Lokalisierung von Maschinensoftware

Schwierigkeiten meistern

Die Lokalisierung von Maschinensoftware erfordert häufig spezielle Prozesse. Besondere Schwierigkeiten treten etwa bei der Trennung von Text und Programmiercode oder bei der Übersetzung von Sätzen ohne Kontext bzw. von zerstückelten Sätzen auf.

Die rasch fortschreitende Automatisierung in der industriellen Fertigung, die Verbreitung der Elektronik in vielen Geräten des Alltagslebens und der Einsatz des Internets als Kommunikations- und Arbeitsplattform bringen althergebrachte Lokalisierungskonzepte durcheinander. Sie stellen neue Herausforderungen für Übersetzer und Softwareentwickler dar. Vor etwa 50 Jahren hielten die ersten numerischen Steuerungen ihren Einzug in die Maschinenwelt. Heute findet man Softwaretexte in so unterschiedlichen Situationen wie beim Autofahren, im OP-Saal, beim Steuern eines Atomkraftwerks oder bei der Bedienung einer Maschine. All diese Meldungen und Befehle will der Benutzer in seiner Muttersprache lesen und verstehen.

Bisher haben die Lokalisierung von Maschinensoftware und Embedded-System-Software die Aufmerksamkeit der Lokalisierungsbranche wenig erregt, obwohl (oder eher weil?) die Herausforderungen recht groß sind. Das führt

dazu, dass viele Hersteller von softwaregesteuerten Maschinen, Anlagen oder Geräten manchmal sehr mühsame und kostspielige Wege gehen, um ihre Software zu lokalisieren. Aber auch Übersetzer haben ihre Schwierigkeiten mit der Bearbeitung von Maschinentexten. Das liegt zum einen an fehlenden Kenntnissen der Softwareingenieure in Bezug auf Internationalisierung und Lokalisierung und zum anderen daran, dass es noch an einheitlichen Lokalisierungskonzepten mangelt. Viele Hersteller sind jedoch inzwischen bestrebt, die Programmierung ihrer Anwendungen zu standardisieren, wie es mit der Norm IEC 61131-3 oder mit der DIN 66025 für die NC-Programmierung der Fall ist.

Bei der Lokalisierung von Maschinensoftware treten technische, sprachliche und organisatorische Herausforderungen auf. Um besser zu verstehen, was die Lokalisierung von Maschinensoftware von „normalen" Lokalisierungsprojekten unterscheidet, fassen wir zuerst zusammen, wie die klassische Lokalisierung einer Anwendung abläuft.

Eine klassische Anwendung wird meistens von Anfang an für den internationalen Einsatz entwickelt. Bei jeder Programmiersprache läuft das Verfahren etwas anders. Aber das Grundprinzip ist, dass die Texte der verschiedenen Programmobjekte (Meldungen, Menüs, Schaltflächen, Dialogtitel, Feldbezeichnungen usw.) in separaten Dateien („Ressourcendateien") gespeichert sind. Im Falle von Windows-.NET-Programmen erzeugen Programmierer für die Übersetzung aus diesen vielen Ressourcendateien sogenannte Satellite Assemblies (*.resources.dll).

Andere Programme verpacken die Softwarestrings zur Weiterbearbeitung in vergleichbare Dateien (z. B. ResourceBundle bei Java-Applikationen). Ab diesem Punkt kommen Lokalisierungsprogramme wie SDL-Passolo, Visual Localize oder Alchemy Catalyst zum Einsatz. Diese Programme können mit Hilfe von Parsern (Filtern) binäre Quelldateien (z. B. DLLs oder EXE-Dateien) und andere Formate (z. B. XML-Dateien) direkt einlesen. Bei Programmiersprachen wie Visual C++ oder C# zeigen sie im Übersetzungseditor die Oberfläche der Software an. Der Übersetzer kann somit die Dialoge oder die Feldlänge anpassen, wenn er mehr Platz für einen Text braucht.

Spezielle Programmiersprachen erschweren die Abwicklung

Die Programmierung von Software für Maschinen und Geräte erfolgt zum Teil auch in Programmiersprachen wie C++. Wenn die Entwickler sich bei der Programmierung an etablierte Lokalisierungskonzepte gehalten haben, lassen sich diese Anwendungen „normal" lokalisieren, wie es bei anderen Programmen praktiziert wird. Nicht alle Entwickler von Maschinensoftware oder auf technische Texte spezialisierte Übersetzungsdienstleister machen von diesen Möglichkeiten Gebrauch, wenn sie vorhanden sind. Sie gehen manchmal umständliche und nicht immer sichere Wege, um die Maschinensoftware zu lokalisieren.

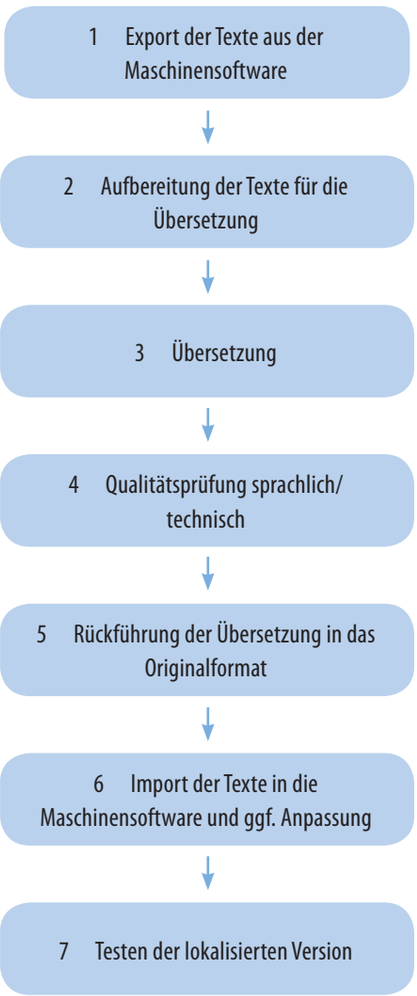
Problematischer sind die Maschinenprogramme, die in speziellen Programmiersprachen wie Siemens S7, APT, EXAPT, COMPACT oder in Hochsprachen wie C für die Programmierung von Mikrocontrollern usw. geschrieben wurden und eigene Compiler verwenden. Die Bearbeitung der zu übersetzenden Texte stellt besondere Herausforderungen an die Entwickler und an die Übersetzer gleichermaßen. Sie betreffen folgende Punkte:

- Die Texte müssen zuerst in ein für den Übersetzer verwendbares Format exportiert und nach der Übersetzung wieder in die Maschinensoftware importiert werden.
- Die Codierung von Sonderzeichen in vielen Sprachen (Spanisch/Französisch, Osteuropa, asiatische Sprachen, Arabisch …) soll unterstützt werden.
- Die festgelegte Breite des Displays lässt oft nur eine bestimmte Anzahl von Zeichen zu. Manche Applikationen verteilen sie situationsabhängig auf eine oder mehrere Zeilen. Die maximale Textlänge wird in Zeichen oder Pixel angegeben.
- Programmierer fügen in die zu übersetzenden Texte Variablen, Tastenkombinationen und Zeilenumbrüche sowie Escapezeichen ein, mit denen der Übersetzer umgehen soll. Sie soll(t)en zu den Sprachregeln der Zielsprache kompatibel sein.
- Viele Texte sind ohne Kontextinformationen oder weitere Erläuterungen kaum zu verstehen. Meistens sieht der Übersetzer das Endergebnis seiner Übersetzung nicht (z. B. alle in einem Dialog erscheinenden Texte und Objekte) und vermisst somit Kontrollmöglichkeiten für seine Arbeit.

Testlauf ist notwendig

Wie sieht der typische Ablauf eines Lokalisierungsprojekts für Maschinensoftware aus? Der Entwickler exportiert (1) zuerst den Text für den Übersetzer. Dieser Text wird vom Übersetzer bzw. von einer Agentur für die Übersetzung aufbereitet (2) (z. B. Schützen von Programmcode). Nach dem Übersetzen (3) erfolgt die Qualitätsprüfung (4), die sowohl sprachliche als auch technische Aspekte berücksichtigt. Anschließend wird die Übersetzung in das Originalformat zurückgespielt (5) und an den Entwickler geschickt, der sie in die Maschinensoftware importiert (6) und gegebenenfalls Anpassungen an den übersetzten Strings (z. B. Länge) vornimmt. Den notwendigen Schritt

des Testens (7) der lokalisierten Version der Maschinensoftware setzen leider viele Firmen nicht immer um (siehe Flowchart). Diese Schritte sollte ein Lokalisierungsprojekt umfassen:



Die meisten eingesetzten Maschinenprogramme erlauben einen Export der Oberflächentexte nach MS-Excel, CSV, XML oder in ein Textformat mit einer bestimmten Struktur (z. B. mit ID-Nr., Text in Anführungszeichen, Felder usw.). Texte, die zur Bedienung einer Maschine über einen Monitor oder einen Berührungsbildschirm dienen, laufen häufig unter der Bezeichnung „Visualisierungstexte“. In einer Idealwelt haben Entwickler die Lokalisierung der Maschinentexte von Anfang an geplant und Text und Code sauber getrennt und mit entsprechenden Zusatzinformationen versehen. Das ist die Voraussetzung für eine kostengünstige Lokalisierung. In Wirklichkeit exportieren Firmen nicht selten nur die zu übersetzenden Texte Zeile für Zeile, was dem Übersetzer die Orientierung erschwert. Einige Firmen liefern teilweise nach zeitraubenden Konvertierungen Metadaten, so dass der Übersetzer erkennen kann, was zusammengehört und um was für ein Objekt es sich handelt. Manche Programmierer sortieren ihre Softwaretexte nach Objekttyp oder nach Themen/Funktionen aus und speichern sie in getrennten Dateien oder Excel-Tabellenblättern. Das ist mit einem gewissen Arbeitsaufwand verbunden.

Meistens werden die exportierten Daten mit gängigen Übersetzungsprogrammen übersetzt, nachdem der Übersetzungsdienstleister Text und Programmcode voneinander getrennt hat. Hier ein Beispiel:
Übersetzt wird das Datenfeld <SX_Bezeichnung> mit max. 20 Zeichen.
— Das Zeichen ‘ ist ein Steuerzeichen und darf im Text nicht vorkommen!

```
INSERT INTO SX_AktionDatenNamenDef (SX_Sprache   ,
  SX_Aktion     ,
  SX_Feldname ,
  SX_Bezeichnung )
VALUES (‘D‘ , ‘678‘, ‘SX_AktionTyp‘ , ‘AktionTyp‘ )
GO

INSERT INTO SX_AktionDatenNamenDef (SX_Sprache   ,
  SX_Aktion     ,
  SX_Feldname ,
  SX_Bezeichnung )
VALUES (‘D‘ , ‘678‘, ‘SX_AktionIdx‘ , ‘AktionIdx‘ )
GO

INSERT INTO SX_AktionDatenNamenDef (SX_Sprache   ,
  SX_Aktion     ,
  SX_Feldname ,
  SX_Bezeichnung )
VALUES (‘D‘ , ‘678‘, ‚SX_ElementIdx‘ , ‚SX_ElementIdx)
GO

INSERT INTO SX_AktionDatenNamenDef (SX_Sprache   ,
  SX_Aktion     ,
  SX_Feldname ,
  SX_Bezeichnung )
VALUES (‘D‘ , ‘678‘, ‘SX_Status‘ , ‘Status )
GO

INSERT INTO SX_AktionDatenNamenDef (SX_Sprache   ,
  SX_Aktion     ,
  SX_Feldname ,
  SX_Bezeichnung )
VALUES ( ‘D‘ , ‘678‘, ‘SX_Process , ‘Process )
GO
```

Hier fängt bereits die erste Schwierigkeit an, je nachdem wie die Programmierung und der Datenexport erfolgten. Text und Code voneinander zu trennen, ist nicht immer eine triviale Aufgabe, denn man braucht einheitliche Mus-

ter, um zwischen zu übersetzendem Text und zu schützendem Programmcode zu trennen.

In manchen exportierten Dateien tauchen für den Übersetzer weitere Probleme auf. Ein klassischer Fall ist die Verteilung von ganzen Sätzen auf zwei oder mehr Zeilen, die in der Maschinensoftware getrennt gespeichert sind.

Beispiel:

21066,VERFAHRWEG DES TASTERS BIS
21067,ZUR KANTE

Da nicht alle Sprachen den gleichen Satzbau haben, führen solche Sätze zu Fehlerquellen, wenn Translation-Memory-Systeme eingesetzt werden. Dann kann es vorkommen, dass im Translation Memory nicht zueinander passende Einheiten falsch wiederverwendet werden.

Ferner steht für die Übersetzung häufig nur ein begrenzter Platz im Maschinendisplay zur Verfügung. Daraus resultieren Vorgaben zur maximalen Anzahl von Zeichen oder Pixel in einer oder mehreren Displayzeile(n). Für eine automatische Prüfung der Textlänge muss der Übersetzungsdienstleister Makros oder Skripte programmieren. Das reicht von einfachen Formeln, die die Länge eines Textes in Excel prüfen, bis hin zu komplexen Routinen, die die Breite von Texten in Abhängigkeit von Fonts und Schriftgröße in Pixel berechnen oder den noch verfügbaren Platz in einem Zweizeiler ermitteln.

Besonders problematisch wird es, wenn die Regeln zur Längenbeschränkung ständig variieren wie in diesem Beispiel einer Anweisung für Übersetzer:

Dies ist eine kurze Erklärung für Übersetzer

Für dieses Beispiel sieht ein zu übersetzender Text folgendermaßen aus:

> Hinter den Zeilen stehen ein Zeilenkürzel und eine Nummer, z. B. (ez/72)
> Das Zeilenkürzel -ez- bedeutet, dass die Zeilen einzeilig bleiben müssen.
> Das Zeilenkürzel -mz- bedeutet, dass die Zeilen auf mehrere Zeilen verteilt werden dürfen.
> Die Nummer besagt, wie viele Zeichen eine übersetzte Zeile höchstens haben darf. Die Länge betrifft die Zeichen innerhalb der Anführungszeichen (im Beispiel: 72 Zeichen).
> Die Leerzeichen innerhalb der Anführungszeichen sollten wie vorhanden beibehalten werden.

73 : „St.-Schwelle der DVR-Modulüberw. -> Konditionierung (18Zellen*2,3V)“(ez/72)

Tückische Vorgaben

Solche Situationen mit komplexen Vorgaben für Übersetzer treten ein, wenn während der Softwareentwicklung der Lokalisierungsprozess nicht bedacht wurde. Dies zu reparieren, ist lösbar, aber kostspielig, und erfordert meistens Programmierkenntnisse seitens des Übersetzungsdienstleisters. Mit Hilfe von Skripten oder Makros soll der Übersetzer oder der Lektor erkennen können, wann er die Übersetzung ändern muss, um den Vorgaben der Entwickler gerecht zu werden. Werden solche Vorgaben nicht eingehalten, bedeutet es nämlich, dass manche Texte nicht sichtbar sein können, was wiederum zu einer Fehlbedienung der Maschine/des Geräts führen kann oder eine kostspielige Korrekturrunde der Übersetzung einleitet.

Besonders tückische Platzprobleme verursacht die Vorgabe, dass bestimmte Einrückungen eingehalten werden müssen. Das ist bei älteren Maschinenprogrammen der Fall, die nichtproportionale Schriftarten einsetzen (alle Zeichen haben die gleiche Breite) und im Display Tabellenspalten durch Leerzeichen erzeugen. Aufgrund der unterschiedlichen Textlänge zwischen Sprachen ist die Lokalisierung schwer umzusetzen. Auch hier wird vom Übersetzungsdienstleister Know-how im Umgang mit Skripten und Programmierwerkzeugen abverlangt, um die Einhaltung solcher Vorgaben zu automatisieren.

Als letztes bedeutendes technisches Problem gilt die korrekte Darstellung von Sonderzeichen in den Fremdsprachen. Nicht alle Maschinenprogramme arbeiten mit Unicode. Sie sind dadurch nicht in der Lage, Doppelbytezeichen wie Chinesisch oder Arabisch darzustellen. Viele Systeme unterstützen heute zumindest die europäischen Sprachen einschließlich Russisch. Manche arbeiten aber noch mit verschiedenen Codeseiten, so dass Programmierer und Übersetzer vorab die richtige Codierung der Sonderzeichen und den Einsatz der passenden Schriftfamilie festlegen müssen.

Aus sprachlicher Sicht stellt die Lokalisierung von Maschinensoftware keine leichte Aufgabe dar. Viele Softwareentwickler exportieren schlichte Wortlisten, die der Übersetzer ohne Zusammenhang „runterübersetzen“ soll. Steht ein Wort wie *Scheibe* oder *Auflage* ganz allein, dann ähnelt das Übersetzen eher russischem Roulette. Teilweise werden Übersetzungen leichtfertig produziert und weiterverwendet. So tauchen oft in Programmen Wörter auf, die je nach Programmobjekt (Dialogtitel, Schaltfläche, Befehl) unterschiedlich zu schreiben oder abzukürzen sind oder gar unterschiedliche Bedeutungen haben. Traditionelle Übersetzungsprogramme können hier Fehler verursachen, wenn Übersetzungen aus dem Translation Memory ungeprüft übernommen werden.

Bei einigen Sprachen lässt sich die Wortart ohne Kontext nicht eindeutig ermitteln. Was ist mit dem Ausdruck *search destination* gemeint? Ist es ein Substantiv (*Suchziel*) oder ein Befehl (*Ziel suchen*)? Es wäre bereits eine erste Abhilfe, wenn die Exportdateien Informationen erhielten, aus denen der Übersetzer erkennen kann, um was für ein Objekttyp es sich handelt und welche Texte zusammengehören (z. B. über die Dialog- oder Maskennummer). Auf jeden Fall ist es wichtig und notwendig, dass Übersetzer Fragen stellen können (und auch stellen) und beim Auftraggeber einen Ansprechpartner mit guten Kenntnissen der Software und des Produkts haben, der diese Fragen kompetent beantworten kann.

Unterschiedliche Wortfolge

Einige technische Einschränkungen haben einen Einfluss auf die Formulierungen des Übersetzers. So erschwert die unterschiedliche Wortfolge der Sprachen die Übersetzung von Softwaretexten. Mitten in der zu übersetzenden Meldung kann ein Escapezeichen wie „\n“ stehen, das einen Zeilenumbruch signalisiert. Der Übersetzer muss dann abschätzen, wo er das Zeichen für den Zeilenumbruch in seiner Übersetzung korrekt setzt. Die Arbeit mit Programmvariablen, die beispielsweise als „%s“ dargestellt sind, macht das genaue Messen der Textlänge schwer. Woher soll der Übersetzer wissen, wie lang der Wert für diese Variable sein wird? In einigen Fällen hat der Programmierer – aus Unkenntnis der Regel in der Zielsprache – in fehlerhafter Weise Variablen eingesetzt. Das kann beispielsweise der Fall sein, wenn die Variable die Flexion benachbarter Wörter wie Adjektive oder die Wortstellung beeinflusst.

Aus Platzmangel sind häufig Abkürzungen oder knappe Wortaneinanderreihungen erforderlich. Wie kürzt man *Spezifischer Wechselrichterertrag* auf 10 Zeichen ab? Das führt zu Konstruktionen wie *SpezWrErtr*, die dem Bediener wie ein Buch mit sieben Siegeln vorkommen. Manche Abkürzungen haben situationsabhängig unterschiedliche Bedeutungen wie z. B. die Abkürzung *Pos*, die in einem konkreten Fall sowohl für *Positive* als auch für *Position* verwendet wurde.

In gleicher Art und Weise werden Wörter oder Befehle nebeneinander geschrieben, ohne dass der Sinn eindeutig erkennbar ist. So ist ein Ausdruck wie *X20 Zero Balance On Duration Timer Setpoint* sogar für manche Auftraggeber selbst ein Rätsel. Hier ist der Auftraggeber gut beraten, dem Übersetzungsauftrag weitere Erläuterungen oder Referenzmaterial beizufügen. Auch sind feste Regeln für die Prägung solcher Ausdrücke hilfreich, die die Erkennung, deren Einzelelemente und Aufbau ermöglichen.

Diese technischen und sprachlichen Anforderungen bedeuten für den Übersetzer ohne Programmierkenntnisse eine enorme und sehr zeitaufwändige Herausforderung, die er oft in stundenlanger manueller Arbeit erfüllt. Eine wirksame und zuverlässige technische Lösung ist mit der Entwicklung entsprechender Skripte oder Prüfroutinen möglich. Dies geht allerdings aufgrund des damit verbundenen Entwicklungsaufwands nur bei Projekten, die einen bestimmten Umfang haben bzw. bei regelmäßiger Beauftragung. Bei kleinen Softwareprojekten bleibt leider viel Handarbeit erforderlich.

Ergebnis im Zusammenhang prüfen

Sprachlich lässt sich mit den Qualitätsfunktionen von Translation Memories bzw. mit einem speziellen Qualitätssicherungsprogramm wie ErrorSpy vieles prüfen.

IT-Glossar – Definition einiger der hier verwendeten Begriffe

Assembly: Datei, die von jeder .NET-Anwendung durch den Compiler automatisch erzeugt wird. Es kann entweder eine Dynamic Link Library (DLL) oder eine ausführbare Datei (EXE) sein.
Compiler: Ein Programm, mit dem in einer Programmiersprache geschriebene Quelltextdateien in ausführbare Maschinenprogramme übersetzt werden. Synonyme von Compiler: „Übersetzer“ oder „Kompilierer“
Embedded System: Computersystem, das in Geräten, Anlagen und Maschinen eingebettet ist und spezielle Anwendungen abarbeitet.
Escapezeichen: Zeichenkombinationen, die Sonderfunktionen ausführen. Ein typischer Fall in Softwareprogrammen ist die Kombination \n für einen Zeilenumbruch.
Mikrocontroller: Mikrorechner auf einem Chip. Wird als eingebettetes System in Geräten eingebaut und übernimmt oft Steuerungs- und Kommunikationsaufgaben.
Pixel: Bildpunkt als kleinste Einheit einer digitalen Rastergrafik. Ein Zeichen benötigt je nach Schriftart und -grad unterschiedlichen Platz (Pixelbreite). Die Umrechnung von Pixel auf Zentimeter hängt von der Auflösung eines Bildes (z. B. 300 dpi) ab.
Unicode: System für die Definition der Zeichen oder Elemente aller bekannten Zeichensysteme. Diese Zeichen werden in Bereiche gruppiert. Unicode erfasst derzeit mehr als 100.000 verschiedene Zeichen. Unicode ist ein Doppelbyte-Zeichensatz, d. h. es kann 2 hoch 16 Zeichen darstellen und eignet sich daher für die Darstellung komplexer Zeichen, wie es bei den asiatischen Sprachen der Fall ist.

Damit sind – vorausgesetzt eine Terminologie wurde erstellt – die Einhaltung einer einheitlichen Terminologie und die Korrektheit der Zahlen in der Maschinensoftware zu gewährleisten.

Aber nicht alles kann man softwaregestützt prüfen. Da ein Teil der Übersetzungen ohne Kontext erfolgt, sollte man das Ergebnis nach der Kompilierung in der Fremdsprache nochmals im Zusammenhang prüfen. Die wenigsten Auftraggeber haben den gesamten Lokalisierungsprozess durchdacht und dafür ausreichend Zeit oder Budget vorgesehen. Das führt leider dazu, dass die übersetzten Texte erst dann angepasst oder moniert werden, wenn sie bereits den Abnehmer erreicht haben.

Einige Hersteller von Maschinensoftware haben diese Problematik erkannt und eigene Applikationen für den Lokalisierungsprozess entwickelt, mit denen man die Dialoge emulieren kann. Damit kann der Lektor alle übersetzten Texte sehen, die in einem Dialog erscheinen. Wenn eigens programmierte Lokalisierungsanwendungen auch zur Übersetzung der Texte dienen sollen, dann stehen sie im Leistungsumfang weit hinter professionellen Lokalisierungswerkzeugen, die über Jahre zu diesem Zweck entwickelt wurden.

Festlegung sprachlicher Regeln

Neben technischen Lösungen ist dem Auftraggeber die Festlegung sprachlicher Regeln zu empfehlen, damit knappe Meldungen im telegrafischen Stil auch in den Fremdsprachen einheitlich aufgebaut sind. So kann man Displaymeldungen wie *Betrieb RUN* mit *RUN Operating mode* oder *Operating mode RUN* übersetzen. Ähnliches gilt für Abkürzungen.

Aus all diesen Gründen kann alternativ zu traditionellen Translation-Memory-Systemen die Verwendung von Lokalisierungsprogrammen für Maschinensoftware und Embedded Systeme sinnvoll sein. Diese Lokalisierungsprogramme bieten beispielsweise folgende Funktionen an:

- Einlesen von Metadaten (z. B. max. Länge in Zeichen oder Pixel) aus den vom Entwickler generierten Dateien (CSV, TXT, XML)
- Prüfung der maximalen Anzahl von Zeichen oder Pixel einer Displayzeile
- Programmierung spezieller Prüfungen oder Dateikonvertierungen über den integrierten Makroeditor
- Bereitstellung für den Übersetzer von mitgelieferten Kontextinformationen in Form von Bildern oder Links zu externen Seiten
- Übersetzungseinheiten werden mit IDs versehen und sind dadurch für kontextabhängige Mehrfachübersetzungen gut geeignet.

Übersetzer und Entwickler planen gemeinsam

Reparieren ist immer teurer. Übersetzer und Entwickler sollten die Zusammenarbeit langfristig auslegen und gemeinsam alle Phasen der Lokalisierung einer Anwendung planen. Dabei sollten sie einige Punkte beherzigen:

- Entwickler und Übersetzer müssen Verständnis für die programmiertechnischen bzw. linguistischen Belange des anderen haben.
- Der Übersetzer braucht Hintergrundinformationen (z. B. Übersichtspläne von Anlagen, Maschinenbilder, Screenshots, Programmbeschreibung …) für ein besseres Verständnis des Zusammenhangs.
- Es sollte eine Terminologie erstellt und regelmäßig gepflegt werden.
- Die Dateien sollten möglichst weitere Informationen über Objekte, Dialog-/Maskennummern usw. enthalten.
- Nach der Übersetzung sollten ein Lektorat sowie technische Tests der kompilierten lokalisierten Software erfolgen.
- Last but not least muss der Aufwand des Dienstleisters entsprechend vergütet werden. Maschinensoftware zu lokalisieren, ist eine komplexere und dadurch zeitaufwändigere Aufgabe als die Übersetzung normaler Textdokumente. ■

François Massion

Dr. François Massion machte 1984 seinen Abschluss als Diplom-Übersetzer an der Johannes-Gutenberg-Universität in Germersheim. 1986 promovierte er an der Friedrich-Alexander-Universität in Erlangen im Fachbereich Lexikographie. Herr Massion verfügt über eine 20-jährige Erfahrung in der Übersetzungsbranche und hat seit dem Jahr 2008 einen Lehrauftrag im Fach Terminologie an der Hochschule Anhalt (FH) in Köthen. Er ist Geschäftsführer und Mitbegründer der D.O.G. Dokumentation ohne Grenzen GmbH in Leonberg bei Stuttgart.
Francois.Massion@dog-gmbh.de