# Improved Usability through Internationalization

Carsten Witte

Siemens AG, Healthcare Sector
Magnetic Resonance Division,
Allee am Röthelheimpark 2, 91052 Erlangen, Germany
carsten.witte@siemens.com

**Abstract.** This article is a field report about usability and internationalization based on the example of Japan. It will explain, based on examples, how proper internationalization improves the usability of software.

**Keywords:** software, internationalization, I18N, localization, L10N, globalization, G11N, Japan, Japanization, J10N, language, usability.

## 1 Introduction

Over the last years Siemens and many other big software and hardware vendors in the life-science industry have focused more and more on workflow in order to support the customer. Solutions weren't centered on single, selective tasks anymore, but on complete chains of tasks. The process of scanning medical images, processing and reading them and finally writing a report of the findings is now seen as one holistic clinical workflow, instead of the former single workstations that didn't know anything about the next step.

I have been working for the Magnetic Resonance division of Siemens Healthcare since 2001. I also have an extensive background in user-interface design. Five years ago, I had the opportunity to visit Japan several times to analyze and learn about the differences between Japanese, American and European methods of operation. At that time, I was responsible for the native language support of our software, too. In a meeting with the then-chairman of the Regional Unit Japan, Bernd Rennebaum, we spoke about these topics: the customer's workflow, supporting Japanese in our software and how to reach an even higher level of usability. And at this meeting, he coined one sentence that ever since became sort of a mantra for my daily work:

> *"How can we better support the workflow of our customers than providing the software in their own language?"*
> *Bernd Rennebaum, Siemens AG, 2006*

This focuses on various issues of usability that need to be taken into account: readability, accessibility, comprehensibility and a continuous flow of reading over all devices in daily life.

A. Marcus (Ed.): Design, User Experience, and Usability, Pt I, HCII 2011, LNCS 6769, pp. 111–116, 2011.

## 2 Why Localize?

Similar to Germany, in Japan, English is the first foreign language taught in every school. Although some Japanese hesitate to actually speak English, the majority read and writes it fluently. So, why should we translate our software at all?

> *"If I am selling to you, I speak your language.*
> *If I am buying, dann müssen Sie Deutsch sprechen."*
> *Willy Brand, 1913-1992, German Chancellor*

Japan is the one of the biggest markets for magnetic resonance devices, with almost the same size as the whole of Europe combined. So, a healthcare-provider will want its share of that market. And when you enter a clinic in Japan, you're of course greeted in Japanese, both spoken and written. All patient-information is in Japanese. Every workstation speaks Japanese: from the registration desk, over to the radiology counter, the scheduler, and even the filming device. Now imagine your device in that consistent Japanese world… displaying English only. Soon this device will be considered an outsider, an obstacle in an otherwise seamless workflow. This is something you really want to avoid. You will want to allow the user to stay in one language, their native language, without always having to translate in the back of their heads, without recalculating calendars and measurements. You will want to enable the users to read their language and use their keyboards. Aside from all valid business figures, you will want to…

> *"Show respect for you customer's linguistic and cultural needs."*
> *Bill Hall, MLM Associates, Inc., 2006*

By increasing the usability of software, you're increasing the customer's satisfaction, the loyalty of the customer base and ultimately your sales figures.

### 2.1 How to Localize?

Internationalization and localization as part of globalization are both easy and very, very complicated. It's easy, because you simply have to follow a basic set of rules in order to ready your software for the international market - especially if you start your project in today's software-landscape based on Unicode. Older systems actually had to deal with a huge load of code-pages, font-problems and conversion issues.

First and foremost you have to separate the code itself from all user-visible text throughout the whole software to enable two colleagues to work independently on the project without too many interference: the software-developer and the translator. Both get their set of files and you can compile and translate independently.

This way, you have to compile the project only once and create the product by combining one constant code-base with the language of choice. This approach saves costs all along the product cycle: development, integration, test, localization, and shipping.

At Siemens Healthcare MR, we provide the user-interface in six standard-languages: English, German, French, Spanish, Chinese, and Japanese; and the user-documentation in up to 30 languages.

### 2.2 What Is So Special about It?

Second, you have to make sure that everything cultural isn't hardcoded anymore either. This includes grammar, calendars, measurement systems, and in rare cases even the color scheme. Below, I will illustrate some major issues:

**Context.** Software developers are trained to break up larger, complex modules into smaller functional packages. Sometimes this splits sentences into multiple parts, too. When, later on, a translator sees these parts without knowing the context, he might translate the wrong meaning. For example "back" can mean going back to the previous page, or it could mean the rear part of the human body. Mixing these up, the result might be a good laugh if you're lucky or a misdiagnosis if you're not. The solution is to keep sentences together and give the translator a chance to do their work correctly.

**Grammar.** Although some languages share rules on how to order words of a sentence, or how a plural is created, there are even more substantial differences. Therefore it is unlikely, that you can come up with a one-size-fits-all language-independent grammar-creator. You have to actually implement for the target language. For example Japanese ordinal-postfixes vary not only on the number, but also on the size and shape of the counted object. There are languages, where the plural ending is based on whether the quantity itself is even or odd. German features five cases, Russian six.

In order to address these issues early, anchor the target languages into the requirements of your software and involve the translation team very early along the development process.

**Numbers. W**hile in English and German the order and listing of numerations is nearly identical, Japanese differs completely. English list the partial number first: "7 of 42" Japanese starts with the whole number: 「４２の７」 While English places words in front of the numbers of ranges: "from 7 to 42" Japanese features postfixes 「７から４２まで」. So do not try to arrange the words in the software, but create placeholders which can be reordered as needed by the translator.

**Sort Order.** Sort order and comparison are very language dependent. While German is sorted by dictionary or by phone-book, Spanish knows a traditional and a modern sort-order. While Taiwan sorts primarily by stroke-count, mainland China prefers pronunciation. Japan then sorts by either XJIS or by Unicode, and sometimes by radical. But there is a good chance that the operating system (Windows, Linux, Mac-OS) already features complex sort-order-algorithms, so you have a good chance to be on the safe side by simply not trying to roll your own.

On that note: today most operating-systems have undergone an extensive test in all sorts of target languages and mostly fulfill not only language, but also even legal requirements of many countries, for example GB18030 for China. It improves the

consistency of usability not to deviate from these standards too much. Stick with the common user-interface. And of course, reusing existing, tested controls will reduce development costs dramatically.

**Names.** People do not only cherish their own name, it is of vital importance to recognize a person correctly, especially if it comes to medical treatment. Because of this importance, the software shall display the name precisely all the time, taking into account the local differences for various countries.

Japanese names can have three equivalent name-representations in parallel. These are written in three distinct writing systems: kanji, kana—katakana or hiragana—which are a phonetic rendering of the kanji, and in a romanization, called romanji. Since there is not one-to-one relationship between kana and kanji, similar kana may represent different names. Therefore, you need to display all available name-representations when applicable. For example, the name "Miki" written in katakana as "ミキ" has about 70 distinct kanji-representations.

**Calendars.** Different calendars and date-representation in general, is another topic already covered by most operating-systems. Not every country is based on the Gregorian calendar. In Japan the emperor-calendar is used, especially for official documents. In that calendar "November, 29th 1966" is represented by「昭和41年11月29日」.

And there are many, many more issues to look out for: the right-to-left reading order for Arabic and Hebrew or compound letters in the Indic derivates. Parsing strings can be difficult when it comes to different white-spaces and abbreviations. Parsing numbers might be impossible, because a character might only look like a number, but actually it isn't one. Or it doesn't look like one at all, but actually is.

Re-casing of words might mangle the meaning. The most prominent example being: élève → Elève → ELEVE→ eleve. Sometimes different Unicode points are depicting the very same character, almost not distinguishable to the human eye, thereby threatening the safety of internet-addresses, for example. Usability also means helping the user to avoid errors and enabling them to use the software without having to worry all the time.

## 3 Usability Testing

One of the most important tasks is to test and verify functionality and translation with the target-user-group. The issue with the Japanese names for example came up at a usability-session with customers at large university-hospitals in Tokyo. We discussed the problem, its background and customer wishes, provided prototypes of possible solutions and the best fitting version for both customer and vendor was chosen. This procedure achieved a technical feasible, cost-effective solution that really solved the problem at hand.

When you compare the following two screenshots, you will recognize that they not simply translated into their target-languages, but also handle different number- and measurements-systems. Yet, the key-feature is the sub-dialog in the Japanese version that enables the input of the three name-representations.

Both dialogs provide specialized usability to different groups of users, and these groups only, making localization an enabler to usability. If mixed up, the English

version would lack important functionality for Japan and the Japanese version would make entering names in England very, very complicated.

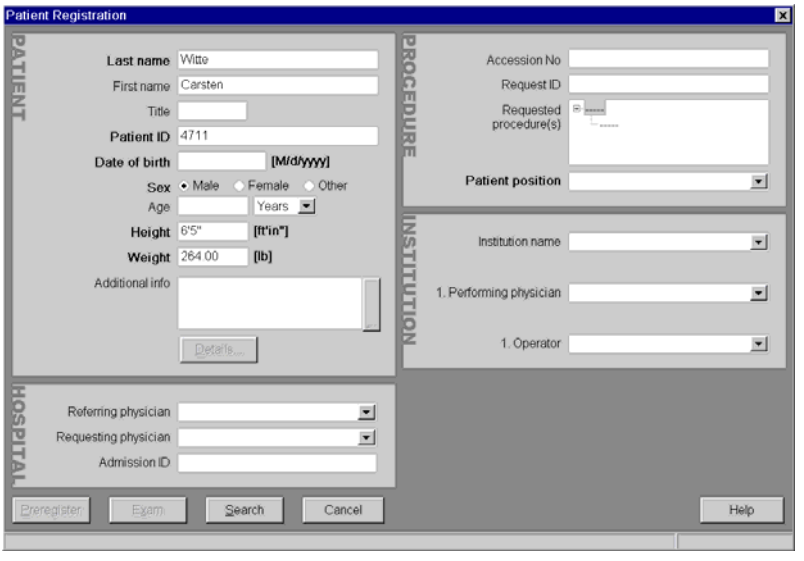


**Fig. 1.** English dialog for patient registration

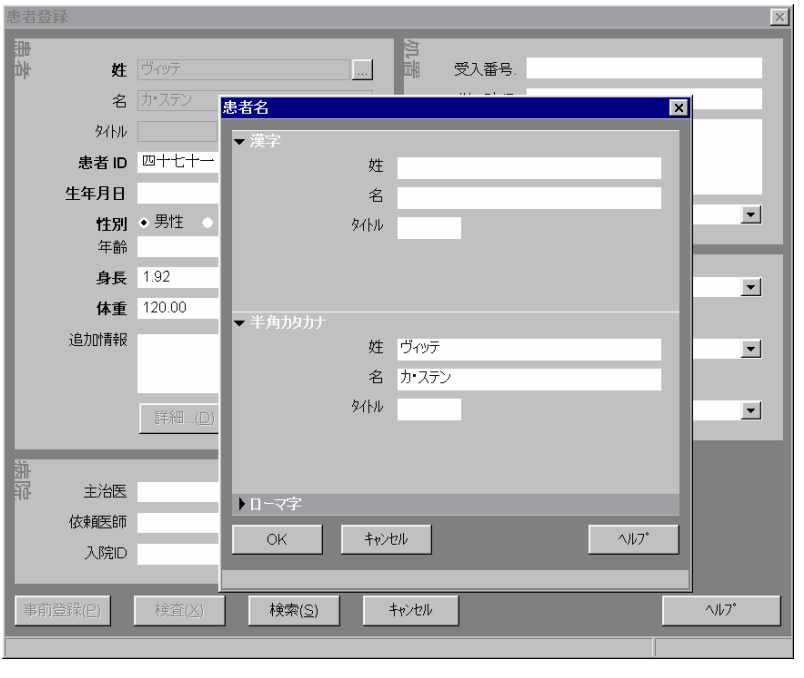


**Fig. 2.** Japanese dialog for patient registration

There are many tools and techniques to constantly test the localization of your product. Especially the screen-layout should be monitored regularly, because it might break when text is translated from a shorter language like English to a much wider language like German that on average takes up to 30% more space. Give the actual user a chance to comment on used fonts and overall readability.

## 4 Summary

Although at first glance this sounds like a lot of difficulties and never-ending work, over 80% of these issues are easily solved, once you are aware of the problem. For each technical complication, there is a big chance that the solution is already out there.

Your biggest task is to make sure that everyone involved knows that these efforts are done for the good of the customer. Because so many people involved in the creation of software never meet the customer, it is your job to communicate and motivate the customer's right to get localized, easy-to-use software.

If you want to market your software with usability, make sure that the software is properly localized, too.