

Developing an Open Source Integrated Development Environment for a Mobile Device

Juha Järvensivu, Matti Kosola, Mikko Kuusipalo, Pekka Reijula, and Tommi Mikkonen

Institute of Software Systems, Tampere University of Technology

P.O.Box 553, FI-33101 Tampere, Finland

laika@cs.tut.fi

Abstract— Software development can be eased with an integrated development environment (IDE), which allows the use of different individual tools from one single development platform. Unfortunately, when developing software for a particular embedded system, the development of an IDE for a certain device can be expensive, as the development of an IDE requires a lot of resources. In this paper, we introduce our experiences regarding the development of an integrated development environment for a mobile device, Nokia 770 Internet Tablet. Our goal was to aim at a fully-fledged IDE with the lowest possible costs. In order to accomplish this, we turned to open source development communities, and targeted our effort to the integration of already existing components into a simple yet practical IDE. Towards the end of the paper, we also discuss the effort it took to develop such a system.

Keywords—component; integrated development environment, open source

I. INTRODUCTION

Software development can be eased using an integrated development environment (IDE), which allows the use of different individual tools from one single development platform. Nowadays, a considerable proportion of applications are built using some form of development environment. When considering the development process of the software used in embedded devices, the development environment is even more important because of the need for cross-compilation and application packaging for installation. Unfortunately, as embedded devices are often specialized and the life span of applications intended for a particular device can be short, the investment in a special tool for that particular device alone can become high. For practical reasons, one can then aim at using already existing components and tools to ease and to speed up the development. In such context, open source communities can become an important source of software, as they contain a lot of already existing code, and, moreover, the code can be adapted for different purposes.

In this paper, we present the preliminary findings and our experiences from open source project Laika, whose aim was to create a user friendly integrated development environment for the software of an embedded Linux system with the lowest possible costs and development effort. The device for which our work is targeted for is the Nokia 770 Internet Tablet [1] (Figure 1.). It is a entirely new type of device for Nokia, since it is not a phone but more like a PDA that runs on a Linux operating system. The operating system is based on embedded Debian, and the most recent version Internet Tablet OS 2006

Edition run the 2.6.16 kernel. To allow the use of Linux on portable devices Nokia has initiated the creation of a new open source platform for embedded development, called *maemo* [2]. In this paper, we will introduce how the tool development took place, what were the main decisions in different phases of the project, and what we have learned from the experiment. Towards the end of the paper, we will also discuss the development effort we have invested in Laika.

The rest of the paper is structured as follows. Section 2 introduces the reader to the open source projects related to the Laika. Sections 3-6 introduce the different releases of the project. Section 7 introduces the main lessons we have learned, and discusses future plans and the invested amount of work. Section 8 concludes the paper.

II. OVERVIEW

As already mentioned above, Laika is an open source project aiming at the creation of an integrated development environment for developing applications for embedded Linux devices. The main idea of the project is to integrate several open source projects into a single software development tool, where individual component subsystems can be used in a seamless fashion. The open source communities related to Laika and the roles they play are listed in TABLE I. In the following, we give a more detailed discussion on their contribution to Laika.



Figure 1. Nokia 770 from Nokia press photos

maemo. The software platform for Nokia Linux handhelds, *maemo*, is composed of popular open source components used on Linux desktop distributions. It consists of a precompiled Linux kernel, platform libraries, and Hildon UI framework, which is based on GTK+; in fact the whole platform is binary compatible with GTK+ binaries. However, one restriction is that all the GTK+ widgets might not function

as they are supposed to, because of the limitations in screen resolution and limited resources. The structure of the **maemo** platform is illustrated in Figure 2. One can regard Hildon Application Framework as the “new” thing of the platform. It is based on components from the GNOME open source platform. For developers, this implies small differences and additions to a normal GNOME application development. There are few new widgets to be used in mobile devices, and generally the framework is more suited for small displays and alternative input methods, such as stylus.

TABLE I. COMMUNITIES RELATED TO LAIKA

Community	Role	Description
CDT	Laika IDE uses CDT source code in its implementation.	C/C++ development environment for the Eclipse platform.
Eclipse	Laika is implemented as Eclipse plug-in.	A vendor neutral open development platform.
Gazpacho	Laika supports Gazpacho to visually design user interfaces.	Graphical user interface builder for the GTK+ toolkit.
Maemo	Provides application framework and platform libraries used in Laika IDE.	Development platform to create applications and new technology for Nokia Linux handhelds.
Pydev	Laika Python integration is based on Pydev project.	Python development environment for the Eclipse platform.
Scratchbox	Laika is utilizing Scratchbox to cross-compile software for embedded Linux devices.	A cross-compilation toolkit used by maemo platform.

Scratchbox. A sandboxed cross-compiling tool and development environment Scratchbox [3] is used to develop applications for embedded devices on a desktop Linux. Different CPU architectures can be used inside Scratchbox as ‘targets’. Each target is can be configured differently, and ‘toolkits’ (such as Debian or Doctools toolkit) can be applied for different target platforms. The toolkits contain the necessary files for developing applications e.g. the Debian toolkit contains the necessary commands for creating Debian installation packages like *dpkg*. In order to support creating and porting applications for Nokia 770, the development of the **maemo** rootstrap for Scratchbox has been initiated. Moreover, Scratchbox provides the possibility to develop applications for **maemo** compliant devices using the **maemo** rootstrap, which is basically a root image.

Laika. As already mentioned, the goal of the project was to create a professionally usable IDE with the lowest possible costs by using and integrating existing open source components. At the beginning of the development, we chose Eclipse [4] as the platform on top of which other subsystems would be integrated. Eclipse is an open source community, whose projects are focused on providing a vendor-neutral open development platform and application frameworks for building software. There were other alternatives, such as Anjuta [5], but we chose Eclipse because the developers were more familiar with it, and because it offers a plugin architecture that can be extended with specialized plugins for different purposes. Therefore, while the Eclipse project provides a generic development platform without any particular programming language in mind, it can be conceived as the foundation for a number of integrated development environments, and it has proven itself as an appropriate platform for Laika IDE as well. The user interface of Laika is displayed in Figure 3.

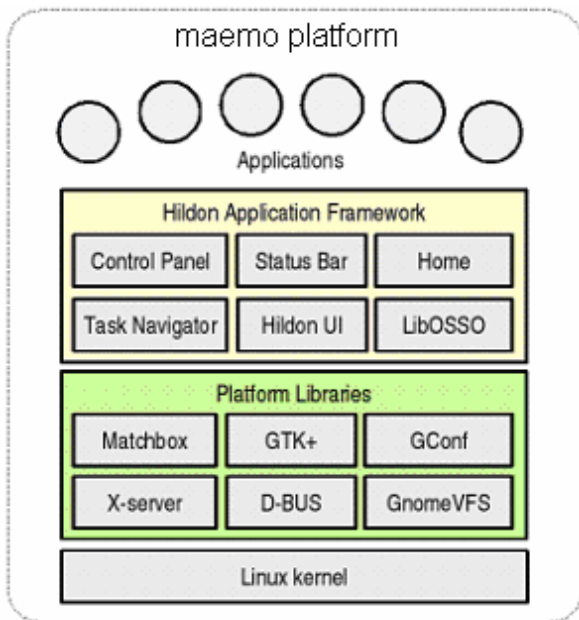


Figure 2. maemo platform

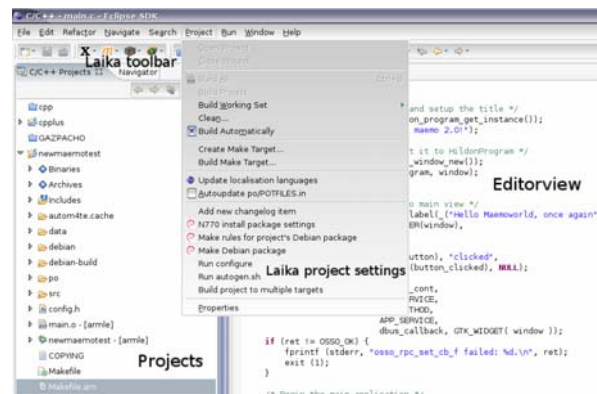


Figure 3. User interface of Laika

III. RELEASE 1: BASIC FUNCTIONS

From our perspective, the work for the first release of Laika started on the 16th of May 2005. At first we familiarized ourselves with the **maemo** platform and Scratchbox toolkit. Next, we went on searching for existing C/C++ -plugins to extend, and decided to begin expanding CDT (C/C++ plugin for Eclipse). The C/C++ Development Toolkit (CDT) is a set of

Eclipse plug-ins that provides C and C++ extensions to the Eclipse workbench, and the CDT Project is working towards providing a fully functional C and C++ Integrated Development Environment (IDE) for the Eclipse platform [6]. The CDT provides a C/C++ IDE that simplifies many of the same tools that you can use from the command line. The CDT can also communicate with many external utilities and interpret their responses. Such auxiliaries include, for example, the following:

- build (such as make),
- compile (such as gcc),
- debug (such as gdb).

The real work began by examining the source codes of CDT plugin in order to expand the features of CDT to work inside a sandboxed environment.

The goal of the first release was to implement features required for using Scratchbox through the IDE and to expand CDT features to work inside the sandbox provided by Scratchbox. The CDT project has many useful features including syntax highlighting, debugger, launcher, parser, and search engine. The idea was to offer support for both normal projects and maemo based projects that are located inside the sandbox. Running and debugging a C/C++ application works similar to CDT, but applications with a graphical user interface can be run on a normal X-window or an X-window using maemo emulator. In addition to the integration of CDT features, a lot of resources were used for the experimental work of utilizing automake tools and a tool that automatically creates installation packages. The first release was also to contain some templates for creating applications. Components used in this phase of the project, as well as their relations, have been illustrated in Figure 4.

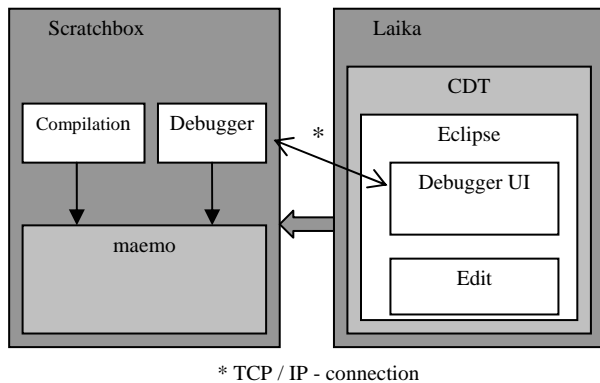


Figure 4. Contents of Release 1

After a few weeks of development we received the groundwork for the same kind of a plugin that had been developed in Nokia Research Center. The groundwork was also based on CDT. This groundwork was provided for us by Matti Sillanpää, who had already started the work earlier. Experiences gained from this phase of the work have already been reported in [7].

The main lessons of this phase of the project include the following:

- Expanding a large size plugin (approximately 483 KLOC) without comments is a complicated task to perform.
- Making CDT work inside the Scratchbox sandbox was confusing at first, but in the end it worked nicely due to the IDE integration features of Scratchbox.

IV. RELEASE 2: IMPROVEMENTS AND STABILIZATION

The work on the second release of the plugin version began on 12 September 2005 and was implemented in co-operation with an open source company, Nomovok Ltd. Our main task was to fix bugs and upgrade some features including automake and the package creation wizard. The other line of development was to merge the Nomovok patches with the further developing plugin. The final result was a plugin extended with Glade and Gazpacho support, the possibility to save project templates, change the log editor, localization support, and a PO-file editor plugin as agreed with Nomovok. The PO-file editor is a tool, which uses .po and .pot files for internationalization and localization. This release stage also stabilized the naming and packaging conventions used.

The main lessons we learned during this phase include the following:

- When development is dispatched to multiple locations, it would be advantageous to have a meeting at least once a week in order to keep up on the changes made and what will be done during the following week. In our case, the lack of communication between developers probably slowed the development because it increased the amount of overlapping work. As a solution, we should have adopted a real community working process already in this phase.

V. RELEASE 3: PYTHON

The idea of integrating Python to Laika came into the picture when a maemo Python project was published. The developers of maemo had decided to make Python the main scripting language of the environment. We had to adapt ourselves to that idea and to start considering whether Python integration to Eclipse is possible.

There were already a number of different Python plugins, and in any case, there was no way we could develop the whole plugin ourselves with the resources we had, which prescribed that we would have to rely on an existing plugin in the development. Eventually, we decided to extend the Pydev [8] plugin - created and maintained by Fabio Zadrozny and Aleks Totic - because Pydev was the only plugin with a working debugger at the time.

At the time we started developing Laika Pydev, the Pydev version was something between 0.9.5 and 0.9.7. It seemed like a good and stable ground for extending Laika, but within the few months we spent developing the first experimental version of Laika Pydev, we were faced with numerous updates from

Pydev, which often forced us to change the structure of our extension. The good thing about constant changes is that we knew that the Pydev project had active developers, so there was an opportunity to contact the developer community.

The constant change in the version was not the only problem we ended up dealing with, but we had some technical challenges in our implementation as well. The structure of Pydev is very different from CDT. For example Pydev’s debugger communicates frequently with Eclipse. That makes it impossible to use it from Scratchbox sandbox. The python debugger is located inside Pydev, unlike in C/C++, where the debugger is a part of the development environment inside Scratchbox (Figure 5.). The other big problem we faced was setting up a Python interpreter. Setting up an ARM interpreter was impossible, since the binary code is ARM executable and it cannot be accessed from Eclipse. Therefore all development had to be done with x86 interpreter. The next problem occurred when we tried to access the Pydev source codes from Sourceforge, which frequently failed in allowing us to download Pydev source codes. For that reason we often had to develop blind without source codes, only having Pydev class files with function names and parameters.

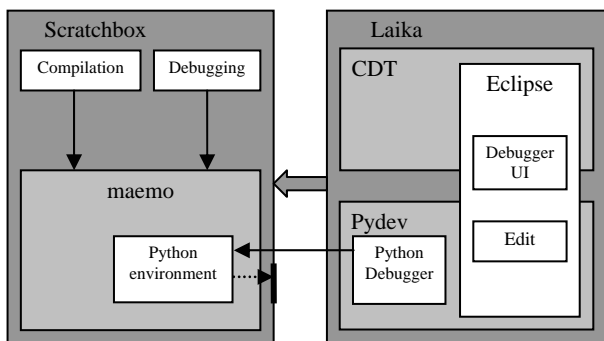


Figure 5. Contents of Release 3

The main lessons of this phase include the following:

- Again the amount of comments in the code was low.
- Sourceforge seems to be insufficient for providing a stable base for code delivery. The ability to participate in different communities and direct contacts to main developers proved to be a practical necessity. This also supports the introduction of features needed by Laika into other systems.

VI. RELEASE 4: GAZPACHO

Gazpacho is an application that helps you to design the graphical user interfaces using visual editor [9]. Gazpacho is designed to work with GTK+ toolkit and is strongly inspired by Glade project [10] but written using Python language. The experimental version of GUI integration was part of the second release of the Laika plugin. Laika supports the Gazpacho user interface builder and enables users to design user interface with the help of the graphical editor. However, the implementation of Laika is very restricted, and it does not support any maemo specific issues yet.

The current maemo SDK includes graphical a user interface library called the Hildon framework that is based on GTK+ library. Moreover, the maemo community is developing a Gazpacho UI development tool that makes it easier to design graphical user interfaces for maemo applications. The goal of the fourth release of Laika is to improve the Gazpacho integration, and support the new Gazpacho UI development tool. While Gazpacho tool is not implemented as an Eclipse plug-in, it can be integrated to the IDE by making some changes to the Laika plugin.

The upcoming release will be planned for maemo 2.0. At this point the future plans for the next release are quite uncertain. The updates will concern mainly Gazpacho UI tool, application installer, Debian packages and of course python. The main developing and planning has been initiated during the summer 2006 when maemo 2.0 became available. The main line of development is to increase the quality and usability of our product and make sure Gazpacho integration is compatible with new versions of maemo.

VII. DISCUSSION

We found that creating open source can be difficult in numerous different ways. The biggest problems we encountered were in expanding existing open source components (plugins), which we believe is a common practice in open source communities. However, although the source code is available in principle, it is not always accessible due to practical limitations. Moreover, even if the code is available, the comments on the code seem to be inadequate for anyone outside the project or completely missing. This is a big disadvantage when expanding open source plugins of different origins. Furthermore, new versions in one part of the product can cause serious problems, which are worsened when integrating multiple different programs with ever changing versions. Problems are more likely to occur when aiming at version compatibility with different versions of multiple programs. Achieving full compatibility can often be impossible without serious effort and resources, especially when the development area is also complex and still evolving. Using existing (not necessarily open source) components with a sandboxed environment complicate matters when bug fixes in the ready components require changes in the structure of the component or features of the component are working fine, but are unsuited for cross-compiling. This may cause pressure to release new concurrent (“improved”) components, which are not updated along with the original component.

When considering Laika, the released versions have grown in complexity, and in future, they are expected to contain more and more conflicting features (mostly in packaging). This will require extreme care in designing features that change packaging rules. The plugin (Version 1.0) was originally released on 16.09.2005 and the bug fix (Version 1.1) was released a week later. It has been downloaded 603 times. The second release (Version 2.0) was released 08.02.2006 and it has been downloaded 321 times (updated 16.05.2006). The experimental release of Python was released on 02.05.2006. Table 2 shows how many working hours have been used for the project during the first year. A time line of different features has been given in Figure 6.

TABLE II. AMOUNT OF WORK INVESTED IN THE PROJECT.

Version	Date	Resources	Working hours
Groundwork	01.2005-05.2005	1 developer, 30h / week (Nokia NRC)	720h
Release 1	05.2005-08.2005	3 developers, 40h / week (TUT)	
	09.2005	3 developers, 20h / week (TUT)	2160h
Release 2	10.2005-01.2006	3 developers, 20h / week (TUT) 2 developers, 20h / week (Nomovok)	1600h
Release 3	02.2006-04.2006	2 developers, 20h / week (TUT)	480h
Total	Now		4240h (4960h)
Release 4	05.2006-	2 developers, 20h / week (TUT)	

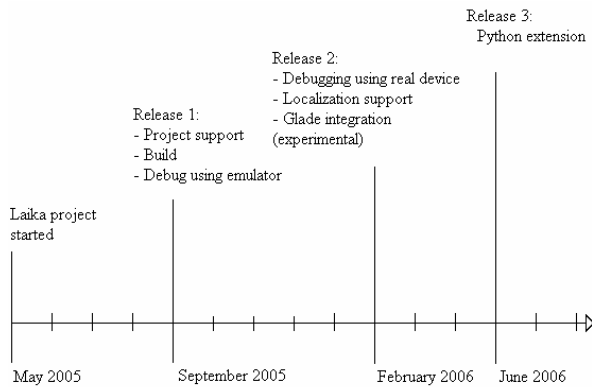


Figure 6. Time line of development

So far, we have received both negative and positive feedback of our project. Feedback has typically been bug reports or questions on how to install or use the Laika IDE. In addition to feedback strictly concentrating on Laika related issues, we have also received some negative feedback about the problems faced when using third party components that are needed for using Laika. For instance there have been problems when installing Java environment and Eclipse to Linux. While surprising at first, we soon realized that to the users of Laika we are in fact acting as integrators of the whole system they plan to use, and therefore the natural first contact point.

VIII. CONCLUSIONS

This paper introduced our experiences on developing an integrated development environment utilizing third party open source components in its implementation. The main conclusion is that it is possible to save resources by utilizing existing open source components. On the other hand we faced some unexpected problems during the integration. The most important ones are listed in the following:

- Lack of documentation of existing open source components, especially when aiming at the selection of a compatible collection of different subsystems.
- Source code of both CDT and Pydev can be regarded as inadequately commented.
- Difficulties to support new versions of third party components, especially when a number of changes were introduced in a short period of time.
- The large number of embedded components rapidly leads to increasing complexity in the integration.
- Usability problems due to the fact that there is no homogenous way in which different plug-ins should be used in Laika; even if we would have a view, imposing it to all the different communities would be impossible.

Eliminating all of these problems is not easy. We believe that the following actions could be taken in order to manage such an effort, if we were to face a similar challenge in the future:

- Start with a mission. Do not aim at solving all the possible problems but have a concrete short-term goal as well.
- Make a strategy on how to integrate different subsystems. For instance, in our case this failed with respect to the sandboxing of Python debugging, which was not considered in the first place when selecting the suitable subsystem. Still, we would have had few other alternatives, if any.
- Select partnership communities carefully. In particular, in many cases it is practical to choose a co-operative community who is still active in developing its project, as this is the best possible way to get support. The importance of this issue is emphasized when only scarce resources are available for development, and it is not possible to make a separate development branch for one's own purposes. Moreover, it may be possible to make other communities to adapt for one's own specific modifications.
- Do not get involved with too many communities. When new versions of component subsystems are introduced, there is a risk that the community cannot adapt to all of them, especially if different version combinations are to be allowed. Moreover, one should also take into account that in order to know what will happen in different communities, active participation is required, which also takes time. This will also be the best weapon against communities where only virtually uncommitted code exists; participate in the development to learn how the system works.
- Use the outcome for something practical, or as [11] puts it, "eat your own dog food". Having developed some sample projects with the tools, as well as having other developers use it in-house has allowed us to learn some shortcomings of the tool set.

- Be open. Have a system in place to allow other developers to participate, and provide an opportunity to get feedback from the users of the system. This will be the final test for any community.

Finally, Laika is not the only project aiming at the creation of an integrated development environment for developing applications on embedded Linux devices. There is also another similar community, Esbox, which has developed a similar system with a different goal in mind [12]. The main idea of the Esbox community is to create an integrated development environment for the Scratchbox building environment instead of maemo platform, which was the starting point of Laika IDE. Despite the different goals, both communities have produced a similar outcome. We believe that the reason for this is that there are actually too few different options on which such an effort could be based on using open source communities as a starting point.

REFERENCES

- [1] Nokia 770, "Nokia 770 Internet Tablet", Available at <http://www.nokia.com/770>, Visited 19/05/06.
- [2] maemo project, "maemo Development Platform", Available at <http://www.maemo.org>, Visited 19/05/06.
- [3] Scratchbox, "Scratchbox – cross-compilation toolkit project". Available at <http://www.scratchbox.org>, Visited 06/05/06.
- [4] Eclipse, "Eclipse", Available at <http://www.eclipse.org>, Visited 6/5/06.
- [5] Anjuta, "Anjuta DevStudio". Available at <http://anjuta.sourceforge.net/>, Visited 15/05/06.
- [6] CDT, "C/C++ development tool". Available at <http://www.eclipse.org/cdt>, Visited 15/05/06.
- [7] Matti Sillanpää. Extending Eclipse and CDT for embedded systems development with Scratchbox. EclipseCon 2006, Santa Clara Convention Center, Santa Clara, CA, USA, March 20-23, 2006. Available at <http://www.eclipsecon.org/2006/Sub.do?id=216>, Visited 07/07/06.
- [8] Pydev, "Python development environment", Available at <http://Pydev.sourceforge.net/>, Visited 7/5/06.
- [9] Gazpacho, "Gazpacho GUI designer". Available at <http://gazpacho.sicem.biz/>, Visited 07/05/06.
- [10] Glade, "Glade – a User Interface Builder for GTK+ and GNOME". Available at <http://glade.gnome.org/>, Visited 07/05/06.
- [11] Joel Spolsky. "Joel on Software: And on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers, and Managers, and to Those Who, Whether by Good Fortune or Ill Luck, Work with Them in Some Capacity", Apress, 2004.
- [12] Esbox "Esbox". Available at <http://tesla.dee.ufcg.edu.br/~omapsdk/>, Visited 07/05/06