Internacionalização de Software – Requisitos Básicos

Cassius Marcellus do Carmo Figueiredo (1)
Paulo Renato Negri Canhadas(1)
Alexandre Rojas (2)

(1) Alunos do Bacharelado em Informática e Tecnologia da Informação- UERJ
(2) Docente Departamento de Informática e Ciencia da Computação- UERJ

Resumo

Este artigo descreve uma pesquisa sobre as práticas utilizadas nos processos de internacionalização e localização de software e destaca seus princípios básicos.
A elaboração do trabalho foi motivada pela pequena quantidade de referências teóricas elaboradas em Língua Portuguesa sobre o tema. As principais referências sobre o assunto foram desenvolvidas pelas grandes empresas de software por intermédio de consórcios (UNICODE) e entidades não-lucrativas que visam o estabelecimento de práticas comuns de desenvolvimento (LISA).

1. Introdução

Empresas investem milhões de dólares anualmente na produção e comercialização de softwares para computador em todos os países do mundo. Porém, nos últimos 15 anos ficou evidente que não é suficiente apenas vender um produto em um idioma que não o nativo do mercado onde ele está sendo inserido. Devido a imposições de mercado percebeu-se que era necessário um investimento ainda maior na adaptação e tradução do produto para os consumidores-alvo.

Assim, surgiram duas áreas de estudo que são responsáveis pelas duas etapas envolvidas no processo de adaptação. A internacionalização responde pelo planejamento e análise do produto de forma a abranger todos os conceitos relevantes e a posterior localização do produto, processo que visa a correta adaptação lingüística para o mercado-alvo.

Por outro lado, as metas agressivas impostas pelo governo federal brasileiro para a exportação de software, acentuou a importancia que as empresas produtoras de software tenham conhecimento das técnicas envolvidas necessárias ao mercado internacional

O objetivo principal deste trabalho e fazer uma revisão bibliográfica dos conceitos envolvidos na internacionalização de softwares de computador de forma que a localização do mesmo possa ocorrer de forma eficiente e econômica.

2. Planejamento da internacionalização de software

Geograficamente falando, uma localidade é um espaço físico. No contexto de software para computadores, uma localidade é um conjunto de informações associadas a um local e/ou cultura. Informações vinculadas à localidade incluem layouts de teclado para entrada de texto, dimensões de papel e envelopes, conjunto de caracteres ou faixas de codificação, direção do texto a ser apresentado ao usuário (da esquerda para direita, direita para esquerda, horizontal ou vertical) e métodos de entrada de dados.

O conceito de localidade pode também ser entendido como ‘o idioma X, como falado no país ou região Y’. A forma como um determinado idioma é falado pode não variar drasticamente entre um país ou região e outro, porém as convenções culturais e padronizações nacionais certamente serão diferentes. No entanto, existem outras localidades onde o idioma é bastante diferente entre países/regiões, como é o caso das localidades referentes ao português brasileiro e ibérico.

Como exemplo de suporte de sistemas operacionais ao conceito de localidade, pode-se citar o Microsoft Windows XP, que suporta 135 localidades. Entre elas encontramos cinco localidades diferentes de língua chinesa (Hong Kong, Macau, República Popular da China, Singapura e Taiwan), treze localidade de língua inglesa (Austrália, Belize,

Canadá, Caribe, Irlanda, Jamaica, Nova Zelândia, Filipinas, África do Sul, Trinidad, Reino Unido, Estados Unidos e Zimbábue), seis localidades de língua francesa (Bélgica, Canadá, França, Luxemburgo, Mônaco e Suíça) e ainda 16 localidades de língua árabe.

Na tabela a seguir, encontram-se informações ligadas a algumas localidades.

Localidade	**Inglês (Estados Unidos)**	**Português (Brasil)**	**Japonês**	**Árabe (Emirados Árabes Unidos)**
País/Região	Estados Unidos	Brasil	Japão	Emirados Árabes Unidos
Idioma	Inglês	Português (Br)	Japonês	Árabe
Escrita	Latina	Latina	Kana, Kanji	Árabe
Direção do texto	Esquerda para direita	Esquerda para direita	Esquerda para direita (Horizontal) Direita para esquerda (Vertical)	Direita para esquerda
Código de página Windows	1252	1252	932	1256
Símbolo de moeda	US$	R$	¥	د.إ
Formato de data longo	January 27, 2004	27 de janeiro de 2004	2004 年 01 月 27 日	٢٧ يناير ,٢٠٠٤
Formato de data curto	1/27/04	27/01/2004	04/01/27	04/01/27
Formato de apresentação de hora	1:00 pm	13h00	13:00	١:٠٠ م
Calendário	Gregoriano	Gregoriano	Gregoriano (Localizado)	Gregoriano (Localizado)
Tamanho de papel padrão	Carta (8-1/2 x 11 in)	A4 (210 x 297 mm)	A4 (210 x 297 mm)	A4 (210 x 297 mm)
Separador decimal	Ponto	Vírgula	Ponto	Vírgula
Separador de listas	Vírgula	Ponto e vírgula	Vírgula	Ponto e vírgula
Separador dos milhares	Vírgula	Ponto	Vírgula	Vírgula

Tabela 2.1 Informações referentes a algumas localidades

2.1 Internacionalização

Entende-se por internacionalização (também conhecida pela sigla L18n) o "processo de desenvolvimento de um produto de forma a funcionar com dados em diversos idiomas e que possa ser adaptado a diversos mercados sem alterações de engenharia" [SY2001]. O objetivo da internacionalização é apresentar ao usuário um aplicativo visual e funcionalmente idêntico nos diversos idiomas para os quais ele tenha sido localizado. A consistência da terminologia empregada é também muito importante para conjuntos (ou suites) de produtos, tais como a linha Office da Microsoft ou StarOffice da Sun. A expectativa dos usuários é que as versões internacionalizadas do produto dêem suporte às mesmas características básicas do aplicativo em seu idioma nativo e com o mesmo nível de qualidade. Além disso, as diferentes versões devem interagir entre si corretamente, portanto, um formato único de documento, que todas as versões leiam e interpretem corretamente, torna-se essencial.

Um exemplo real desta expectativa é a interface do Microsoft Windows XP. Como pode ser visto a seguir, a utilização de menus e ícones nas edições japonesa e americana é consistente, de forma que um usuário familiarizado com a versão em inglês tem grande chance de navegar corretamente pela versão em japonês, mesmo sem conhecer o idioma e vice-versa.

Figura 2.1.Painel de controle do Windows XP em inglês e japonês

Assim como a consistência visual é uma importante medida de como um produto foi bem desenvolvido, o suporte às convenções internacionais é igualmente importante. A característica mais importante é a entrada e visualização de caracteres não-latinos. No passado, aplicativos no idioma inglês eram limitados aos caracteres do conjunto ASCII. Esta limitação não é mais aceitável, principalmente após o surgimento da codificação de caracteres chamada Unicode. Com o suporte aos dados multilíngües agora é possível criar aplicativos que podem apresentar aos usuários documentos criados em praticamente todos os idiomas.

Outras convenções internacionais incluem regras para ordenação de strings, formatação de datas, horas, números e moeda. A parte do processo de internacionalização que o usuário não vê diretamente envolve codificação, teste, depuração e a definição dos mecanismos de tradução que serão utilizados. Usuários não familiarizados com os requisitos para criação de produtos internacionalizados certamente ficarão surpresos com a quantidade de variáveis envolvidas neste

processo e dos detalhes de projeto que precisam ser resolvidos.

2.2. Localização

A localização (também conhecida pela sigla L10n) é o "processo, subseqüente à internacionalização, de tradução e adaptação de um determinado produto, levando-se em consideração o conjunto de convenções culturais de um determinado mercado" [SY2001]. Um código-fonte internacionalizado é um requisito básico para aplicativos que serão lançados fora do mercado de origem. Porém, os gastos envolvidos e o esforço a ser empregado na localização podem variar, dependendo do mercado a ser atingido e o tipo de aplicação.

Alguns mercados podem ser considerados pequenos, portanto, o lançamento do produto deve ser considerado um investimento para o futuro. Outros mercados, porém, estão bem estabelecidos e a manutenção destes depende de um produto de alta qualidade e de investimentos consideráveis. Além disso, temos os pequenos mercados que estão em fase de expansão rápida. É possível estabelecer uma presença em mercados de menor porte lançando rapidamente um produto parcialmente localizado e, posteriormente, ir ampliando o nível de localização em revisões ou edições posteriores. A figura a seguir mostra os diferentes níveis de localização e sua dependência direta com o nível de risco e investimentos relacionados.

A tabela a seguir apresenta os níveis de localização e a relação entre o risco de sucesso de um software e o retorno financeiro:

Menor Risco, menor retorno financeiro:

1. Traduzir nada;

2. Traduzir a documentação impressa na embalagem;

3. Internacionalizar o código fonte;

4. Traduzir os menus e caixas de diálogo;

5. Traduzir o sistema de ajuda on-line, tutoriais, arquivos de exemplos e arquivos "Leiame";

6. Adicionar suporte a hardware específico do mercado-alvo;

7. Personalizar recursos

Maior risco, maior retorno financeiro

Caso o produto seja originariamente desenvolvido em inglês, é interessante testar a aceitação em mercados onde os usuários tenham o costume de adquirir produtos em língua inglesa até que uma versão completamente localizada esteja disponível. A próxima lista apresenta alguns exemplos de mercados onde produtos na língua inglesa são amplamente aceitos:

Pequenos mercados (por exemplo, Tailândia e Filipinas);

Mercados onde não há competidores para o produto;

Mercados onde o público-alvo fala inglês (comunidade científica, médica ou de alta-tecnologia).

Em mercados onde apenas produtos localizados obtêm sucesso, uma localização parcial é aconselhável para um lançamento rápido e caso não se possa pagar ou esperar pela localização completa. Como regra geral, quanto maior o nível de localização, maior a quantidade de questões a serem resolvidas pela equipe de desenvolvimento e maior a quantidade de problemas a serem resolvidos. Em mercados mais competitivos, recursos personalizados se tornam necessários e, em alguns casos, também o suporte a hardware local.

O processo de criação de um produto localizado envolve um nível elevado de comunicação entre diferentes partes de um time de desenvolvimento. É apresentado a seguir o fluxo tradicional do processo de desenvolvimento e localização.

Durante o desenvolvimento a equipe envia arquivos para a equipe de localização, responsável pela tradução do texto, redimensionamento de caixas de diálogo e, em muitos casos, recompilação. O programa executável localizado passa então à equipe de teste que reporta problemas relacionados à funcionalidade para a equipe de desenvolvimento e problemas relativos à interface com o usuário à equipe de localização. Todos os três grupos trabalham em conjunto para a solução dos problemas e o ciclo continua até que o produto seja considerado em sua versão final.

O processo de localização mais fácil é aquele onde a equipe responsável pelo produto considera as questões relativas à internacionalização durante a fase inicial de planejamento de recursos e análise de sistemas (início da fase de desenvolvimento). Pesquisas de marketing, testes de usabilidade e limitações de desenvolvimento irão também afetar diretamente o projeto do produto. A tradução e o teste do código em paralelo com o desenvolvimento ajudam na descoberta de falhas na codificação e torna mais fácil o processo de correção destas falhas. É importante também finalizar o máximo

possível da interface com o usuário e do planejamento de recursos bem antes do teste final. Cada mudança da interface causa uma necessidade de nova tradução, com conseqüentes alterações em documentação impressa, sistemas de ajuda, tutoriais e materiais de marketing, que por sua vez também necessitarão de nova tradução. Portanto alterações de última hora podem causar atrasos e altos custos no lançamento de produtos localizados e devem ser evitados.

3. Planejando uma aplicação internacionalizada

A decisão de lançar uma aplicação em diversos idiomas, ou pelo menos que reconheça escritas diversas (latinas ou não) é fundamental para que a equipe de desenvolvimento também decida corretamente como irá tratar a codificação e a interface com o usuário durante a etapa de planejamento do produto.

Como em qualquer projeto, grande ou pequeno, a chave para o sucesso é o planejamento. Poucas tarefas em desenvolvimento são tão difíceis e caras quanto reescrever um código já existente ou adaptá-lo para que funcione em outros idiomas. Com um bom planejamento pode ser criada uma aplicação que acomode múltiplos idiomas. Seguem algumas vantagens desta opção:

Ter apenas um arquivo executável para todas as plataformas e todos os idiomas reduz significativamente dificuldades e custos na etapa de desenvolvimento;

Uma compilação condicional torna-se desnecessária;

Não há a necessidade de códigos-fonte e times de desenvolvimento independentes;

Todos os idiomas podem ser lançados quase simultaneamente, o que pode aumentar a receita proveniente destas versões;

Atualizações de software são mais simples, pois os ajustes podem ser aplicados a todos os idiomas sem esforços adicionais de engenharia.

3.1. Identificando os requisitos

Uma parte importante do projeto de uma aplicação que será internacionalizada é a etapa de especificação das características do produto. Aspectos gerais devem ser levados em consideração, tais como:

- Definir o grau de internacionalização/localização desejado;
- Determinar recursos específicos a determinados mercados;
- Planejar uma interface com o usuário que seja localizável;
- Verificar as questões legais;
- Definir as opções de acessibilidade que estarão disponíveis.

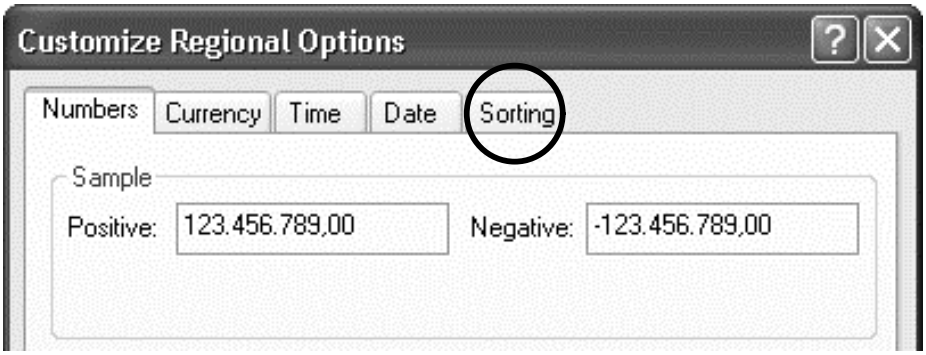
Portanto, fatores tais como quais idiomas e localidades serão contemplados e o nível de localização devem ser levados em consideração. Quando a equipe de análise estiver descrevendo a aparência e a funcionalidade dos novos recursos ele deverá também especificar quais destes recursos farão parte de cada um dos diferentes idiomas do produto e quando um determinado recurso será afetado pela configuração da localidade. Por exemplo, o que acontecerá quando o usuário entrar caracteres russos (alfabeto cirílico)? É necessário ajustar o tamanho das caixas de diálogo na versão japonesa? Caso a equipe não tenha familiaridade com outros idiomas, uma consulta a pessoas com este conhecimento se torna necessária.

Ter a certeza que todas as versões da aplicação têm as mesmas características que dêem suporte a todos os idiomas e localidades-alvo. Este é um processo que envolve basicamente os três passos a seguir:

- Identificar dos idiomas e localidades que serão suportados;
- Planejar e especificação de recursos que dêem suporte aos mercados, idiomas e localidades-alvo;
- Escrever um código-fonte que funcione bem em qualquer uma das localidades suportadas.

3.2. Detalhando os requisitos

Uma parte importante do processo de internacionalização é o correto detalhamento dos requisitos do mercado-alvo que se deseja alcançar. A seguir têm-se alguns dos requisitos básicos.



Configuração em Alemão

Configuração em Italiano

Figura 3.1 .Diferenças entre configurações do alemão e italiano (a interface foi mantida em inglês para facilitar o entendimento)

3.3 Planejando uma interface com o usuário localizável

Um fator muito importante no planejamento de uma aplicação a ser internacionalizada é a interface com o usuário. Devemos lembrar sempre que é da natureza humana tomar como "óbvio" o que lhe é confortável e familiar. Por exemplo, várias regras gramaticais de nosso idioma nativo, que aprendemos enquanto ainda muito jovens, tendem a ficar guardadas em nosso subconsciente e passamos a utilizar estas regras de forma inconsciente. Da mesma forma, pode ser tentador ou mais fácil assumir que determinadas expressões idiomáticas serão similares em outros idiomas. Um exemplo é a expressão "custa os olhos da cara" que utilizamos para identificar algo muito caro. Ela é similar na Língua Espanhola ("cuesta un ojo de la cara") e bastante diferente na Língua Inglesa ("costs an arm and a leg"), porém mantendo o mesmo sentido. Da mesma forma, no processo de desenvolvimento de uma aplicação, é fácil para os analistas e desenvolvedores desconsiderar as diferenças entre os idiomas. Seguem a seguir algumas regras para que o contexto geral de utilização da aplicação não seja afetado por maus hábitos de programação.

Não incluir strings da interface diretamente no código-fonte;

Não economizar espaço de armazenamento utilizando-se de concatenação de frases e/ou palavras. Desta forma, podemos evitar armadilhas causadas pelos diferentes idiomas, tais como flexão de verbos e variações de gênero e número;

Deixar sempre espaço nas caixas de diálogo de forma a acomodar as diferentes traduções e evitar soluções do tipo esconder um botão atrás do outro em caixas muito apertadas.

3.4 Aspectos legais

Um outro importante componente de personalização envolve a investigação de questões legais pertinentes a cada mercado-alvo. Torna-se necessária então uma consulta para verificar a legalidade de cada recurso "sensível" da aplicação.

Podemos citar como exemplo as leis específicas de determinados países quanto à utilização e exportação de algoritmos de criptografia, utilizados em simples procedimentos de compressão de arquivos e proteção contra cópia. Outro exemplo diz respeito à documentação que acompanha o produto: em determinados países, ela poderá ser mantida em inglês, porém em outros, como na França, ela deve estar obrigatoriamente no idioma local.

Além disso, em determinados países, como a Alemanha, as leis que regem a competição entre as empresas restringem a possibilidade de uma empresa explicitar que seu produto é melhor que o concorrente. Isso afeta diretamente o marketing do produto, podendo também afetar a documentação, sistemas de ajuda e arquivos de exemplo.

3.5 Personalizando os recursos

O processo de especificação de uma aplicação internacionalizada torna obrigatória a definição de quais recursos serão personalizados para determinados mercados. Durante a especificação, é possível verificar que recursos importantes para o mercado principal podem não fazer muito sentido em outros mercados e vice-versa. Um exemplo são as versões asiáticas do Windows XP que automaticamente instalam diferentes métodos de entrada de dados que não são instalados na versão em inglês.

Além disso, alguns recursos essenciais a todas as versões da aplicação podem exigir que um código diferente seja executado para determinados idiomas, por questões de complexidade. Um exemplo típico envolve os recursos de análise léxica, gramatical e ortográfica. As regras são muito diferentes para cada idioma e, enquanto são extremamente simples e consistentes para idiomas como o italiano, podem ser também extremamente difíceis para outros como o japonês e até o português. Um outro exemplo interessante envolve componentes do painel de controle do Windows XP. Enquanto opções relacionadas são mostradas para alguns idiomas, em outros não estão presentes. A figura a seguir mostra um exemplo claro de diferenças entre as configurações regionais existentes para o alemão e o italiano.

4. Unicode

Hoje em dia, softwares são utilizados nas mais variadas situações e por pessoas com diferentes perfis culturais. Seja em ambiente acadêmico ou empresarial muito provavelmente um determinado aplicativo será utilizado por pessoas que vêm de diversas partes e que podem necessitar utilizá-lo de acordo com suas convenções culturais e em seu idioma nativo. Neste caso, se o aplicativo não for capaz de coletar, armazenar, tratar e apresentar um nome, endereço ou qualquer outra informação em, por exemplo, chinês, bengali ou grego, existe uma grande chance que este software se torne um problema em curto espaço de tempo.

O número de idiomas falados é muito maior que o de idiomas escritos e diversos destes não utilizam nosso tradicional alfabeto de A a Z. Na verdade, muitos destes idiomas nem podem ser representados com alfabetos! Vários sistemas de escrita não são executados da esquerda para a direita e de cima para baixo, não utilizam espaços entre as palavras, não possuem nenhum tipo de ordem alfabética e não seguem os padrões ocidentais em diversos aspectos. Além disso, temos que considerar ainda os símbolos de moeda, simbologia matemática e científica.

Os primeiros pontos a serem considerados na solução deste problema são os métodos de entradas de dados e as fontes que serão utilizadas para apresentar todo este conjunto de opções para o usuário. Estes dois pontos serão abordados com detalhes mais a frente. A codificação de caracteres chamada Unicode foi criada para que os dados digitados em diversos sistemas de escritas fossem corretamente processados e os resultados corretamente apresentados de volta ao usuário, em seu idioma nativo.

4.1 Resumo histórico

Passado, presente e futuro do Unicode estão e sempre estarão intimamente ligados ao padrão ISO/IEC 10646. Como veremos adiante, os dois padrões foram unificados em 1991 e hoje em dia novos sistemas de escritas são incorporados aos dois simultaneamente.

Um conjunto universal de caracteres foi uma idéia necessária com a evolução da computação. Os esforços da entidade de padronização ISO e do consórcio de empresas chamado Unicode para alcançar este objetivo foram iniciados quase ao mesmo tempo (1985) e com objetivos similares. A percepção que duas diferentes padronizações, incompatíveis entre si, para o mesmo objetivo só traria prejuízos para a comunidade foi fruto da visibilidade avançada dos principais envolvidos no processo de padronização então, em 1991, as diferenças entre os padrões foram eliminadas.

4.2 Passado: a codificação tradicional de caracteres e sua variações

O mecanismo de correspondência entre caracteres e um determinado código tomou diversas formas antes do surgimento do Unicode.

No passado, a maioria dos computadores "falava" inglês, com cada um dos caracteres do idioma relacionados a um número em uma tabela de caracteres. Nesta representação, apenas 128 códigos eram necessários para o mapeamento de todos os caracteres utilizados, com pode ser visto a seguir.

Maiúsculas (A – Z)	26
Minúsculas (a – z)	26
Dígitos (0 – 9)	10
Sinais de pontuação (. , + { [ ) % $)	32
Espaço	01
Caracteres de controle (TAB, CR, LF, etc.)	33
Total	128

Como são necessários 7 bits para representar as 128 posições existentes (27=128), uma codificação de 7 bits foi criada e chamada de ASCII (acrônimo para American Standard Code for Information Exchange).

Conforme o uso de computadores foi se expandindo, novos idiomas precisaram ser acomodados, incluindo aí o suporte a acentuação. Assim, foi introduzido o suporte a novos sistemas de codificação como os SBCSs (Single-Byte Character Sets) e MBCSs (Multi-Byte Character Sets).

Percebe-se que o sistema ASCII não comporta a representação de outros idiomas diferentes do inglês, incluindo idiomas latinos que possuem acentuação. Então algumas empresas e organismos de padronização criaram novas páginas de código usando o byte inteiro (8 bits), dando assim origem ao SBCS. Nestas novas páginas de código o conjunto de caracteres codificados entre as posições 32 e 127 foi mantido para permitir a compatibilidade com o ASCII. Os caracteres entre as posições 128 e 255 foram chamados "caracteres estendidos" e variam de acordo com a página de código utilizada. Desta forma, páginas distintas foram criadas para os diversos idiomas latinos ocidentais, bem como para árabe, grego, turco e alguns outros idiomas não-latinos.

O SBCS possui limitações, entre elas o não tratamento de caracteres do leste asiático (por exemplo, japonês e coreano) devido ao grande número de caracteres existentes nestes idiomas.

Surgiu então o MBCS que contempla os idiomas não cobertos pelo SBCS. A escrita chinesa, por exemplo, possui mais de 10.000 ideogramas básicos, inviabilizando o uso do SBCS para sua codificação.

O MBCS utiliza 2 bytes (ou até 4 no Unix) para codificar os ideogramas e bytes simples para a codificação de caracteres normais, utilizados em conjunto com os ideogramas. O resultado é uma página de código que mistura caracteres single-byte e double-byte.

Porém, a introdução destes novos métodos de codificação ainda não foram suficientes para solucionar de forma prática os problemas relacionados a inclusão de novos idiomas, cada vez mais importante para que os aplicativos tivessem uma maior penetração nos diversos mercados. Surge então a idéia de criação de um sistema de codificação único.

4.3 Presente e futuro: Unicode

Os métodos complexos de programação, necessários para a manipulação de codificações mistas de caracteres, o processo de criação de novas páginas de código para cada novo idioma que necessite de suporte e a importância da troca e mistura de informações em uma variedade de idiomas distintos e entre plataformas de computação diferentes motivou a criação do padrão Unicode. Xerox e Apple tomaram a frente da criação do padrão, sendo logo seguidas por IBM e Microsoft, fundando assim o Consórcio Unicode (ver [UNICODE]), que atualmente inclui as maiores e mais importantes empresas de tecnologia da informação do mundo.

O Unicode abrange praticamente todos os caracteres utilizados em computação nos dias de hoje, sendo capaz de endereçar 1,1 milhões de caracteres. O padrão prevê codificações que utilizam 8, 16 e 32 bits, sendo a de 16 bits considerada como a codificação padrão e tendo seus 1,1 milhões de code points (i.e. a menor combinação de bits que pode representar uma unidade de texto codificado para efeitos de processamento ou troca [SN2002]) distribuídos em 17 planos, cada um destes endereçando mais de 65.000 caracteres. Os caracteres no plano 0 (zero), também conhecido como plano multilíngüe básico são utilizados para representar a maior parte dos idiomas escritos do globo, simbologia matemática e

científica, formas geométricas e marcas de pontuação. Porém, além de oferecer suporte aos caracteres mais comuns, o Unicode cobre também os ideogramas japoneses, chineses e coreanos (idiomas também conhecidos pelas iniciais de seus nomes em inglês – CJK – Chinese, Japanese and Korean), formas de representação do idioma árabe e até símbolos musicais. Muitos destes caracteres são mapeados além do plano 0 através do recurso chamado pares surrogate que será visto adiante.

.A maior parte dos caracteres são definidos de forma única, porém alguns podem ser combinados, formando caracteres completamente diferentes, tais como caracteres acentuados. Os caracteres acentuados mais comuns existem em sua forma combinada com code points próprios. Os mesmos caracteres podem ser formados pela combinação de um caracter base com um diacrítico (i.e. diferentes marcas que são agregadas aos caracteres básicos de um idioma [KM2000]), sem espaço. Por exemplo, ‘a’ seguido do diacrítico ‘`’ é apresentado como ‘à’. Este recurso de combinação permite uma grande variedade de caracteres acentuados sem que seja necessário um code point distinto para cada um deles. A figura a seguir apresenta alguns exemplos.

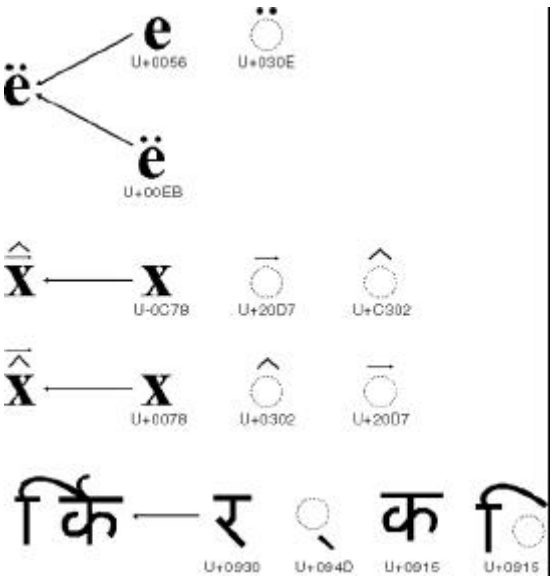


Figura 4.1.Caracteres combinados e seus code points

Transformações de code points

Existem diferentes métodos de representar cada um dos code points Unicode em formato binário. Cada um dos métodos a seguir se utiliza de um mapeamento diferente para representar caracteres únicos.

UTF-8: Criado para manter a compatibilidade com sistemas ASCII, SBCS e MBCS, o mapeamento UTF-8 foi definido pelo padrão Unicode. Cada caractere é representado por uma seqüência de 4 bytes, onde o primeiro destes bytes indica o número de bytes que se seguem em uma seqüência multi-byte de forma a facilitar o processamento da string. O UTF-8 é muito utilizado em transmissões via protocolos internet, bem como na criação de conteúdo para web.

UTF-16: Esta é a codificação de 16 bits da padronização, onde os caracteres recebem um valor de 16 bits único, exceção feita aos caracteres codificados via pares surrogate, estes últimos consistindo em um par de valores de 16 bits.

Portanto, cada caracter mapeado até a posição de número 65.535 é codificado por meio de um único valor de 16 bits e caracteres acima desta posição recebem dois valores de 16 bits. Aqui é utilizado o conceito de BOM (acrônimo para Byte-Order Marks). Processadores podem ordenar os bytes dentro de uma word (conjunto de 2 bytes) de forma que o byte menos significativo seja o mais a esquerda ou o mais a direita. Esta convenção utilizada pela máquina é chamada de Byte Order [PD&HJ1998]. Quando o byte menos significativo vem antes, é chamado de little-endian. Quando o byte mais significativo vem antes, é chamado de big-endian. O BOM pode ser utilizado como uma identificação da codificação de um arquivo texto, por exemplo. No caso no Windows Notepad, o BOM é adicionado ao início de cada arquivo, dependendo da codificação utilizada ao gravá-lo. Esta “assinatura” permite ao Notepad abrir o arquivo corretamente quando for necessário. Os sistemas operacionais da família Microsoft Windows utilizam o UTF-16 little-endian como codificação padrão. A forma de codificar apresentada neste trabalho (U+9662, por exemplo) segue a notação UTF-16.

UTF-32: Neste caso, cada caractere é representado como um inteiro de 32 bits.

A figura a seguir mostra dois caracteres codificados em três das formas vistas até aqui.

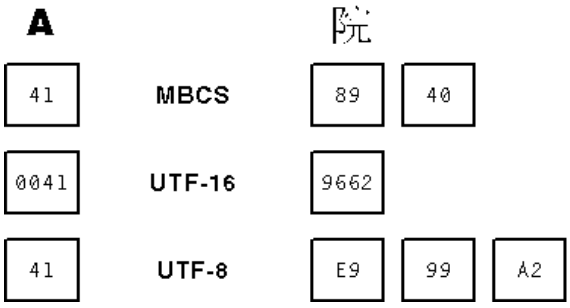


Figura 4.2. O caractere ‘A’ e um caractere Kanji codificado em MBCS, Unicode UTF-16 e UTF-8

Pares surrogate

Na codificação Unicode de 16 bits, mais de 65.000 caracteres podem ser incluídos (216=65.536). No entanto, o número de caracteres que necessitam de codificação nos dias de hoje ultrapassa estes limites, principalmente por conta da inclusão dos caracteres CJK. De forma a acomodar os novos caracteres, os desenvolvedores do padrão Unicode decidiram introduzir o conceito de pares surrogate. Neste caso, os code points Unicode entre U+D800 e U+DBFF (chamados surrogates altos) são combinados com outros code points na faixa entre U+DC00 e U+DFFF (chamados surrogates baixos) de forma a gerar novos caracteres, permitindo assim uma quantidade adicional de um milhão de caracteres novos. Diferentemente de caracteres MBCS, surrogates altos e baixos não podem ser interpretados quando apresentados fora de um par surrogate.

5. Aspectos gerais da internacionalização de software

Existem diversos aspectos que devem ser levados em consideração quando do planejamento e desenvolvimento de aplicações que serão utilizadas em outros mercados diferentes do de origem. Estes conceitos envolvem a forma de manipulação e representação da informação para o usuário, sendo importantes para o sucesso ou fracasso de uma aplicação em determinado mercado.

5.1 Localidade

A maneira mais eficiente de internacionalizar a entrada, apresentação e saída de dados de uma aplicação é utilizar o “modelo de localidade”, definido na linguagem C ISO. Uma localidade é o nome dado a um mercado internacional específico onde o usuário final está trabalhando [KM2000]. Estas regras e dados incluem informações sobre:
Classificação de caracteres;
Formatação de data e hora;
Convenções numéricas, monetárias, de pesos e medidas;

Regras de ordenação.

Para que seja fácil incorporar o “modelo de localidade” nas aplicações, os sistemas operacionais que suportam este conceito possuem um conjunto de APIs que gerenciam estas regras de forma automática, sendo apenas necessário que a localidade do usuário seja configurada e então estas APIs inicializarão o conjunto correto de regras e dados referentes àquele idioma e/ou área geográfica.
Existem diversas variáveis responsáveis pela configuração da localidade como um todo e que regem a forma como o sistema irá se comportar nas diferentes combinações de idiomas existentes e entre as mais importantes estão a localidade do sistema, localidade do usuário, localidade de entrada de dados, ID geográfica e idioma da interface com o usuário.
No Windows XP, por exemplo, a localidade do sistema é uma configuração única para cada sistema. Apenas o administrador possui direitos de acesso para alterá-la e o computador precisa ser reiniciado para que a nova configuração tenha efeito. O administrador apenas pode fazer a alteração caso o suporte a ela tenha sido previamente instalado, sendo a localidade uma variável de sistema não-programável.
Para exemplificar a vasta gama de opções de uso de localidades, seguem três cenários de uso do sistema operacional Windows:
Um usuário utilizando o Windows XP em alemão precisa executar uma aplicação em japonês que foi feita para o Windows 95. O usuário seleciona então o japonês como localidade. Neste caso, as aplicações em alemão não mais serão executadas de forma normal e caracteres como os "umlauts" não serão mais representados corretamente.
O mesmo usuário agora deseja digitar texto japonês em uma aplicação em alemão e deve obrigatoriamente selecionar a localidade “Japonês” para fazer isso com sucesso. Mais uma vez, esta operação acarretará problemas durante a digitação de texto em alemão na mesma aplicação.
Um usuário árabe deseja digitar textos em árabe, francês e inglês na mesma aplicação, desenvolvida para a plataforma Windows XP. Ele deverá então selecionar a localidade “Árabe”, pois esta contém todos os caracteres necessários para a correta representação do inglês e do francês.
A localidade do usuário determina as configurações padrão que o usuário deseja na formatação de datas, horários, moeda e numeração. Mesmo normalmente sendo apresentada como um idioma, não está relacionada ao idioma do sistema. Isto é, caso o usuário escolha o hebreu como localidade do usuário, ele realmente quer que o

sistema reconheça a padronização geral utilizada em Israel sem necessariamente utilizar o idioma hebreu. Para evitar confusões, alguns ambientes de desenvolvimento chamam a localidade do usuário de "informação cultural". Como o próprio nome indica, esta variável é configurada pelo próprio usuário, o que pode ser feito (no caso do Windows XP) a qualquer momento, sem que uma reinicialização do sistema seja necessária.

Exemplo:

Um usuário brasileiro morando na França e rodando a versão brasileira do Windows XP pode configurar seu sistema para a localidade do usuário 'Francês' de forma que ele utilize corretamente os padrões locais de data, horário e moeda.

A variável 'localidade de entrada de dados' determina o idioma no qual o usuário deseja entrar dados em uma aplicação e o método de entrada de dados. O sistema operacional pode ter várias localidades de entrada de dados instaladas e pode alternar entre elas quando desejar. A localidade de entrada de dados padrão é aquela que está ativa no momento que a aplicação é iniciada ou, em alguns casos, aquela que está selecionada quando uma nova janela é aberta. A alternância entre estas localidades é feita pelo mecanismo de threads, ou seja, podem haver duas localidades diferentes em uso ao mesmo tempo em duas aplicações diferentes, como por exemplo:

Um usuário decidiu que iria digitar um texto em italiano utilizando-se de um teclado com layout também italiano. Este mesmo usuário precisa também entrar dados em alemão com uma ferramenta de reconhecimento de voz. Ele poderá então ter todas estas localidades de entrada de dados instaladas e alternar entre elas sempre que necessário.

Uma secretária precisa entrar em um sistema de contabilidade os nomes das companhias internacionais em cada um dos idiomas nativos destas companhias. Isso é possível instalando diversas localidades de entradas de dados e alternando entre elas.

A figura a seguir mostra como é este processo no Windows XP.

Figura 5.1.Alternando entre localidades de entrada de dados

A variável chamada 'ID geográfica' é responsável por definir o país do usuário. As alterações neste parâmetro podem ser feitas a qualquer momento e não precisam de reinicialização do sistema para serem ativadas. Ao selecionar um loca, o usuário configura uma variável que, por exemplo, um serviço web pode detectar e enviar informação específica da área onde ele se encontra.

Exemplo:

Um usuário viajando pela Malásia gostaria de obter informações meteorológicas sobre aquela área.

A variável chamada 'idioma da interface' com o usuário possibilita ao usuário selecionar em qual idioma ele deseja que a interface seja apresentada. Ela afeta diretamente as caixas de diálogo, opções de menu e sistemas de ajuda. Em alguns sistemas operacionais, são necessários pacotes adicionais para que esta funcionalidade esteja disponível ao usuário.

Aqui deve ser feita uma distinção clara entre o idioma da interface com o usuário do sistema operacional e o idioma da interface com o usuário da aplicação. Mesmo sendo verdade que muitas vezes o idioma do sistema operacional é o mesmo que o usuário deseja utilizar, algumas vezes isso não reflete a realidade. O idioma do sistema operacional é aquele indicado pela versão que foi utilizada durante a instalação do mesmo e todas as caixas de diálogo, menus e sistemas de ajuda devem estar neste idioma. Porém o idioma da interface com o usuário pode ser definido a qualquer momento pelo próprio usuário, desde que o sistema o permita e tenha todos os componentes instalados para isso.

5.2 Escrita complexa

Uma escrita complexa é aquela que necessita de um tratamento especial para ser processada e apresentada, não podendo ser manipulada pelos métodos usuais disponibilizados pelo sistema operacional.

A figura a seguir mostra claramente as dificuldades envolvidas.

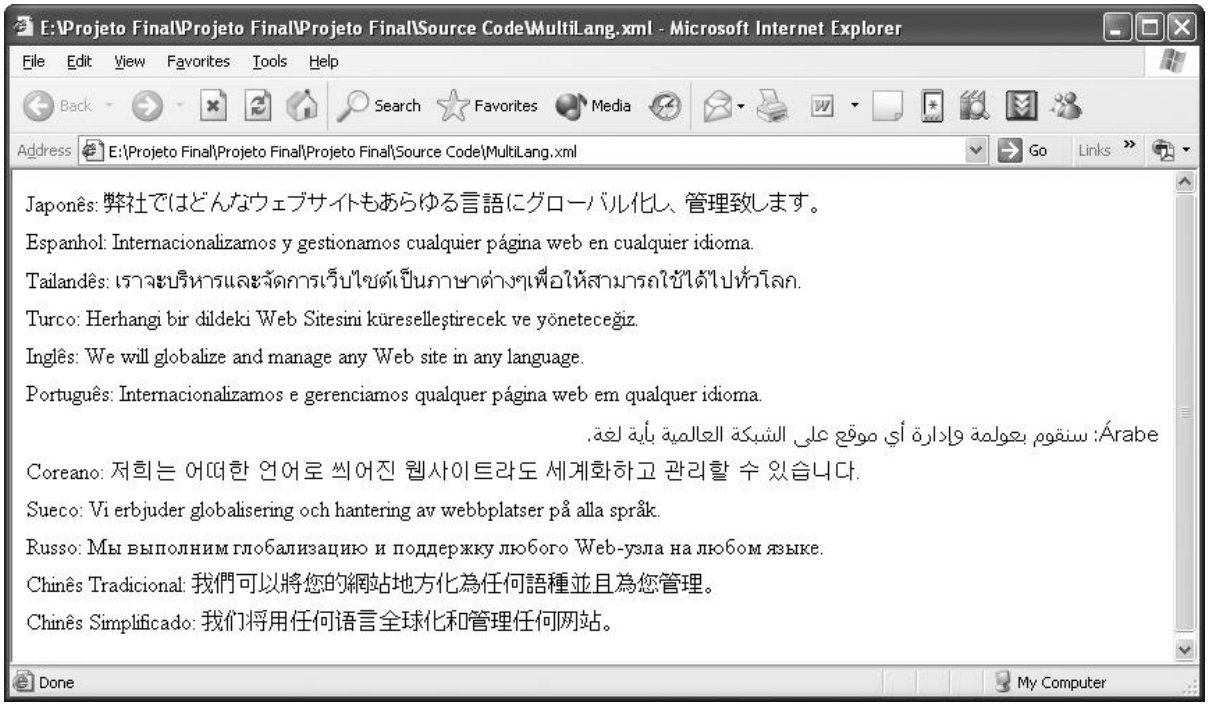


Figura 5.2.Frase em diversas escritas complexas [SY2001, alterado]

Este tratamento especial pode envolver uma ou mais das seguintes características:

- Reordenação de caracteres;
- Modelagem contextual;
- Apresentação de caracteres combinados e diacríticos;
- Regras especiais para separação de sílabas e justificação;
- Posicionamento do cursor;
- Filtragem de combinações ilegais de caracteres.

As escritas consideradas complexas são as seguintes:

- Família de idiomas árabes;
- Hebreu;
- Tailandês;
- Vietnamita;
- Coreano;
- Grego;
- Família de idiomas índicos.

Independentemente do suporte do sistema operacional às escritas complexas, alguns pontos, como os a seguir, devem ser levados em consideração ar tratar estes idiomas:

Ao apresentar os caracteres, não fazê-lo um de cada vez. Estes devem ser armazenados em um buffer e apresentados como string completa, valendo-se de recursos Unicode do sistema operacional.

Para alocar memória para os buffers não deve ser considerado que um glyph (i.e., uma figura que representa um caractere [KM2000]) é igual a um caractere. Deve-se utilizar o suporte Unicode do sistema operacional para alocação de memória para estes buffers.

Para calcular os tamanhos das linhas, não deve ser levado em consideração apenas os comprimentos dos caracteres. Devem existir funções Unicode no sistema operacional específicas para este fim.

5.3 Bi-direcionalidade e reordenação de caracteres

Bi-direcional (ou Bidi) é o termo utilizado para descrever um texto que possui uma escrita que flui da esquerda para direita (LTR – Left To Right) e da direita para esquerda (RTL – Right To Left).

A figura a seguir mostra um texto em árabe com inserções em inglês.

Hello سلام

Figura 5.3.Texto em árabe e inglês

Existem algumas questões que devem ser levadas em consideração para que uma aplicação seja considerada preparada para suportar idiomas bi-direcionais:

5.3.1 Armazenamento interno de dados

Como mencionado acima, um texto Bidi flui da esquerda para a direita e vice-versa. Mesmo fluindo de duas diferentes maneiras, o texto deverá ser armazenado na mesma ordem, do primeiro ao último caractere. A melhor forma de visualizar esta situação é pensar que o texto é armazenado em um buffer de forma ordenada, existindo então um mecanismo direcional de apresentação que será responsável por apresentar os caracteres na forma correta, RTL ou LTR.

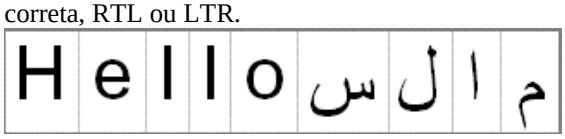


Figura 5.4.Armazenamento interno de dados no exemplo da figura 5.3

5.3.2 Fluxo de apresentação

No passado, a maior parte dos idiomas latinos era apresentada caractere a caractere. Hoje em dia, devido às diferentes propriedades dos textos bi-direcionais, referentes à posição dos caracteres, fluxo de escrita e a forma com a qual as ligações (características do idioma árabe) alteram seu formato dependendo do caractere que vem antes ou depois, é melhor guardar a linha que está sendo apresentada em um buffer e apresentar todo o buffer toda vez que um caractere for alterado naquela linha.

5.3.3 Tamanho da linha

Também devido às ligações entre os caracteres, não é uma prática recomendada somar os tamanhos dos caracteres independentes para chegar ao tamanho da linha. É melhor pegar o tamanho do texto no buffer.

O algoritmo bi-direcional Unicode define o layout final de textos em mais de uma direção na ausência de protocolos de mais alto nível designados para tal fim. A seguir seguem alguns exemplos de considerações feitas por este algoritmo para a tomada de decisões:

Números após palavras LTR devem ser mostrados à direita das palavras;

Números após palavras RTL devem ser mostrados à esquerda das palavras;

Pontuação entre palavras de mesma direção deve ser mostrada entre estas palavras;

Pontuações entre seqüências de palavras com direções opostas aparecem entre as seqüências;

Pontuação no início ou final de um parágrafo é tratada de acordo com a direção do parágrafo, não sendo afetada pela direção de textos adjacentes;

Pontuação no início ou final de um parágrafo é tratada de acordo com a direção do parágrafo, não sendo afetada pela direção de textos adjacentes;

Os dígitos de um número são tratados da esquerda para a direita dentro do número;

Vírgulas e pontos são considerados parte dos números quando imediatamente cercados por dígitos. Outros caracteres, tais como símbolos de moeda, são considerados parte do número quando imediatamente adjacente a um dígito.

Nas famílias de idiomas árabes e índicos, um glyph pode mudar drasticamente dependendo da posição do mesmo dentro da palavra e do caractere que precede ou vem logo após o glyph. No árabe, por exemplo, o mesmo caractere pode ter diferentes formatos, dependendo do contexto.

Na escrita latina, existe um mapeamento um-para-um entre caracteres e o seu glyph correspondente, por exemplo, o caractere ‘h’ sempre é representado pelo mesmo glyph ‘h’. Em escritas complexas, vários caracteres podem ser combinados, criando assim um glyph completamente novo e independente dos caracteres originais. Existem também casos onde o número de novos glyphs resultantes é maior que o número de caracteres utilizados para gerá-los. Caracteres são empilhados ou combinados com freqüência para gerar um novo grupo, indivisível na maioria das escritas complexas. Em árabe, no entanto, este grupo pode ser dividido quebrando o glyph nos caracteres e diacríticos que o compõe. Veja a figura a seguir para um exemplo de como isso acontece.

Figura 5.5 .(da esquerda para a direita): no idioma hindu, onde quatro caracteres individuais se juntam para formar um único glyph indivisível; no idioma tamil, onde dois caracteres individuais se juntam para formar um grupo indivisível de três glyphs; e árabe, onde dois caracteres individuais formam um grupo divisível composto por apenas um glyph.

Outra consideração importante é como tratar a justificação de texto. Para justificar um texto latino adicionamos espaços entre as palavras e/ou

caracteres. Porém não podemos usar este método para justificar um texto em árabe ou a formatação contextual mencionada acima será afetada. Portanto, neste caso, linhas contínuas (chamadas kashidas) são inseridas entre caracteres contíguos para que cada palavra fique um pouco mais longa. A figura a seguir mostra um exemplo de texto em árabe com a inserção de kashidas para justificação.

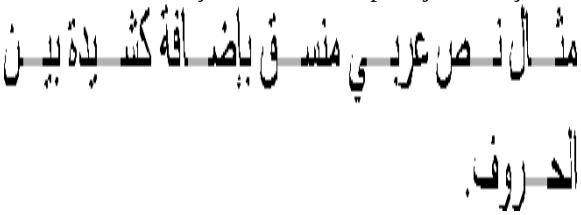


Figura 5.6.Uso de kashidas (em cinza) no idioma árabe

5.4 Diferenças de calendários

O calendário gregoriano é utilizado na maioria dos países de língua inglesa, porém ao desenvolver uma aplicação internacionalizada deve ser levado em consideração que existem outros calendários em uso nos dias de hoje, tais como os calendários japonês, budista, islâmico, hebreu e chinês.

As maiores diferenças entre os diversos calendários são:

Cada calendário pode estar em um ano diferente. O ano gregoriano de 2000 equivale ao 12º ano da era Heisei japonesa e ao ano 1421 do calendário islâmico;

O primeiro dia do ano pode não ser o primeiro de Janeiro. O Ano Novo chinês, por exemplo, foi comemorado no dia 5 de Fevereiro no ano 2000 gregoriano;

A duração dos meses e dos anos pode variar, bem como a forma de tratar anos bissextos;

Primeiro dia da semana pode não ser o Domingo. A maioria dos calendários em uso na Europa inicia a semana pela Segunda-feira.

Diferentemente do idioma inglês falado nos EUA, existem outras localidades que podem utilizar mais de um tipo de calendário. Nas localidades árabes da Nigéria, por exemplo, todos os calendários a seguir estão disponíveis:

Hijri ou calendário islâmico;

Gregoriano inglês no idioma nativo;

Gregoriano francês no idioma nativo;

Gregoriano árabe no idioma nativo;

Gregoriano no idioma inglês;

Gregoriano no idioma francês.

5.5 Capitalização

Um dos interessantes recursos embutidos no conjunto ASCII de caracteres é a fácil conversão entre maiúsculas e minúsculas. Estas podem ser criadas somando ou subtraindo 0x0020 ao seu ponto respectivo no código.

Ex.: A [0x0041] + 0x0020 = a [0x0061]

Logo, o problema de converter entre maiúsculas e minúsculas fica reduzido a um simples algoritmo de adição e subtração. Porém, este não é o caso para conversão de caracteres latinos acentuados.

Existem várias outras razões pelas quais um simples, ou mesmo complexo, algoritmo não cobre todas as ocorrências. Seguem alguns exemplos:

Alguns idiomas não possuem um mapeamento um-para-um entre seus caracteres maiúsculos e minúsculos;

Caracteres do francês europeu perdem os acentos quando são maiúsculas (é ⇒ E). Porém no francês canadense, isso não acontece (é ⇒ É);

A ligadura ‘ij’ é capitalizada como ‘IJ’ em holandês e como ‘Ij´ em croata [SY2001];

O equivalente maiúsculo do alemão ‘ß’ é ‘ss’ [SY2001];

A maior parte dos idiomas não-latinos não tem sequer o conceito de minúsculas e maiúsculas (ex.: chinês, japonês e tailandês).

5.6 Endereços

Um dos itens mais sem padronização de formatos a ser considerado na internacionalização é o formato de endereço. Assim, os campos de entrada e as rotinas que processam as informações relativas aos endereços devem ser capazes de entender e manipular os mais variados formatos.

Um dos erros mais comuns é insistir que o usuário entre com informações em um campo chamado ‘Estado’ (ou ‘Província’ para os canadenses). Enquanto esta informação faz sentido para as pessoas nos EUA ou Canadá, pode confundir usuários de outras partes do mundo que não possuem um ‘Estado’ em seus endereços. Deve haver também uma flexibilidade ao validar os dados inseridos. Por exemplo, é prudente evitar a validação de campos de CEP, pois eles variam enormemente de país para país e podem inclusive conter letras e não somente números.

5.7 Formatação de simbologia financeira

Para a formatação de moeda deve-se levar em consideração os seguintes elementos:

5.7.1 Símbolo da moeda

Pode ser um elemento pré-definido, como o Euro (€) ou uma combinação de símbolos, como o Marco alemão (DM), podendo ser posicionado antes ou depois do valor propriamente dito.

5.7.2 Valores negativos

Existem diversas formas de apresentar valores negativos, entre elas:

Sinal negativo antes do símbolo da moeda e do número:
Reino Unido: -£127.54
França: -127,54 F (antigo)
Sinal negativo antes do número, porém após o símbolo da moeda:
Dinamarca: kr-127,54
Sinal negativo após o número:
Países Baixos: 127,54 F-
Uso de parênteses:
EUA: ($127.54)

5.7.3 Separador decimal

A maioria das moedas utiliza-se dos mesmos separadores decimais e de milhares que a numeração local, porém isso não é sempre verdadeiro. Em alguns lugares da Suíça o ponto é utilizado como separador decimal para Francos Suíços (Sfr. 127.54), porém a vírgula é utilizada como separador decimal no restante do país (127,54).
Foram utilizadas nestes exemplos as moedas existentes na Europa antes da implantação do Euro (€) de forma a mostrar a grande variedade de possibilidades existentes.

5.8 Formatos de data

A formatação de data não é constante entre os diversos países do mundo. Apesar das datas apresentarem basicamente o dia, mês e ano, a sua ordem de apresentação e os separadores mudam bastante. De fato, podem existir muitas diferenças entre regiões de um mesmo país.
Existem duas formas básicas de apresentação de datas:
Data longa;
Data abreviada.

5.8.1 Data longa

A tabela a seguir ilustra as diferentes formas de apresentação de datas longas.

Idioma	Data longa
Inglês (EUA)	Tuesday, October 12, 1954
Espanhol (México)	Martes, 12 de octubre de 1954
Japonês	1954 年 10 月 12 日

Tabela 5.1.Diferentes formatos de datas longas

Obviamente os nomes dos meses e dias da semana são diferentes de local para local, porém no espanhol, o dia vem antes do mês, todas as letras estão em caixa baixa e o artigo “de” foi adicionado. Em Japonês, o dia da semana não é mostrado, e as traduções para dia, mês e ano atuam mais como separadores.

5.8.2 Data abreviada

Seguem alguns diferentes formatos de datas abreviadas:

Idioma	Data abreviada
Inglês (EUA)	10/12/54
Espanhol (México)	12/10/54
Japonês	54/10/12

Tabela 5.2.Diferentes formatos de datas abreviadas
Na data abreviada, observa-se que, no espanhol, a ordem é dia/mês/ano, diferente do inglês americano, que é mês/dia/ano. No Japão, a ordem é ano/mês/dia. Isto pode causar grandes problemas se não for tratado com cuidado. Por exemplo, dependendo do país que se está a data 07/04/01 pode significar:

- Inglês: 04 de Julho de 2001;
- Espanhol: 07 de Abril de 2001;
- Japonês: 01 de Abril de 2007.

5.9 Fontes

Existem dois aspectos a serem observados no tratamento de fontes.

5.9.1 Nomes de fontes explicitados diretamente no código

Com o uso do Unicode, existem milhares de caracteres diferentes para exibir, ao invés de centenas. A maioria das fontes não cobre todo o conjunto de caracteres Unicode. Assim se for inserido no código um nome de fonte que exiba caracteres latinos e não japoneses, todo o seu texto localizado para japonês será exibido com pontos de interrogação ou quadrados inseridos para ajuste.
Outra razão para não usar nomes de fonte no código é que a fonte desejada, pode não estar disponível no sistema no qual o texto está sendo exibido.

5.9.2 Tamanhos de fontes explícitos no código

Algumas escritas são mais complexas que outras. As mais complexas necessitam de mais pixels para serem exibidas corretamente. Por exemplo, a maioria dos caracteres latinos pode ser exibida numa grade de 5x7, porém caracteres japoneses precisam de no mínimo uma grade de 16x16 para serem vistos com clareza. O chinês precisa de uma grade de 24x24.

5.10 Métodos de entrada de dados

Os Editores de Métodos de Entrada de Dados (Input Method Editor - IME) são aplicações que possibilitam ao usuário inserir os milhares de caracteres utilizados nas linguagens escritas orientais, por meio de um teclado padrão de 101 teclas.

O usuário compõe cada caractere de uma das seguintes maneiras:

- Pelo radical;
- Pela representação fonética;
- Digitando o índice do código de página do caractere.

Um IME consiste em um código de programa que converte as teclas digitadas em caracteres fonéticos e ideográficos, e um dicionário de ideogramas utilizados freqüentemente. Enquanto o usuário digita, o IME tenta encontrar para qual ideograma ou caractere deve ser convertido os dados inseridos. Como alguns ideogramas têm a pronúncia idêntica, a primeira interpretação do IME nem sempre é a correta. Quando a sugestão é incorreta, o usuário pode escolher a partir de uma lista de homófonos (i.e. vocábulo que tem o mesmo som de outro com grafia e sentido diferente [ABHF1986]); o homófono escolhido passa a ser a primeira opção do IME daí em diante.

Não é necessário usar um teclado localizado para digitar caracteres ideográficos. Enquanto teclados localizados podem gerar sílabas fonéticas diretamente, o usuário pode representar sílabas fonéticas usando caracteres latinos.

5.10.1 Características dos idiomas orientais

Os sistemas de escrita chinês, coreano e japonês oferecem complexidades interessantes que não são encontradas nos sistemas de escrita latinos. Abaixo são descritas algumas características destes complexos mecanismos de escrita:

Chinês: existem duas formas de caracteres ideográficos normalmente utilizados no mundo de hoje: o chinês tradicional e o chinês simplificado. Os caracteres do chinês tradicional têm milhares de anos de existência e mantiveram suas formas originais. São utilizados em Taiwan e possuem mais traços que outros métodos ideográficos. O chinês simplificado, baseado no chinês tradicional, foi criado na China para tornar a leitura e escrita mais simples e acessível. Apesar de compartilhar alguns caracteres com o chinês tradicional, os caracteres mais simples, que são menos de 7000, são compostos por menos traços e na maioria dos casos são diferentes dos irmãos mais complexos. Exatamente por este motivo, aplicações desenvolvidas para o mercado chinês normalmente são lançadas em duas versões, uma no chinês tradicional e outra no simplificado.

Japonês: são chamados kanji e misturados com caracteres de outros silabários (i.e. conjunto de sinais componentes de uma escrita silábica [ABHLF1896]) chamados coletivamente de kana. As duas formas existentes do kana são identificadas como hiragana e katakana. O primeiro é uma escrita cursiva utilizada no texto japonês para representar inflexões finais de verbos e na escrita de palavras em japonês nativo que não possuem equivalentes em kanji, tais como ‘e’, ‘de’ e ‘para’. O segundo é utilizado na representação de palavras importadas de outros idiomas. Todos os símbolos kana, exceto os caracteres de vogal simples e o ‘n’ representam uma consoante seguida de uma das cinco vogais. O hiragana e o katakana conseguem juntos representar o conjunto completo de sons existentes no japonês. Outra forma de representação de caracteres existente no japonês é a tradução dos ideogramas para o alfabeto latino, chamada de romaji.

A escrita coreana utiliza-se de dois tipos de caracteres: hangul e hanja. Um caractere hangul é uma sílaba única formada pela combinação de uma ou mais consoantes, seguida de uma vogal. Existem 24 elementos básicos (14 consoantes e 10 vogais), também chamados fonemas, utilizados na simbologia; estes elementos são chamados jamos. Até 51 jamos podem ser criados com a combinação de dois ou mais elementos básicos para formar vogais e consoantes adicionais chamados “compostos”. Os compostos e os elementos básicos juntos totalizam 21 vogais (10 vogais básicas e 11 vogais compostas) e 30 consoantes (14 básicas e 16 compostas). Um caractere hangul (silábico) consiste em uma consoante inicial, uma vogal intermediária e, algumas vezes, uma consoante final. Dezenove das trinta consoantes podem ser iniciais, todas as 21 vogais podem ser intermediárias e 27 das 30 consoantes podem ser finais, portanto temos 11.172 combinações de caracteres possíveis, porém apenas uma pequena quantidade destas é realmente utilizada. O idioma coreano também adotou caracteres hanja do chinês e utiliza-os para a escrita mais formal e para representar nomes, porém maior parte da comunicação diária escrita é feita por meio do hangul.

5.11 Layouts de teclado, quebras de linha e hifenização

Layouts de teclado mudam de acordo com a localidade. Nem todos os caracteres existem em

todos os layouts de teclado. Quando se usam combinações de teclas para criar um atalho, deve-se ter certeza que essa combinação poderá ser reproduzida usando teclados em outros idiomas.
Já que cada localidade usa um teclado diferente, deve-se optar por usar caracteres de funções (F4, F5, etc.) e números ao invés de letras para combinações de teclas de atalhos. As combinações de teclas de função e/ou números não são tão intuitivas para os usuários como as combinações de letras, porém não precisam ser localizados. Criar combinações que podem ser produzidas com o teclado do local alvo pode ser cômodo para a maioria dos usuários, porém não para todos, já que em alguns locais como a Europa Central, Leste Europeu e países de língua árabe e hebraica possuem mais de um teclado padrão.
Algoritmos de quebra de linhas e separação de sílabas são importantes para a análise e exibição de um texto. Nos idiomas ocidentais, utilizamos quebras de linhas com regras de hifenização ou limites de palavras baseados em espaços em branco (espaços, tabulações, fim de linha, pontuações, etc.).
As regras para idiomas asiáticos, são totalmente diferentes das regras dos idiomas ocidentais. Por exemplo, diferentemente da maioria dos idiomas ocidentais escritos, o chinês, japonês e o coreano não indicam necessariamente a distinção entre palavras utilizando espaços. Para estes idiomas, aplicações internacionalizadas não podem basear algoritmos de quebra de linha em caracteres de espaço, ou em regras padrão de hifenização. Precisam seguir outras regras.
Vamos pegar como exemplo o japonês. A quebra de linha neste idioma é baseada na regra kinsoku: podem-se quebrar linhas entre qualquer dois caracteres, respeitando as seguintes exceções:

1ª Exceção: Uma linha de texto não pode terminar com nenhum caractere do tipo abre aspas, abre parêntese, e símbolos de moeda, que não podem ser separados dos caracteres que o sucedem.
2ª Exceção: Uma linha de texto não pode começar com nenhum caractere do tipo fecha aspas, fecha parêntese e pontuações, que não podem ser separados dos caracteres que o precedem.
3ª Exceção: Para certos caracteres de pontuação é permitido se estender além da margem direita (texto horizontal) ou da margem inferior (texto vertical).

5.12 Dados neutros em relação à localidade

Como aplicações internacionalizadas nunca podem assumir em qual idioma suas informações serão exibidas para os usuários, todos os dados internos devem ser armazenados em um formato binário ou string pré-definida e livres de qualquer localização (sem referências a idioma ou área geográfica). Desta forma, os dados podem ser compartilhados, sem necessidade de conversão, entre usuários rodando suas aplicações com diferentes configurações de localidade.
Uma técnica interessante é armazenar datas como seqüências de números, assim como horas e minutos como frações, pois o tempo é considerado uma porção do dia. Além de ajudar na aritmética com data e hora, também possibilita mostrar a data e a hora em qualquer formato que o usuário escolha.

5.13 Espelhamento

Para idiomas nos quais a ordem de escrita é da direita para a esquerda, não são apenas o alinhamento e a ordem de leitura do texto que são alterados, mas também todos os elementos de layout da interface do usuário devem seguir a nova direção. Isso pode causar vários problemas em barras de título, exibição de árvores, combo-boxes, etc. Segue um exemplo de uma área de trabalho espelhada:

Figura 5.7.Área de trabalho em árabe no Windows XP

Espelhar de fato, nada mais é que transformar coordenadas, como apresentado a seguir:

- A origem (0,0) é no canto superior direito da janela e não no superior esquerdo;
- Fator de escala de x é -1 (ou seja, valores são incrementados da direita para esquerda).

Uma questão interessante do espelhamento diz respeito ao tratamento a ser dado às imagens. Imagens sensíveis à direção apresentam novos desafios ligados ao espelhamento, pois algumas imagens possuem significado diferente quando espelhadas. Quando estas setas são espelhadas em um layout RTL, o significado é exatamente o oposto, como mostrado na figura a seguir.

Figura 5.8.Exemplo de seta de direção que sofre uma alteração de contexto quando espelhada

Outros gráficos, no entanto, não devem ser espelhados de forma nenhuma, pois representam marcas registradas ou logotipos e perdem o sentido

quando espelhados incorretamente, como pode ser visto na figura a seguir.

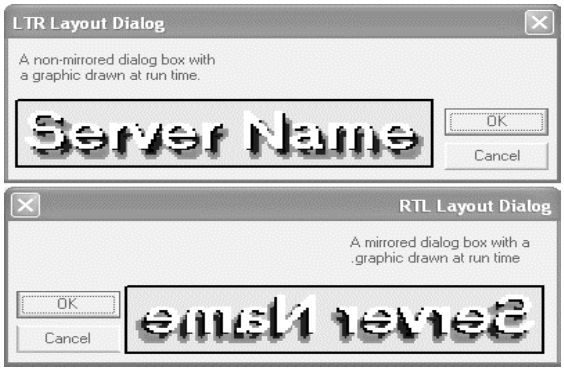


Figura 5.9.Exemplo de imagem que não deve ser espelhada

5.14 Formatação numérica

Ao tratar dos valores numéricos, deve-se dar atenção aos seguintes itens:

5.14.1 O caractere usado como separador dos milhares

Nos EUA, o separador dos milhares é a vírgula, enquanto que no Brasil o caractere utilizado é o ponto. Assim "mil e vinte e cinco" é indicado como 1,025 nos EUA e 1.025 no Brasil.

5.14.2 O caractere usado como separador decimal

Nos EUA, o caractere utilizado é o ponto. No Brasil o caractere é a vírgula. Assim, "mil e vinte e cinco e sete décimos" são indicados como 1,024.7 nos EUA e 1.025,7 no Brasil.

5.14.3 A forma como os números negativos são identificados

Além do uso do caractere '-' no início do número, ele também pode vir após o número. O número pode também ser indicado entre parênteses ou ser indicado em cor vermelha. Assim, existem quatro formas diferentes de indicar o número "menos quinhentos e vinte e sete":

- 527
527 -
(527)
527 (em vermelho)

5.14.4 Os números podem ser apresentados de forma diferenciada ou não ter uma correspondência direta

Os números também podem ser representados de forma diferente, conforme o idioma:

Idioma	Algarismos
Português	0 1 2 3 4 5 6 7 8 9
Árabe	٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩
Japonês	○ 一 二 三 四 五 六 七 八 九 十

Tabela 5.3 Algarismos em diferentes idiomas

5.14.5 Agrupamentos de dígitos

Tamanhos diferenciados para cada grupo de dígitos à esquerda do decimal. Por exemplo: EUA 123,456,789 e Hindu 12,34,56,789

5.14.6 O posicionamento do símbolo '%' (percentual)

Pode estar posicionado de diversas formas, portanto, não é uma boa medida fixar na programação o posicionamento do símbolo '%'.

98%
98 %
98 pct
%98

5.15 Tamanho do papel

Os tamanhos de papel dos EUA e Canadá (Carta, Ofício, etc.) não satisfazem às necessidades de todos os usuários do mercado mundial. Por exemplo, a maioria dos países da Europa e Ásia usa um padrão maior como o A4 (297 x 210 milímetros) que é ligeiramente mais longo e estreito que o tamanho do papel utilizado nos EUA (279 x 216 milímetros).
Assim se a aplicação necessita utilizar recursos de impressão, deve ser permitido que o tamanho do papel a ser utilizado seja configurado pelo usuário.

5.16 Ordenação e comparação de cadeias de caracteres

Pode-se pensar que todas as formas de ordenação existentes são conhecidas e que isso é um dos assuntos mais fáceis quando da internacionalização de uma aplicação, porém os usuários finais podem

ter expectativas diferenciadas do que constitui uma lista 'ordenada'. Não somente a ordem alfabética varia entre idiomas, mas também as convenções para ordenação de artigos nos dicionários e nos catálogos telefônicos podem ser completamente diferentes.
Na língua francesa, por exemplo, as palavras que são soletradas de forma idêntica (à exceção dos diacríticos) são classificadas baseando-se em uma comparação da direita para esquerda dos caracteres ao invés de uma comparação da esquerda para direita, como utilizado na língua inglesa.
Os idiomas asiáticos possuem ordenações diversas, dependendo da fonética, da ordem radical, etc.
Seguem abaixo alguns exemplos de ordenação de caracteres [SY2001]:
No norueguês e no dinamarquês 'æ' vem após 'z', 'ø' após 'æ', e 'å' após 'ø'.
No sueco e no finlandês 'ü' é equivalente a 'y', 'w' é equivalente a 'v', 'å' vem após o 'z', 'ä' após 'å' e 'ö' após 'ä'.
No alemão 'ae' é equivalente a 'ä', que vem após 'a', 'oe' é equivalente a 'ö' e vem após 'o', 'ue' é equivalente a 'ü' que vem após o 'u' e 'â' é equivalente a 'ss'.

5.17 Números de telefone

Os números de telefone, juntamente com o endereço, são um dos formatos menos padronizados e que necessitam ser tratados cuidadosamente na internacionalização de um aplicativo. Eles podem ter vários formatos:

País	Formato de telefone
China	1234 5678
França	01-23-45-67-89
Polônia	(12) 345,67,89
Singapura	123 4567
Tailândia	(01)234-5678 ou (012) 34-5678
Reino Unido	0123 456 7890 ou 01234 567890
EUA	(123) 456 7890

Tabela 5.4.Diversos formatos de números de telefone

Observa-se que há vários grupos diferentes de separadores (hífen, ponto, espaços), diferentes formas de agrupamentos (dois e três), e número de dígitos (7 a 11). Nota-se também que os exemplos acima não incluem códigos de país que podem ser quaisquer números de um a três dígitos.
O padrão ISO para o número de dígitos em um número de telefone é 14. Porém isso não inclui o espaço utilizado nos seguintes itens:

- Números da saída do PBX;
- Códigos de acesso interurbano;
- Senhas;
- Números de cartão do crédito.

Assim, não se deve pré-definir um formato de número de telefone ao projetar uma aplicação voltada para mercados internacionais.

5.18 Formatos de hora

Há dois pontos principais que devem ser considerados em relação à apresentação das horas:

Uso de 12 ou 24 horas – Na Europa e na Ásia, utiliza-se o formato de 24 horas ao invés do modelo de 12 horas AM/PM, utilizado nos EUA;
Caractere separador de horas, minutos e segundos - Embora os dois pontos seja o caractere mais usado para separar as horas, os minutos e os segundos, alguns idiomas asiáticos valem-se de ideogramas. Em alguns locais é necessário utilizar os caracteres 'h', 'm' e 's' como parte do indicador.
Uma forma freqüente de indicar o fuso-horário é a utilização do GMT (Greenwich Mean Time - Hora Média de Greenwich) ou seu UTC (Universal Time Coordinator - Coordenador de Hora Universal) como a base seguida do fuso-horário, indicado como um positivo ou offset do negativo nas horas e nos minutos (alguns fusos possuem 30 ou 45 minutos de offset).
Outra forma de indicar fusos-horários é utilizando os nomes do fuso-horário local, porém alguns pontos devem ser levados em consideração:

- Nem todos os países usam nomes locais;
- As abreviaturas do fuso-horário não são únicas;
- Nem todos os países valem-se do "Horário de Verão" que, em alguns casos não começam nem terminam no mesmo dia em cada país;
- Um fuso-horário pode ter muitos nomes diferentes dependendo do país e do idioma.

5.19 Unidades de medida

Em todo o mundo medidas são feitas utilizando unidades e escalas diferentes. O sistema de medida mais popular é o Sistema Métrico (litros, gramas, etc.). Nos EUA é utilizado o Sistema Imperial (pés, polegadas, libras, etc.).
Os vários tipos de unidades de medida são utilizados para:
Comprimentos;

- Pesos;
- Área;
- Volume;
- Temperaturas;
- Tamanhos de papel;
- Notação de ângulos, etc.

Assim, uma aplicação internacionalizada deve tratar corretamente todos os sistemas de medida.

6. Conclusão

O presente trabalho apresentou um resumo bibliográfico dos conceitos técnicos e práticas gerenciais necessárias para um processo de internacionalização rápido, eficiente e de custo controlado.

O trabalho passa por detalhes específicos de diversos idiomas considerados exóticos e tenta motivar os atuais e futuros analistas de sistemas e programadores a pensar em como criar um produto de forma que ele possa facilmente ser adaptado para outros idiomas sem que o usuário final seja o maior prejudicado, tanto no nível de funcionalidades básicas como no nível de entendimento e usabilidade geral.

É importante ressaltar que empresas investem muito dinheiro para penetração nestes novos mercados, principalmente aqueles com alta possibilidade de retorno financeiro tanto na quantidade de usuários (China e Brasil, por exemplo) quanto no poder aquisitivo dos mesmos, como é o caso dos mercados europeus e o mercado japonês.

Considerando a relevância do assunto aqui apresentado e as metas de exportação de software do governo federal, recomenda-se que o assunto seja incluído em cursos técnicos de informática e no Bacharelado em Informática e Tecnologia da Informação de forma que os futuros profissionais estejam habituados com as técnicas aqui apresentadas e possam chegar mais capacitados ao mercado de trabalho.

Referências

[SY2001] Savourel, Yves. XML Internationalization and Localization. Sams Publishing, 2001.

[KM2000] Kaplan, Michael S. Internationalization with Visual Basic. Sams Publishing, 2000.

[GT2000] Graham, Tony. Unicode, A Primer. M&T Books, 2000.

[SN2002] Symmonds, Nick. Internationalization and Localization Using Microsoft .NET. A!Press, 2002.

[PD&HJ1998] Patterson, David A. & Hennessy, John L. Computer Organization and Design: The Hardware and Software Interface. Morgan Kaufmann Publishers. Inc. 1998, 2nd Edition.

[ABHF1986] Ferreira, Aurélio B. de Holanda. Novo Dicionário da Língua Portuguesa. Editora Nova Fronteira 1986, 2ª. Edição.

[UNICODE] Unicode Home Page. Disponível [Online] em http://www.unicode.org/. 10 de Janeiro de 2005.

[LISA] Localization Industry Standards Association. Disponível [Online] em http://www.lisa.org/. 10 de Janeiro de 2005.

[GAZONL2004] Empresas de software criam holding para exportar para os EUA. Disponível [Online] em http://gazetaonline.globo.com/minutoaminuto/minutoaminuto_materia.php?id=041d1b35cb8328. 10 de Janeiro de 2005.