DEV-14: Ready to Translate Your Application?

Martin Wolf, VP Development & Technology, proALPHA Software AG
Gunnar Schug, Group Development Manager, proALPHA Software AG

Agenda

proALPHA – Company & Product

Translation and Internationalization

- Overview

Progress® Features to prepare ABL-Code for Translation

Preparing the Application for Translation

- Just adding String-Attributes?

Managing Terminology as Key to efficient Translation

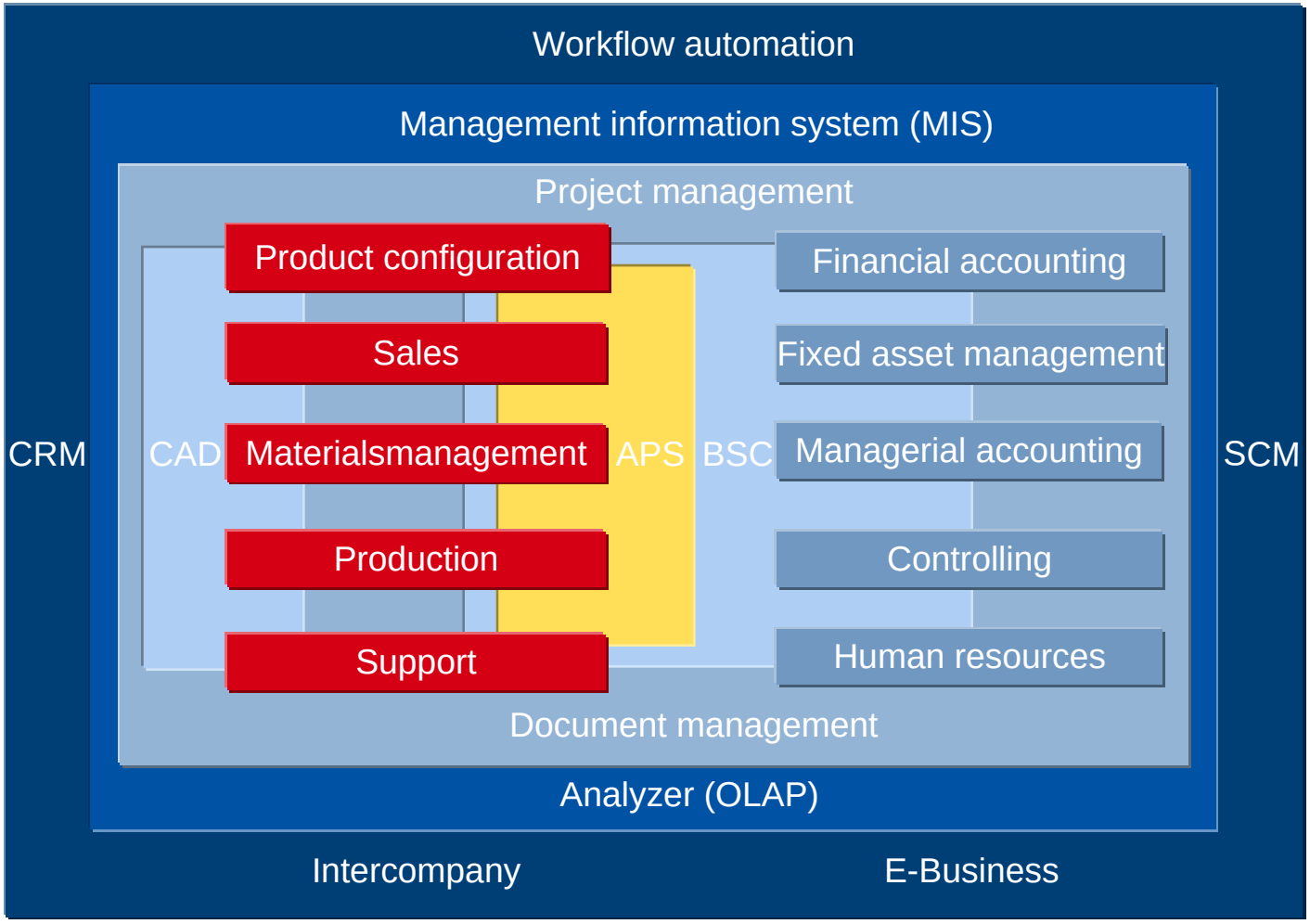
proALPHA Translation Tools and Process in Action

Summary

Q & A

proALPHA – Company and Product

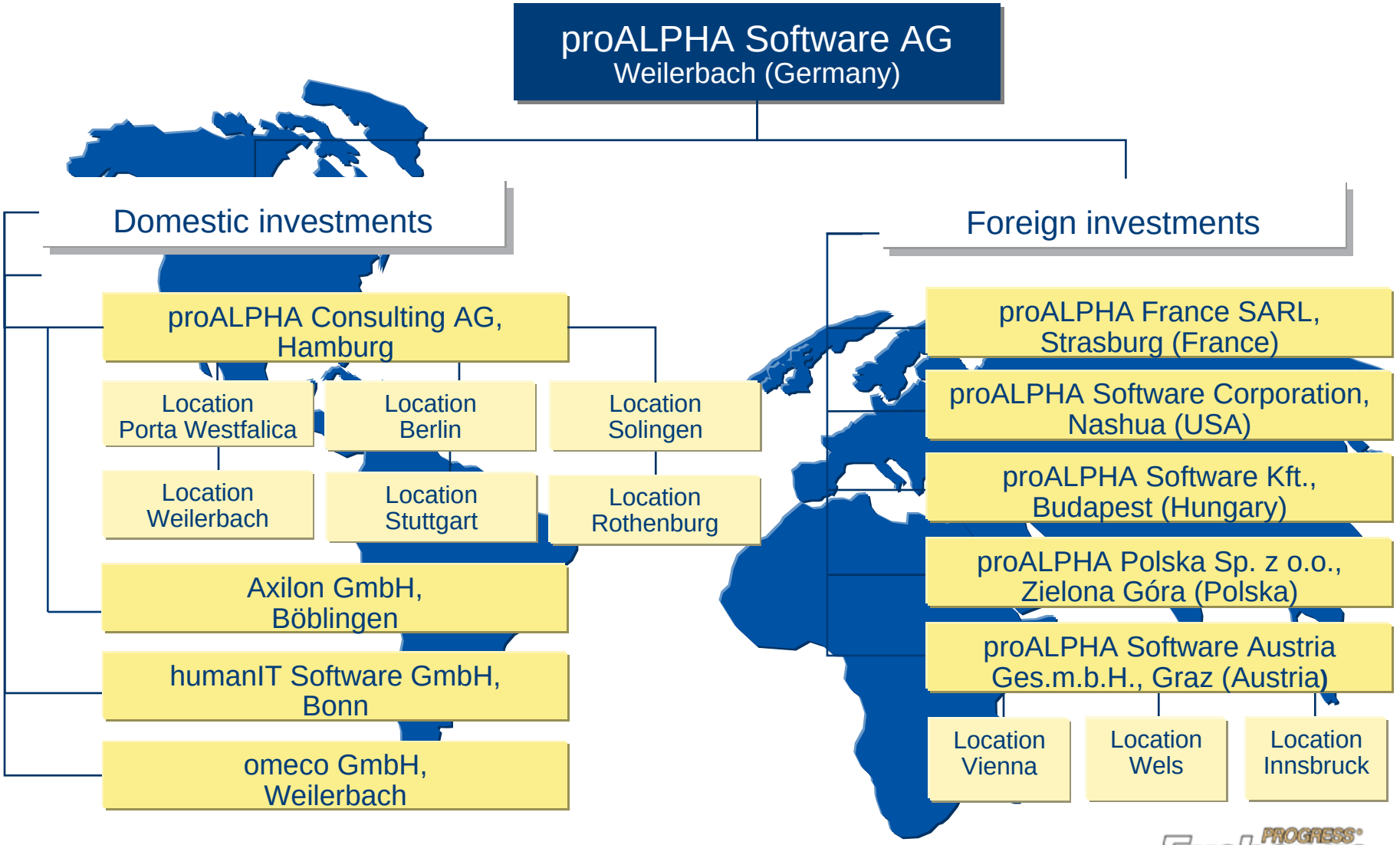
proALPHA® Complete Solution



proALPHA - Partner for medium-sized business

Product:	proALPHA® standard ERP software	
Services:	IT consulting, project management, implementation, seminars, maintenance, support, and hotline	
Target market:	Medium-sized industrial and trade companies	
Business numbers:	Customer Base	> 1,500
	Sales 2006/2007	41,0 mill. €
	EBIT	6,8 mill. €
	Employees (as of 01/2008)	> 400
Managing board:	Leo Ernst	Commercial Management
	Werner Ernst	Technical Management
Shareholders:	Employees	59,8 %
	Venture Capital	27,6 %
	Others	10,7 %
	Company	1,9%

The proALPHA Group



Customer-Oriented Organization

Strategic product development!

proALPHA Software AG
Central services

- Standard development
- Marketing
- Documentation
- Academy

- High innovative power
- Technologically and functionally leading software
- Professional training and documentation

Competence close to you!

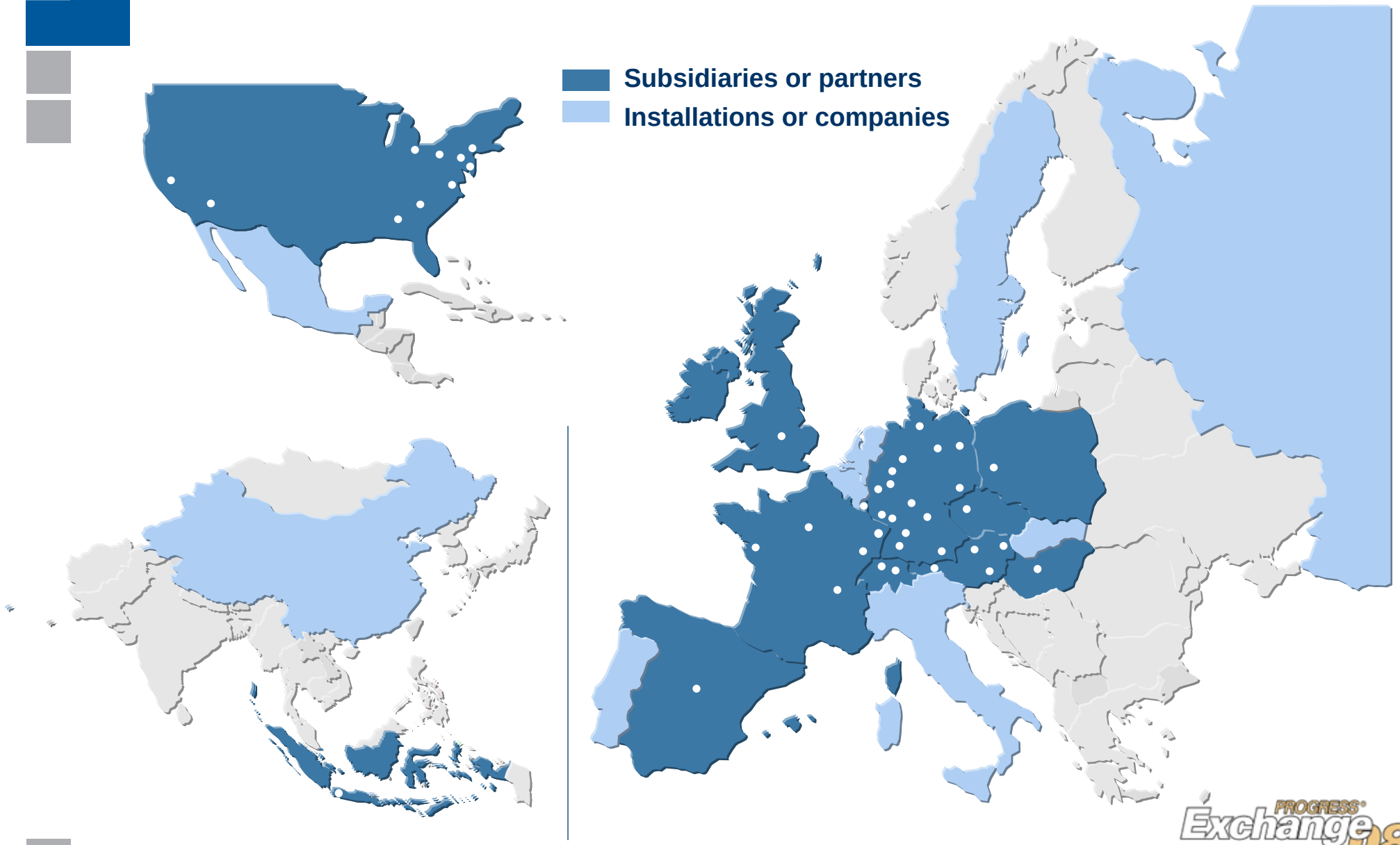
proALPHA computer retailers

- Customer projects

- Sales
- Project management
- Customizing
- Modification
- Training, support, hotline

- Optimal customer orientation
- Short introduction times
- Personal support

International Presence



Translation and Internationalization

Overview

Definitions

Internationalization

- Translation
- Localization
  - Fulfilling local legal Requirements
  - Taking into account local „common sense“

We do not talk about localization in detail, but you should worry about

- Validate whatever you are told
  - Is it really true or what your dialog partner expects to be true?
- Expect the unbelievable
  - Governments are genius in creating rules hard to implement

Translation

Translation includes

- Translation of the User-Interface
- Translation of the (Online-)Documentation
- Translation of numerous documents related to setup and operate the system
  - Customization Guide
  - Setup Guide
  - Release Notes

We will primary focus on the UI

- ABL UI
- Repository Data
- Other components integrated into the UI

Costs of translation (real data from current version)

User Interface

- Core UI (ABL) — 47 000 words
- Repository — 250 000 words
- Other Components — 18 000 words

- Total Costs — 44 000€

Online-Documentation

- Overall — 1 560 000 words

- Total Costs — 218 000€

Progress Features to prepare ABL-Code for Translation

String markup with string-attributes

Information for translation is added with string-attributes

Justification

- **L – eft**
- **R – ight**
- **C - enter**
- **T – trim**
- **U - ntranslatable**

Max length

- **Length [1 .. 999]**

Example

- **Label ‘Item Number‘:R18**

Assumptions about string attributes

Add string attributes to any string

Without attributes, length is limited to original length

Adding :U to untranslatable strings is absolutely necessary

- **Translating strings internally used to control program flow will cause errors**
- **Change is made by translator, invisible to the developer!**

Translation process (ABL UI)

Strings are extracted from code using the compiler

Translatable strings are imported to a Translation Manager (TranMan) database

Strings are translated using tools like Visual Translator

Compilation is done against a TranMan database containing

- **Original strings**
- **Translations in one ore more target languages**

Compiler creates r-code with one text-segment per required language

- **Missing translations use original strings**
- **At runtime, text-segment matching sessions current-language setting is loaded into execution buffer**

Demo: String-Attributes

Preparing the Application for Translation

Just Adding String-Attributes?

UI-Standards required

In order to be ready for translation

- Screen layout must take into account enlargement of labels during translation
- Best solution is to define a standard length for all side-labels
- And create all screens to be able to display objects of this size without overlapping

We have chosen

- :R18 for side-labels
- :L18 for toggle-box labels
- :T40 for menu-items
- :T65 for titles
- :T80 for tooltips

The story of column-labels

Approach #1:

Length = max(field-format,30)

- Some languages have letters requiring two character positions
- Single character shortcuts are not self-explaining

Approach #2:

Length = next even number max(2,field-format,30)

- Even 2 letter shortcuts are not very self-explaining
- Chinese does not have shortcuts at all and each symbol requires 2 character positions

Approach #3:

Length = next even number max(6,field-format,30)

- This gave us self-explaining column-labels and enough room for Chinese symbols

Pitfalls

Persistent procedures running on a stateless AppServer™

- Decision about language made at startup
- Serving clients using different languages does not work

Bytes and character positions required do not match for multi-byte characters

- Max-length defines # of bytes within text-segment
- Using multi-byte characters (ex German Umlauts with UNICODE) can result in compiler-warnings like
  - **Cannot fit FILL-IN Straße/Hausnummer with COLON-ALIGNED within FRAME F-Main. (4027)
  - Ignoring position info for COLON-ALIGNED (2054)
- Startup-parameter –ulayout solves the problem
  - Reserved bytes = max-length * value of –ulayout
- Strings can be truncated in UI depending on content!

And what about the repository?

Repository based applications hold more text in the database than in ABL

There are no tools available to support that translation process

Also a strategy to store data for different languages is required

Our solution

- **1:n relation between base-table and any component to translate**
- **Translation to any target-language is not required to run the application**
- **Missing translations are replaced by string in default-language**
- **Integrated toolset to support translation of ABL UI and repository data**

Don‘t forget Codepages

8bit codepages have many limitations

- No single codepage availabel for Western & Eastern Europe

Unicode is the best choice to overcome those limitations

- Translation data
- Repository data
- Source code itself can use 8bit codepage

Unicode support within the Progress® world is great, but

- Third-party components like OCX‘s need to be checked
- Interaction with the OS like printing might be strange

Demo: proALPHA Translation Tools

Managing Terminology is Key to Efficient Translation

Terminology management

Baseline of terminology management is to guarantee the rules

- **One word – one meaning**
- **One meaning – one word**
- **Translator normally does not have the context, so non-unique terms will result in invalid translations**

In its extended version, it also covers consistent phrasing

Terminology management II

Terminology management requires specialists and processes

- **Developers do not like it!**

Consistent terminology can reduce costs dramatically

- **Reduced volume**
- **Automated pre-translation**

Un-translated version has a benefit of being more consistent and usable

Some examples

German word „Anlage“

- #1 Asset
- #2 Date of Creation

- Solution: Use „Anlagegut“ instead of „Anlage“ for „Asset“

Different words for same object

- Item
- Part
- Item-Number
- Item-No
- …

- Solution: define one and only one standard

Steps to go through

Scan current terminology for unexpected differences

Scan also for identical words with different meaning

- Compiler String-XRef gives a complete view of strings used within the ABL UI

Define standards

- Common shortcuts, ex „Desc“ for „Description“
- Common extensions, ex „(*extension*)“
  - Name (Customer)
  - Name (Supplier)
  - Zip (Billing Address)
  - Zip (Shipping Address)

Steps to go through

Setup a terminology database

- Management of terms, shortcuts and description of terms
- Robust foundation for pre-translation

Setup a terminology management process

- The terminology management process requires skills typically not available in development
- Identify new terms ASAP
- Give developers hints to already defined terms

proALPHA Translation Tools and Process in Action

proALPHA Translation Tools and Process in Action

Summary

Translating an application is a complex and costly project

It requires a lot of preparation and project management

Standards for screen layout are a must

Knowledge about potential target languages is required

- **Chinese or Arabian are very different**

Summary

A proper terminology management is a key factor for success

Local version will benefit from clean terminology too

Finally:

Translation is only one part of Internationalization

Questions

Thank You