companies, Global Financial services organizations like **CitiGroup, Wells Fargo, Charles Schwab**, Major Telecom companies like AT&T, Nokia Siemens, **Nortel, Lucent** Technology and consulting companies like **KPMG, Bearingpoint**.

Xoriant's global delivery model offers both agility and value. Xoriant strives to help the creators and users of software achieve more with less: people, time, and resources.

Xoriant has more than 450 employees who provide leading edge technology solutions to global customers. **Xoriant has a 20,000 sq. ft state of the art facility at Mumbai.** About 50 percent of the employees are based out of Mumbai, India.

## 1.2. Xoriant's Technical competence

Over the last 17 years Xoriant has been facilitating their customers to build, maintain and test technology solutions. In addition, Xoriant has several proprietary technologies and methodologies to simplify the process of outsourcing product development activities.

Xoriant's expertise includes analysis, architecture, design, building, testing, and integrating business critical systems to provide unified business solutions to clients. Proven methodologies, mature Processes, expertise in diverse and niche technologies, in-Tandem working of onsite/offshore teams and product-based service delivery positions Xoriant uniquely amongst competitors.

Xoriant Corporation brings to the table more than a decade of deep experience and immense horizontal expertise in product engineering services. In our 15-year product lifecycle track

record, we have partnered with small and large product companies, who are constantly facing shorter time-to-market and high innovation cycles. For these companies, our expertise in different stacks of core technology layers and component recycling solutions to enhance productivity, bring high ROI and increased customer loyalty from product use and support.

Xoriant's globally-aware team has in-depth experience in bringing products to market for software and systems companies. We understand the intricacies of the complex people-process mix. We believe in building brands by explicitly linking it to our clients' perceptions of innovation. With broad experience in areas like compatibility testing, technical support, sustaining and internationalization, our goal is to expand the product acceptance across industries and geographies.

We understand that today's truly global markets require services mechanisms that are a mix of onsite, offsite, and offshore infrastructure and resources. As a globally operational company, we bring flexible services to our clients - right delivery at the right price and in the right time zone and geography. With the extended flexible services bonus, our software engineering services allow clients to "take ownership" of the following benefits:

- Proven ROI
- Increased development/testing and support productivity
- Lower costs
- Enhanced quality
- Faster time to market
- Customer satisfaction

<b>Languages &amp; Utilities</b>	Java/J2EE, C, C++, VC++, C#, XML, SQL, Perl, Assembly, VisualBasic, Autosys, Hermes Packaging, ASP, Unix shell scripting, .NET
<b>Applications</b>	  
<b>Middleware</b>	      
<b>Databases &amp; BI Tools</b>	       
<b>Platforms</b>	    
<b>Testing Tools</b>	  

This technical white paper provides details on Internationalization concepts, Internationalization challenges and Xoriant's recommended approach to Internationalization.

## 2. Xoriant Internationalization Services

Xoriant India functions as the Offshore Development center for multiple Global clients. Being a Global services organization, **Internationalization is one of the key offerings to its customers.**

Internationalization services offer huge ROI to its customers. The main focus areas for such services are,

- Architect, design & develop applications / products with Internationalization
- Technical assessment, re-engineering & transition of existing applications / products for Internationalization
- Testing of applications / products with Internationalization

Xoriant works with their partners for Translation (Localization) services.

## 3. Overview and Concept

Internationalization is the process of designing software so that it can be adapted (localized) to various languages and regions easily, cost-effectively, and in particular without engineering changes to the software. Localization is performed by simply adding locale-specific components, such as translated text, data describing locale-specific behavior, fonts, and input methods.

An internationalized program has the following characteristics:

- ❖ With the addition of localized data, the same executable can run worldwide.
- ❖ Textual elements, such as status messages and the GUI component labels, are not hard-coded in the program. Instead they are stored outside the source code and retrieved dynamically.
- ❖ Support for new languages does not require recompilation/code change.
- ❖ Culturally dependent data, such as dates and currencies, appear in formats that conform to the end user's region and language.
- ❖ It can be localized quickly.

The term internationalization is abbreviated as **I18N**, because there are 18 letters between the first "I" and the last "N". I18N is an inherent part of the system architecture and should therefore be considered from the initial architectural design stages. This will involve the removal of locale specific elements from the code. Locale specific information on the GUI will be stored in text files (aka property files or resource bundles). Localization will involve the translation of these files into various languages.

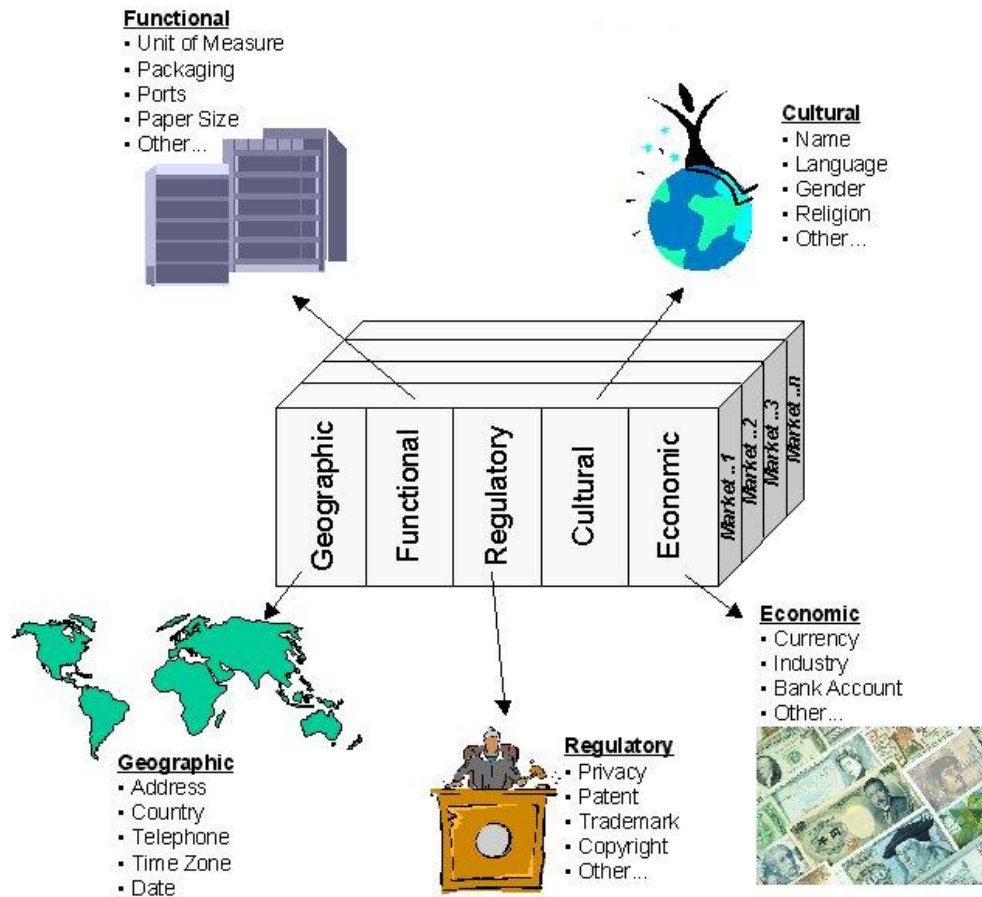
Similarly, *localization* (which you may see referred to as **L10N**) refers to the process of making an application suitable for a particular geographic location, country, or culture. Usually, the most time-consuming portion of the localization phase is the translation of text. Other types of data, such as sounds and images, may require localization if they are culturally sensitive. Localizers also verify that the formatting of dates, numbers, and currencies conforms to local requirements.

## 4. Design factors in Internationalization

There are many factors to consider when building a culturally and geographically neutral application. For example, how do you effectively display numbers, currencies, dates, times, text, and sort orders of strings, along with the entire user interface? One of the most important aspects of creating a global application is to avoid hard-coding any locale-specific data or text.

Internationalization support offers many challenges to the Architects' community. In this document, some of the critical design factors have been described.

### Web Globalization Dimensions with Abbreviated Examples



### 4.1. Geographic factors

In this category, the design needs to consider geographic specific deviations like Address format, Telephone number conventions, Time zone variance, Date variance, and so on.

#### **Date Manipulation:**

The format for displaying date and time information varies according to local conventions. The displayed form of dates can vary quite a bit from language to language. Not only do the words for the days and months change, but also so do the order of the fields themselves and the punctuation around them.

For example, the following date formats are acceptable in the United States:

10/4/2004  
October 4th 2004  
Monday, October 04, 2004  
Mon, Oct 4, 2004  
4-OCT-2004

While France has the following date formats:

04/10/04  
4 octobre 2004  
Lundi 4 octobre 2004  
lun. 4 octobre 2004  
4-oct.-04

One can see that there are many differences between the accepted formats for these two countries. For instance, the U.S. follows mm/dd/yy for the first date format, while for the same format France follows dd/mm/yy.

In some countries, the calendar system in use also changes. For example, in Hebrew, April 2 1999 is the 16<sup>th</sup> of Nisan, 6759. Japan has changed to the Gregorian calendar, but they number their years by the reigns of the emperors: For example, the year 1999 is 11 in Heisei in Japan.

#### **Text Translation and Formation:**

Translating messages in a software program is different from translating a natural language essay. For example, menu items may have only one or two words and these are not very meaningful on their own. The translator must therefore be careful to base his translations so that they are consistent with existing software packages in the target locale, otherwise users may be confused and acceptance of the translated software will be adversely affected.

Another difficulty is encountered with technical terms. If a technical term does not have an “official” translation, the translator may have to invent a new, unique term for it. In such a case, the translator must choose between inventing a new term or using an existing, but possibly misleading term for the translation in the target language. Either way, the step is risky, since one cannot be sure if the translated term may mislead the user or be accepted by the public.

Formatting the text messages that need to be displayed to the user also pose several problems. The rules of grammar that apply to one language are often different from those that apply to other languages.

## **4.2. Functional factors**

In this category, the design needs to consider functional changes across different regions. Some of the examples of such deviations are Units of measurement, Packaging standards, alignment, and so on.

**Number Formatting:**

Number formats also change according to local conventions. For example, consider the number **1,234**.

An American, will read the number as one thousand two hundred and thirty-four. But a Frenchman will read this number as one point two three four. That is because in France, a comma instead of a period denotes the decimal point. The comma is used in many European countries including Great Britain.

This could obviously lead to serious misunderstandings if operations are carried out across country boundaries.

The decimal point is not the only character that can vary. Given below are some of the ways in which the number, **One thousand two hundred and thirty-four point five six** would be written in some countries.

<u>Country</u>	<u>Format</u>
America	1,234.56
France	1 234,56
Germany	1'234.56

**Aligning of Text:**

The text alignment is different in different languages. Mostly the languages have left alignment, but there are few languages that are right aligned like Hebrew, Arabic, Urdu, etc. Hence for these languages what one would require showing at right would require showing to the left. In addition any formatting done to the UI for alignment (left or right) would require becoming opposite in right aligned languages as compared to left aligned languages.

**Acronyms Usage:**

The usage of acronyms in English language should be avoided as to get a translation of all the words in other languages is not possible. This will build irregularity in the UI when we do translation from English to other languages.

**4.3. Regulatory factors**

In this category, the design should address regulatory compliance requirements. It could be in the area of privacy, patent, trademark, copyright etc. For example, any software having an in-built tax calculation module should be aware of rules for tax computation for varying geographies. Another example is the foreign exchange trading system, in which interest earned on the security deposit for a forward contract would be 0.8% in one region and 1.2% in some other region.

**4.4. Cultural factors**

In this category, design should address culturally sensitive issues like Language, Color preferences, Religious preferences, and so on.

**Colors:**

Different cultures have different meanings for different colors. A color, which is agreeable to one culture, may be offensive to other cultures. For example, in most Western countries, black is generally used to denote death, while white means the same thing in Asian countries.

The following table shows some cultural associations of color (Russo and Boor, 1993)

Colors	Red	Blue	Green	Yellow	White
U.S.	Danger	Masculinity	Safety	Cowardice	Purity
France	Aristocracy	Freedom, peace	Criminality	Temporary	Neutrality
Egypt	Death	Virtue, faith, truth	Fertility, strength	Happiness	Joy
India	Life, creativity		Prosperity, fertility	Success	Death, purity
Japan	Anger, danger	Villainy	Future, youth, energy	Grace, nobility	Death
China	Happiness	Heavens, clouds	Ming Dynasty, heavens, clouds	Birth, wealth, power	Death, purity

#### 4.5. Economic factors

In this category, factors pertaining to commercial transactions will be taken into consideration. Some the areas to be considered are Currency, Industry, Bank account, and so on.

##### Currency:

Handling currency can be even more difficult than handling other kinds of numbers. Design consideration items include currency symbols and where they get placed relative to the number, alternate decimal-point characters, how many decimal places to show, and how much to round the value.

As if all that weren't complicated enough, software design should address requirement exchange rates between different currencies. This is particularly true when an application has to display monetary values in more than one currency.

## 5. Design guidelines for Internationalization

Design is the most crucial element for Internationalization and is the key success factor. A few of the design guidelines have been listed below.

### 5.1. Presentation layer

- Allow customization for icons and other graphical elements. Use different icons for different countries/locales and load them at run time. Colors and sounds should also be customizable as they are also culturally sensitive.
- Design graphical components like menu items, buttons, labels, and so on, to allow text of variable length. Sometimes there is a restriction on the size of a menu item. The menu item size may exceed the allocated size in some language. In such cases it is preferable to show a portion of the text fitting into the allocated size. This menu item can be superimposed with a tool tip, which displays the complete text.
- All functions that deal with the format of user-visible components should be identified and replaced with functions, which are specially designed format functions that allow greater flexibility in handling grammar structure and formatting style.
- Do not have text embedded in the graphics as far as possible as this will require the graphics to be rebuilt again in the different languages.
- For web-based projects it would be best to have all alignments to be mentioned in style

sheets and to have one style sheet per language. As far as possible shift code into style sheets instead of having them hard coded in the html page.

## 5.2. Application layer

- Identify language dependent features and develop them as components. Example: Spell Checker and Grammar Checker in Editor Software.
- Avoid the hard coding of language specific strings that will propagate to the GUI. If there are no alternatives, then define string constants and use those in the code. Isolate user visible strings in a separate file. This reduces the chance of translators accidentally damaging the system.
- Always keep internationalization in mind when dealing with data input, validation and output. ISO 639 defines the standard set of two-letter, lowercase codes used to identify languages from around the world. This standard is not stable and can change over the years. Three languages - Hebrew, Indonesian and Yiddish - have two distinct ISO-639 codes. New software should follow these simple guidelines to ensure maximum compatibility:
  - When reading in data, read and interpret both variations of ISO language code
  - When writing out data, use the oldest variation of the language.
- For sorting, do not use the `String.equals()` or `String.compareTo()` in Java or the overloaded string operators in C++ for comparisons, as they simply perform bitwise comparison and this won't work for most languages.
- Traditional searching algorithms won't work when using languages other than English. Use functions, which allows support for locale specific comparisons, for example, in java the `Collator` and `CollationElementIterator` classes make it possible to write faster and more powerful search routines that will support multiple languages.

## 5.3. Database

- Store date as `nvarchar2` if not very sure of the target language and do the required processing in the application such as sorting. This solves the date format problems.
- Script should have different column lengths for Double Byte Character Set (DBCS) and Single Byte Character Set (SBCS) languages. This is more important if the same version of the software would have multiple installations across geographies. However, for centralized deployment, an appropriate character set should be chosen.
- While dealing with different currencies, care should be exercised to allocate field size properly (particularly for currencies like the Italian Lira and Japanese Yen).
- For multi-lingual support, one would require to set the collation in the database to support the language.

## 6. Challenges in development for Internationalization

Internationalization offers many challenges to the development community. Xoriant has been providing Internationalization services for several years and some of the major challenges have been highlighted below based on real-life experiences.

**Font availability:**

Normally internationalized software packages are developed in Unicode encoding so that they can display all languages that are covered in Unicode. However Unicode is not yet the most widely used encoding method in the world of computing. In most cases the systems are run on platforms with a default encoding method other than Unicode, which can cause problems. There is no guarantee that the user's system platform can provide the font that is required by the application. It is possible to ship the software application with the font, but that will be the user's font options. If a font, which the software uses, is not available on the platform on which it is run, the text displayed to the user will be erroneous.

**Font accessibility:**

Ideally, Unicode can be displayed properly as long as there are fonts that match the characters to be displayed on screen, and generally most fonts support more than one language. For example, the Times New Roman font supports Greek, Baltic, Central European, Cyrillic, Turkish and Western characters. For an unrestricted multilingual system to be available, however, one would need a font that supports the characters of every language. A search for a font that includes all characters that are supported by Unicode has proven to be unsuccessful, but if there is such a font that supports every language, the size of that font is likely to be quite large. If a universal font becomes widely available and distributed in the future, an unrestricted multilingual system could be achieved by providing a universal font.

**Screen Layout:**

While designing the Graphical User Interface for a software application, designers generally use fixed sizes for components such as Buttons, Text Fields, Labels, etc. This does not work the same way for an internationalized product as the translated text may require more or less space to be displayed. This may result in an ungainly appearance of the interface.

**Operating System components:**

It is not necessary that the operating system supports all the features provided by the programming language. In such cases the functionality is limited to the one provided by the operating system. For example, frame titles in a Java- built system are rendered by the operating system. Therefore unless the user's operating system supports Unicode and has a suitable font for the text in the frame title, the title cannot be displayed properly and must be limited to supported languages, such as English.

**Integrated subsystems**

Software systems sometimes consist of subsystems that to a certain extent are independent of the main system. This independence can lead to a lack of information about the system and the subsystem or, more precisely, it can lead to a lack of coordination. For example, a word processor can use a third party developed spell checker, which is not internationalized.

**Database**

The database used by application should support internationalized characters.

## 7. Challenges in testing for Internationalization

It is a well-known fact that it is less expensive to fix a bug, which is discovered early in a product development process. Depending on the level of I18N support built into the base code, the testing that needs to be done will vary.

When a product is fully internationalized, the internationalized version should support all of the features required in the localized versions. For instance, even though the user interface may still be in English, an i18n product supports the correct character sets required by the localized versions.

The most serious globalization problem is functionality loss, either when a System locale is changed or later when accessing input data.

Some functionality problems can be detected as display problems:

- Question marks (???) appearing instead of displayed text indicate problems in Unicode-to-ANSI conversion.
- Random High ANSI characters "¼, «, †, %o, ‡, ¶, etc. appearing instead of readable text indicate problems in ANSI code using wrong code page.

The appearance of boxes, vertical bars, or tildes (default glyphs)[□, |, ~] indicates that the selected font cannot display some of the characters.

## 8. Internationalization Using JAVA/Oracle

Internationalization solutions can be provided with multiple technology solutions. Only Java/Oracle based Internationalization approach is highlighted here. Both provide features, which are very well suited for the purpose of Internationalization. Moreover, both provide support for Unicode.

### Why Unicode?

People in different countries use different characters to represent the words of their native languages. Nowadays, most applications, including email systems and web browsers, are 8-bit clean, that is, they can operate on and display text correctly provided that it is represented in an 8-bit character set, like ISO-8859-1.

There are far more than 256 characters in the world - think of Cyrillic, Hebrew, Arabic, Chinese, Japanese, Korean and Thai - and new characters are being invented now and then. The problems that come up for users are:

It is impossible to store text with characters from different character sets in the same document.

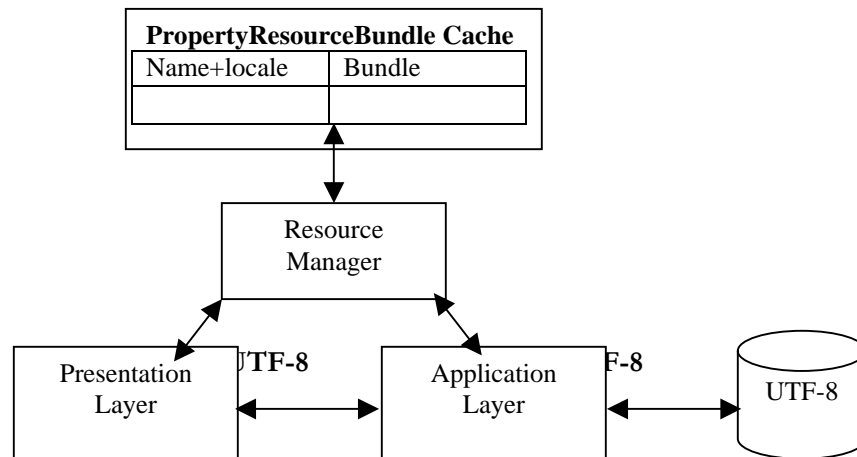
As long as every document has its own character set, and recognition of the character set is not automatic, manual user intervention is inevitable.

New symbols like the Euro are being invented. ISO has issued a new standard ISO-8859-15, which is mostly like ISO-8859-1 except that it removes some rarely used characters (the old currency sign) and replaces it with the Euro sign. If users adopt this standard, they have documents in different character sets on their disk, and they start having to think about it daily. But computers should make things simpler, not more complicated.

The solution of this problem is the adoption of a worldwide usable character set. This character set is Unicode.

## Java support for I18N

One of the major reasons why java is being chosen for I18N, is that, java comes with a vast set of in built libraries for I18N. All characters in java are stored as 16 bits encoded in Unicode, which makes it as the most preferred language for I18N. Java itself is internationalized and supports more than 100 locales, depending on the version, and also supports bi-directional languages such as Arabic and Hebrew.



The above diagram is of a 3-tier architecture application. The presentation layer and the Application layer both use Resource Manager, which stores locale information. Data encoded in Unicode format is transferred between these 3 layers. The details are discussed in the following section.

**Resource bundles** contain locale-specific objects. When the program needs a locale-specific resource, a String for example, it can load it from the resource bundle that is appropriate for the current user's locale. In this way, one can write program code that is largely independent of the user's locale isolating most, if not all, of the locale-specific information in resource bundles.

One resource bundle is, conceptually, a set of related classes that inherit from ResourceBundle. Each related subclass of ResourceBundle has the same base name plus an additional component that identifies its locale. For example, suppose the resource bundle is named MyResources. The first class one is likely to write is the default resource bundle, which simply has the same name as its family--MyResources. One can also provide as many related locale-specific classes, as one needs: for example, perhaps one would provide a German one named MyResources\_de\_DE.

Each related subclass of ResourceBundle contains the same items, but the items have been translated for the locale represented by that ResourceBundle subclass. For example, both MyResources and MyResources\_de\_DE may have a String that is used on a button for canceling operations. In MyResources the String may contain Cancel and in MyResources\_de\_DE it may contain Abbrechen.

**PropertyResourceBundle** is a concrete subclass of ResourceBundle that manages resources for a locale using a set of static strings from a property file.

Unlike other types of resource bundle, PropertyResourceBundle is not subclassed. Instead, the properties files, containing resource data, are supplied. ResourceBundle.getBundle() will automatically look for the appropriate properties file and create a PropertyResourceBundle that refers to it. The resource bundle name that is passed to ResourceBundle.getBundle() is the file name of the properties file, not the class name of the object that is returned.

The advantage of PropertyResourceBundle is that a resource bundle doesn't have to be compiled when new properties are added or existing ones are modified.

So it is generally recommended to use PropertyResourceBundles.

ResourceBundle Example:

```
class Sample{
public static void main(String args[]) {
Locale currLocale = new Locale (args[0], arg[1]);
ResourceBundle messages = ResourceBundle.getBundle("MBundle",currLocale) ;

System.out.println (messages.getString ("greetings"));
System.out.println (messages.getString ("inquiry"));
System.out.println (messages.getString ("farewell"));
}
}
```

The above program will show the following result:

```
C:\> java Sample en US
```

```
Hi there!
How are you!
Good Bye!
```

```
C:\> java Sample fr FR
```

```
Bonjour
Comment allez-vous?
Au revoir.
```

#### **Property file format: (containing French strings)**

```
greetings=Bonjour
inquiry=Comment allez-vous?
farewell=Au revoir.
```

#### **Currency Format Example:**

```
class SampleCurrency{
...
Locale currLocale = new Locale(args[0], args[1]) ;
Double currency = new Double(987.50) ;
NumberFormat currFormatter = NumberFormat.getCurrencyInstance(currLocale) ;
String currencyOut = currFormatter.format(currency) ;
System.out.println(currencyOut+" " +currLocale.toString());
}
}
```

The above program will show following result:

```
java SampleCurrency en US
$987.50 en_US
```

```
java SampleCurrency fr FR
    987,50 Ç fr_FR
```

### Date Format Example:

```
Locale currLocale = new Locale(args[0], args[1]) ;
// instantiate a GregorianCalendar with a given date
GregorianCalendar cal = new GregorianCalendar (1965,Calendar.JUNE, 16);

// get a DateFormat object using a LONG style and defined locale
DateFormat df = DateFormat.getDateInstance(DateFormat.FULL,currLocale);

// format the date
String frDate = df.format(cal.getTime());
System.out.println(frDate) ;
```

The output of above program will be:

```
C:\> java DateManip fr FR
mercredi 16 juin 1965
C:\> java DateManip en US
Wednesday, June 16, 1965
```

### Sorting and Searching

Conventionally, when character data is sorted, the sort sequence is based on the numeric values of the characters defined by the character-encoding scheme. Such a sort is called a binary sort. Such a sort produces reasonable results for the English alphabet because the ASCII and EBCDIC standards define the letters A to Z in ascending numeric value.

Note, however, that in the ASCII standard all uppercase letters appear before any lowercase letters. In the EBCDIC standard, the opposite is true: all lowercase letters appear before any uppercase letters.

To produce a sort sequence that matches the alphabetic sequence of characters for a particular language, another sort technique must be used that sorts characters independently of their numeric values in the character-encoding scheme. This technique is called a linguistic sort. A linguistic sort operates by replacing characters with other binary values that reflect the character's proper linguistic order so that a binary sort returns the desired result.

Oracle supports both. See the NLS\_SORT parameter. Note that linguistic sort will obviously not be as fast as a binary sort. GROUP BY uses a binary sort, so use ORDER BY instead.

If sorting is to be done using Java, the `String.equals()` or `String.compareTo()` method should not be used for comparisons, as they simply perform bitwise comparison and this won't work for most languages. The simplest solution is to use the abstract class `java.text.Collator`.

`Collator` class knows how to compare strings in a language-sensitive way. When comparing natural-language strings, create a `Collator` object and use its `compare()` method to compare strings.

For example,

**Wrong way**

```
boolean didSwap = true;
while (didSwap) {
    didSwap = false;
    int top = list.length;
    for (int i = 0; i < --top; i++) {
        if (list[i].compareTo(list[i + 1]) > 0) {
            String temp = list[i];
            list[i] = list[i + 1];
            list[i + 1] = temp;
            didSwap = true;
        }
    }
}
```

**Correct way**

```
Collator coll = Collator.getInstance();
boolean didSwap = true;
while (didSwap) {
    didSwap = false;
    int top = list.length;
    for (int i = 0; i < --top; i++) {
        if (coll.compare(list[i], list[i+1]) > 0) {
            String temp = list[i];
            list[i] = list[i + 1];
            list[i + 1] = temp;
            didSwap = true;
        }
    }
}
```

**Sort using Java or using Oracle?**

The advantage of using Java would be to make the software RDBMS independent. Not all databases will support multi-lingual sorting, and if one has to switch to a different database vendor, this functionality may be lost, or at the very least be implemented differently.

The advantage of using Oracle would be for performance reasons, but it would depend on how much data is to be sorted.

Like sorting, searching through text is much more complicated when one has to worry about locale-sensitive rules. In addition to the complexities of natural languages, Unicode allows different ways of encoding characters. For example, to search for the surname “Müller” in a block of German text, one needs to be able to handle the following situations:

- The character ü may substitute for ue.
- The character ü can be represented as the 16-bit Unicode value \u00FC
- The character ü can be represented as the two 16-bit Unicode values \u0075 and \u0308.

Not all methods in the `String` class are internationalized, so methods such as `indexOf`, `compareTo` are not able to handle these situations. Java does not supply simple internationalized compliant replacements for these methods, although it does provide infrastructure, which allows building personalized matching routines. This support comes in the form of the `CollationElementIterator` class.

### Java I18N/L10N toolkit

Java has come up with the Java Internationalization (i18n) and Localization (l10n) Development Toolkit, which is being developed by the China Technical Development Center (TDC). This toolkit makes Internationalization and Localization easier by helping with some key processes:

- Checking Java files for methods, classes, class constants, and strings that are locale specific
- Checking for hard-coded or inconsistent message strings
- Translating message strings and protecting strings that should not be translated
- Generating new resource bundles
- The toolkit has four independent tools, which can be used separately. They are:
  - I18N Verifier to test whether a program is international or not and suggest corrections
  - Message Tool to find and optionally correct hard-coded or inconsistent messages
  - Translator to translate messages in a resource bundle file into the target locale language and protect strings that should not be translated
  - Resource tool to merge more than 2 resource files to a new resource bundle or find the differences in the resource files

### Database changes for I18N

An internationalized application should be able to store and retrieve data in multiple languages and the choice of database encoding is intrinsic to this support. Oracle's National Language Support (NLS) architecture allows storing, processing, and retrieval of data in native languages. It ensures that database utilities and SQL error messages, sort order, date, time, monetary, numeric, and calendar conventions automatically adapt to the native language and locale.

The character datatypes `CHAR` and `VARCHAR2` are specified in bytes, not characters. Hence, the specification `CHAR(20)` in a table definition allows 20 bytes for storing character data. This works out well if the database character set uses a single-byte character-encoding scheme because the number of characters will be the same as the number of bytes. If the database character set uses a multibyte character-encoding scheme, there is no such correspondence. That is, the number of bytes no longer equals the number of characters since a character can consist of one or more bytes. Thus, column widths must be chosen with care to allow for the maximum possible number of bytes for a given number of characters.

### Migrating the database character set

When migrating to a new database character set, the Export and Import utilities can handle character set conversions from the original database character set to the new database character set. However, character set conversions can sometimes cause data loss or data corruption. For example, while migrating from character set A to character set B, the destination character set B should be a superset of character set A. Characters that are not available in character set B will be converted to replacement characters, which are usually specified as '?' or '¿' or other linguistically-related characters. For example, ä (a with an umlaut) will be converted to 'a'. The target character

set defines replacement characters.

Migrating to Unicode is an attractive option because UTF8 contains characters from most legacy character sets.

To ensure that there is no loss of data during the migration process, it is recommended to run Oracle's Character Set Scanner Utility (CSSCAN) before migration. This utility will produce a report that will identify any tables or columns where character set migration will result in loss of data. See the following document for details on running this utility and interpreting its reports:

[http://www.oradoc.com/ora817/addendum.817/a85455/nlsadden.htm - 1000539](http://www.oradoc.com/ora817/addendum.817/a85455/nlsadden.htm-1000539)

Provided that the CSSCAN report shows no problems, the database character set can be changed to UTF8 by performing the following steps: (requires DBA privileges)

```
SHUTDOWN IMMEDIATE; -- or NORMAL
<do a full backup>
STARTUP MOUNT;
ALTER SYSTEM ENABLE RESTRICTED SESSION;
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=0;
ALTER SYSTEM SET IAQ_TM_PROCESSES=0;
ALTER DATABASE OPEN;
ALTER DATABASE CHARACTER SET UTF8;
SHUTDOWN IMMEDIATE; -- or NORMAL
STARTUP;
```

## 9. Benefits of Xoriant's services

Xoriant leverages **methodology and technology to differentiate from its competitors**. Xoriant team's architecture and design capabilities on Internationalization have evolved over a period of time and it has incorporated some of the industry best practices. **Xoriant's drive on quality & productivity ensures that product roll out is done on time, in budget and with high quality.**

Xoriant's past experiences of working for internationalized product enhancement as well as IT services based internationalized projects has created **technical, functional as well as domain specific skill sets** in abundance.

Xoriant has been involved in different types of engagement models with its customer. Xoriant has a successful track record of working with banks, financial services companies and telecom companies. [Click here](#) for more.

Working with aggressive clients like CitiGroup, Maxblue and Microsoft has helped us understand as well as **implement high quality standards** to deliver the best possible services. Xoriant's approach towards Internationalization is not limited to achieve best results only at the final delivery stage, but to ensure that every phase of Internationalization nurtures the end-result into a perfect product as per customer expectations.

To meet these objective, correct actions needs to be taken at all the stages of life cycle (as summarized below).

Phase	Actions taken	Benefits
<b>Requirements definition and gathering</b>	<ul style="list-style-type: none"> <li>❖ Right questions asked at the right time to the right people.</li> <li>❖ Understand the client requirement. Scope of Internationalization and Localization.</li> </ul>	<ul style="list-style-type: none"> <li>❖ Clearly defines end-user needs thus avoiding high cost defects in the deliverables.</li> </ul>
<b>Proof of concept and Architecting</b>	<ul style="list-style-type: none"> <li>❖ Dedicated team of experienced Architects.</li> <li>❖ Develop proof of concept.</li> <li>❖ Recommend architecture layer alternatives to support Internationalization</li> <li>❖ Assessment of existing product and identify 'Gaps' for Internationalization.</li> </ul>	<ul style="list-style-type: none"> <li>❖ Enables customer to take appropriate decisions.</li> <li>❖ Detailed 'Gap Analysis' along with roadmap of implementation minimizes the development risk.</li> </ul>
<b>Analysis and Design</b>	<ul style="list-style-type: none"> <li>❖ Use of Design modeling tools.</li> <li>❖ Implementation of design patterns.</li> <li>❖ Recommend multiple technology solutions with pros and cons.</li> </ul>	<ul style="list-style-type: none"> <li>❖ Robust product architecture and design meeting customer requirements.</li> <li>❖ Improve product life.</li> </ul>
<b>Coding and Implementation</b>	<ul style="list-style-type: none"> <li>❖ Internal coding standards with flexibility to implement customer coding standards.</li> <li>❖ Implementation of CMM compliant quality processes.</li> <li>❖ Knowledge repository on best practices, and past learnings.</li> </ul>	<ul style="list-style-type: none"> <li>❖ Consistent Quality deliverables.</li> <li>❖ Improve problem resolution time.</li> <li>❖ Reduced rework efforts.</li> <li>❖ Alignment to customer processes.</li> </ul>

<b>Translation</b>	<ul style="list-style-type: none"> <li>❖ Work with partners for translation.</li> </ul>	<ul style="list-style-type: none"> <li>❖ Focus on core technical areas for Internationalization and leave the translation to experts.</li> </ul>
<b>Testing and QA</b>	<ul style="list-style-type: none"> <li>❖ Mature test management process.</li> <li>❖ Team of experts for specialized testing like performance testing and stress testing.</li> <li>❖ Use of appropriate tools in testing cycle.</li> <li>❖ Perform platform compatibility testing.</li> <li>❖ Perform Localization testing.</li> <li>❖ Defect tracking to closure.</li> </ul>	<ul style="list-style-type: none"> <li>❖ Reduce the risk of requirement gaps.</li> <li>❖ Optimized testing efforts due to processes and tools.</li> <li>❖ End to end testing by experts.</li> </ul>
<b>Documentation</b>	<ul style="list-style-type: none"> <li>❖ Multi-lingual documentation.</li> </ul>	<ul style="list-style-type: none"> <li>❖ Global customers have choice to get documentation in their preferred language.</li> </ul>
<b>Release and Maintenance</b>	<ul style="list-style-type: none"> <li>❖ Maintain multiple versions.</li> <li>❖ Implement standard backup and restoration process.</li> </ul>	<ul style="list-style-type: none"> <li>❖ Access to product change history.</li> <li>❖ Risk management and higher sustainability.</li> </ul>

## 10. Work Culture at Xoriant

- ❖ Strong belief in **teamwork** and collective efforts to achieve common goals
- ❖ Encouragement for **new and innovative ideas** to improve process
- ❖ **Knowledge sharing** among team members
- ❖ **Develop core competence** in specific fields as per the role, although with **equal opportunities** to explore different work areas
- ❖ Develop **domain knowledge** to provide 'Value add'
- ❖ Continuous improvements through **formal training**
- ❖ **Flexible working hours** to accommodate customer time zone, for better response time
- ❖ Continuous efforts for **higher quality standards** in development as well as support services

## 11. Case studies

- **Maxblue**

A complete banking portal developed for Deutsche Bank, Brazil.

The portal was developed using Java/J2EE. The application server used was Weblogic7.0. The database used was Oracle8i.

It provides the user with a vast range of services like online transaction, electronic transfer of money between accounts, buying and selling of mutual funds, planning tools, exhaustive content/advice on financial market, portfolio management. Globalization is also implemented in this application, as majority of the customers are Portuguese, who are more comfortable using the portal in Portuguese. Therefore, it was very vital to implement globalization in the system so as to facilitate the customers to use the system in Portuguese as well as English.

- **UniversaLink**

It provides an automated e-brokerage solution for the Dubai and Abu Dhabi stock exchange for end-to-end electronic trade processing linking to vital internal and external entities for real-time, transparent and paperless communications.

The application was developed using ASP, .NET. The database used was Microsoft SQL Server 2005.

UniversaLink offers fully integrated front-to-back office solutions facilitating web based client trade capturing, pricing, modeling, processing, accounting, performance and reporting. This is a full-fledged online brokerage application comprising of three core modules that can be implemented jointly or separately:

1. BaseLink (Accounting) module
2. eLink (online trading) module
3. OmniLink (trade capture / trader screen) module.

Globalization is also implemented in this application, more specifically, eLink (the web interface). This product is targeted mainly for the Dubai Financial Market (DFM) and Abu Dhabi Stock Market (ADSM). The end-users of this system are UAE-based brokers and citizens who are more comfortable using Arabic locales in their system. Therefore, it was very vital to implement globalization in the system so as to facilitate the brokers to use the system in Arabic, with a default option in English.

- **China Mobile**

The X-treme Service Activity Manager (XSAM) is a service mediation platform. It mediates between subscribers and service providers, allowing a mobile network operator to quickly create and manage a wide range of value-added services. This product supports internationalization and it has been localized for English and Chinese languages.

The product was developed using Java/J2EE. The application server used was Jboss4.0.2. The database used was Oracle10g.

China Mobile were using Aspire's Monternet as legacy CP/SP connecting point and Aspire's MISC is managing all the workflows and business logics. Current rules within CMCC is that CP/SPs have to go through Aspire's platforms to provide value added data services. Many provincial CMCCs do not like Aspire's platform since it is less flexible and hard to use. Therefore, provincial CMCCs want to build a "master SP" to provide their own branded services through their own platform.

Built new platform using XSAM not to replace the Aspire platforms mentioned above, but to move provincial CMCC's own branded premium services onto a different platform.

## 12. Appendix:

### Internationalization Terminology

Internationalization needs knowledge of key concepts and this section briefly describes some of these key concepts.

**Character:** A character in software development is an abstraction. The natural understanding of a character is that of a written character; one intuitively associates a certain graphic representation with a given character. This is what is called a *glyph*: the actual shape of a character image. A glyph appears on a display or is produced by a printer. Naturally, there can be many such representations. One can represent the characters ABC as: **ABC** or *ABC* or **ABC**. A set of glyphs is called a *font*. So indeed, one aspect of a character is its graphic representation. However, for the purpose of data processing in software development, a character also needs to have a data representation as a sequence of bits. This is called a *code*.

**Encoding:** A character-encoding scheme is a set of rules for translating a byte sequence into a sequence of character codes.

**G11N:** The abbreviation for globalization - 11 characters between g and n.

**Globalize, Globalization:** The term is used for the internationalization and the localization process together or the concept to produce software that works globally. The Localization Industry Standards Association (LISA) defines globalization as follows:

"Globalization addresses the business issues associated with taking a product global. In the globalization of high-tech products this involves integrating localization throughout a company, after proper internationalization and product design, as well as marketing, sales, and support in the world market."

**I18N:** The abbreviation for internationalization - 18 characters between i and n.

**Internationalize, Internationalization:** The process of enabling single source for the international market. Internationalization is the design and development of software in a way that allows it to be localized (translated) to other locales (languages) without the need to alter the source code. Common errors are due to both cultural and locale differences. The Localization Industry Standards Association (LISA) defines internationalization as follows:

"Internationalization is the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for re-design. Internationalization takes places at the level of program design and document development."

**L10N:** The abbreviation for localization - 10 characters between l and n.

**Localize, Localization:** The process of adapting, translating and customizing a product for a specific market (for a specific locale). The Localization Industry Standards Association (LISA) defines localization as follows:

"Localization involves taking a product and making it linguistically and culturally appropriate to the target locale (country/region and language) where it will be used and sold."

**Locale:** The features of the user's environment those are dependent on language, country, and cultural conventions. The locale determine convents such as sort order; keyboard layout; date, time, number and currency formats. In Windows, locales usually provide more information about cultural conventions than about languages. So simply put locales are simply a bunch of user preference information that is related to the user's language and sub-language.

**Multilingual, Multilanguage:** Supporting more than one language simultaneously. Often implies the ability to handle more than one script of character sets.

**Script:** A system of characters used to write one or several languages.

**Translation:** The process of translating a piece of text from one language (the source language) to another language (the target language). The translation process is one of the major parts of localization

**Unicode:** A global standard for worldwide usable character set.

**UTF-8:** 128 characters are encoded using 1 byte (the ASCII characters). 1920 characters are encoded using 2 bytes (Roman, Greek, Cyrillic, Coptic, Armenian, Hebrew, Arabic characters). 63488 characters are encoded using 3 bytes (Chinese and Japanese among others). The other 2147418112 characters (not assigned yet) can be encoded using 4, 5 or 6 bytes.