# Cross-Development as a Service

Vladimir Moltchanov, Miska Kaipiainen

EmbedOne
Finland
Email: vladimir.moltchanov@embedone.com, miska.kaipiainen@embedone.com

**Abstract**

In modern world embedded systems became an integral part of every-day life. Nowadays there are a large number of professionals and amateurs who develop software for such systems. And the most typical approach to the development is to acquire some sort of development environment from the manufacturer that provides a toolchain for cross-compilation. In many cases, setting it up is a complicated and annoying process of trial and error, providing that the user's hardware is powerful enough to do the builds. EmbedOne is developing a new concept that addresses these issues – "cross-development as a service". This approach enables wider range of developers to perform cross-platform development while keeping the burden of setting up and maintaining the environment at the place where the knowledge is: the manufacturer.

## I. INTRODUCTION

### *A. Cross-Platform Development*

Every piece of software is built to run under one or another operating system [1] on some hardware architecture [2]. In some cases, there is no operating system at all, or the developed software is the operating system. When the target operating system or the architecture differs from the one that the development environment is running on, then cross-development tools should be used to produce the final system images or application binaries[3], [4]. This impose a number of requirements in terms of resources of the development system, development tools, and skills of the developer. In this article we introduce a new concept that is aimed at separating the issues of the purely cross-platform development from the actual software development: "cross-development as a service". This article provides the overall reasoning behind the concept of this service and points out the direction for future development.

### *B. Tools are Important*

In many areas of software engineering the existence of well designed and well functioning development tools may play a major role in determining the popularity of the product. One of the previous studies (conducted already in 1999) have shown that the existence of good development environment may determine the target hardware platform that will be used in the project [5], [6]. More recent survey have shown that for more then 35% of the responders good software tools could be the decisive factor in selecting one or another operating system; yet less then 30% named documentation or technical support as important [7]. For the crossplatform development and embedded systems design good development tools are as important. And according to [8] for 55% the compiler and the assembler is the most important tools. Yet, according to one report [9] only as much as 20% of companies use standardized tools for their embedded development. While 48% don't have any standardized tools or use them very lightly. Yet, more then 50% responded that introduction of embedded software has increased the time to market or development costs [9].

So the reasons why good tools are so important are simple:

- The requirements on the skills of the developers are lower.

- The development times are shorter.
- Easier, hence cheaper, maintenance.

The main benefit is that with the good development environment the developer doesn't have to know all the bits and pieces of the target operating system or the hardware. It is all provided and encapsulated in the development environment. And the developer has only to be proficient enough to develop the required piece of software.

*C. Community is Important*

Open source software [10][11] became an integral part of software development industry. The whole open source ideology is based on the fact that everyone is able to see the code what is installed in your system; and to contribute by fixing broken applications and services. Essentially, it allows creating communities of volunteer developers around certain systems or pieces of software, accumulating the knowledge and promoting the deployment of the software. Despite some challenges of clearly human or social nature, well captured in some related studies [12], the knowledge accumulation and sharing properties of the community are very important for projects with complex architectures that are meant to be extensible. This is also true for the generic cross-platform development environment, where the number of target platforms and possible software components are undefined. However, in embedded applications, it is not always easy to even compile the software due to cross-development environment complexity; and therefore the contributions are typically done only by handful of experts. But when there are clear benefits the new mode of operation will emerge [13].

Nowadays, many commercial vendors have recognized the value of the communities and open source software. Leaders of the different segments of the market such as Apple (iPhone)[14], Google (Android)[15], and Nokia[16] are introducing products with "open" API where software developers can develop extra applications. Most of the APIs and SDKs provided are partly open, and due to proprietary operating system components, they don't support open source distribution of the applications. And even if they would, it is not possible to compile such applications by general public. And in many cases such SDK imposes relatively heavy requirements on to the end-user's hardware. So, while the products are very good and the development tools are adequate, it is the lack of the right mode of operation that holds the community involvement back.

*D. Divide and Conquer*

So far it is possible to summarize that good tools will enable for wider range of developers to do the cross-platform development; and that the community of enthusiasts and professional can accumulate and share knowledge effectively. The next important step is to understand what help a typical novice user may require from more experienced counterparts in order to do the development. One of the first tasks is to setup the cross-development environment. In many cases even to understand the instructions for doing it a person needs to have relatively advanced level of experience with the cross-platform development. Often, this task may take days or even weeks of searching the Internet for answers to the wrong questions, since experience is also required to know the right question. The solution to this problem is very simple and thousands of years old: division of labor. Let the guys who have the knowledge do the job.

The division of labor is already a fact in large or medium companies as they can afford to have in-house specialists to handle particular tasks, and studies similar to [17]have been conducted for some time now. However, when it comes to small scale projects or private enthusiasts, it is pretty much "each man for him self". It is almost imperative that a person

doing the cross-development would setup own environment and it is a “bad news”, if there is a need to work with many different target platforms.

*E. The “New” Way of Working*

In the past years the development processes has been experiencing some changes. The platforms became more powerful, hence more complex systems became available. But the need to setup own environment in order to do the cross-platform builds remained the same due to a single fact: there is no other way to get the job done. Or, at least, there was no other way.

Increase in the Internet connectivity and speed made many software systems to rely on the network-distributed services for updates, licensing, and security. Following same principles it is possible to develop a service that will provide the cross-platform building. Benefits of such service are:

- Access to multiple build environments
- It is easy to start using the service and almost no experience with toolchains is required
- Developer can focus on the actual tasks
- Freedom of visualization/editing tools to use

In essence, this service brings the idea of web-distributed software development (similar to [18]), tool integration [19], and process visualization[20] into the field of cross-platform development. The remainder of this article describes the most important aspects of the service.

## II. CROSS-DEVELOPMENT AS A SERVICE

*A. Components of the Service*

The building process itself involves:

1) Specifying the target platform and the project contents to build
2) Setting up the cross-development environment
3) Getting the sources for the project contents
4) Building
5) Reporting the results

Out of all these tasks only 1 and 3 will remain for the user of the service. The rest will be encapsulated and handled by the service. A simple cross-development service may have typical client-server architecture with some considerations for the network type [21]. The responsibility of the service provider will be to maintain the server and to update the toolchains and cross-development environments for different targets. It will be beneficial for the service also to provide advanced level of result caching [22], version control [23], and reporting.

In this respect, the implementation of the client software for the service could be relatively light configuration management system, that allows the user to assemble together different software components into projects. Hence, the service becomes easily access-able on many types of devices running on different operating systems, as there are no longer preferences/requirements on the high processing power or on the particular set of software

tools. However, service may also benefit from having a good integration with mainstream development environments such as Eclipse [24] or Microsoft Visual Studio [25].

Finally, service should include the community forums and, eventually, the file repository for the knowledge accumulation and transfer. This file repository further enhances the possibilities for sharing the software components, development environments, and other tools.

*B. Service Management Issues*

In order to be successful today a service (doesn't matter if it is public or commercial) should provide a guaranteed availability and an adequate support. As a commercial deployment, cross-development service could be interesting to the hardware vendors or to the specialized companies. This type of deployment will result in a higher level of expertise and responsibility of all involved parties, but will eventually be limited in a number of supported target platforms and software components. Hence the community interest towards the service might be lower.

As an open-source deployment, such service itself may have an adequate support, but will require some sort of defined chain-of-command when it comes to accepting the enhancements from the community. On the down side, it is hard to guarantee that the community will gain enough momentum and experience to be able to provide as good support as processional service. Finally, the issue of trust will always be in the picture, as sources have to be uploaded to the server in order to be cross-compiled. But security measures should always be balanced not to interfere with the knowledge and work sharing.

*C. Benefits from Using the Service*

There are few clear benefits for the users of this service:

- First of all, it becomes possible to start doing the cross-platform development almost instantly. No local build environment is required, just the service client software. The requirements on the processing power of the user machine due to the build process are no longer relevant. And the choice of the local operating system is limited only by the availability of the service client software.
- Secondly, the cross-development environment is setup correctly by the professionals with the adequate level of expertise.
- Thirdly, through the community part of the service, it is possible to get in touch with the active experts in the area or even with the vendors of the target platforms.
- Finally, if it is an open source project, the service may be used as a good development and hosting platform for the project, due to the file repository and the community components.

Combining all above, a piece of software could be compiled to all supported platforms (of the cross-development as a service system) at almost no extra work thus increasing the numbers of users of such software. This is very important for many open source projects; as well as middleware software vendors whose business model is based on providing software to as many target platforms as possible with minimal integration time.

*D. Benefits from Providing the Service*

While the benefits of using this service are relatively easy to define, the benefits of providing such service are more illusive, especially when it is deployed as a non-commercial service.

In case of commercial deployment such service will double as a user support for the platform vendor. In fact, this service will make it easy to distribute updates and fixes in a centralized manner. Problem reporting will become easier and could be almost built-in into the service. These features will provide added value to the vendor's products, besides the fact that service fees could be charged.

However, in case of public deployment, the commercial benefits are not very obvious. The key value of the service itself will be the community around it. Community members will be accumulating technological know-how and experience that could be useful in the commercial projects where the target platforms are in use. Existence of such service will also have added value to the vendors of the supported target platforms.

Another application of the service might be the software distribution. Software that may be run on different platforms; could be distributed in form of service requests that will in turn build the software package for the required target.

### *E. Future Work*

EmbedOne[26] has been developing the first version of the service. It scheduled to be publicly available in fall 2009. One of the objectives of the first release is to verify our expectations on the both commercial and public domain aspects of the service. Currently, there are a limited number of target platforms that will be supported. At EmbedOne we will test basic use-cases for the service and pin-point bottlenecks of the system for the future development.

The next objectives include forming a community and promoting the service to EmbedOne's partners. Only with adequate level of deployment it will be possible to estimate the real impact of the service on the production times and no the learning curves of the cross-platform development projects.

Finally, EmbedOne will continue to improve and enhance the technical performance of the service and extend the capabilities of the client software.

## III. CONCLUSIONS

This article presented a short summary on the most important aspects of the cross-platform software development that have been found in the previous studies. Based on the requirements of the cross-platform development and on the capabilities of the modern Internet, a new service was proposed: "cross-development service". The purpose of this service is to provide the expert-level cross-platform building environments to the end-users, enabling them to concentrate on the actual design and development tasks. After analyzing the implications of different deployment modes of this service it is possible to conclude that the service will generate added value to the target platform vendors in form of expertise and development tools. In the long run such service could revolutionize the way software is developed and distributed. In the future, it would enable users to choose any piece of software (in source code format) and just running it through the cross-development service to produce working software for different target devices. But already at the near term such service could be deployed to support some particular target platforms. Future work will help EmbedOne to improve the service and to validate the expectations that were presented in this article.

## ACKNOWLEDGMENT

The authors would like thank the entire team of EmbedOne for their input in the development of cross-development service concept.

## REFERENCES

[1] A. S. Tanenbaum, *Modern Operating Systems (3rd Edition)*. Prentice Hall, December 21, 2007.

[2] J. L. H. David A. Patterson, *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (4th Edition)*. Morgan Kaufmann, November 10, 2008.

[3] A. M. Michael Barr, *Programming Embedded Systems: With C and GNU Development Tools, 2nd Edition*. O'Reilly Media, Inc., October 1, 2006.

[4] S. Logan, *Cross-Platform Development in C++: Building Mac OS X, Linux, and Windows Applications*. Addison-Wesley Professional, December 7, 2007.

[5] L. P. R. to Maguire], T. M. R. to McGinnity], and L. J. R. to McDaid], "Issues in the development of an integrated environment for embedded system design: Part a: user needs and commercial products," *Microprocessors and Microsystems*, vol. 23, no. 4, pp. 191 – 197, 1999.

[6] L. P. Maguire, T. M. McGinnity, and L. J. McDaid, "Issues in the development of an integrated environment for embedded system design: Part b: design and implementation," *Microprocessors and Microsystems*, vol. 23, no. 4, pp. 199 – 206, 1999.

[7] J. Turley, "Embedded systems survey: Operating systems up for grabs." http://www.embedded.com/columns/surveys/163700590, May 2005.

[8] J. Ganssle, "Examining your most important tools." http://www.dspdesignline.com/howto/218401303, August 2009.

[9] J. Ganssle, "The embedded software industry: Challenges and successes." http://www.dspdesignline.com/howto/218401303, 2006.

[10] "Open source initiative." http://www.opensource.org/, 20, September 2009.

[11] W. Scacchi, "Free/open source software development," in *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, (New York, NY, USA), pp. 459–468, ACM, 2007.

[12] M. S. Elliott and W. Scacchi, "Free software developers as an occupational community: resolving conflicts and fostering collaboration," in *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, (New York, NY, USA), pp. 21–30, ACM, 2003.

[13] R. Kazman and H.-M. Chen, "The metropolis model a new logic for development of crowdsourced systems," *Commun. ACM*, vol. 52, no. 7, pp. 76–84, 2009.

[14] "iphone developer program." http://developer.apple.com/iphone/program/, September 2009.

[15] "Android developers." http://developer.android.com/index.html, September 2009.

[16] "Forum nokia." http://www.forum.nokia.com/, September 2009.

[17] J. Altmann and G. Pomberger, "Cooperative software development: concepts, model and tools," in *Technology of Object-Oriented Languages and Systems, 1999. TOOLS 30. Proceedings*, pp. 194–207, Aug 1999.

[18] M. Baentsch, G. Molter, and P. Sturm, "Webmake: Integrating distributed software development in a structure-enhanced web," *Computer Networks and ISDN Systems*, vol. 27, no. 6, pp. 789 – 800, 1995. Proceedings of the Third International World-Wide Web Conference.

[19] I. Thomas and B. Nejmeh, "Definitions of tool integration for environments," *Software, IEEE*, vol. 9, pp. 29–35, Mar 1992.

[20] C. A. Halverson, J. B. Ellis, C. Danis, and W. A. Kellogg, "Designing task visualizations to support the coordination of work in software development," in *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, (New York, NY, USA), pp. 39–48, ACM, 2006.

[21] T. Zhang and J. Hayes, "Client/server architectures over wide area networks," in *Communications, 1997. ICC 97 Montreal, 'Towards the Knowledge Millennium'. 1997 IEEE International Conference on*, vol. 2, pp. 580–584 vol.2, Jun 1997.

[22] ""wikipedia: Cache"." http://en.wikipedia.org/wiki/Cache, September 2009.

[23] "Wikipedia: Revision control." http://en.wikipedia.org/wiki/Version_control, September 2009.

[24] ""wikipedia: Eclipse (software)"." http://en.wikipedia.org/wiki/Eclipse_(software), September 2009.

[25] "Wikipedia: Microsoft visual studo." http://en.wikipedia.org/wiki/Microsoft_Visual_Studio, September 2009.

[26] "Embedone." http://www.embedone.com, September 2009.