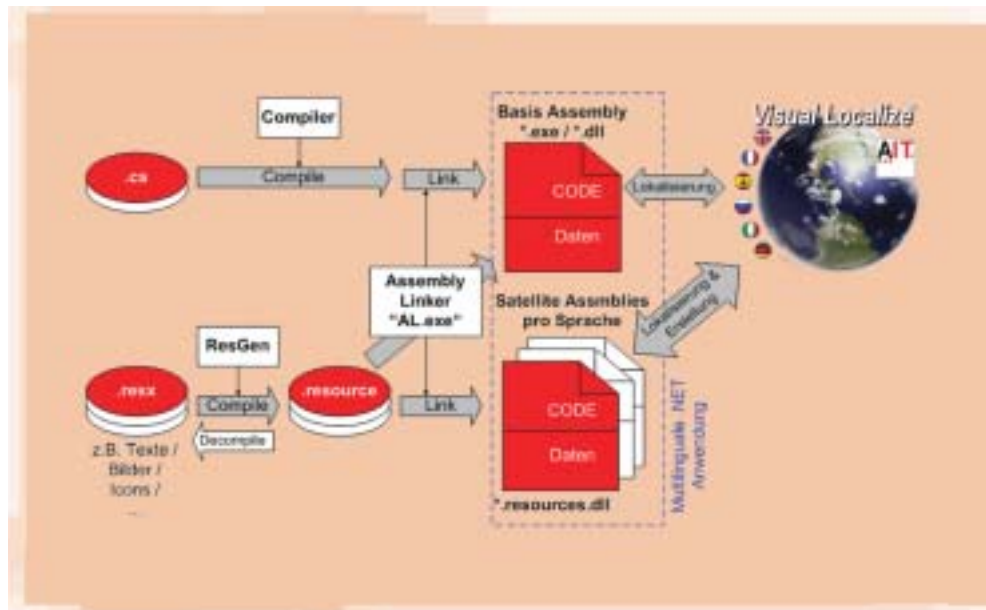


Lokalisierung von „.NET“-Anwendungen

Manuelles Übersetzen von Software in die jeweilige Landessprache ist aufwändig und fehleranfällig

Software ist heute ein wesentlicher Bestandteil von Maschinen und Anlagen. Dabei ist es inzwischen selbstverständlich, dass die Software in der Landessprache vorliegen muss, um Bedienfehler mit hohen Folgekosten bei komplexen Anwendungen wie etwa Maschinensteuerungen oder Messprogrammen zu vermeiden. Eine Voraussetzung dafür ist, dass die Bedienoberfläche der Software – das so genannte Graphical User Interface (GUI) – einheitlich gestaltet und übersetzt wurde. Das geht bei „.NET“-Anwendungen zwar auch in der Microsoft Entwicklungsumgebung, doch ist dies häufig ein fehleranfälliger und zeitraubender Prozess. Spezielle Software-Lokalisierungstools wie Visual Localize unterstützen den Anwender und „automatisieren“ viele Schritte bei der Übersetzung von Software und Updates in die Landessprache. DANIEL RISSMANN



Software-Lokalisierungstools wie Visual Localize unterstützen den Anwender und „automatisieren“ viele Schritte



DANIEL RISSMANN ist Localization Manager bei der AIT – Applied Information Technologies AG in Fellbach

KONTAKT

T +49/711/520473-10
daniel.rissmann@aitag.com

Um Missverständnisse von vornherein zu vermeiden: Unter Lokalisierung verstehen wir im Rahmen dieses Beitrags nicht das Aufspüren und Suchen, sondern das Übersetzen von Software in unterschiedliche Landessprachen unter Berücksichtigung der jeweiligen kulturellen Anpassungen. Das betrifft nicht allein die Texte von Steuerelementen in Windows-Dialogen, sondern auch das Anpassen oder das Ändern von Icons, Bildern, Position und Größe von Steuerelementen und

Dialogen. Auch die Berücksichtigung der unterschiedlichen Schreib- und Leserichtungen gehört dazu.

Ressource- oder Binärdateien übersetzen

In den bisherigen Entwicklungsumgebungen (IDEs) von Microsoft war das Lokalisieren von Anwendungen kompliziert und zeitaufwändig. Sehr häufig wurden viele einzelne Ressourcenda-

teien (*.rc) erstellt und verwaltet. Dabei mussten verschiedene Ressourcentypen wie Grafiken, Texte und Icons mit unterschiedlichen Tools generiert werden. Der Aufwand konnte explodieren, wenn die Möglichkeit einer landessprachlichen Anpassung nicht berücksichtigt wurde. Der Entwicklungs- und Übersetzungsprozess für Standard-Windows-Anwendungen ist in der Industrie heute ein wohl bekannter und verstandener Vorgang. Während einige Unternehmen es vorziehen, Ressourcdateien zu lokalisieren und ihre Übersetzungen im lesbaren Quellcode-Format vorhalten, entscheiden sich andere, auf Basis von Windows-Binärdateien (EXEs oder DLLs) zu arbeiten, um Zeit und Kosten einzusparen.

Die bedeutendsten Vorteile, die aus der Übersetzung von binären Anwendungsdateien abgeleitet werden können, sind die Verringerung der Komplexität aufgrund der deutlichen Reduktion an Dateien, die bearbeitet werden müssen, sowie die wesentlich reduzierte Fehleranfälligkeit der übersetzten Dateien. Während eine Anwendung aus mehreren hundert kleineren Ressourcdateien bestehen kann, muss der Übersetzer bei der Lokalisierung auf Binärdatei-Ebene nur mit einer einzigen, kompilierten Binärdatei arbeiten.

Im Gegensatz zu der Handhabung von Ressourcdateien muss bei der Lokalisierung auf Basis von Binärdateien der Entwickler die Software nicht erneut kompilieren. Das lauffähige Programm wird direkt aus dem Lokalisierungswerkzeug erzeugt. Der Entwickler erspart sich in der Regel Eingriffe in den Lokalisierungsprozess, wodurch viel Aufwand und Entwicklungskosten gespart werden können.

Workflow bei „.NET“-Lokalisierungen

Mit der Entwicklung des „.NET“-Frameworks von Microsoft wurde ein neuer Meilenstein in der Erstellung von mehrsprachigen Softwareanwendungen mittels C oder VB „.NET“ gelegt. Anwendungsressourcen werden im Gegensatz zur früheren Entwicklung mit MS Visual C++ in einem neuen Dateityp, der so genannten .RESX-Datei definiert. Diese Dateien sind XML-Container, welche die Bestandteile der Anwendungsoberfläche wie Dialoge, Menüs und Strings beschreiben. Es existiert immer eine .RESX-Datei für jeden Ressourcentyp (Dialog, Texte, Menüs, Icons,...). Um „embedded“, also eingebundene Ressourcen-Dateien zu erstellen, geht die Entwicklungsumgebung (IDE) „Microsoft Visual Studio .NET“ in zwei Schritten vor:

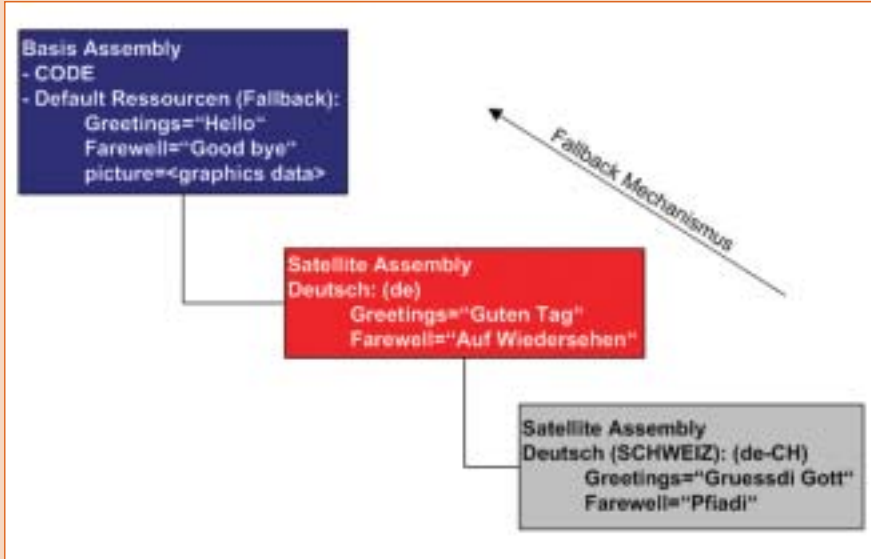
Eine .RESX-Datei, welche die XML-basierende Beschreibung einer Anwendungs-Ressource enthält, wird mittels des Microsoft „ResGen“-Werkzeugs in eine Ressourcdatei kompiliert. Dieses Tool wird als Teil des

„.NET“-Framework SDK ausgeliefert. Die erzeugte Ressourcdatei enthält ausschließlich binäre Ressourcen. Die so erstellte Ressourcdatei wird in einem zweiten Schritt in ein Assembly eingebaut, also „embedded“. Ein Assembly ist die technische Beschreibung von Microsoft, um eine Kombination aus Anwendungscode und Anwendungsressource zu beschreiben. Diese werden verbunden, um eine einzelne „.NET“-Anwendung zu dokumentieren. Hierzu wird entweder das Werkzeug „Assembly Linker“ oder ein entsprechender Sprach-Compiler verwendet.

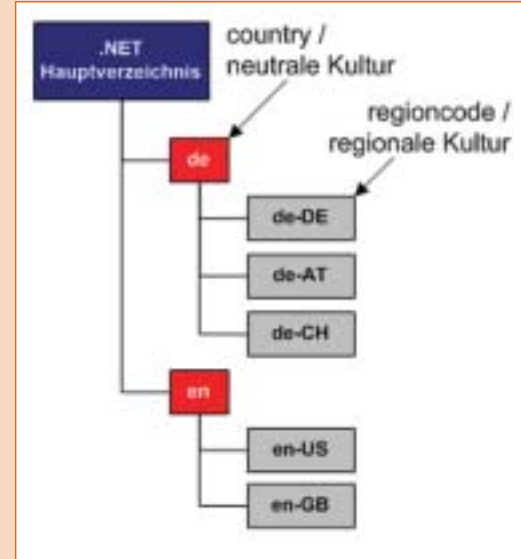
Die „Kultur“ der „.NET“-Anwendung stellt regional-spezifische Informationen einschließlich des verwendeten Schreibsystems, des Sortierverfahrens, des Datumsformats, der verwendeten Währung, Zeitformate, Postleitzahlenformate und vieles andere dar. Das „.NET“-Framework bietet hierzu eine Culture-Info-Klasse, um einen einfachen programmatischen Zugriff auf die einzelnen Sprachsysteme zu haben. Innerhalb dieser Klasse bestimmt das „UICulture data member“, wie die Anwendungsressourcen geladen werden. Das data member spezifiziert somit die Sprache der Bedienoberfläche für die „.NET“-Anwendung. Durch den Wechsel des data member kann der Entwickler die Bedienoberfläche von seiner „.NET“-Anwendung beliebig umschalten.

Sobald die UICulture definiert wurde, findet und lädt die „.NET“-Anwendung die gewünschte Zielsprache selbst und zeigt die Bedienoberfläche in dieser Sprache an. Die sprachabhängigen Dateien für die Bedienoberfläche sind bei einer „.NET“-Anwendung in so genannten Satellite Resource DLLs (*.resources.dll) enthalten und werden von der Anwendung automatisch geladen. Sie sind in einer vordefinierten Verzeichnisstruktur des Hauptverzeichnisses abgelegt, von der aus die „.NET“-Anwendung gestartet wird. Durch das Festlegen dieser Verzeichnisstruktur wurde ein Mechanismus entwickelt, um mehrsprachige Applikationen zu erhalten, die sich auf einem Server befinden.

Die Bedienoberfläche der Anwendung wird zur Laufzeit eingestellt, wenn die „.NET“-Anwendung unter Berücksichtigung der zuvor gewählten UICulture eines Clients die Sprachinformationen anfordert. Mit diesem Mechanismus ist es möglich, dass eine Anwendung auf dem Server abgelegt ist, während mehrere Clients verschiedene und voneinander unabhängige Sprachen und Bedienoberflächen darstellen. Wenn eine „.NET“-Anwendung ein sprachabhängiges Ressourcenset anfordert, welches nicht verfügbar ist, tritt ein Fallback-Mechanismus in Kraft. Die Abfrage wird hierbei von unten nach oben geleitet. Erst wenn in höchster Hierarchieebene – im Basis Assembly – keine Ressource gefunden wird, spricht das Ausnahme-Handling an. ►



Fehlt ein angefordertes sprachabhängiges Ressourcenset, sorgt der Fallback-Mechanismus dafür, dass die nächsthöhere Assembly genutzt wird



Der SatelliteRegionCode enthält spezifische Informationen zu den jeweiligen Sprachregionen

Übersetzung in der Microsoft Entwicklungsumgebung

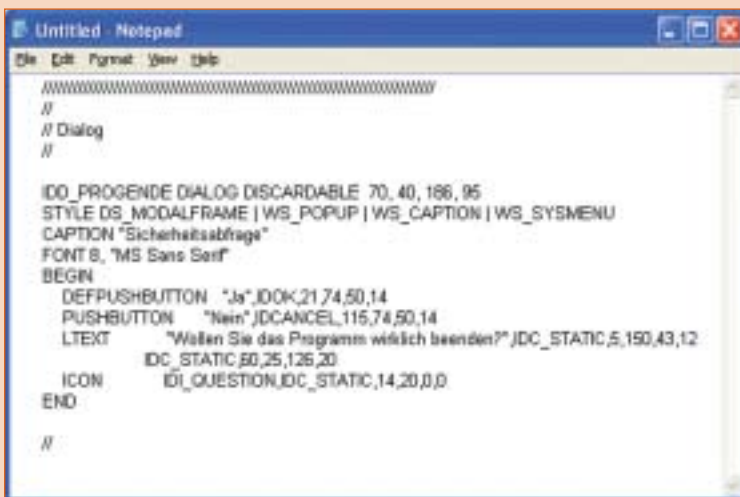
Ohne Zweifel stellt die „.NET“-Technologie eine effektive Art und Weise dar, um mehrsprachige Anwendungen in der Design-Phase zu generieren. Microsoft stellt eine Entwicklungsumgebung (IDE) bereit, die Werkzeuge für die Lokalisierung von „.NET“-Anwendungen enthält. Leider wird der Lokalisierungsprozess innerhalb der IDE schnell unhandlich, wenn viele Formulare (WinForms) mit zahlreichen Steuerelementen in verschiedenen Landessprachen benötigt werden. Der Aufwand, eine größere Anwendung auf diese Weise zu lokalisieren, ist sehr hoch, denn die Entwickler werden mit der Verwaltung von .RESX-Dateien und Assemblies belastet und so von ihren eigentlichen Aufgaben abgehalten. Bei der Lokalisierung innerhalb der Entwicklungsumgebung muss der Anwender mit einer

Reihe von Einschränkungen rechnen. Da ist zunächst der komplexe Lokalisierungsprozess: Seitdem jede WinForm eine individuelle .RESX-Datei und jede Zielsprache eine Variante von dieser Datei benötigt, wird das Verwalten von großen Anwendungen mit einer Vielzahl an WinForms und Sprachen extrem aufwändig. In diesem Zusammenhang muss ebenfalls der Entwicklungs-Overhead bei der Pflege von mehreren Sprachvarianten berücksichtigt werden. Zudem können inkrementelle Ressourcendateien nicht direkt übersetzt werden, weil die RESX-Sprachdateien und die mehrsprachigen Satellite Assemblies, die von einer „.NET“-Applikation benutzt werden, keine vollständigen Kopien der kompletten Basis-Ressource sind. Mangels vollständigem Inhalt und damit Kontext ist es sehr schwer, die Inhalte korrekt zu übersetzen. Diese Dateien sind im Hinblick auf einen brauchbaren Lokalisierungsprozess eher hinderlich.

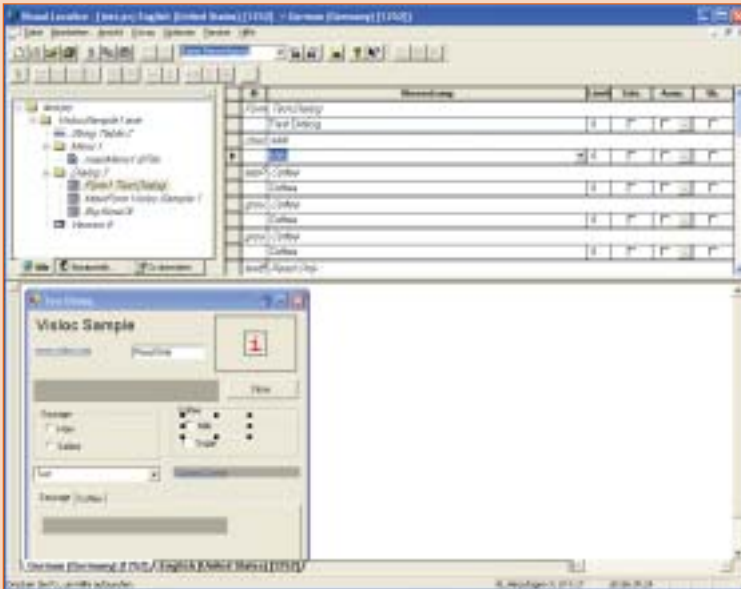
Ein weiteres Problem ist die fehlende Versionskontrolle, die von der Entwicklungsumgebung Microsoft Visual Studio „.NET“ nicht zur Verfügung gestellt wird. Um Änderungen des Basisprojekts für die einzelnen Übersetzungen und deren Sprachlayouts in den .RESX-Sprachvarianten zu überwachen, muss diese Kontrolle manuell erfolgen – was ein hohes Fehlerrisiko birgt. Gleiches gilt für Änderungen der Ausgangsdateien: Auch hier müssen Entwickler alle Änderungen in der Basis der Anwendungs-Ressource manuell nacharbeiten, da diese nicht automatisch in den bereits übersetzten Dateivarianten (pro Sprache / pro Version) übernommen werden.

Lokalisierung mit Übersetzungssoftware

Bequemer und fehlerfreier lassen sich „.NET“-Anwendungen mit Hilfe externer Übersetzungswerkzeuge wie beispielsweise „Visual Localize“ von AIT durchführen. Mit der Entwicklung der „Visual Localize-.NET“-Edition wurde ein Durchbruch in der Lokalisierung von „.NET“-Anwendungen erreicht. Die einfach zu bedienende WYSIWYG-Benutzeroberfläche unterstützt den Anwender in allen Phasen der Erstellung lokalisierter Software. Es stellt durch die praxisorientierte Vorgehensweise eine deutliche Vereinfachung bei der Übersetzung und Anpassung von „.NET“-Anwendungen dar. Alle wesentlichen Nachteile, die im Lokalisierungsprozess mittels der Entwicklungsumgebung entstehen, werden mit einem Software-Lokalisierungstool gelöst. Wie bereits erwähnt, benötigt jeder Ressourcentyp innerhalb einer Anwendung eine einzelne .RESX-Datei. Sobald Updates der „.NET“-Anwendung veröffentlicht werden, eskaliert der Verwaltungsaufwand für mehrere



Aufwändig: Bisher wurden in Microsoft Entwicklungsumgebungen meist viele einzelne Ressourcendateien (*.rc) erstellt und verwaltet



Lokalisierung einer „.NET“-Anwendung im WYSIWYG-Editor von Visual Localize

Versionen und Sprachen. Um diesem Problem Herr zu werden, verwendet Visual Localize die kompilierte „.NET“-Anwendung und bearbeitet diese direkt. Das Lokalisierungstool verwaltet in einer einzelnen Projektdatei alle kompilierten „.NET“-Assemblies und andere Binärdateien (16- und 32-Bit-EXE, DLL, OCX). Nach der Übersetzung in eine beliebige Landessprache und der Layoutanpassung der Bedienoberfläche können auf Knopfdruck die fertigen Zieldateien (EXEs, DLLs) bzw. „.NET“-Satellite Assemblies (*.resource.DLLs) erstellt werden. Für die Übersetzung und Anpassung des Dialoglayouts sind jetzt keine Programmierkenntnisse mehr notwendig. Mit Hilfe von Visual Localize können sogar alte Übersetzungen aus herkömmlichen Windows-Applikationen in den „.NET“-Assemblies wieder verwendet werden – ohne großen Aufwand.

Das Übersetzen, Testen und Anpassen von „.NET“-Binärdateien entspricht dabei exakt dem Bearbeiten von .RESX-Dateien innerhalb der IDE, allerdings mit reduziertem Verwaltungsaufwand und erhöhtem Wiederverwendungsgrad der Übersetzungen. Da alle Dateien einfach und direkt bearbeitet werden können sowie keine Kompilierungsschritte mehr notwendig sind (das mehrmalige Hin- und Herschicken der Dateien zwischen Entwickler und Übersetzer entfällt ebenso), reduzieren sich die Projektdauer und die damit verbundenen Lokalisierungskosten. Visual Localize kann darüber hinaus aus einem „.NET“-Assembly (EXE/DLL), welches alle Informationen der Anwendung enthält, selbstständig sprachspezifische Satellite Assemblies erstellen. Ein solches Vorgehen entspricht dem tatsächlichen Übersetzungsprozess, da der Softwarehersteller zum Zeitpunkt der Entwicklung meist noch nicht weiß, für welche zusätzlichen Landessprachen das Produkt später lokalisiert werden soll.

Nach einem Update der zu lokalisierenden „.NET“-Software erkennt Visual Localize automatisch alle Änderungen und Neuerungen. Dadurch wird nur die Differenz zur Vorgängerversion übersetzt und an die Zielsprache angepasst. Durch diese integrierte Versionskontrolle lassen sich mehrere Programmversionen und -sprachen komfortabel verwalten und Übersetzungskosten für Folgeversionen einsparen. Einer schnellen Markteinführung beziehungsweise Updates neuer Software steht damit nichts mehr im Wege.

Fazit

Die Übersetzung von „.NET“-Anwendungen in die jeweilige Landessprache allein auf Basis der Microsoft Entwicklungsumgebung ist ein zeitraubender und fehleranfälliger Prozess. Software-Lokalisierungstools wie Visual Localize unterstützen den Anwender und „automatisieren“ viele Schritte. Zahlreiche Industrieanwender wie etwa Siemens, Robert Bosch, Schüco, Honeywell, Foss etc. nutzen bereits die Übersetzungslösung von AIT. Die zu übersetzende Software wird in diesen Unternehmen als alleinstehendes Produkt oder als Ergänzungsprodukt für eine komplette Anlage (z. B. Steuerungssystem für eine Fertigungsstraße, für eine speziell entwickelte Anlage in der Automobilindustrie oder ein Überwachungssystem) verwendet und an Niederlassungen bzw. Kunden in aller Welt verkauft. ■

Dieser Beitrag als PDF und weiterführende Informationen (ähnliche Beiträge, technische Daten, Direktlinks zum Hersteller etc.) sind online verfügbar auf www.aud24.net

more @ click AD034401 >