# 7.13 Internationalisierung

**von Kerstin Dittert**

Lange bevor Softwareprodukte international eingesetzt werden sollten, entwickelte die Industrie schon Produkte für globale Märkte. Stellen Sie sich folgendes Szenario vor:

Ein Automobilhersteller des europäischen Fantasielandes *Expando* entwickelt einen neuen PKW, der sich im Heimatland zum Verkaufsschlager entwickelt. Das Management beschliesst, den Wagen auf den europäischen und amerikanischen Markt zu bringen. Ein Team erhält den Auftrag festzustellen, welche Ergänzungen notwendig sind, um das Fahrzeug international zu vertreiben.

Die Gruppe erstellt in aufwändiger Detailarbeit einer Liste aller Beschriftungen im Wageninneren. Sie fordern eine Übersetzung dieser Texte und der Betriebsanleitung in alle europäischen Sprachen an. Mit beträchtlichem Aufwand werden die Texte übersetzt. Der Wagen lässt sich nun für jedes Zielland anders konfigurieren:

- Alle Beschriftungen sind in der Landessprache verfasst.
- Das eingebaute Navigationssystem ist für jedes Land mit den entsprechenden Karten und der passenden Sprachausgabe ausgestattet.
- Für jedes Land gibt es eine andere Betriebsanleitung.

Das Marketing beginnt, der Wagen wird europaweit und in Amerika vertrieben. Zur Überraschung aller entwickelt sich der ausländische Umsatz mehr als schleppend. In England wird nicht ein einziges Auto verkauft. In Deutschland veröffentlich der ADAC Testergebnisse, nach denen der Wagen bei 180 km/h in der Kurve ausbricht. In Amerika wird der Wagen gar nicht erst für den Verkauf zugelassen! Das Management beruft eine Krisensitzung ein. Es war doch soviel Aufwand in die Übersetzungsarbeit gesteckt worden! Wieso nahmen die Kunden den Wagen nicht an? Die Ursachen für die mangelnde Akzeptanz werden gemeinsam vom Marketing und Mitarbeitern der Konstruktionsabteilungen untersucht. Die Ergebnisse fallen ernüchternd aus:

- In England hatte man vergessen, das Lenkrad rechts einzubauen.
- In der amerikanischen Version hatte man vergessen, den Tacho auf die Masseinheit Meilen/Stunde umzurechnen.
- Aufgrund eines Tempolimits vom 130 km/h in *Expando*, wurde die Konstruktion auf die Höchstgeschwindigkeit ausgelegt. Der Fehler wurde auch bei Prüfungen nicht entdeckt, da spezielle Tests für einzelne Länder-Konfigurationen nicht durchgeführt wurden!

Bei der Konzentration auf die sprachlichen Aspekte hatte man andere Aspekte wie Masseinheiten, technischen Randbedingungen etc. völlig vergessen.

## 7.13.1 Globale Märkte erfordern neue Prozesse

So überspitzt die obige Geschichte auch sein mag: Die Erstellung international einsetzbarer Software stellt Sie als Software-Architekten vor ähnliche Probleme. Der Aspekt der Internationalisierung beeinflusst den gesamten Softwareentwicklungsprozess. Sie sollten ihn deshalb frühzeitig in Ihre Überlegungen mit einbeziehen. Neben der Berücksichtigung verschiedenen Sprachen müssen Sie manchmal auch nationale Unterschiede wie Währungen, gängige Formatierungen, Zeitzonen etc. berücksichtigen.

Je nach Zielgruppe und angestrebtem Markt kann es sogar erforderlich sein, kulturelle und politische Aspekte mit zu betrachten. Dies ist besonders wichtig für Web-Anwendungen, die sich an ein weltweites und unspezifisches Publikum wenden (wie z.B. Suchmaschinen, Free-Mail-Provider etc.).

### Lokalisierung und Internationalisierung

Der gesamte Themenkomplex internationaler Anwendungen wird häufig mit den Schlagworten *Internationalisierung* und *Lokalisierung* in Zusammenhang gebracht..

Unter ***Lokalisierung (l10n[1])*** versteht man die Anpassung einer *bestehenden* Software an eine andere Sprache oder ein anderes Land. Dieser Prozess umfasst die Erstellung und Anbindung angepasster Ressourcen, wie z.B. Texte, Bilder oder Sprachdateien. Die Art der Ressourcen-Anbindung ist dabei beliebig. Es kann sich sowohl um extern referenzierbare Dateien eines Software-Paketes handeln, als auch um unterschiedliche Anwendungsdistributionen für jede Sprache oder jedes Land.

Sobald *ein* Software-Paket mehrere Sprachen und unterschiedliche nationale Merkmale unterstützt, spricht man von der ***Internationalisierung (i18n[2])*** einer Anwendung. Eine *internationalisierte* Anwendung beinhaltet also immer *lokalisierte* Ressourcen.

## 7.13.2 Dimensionen der Internationalisierung

Internationalisierung ist teuer, und nicht jede Anwendung ist für den chinesischen Markt mit einer Fülle von über 40.000 Schriftzeichen geplant. Allgemeine generische Ansätze sind aufwändig in der Entwicklung, ohne dass da-

[1] abgeleitet vom engl. *Localization*: das Wort beginnt mit ‚l', hört mit ‚n' auf, dazwischen sind 10 Buchstaben

[2] abgeleitet vom engl. *Internationalization*

mit zwangsläufig ein wirtschaftlicher Mehrwert verbunden ist. Zu Beginn des Architekturentwurfs sollten Sie deshalb die Anwender-Zielgruppe möglichst genau eingrenzen. Dabei sollten Sie folgende Fragen stellen:

- Welche Sprachen sollen unterstützt werden?
- Verwenden alle Sprachen ausschließlich lateinische Schriftzeichen?
- Wird Text immer horizontal von links nach rechts geschrieben?
- In welchen Ländern soll die Anwendung eingesetzt werden?

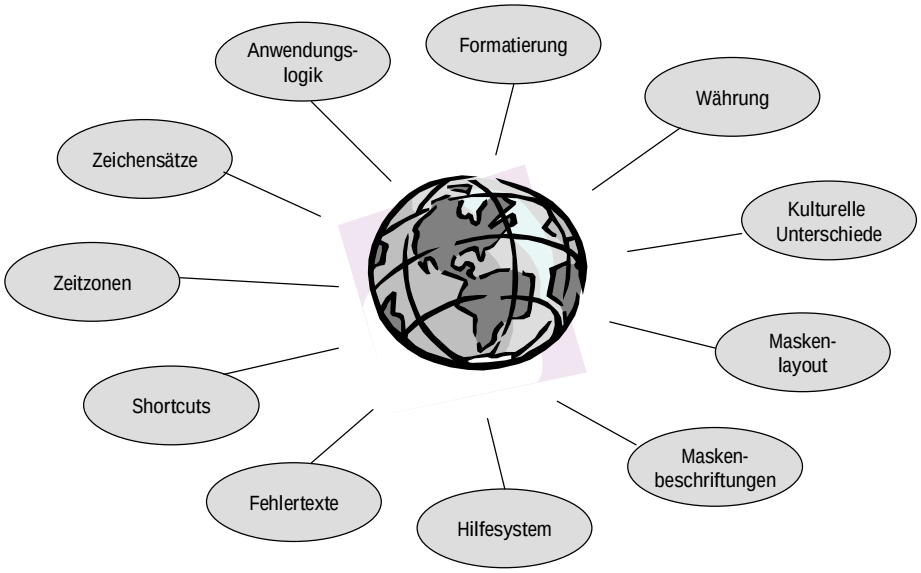


**Abbildung 1 - Aspekte der Internationalisierung**

Sie und Ihr Team müssen herausfinden, wie hoch der Anteil sprachabhängiger oder landestypischer Anwendungslogik ist. Folgende Fragen können Ihnen dabei helfen:

- Welche Formatierungen werden benötigt?
- Müssen unterschiedliche Währungen verarbeitet werden?
- Enthalten die Anwendungsdaten Bezeichnungen, die übersetzt werden müssen?
- Enthält die Anwendung zeitabhängige Dienste?
- Welchen Umfang soll das Hilfesystem haben?
- Welche Anwendungsteile sollen über Tastaturkürzel[3] gesteuert werden?
- Gibt es kulturelle Unterschiede, die für die Anwendung relevant sind?

Haben Sie diese Fragen beantwortet, so können Sie entscheiden, welche der folgenden Konzepte für Ihre Architektur relevant sind.

[3] Shortcuts (z.B. CTRL-C) oder Mnemonics (ALT-Buchstabe).

## 7.13.3 Lösungskonzepte

Im Zusammenhang mit der Internationalisierung stellen sich vielfältige architekturrelevante Fragen. Aber welche Konzepte greifen, um die Komplexität zu reduzieren? Das wichtigste Hilfsmittel wird häufig im Sog allgemeiner und generischer Ansätze vergessen: die Reduzierung auf das Wesentliche!

Treffen Sie eine Auswahl und grenzen Sie die Internationalisierungsaspekte ein.

### Aspekte der Mehrsprachigkeit

Fast alle internationalisierten Anwendungen unterstützen mehrere Sprachen, manche sogar auch Sprachvarianten, wie z.B. *Britisches* und *Amerikanisches Englisch*. Die Notwendigkeit solch feiner Unterscheidung müssen Sie sorgfältig abwägen. Die Ziele einer hohen Anwenderakzeptanz und des möglichst geringen Entwicklungs- und Wartungsaufwandes stehen sich komplementär gegenüber. Für die Rechtschreibprüfung einer Textverarbeitung ist die Berücksichtigung von Sprachvarianten unerlässlich. Die Webseiten einer Internet-Suchmaschine können hingegen ausschließlich in einer englischen Sprachvariante angeboten werden.

Sämtliche Oberflächenelemente müssen sprachabhängig beschriftet werden und werden deshalb als externe Ressourcen benötigt. Rückmeldungen an den Anwender wie Fehlermeldungen, Statuszeilentexte etc. müssen ebenfalls in allen Sprachen vorliegen und beim Satzbau den Regeln der jeweiligen Grammatik folgen.

Textkonstanten sind in verschiedenen Sprachen unterschiedlich lang und beanspruchen dementsprechend mehr oder weniger Platz. Grafische Benutzeroberflächen mehrsprachiger Systeme sollten deshalb niemals ein statisches GUI-Layout[4] vorgeben. Statt dessen sollte sich die Größe beschrifteter Elemente (Buttons, Label, Menüeinträge etc.) dynamisch an den enthaltenen Text anpassen.

Die Oberflächenorientierung einer GUI folgt immer der Textausrichtung der Sprache. Menüs werden z.B. in deutschsprachigen Ländern von links nach rechts angeordnet, in arabisch sprechenden Ländern jedoch von rechts nach links. Ebenso werden die Elemente einer Maske entweder von links nach rechts oder umgekehrt angeordnet. Falls also Sprachen mit unterschiedlicher Textausrichtung unterstützt werden sollen (z.B. Englisch und Arabisch), so wird eine variable GUI-Komponentenorientierung benötigt. Hierfür muss das

[4] Bei einem statischen GUI-Layout sind die Position und Größe der einzelnen Oberflächenelemente fest vorgegeben. Bei unbekannten Beschriftungslängen muss also vorsorglich einiges an zusätzlichem Platz reserviert werden, damit die Texte nicht abgeschnitten werden.

GUI-Layout-Management neben der variablen Positionierung und Größenbestimmung um dynamische Orientierung ergänzt werden.

Die folgende Heuristik kann Ihnen helfen, Ihre Architektur für mehrere Sprachen auszulegen:

- Legen Sie zunächst die für das System erforderlichen Sprachen fest (z.B. Deutsch, Englisch, Arabisch etc.).
- Überprüfen Sie sorgfältig, ob Sie unterschiedliche Sprachvarianten benötigen (z.B. Deutsch (Schweiz), Deutsch (Deutschland)). Der Übersetzungsaufwand steht häufig in keinem vernünftigen Verhältnis zum Mehrwert innerhalb der Anwendung!
- Setzen Sie bei der Masken-Gestaltung Layout-Manager-Klassen ein. Diese passen die Grösse der Maskenelemente an die aktuellen Beschriftungslängen an.
- Sollte die von Ihnen eingesetzte Programmiersprache kein dynamisches Layout-Management unterstützen, so berücksichtigen Sie zusätzlichen Platzbedarf für andere Sprachen. Geben Sie in Entwurfsrichtlinien maximale Beschriftungslängen vor (z.B. 30 Zeichen).
- Ordnen Sie Menüs und andere Oberflächenelemente gemäß der Textausrichtungen der verwendeten Sprache an.
- Legen Sie sämtliche sprachabhängigen Elemente in Ressource-Dateien ab (Oberflächenbeschriftungen, Fehlertexte, Shortcuts[5], Tooltips, Audio-Dateien etc.).
- Beachten Sie bei Fehlermeldungen den unterschiedlichen Satzbau verschiedener Sprachen. Einige Klassenbibliotheken bieten hierfür spezielle Formatierungsklassen an[6].

## Anbindung externer Ressourcen

Sämtliche statischen Texte und alle übrigen sprachabhängigen Inhalte, wie Bilder oder Audio-Dateien, werden in sprachspezifischen Ressource-Dateien oder –Klassen von der Anwendungsimplementierung entkoppelt. Abbildung 2 zeigt ein Design zur Anbindung von Ressource-Dateien innerhalb der GUI-Schicht.

Zur Laufzeit bestimmt das Programm die aktuelle Kombination von Land und Sprache. Dies kann mit Betriebssystemmitteln, über ein Benutzerprofil oder auf Anforderung des Benutzers erfolgen. Diese Länder-Sprachkombination wird allen internationalisierten Anwendungsteilen zur Verfügung gestellt[7].

[5] viele Anwendungen mit GUI sollen auch mauslos gesteuert werden. Hierfür werden Tastaturkürzel wie *Shortcuts* (z.B. CTRL-C) oder *Mnemonics* (z.B. ALT-D) eingesetzt.

[6] in Java ist dies die Klasse `MessageFormat`

[7] sie wird häufig als *Locale*-Objekt bezeichnet.

Anschließend können die jeweils benötigten Ressourcen geladen werden. Hierfür werden alle Elemente der GUI-Schicht benachrichtigt. Sie lesen die jeweils benötigten Daten aus den Ressource-Dateien aus und stellen sie an der Oberfläche dar.

xxx: GUI-Schicht

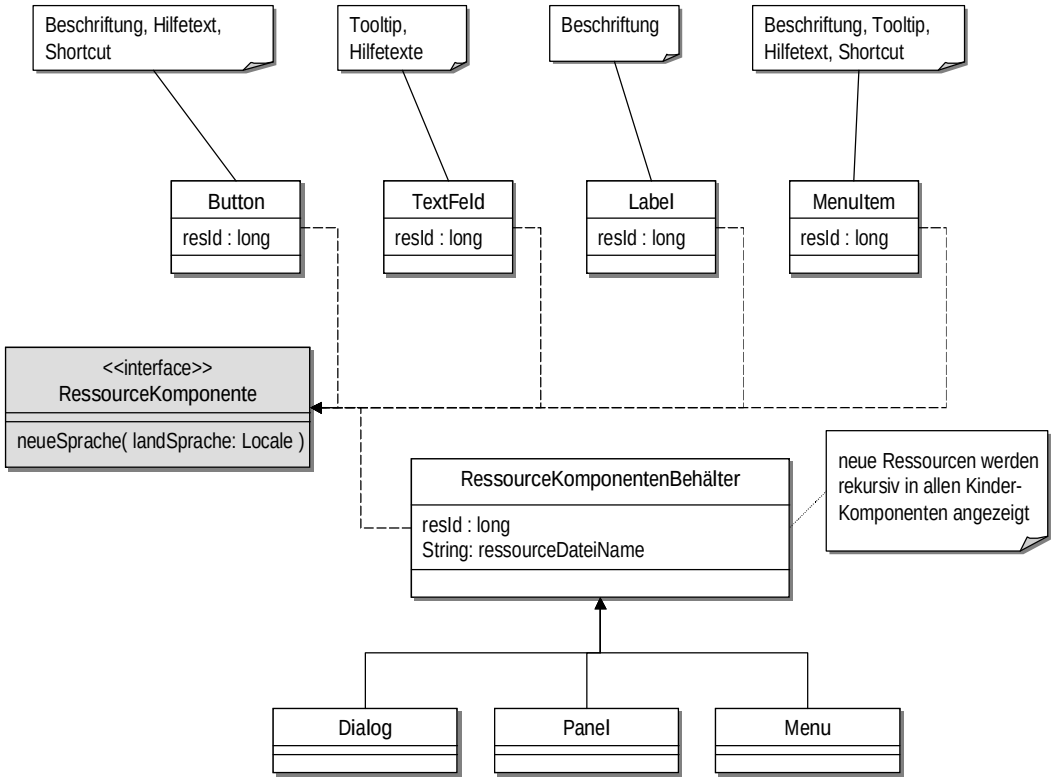


**Abbildung 2 - Ressourcen-Anbindung innerhalb der GUI-Schicht**

Auch Tastaturkürzel gehören zu den Daten, welche erst nach Auswahl der Sprache zugeordnet werden können. Sie werden deshalb ebenfalls als Ressourcen verwaltet und den Oberflächenelementen dynamisch zugeordnet Hierbei ist besonders darauf zu achten, dass die Kürzel als Buchstabe in der zugeordneten Beschriftung enthalten sind und pro Maske eindeutig sind.

Das gesamte Hilfessystem (hierzu gehören auch Tooltips und Benutzerhandbücher) muss ebenfalls in allen Sprachen vorliegen.

Fehlermeldungen und sonstige Nachrichten an den Benutzer werden auch als Ressourcen gehalten und in einem zentralen Pool verwaltet. Dies hat den angenehmen Nebeneffekt, dass eine Standardisierung der Nachrichtentexte einfach zu erreichen ist. Nachrichten können Platzhalter enthalten, welche zur Laufzeit kontextabhängige Inhalte aufnehmen können. Ein Beispiel hierfür wären zwei Nachrichtenformate

„Benutzer {0} ist im System nicht bekannt.“

„Unknown user {0}.“

welche zur Laufzeit mit dem Benutzernamen „Meier“ gefüllt werden.

Achten Sie darauf, dass die Reihenfolge der Satzteile sowie die Interpunktion in jeder Sprache unterschiedlich sein kann. Formatieren Sie Meldungen mit speziellen Klassen, welche die Platzhalter füllen und die Sprachgrammatik berücksichtigen.

## Zeichensätze und Zeichenkodierung

Sobald die Sprachen und Sprachvarianten festgelegt worden sind, können geeignete Kodierungsstandards für alphanumerische Zeichen identifiziert werden. Der ASCII-Standard wird keinesfalls ausreichen, da dieser für den Einsatz in den USA optimiert wurde. Ein britischer Anwender würde bereits das £-Zeichen für die Währungsdarstellung vermissen. Im Idealfall kommt man im europäischen Raum mit einer der ISO-Kodierungen zur Unterstützung mehrer Sprachen aus[8].

Die Angabe geeigneter Kodierungen gewährleistet jedoch nur die richtige *programminterne* Darstellung der Zeichen. Sind auf dem Rechner des Anwenders keine dazu passenden Zeichensätze installiert, so kann es passieren, dass Buchstaben nicht darstellbar sind und als Kästchen oder sonstige Platzhalter am Bildschirm angezeigt werden. Ein solches Verhalten ist zum Beispiel zu erwarten, wenn auf einem Computer in Deutschland japanische oder arabische Schriftzeichen dargestellt werden sollten. In diesem Fällen muss gewährleistet werden, dass passende Zeichensätze mit zum Distributionsumfang der Software gehören[9].

Stellen Sie sicher,

- dass jede Sprache in einem passenden Zeichensatz kodiert wird[10].,
- in der Laufzeitumgebung die passenden Fonts installiert sind.

## Sprachabhängige Anwendungsdaten

Die Notwendigkeit mehrsprachiger Anwendungsdaten wird häufig übersehen. Die schönsten Maskenübersetzungen und angepassten Fehlermeldungen nützen dem Anwender wenig, wenn er beispielsweise in Auswahllisten (z.B. für Farben, Warengruppen etc.) nur fremdsprachige Wörter sieht.

[8] z.B. ISO-8859-1 für die Sprachen Albanisch, Baskisch, Katalanisch, Dänisch, Niederländisch, Englisch, Finnisch, Französisch, Deutsch, Isländisch, Irisch, Italienisch, Norwegisch, Portugiesisch, Rhäto-Romanisch, Schottisch, Spanisch, Schwedisch

[9] Der Windows-Freeware-Font „Bitstream Cyberbit“ unterstützt mehr als 30.000 Symbole und kann u.a. Arabisch, Hebräisch und Thailändisch darstellen.

[10] eine Übersicht gängiger Zeichensätze für den europäischen Sprachraum finden Sie in [CzaDei01]

Die meisten sprechenden Bezeichnungen der Anwendungsdaten müssen deshalb ebenfalls mehrsprachig vorliegen. Potentielle Kandidaten dafür sind Attribute,

- die der Bezeichnung dienen und als Freitext eingegeben werden können (z.B. Artikelnamen),
- die einen endlichen Wertebereich haben (z.B. Artikeleigenschaften wie Farbe, Grösse, Material). Derartige Attribute werden meist in Auswahllisten[11] dargestellt.

Die Unterstützung mehrsprachiger Anwendungsdaten ist ein komplexes Thema. Neben dem Entwurf eines geeigneten Objekt- und Datenmodells müssen Mechanismen zur Übersetzung bereit gestellt werden. Die Erhaltung konsistenter Datenbestände erfordert häufig die Auszeichnung einer führenden Sprache. Besonders schwierig wird es, wenn Altdaten übernommen oder Fremdsysteme integriert werden müssen, da diese oft nicht mit den neuen Konzepten harmonieren.

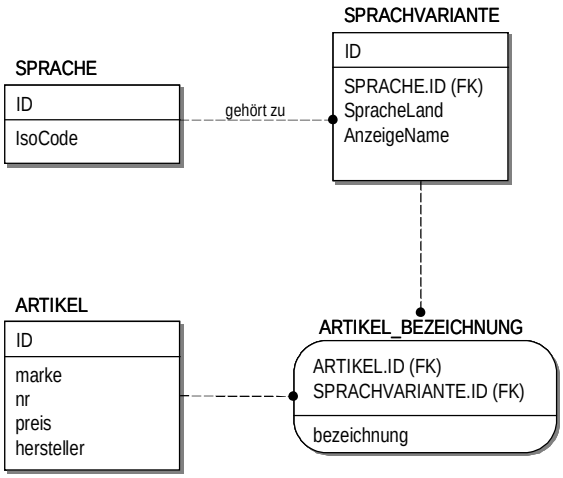


**Abbildung 3 - ER-Diagramm für mehrsprachige Anwendungsdaten**

Das Objekt- und Datenmodell muss um Sprachvariationen ergänzt werden. Im Objektmodell führt dies zu keiner weitreichenden Modellveränderung, da zur Laufzeit jeweils nur eine konkrete Sprachinstanz referenziert wird. Das Datenmodell muss jedoch für alle sprachabhängigen Anwendungsdaten um Sprach- und Länderschlüssel ergänzt werden.

Konzeptionell sind relativ statische Daten von „dynamischeren“ Attributen zu unterscheiden. Eher statische Daten sind beispielsweise Farbkataloge, Postleitzahlenverzeichnisse etc[12]. Diese werden periodisch und an zentraler Stelle geändert und können über geeignete Importschnittstellen aktualisiert werden.

[11] auch Combo-Boxen genannt

[12] in sehr vielen Fällen handelt es sich um die Klartextbezeichnung geschlüsselter Felder.

Sofern die Importquellen nur einsprachige Daten zur Verfügung stellen, ist anschließend eine manuelle Übersetzung erforderlich. Diese kann insbesondere auch bei Altdatenübernahmen erforderlich sein.

Im Gegensatz dazu sind *Bezeichner*, wie beispielsweise Artikelnamen, Warengruppennamen etc. dynamische Daten, welche vom Anwender gepflegt werden. Da diese Datenpflege dezentral ist, muss festgelegt werden, wie die Übersetzungen dieser Daten erfolgen. Hierfür sind unterschiedliche Ansätze möglich:

- Es wird eine führende Systemsprache ausgezeichnet (z.B. Englisch). Bei der Bearbeitung sprachabhängiger Bezeichner muss die Übersetzung in der Systemsprache mit aktualisiert werden. Bei der Anzeige wird stets versucht, auf die Übersetzung der aktuell gewählten Sprache zuzugreifen. Liegt eine solche Übersetzung nicht vor, so wird die Bezeichnung statt dessen in der Systemsprache angezeigt. Übersetzungsdialoge stehen zusätzlich zur Verfügung. In regelmässigen Abständen werden nicht übersetzte Bezeichnungen einem Übersetzungsteam automatisch vorgelegt. Der Vorteil dieser Vorgehensweise ist die einfache und schnelle Bearbeitung der Daten durch den Anwender. Qualität und Quantität der Übersetzungen können jedoch stark schwanken und sind durch das System nicht zu beeinflussen.
- Bezeichnungen müssen bei der Bearbeitung stets in alle Systemsprachen übersetzt werden. Hierfür kann ein workflow definiert werden, in den ein Übersetzungsteam einbezogen wird. Erst nach der Übersetzung werden die Daten im System produktiv. Dem Vorteil einer hohen Anwenderunterstützung durch vollständig mehrsprachige Daten stehen der hohe Arbeitsaufwand und Zeitverluste durch den Workflow-Prozess gegenüber.

Mehrsprachige Bezeichnungen erzeugen durch die permanente Übersetzungsproblematik einen hohen Arbeitsaufwand. Dies gilt sowohl für den Entwurf und die Implementierung des Systems, als auch für die laufenden Kosten des Produktionssystems, welches ständig manuelle Eingriffe erfordert.

Prüfen Sie deshalb sorgfältig, ob die Einführung mehrsprachiger Bezeichnungen einen großen Mehrwert des Systems darstellt. In den meisten Fällen ist es ausreichend, ausschließlich die statischeren Anwendungsdaten wie z.B. Kataloge mehrsprachig vorliegen zu haben.

Bei der Umsetzung mehrsprachiger Anwendungsdaten

- sollten Sie eine Referenzsprache auszeichnen,
- den Umfang mehrsprachiger Daten auf das absolut erforderlicher Minimum reduzieren,
- Daten- und Objektmodelle um Sprach- und Länderausprägungen erweitern,
- den Arbeitsablauf zur Bearbeitung der im produktiven Betrieb anfallenden Übersetzungsarbeiten definieren.

## Formatierung

Datums- und Zeitangaben, Währungsfelder und numerische Angaben werden in unterschiedlichen Ländern auf verschiedene Art formatiert. Die Darstellungsreihenfolge, Trennzeichen und die Position einzelner Bezeichner variieren. Ignoriert man die länderspezifischen Konventionen, so kann das in einigen Fällen zur kompletten Fehlinterpretation der Anwendungsdaten führen. Die deutsche Datums- und Zeitangabe „12.02.2001 14:22“ wird in anderen Ländern folgendermaßen dargestellt:

- Frankreich: 12/02/2001 14:22
- USA: 02/12/2001 2:22 PM
- Israel: 14:22 12/02/2001

Die *visuelle* Formatierung eines Datumsfeldes unterscheidet sich also z.B. in Frankreich und den USA nicht, der *Inhalt* wird jedoch länderabhängig unterschiedlich interpretiert!

Die Formatierung von Datum, Zeitangaben, Geldbeträgen mit Währungssymbol sowie numerischer Felder setzen Sie mit Hilfe spezieller Formatierungsklassen um. Hierfür werden Standardbibliotheken eingesetzt, welche die länderspezifischen Regeln enthalten und bei Bedarf um zusätzliche Regeln ergänzt werden können[13].

xxx: GUI-Schicht

- Formatierungsaufgaben setzen Sie innerhalb der GUI-Schicht mit Hilfe spezieller Formatierungskomponenten um.

## Anwendungslogik und Steuerung

Programmierer gehen oft implizit von bestimmten Annahmen aus, die in internationalen Anwendungen nicht immer gegeben sind. Tabelle 1 nennt die häufigsten Unterstellungen und gibt Gegenbeispiele dazu an. Prüfen Sie, ob alle Annahmen für die Anwendung zutreffend sind! Ist dies nicht der Fall, so muss die Anwendungslogik darauf abgestimmt werden.

Sobald ein Programm länderübergreifend eingesetzt wird, sind häufig auch verschiedene Währungen mit im Spiel. Neben dem Formatierungsaspekt, der bereits angesprochen wurde, ist die Mehrwährungsfähigkeit eine Anforderung, die auch die Anwendungslogik betrifft. Gegebenenfalls müssen Umrechnungen vorgenommen, Wechselkurse importiert und eine Systemwährung ausgezeichnet werden. Sie sollten frühzeitig feststellen, ob die Kurse zeitnah über entsprechende Serviceprovider abgefragt werden müssen (z.B. bei Online-Banking-Anwendungen), oder ob die Umrechnungstabellen periodisch ins System importiert werden können.

[13] die Java-Standard-Bibliotheken enthalten hierfür z.B. die Klassen `DateFormat`, `DecimalFormat` und `NumberFormat`

| Annahme | Gegenbeispiel |
|---|---|
| Das Alphabet besteht aus den Zeichen ‚A' bis ‚Z'. | *In Dänemark folgen die Umlaute nach dem ‚Z'.* |
| Worte werden durch Leerzeichen voneinander getrennt. | *Im Thailändischen gibt es keine Worttrennung.* |
| Die Interpunktion ist in allen Sprachen gleich. | *Im Spanischen wird eine Frage von ‚¿' eingeleitet und mit ‚?' beendet.* |
| Ein Buchstabe kann in einem Byte gespeichert werden. | *8 bit bieten Speicherplatz für maximal 256 Zeichen. Die chinesische Sprache kennt über 40.000 Zeichen!* |
| Sprachen, die das gleiche Alphabet verwenden haben auch die gleiche Sortierreihenfolge. | *‚A' und ‚Ä' repräsentieren im Deutschen den gleichen Buchstaben mit unterschiedlicher Aussprache. Sie sind bezüglich der Sortierung äquivalent.*<br>*Im Schwedischen stehen diese Symbole für unterschiedliche Buchstaben mit anderer Sortierung.* |
| Es wird der gregorianische Kalender verwendet. | *In einigen arabisch sprechenden Ländern gilt zusätzlich der islamische Kalender.* |

**Tabelle 1 - Internationalisierung: ungültige Annahmen bzgl. der Anwendungslogik**

Arbeitet eine Anwendung mit Zeitstempeln, so müssen Sie feststellen, ob Ihre Anwendung über mehrere Zeitzonen verteilt ist. Berücksichtigen Sie dabei, dass die Zeitzonen nicht immer mit Landesgrenzen zusammenfallen! Länder mit grosser Flächenausdehnung wie z.B. die USA umfassen mehrere Zeitzonen. Bei zeitzonenübergreifenden Anwendungen müssen Sie einen zentralen Zeitstempelservice vorsehen.

- Stellen Sie sicher, dass die Anwendungslogik nicht durch Stringvergleiche[14]. gesteuert wird! Arbeiten Sie statt dessen mit numerischen Konstanten, Klassenbezeichnern etc. Überprüfen Sie die Einhaltung dieser Vorgaben in Code- und Design-Reviews.
- Setzen Sie spezielle Klassen für die Sortierung von Zeichenketten ein. Bei Bedarf können diese um sprachspezifische Sortierungsregeln ergänzt werden.
- Legen Sie in mehrwährungsfähigen System eine Referenzwährung fest und überprüfen Sie, wie zeitnah die Kursumrechnungen erfolgen müssen.

[14] 4GL-Tools generieren häufig derartige Vergleiche, um die Anwendungscontroller zu implementieren. Hierfür wird auf Vergleiche mit Menü- oder Buttonbeschriftungen zurückgegriffen.

- Sehen Sie in zeitzonenübergreifenden Anwendungen einen Zeitstempelservice vor.

### Kulturelle Besonderheiten

Soll eine Anwendung kulturübergreifend eingesetzt werden (z.B. im Internet), so ist besondere Vorsicht bei der Verwendung von Zeichen, Symbolen und Metaphern geboten. Im harmloseren Fall werden sie schlichtweg nicht verstanden, im ungünstigen Fall können sie aber auch eine völlig gegensätzliche Bedeutung haben und deshalb zu Akzeptanzverlusten führen. Beispielhaft hierfür ist die symbolische Verwendung von Zahlen: Vielen Westeuropäern gilt die 13 als Unglückszahl, in Hongkong ist es hingegen die 7. Mit der Zahl 666 verbinden die meisten Deutschen nichts, in den USA ist sie dagegen das Zeichen des Teufels.

Die Missverständnisse können noch weitaus größer werden, wenn Abbildungen von Gesten, Farben, Verkehrsschildern etc. zur Kommunikation mit dem Anwender eingesetzt werden. Die Bedeutung hängt oftmals stark vom Kulturkreis ab.

- Stimmen Sie die Bild- und Zeichensprache Ihrer Anwendung auf den Kulturkreis der Anwender ab.
- Ist die Zielgruppe stark heterogen, so verzichten Sie völlig auf den Einsatz von Symbolen und Metaphern.

## 7.13.4 Interaktionen

Internationalisierte Anwendungen verwenden spezielle Konzepte für die Architekturbausteine *Persistenz*, *Benutzerschnittstelle*, *Ablaufsteuerung*, *Fehlerbehandlung* und *Protokollierung*. Im einzelnen gehören hierzu

- die Verwaltung mehrsprachiger Kataloge und Bezeichnungen in der Persistenzsschicht,
- die sprachabhängige Oberflächenbeschriftung, das dynamische Maskenlayout und die Formatierung in der GUI-Schicht,
- Besonderheiten bei der Zeichen- und Stringverarbeitung in der Ablaufsteuerung,
- die Verwendung mehrsprachige Texte mit unterschiedlichem Satzbau in der Fehlerbehandlung,
- die Bereitstellung eines Zeitstempelservice für die Protokollierung.

## 7.13.5 Weiterführende Literatur

[CzarDei01] stellen beschreiben ausführlich das Design und die Realisierung mehrsprachiger Java-Anwendungen.

[Unicode] bietet eine Fülle von Ressourcen zum Thema Mehrsprachigkeit.


[CzaDei01] Java Internationalization, David Czarnecki, Andrew Deitsch, O'Reilly 2001

[OConn1] Internationalization: Localization with ResourceBundles, John O'Connor, Java Delevoper Connection 1998, http://devoloper.java.sun.com/…

[OConn2] Internationalization: An overview, John O'Connor, Java Delevoper Connection 1998, http://devoloper.java.sun.com/…

[KreLan] Internationalization using Standard C++, Klaus Kreft, Angelica Langer, C/C++ User Journal, 1997, http://home.CameloT.de/langer/Articles/Internationalization/I18N.htm

[Unicode] Unicode und Internationalisierung http://www.unicode.org