Nations Unies ECE/CES/2012/8

Conseil économique et social

Distr. générale
26 mars 2012
Français
Original: anglais

Commission économique pour l'Europe

Conférence des statisticiens européens

Soixantième réunion plénière
Paris, 6-8 juin 2012
Point 8 de l'ordre du jour provisoire
Groupe de haut niveau sur l'évolution stratégique de l'architecture d'entreprise dans le domaine de la statistique

Principes et directives concernant la création d'applications multilingues pour les statistiques officielles

Note du secrétariat

Résumé

La présente note expose les principes et directives concernant la création d'applications multilingues pour les statistiques officielles élaborés par le Conseil consultatif pour le partage des systèmes d'information statistique.

Des organismes nationaux de statistique ont demandé au Conseil consultatif des informations sur les normes ayant trait au développement de logiciels multilingues. De telles normes n'existent pas pour les applications statistiques, mais l'absence d'un support multilingue est l'un des principaux obstacles à la collaboration dans ce domaine. Le Conseil consultatif a donc élaboré l'ensemble des principes et directives reproduits dans la note.

Établis à l'intention du personnel et des sous-traitants qui développent des applications statistiques, ces principes et directives s'inspirent dans la mesure du possible des normes de plus vaste portée visant les technologies de l'information, mais s'appliquent principalement aux problèmes rencontrés par les organismes de statistique. Ils ont été analysés en profondeur par les spécialistes de l'application des technologies de l'information dans le domaine de la statistique. Leur but est de donner des conseils pratiques sur la façon de rendre autonome dès le début le support linguistique des applications, ce qui est manifestement plus efficace que d'essayer d'ajouter ultérieurement des capacités multilingues.

Les principes et directives sont destinés à être diffusés par voie électronique uniquement et sont censés être de nature évolutive. En raison de la constante évolution des technologies de l'information, ils seront périodiquement actualisés pour prendre en compte les nouvelles normes et technologies. Le Groupe de haut niveau sur l'architecture d'entreprise dans le domaine de la statistique a approuvé les directives et a recommandé qu'elles soient mises en œuvre par tous les organismes de statistique. Pour faciliter ce processus, les principes et directives sont présentés à la Conférence pour information.

1. Les présents principes et directives ont été établis par le Conseil consultatif pour le partage des systèmes d'information statistique de la Conférence des statisticiens européens et par le secrétariat de la Commission économique pour l'Europe (CEE) et sont également fondés sur les contributions des participants à la réunion conjointe CEE/Eurostat/Organisation de coopération et de développement économiques portant sur la gestion des systèmes d'information statistique (MSIS), qui s'est tenue en 2011. La présente version a été publiée en février 2012.

I. Introduction

2. La transition vers une normalisation mondiale des modèles tels que le modèle générique du processus de production statistique (GSBPM) et le modèle générique d'informations statistiques (GSIM), conjuguée aux avancées réalisées dans le développement de normes pour l'échange de données et de métadonnées, a attiré l'attention des fournisseurs d'applications statistiques sur la possibilité d'échanger des logiciels au plan international. La question s'est donc posée de savoir comment incorporer cette possibilité dès le début du processus de développement des logiciels.

3. Les applications statistiques sont souvent conçues dans un cadre national, le support multilingue n'entrant éventuellement en ligne de compte qu'après coup. Il est souvent bien plus difficile d'ajouter ultérieurement ce support si la langue n'est pas perçue dès le départ comme une partie essentielle de l'architecture logicielle.

4. Il est plus intéressant de créer des logiciels qui seront utilisés dans de nombreux pays que de se concentrer uniquement sur les questions de langue. Adopter les pratiques optimales dans l'internationalisation des logiciels équivaut à accroître les possibilités de partager des données au sein de la communauté des statisticiens. La mise en application des directives renforcera la portabilité et la réutilisation des logiciels.

5. Les présentes directives ont pour objet de permettre l'examen de quelques pratiques optimales en matière d'internationalisation des logiciels, de mettre en évidence certaines des ressources et des normes communes dans le domaine de la statistique, et de cibler les sujets qui pourraient s'appliquer spécifiquement aux logiciels de statistique, par exemple le traitement des nombres, du formatage, des dates, des formules et des notations.

II. Définitions

A. Internationalisation et localisation

6. Le développement de logiciels destinés à être utilisés dans un grand nombre de langues, de cultures et de pays peut être perçu comme deux processus distincts, l'internationalisation (I18N) et la localisation (L10N). L'internationalisation consiste à concevoir un logiciel qui pourra s'adapter à des environnements changeants sans avoir à en modifier le code. La localisation porte sur le cas plus particulier de la conception, par exemple pour un pays, une langue ou une région. Ainsi, bien que l'internationalisation concerne la fourniture d'un système qui permette d'utiliser plusieurs langues, la localisation vise plutôt l'application de ce concept, par exemple la traduction, etc.

B. Paramètres régionaux

7. Les paramètres régionaux sont un ensemble de préférences des utilisateurs applicable à une langue, à un pays ou à une culture. Leurs identificateurs sont généralement

une langue, souvent associée à un pays et parfois associée à un autre paramètre indiquant le code et le modificateur; ainsi, en_GB est le paramètre régional identifié pour l'anglais utilisé au Royaume-Uni. Autrement dit, il est possible de différencier l'anglais des États-Unis et l'anglais du Royaume-Uni. Les paramètres régionaux comportent un certain nombre d'éléments comprenant par exemple le nom et l'identificateur ISO de la langue, la monnaie, les critères de tri, les préférences numériques telles que les séparateurs des milliers, les calendriers à utiliser et d'autres éléments comme le sens du texte (de gauche à droite ou de droite à gauche, horizontal ou vertical), etc.[1].

III. Principes

A. Principe 1
Le logiciel devrait être de conception multilingue

8. Le logiciel devrait de préférence être conçu et testé sous la forme d'une application multilingue au lieu d'être ultérieurement mis à niveau. Il est recommandé d'inclure dans le cahier des charges initial l'obligation de faciliter l'utilisation de plusieurs langues. Il est plus facile de mettre en œuvre l'internationalisation si elle est incorporée tout au début. Lorsque cette caractéristique est incluse dans les spécifications, elle est mise au point et testée pendant tout le processus de développement, ce qui donnera un produit de meilleure qualité que si les prescriptions en matière de langue sont ajoutées à la fin du processus.

9. Les spécifications devraient préciser s'il faudrait stocker, présenter et produire plusieurs langues en même temps, par exemple dans un pays bilingue, ce qui influe sur la façon dont les applications multilingues sont mises en œuvre.

B. Principe 2
Les applications multilingues devraient prendre en charge des langues supplémentaires sans avoir à être remaniées

10. Lorsqu'une application a été conçue pour permettre l'utilisation de plusieurs langues, elle devrait faciliter l'introduction de nouvelles langues sans avoir à être remaniée en profondeur. Dans la mesure du possible, le programme devrait être conçu de façon à pouvoir s'adapter à de nouvelles langues, ce qui devrait également être facilité par le stockage des éléments de l'interface utilisateur et des données d'application. Les directives indiquées dans le présent document mettront en lumière certaines des questions qu'il faut garder à l'esprit.

11. Il est recommandé de prévoir des systèmes multilingues disposant si possible d'un ensemble de binaires, afin d'assurer un code homogène et de réduire les coûts d'installation, de prise en charge, etc. Dans ce cas, il n'est nécessaire de tenir à jour qu'une seule version. Un problème se pose cependant, à savoir que les éléments d'affichage doivent être conçus de manière à accepter des langues et des longueurs variables, ce qui peut aussi accroître la taille de l'application.

12. Le cas échéant, cela réduira le délai nécessaire pour fournir des versions internationales et améliorera la qualité du produit dans son ensemble.

[1] http://www.jbase.com/new/support/41docs/jBASE%20internationalization.pdf.

C. Principe 3
L'interface utilisateur devrait toujours être séparée du code

13. Lorsque l'interface utilisateur est séparée du code, le reste de l'application est neutre quant à la langue, d'où un certain nombre d'avantages:

a) L'ajout de nouvelles langues ou une modification de l'interface utilisateur n'exige pas une nouvelle compilation ou une reprogrammation du code;

b) Il n'est pas nécessaire pour les traducteurs de connaître le code ou la logique du programme.

14. Comme exemple, on pourrait citer l'utilisation de chaînes de variables au lieu de chaînes de constantes dans le code. Les chaînes de variables sont ensuite chargées selon la langue ou les paramètres régionaux choisis.

D. Principe 4
Identifier et adapter tous les éléments de l'interface utilisateur

15. S'agissant de l'interface utilisateur, il faudrait garder à l'esprit qu'elle comprend aussi le texte qui y est utilisé. Des changements de langue peuvent également modifier la présentation de l'interface, par exemple les boutons, la taille de la fenêtre de l'application, etc.

16. Lorsqu'il définit l'interface, le concepteur doit prévoir les changements susceptibles d'être apportés à la longueur des textes des formulaires. La longueur des mots varie d'une langue à l'autre, ce qui influe tout particulièrement sur les étiquettes, car la variabilité de la longueur des textes dans des langues différentes est plus marquée dans des expressions ou textes courts. À mesure que le nombre de mots augmente dans les phrases, la différence s'équilibre sur les textes plus longs. Par exemple, l'expression anglaise «Click here» (Cliquez ici) (10 lettres) se traduit par «Klicken Sie hier» en allemand (16 lettres).

17. On trouvera ci-après un guide concernant l'espace requis pour un texte traduit, extrait du document «Understanding Application Development Guidelines» d'Oracle.

Nombre de caractères anglais	*Espace supplémentaire requis*
1 caractère	400 % ou 4 caractères
2-10 caractères	101-200 %
11-20 caractères	81-100 %
21-30 caractères	61-80 %
31-70 caractères	31-40 %
Plus de 70 caractères	30 %

18. Il doit être possible d'agrandir l'espace prévu pour les menus, les étiquettes et les dialogues. Une fois traduits, tous ces éléments doivent de nouveau être testés. Il est recommandé de ne pas surcharger le texte des formulaires et des demandes, ce qui permet un affichage efficace des éléments traduits. Dans la mesure du possible, un dimensionnement relatif d'éléments tels que les cases d'étiquettes, les conteneurs, etc., devrait être préféré à un dimensionnement fixe.

19. Les icônes et les graphiques devraient aussi être reconnus comme faisant partie de l'interface utilisateur; pour permettre la traduction de l'interface, il faudrait limiter le texte intégré dans les icônes et les graphiques, voire en éviter l'utilisation. Tout texte utilisé devrait être extrait de chaînes de variables.

20. Les icônes et les graphiques devraient être vérifiés et traduits de la même façon que la chaîne de texte pour déceler toute éventuelle erreur d'interprétation. L'interprétation des icônes et des graphiques diffère selon les cultures. Des doigts pointés peuvent par exemple être offensants dans certaines cultures. Les couleurs peuvent avoir de fortes connotations positives ou négatives dans des cultures différentes. Ces éléments devraient être utilisés avec tact.

21. Les touches d'accélération, les raccourcis de menus, etc., sont tous des éléments de l'interface utilisateur qui doivent aussi être adaptés. Les raccourcis devraient être adaptés pour des langues différentes, d'où la nécessité de mémoriser des touches de raccourci dans le cadre de l'interface utilisateur. Les touches de raccourci doivent aussi être accessibles à partir de claviers différents, ce qui exige une vérification de la disposition des claviers pour divers paramètres régionaux.

22. À un certain stade des tests, l'interface utilisateur tout entière devrait être passée en revue en ce qui concerne la traduction et doit être mise à la disposition de l'équipe de localisation.

E. Principe 5
Planification du stockage et gestion des éléments des interfaces utilisateur multilingues

23. Les éléments des interfaces utilisateur devraient être conservés dans un format accessible et détectable, par exemple dans le format d'un fichier de ressources, d'une table de base de données, etc. Il faudrait mémoriser des métadonnées concernant des éléments d'interface tels que les observations, le contexte, les identificateurs, etc., qui peuvent être utilisés pour communiquer avec les traducteurs, par exemple l'endroit où certains textes sont utilisés, le moment où ils s'affichent et ce qu'ils véhiculent, et pour simplifier les conversations entre traducteurs et développeurs à des fins de clarification.

24. Les identificateurs permettent de repérer les éléments susceptibles d'être traduits et d'améliorer leur réutilisation entre différentes versions ou applications. Cela permet de normaliser les traductions et d'en réduire le coût si le même texte peut être réutilisé. Avec le langage de balisage extensible (XML) par exemple, il est recommandé, d'après le jeu de balises d'internationalisation (ITS), de stocker le texte susceptible d'être traduit dans des éléments plutôt que dans des attributs pour autoriser l'utilisation d'identificateurs uniques, d'observations, etc.

25. Les chaînes d'éléments exigées pour des langues différentes peuvent varier en longueur et donc en taille; il faut veiller à ce que les éléments de stockage puissent être agrandis pour prendre en charge des chaînes dans des langues différentes[2].

26. L'ordre des messages peut changer d'une langue à une autre de sorte que les messages devraient être mémorisés sous la forme de phrases complètes au lieu d'être reconstruits à partir de mots clefs. Il faut aussi éviter d'utiliser des nombres ordinaux, car ils ne sont pas toujours faciles à traduire. Au lieu de «Le 1er enregistrement du tableau est», utiliser «Enregistrement 1:».

27. Un texte qui contient des données variables peut poser problème. Par exemple, la phrase «La table contient 10 000 *enregistrements*» se traduit en roumain par «Tabelul are 10.000 *de înregistrări*», la position de la variable changeant entre le début et la fin du texte. La phrase même pourrait être simplifiée de cette façon: 10 000 enregistrements

[2] Voir les directives 8 et 12 pour un examen plus approfondi de la sauvegarde des données multilingues.

→ 10 000 înregistrări. Il est également possible d'utiliser des paramètres substituables, par exemple: «La table contient {1} enregistrements», {1} étant remplacé par la valeur (10 000) pendant l'exécution.

28. Il faudrait dans tous les cas définir une langue par défaut, de sorte qu'au moins un message quelconque est affiché en l'absence de texte ou d'image. L'utilisateur devrait être informé que l'élément ne figure pas dans la langue demandée. La traduction manquante devrait être consignée dans le journal comme une erreur système.

F. Principe 6
Décider de la meilleure façon de choisir les paramètres régionaux

29. Les paramètres régionaux convenant le mieux à l'utilisateur devraient normalement être choisis tout au début. On peut utiliser, selon qu'il sera approprié, les paramètres régionaux système ou les paramètres régionaux par défaut de l'utilisateur dans le système d'exploitation, les caractères détectés dans un document ou une option par défaut standard.

30. Il faut ensuite ménager aux utilisateurs la possibilité de choisir et de modifier leurs paramètres régionaux. Ainsi, une langue par défaut peut être définie par le pays où se trouvent les adresses IP, mais la plupart des sites Web internationaux reconnaissent qu'un utilisateur peut préférer une langue différente et permettent de modifier la valeur par défaut.

31. Il convient de se rappeler les paramètres régionaux choisis et de continuer à les utiliser pendant la session. Lorsqu'un utilisateur peut être identifié, sauvegarder les paramètres régionaux préférés et utiliser les mêmes préférences à l'avenir. Pour un site Web par exemple, on peut utiliser des cookies pour définir par défaut la langue souhaitée dans les futures sessions.

32. Théoriquement, un utilisateur devrait pouvoir modifier les paramètres régionaux à toute étape de l'exécution de sorte qu'il n'est pas nécessaire de redémarrer l'application pour modifier ces paramètres. Une modification après le démarrage peut entraîner des charges supplémentaires pour le serveur (formulation de plus d'une requête) et cela devrait être pris en compte dans les spécifications.

33. S'il est permis de modifier ultérieurement les paramètres régionaux, les données existantes devraient être conservées ou, à défaut, une alerte de perte de données doit être émise. Il devrait être permis d'annuler la modification des paramètres régionaux avant de rafraîchir les données saisies. Le mieux serait que l'état de la page soit maintenu lors des mises à jour des paramètres régionaux; les options de choix de langue devraient utiliser les noms d'origine, l'anglais, le français ou leur abréviation ISO (en, fr, de, etc.). Ne pas utiliser de drapeaux nationaux pour identifier les langues.

34. Les utilisateurs devraient être en mesure de choisir leur propre méthode de saisie, par exemple la disposition du clavier, pour qu'ils puissent saisir des caractères à l'aide des combinaisons de touches auxquelles ils sont habitués.

G. Principe 7
La présentation des données devrait être conforme aux usages régionaux

35. La présentation des données varie selon les paramètres régionaux. Certains des domaines où l'on observe le plus souvent des différences sont indiqués ci-après. Les éventuelles différences devraient être connues et prises en compte lors de la conception des logiciels. Il faudrait veiller à ce que l'application ne s'appuie pas sur le mode de présentation. On peut citer par exemple le refus d'accepter les formats de code postal de

différents paramètres régionaux. Le format approprié défini par les paramètres régionaux devrait être appliqué. Lorsque plusieurs formats existent, les préférences des utilisateurs devraient être sauvegardées. Les ensembles de caractères requis devraient pouvoir être affichés.

1. Exemples de différences de paramètres régionaux dans les modes de présentation

- Nombres:
    - Différence entre les formats des chiffres, par exemple position des séparateurs décimaux, etc.;
    - Différence entre les formats des chiffres ordinaux, par exemple 1er, etc.;
    - Nombres à échelle longue ou courte (différence entre billion et trillion);
    - Nombre de chiffres significatifs qui devraient être affichés;
- Dates: utilisation de différents formats, de différents calendriers, de différents jours ouvrables;
- Unités de mesure:
    - Système impérial ou métrique;
    - Monnaies, par exemple leurs symboles et la place de ceux-ci;
- Saisie et présentation des textes: les textes peuvent s'afficher dans différents sens, par exemple de droite à gauche;
- Format de la date et de l'heure:
    - Calendrier grégorien, thaïlandais, etc.;
    - Fuseaux horaires;
    - Jours ouvrables, par exemple début de la semaine;
    - Formats, par exemple format des États-Unis: MM/JJ/AA;
- Format de papier:
    - Le format de papier des États-Unis diffère de celui de l'Europe;
    - Les impressions ne devraient pas être limitées à un format «standard»;
- Format des noms, titres utilisés, par exemple M., Mme; disposition des noms, à savoir prénom ou nom de famille d'abord;
- Prescriptions juridiques:
    - Déterminer si des marques de fabrique ou de commerce, des logos ou autres sont utilisés au niveau local et s'ils peuvent ou devraient être utilisés pour d'autres paramètres régionaux;
    - Se conformer aux réglementations;
- Format des adresses et des numéros de téléphone, question pertinente pour de nombreuses applications statistiques. Une attention particulière devrait être prêtée aux points suivants:
    - Possibilité de saisir des adresses dans les langues souhaitées;
    - Ordre des champs d'adresse, par exemple nom de la rue d'abord, etc.;
    - Existence de codes postaux et format de ceux-ci;
    - Noms de pays: les paramètres régionaux indiqueront les listes qui devraient être utilisées pour les pays et les régions.

H. Principe 8
Le traitement et la sauvegarde des données devraient être conformes aux usages régionaux

36. Le traitement des données devrait prendre en compte les éventuelles différences entre les paramètres régionaux. Il convient de noter que les fonctions ci-après qui peuvent être utilisées pour le traitement sont susceptibles de varier selon les paramètres régionaux.

Arrondissement

- Il existe des différences, par exemple:
    - Combien de chiffres significatifs devraient être utilisés (par exemple pour l'arrondissement des monnaies)?
    - Lorsqu'il n'y a pas de comparaison de données entre les paramètres régionaux, utiliser l'arrondissement défini par les paramètres régionaux pertinents;
    - Si des données sont comparées, l'arrondissement standard devrait être utilisé;
- Chaînes[3]:
    - Saisie et sauvegarde;
    - Tous les caractères peuvent-ils être saisis et sauvegardés?
    - L'utilisation de claviers différents est-elle facilitée?
    - Des caractères uniques pour des frappes distinctes qui peuvent ne pas être disponibles sur tous les claviers sont-ils utilisés?
- Comparaison de chaînes:
    - L'ordre de recherche et de tri utilisé devrait être celui qui correspond aux préférences de l'utilisateur;
    - Il faudrait utiliser, s'il y a lieu, des fonctions de tri de chaînes dépendant des paramètres régionaux pour comparer des textes dépendant de ces paramètres (par exemple lstrcomp). Les fonctions standard de comparaison de chaînes peuvent ne pas prendre en compte les ordres de tri localisés;
    - Les fonctions de recherche de chaînes et autres fonctions devraient s'effectuer par caractère et non par taille en octets à cause des tailles variables des caractères;
    - Lorsque le texte ne dépend pas de paramètres régionaux, utiliser des fonctions de chaîne indépendantes de ces paramètres;
- Manipulation de chaînes:
    - Les sauts de ligne, l'espacement, etc., varient. Ne pas recourir à un saut de ligne standard pour le traitement;
- Tri de chaînes:
    - Les tris diffèrent selon les cultures et les alphabets;
    - Il peut exister plusieurs ordres de tri. Voir, par exemple, l'ordre des annuaires téléphoniques allemands;

[3] Voir la section «Chaînes de caractères» dans les références pour obtenir des liens menant à une discussion plus détaillée de la question.

- L'ordre de tri des caractères Unicode ne concorde pas avec l'ordre de tri linguistique ou l'ordre de tri binaire prévu. Ne pas recourir à un ordre de tri implicite;
- Décider si les recherches de données en plusieurs langues devraient être autorisées, ce qui peut être nécessaire dans des pays multilingues comme la Suisse;
- Un ordre de recherche dominant devrait être défini;
- Comment les caractères accentués devraient-ils être traités dans les recherches?
- Connaître les éventuelles différences avec les ordres de tri des bases de données et les collationnements utilisés;

- Monnaies:
  - Stocker les montants avec les identificateurs de monnaies;
  - Plusieurs monnaies devraient-elles être stockées?
- Unités de mesure:
  - Stocker l'unité de mesure avec la mesure.

37. Il est très important de faire attention aux suppositions implicites concernant la façon dont les choses fonctionnent.

I. Principe 9
Réutilisation de normes telles que des bibliothèques de programmes aux fins de la localisation

38. Les paramètres régionaux standard devraient être utilisés pour définir les configurations personnalisées. Il est possible d'accéder à ce type de renseignement grâce à de nombreuses bibliothèques et ressources de développement de logiciels. Le répertoire de données de paramètres régionaux classiques (*Unicode Common Locale Data Repository*) est un exemple de bibliothèque très complète contenant des informations téléchargeables sur les paramètres régionaux (voir les références pour de plus amples détails).

39. La plupart des logiciels de développement ont des routines de localisation qui donnent accès aux informations relatives aux paramètres régionaux, par exemple: setlocale de la plate-forme Microsoft Visual studio; International Components for Unicode (ICU) pour le développement en java et C++.

J. Principe 10
Inclure une estimation du coût du support multilingue dans l'estimation du coût de développement

40. Il faudra plus de temps pour développer et tester des logiciels multilingues. Si la traduction est incluse dans le logiciel, il faut prévoir le temps de traduction dans le projet de développement, non seulement pour le texte mais aussi pour la vérification de l'interface utilisateur. Comme par le passé, vérifier que les chaînes de textes s'affichent correctement, qu'elles ne sont pas tronquées et que les icônes et graphiques sont acceptables.

K. Principe 11
Inclure la documentation dans les activités de localisation

41. La documentation destinée à l'utilisateur devrait être disponible dans la langue définie par les paramètres régionaux et devrait être considérée comme faisant partie de l'interface utilisateur. La traduction peut être incorporée dans les activités de localisation. Au minimum, les manuels utilisateur devraient être traduits, et les fichiers administrateur devraient être inclus dans la mesure du possible.

42. Par ailleurs, d'autres objets textuels comme les fichiers d'aide devraient être inclus dans les activités de localisation, ainsi que les messages système.

L. Principe 12
Comprendre la norme Unicode et l'utiliser quand c'est possible

43. En interne, les caractères sont sauvegardés et mémorisés dans les ordinateurs sous la forme de nombres binaires, le mappage entre ces nombres et les caractères correspondants étant connu sous le nom de «jeu de caractères». Le code ASCII est l'un des premiers jeux de caractères standard. À l'origine, les caractères étaient stockés dans 7 bits de mémoire et étaient censés représenter l'alphabet latin standard, les nombres ainsi que la ponctuation, les mappages étant au nombre de $2^7 = 128$.

44. Le caractère «a» pouvait ainsi être représenté comme suit.

Chiffre binaire	*Code ASCII*	*Caractère*
110 0001	97	a

45. Pour représenter le jeu de caractères anglais standard, des codes ont été définis pour un jeu compris entre 1 et 127. Les codes inférieurs à 32 étant utilisés pour les processus système, les codes ASCII compris entre 32 et 127 étaient réservés au jeu de caractères latin classique comprenant 94 caractères imprimables et un caractère espace.

46. L'introduction de 8 bits pour le stockage de caractères a permis de stocker jusqu'à 255 caractères différents, y compris les mappages initiaux, ce qui voulait dire que les codes compris entre 128 et 255 étaient «libres». Le jeu de caractères a donc pu être étendu pour inclure les caractères accentués ou d'autres mappages, le cas échéant. Les codes supérieurs à 128 ont alors été mis en œuvre de différentes façons dans divers pays. Cette liberté signifiait que le texte envoyé à l'aide du jeu de caractères ASCII pouvait être représenté différemment dans différents paramètres régionaux, le même code binaire étant attribué à plusieurs caractères. Des jeux de codes ont été mis au point pour définir et montrer la façon dont les valeurs ASCII supérieures à 128 étaient utilisées, par exemple ISO Latin 5 pour le turc. Un nombre de 8 bits et un nombre de 255 caractères étaient encore insuffisants pour certaines langues comme le chinois, le coréen, le japonais, etc.

47. Les normes Unicode et ISO 10646 (jeu universel de caractères) ont été élaborées en tant que normes internationales pour permettre une représentation et un traitement uniformes de tous les caractères. Chacun de ceux-ci est assorti d'un nombre défini ou «point de code», un code U+0021 désignant par exemple le point d'exclamation «!». Unicode peut prendre en charge plus d'un million de caractères, dont plus de 100 000 sont attribués dans la version actuelle. Les 128 premiers caractères sont compatibles avec le jeu de caractères ASCII.

48. L'introduction et l'adoption de la norme Unicode ont permis de séparer le codage et le stockage des caractères. Les points de code pourraient être mémorisés à l'aide de différents systèmes de codage.

49. Les systèmes de codage les plus courants sont l'UTF-8, l'UTF-16 et l'UTF-32. Pour déterminer le système à utiliser, il est primordial de prendre en compte les limitations imposées par la mémoire, le stockage et la nécessité d'une compatibilité en amont. On trouvera un bref aperçu des systèmes ainsi que des renseignements complémentaires sur les comparaisons entre eux sur le site de Wikipedia: http://en.wikipedia.org/wiki/Comparison_of_Unicode_encodings.

50. Dans l'UTF-8, chaque point de code compris entre 0 et 127 est stocké sur 1 octet. Les points de code d'une valeur de 128 ou plus sont stockés sur un nombre allant jusqu'à 6 octets. Autrement dit, pour les 128 premiers caractères, le jeu de caractères ASCII est mis en correspondance avec l'UTF-8. Le premier octet d'un caractère multioctet indique combien d'octets sont utilisés pour le caractère. C'est la valeur par défaut employée pour le langage XML.

51. L'UTF-16 est la norme relative aux fenêtres, soit 2 octets par caractère. La plupart des caractères Unicode sont codés par leurs points de code. L'UTF-16, qui fonctionne sous Java et Windows, est très utilisé dans des régions comme la Chine et le Japon qui appliquent le système DCBS (caractère à double octet).

52. L'UTF-32 a recours à 4 octets par caractère.

53. La norme Unicode permet aussi d'utiliser d'anciens systèmes de codage. Lorsqu'un code n'existe pas dans l'ancien système, une valeur manquante par défaut s'affiche, ce qui élimine le problème de permutation des caractères dans des systèmes différents. Toutefois, pour représenter un texte correctement, il faut au minimum spécifier le texte de codage.

1. Utilisation de la norme Unicode

54. Pour utiliser la norme Unicode dans les applications:

a) Utiliser des jeux de codes compatibles et les communiquer. Les données créées avec des jeux de codes différents devraient pouvoir être lues et traitées correctement. Par exemple:

i) Dans les courriels, le système de codage devrait être indiqué dans l'en-tête, par exemple **Content-Type: text/plain; charset="UTF-8"**;

ii) Avec les pages HTML, le codage devrait toujours être indiqué dans l'étiquette Meta de <head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />. Depuis décembre 2007, l'**UTF-8** a pris le pas sur l'ASCII et d'autres jeux de codes pour devenir le système de codage le plus utilisé sur les pages Web;

b) Utiliser des types de données et des fonctions compatibles avec Unicode dans toute la mesure possible.

2. Sauvegarde de données dans le format Unicode

55. Les données devraient être stockées dans des types de données Unicode; en langage SQL, par exemple, le serveur utilise nchar, nvarchar et nvarchar(max), au lieu de leurs équivalents non Unicode char, varchar et text. Ces types et fonctions de stockage peuvent accepter des caractères plus larges. Les données devront être sauvegardées ou converties dans un format Unicode selon qu'il sera approprié.

56. Il faut faire attention lorsque des données non stockées en format Unicode sont converties en types de données Unicode, car cela pourrait causer des problèmes avec les données manquantes. Une erreur devrait être signalée si ce phénomène est détecté.

57. Utiliser des fonctions compatibles avec Unicode si elles sont disponibles.

58. Il peut y avoir des coûts et des différences en matière de réalisation avec les fonctions de chaîne Unicode, mais cela ne devrait pas être prohibitif. Toutefois, une optimisation pourrait s'imposer en cas de traitement intensif des textes[4].

M. Principe 13
Considérer quelles polices de caractères utiliser et comment les utiliser

59. Il existe un ensemble limité de polices de caractères Unicode qui sont censées représenter la plupart, sinon la totalité du jeu de caractères Unicode, y compris la police Arial Unicode MS. En général, il est recommandé d'utiliser les polices les plus appropriées pour la langue et les paramètres régionaux plutôt que les polices Unicode. Théoriquement, il faudrait choisir une police qui prenne aussi en charge une large gamme de caractères accentués tout en satisfaisant aux critères de conception. Il convient de préférer des polices simples à une police de caractères manuscrits ou calligraphiques.

60. Pour certains choix de polices, il faut utiliser des noms logiques, par exemple sans serif, définir la police par son nom, recourir à des polices TrueType et regrouper les polices avec l'application.

61. Une police de substitution devrait être définie.

62. Les caractéristiques de formatage varient d'une langue à l'autre. Les caractères gras, par exemple, sont souvent utilisés pour mettre en évidence des mots dans de nombreux alphabets, mais en kanji et dans d'autres alphabets, ils peuvent causer des problèmes de lisibilité. D'autres techniques de mise en évidence comme le soulignement peuvent être utilisées à leur place.

IV. Conclusion

63. Il est devenu essentiel de fournir un logiciel qui puisse être utilisé dans plusieurs langues ou cultures. Au cours de la dernière décennie, l'essor d'Internet a favorisé le partage de logiciels. Pour tirer parti des futures avancées réalisées au sein de la communauté des statisticiens, il faudrait développer les logiciels en prévoyant qu'ils seront utilisés dans plusieurs environnements.

64. Ce document présente certaines des principales recommandations ayant trait à l'élaboration de logiciels susceptibles d'être adaptés à un usage multilingue. Il ne prétend pas donner une liste exhaustive des problèmes qui risquent de surgir, mais plutôt des éléments de base sur les problèmes souvent rencontrés et quelques informations sur la façon de les éviter ou d'y remédier.

65. Les directives ci-dessus ont été établies dans le cadre d'un examen de documents relatifs à des exemples pratiques d'internationalisation des logiciels et de la recherche. Il est admis qu'il n'est peut-être pas possible de mettre en œuvre chaque directive dans chaque

[4] http://support.microsoft.com/kb/322112 Comparaison des classements SQL sur les interclassements Windows. Les règles de comparaison concernant les données non Unicode et les données Unicode étant différentes, l'utilisation d'un classement SQL peut donner des résultats différents pour les comparaisons des mêmes caractères, selon le type de données.

application, mais les directives ont pour objet de donner un aperçu des meilleures pratiques vers lesquelles il faut tendre.

V. Ressources

A. Guides

IBM a également publié un guide très complet sur le développement de logiciels internationaux qui analyse en profondeur les problèmes que pose la mise au point de logiciels multiculturels et qui peut être consulté à l'adresse suivante: http://www-01.ibm.com/software/globalization/guidelines/.

Globalization Step-by-Step: http://msdn.microsoft.com/en-us/goglobal/bb688121.

Get World-Ready Microsoft: http://msdn.microsoft.com/en-us/goglobal/bb895995.aspx.

Wikipedia: http://en.wikipedia.org/wiki/Internationalization_and_localization.

jBASE Internationalization Publication detailing internationalisation efforts for Jbase software: http://www.jbase.com/new/support/41docs/jBASE%20internationalization.pdf.

Best practices for XML localisation: http://www.w3.org/TR/xml-i18n-bp/.

UTF-8 and Unicode FAQ for Unix/Linux: http://www.cl.cam.ac.uk/~mgk25/unicode.html.

The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets: An easy to understand guide to Unicode for programmers: http://www.joelonsoftware.com/articles/Unicode.html.

Description of basic concepts for internationalization, how to write internationalized software, and how to modify and internationalize software: http://www.debian.org/doc/manuals/intro-i18n/.

Key challenges in Software internationalisation: http://www.acs.org.au/documents/public/crpit/CRPITV32Hogan.pdf.

B. Chaînes de caractères

Microsoft Sorting and String Comparison: http://msdn.microsoft.com/en-us/goglobal/bb688122.

Guide d'Oracle: Linguistic Sorting and String Searching: http://download.oracle.com/docs/cd/B19306_01/server.102/b14225/ch5lingsort.htm#NLSPG005.

C. Listes de vérification

Microsoft Win32 Internationalization Checklist: http://msdn.microsoft.com/en-us/library/cc194756.aspx.

Liste de vérification d'Oracle pour la localisation des logiciels: http://developers.sun.com/dev/gadc/i18ntesting/checklists/allquestions/allquestions.html.

Guidelines, Checklists, and Resources: http://www.i18nguy.com/guidelines.html.

D. Bibliothèques et ressources logicielles communes

CLDR: Common Locale Data Repository (répertoire de données de paramètres régionaux classiques Unicode). Le CLDR Unicode offre un répertoire normalisé de données de paramètres régionaux en format xml: http://cldr.unicode.org/.

Bibliothèque GLIBC: bibliothèque standard C distribuée dans le cadre du projet GNU: http://www.gnu.org/software/libc/. Voir, par exemple, 7.6 Accessing Locale Information: http://www.gnu.org/s/libc/manual/html_node/Locale-Information.html.

Plate-forme Java: java.util.Locale: http://java.sun.com/developer/technicalArticles/J2SE/locale/.

Projet ICU: un ensemble éprouvé et largement utilisé de bibliothèques C/C++ et Java offrant une prise en charge du format Unicode et de l'internationalisation aux applications logicielles: http://userguide.icu-project.org/i18n.

E. Normes

Unicode: le consortium Unicode est une organisation à but non lucratif ayant pour mission de développer, de mettre à jour et de promouvoir des normes et des données relatives à l'internationalisation des logiciels, notamment la norme Unicode, qui définit la représentation des textes dans tous les produits logiciels et normes modernes connexes: http://www.unicode.org.

La norme Unicode actuellement publiée est la version 6, qui contient 109 000 caractères: http://www.unicode.org/versions/Unicode6.0.0/ch01.pdf.

La page d'accueil de l'Internationalization Core Working Group contient des conseils en matière d'internationalisation à l'intention d'autres groupes qui élaborent des normes pour le Web: http://www.w3.org/International/core/.

Article de Wikipedia sur le format ASCII: http://en.wikipedia.org/wiki/ASCII.

Article de Wikipedia sur le format UNICODE: http://en.wikipedia.org/wiki/Unicode.

F. Guides des organismes nationaux

BILINGUAL SOFTWARE GUIDELINES AND STANDARDS - Welsh Language Board: http://www.byig-wlb.org.uk/english/publications/publications/3963.pdf.

Building bilingual applications at StatCan (Karen Doherty, Statistics Canada): http://www1.unece.org/stat/platform/download/attachments/22478904/issue+4.pdf?version=1.

G. Exemples

Lessons Learned From Internationalizing - a Web Site Accessibility Evaluator: http://citeseerx.ist. psu.edu/viewdoc/download?doi=10.1.1.13.2698&rep=rep1&type=pdf.