

Balancing the trade-off in software internationalization – a guide for user interface visualizations

Alexander Friedel

macio GmbH
Am Kiel-Kanal 1
D- 24106 Kiel
Tel: +49 (0) 431 67072-20
alexander.friedel@macio.de

Abstract

With the growing strength in performance embedded systems are more likely to contain advanced graphical user interfaces that are capable of user-centred visualizations. It is strongly recommended then to specify and fulfill operational and technical user requirements, Today user- centred designs is the leading way to create usable products and software. Misunderstandings and malfunction due to wrong usage of the software have to be avoided, due to product safety, law or other reasons. That means time to market, development costs and content of embedded software have to be balanced to guarantee and hold embedded software quality in different localizations. It is a problem of uncertainty that is discussed in this article, what topics are relevant to make embedded software capable of localizations? Is it costly to ignore software internationalization in the first place?

1 Introduction

1.1 Discussion of the trade-off

Software is of growing importance in modern industrial and consumer appliances. As „accessible“ functionality of a device is the most important feature a software can deliver, it is crucial for market success to make sure a product is of an excellent usability. International standards, law or industry regulations (like in medical appliances or safety critical systems) require to be localized and exhibit a proven usability.

As budgets are always short and a new product will be launched mostly first in one market not in every market at the same time, marketing has to answer what locales are targeted first.

Often marketing does not know (or tell) in which markets the product will be sold during its life cycle. From the developers point of view now there is a decision to make: Should i ignore future localizations and just concentrate the budget on the existing requirements? Development time and costs would be smaller, but i know a late internationalization would be much more costly than today! That is exactly the trade-off. It is a problem of uncertainty, that first has to be categorized (what this article is about) and then to be answered individually.

To solve this problem of uncertainty about future returns or development budget needs it is necessary to have an idea what exactly, means software internationalization for my software development project. The next chapters point out what topics (in general) have to be thought of to make the best decision for your project.

For this discussion we ignore other influences like brand or market power for the success of a product. We assume „all things equal“ except the user interface software of a device.

1.2 Definition: Software Internationalization vs. Localization

In this document, internationalization means making program code generic and flexible so that specifications from markets around the world can be easily accommodated. Part of internationalization is making the product localizable, that is, enabling the user interface to be adapted with a minimal impact on the program code.

There are several variants of internationalization [1]:

- 1) Monolingual internationalization - Enables the creation of localized versions of a product, where each localized version supports only its target locale. This is no longer sufficient for business requirements.
- 2) Internationalization for multilocalization - Supports localization and data processing for multiple locales, where the actual locale is selected on execution of the product or at runtime.
- 3) Multilingualization - Enables data processing and display in multiple languages and locales simultaneously, for example to use two displays at once at the same machinery.

The minimal requirement today is Internationalization for multilocalization. Most products, especially in environments with „integrated“ devices like printing machines, fulfill the requirement of multilingualization.

Localization now is the process of customizing a product for a particular market or locale. Different types of customization can be included in a localization; for example translation, altering format fields in resource files, changing date format and so on.

2 Impact on the software development process

2.1 Requirements Engineering and Planning the Road map

It is important to realize, that software internationalization is nothing extra-ordinary. It is a topic like performance or functions where all requirements have to be fulfilled. Internationalization is a business decision (like functions) not a technical decision.

Taking into account different user groups, as the the user-centred design paradigma tells us, and different locales the requirement documents will have more content. Make sure to choose as well the right test method and tests for these additional requirements. Internationalization needs more resources than a development neglecting this issue. How many more resources are needed that should be a result of your requirements engineering.

There are a lot of cases how software internationalization can affect the usage of technology. It is the nature of the problem discussed here that there are endlessly different solutions. A general advise or golden rule is therefore not possible. An example for the impact of software

internationalization could be an embedded device with advanced graphics that uses java technology for visualization. For a software developed in Java, take into account that the integration of a rendering engine for certain texts for example arabic might be mandatory if true platform independency is required. The reason is the eventual missing support of the operating system.

Luckily Java (as well as the .net environment or others) brings already a lot of features that helps us dealing with the issue of software internationalization.

2.2 Software Architecture

Software internationalization affects the creation of the software architecture twofold. It is necessary to examine the functional and non- functional aspects of the software application. Functional criteria for example are the different languages or changing formats and units on the fly. Also functional differences can result from different hardware configurations. With „different hardware configurations“ is meant 1st the hardwar difference between an embedded device and a desktop computer, and 2nd the different machine configuration itself. (for example think of steering wheels in cars, which are positioned at the right or at the left side of the car).

Non-functional criteria have impact on the quality of the resulting software. They are important for software internationalization. Performance, platform independency, security of the software or serviceability affects the costs and therefore are interesting to examine in case of a decision problem like software internationalization: today or later?.

A concept to integrate functional requirements in the software architecture is the layer model. Distinct user interface layers, business model layers and persistence layers are required to create a localizable software application. Otherwise hard coded variables would prevent any correct switching from one locale to another.

Resource managers have been proven concepts in dealing with data that is affected by the internationalization topic.

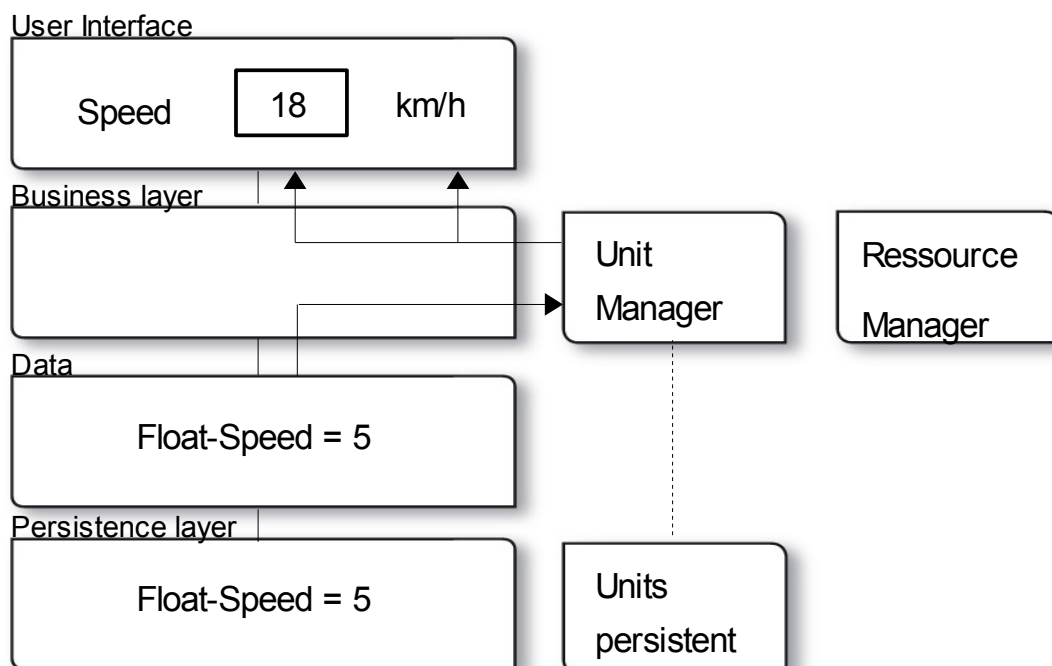


Figure 1: Internationalization affects every software layer [2]

The visualized variable Speed with the value „18“, stated in the unit „km/h“ in the user interface is different from the internal value of the variable „Speed“. The unit manager, like any other resource manager, is responsible of the correct visualization, the fit of variable and unit. Internal the computer calculates with a floating value, e.g. 5m/s which should not be too short. The floating value format has to be defined. This is the unit in which the data is persistent available. When small differences are important a float value should exhibit enough places after the decimal point to avoid too big differences when calculating in other units.

Usage of third party devices or interfaces is often important for software applications in integrated devices. If there is a communication with external libraries, tools or another device these third party widgets have to fulfill the requirements of software internationalization as well. To classify the excellence of third party widgets is therefore an important issue in finding the best way to global products.

The development resources are also dependent on:

- how many sorting methods are applicable,
- is multilocalization necessary,
- is the functionality different across countries
- is an on-screenkeyboard necessary
- how are the users expected to make inputs into the embedded system, are all possibilities supported by the operating system?

2.3 User Interface

The Literature [5] recommends a user interface outlined for the specific needs of users in a market, that acknowledges cultural differences between users for example. No doubt, these differences are existent. It is very costly to develop different look and feels for every market. In real life, as it is done for machines and embedded-systems it is an alternative strategy to concentrate on how to design a user interface that works in different locales. Such a user interface won't use cultural dependent images, colours, gestures or text, e.g.. In this chapter it is stated out what topics concerning the user interface might affect your decision of internationalizing today or later in the product life cycle. Topics that need resources are for example:

1) Text and translation

The most obvious topic. In most cases translation is necessary. A consistent terminology throughout the product is a key success factor for a localization, it makes consistent translation easier. Solution provider for translation services recommend as well that the developer should develop in his native language. This ensures the highest quality of the source text.

A text is different in length when it is translated into other languages. This makes it difficult to define the correct space of tables or boxes. The direction of reading is also different and this could be of importance when developing the user interface of an embedded system. A layout manager as additional software might be appropriate to cope with this challenge.

2) Fonts

The question here is, what characters have to be visualized. If asian characters are displayed you might need a layout engine for complex characters and to display them in smaller sizes. The development budget will also be affected by the choice of the font. The font might need to be licenced.

3) Images and Icons

The graphical items are very important. It is strongly recommended to invest a part of the budget in the design of the user interface. The graphical and interaction design ensures safe and confident operation with the device. User Interface Designers can fulfill for example the criteria to create a design that works for different cultures. Within the design icon-metaphors should be avoided, as well as body parts or cultural dependent gestures like putting a thumb up in order to visualize „ok“. So for most cases internationalization does not affect your budget for user interface design. It even might save money and resources for the software development.

4) On Screen Keyboard

Some operating systems like Windows CE have their native interface and already bring an on screen keyboard. It is very nice to use this feature, because to create your own input device for different languages could be quite costly. The disadvantage is the windows standard. Whenever you present your own look and feel and somebody changes the windows parameters your on-screen-keyboard will look different.

The recommendation in case of input devices is to use as many functions of the standard operating system as possible, but to be aware that this might colide with the user interface design.

3 Guidelines for balancing the trade-off

- Which languages need to be supported? Can you eliminate a writing system?
- Is on-the-fly localization necessary (swapping from one to another language)?
- If you change the language, do you have to change the fonts as well? Do you own these fonts, including the licence?
- What date and time formats need to be included?
- What formats of numbers are important? Do they have to be dynamic? Are numberformats fixed?
- What units and currencies have to be displayed? Is clear in what format data is saved? Do unit formats have to be independent from changing the language?
- Do you need an on screen keyboard? Can you use a 3rd party keyboard? For what languages is the keyboard used? Do you need alphanumeric and numeric keyboards? Are numberformats fix for the numeric keyboard? Is there a concept of error-handling with the numeric keyboard?
- Does the user interface layout has to be variable? Do you need variability in textlength, textdirection and resolution? Do you have to support different display resolutions?
- Are icons localizable? Is the software able to start a local version with different than standard icons? What icons have to be localized, respectively have to be internationalized? Is the process defined and described?
- Is guaranteed that just data is saved that is independent from a locale?

- Is there a concept for the on screen help? Is a process defined to localize the on-screen-help?
- Do functions need to be localizable? Is the software capable of localizations in functions of the software?
- The build process needs to support all locales.
- Is there a concept for communication with 3rd party devices? Are the interfaces described? Is it possible to verify the correct interpretation of characters and numbers on all devices?
- What user interface elements are part of 3rd party libraries? How are these elements localized?
- If sorting methods are part of the software, do you need different sorting rules?
- Address- and telephone numbers: Are all required formats defined?

4 Summing Up

Internationalization is a regular part of the software development process. In every stage from requirements engineering until testing and integration the topics of software internationalization are relevant for a successful development. Especially for embedded devices where the support of the operating system is different from desktop applications is more at stake. Software internationalization does cost something. The right requirements engineering will guide you through the relevant topics. A balanced trade-off is then possible. The requirements engineering and software development will cost some more resources. The user interface, when it is designed with a professional interaction designer, will help you to keep costs down. The reason is that solution providers for interface design and software technology at the same time know the pitfalls and can guide you to the appropriate solution.

References

- [1] I18N taxonomy, Sun Developers Network, 2008
- [2] Figure 1: macio GmbH, 2008
- [3] I18n in Software Design, Architecture and Implementation, Sun Developers Network, 2008
- [4] Testing & I18N, Manfred Rätzmann, Galileo Computing, 2004
- [5] A Practical Guide for Localization (2. Ausgabe), Esselink, Bert; John Benjamins, 2000