



Zusammenfassung

Software-Lokalisierung

Ausgabe 1. März 2005

Copyright© 2005 DataDax EDV-Lösungen GmbH

Inhaltsverzeichnis

Kapitel 1	Software-Lokalisierung	1
1.1	Begriffsdefinitionen (was heißt eigentlich GILT?)	1
1.2	Warum lokalisieren? Rechtliche Gründe	1
1.3	Warum lokalisieren? Weitere Gründe	1
1.4	Geschichte der Software-Lokalisierung	2
1.5	Der Markt für Software-Lokalisierung	2
1.6	Wie wird Software lokalisiert?	2
1.6.1	Fallbeispiel (1): SAP Globalization Services	2
1.6.2	Fallbeispiel (2): KDE Internationalization	3
1.7	Umfang eines Lokalisierungsprojektes	3
Kapitel 2	Kulturell bedingte Inhalte	4
2.1	Bedeutung von Farben	4
2.2	Bedeutung von Symbolen	4
2.3	Eigennamen und Produktnamen	4
2.4	Datum und Uhrzeit	5
2.5	Zahlen und Währungen	5
2.6	Lokalisierung von Text	5
2.6.1	Unicode	5
2.7	Aneinanderreihung von Zeichenketten	6
2.8	Laufrihtung (LtR und RtL)	6
Kapitel 3	Konsequenzen für die Software-Entwicklung	7
3.1	Verwendung von Gebietsschemata	7
3.2	Techniken zur Lokalisierung von Benutzeroberflächen	7
3.3	Ein Wort zur Übersetzung	7

3.4 Regeln für GILT-fähige Software (Zusammenfassung) 7

Kapitel 1 Software-Lokalisierung

Zusammenfassung eines Vortrags im Rahmen der Vortragsreihe der Software-Ring Business Solutions eG am 1. März 2005, von Markus Schütz (markus.schuetz@cafesolo.biz).

Die Vortragsfolien finden Sie unter http://www.cafesolo.biz/de/slides/localization/_intro.html.

1.1 Begriffsdefinitionen (was heißt eigentlich GILT?)

Im Zusammenhang mit Software-Lokalisierung stößt man früher oder später auf folgende Begriffe:

- Globalization (G11N): Alle unternehmerischen Tätigkeiten, die für eine Geschäftstätigkeit im globalen Umfeld erforderlich sind. (Die englische Abkürzung ist von der Tatsache motiviert, dass zwischen dem ersten und dem letzten Buchstaben 9 Zeichen stehen.)
- Internationalization (I18N): Gestaltung eines Produktes in einer Weise, die eine leichte Anpassung auf verschiedene Zielmärkte zulässt. Dies kann bei bereits eingeführten Produkten ein kostspieliges Reengineering erforderlich machen.
- Localization (L10N): Anpassung eines Produktes auf einen Zielmarkt. (Bei einem Auto ist es beispielsweise erfolgversprechend, das Lenkrad auf der rechten Seite anzubringen, wenn man es in Großbritannien vermarkten will.)
- Translation: Bezeichnet die rein sprachliche Übersetzung von Texten im üblichen Sinne.

Zusammenfassend findet man übrigens auch den Begriff "GILT", der sich aus den Anfangsbuchstaben der Einzelbegriffe zusammensetzt.

1.2 Warum lokalisieren? Rechtliche Gründe

Software-Lokalisierung kostet Zeit und Geld. Warum investieren Hersteller dennoch in diesem Bereich?

- Produkthaftungsgesetze verlangen im Einzelfall, dass die Benutzeroberfläche sowie die Dokumentation einer Software in wenigstens einer Amtssprache verfügbar sind. Ungefähr 75% der Menschheit verstehen kein Englisch! Bei Bedienungsfehlern, die nachweislich aufgrund der Tatsache entstanden sind, dass die Software nur in einer Sprache vorlag, die von Anwender nicht verstanden werden konnte, entstehen leicht Regressansprüche.
- Für Software, die im Zusammenhang mit Maschinen oder Medizinprodukten eingesetzt wird, gelten in der EU und anderen Ländern noch strengere Bestimmungen (EU-Maschinenrichtlinie bzw. Medizingerätherichtlinie). Hier wird explizit verlangt, dass eine Benutzeroberfläche bzw. Dokumentation in der Landessprache vorliegen muss.
- Staatliche Behörden verlangen meist, dass an sie gelieferte Software in die Landessprache(n) übersetzt ist. Dies ist insbesondere bei öffentlichen Ausschreibungen von Belang.

1.3 Warum lokalisieren? Weitere Gründe

Neben den rein rechtlichen gibt es noch weitere Gründe für Software-Lokalisierung:

- Bessere Akzeptanz bei der Zielgruppe
- Verringerter Support-Aufwand: Je geringer die Hemmschwelle eines Anwenders ist, die Dokumentation bzw. Online-Hilfe zur Hilfestellung in Anspruch zu nehmen, umso geringer wird die Tendenz sein, bei der Hotline anzurufen. Wenn nicht noch zusätzlich eine Sprachbarriere überwunden werden muss, fällt dies umso leichter.

- Unterscheidungsmerkmal gegenüber der Konkurrenz: Gerade in Branchen, in denen viele weitgehend funktionsgleiche Produkte miteinander konkurrieren, kann eine gut gemachte Anpassung auf kulturelle und landestypische Gegebenheiten ein Unterscheidungsmerkmal darstellen, das letztendlich zur Kaufentscheidung beim Kunden führt.

Faustregel: Je "laienhafter" der Anwenderkreis einer Software ist, umso wichtiger ist eine Verfügbarkeit in der Landessprache!

1.4 Geschichte der Software-Lokalisierung

Die Geschichte der Software-Lokalisierung ist eng mit der Evolution der Mensch-Maschine-Interaktion verknüpft:

- Bis 1970: Dominanz der Mainframe-Systeme, kaum natürlichsprachliche Datenverarbeitung, Ein-/Ausgabe größtenteils noch mit Lockkarten
- Ab ca. 1980: Mittlere Datentechnik; kommandozeilenorientierte Betriebssysteme (UNIX, VMS); beschränkt auf Expertenkreis; gleichzeitig Standardisierung von ASCII (zuerst 7-Bit)
- Seit 1990: Siegeszug des PC und der grafischen Benutzeroberflächen sowie der OO-3G-Programmiersprachen (C++, Java)
- Seitdem: Verstärkte Verwendung von Software durch Fachleute, die nicht aus der IT-Branche kommen

1.5 Der Markt für Software-Lokalisierung

Der Markt für Software-Lokalisierung lässt sich durch folgende Zahlen kennzeichnen (aus dem Jahr 2003, Quelle: LISA, www.lisa.org):

- Die 20 größten Software-Hersteller weltweit geben jährlich ca. 1,5 Mrd. USD für die Lokalisierung ihrer Produkte aus (Industrie gesamt: 5 - 15 Mrd. USD)
- Der mit lokalisierter Software erzielte Umsatz für die Top 20 beläuft sich auf jährlich 15 Mrd. USD
- Diese Tendenz ist stark steigend
- Oft versuchen Hersteller, Lokalisierungskosten auf ihre Kunden abzuwälzen

1.6 Wie wird Software lokalisiert?

Folgende Szenarien bieten sich bei der Organisation einer Software-Lokalisierung an:

- Intern durch Fachabteilung
- Extern durch Landesgesellschaft oder Fachhandelspartner
- Extern durch Dienstleister

Aufgrund des Zeitdrucks und der Dezentralisierung von L10N-Projekten bestehen hohe Anforderungen an das Projektmanagement, weil Fachkräfte mit unterschiedlichen Kompetenzen (Programmierung, Architektur, Fachübersetzung) koordiniert werden müssen.

Ein verlässliches Terminologie-Management ist aus Konsistenzgründen unverzichtbar.

1.6.1 Fallbeispiel (1): SAP Globalization Services

Die SAP AG liefert ein Beispiel für ein Unternehmen, das seine Software intern durch eine Fachabteilung lokalisieren lässt:

- Intern drei Hauptentwicklungssprachen (Deutsch, Englisch, Japanisch); Lokalisierung in ca. 30 weitere

Sprachen

- Täglich zu übersetzender Umfang: ca. 50.000 Wörter (Systemtexte)
- ca. 180 firmeneigene Übersetzer, ca. 60 externe Agenturen
- Große Vielfalt an Quellformaten (ABAP, C++, Visual Basic, Java, XHTML, XML)

1.6.2 Fallbeispiel (2): KDE Internationalization

Nicht zuletzt auch der Bereich Open Source hat sich durch umfassende Lokalisierung seiner Ergebnisse profiliert, beispielsweise Linux und Mozilla (ca. 40 lokalisierte Versionen). Auch das KDE-Projekt (grafische Benutzeroberfläche für Linux) kann beachtliche Erfolge vorweisen:

- Das KDE Translators Center (<http://l10n.kde.org/teams>) verwaltet die Lokalisierungsaktivitäten in 92 Zielsprachen
- Ca. 4.000 Zeichenketten sind pro Sprachversion zu übersetzen
- Realisierung durch GNU-Werkzeug Gettext (PO-Dateien) und KBabel
- Auswertungen pro Zielsprache sind im Internet abrufbar
- Teilweise wird Basisarbeit bei "exotischen" Sprachen geleistet, in denen es bislang keine oder kaum lokalisierte Software gab.

1.7 Umfang eines Lokalisierungsprojektes

Bei der Planung eines Lokalisierungsprojektes sollten Sie folgende Bereiche in Betracht ziehen:

- Benutzeroberfläche (Menüs, Dialoge, allg. String Tables), abhängig von Programmiersprache
- Online-Hilfe / Dokumentation
- Berichte und sonstige von der Software ausgegebene Daten
- Software von Fremdanbietern (z.B. Laufzeitsysteme: Java RE, .Net Framework)

Kapitel 2 Kulturell bedingte Inhalte

Software-Entwicklung basiert stets auf gewissen Annahmen, um die Komplexität einer Architektur so gering wie möglich zu halten (Objektmodellierung). In einem internationalen Zusammenhang gelten viele offensichtlich erscheinende Annahmen nicht! Betrachten Sie folgende Beispiele:

- "A-Z enthält alle Buchstaben im Alphabet" (Stimmt nicht, im Schwedischen kommt der Buchstabe "ä" nach "z")
- "Alle Schriftarten enthalten Groß- und Kleinbuchstaben" (Das Konzept der Groß- und Kleinschreibung ist im Arabischen unbekannt; stattdessen gibt es unterschiedliche grafische Ausprägungen eines Zeichens am Beginn, Ende oder in der Mitte eines Wortes.)
- "Wörter im Satz werden durch Leerzeichen getrennt" (In ostasiatischen Sprachen wie dem Chinesischen gibt es keine Leerzeichen, weil jedes Zeichen für sich bedeutsam ist.)
- "In allen Sprachen ist die Interpunktion gleich" (Im Griechischen steht nach Fragesätzen der Strichpunkt an Stelle des Fragezeichens; im Spanischen muss ein Fragesatz zusätzlich durch ein invertiertes Fragezeichen eingeleitet werden, "¿Qué tal?")
- "Die Sortierreihenfolge ist für alle Sprachen mit dem selben Zeichensatz ist gleich" (Das stimmt nicht einmal in einer Sprache für sich; bereits im Deutschen gibt es zwei Weisen, wie Umlaute einsortiert werden - z.B. "ä" entweder zusammen mit "a", oder zwischen "ad" und "af")

2.1 Bedeutung von Farben

Im täglichen Leben machen wir häufig Gebrauch von Farbsymboliken. Farben können in unterschiedlichen Kulturen jedoch unterschiedliche Bedeutungen haben!

ROT

- In westlichen Kulturen: HALT oder GEFAHR
- In China: Glück & Freude

WEISS

- In westlichen Kulturen: Reinheit
- In fernöstlichen Kulturen: Trauer oder Tod

2.2 Bedeutung von Symbolen

Ebenso wie Farben werden Gesten und Symbole lokal unterschiedlich interpretiert:

- Piktogramme auf Symbolleisten und Schaltflächen haben oftmals eine angelsächsische Prägung, die in anderen Teilen der Welt nicht verstanden werden kann (z.B. Briefkasten, Papierkorb)
- Für Gefahrenzeichen und andere Piktogramme existiert eine internationale anerkannte Norm (ISO 7001)
- Gesten können sehr leicht missverstanden werden und sollten deshalb nur mit größter Vorsicht eingesetzt werden (Mauszeiger!)

2.3 Eigennamen und Produktnamen

Produktnamen sollen möglichst positive Eigenschaften eines Produktes assoziieren. Hier einige Eigentore aus dem Automobilbau:

- Der Ford Probe verkaufte sich in Deutschland sehr schlecht (die Kunden assoziierten mit dem Namen, dass das Auto irgendwie "unfertig" sei - nur wenige wollten ihr Auto immer "zur Probe" fahren)
- Der Chevy Nova war in Lateinamerika ein Ladenhüter ("no va" heißt auf Spanisch "geht nicht" oder "fährt nicht")
- Der Mitsubishi Pajero ist in Spanien praktisch unverkäuflich (es handelt sich um eine schlimme Beleidigung...)

Bei Eigennamen sind kulturell gegebene Besonderheiten zu beachten! Das in unserer Kultur gängige Schema "Vorname(n) - Nachname" ist nicht weltweit gültig.

2.4 Datum und Uhrzeit

Zeit- und Datumsanzeige müssen gebietsschemaspezifisch angezeigt werden können:

- USA: March 1, 2005 7:45 PM
- Deutschland: 1. März 2005, 19:45

Die Frage "Welcher Tag ist heute?" wird regional unterschiedlich beantwortet:

- In den meisten Teilen der Welt gilt der Gregorianische Kalender: 1. März 2005
- In Japan gilt zusätzlich der kaiserliche Kalender (Zeit seit der Thronbesteigung des amtierenden Kaisers)
- In Israel gilt der jüdische Kalender (5.762 Jahre seit der Erschaffung der Welt - der Monat dauert 28 Tage)
- Ebenso sind der arabische und der buddhistische Kalender in den betreffenden Kulturen gebräuchlich.

2.5 Zahlen und Währungen

Genau wie Zeit und Datum müssen auch Zahlen und Währungsangaben in den verschiedenen Sprachen unterschiedlich angezeigt werden:

- USA: 1,355,321.76 \$
- Deutschland: 1.355.321,76 €
- Frankreich 1 355 321.76 €

2.6 Lokalisierung von Text

Bei der Übersetzung in Fremdsprachen reicht das lateinische Alphabet offensichtlich nicht aus. Folgende Aspekte müssen außerdem betrachtet werden:

- Zeichensätze und Schriftarten
- Sortierreihenfolgen
- Ligaturen (im Arabischen z.B. können Zeichen nicht isoliert stehen, sondern müssen miteinander verbunden werden, wie bei der Schreibschrift - das stellt erhöhte Anforderung an der Darstellung z.B. auf einer Benutzeroberfläche)

2.6.1 Unicode

Unicode ist ein 16-Bit Zeichensatz, der alle bedeutsamen Schriftarten unterstützt (Lateinisch, Griechisch, Kyrrillisch, Hebräisch, Arabisch, CJK...)

- Unicode wird von einem herstellerübergreifenden Konsortium betreut (www.unicode.org)
- Umfasst ca. 30.000 verschiedene Zeichen. Die ersten 256 Zeichen entsprechen exakt ISO-8859-1 (ISO Latin-1)

- Moderne Betriebssysteme und Laufzeitumgebungen und Programmiersprachen unterstützen Unicode größtenteils
- Die Implementierung Unicode-fähiger Software ist trotzdem ein nicht-triviales Unterfangen (Mehrdeutigkeit von Zeichendarstellungen durch Kombinationen)

2.7 Aneinanderreihung von Zeichenketten

Naives Aneinanderreihen von Teilsätzen und Variablen im Quellcode führt unter anderem zu folgenden Problemen:

- Die Reihenfolge von Zeichenketten und Variablen ist sprachabhängig
- Im Gegensatz zum Englischen erfordern die meisten Sprachen eine Kongruenz zwischen Satzgliedern

2.8 Laufrichtung (LtR und Rtl)

Von rechts nach links betrachtet sieht die Welt manchmal anders aus:

- BIDI-Sprachen (z.B. Arabisch und Hebräisch) brauchen Rtl für ihre eigene Sprache und LtR für fremden Text
- Beispiel: Eine arabische Homepage: www.aljazeera.net



Die Homepage von Al-Jazeera am 28. Februar 2005; Beachten Sie, dass die Navigation auf der rechten Seite angeordnet ist, und dass auch alle Listfelder etc. seitenverkehrt sind. Die Umschaltung auf Englisch befindet sich sinnvollerweise links oben, wo man als westlich geprägter Beobachter zunächst hinschaut.

Kapitel 3 Konsequenzen für die Software-Entwicklung

3.1 Verwendung von Gebietsschemata

Gebietsschemata des Betriebssystems oder der Laufzeitumgebung bieten Unterstützung für:

- Zahlen, Datums- und Zeitformaten, Währungen
- Trennzeichen

Ebenso kann das Gebietsschema zur Sprachumschaltung verwendet werden.

3.2 Techniken zur Lokalisierung von Benutzeroberflächen

Zur Lokalisierung von Benutzeroberflächen wird heutzutage Spezialsoftware benutzt (z.B. Passolo).

- Geeignet zur Bearbeitung von Quelldateien im Format RC, RES, VCL, RESX
- Ebenso können auch binäre Dateien (DLL, EXE, OCX) bearbeitet und ausgegeben werden. Dadurch ist auch die Lokalisierung von Software möglich, deren Quellen nicht mehr vorliegen.
- Voraussetzung ist in jedem Fall, dass lokalisierungsrelevante Inhalte nicht fest kodiert werden!

3.3 Ein Wort zur Übersetzung

Wenn Ihr Lokalisierungsprojekt läuft, muss immer noch jemand die eigentliche Arbeit machen und die Texte übersetzen. Bedenken Sie hierbei:

- Translation-Memory-Systeme (z.B. Trados, Transit, Heartsome Araya und andere) können Übersetzungskosten drastisch reduzieren
- XML-basierte Technologien (XLIFF, TMX) unterstützen die Wiederverwertung von Übersetzungen und schützen somit Ihre Investitionen
- Nach Möglichkeit sollten muttersprachliche Übersetzer bevorzugt werden, die im Zielland wohnen

3.4 Regeln für GILT-fähige Software (Zusammenfassung)

Zusammenfassend einige Regeln zur möglichst problemlosen Erstellung von lokalisierbarer Software:

- Verwenden Sie Unicode als Zeichenformat, wo immer möglich!
- Sorgen Sie dafür, dass lokalisierungsrelevante Inhalte (Text, Formate, Trennzeichen) nicht fest kodiert werden!
- Verwenden Sie nach Möglichkeit Resource Bundles oder sonstige von Ihrer Entwicklungsumgebung bereitgestellte Techniken um Programmcode und Texte getrennt zu halten!
- Implementieren Sie Gebietsschemata!
- Suchen Sie nach geeigneten Partner (Berater, Vertriebspartner, Dienstleister) bei der Durchführung Ihrer Lokalisierungsprojekte!