Effective Testing of Localized Software

Testing localized software is a complex and time-consuming task. With the help of the testing tools developed for HP patient monitors, local language validation for these products is fully automated.

by Evangelos Nikolaropoulos, Jörg Schwering, and Andreas Pirrung

Localization plays a very important role in the successful marketing of software all over the world. For medical devices there are legal requirements to provide instruments and accompanying documentation in the language of the healthcare personnel who use them (as is the case in the European Union). It is often forgotten that localized software is *different* software from the original (most probably in English) that was used for system integration and final validation. Localized software undergoes a proper integration cycle (integration of software and translated strings) and must be validated separately. The complexity of this validation is obvious if one considers the efforts required to check all error conditions and the corresponding error messages (and to understand them) for software in every language where the product is marketed.

The most common errors in localized software, assuming that the translation is done by a professional translator for this language and is correct, are:

- Missing strings (empty messages, parts of screen text missing, menu selection items missing)
- Strings with wrong attributes (maximum length exceeded— a possible crash cause) or strange characters filling up the remainder of a field
- Wrong strings (not reflecting the intentions of the author for this particular context)
- Various misspellings or violations of grammar rules applied to the language produced through the combination of translated strings by the software
- Strings not properly cleared in a text field before a new string is displayed.

Local language testing in our laboratory is composed of two steps: the verification of the translation and the validation (regression testing) of the localized software.

To verify the translation, a translator goes over all possible screens, messages, help texts, printouts, and so on to check for translation errors. The difficulty here is that in most cases the translator is not a frequent user of the device under test, and needs assistance in operating the medical instrument and generating all possible string combinations.

The aim of validation (regression testing) of the localized software is to prove that the localization has not negatively affected the functionality and performance of the instrument. Additional attention must be paid to typical localization errors (overflows or garbage generation).

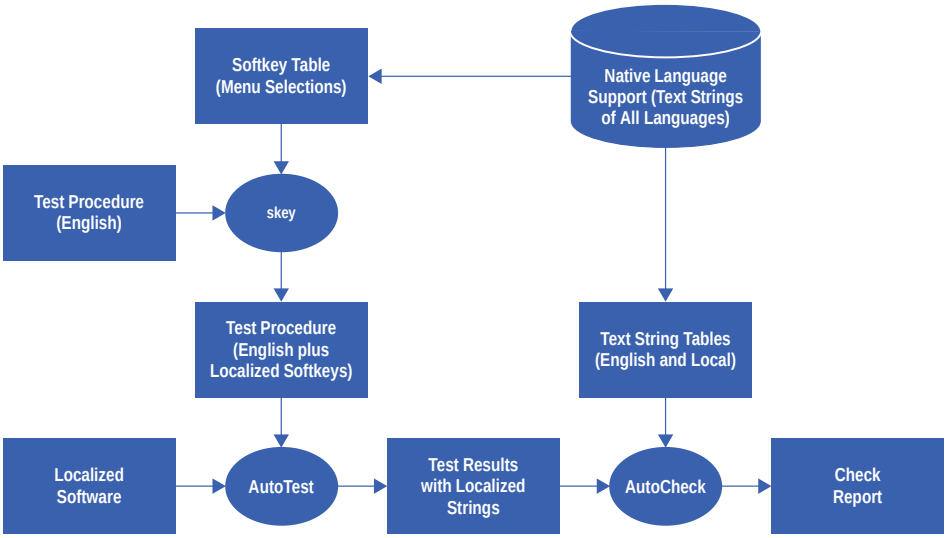

Fig 1. *Local language validation process for patient monitors.*

(a) Extract of a Test Procedure for Blood Pressure as it Is Used for Tests of English Software

```
                .....
* alarm suspended -> alarms not suspended
merlin mainscrn
mecif skey -kalarmvol                                   Press a hardkey
merlin "SwitchOnAlarms"                                 Make a selection from a menu
* ======================================================
^ Verify begin                                          Verify statement (AutoCheck)
^  INOP "NBP EQUIP MALF" ;
^  Sound is hard inop ;
^ Verify end
* ======================================================
                .....
* ======================================================
^ Verify begin
^  twprompt is "Problems with the pneumatic are detected" ;   Verify statement (AutoCheck)
^ Verify end
* ======================================================
                .....
```

(b) The Procedure after Translation of Softkeys (Menu Selections) to Finnish

```
                .....
* alarm suspended -> alarms not suspended
merlin mainscrn
mecif skey -kalarmvol                                   Press a hardkey (no translation)
merlin "HälytPäälle"                                    Make a selection from a menu
* ======================================================   (translated)
^ Verify begin                                          Verify statement (AutoCheck)
^  INOP "NBP EQUIP MALF" ;
^  Sound is hard inop ;
^ Verify end
* ======================================================
                .....
* ======================================================
^ Verify begin
^  twprompt is "Problems with the pneumatic are detected" ;   Verify statement (AutoCheck)
^ Verify end
* ======================================================
```

Fig 2. *Example of the steps of the local language test process.*

(c) Protocol File after Test Run with Software in Finnish

```
                .....
23:35:14 * alarm suspended -> alarms not suspended
23:35:14 merlin mainscrn
23:35:17 mecif skey -kalarmvol
23:35:21 merlin ”HälytPäälle”
                .....
23:36:11 * ==================================================
23:36:11 ^ Verify begin
23:36:11 ^  INOP ”NBP EQUIP MALF” ;
23:36:11 ^  Sound is hard inop ;
23:36:11 ^ Verify end
23:36:11 * ==================================================
Filter: NBP -NU, @
23:36:13 (hard inop sound ARec: CS)
23:36:13 I ”NBP LAITEVIRHE ” O - 1 NBP -NU p=---o--H
                .....
23:36:23 * ==================================================
23:36:23 ^ Verify begin
23:36:23 ^  twprompt is ”Problems with the pneumatic are detected” ;
23:36:23 ^ Verify end
23:36:23 * ================================================== 0
Tuned : TWPROMPT,
23:36:26 F ” NBP ” 8, 15
23:36:26 P ”NBP last calibration done 16 KES 94 15:32 ” W 3 T @
23:36:29 P ”Pneumatiikassa on havaittu ongelmia ” W 3 T
23:36:31 EOT
                .....
```

String translated

Combined string only partly translated

String translated

Fig 2. *(Cont.)*

Automated Local Language Validation

With the help of the testing tools developed for our patient monitors (see Fig. 1 and the accompanying articles in this issue), local language validation is a fully automated process:

1. Translation tables called *local language tables* are prepared from the native language support database containing all the English strings and their corresponding translations.
2. A test package, a subset of the test procedures designed for the regression testing of the original English version, is compiled.
3. A copy of each test procedure out of this package is translated into the local language by a tool developed by software quality engineering called skey. The intention here is to replace in the test procedures the selection menu items (softkeys) with the corresponding localized terms. Of course, a test can be executed by passing the position of a selection item (e.g., “press the second selection on the third menu”) to the test execution tool, but this approach has proved to be ineffective. The issue here is not to test that the “second selection of the third menu” works, because this was already tested in English, but to prove that the “second selection of the third menu” is translated correctly, and if selected, produces exactly the same behavior as its English counterpart. By calling the selections by their values and not by their positions we achieve higher test coverage and we test functionality and translation at the same time. Another argument for this approach is that the “second selection on the third menu” may be configuration dependent (even local configuration dependent) and therefore not accessible by a position dependent test (e.g., it is still on the third menu but in the sixth place). Thus, calls by value make test procedures more robust.
4. The translated test procedure is passed to AutoTest[1] and is run on the localized software. The results are saved in protocol files containing English verify statements (the expected results), localized softkeys (selections), and localized actual results (see the example in Fig. 2).
5. The protocol files are submitted to AutoCheck (see ***Article 14***). First, the AutoCheck preprocessor takes over the task of translating the verify statements. It uses the local language tables to replace the English text in the verify clauses with the localized text. On the second pass these translated expected results are compared with the localized actual results. Discrepancies are reported in the normal way but with localized content.

A special solution is also provided for Asian languages (simplified and traditional Chinese and Japanese), which use 16-bit codes. For these languages the hexadecimal equivalent for each character is used in the test procedures instead of the "drawn" character. This enables us to keep such characters in ASCII files (like the test procedures and the protocol files) and use them with the test execution and evaluation tools.

The automated local language validation has dramatically improved the process of localized software release. It has reduced the effort for local language testing for a new patient monitor release from twelve to four weeks and has significantly increased the test coverage compared with the traditional manual testing approach.

Acknowledgments

The authors wish to thank Gerhard Tivig for his support during the development of the local language test tools.

Reference

1. D. Göring, "An Automated test Environment for a Medical Patient Monitoring System," *Hewlett-Packard Journal*, Vol. 42, no. 4, October 1991, pp. 49-52.

▶ Go to Journal Home Page