

Exhibit 1023

A Characterization to Evaluate Graphical User Interface Frameworks for Television Receivers

Milan Z. Bjelica¹, Nikola Teslić², Istvan Papp³, Milan Savić⁴

Abstract – This paper presents an analysis of the existing Graphical User Interface (GUI) creation frameworks, in order to investigate how well these frameworks can be applied for graphical user applications in modern television services. The analysis is based on a characterization presented in this paper and it provides a list of key enablers and a quick methodology that decide the most suitable graphical framework. Some of the considered aspects were the license (possibilities of commercial use of the solution), levity of integration with the standard user interface for television receivers (remote controller), usability for the end user, development effort, system resources utilization and energy consumption. Finally, we present a case study where the characterization with the specific GUI application (TVHome) has been used. We introduce several GUI design conventions, to make GUI framework the most applicable for the television receiver.

Keywords – GUI, TV, metrics

I. INTRODUCTION

Thinking of a possibility to utilize off the shelf graphical frameworks in TV receivers started right after TV receiver became a platform capable to render and display graphical user interface elements. Currently modern TV receivers are equipped with a processor and adequate graphical subsystems. TV platforms are controlled by real-time operating systems, which are very often based on Linux kernel. This gives a possibility of execution of a vast number of applications designed for Linux, directly on TV receiver. One of the basic requirements of the system to support execution of software applications that are based on the existing graphical frameworks, is the existence of a fully functional hardware abstraction layer (HAL) as a software middleware. In the case of Linux, common middleware for this purpose are *DirectFB* [1] or *Xwindow* [2]. It must be born in mind the nature of TV receiver as the embedded computer system with real-time constraints, and the fact it must always be ready to accomplish its primary tasks, such as demultiplexing of digital TV (DVB) streams, decoding audio and video packets etc. Regarding that, graphical user interface must economically use TV receiver resources, such as system memory and processing power.

¹Milan Z. Bjelica is with the Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia, E-mail: milan.bjelica@rt-rk.com

²Nikola Teslić is with the Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia, E-mail: nikola.teslic@rt-rk.com

³Istvan Papp is with the Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia, E-mail: istvan.papp@rt-rk.com

⁴Milan Savić is with the Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia, E-mail: milan.savic@rt-rk.com

Therefore, the start-up analysis set, according to the authors of this paper, may consist of the following (order of enumeration is random): *Nano-X* [3], *EmbeddedWizard* [4], *QT (Qtopia)* [5], *GTK (Gimp ToolKit)* [6]. The descriptions of other possible frameworks can be found in [7]. Researches on the usage of these frameworks for the embedded systems have been partially conducted in the past. Zhong and Jha have evaluated the usage of *GTK* and *QT* frameworks for the handheld devices, basing their research on the estimation of energy consumption depending on software [8]. Hansen provided an empirical testing and comparison of *Qtopia* and *Nano-X* running on MIPS-based embedded system with 6,5" TFT display used for the output. It regarded the perspective of needed development efforts, memory consumption and subjective impression on the GUI appearance. Some experiences with the *EmbeddedWizard* framework have been presented in [10]. To the best of our knowledge, there has been no systematic analysis and review presented to this moment, that would provide a reference to engineers and researchers to quickly and easily decide on the optimal graphical framework for their embedded device (and especially TV receiver).

As following, we give a brief review of the characteristics and possibilities of listed frameworks for GUI creation. We define key parameters and give comparative view. Some inputs to the comparison were obtained from the evaluations from [8] and [9]. We have described properties of TV receiver from the perspective of control, display etc. and then proposed a set of additional parameters for GUI frameworks assessment for TV. Finally, we have presented a development process of a GUI application based on one of the frameworks (*GTK*) in TV receiver and assess to its key parameters. An overall grade of usability has been given, with respect to TV receiver.

II. GUI CREATION FRAMEWORKS

Nano-X, previously known as *Microwindows*, is an open source project from the *Nano-X.org* community. *Nano-X* has an MPL license, that allows creation of commercial software applications, with the constraint that the source code of the framework itself must remain open and available. *Nano-X* is a framework that relays on *framebuffer* graphical subsystem of *Linux* kernel, and is written in C. Programming GUI that relays on *Nano-X* demands a significant amount of code dedicated to graphic processing, having in mind that *Nano-X* is a framework of low abstraction level. It is possible to handle graphical regions, windows, messages and focus, as well as draw different shapes and write text to screen, but there are no high level containers such as ready made graphical controls or widgets that could be used quickly and easily (e.g. buttons, edit boxes etc). Porting libraries to *Linux*

requires effort in identifying and cross-compiling all dependancies (e.g. for PNG image decoding). This altogether increases the lead time needed to start using the framework. The framework is fast enough and consumes little system memory (< 100 Kb) [9].

EmbeddedWizard is a software product from *Tara Systems*. It is a framework for the development of portable GUIs dedicated to embedded systems. The framework provides a graphic GUI editor for Windows. Using this editor GUI creation is easily made. The editor is object-oriented, containing many ready-made classes that can be instantiated as GUI elements. GUI behavior is completely programmable from within the editor, by using a high level programming language *Chora*, which resembles *Object Pascal*. *EmbeddedWizard* makes the usage of *DirectFB* graphical subsystem for *Linux* possible, and automatically compiles C program code that describes GUI and can be easily integrated to the embedded software application. The connection from the GUI code to the application code is established by *native* calls to external application functions. For commercial purposes, it is necessary to obtain a proper license from the manufacturer. *EmbeddedWizard* is the only field proven framework that is presented in this paper, which is extensively used to develop GUIs for TV receivers.

Qtopia is a part of a software package *QT* by *Nokia* (previously designed by *Trolltech*). It consists of a C++ SDK (*Software Development Kit*) that runs on top of *DirectFB*, and of appropriate GUI editors for both *Windows* and *Linux*. This framework provides a number of graphical controls and widgets, that can communicate by using slots and events. *Qtopia* is dedicated to embedded systems, as opposed to *QT* that is used on personal computers. Setting up the framework is simple, since all needed libraries are integral parts of the framework. This reduces the lead time needed to start the usage. The latter, however, results in increased demand for memory resources, when compression can reduce this consumption to 1.1-3.2 MB [9]. By the time this paper is submitted, *QT SDK* license has been changed from GPL (*General Public License*) to LGPL (*Lesser GPL*) in March 2009, allowing commercial applications free of cost. Earlier, a special license needed to be purchased for commercial purposes.

GTK is a framework completely written in C. Originally, it was intended to support *GIMP* software application for PC. By adding support for *DirectFB*, its application for the embedded devices has been made frequently. *GTK* supplies a number of already made graphical widgets that are object-oriented (in spite of C as the programming language – *Glib* library is being used to provide inheritance mechanisms). *GTK* is somewhat harder to set up, since it has many dependancies to separate libraries for different purposes (such as decoding different image formats, support for fonts etc). There are editors that can speed up GUI development, although there is no a complete SDK for GUI programming. One of these editors is *Glade* that can define GUI by using XML description. One approach with similar concept was given in a research from Lamberti and Sanna in [11]. Authors of this paper determined that memory footprint of *GTK* is

similar to that of *Qtopia* (2.2-3.2 MB) [12]. One of the advantages of *GTK* is LGPL license that makes usage of *GTK* in commercial purposes possible without disclosure need for application source code.

III. BASIC EVALUATION CRITERIA

With although respect to usage of the embedded device based on *Linux* operating system, this paper defines eight evaluation criteria for the GUI creation frameworks.

Programming language. Given the nature of the embedded systems, it is often assumed that C or C++ will be used for the development. If the resources of the target development platform are very much limited (e.g. 8-bit processing) or there is no C++ compiler available, C language comes as the first choice. If we mind the development time and software structure (object-oriented approach, inheritance) the advantage is C++. Finally, the basis of the decision on which programming language to use is the language that was used for the rest of the system – to avoid problems with later integration of the system with GUI.

Lead time. Within this category a subjective evaluation should be given (expressed in man days) on the time needed for an engineer to get to know with the graphical framework before he becomes capable to use it to develop software.

Memory footprint. The amount of memory that the framework consumes (including all dynamically loaded dependencies) expressed in Kb or Mb.

Documentation. This category is of extreme importance when open source frameworks are considered; since it has been often the case, such frameworks are not well documented. The evaluation on the quality of documentation includes documentation of API functions, as well as general instructions on the use of the framework. The evaluation returns a subjective score (Likert 1-5).

Installation. The effort needed to cross-compile the framework to target platform (giving the score from 1 to 5, where 5 depicts least effort needed).

License. It is very important to know the license, especially if there is an intention to use the framework commercially. The score for this criterion is descriptive, showing whether it is possible to use the framework for commercial software development without disclosing source code, and what are the general legal terms of such usage.

Application suitability. Depending on the final device that should be developed, graphical framework and its graphical controls can be more or less suitable for the user application. In the scope of this paper we further have further discussed and evaluated this category with respect to TV receiver, what has been presented in Chapter 4.

Energy consumption. It is significant to take account on the energy consumption, what is nowadays a keyword for the electronics consumers. This criterion is of extreme importance for handheld devices running on batteries. The consumption is expressed in *energy units (EU)* as in [8], thus enabling comparison of energy consumption that depends on graphical framework only.

IV. CHARACTERISTICS OF GUIs FOR TV RECEIVERS

To achieve desired functionality for end user, graphical frameworks should fulfil the requirements of user interface for TV applications. Lekakos and Chorionopoulos in their research [13] give an overview of GUI elements of importance for TV. We identify nine criteria for the GUI creation frameworks, derived from the nature of TV receivers. Each criterion is evaluated by the subjective score from the side of technical person, on the Likert scale 1-5.

Integration with the remote controller. Having in mind that the remote controller is the only device for the control of TV receiver, framework must facilitate integration with the subsystem for handling remote controller events.

Navigation. Framework should support a simple mechanism of navigation including four arrow keys (up, down, left, right) and an OK button for confirmation. A button to undo last action should also be included.

Imitating TV user interface. Framework should make a simple development of a GUI possible that is similar to GUI already established as a standard for TV receivers. This includes creation of vertical menus with a possibility to support multiple levels of hierarchy. Current TV programme being shown on screen should be covered as little as possible. It would be convenient if a framework supports transparent GUI elements, and is able to provide mechanism to show GUI in a part of the screen (Picture in Picture – PIP functionality).

Paging. It is convenient if a framework supports natively GUI decomposition into multiple pages that can be easily listed and switched back and forth.

Undo. Framework should facilitate cancelling operation, or reverting to previous software state in GUI.

Messages. Framework should support a mechanism of showing messages that do not require users to explicitly confirm the message has been read. This makes GUI less intrusive. Ideally, framework should support a mechanism to remove a message after a predefined time period elapses.

Help. It is convenient that a framework provides support for context help in GUI. The best way to achieve this is to provide a text bar showing currently selected function.

Text entry. A framework should facilitate text entry to graphical text controls (widgets) such as edit boxes, given the remote controller as the only means of input. Ideally, system should natively support entry by multiple keypress (as on mobile phones) or T9 [14].

Display. TV receiver is being regarded from the distance, while resolution of screen can be low (e.g. PAL – 720 x 468). This is why a framework should make possible creation of sharp, clear and highly visible graphical elements, and avoid too small details.

IV. COMPARATIVE VIEW

According to the given criteria, we give a comparative view of the GUI creation frameworks. Each category was evaluated by engineers of embedded systems. According to that view, we have selected a framework to develop a specific GUI TV application. Based on the criteria used to address applicativeness of a framework on TV receiver, we have calculated *TV factor* as a median value (TABLE 1).

The table reads clearly for the crucial parameters to decide the most suitable framework for the TV receiver. If we take into account low abstraction level, lower grades for *Nano-X* were expected. As modern TV receivers usually consist of better hardware (32-bit processing, 64 MB of memory, 300 MHz system clock or higher), usage of *Nano-X* can be justified only for the development for older models, where C is an absolute must.

TABLE 1
COMPARATIVE VIEW OF GUI CREATION FRAMEWORKS

	<i>Nano-X</i>	<i>Embedded Wizard</i>	<i>Qtopia (QT)</i>	<i>GTK</i>
Programming language	C	Chora -> C	C++	C
Lead time	8 md	5 md	5 md	6 md
Memory footprint	100 Kb	1 Mb	1.1-3.2 Mb	2.2-3.3 Mb
Documentation	3	5	5	4
Installation	3	4	4	3
License	MPL OK for commercial software	Tara Systems proprietary license	LGPL OK for commercial software	LGPL OK for commercial software
Energy consumption [8]	not available	not available	19 K EU	20.5 K EU
Integration with the remote	1	4	4	3
Navigation	2	4	4	3
Imitating TV GUI	2	5	4	3
Paging	1	2	5	5
Undo	1	2	4	4
Messages	1	5	3	3
Help	1	2	5	5
Text entry	1	2	3	3
Display	4	4	3	2
TV factor	1.56	3.34	3.87	3.45

EmbeddedWizard was proven very convenient, where the only problem can be its license. QT is well positioned, when the usage of C++ can be a blocking issue for some systems. Finally, although it scored slightly lower than *QT* or *EmbeddedWizard*, *GTK* uses a license open to commercial application, is rather portable and can be a good compromise. Energy consumption with *GTK* is based on the research when this frameworks runs on top of *Xwindow* instead of *DirectFB*, when it should score much better for energy (by the opinion of the author).

V. TV GUI APPLICATION DEVELOPMENT

Test TV platform that was used for testing GUI creation framework, is a 32-bit TV platform from *Micronas*, equipped with 64 MB of memory. Platform runs on *MIPS* processor, with all needed hardware blocks also integrated (input stream demultiplexing, audio and video decoding, graphical layers rendering, OSD rendering). One of the goals to be achieved with GUI creation framework, is to develop a middleware that would provide means for rapid GUI development for TV receivers. This made license crucial for the final decision. It was also important to use C programming language, since there was no C++ toolchain available for the target platform. Having in mind the discussion from the Chapter V, it was decided to use *GTK* over *DirectFB* as preferred framework. Following we present some details of the implementation and

experiences with this framework, going through all phases in the development process.

Porting libraries to target platform. *Linux* distribution available for TV platform was free of most of needed libraries that *GTK* depends on, therefore all needed libraries had to be cross-compiled prior to *GTK*. Cross-compiling and bringup, until *GTK* test application was shown on TV screen, took 2 man weeks to complete. In many cases workarounds were applied, including changes in the framework source code, what appears to be usual with *Linux* open source, to make all libraries compile and/or run correctly. It was also a problem to link all the libraries together, and to make the solution compatible with the existing *DirectFB* subsystem. Cross-compile was done on a separate PC, and the output was taken in the form of a number of dynamic *Linux* libraries (.so).

Adaptation middleware implementation. Main goals of the middleware, related to *GTK* framework, are the provision of: (1) processing of messages received from the remote controller, their interpretation and appropriate signalling to *GTK*; (2) a set of new graphical widgets compatible with TV receiver (e.g. edit box with the ability to enter text with a remote control using multi tap, vertical menu etc); (3) a method to specify GUI based on XML, including characteristics of newly added *GTK* widgets; (4) API towards user application, with services to show GUI elements and change their current state. To specify GUI *Glade* tool has been used, with the introduction of design conventions. Design conventions specify the way in which graphical elements should be placed and named, to be interpreted accordingly, by the middleware. This tool creates XML GUI description, which can be loaded at runtime and interpreted by *glade* library on *Linux*, as a counterpart. Upon loading, system populates a tree of *GTK* GUI elements, which are further processed by the middleware. XML specification itself provides means to define names for C functions to be called on certain GUI events. Pairing GUI and implemented functions is done dynamically, after the XML specification is loaded. Detailed description on how design conventions are applied is given in [12].

Test application. For the purposes of testing, *GTK* based GUI application was created, which would allow controlling appliances in a household (*Smart Home*) from TV receiver [12] (Fig. 1).

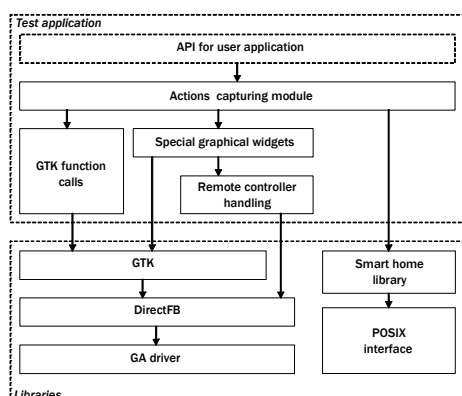


Fig. 1. Test application with its layers including the middleware and *GTK* libraries

Although the response to user input was slow at the beginning, due to the high memory requirements by graphical elements that were embedded to test application (e.g. size of images used to draw menus), after optimization the response time was improved to less then 1 second. Real testing showed that *GTK* was applicable for TV OSD, but detailed optimization procedure for all the libraries is needed, and economical use of GUI elements must be applied. One GUI form is presented in Fig. 2.

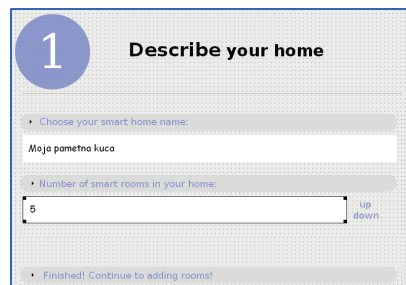


Fig. 2. One *Glade* GUI form

VI. CONCLUSION

This paper contributes by giving clear instructions when choosing GUI creation framework for an embedded device based on *Linux* OS. Paper emphasizes the application of the frameworks for TV receiver, as one specific device. To the best of authors' knowledge, this is a new analysis. We have proven the applicativeness of GUI for TV receiver, developed using *GTK* framework. Final conclusion would be that it is essential to optimize and adapt, when applying an open source framework to a TV receiver. Further research in applying frameworks for GUI development is justified, while the topic is commercially vital.

REFERENCES

- [1] *DirectFB*, www.directfb.org
- [2] H. Gajewska, M. Mannase, and J. McCormack, *Why X is not our Ideal Window System*, Software – Practice & Experience, Vol 20, Issue S2, 1990.
- [3] *Nano-X*, www.microwindows.org
- [4] *EmbeddedWizard*, Tara Systems, www.tara-systems.de/emwi/
- [5] *QT*, www.qtsoftware.com
- [6] *GTK*, www.gtk.org
- [7] *The Embedded Linux GUI/Windowing quick reference guide*, www.linuxdevices.com/articles/AT9202043619.html
- [8] L. Zhong, and N. K. Jha, *Graphical User Interface Energy Characterization for Handheld Computers*, CASES '03, ACM, 2003.
- [9] M. Hansen, *Choosing a GUI Library for Your Embedded Device*, *LinuxJournal*, 2007.
- [10] M. Z. Bjelica, M. Savic, and T. Aleksic, "One Solution of Integrated File Browser for Connected Bluetooth Device as an Application for TV Receiver", *Proceedings of Conference ETRAN*, 2008.
- [11] F. Lamberti, and A. Sanna, *Extensible GUIs for Remote Application Control on Mobile Devices*, *IEEE*, 2008.
- [12] M. Z. Bjelica, *Implementation of a Home Automation System with Graphical User Interface on TV Receiver*, master thesis, Library of Faculty of Technical Sciences, Novi Sad, 2008.
- [13] G. Lekakos, K. Chorianopoulos, and D. Spinellis, "Information Systems in the Living Room: A Case Study of Personal Interactive TV Design", *9th European Conference on Information Systems*, Bled, Slovenia, 2001.
- [14] *T9*, Nuance, www.t9.com