(CE Linux White Paper)

How Community-Style Innovations Have Changed Embedded Software Development

CE Linux Forum

This white paper holds a message directed not only to software engineers directly involved in embedded software development, but also to anyone who values software as an important part of embedded system business. By reviewing the activities of the CE Linux Forum since its establishment in June 2003 as a forum for developers of Linux-based open source software, I will discuss the potential of community-style development, and the excitement that those of us within the Forum are beginning to feel at the prospects ahead. Open source community-style development demands our attention as an efficient, flexible development method for embedded software that is becoming increasingly complex and sophisticated.

Community-style collaboration is not appropriate for all types of embedded software development. It is most conducive to the open source software epitomized by Linux. However, the underlying principles of this collaborative development style hold a wealth of suggestions that can be broadly applied in a corporate or organizational environment to increase the efficiency of closed software development, as well as to improve quality and to develop human resources. The open source community is expected to have a growing positive impact on embedded software development now and in the future.

In this White Paper, I begin with a description of open source values and the allure of community-style development. Next I will introduce several examples of achievements that are being made by the CE Linux Forum, an open source development community for embedded systems. It will be from this background that I will then discuss the importance of embedded software engineers and related corporations and research organizations to work within this community and suggest actual ways to participate. At the end of the paper, I will also provide arguments to support this view, first from the perspective of corporate or organizational managers, and next, from the viewpoint of embedded software developers.

1. What are the challenges facing embedded software development and how can open source software help to surmount them?

Embedded software is becoming increasingly important in product development. Operating systems and other core software are like the centerpiece that holds the ribs of a Japanese fan together. Everyone involved wants to develop the preeminent embedded software for a new device and ensure that it can be continuously updated long into the future. Collaboration through the open source development community is the best way to make this happen.

The waves of the IT revolution are changing the embedded system business. Customers' dreams continue to soar, and developers struggle to be the first to make those dreams a reality. Embedded devices are rapidly becoming more sophisticated and the required specifications more complex. That is, software engineers, who carry the burden of the IT revolution on their shoulders, face a myriad of difficult demands. They are rightfully expected to get the most possible mileage from the hardware such as processors, memory and interface. At the same time, they are expected to develop software that can contribute to stable operation, superior usability, fail-proof security and 10 years or more of long term usage. On top of that, the expectation is for flexible solutions that can be used with a variety of hardware configurations, while withstanding the most rigorous of environments and providing the ease of use for non-specialist users. It is the raison d'etre – the destiny - of embedded software technology to come up with creative solutions that meet all of these conditions. This is a new and difficult challenge that could not even have been imagined under conventional software technology, and is truly where software engineers can show off their programming skill. When this challenge is met, it will be possible to design software products that fully meet customer expectations.

There is no longer enough time to develop sophisticated embedded software from scratch for each and every new product. Looking at the history of the computer's evolution, there was a period when core software, such as operating systems, was considered to be most crucial. Today, embedded software finds itself in exactly the same position. There is no room for doubt about the importance of core software such as operating systems that can evolve not only to meet the needs of current embedded software but also to meet the needs of advanced, next-generation embedded software, as those needs become clear in the years

ahead. How then can we achieve such software? In-house development is certainly one way, and may even be the best choice in terms of de-mystifying the technology in your company's core software system. If you have the programming talent in-house, then this is a real option. The drawback, however, is that not only does core software require continuous maintenance; it also requires constant updating and modification to keep pace with the variety of new devices appearing on the market. If the core software has been developed in-house, all of this post-development work forces the company to bear a major burden in terms of enormous human resources and cost, which could be too high of a risk.

A second possibility is to purchase proprietary software from an outside source. This has the major advantage that all support can be entrusted to the developer. Even if the supplier opens up the source code to you, however, it will be extremely difficult for your in-house engineers to completely understand the complexity of software they didn't develop. And, if by some chance the developer discontinues the technology or decides to stop its support, there is a real danger of your core software disappearing into a black hole, because no one who knows it inside and out is around any longer. In the case of embedded products, it is accepted practice for the manufacturer to assume full responsibility for all technical aspects of the product. Conversely speaking, it is unacceptable for the manufacturer not to fully understand every aspect of the internal workings of the products it releases. This applies to core software as well, of course. Can your company afford to take the risk of its core software falling into a black hole that may then spread across its entire operations? That's a real possibility when you choose software from an outside supplier.

Open source software offers the possibility of a solution to all of these problems. The more software developers join the open source community, and the more active it becomes, this possibility expands by leaps and bounds. To software engineers, developing their own core software optimized for use in their environment may be the ideal. However, this is totally unrealistic when you consider the enormous amount of headcount that must be retained in order to continuously maintain software as complex and sophisticated as operating systems are. Only in very special cases, would a company be able to financially justify having such unlimited access to engineers in-house.

However, what would happen if there were a group of like-minded software engineers who wanted to collaborate on core software development, despite the barriers of national or regional borders and corporate or organizational boundaries? Even if your company does not have the human resources to develop and maintain core software, or even if your

company's in-house engineers are all expert in a very specific area of core software, the potential for collaboration is there unless you are trying to build completely distinctive and unique software. If it is possible to take core software that might not be the exact match for your company's products but can be tweaked afterwards into something that is more than satisfactory, then collaboration can make a big difference from starting from scratch on your own. Think for a moment about the true value of using open source software. Is it purely financial, that you can expect great savings on software licensing costs? Of course, that is one advantage, but for embedded systems, there is another major advantage – you can build your products using your own core software, every detail of which is known to you. You are therefore able to market that product with complete confidence, and rest assured that the consumer will enjoy and benefit from it.

The open source development community is a group of individuals who share the same values, and based upon those values, are willing to collaborate for the advancement of software development. The Linux community, for example, is a group of people who share the value of developing Linux into the best operating system that exists. The developers in the CE Linux Forum collaborate based on the shared value of building core software perfect for embedded systems. Of course, Linux is the center of this core software. By participating in the community, and making technical advances within the community, the shared values of the community accelerate the achievement of superior results. It is by far the most realistic of strategies.

Today's embedded devices carry multiple functions of great complexity. For example, a terrestrial digital television provides a network interface, a web browser, a flash memory device, a viewer for digital photos and many other possible features. It's no exaggeration to call such televisions 'computers masquerading as TVs'. In order to create such products, product designers have to depend on experienced engineers outside the area of embedded software for programming features such as network protocol stacks and file systems. One of the major attractions of Linux-based open software family is that it includes a myriad of advanced technologies that have been polished and proven in a variety of fields, ranging from embedded systems to desktop computing and server applications. We are now seeing embedded system technology that migrated to another field of computing. It was polished and refined and now is back in our embedded systems although when we developed it, we could hardly imagine how it could be implemented outside embedded systems. The power saving function is a good example of this kind of technology. It is a very important part of designing embedded systems, but we found that it can also make a great contribution in

controlling power consumption in enterprise server systems and blade servers. Engineers from the miniature world of cell phones are working together with engineers from the world of giant systems such as blade servers! Hard to imagine, but this is the wonderful world of open source development!

In addition, open source software can be maintained over an extremely long period of time. The reason is very simple: all information is in the public domain. In the case of proprietary core software developed by one company, there is nothing anyone can do if that company becomes unable to provide support for that software. The possibility of this happening is a real fear for embedded system manufacturers, since their products will be in use for many years to come. With open source software, however, it is not only the source code itself which is public. All valuable background information, including the content of technical discussions, are saved and made available to the public, providing a wealth of resources when trying to resolve a technical issue. Even more rewarding is the wealth of information that can be obtained by directly contacting the developers of the software in question. It is usually not easy to solve problems detected in software that has been out on the market for a long period, but with open source, it is not impossible.

2. What does the success of the CE Linux Forum mean?

The open source development community for embedded software (CE Linux Forum) is showing steady results. Greater community collaboration is directly tied to the development and enhancement of advanced core software for embedded devices.

Many achievements, which could only have come from an open process, have been born out of the CE Linux Forum community. One example is the pursuit of faster system startup times, a pressing issue common to all embedded systems. Of course, the importance of increasingly high-speed startup of the operating system is obvious, but reducing the time application programs need to start is equally important. In order to facilitate this, prelink technology was developed. Instead of linking relevant libraries every time a program is run, an application binary code representing the prelinked executable files is used to cut the time required for linking. And here lies a great example of the strengths of the community style of development: Finding a report on the successful implementation of this technology into a CPU architecture, supported by actual test results, one of the Forum member asked a very

simple question, “Why doesn’t the CPU I use have prelink?” Then, almost immediately, communication among the developers of the prelink technology, engineers engaged in CPU designs and users of CPUs was established and about 6 months later, implementation of the technology into another CPU architecture was completed.

In another example, a company uploaded to the CE Linux Forum server the result of de-bugging the tool chain for the command set specific to the Linux version 2.4. Several months later, when they tried to do the same for the latest version 2.6, they were pleasantly surprised to find that an engineer from another company had already done it based on the information they disclosed before, and in fact presented the results to the Forum. In other words, the company that had performed de-bugging with the older version found that the favor had been returned to them from another company! Both companies that have publicized the outcomes of their efforts were winners in this situation.

There are other examples like this, not only from the development side. Through the CE Linux Forum, we are beginning to see the birth of a new style of software engineers. Partners in a global community, these new engineers are producing extraordinary results in a short time period. Although these engineers hold impressive skills within their area of specialization, the skills needed to develop software for today’s embedded systems require extensive knowledge across a broad base. It is almost impossible for one person to cover all areas. That is where these new engineers know to seek out the strength of the community and begin dialoging with other members of the community, surmounting this issue through collaboration. When these engineers encounter a difficult problem, they think nothing of sending an email to the other side of the world in search of an answer to their questions. The borders of one country are too small for them! By the next morning, hints and advices come flooding in from all corners of the globe. In some cases, not just hints and advices – the solution itself could be found among the emailed replies. In contrast, these engineers would share their expertise if an email from overseas asking for suggestions is received. It is situations like this, where one person helps another that is truly the community-based ideal of a network of engineers. In the future, the world of embedded software devices will undoubtedly become more sophisticated and complex, but the power of this community holds the possibility of incredible results.

3. What is the value of working within the larger community?

Core software is too large and complex to be handled by embedded software engineers only. Collaboration with the larger community is a must. CE Linux Forum believes in the importance of the broad developmental perspective gained through such collaboration, and has been an active supporter at the Linux Symposium held every summer in Ottawa.

A community is alive. When Linus Torvalds, the progenitor of Linux, said “Linux is evolution, not intelligent design,” he expressed his concept of Linux as a living organism which evolves through the contributions of the development community. The Linux development community can be compared to a “bazaar” as described in Eric Raymond’s book *Cathedral and Bazaar*: a place where people gather from different places to barter their own knowledge in the marketplace. Actually, a development community is the exact opposite of building a magnificent cathedral under one person’s leadership. In other words, there is no one who can precisely describe the road map Linux will follow or predict what form Linux will take in the near future. Open source development communities such as Linux do not have a certain individual or leader who draws out the road map and delegates tasks to software engineers. Yet, Linux is growing despite this. In fact, Linux is evolving furiously. This is solely because many engineers from around the world have actively contributed to the community in the hopes of improving Linux.

Fortunately, in these past few years, Linux has started to evolve into the best operating system for embedded devices. As a result, Linux is being used extensively as the foundation for embedded software in a wide variety of consumer electronics products including intelligent home appliances. This may be due to the fact that embedded devices became an integral part of the Linux evolution history through the active participation of embedded software developers in the open source development community. In actual fact, many functions convenient for the development of embedded devices have been adopted in the Linux mainline.

Does evolution stop here? Has the operating system for embedded devices achieved its maximum value? No, not only is Linux still evolving as a whole, but even within the embedded device area, there is still room for improvement. For example, we see possibilities for further development corresponding to new communication infrastructure,

advanced devices, discovery of a revolutionary new file system, and construction of a more secure and flexible system environment.

On the other hand, if participation from embedded software engineers in the Linux community slacks off, embedded software will be left behind in a flash. The same goes for other open source development communities - without the contributions of embedded software engineers, evolution stops. If the crossing of knowledge is diminished, there might be a future where the “gene” of embedded devices is no longer found within the open source software development DNA.

4. What can be done for the development of advanced open source software for embedded devices?

The answer is easy – join in and contribute! The CE Linux Forum offers some concrete ideas for business owners and executives, and others for embedded software developers themselves.

If you are an engineer, perhaps the easiest and most productive way to contribute is to post in the Forum the code you developed. Or, as a first step, why not start by sending in a patch when you have made an improvement to an existing software program. Of course, there is nothing wrong with simply using open source software without making a contribution of your own. If you make some sort of improvement to open source software, you can simply provide the source code when asked. However, in the case of embedded software, the user is not generally another software developer. Changes in only a few lines of code can mean a lot. Please give them back to the community, and to your fellow developers. If you do so, those who are involved in embedded software development will put them to good use. This type of sharing is what contributes to the evolution of open source software, and is a first step to fostering the “genes” of embedded devices in the open source software lineage.

This White Paper will be read by people from many different perspectives. Acknowledging this, we have added messages below, addressed to some of those diverse perspectives.

(1) For Business Owners and Executives of Companies in the Embedded System Industry

---

The community creates value in many ways. Why not give your support to engineers who want to participate in the open source community?

The open source community accelerates the development of technology. At the same time, it accelerates the development of talent in your organization.

When software engineers from your company participate in the open source development community, there is a concern that valuable proprietary technology necessary to maintain a company's competitive edge in the marketplace might be leaked. However, I don't really need to remind you that there is an important licensing agreement called GPL (General Public License) that requires the source code of open source software to be publicized. As soon as a product hits the market, the user can see the source code. If it is going to become open knowledge anyway, why not open it up at an early stage? This seems to be the more practical step because software such as the operating system is not a direct factor in market competition but needs to be efficient and high quality. In many cases, moreover, there may already exist someone in the community who is not only working on something similar but is almost finished with it. Even if this is not the case, you may find some valuable technical leads, someone to partner with, or some other possibility that will make your development more efficient. The global Linux community is extensive, and the level of technology that you can come across is top notch. The open source community is the one sure place where technology is infallibly advancing at an incredibly rapid pace.

The collaborative spirit of the open source community, where the highest level of developers in the world provide mutual support and mentoring to others in the community, helps develop your in-house talent. There are already numerous companies who have realized the value of the open source community as an arena for workforce development and training. Compared to learning within the tiny pool of a corporate organization, the vastness of the open source community is like the ocean. Software engineers who have experienced the mentoring, knowledge-sharing, cooperation and respect of their peers within the community make up the type of talent necessary in the future, when embedded software engineering will be dealing with an anticipated increase in complexity and sophistication of devices that will be beyond any one engineer's scope. The open source development community is the perfect

incubator for programming talent. It is also beginning to show its worth in the development of next-generation managers, as people can learn how to manage outstanding talent from the leaders of the community. Let's not forget that this type of demanding technical environment has a tremendous motivating power to an engineer.

Some companies are using the open source community approach to organize development groups made up of highly qualified software engineers with the sole mission of working on the basics of core software development for new product lines which are still far in the future. This type of organization acts as an in-house university, nurturing and developing the next generation of talent and skills. Graduates of these groups will be sure to become key leaders within the corporation through their dedication to connecting people and promoting shared use of software development techniques across organizational boundaries. It is abundantly clear that this will make an enormous contribution to software development throughout the corporation. The attitude of sharing is not just necessary within Linux-based open source software but is also indispensable for development of in-house, shared usage software modules. It is, in fact, effective in raising the level of software development skills in all parts of the organization.

By the way, Linux has distributors who take the achievements made within the community, package them with related software, and supply/deliver these packages to the user. There are also distributors who focus on embedded systems. Here, some people might think that it is enough to purchase core software from such distributors. However, things are not so simple. For industries such as enterprise server systems or desktop computing, where a high level of hardware standardization has been achieved, distributors can play an effective and efficient role. With embedded devices, however, it is difficult to adopt a single hardware platform as standard. In the vast majority of cases, the hardware must be thoroughly customized to function in a variety of scenarios, making it difficult for distributors to operate in this area. Understanding this background and looking at examples of the successful use of Linux in embedded devices, companies that use Linux most effectively are those who don't blindly rely on distributors but see them as a strategic factor, working together with the distributors in development. This same thing can be seen in relationships with suppliers of development support tools and CPU architecture. Companies which grasp the importance of collaboration - or the spirit of helping each other - have a real strength within that grasp.

I sincerely hope that, you give your approval if one of your software engineers wants to join the open source community. In particular, there is no reason to second guess your decision

if he is involved in the project for improving software that falls under GPL or other licensing agreement where the source code must be publicized by the licensing terms. In reality, most members of the open source community are working as software engineers at private corporations during their working hours and active in the community in their spare time. The companies well understand the value and advantage of such activities to their employees.

(2) For those working directly in embedded software development

---

The best thing to do is to jump right in and participate in the community. Here are a few hints on taking that first step.

First I'd like to throw out a challenge to you – why not try it? It doesn't matter whether it is only a slight modification to a function or a debugging technique, or whether your code is still in the beta stage or in the conception stage. In fact, the community welcomes the code at the very early stages even more than something that is already nearly perfect. The only advice I would give is to make it something readily understandable to other people. By that, I mean, clarify the reasons that you want to develop the code (or why you were forced into the position where you needed to develop the code). Why is this information helpful? By letting others know why you are working on the code, they will understand what functions you want. And if they know why you were forced into the position, they will be able to look at the existing code and point out the problem it contains.

Saying that there is a problem in the existing code is the starting point for communication with the developer of that code. It could very well be that the developer of the code is in the middle of implementing a brilliant solution based on a wonderful idea he's had. In that case, you can help out in the programming so that it can be finished and published faster for the benefit of many people waiting for a more precisely coded result. Or, it could be that the only a small change to the settings of the existing code is required to reach your objective.

On the other hand, you may find that another engineer in the community points out a new defect in the code that you were not even aware of. And, the solution you come up with to correct the defect may, to your regret, be rejected by the community! Even if rejected, however, please don't be discouraged – the community will honor the spirit of your contribution. You can be assured that it will have a positive effect at some time in the future.

Andrew Morton, the central figure in the Linux development community, once said in a speech that “it is not necessary at all to send highly sophisticated and nearly perfected code to the community. For the community, such code is very hard to deal with.” If there are too many diverse technical functions loaded into the code to be shared, the discussion lacks focus. To get the most from community development, it works better to divide the code into individual elements because this fosters discussion. Especially, if something is still in the development stage, you can invite constructive suggestions from many different points of view, making it easier to gain even better ideas. Morton’s heartfelt message to all software developers was: “Relax! It doesn’t have to be perfect”. Once accepted by the community, the code is absorbed into the evolution of the open source tree. Then, it can grow and mature as people use it, improve it, and maintain it. When the first person to develop a code goes back to use it again some time later, he or she may end up surprised at the advancements that have been made!

As the open source community grows, it gains diversity, with software engineers from a variety of different backgrounds and a variety of different countries. For example, in the Linux mainline development community, there are people dedicated to enterprise server development, people who want to see Linux gain popularity in desktop computers, and people who are pushing for the development of low-cost, high performance computers that could become the workhorse of the developing world – all sorts of people! And of course, that includes engineers writing programs for Linux-based embedded systems, too. In this circle of like-minded people from all over the world, the last thing you want is people with an arrogant attitude, who feel that they know it all. You need to be able to listen to ideas from a variety of people and take away the essence of the technology. With this sort of attitude, you will gain an incredible breadth of knowledge.

Embedded software engineers often find themselves working incredibly hard during a product’s peak development stages. I doubt that anyone in such a situation could find the time or inclination to contribute to community activities. This means that the best timing is either right as you begin developing the code or else after product design has reached a temporary breathing space. If you send the code right after development has started, there is a real possibility that the burden of programming can be drastically reduced. You can expect replies from the community like “If you need that type of code, why not try this?” and you may very well find that the suggestion is right on the mark, or perhaps needs only slight modifications to fit your needs. Then, your programming tasks could be nearly completed! If you are involved in the development of core software within a corporate research and

development division or work at a university research organization, why not consider placing open source community development at the center of your development projects? Without a doubt, research groups which have the power of community knowledge behind them will earn the trust and respect of the product designers they work with.

If you feel that the code that was developed at the peak stages of product design could be of help to others, send them to the community after the design project has been completed or when you have time to catch your breath. It's not too late for the community to benefit from the value of the code. I've sometimes seen pathetic cases where the developer himself, taking a look at the code he wrote during the mad rush of product design, asks himself why he made the coding that way. Since the open source community is the public domain, you need to send in code that anyone can understand. In other words, sending code to the community is like a message to your future self – it is a great way to review and rethink what you've done. To be understandable to anyone, not just yourself – that's an important gift to your colleagues, the people in your company in other divisions, and anyone else who will be using that code.

In the Linux communities, certain coding rules that reflect the experiences built up during years of activities have been established. They are flexible and come from common practice. This means that the code you developed could quite possibly become even better, if you send it to the community.

In the case of the Linux mainline development community, do you find it hard to just show up one day? If so, why not join us at the CE Linux Forum? We are all developers who found the answer to an embedded software problem or who had the solution to someone's problem. We also welcome non-members to register on the software engineer mailing list or to use the Wiki page and other resources.

5. In conclusion

In this white paper, I have focused mainly on the Linux kernel development community to explain how the involvement of a community in the development of the embedded software works. Of course, the Linux kernel is not the only area that can benefit from the community-style development. There are many other examples: for instance, graphics libraries for embedded systems are suitable for community-style development. CE Linux

Forum welcomes any topics that are useful for the development of open source software for embedded systems, whether it is based on Linux or not.

The community style improves the developmental power by networking person to person, sharing knowledge to knowledge. In the embedded software development area as well, this style is expected to be the wave of the future that brings the maximum result.

About CE Linux Forum

The CE Linux Forum is an international open source software development community (a nonprofit corporation based on the laws of the state of California) established in 2003 at the joint proposal of Matsushita Electric Industrial Corporation and Sony Corporation. It is an forum of like-minded software engineers dedicated to the development and enhancement of Linux-based embedded devices through the irreplaceable resource of shared knowledge. These engineers bring their ideas and finest skills to such missions as decreasing system size, startup/shutdown time, and power consumption; improving compatibility to various CPU architectures, and developing middleware. Although CE stands for consumer electronics, engineers from many different fields are participating in the embedded system development. At the same time, the CE Linux Forum cooperates with many open source communities, including the Linux mainline development community, with the aim of contributing to the advancement of Linux. The CE Linux Forum cherishes the freedom of open source software and community-style development, providing a concentration of high-level of software technology that surmounts corporate and national boundaries.

On the CE Linux Forum homepage, you will find various information such as guides to various events, links to its Wiki site, and registration procedures for the free mailing list. We look forward to your contribution.