UNIVERSITY OF GOTHENBURG

Migrating from Proprietary RTOS to Embedded Linux: An Empirical Study

Bachelor of Science Thesis Software Engineering and Management

OSCAR MUCHOW
DAVID USTARBOWSKI

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, June 2014

Migrating from Proprietary RTOS to Embedded Linux: An Empirical Study

OSCAR MUCHOW
DAVID USTARBOWSKI

Examiner: HÅKAN BURDEN

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2014

Migrating from Proprietary RTOS to Embedded Linux: An Empirical Study

Oscar Muchow and David Ustarbowski

Department of Engineering and Computer Science, University of Gothenburg, Gothenburg - Sweden

Abstract. Embedded systems and the open source operating system Linux are some things that has been going hand in hand for a long time now. Companies using Linux for their embedded products are praising it for being cost and time efficient when it comes to performance and maintainability. Another solution for embedded systems is a Real-Time Operating System (RTOS).
The goal of this this paper was to investigate whether a traditional proprietary RTOS can be substituted with embedded Linux, and if this kind of migration can lead to reduced licensing costs and increased general quality of the system.
We used a qualitative research method for this case-study. The investigation was conducted with interviews as the main source of information. The result of this study was an empirical model we named 'Embedded Linux Adoption Model'. We concluded that in many cases a proprietary RTOS can be substituted with embedded Linux without affecting the critical needs of the system. The study also showed that many embedded system developers are very receptive to open source solutions and could think of contributing to the community.

Keywords: RTOS, Embedded Linux, Linux, Migration, Adoption

1 Introduction

The use of embedded systems is a rapidly growing industry, resulting in changing requirements for the market. There are many organizations that today use proprietary Real-Time Operating Systems (RTOS) for their products with high licensing costs as a side effect, some example products are smartphones, elevators, dishwashers, door locks, and cars.

A successful migration from an RTOS to embedded Linux can bring a lot of advantages for an organization; reduced licensing costs, increased maintainability, among others [13,31], especially for organizations that uses obsolete systems. A general problem in this kind of migration is about critical factors like execution time and how they can be solved or substituted without affecting crucial aspects of the system [8].

RTOS has for a long time been seen as a critical component of embedded systems holding an extensive role for integrated circuits [14]. RTOS provide the

abilities of structuring the system on hardware platforms and can be customized so it guarantees response to events within microseconds [14]. Linux does not have the same real-time performance but can in many cases be tailored to perform sufficiently for the intended task. Although, it is important to understand that the results of these customizations can vary from time to time, thus the timing performance of Linux can not be guaranteed. However, there are solutions that take care of this. By using Linux as a process in an RTOS hard real-time constraints can be achieved. This gives hard real-time on the processes that needs it, and Linux, with all its advantages can be used on processes that do not need real-time pre-emption.

There are trends showing that the market is interested in large scale software systems which most likely results in a limited amount of dominating RTOS [14]. With escalating complexity in systems and hardware platforms, and increasing demands for faster time-to-market, the need for general-purpose software platforms such as Linux became high [15]. Modern communication systems require advanced services which closed software systems cannot provide without additional development [15].

This study was based on investigating the needs and challenges behind a possible migration. Also, we wanted to research on the opinions about open source solutions within different organizations that today use RTOS. The study was conducted according to the guidelines in [23]. As a part of the research, interviews were conducted together with relevant stakeholders or organization. The interview questions are based on the ISO/IEC 25010:2011 Product Quality Model [4] where the main points from this model suitable for embedded systems where taken into consideration when the interviews were conducted.

Our goal with this paper was to find what perceived quality aspects needs to be taken into consideration when a possible migration from RTOS to embedded Linux is to be conducted.

2 Purpose of the Study

The purpose of this case-study is to explore the possibilities of migrating from RTOS to embedded Linux and discover the challenges and advantages of such a migration.

Today many organizations are using proprietary source RTOS, which gives the organization expensive licensing costs. This could be solved by introducing the open source operating system embedded Linux.

We investigated the possibilities of migrating from RTOS to embedded Linux solutions to ascertain if there are any specific challenges with this kind of changes. Existing systems, needs, and cost should be taken into consideration when ascertaining if switching systems may be possible and worth conducting.

The most important thing to have in mind when conducting a migration from RTOS to embedded Linux is, a standard Linux solution can not guarantee any response time, thus it is not a hard real-time system.

The research questions we have decided to answer for this study is the following:

- RQ1: What are the perceived quality gains of migrating from RTOS to embedded Linux?
- RQ2: In what cases are RTOS needed?

This research will help to understand the possibilities and challenges that evolve from migrating from RTOS to Embedded Linux, which in turn will help to understand the quality trade-offs when comparing RTOS and embedded Linux.

3 Case Company Description

This case study [23] was conducted in collaboration with HiQ [12]. Inquiry methods include reflections on data elicited by existential investigation of different companies within a wide variety of sectors in the embedded software industry and investigation of existing research.

3.1 Collaboration Company

HiQ Göteborg AB (which is a branch of HiQ International) is a consultant company with over 300 specialists with expertise within fields such as software development, quality management, project management and business development, mobility and web development [12]. The branch of HiQ that we are collaborating with operates for a large variety of companies within the embedded industry.

Their main goal is to sell competent staff to help companies with implementation and development of embedded systems. HiQ provided us with contact information to different companies that we conducted our study on. These companies all develop different types of embedded systems, using both RTOS and embedded Linux.

3.2 Interview Companies

Company 1 A large Swedish company within the Telecom domain, providing communication technology and services for Telecom operators. They have a large amount of products for different markets within the Telecom sector, and are all the time trying to shape the future in a rapid changing area of technology. The company has a total of over 110 000 employees globally. This company currently run both RTOS and embedded Linux in their products.

Company 2 A medium size company within the Telecom domain, specialized in Mission-Critical Communication, focusing on developing wireless solutions for healthcare, penal system, retirement homes, hotels, and industries. The company has approximately 1 500 employees globally. They currently run both RTOS and embedded Linux in their products.

Company 3 A medium size Swedish industrial appliance company within the industry domain known for their professional high quality products for welding and cutting. The company has around 8 000 employees globally. This company currently run an old RTOS solution and are discussing a possible migration to a solution with both RTOS and embedded Linux.

Company 4 A large size worldwide Software development company focusing on developing solutions for the automotive industry. The whole corporation has a total of over 160 000 employees globally. This company has recently done a migration from RTOS to embedded Linux.

4 Background and Related Work

There are different options when choosing what type of solution to run on an embedded system, where the simplest of these is a commercial distribution, this gives you a tested kernel as well as support and maintenance. Another option is to use a complete open source solution, this option is free to download and open source solutions are usually licensed under GPL [9]. There exists many different open source options, for example FreeRTOS, ChibiOS, and embedded Linux. This study is limited to embedded Linux. Usually open source solutions are well maintained and supported, but there are no guarantees for that to be the case. The last option available is to create your own solution, this can be a quite hard and time consuming development, but with the help of Yocto Project [22] the procedure has been made somewhat easier. When building your own system it can customize to fit the actual needs you have for the system, although you have to maintain the system yourself.

4.1 Proprietary Real-Time Operating Systems

We have decided to limit our research to grouping all Proprietary Real-Time Operating Systems (PRTOS) into one entity. Since there are so many different PRTOS, for example VxWorks, Windows CE, and threadX, available on the market a more general approach to PROTS where taken. The major attribute of PRTOS for us when doing this research is that they all have a licensing cost, whether it is single license or per unit license.

A benefit of PRTOS is that when you buy it you usually get support included in the purchase. As we discussed in the section above this option is also available for some Linux solutions.

4.2 Open Source Software

Open-source software development has for a long time being impacting the software industry and organizations [11]. The way of developing open source software with help of the community have inspired many organizations to evolve their

development process into more collaborative ones [7]. It is possessed that the availability of open source software opens up the possibilities for shorter adoption time, increased innovation, faster time-to-market, and reduced costs [3, 16].

GNU General Public Licence (GPL) is today the most widely used license for free software. The main purpose with GPL is to give freedom in usage, sharing and modification of software. Anyone is free to investigate, develop, and distribute it further without any additional permit [1, 9, 21]. There are three existing versions of GPL: GLPv1, GPLv2, and GPLv3.

Linux and GNU Since a very early version Linux is licensed under GPL which open the possibility for everyone to make custom builds for commercial use. Linux was in the beginning a hobby project by Linus Torvalds during his studies at University of Helsinki, the development was made on MINIX with the GNU C Compiler. He announced his work on 25 August 1991.

The work with the Linux kernel continued and in the release notes of version 0.12 he suggested releasing the kernel under the GLP [27]. Linus and developers at GNU started working on Linux in order to make it a fully functional operating system. Today, 23 years later and with a huge contribution from the community, Linux is a widely used operating system. The fact is that more than 95% of the fastest supercomputers in the world run some kind of Linux distribution [26]. And of course Linux is not only being used as an operating system for desktop computers, is well suited for embedded systems and is being used as OS in devices such as mobile phones, tablets, video game consoles, routers and infotainment systems.

A standard Linux solution cannot guarantee any response time, thus it is not a hard real-time system. However, during the later years, and with kernel 2.6, many improvements have been done to Linux, for example, the support for kernel pre-emption [10]. Before this was added into the Linux kernel system calls could not be interrupted, causing other processes to be blocked for a long time. One other significant improvement that was made to the scheduling algorithm, was that now it runs at a constant time.

Yocto Project The Yocto Project [22] is a powerful and complete development environment including documentation. Yocto is open source and gives the user the possibility to create custom built Linux for embedded systems [15]. It was founded in 2010 thanks to collaboration between hardware manufacturers, open source operating system providers and electronic companies. The main idea of Yocto Project [22], is to tailor-make embedded Linux to suit the specific organization needs. Yocto is an 'Open Source embedded Linux build system, package meta-data and SDK generator' [22].

4.3 Embedded Linux

There has been an interest for making Linux work on embedded devices for a long time, but it was not until the RTLinux project was started back in 1997, as a masters thesis by Michael Barabanov [2]. The aim of his thesis was to give Linux real-time properties. In RTLinux the kernel runs as a process underneath a real-time kernel. This kernel then handles all of the real-time threads as well as the scheduling of the normal applications and Linux, this is the common set-up of Embedded Linux distributions.

During the end of the 1990's Lineo, Montavista and other big organizations started to push for having Linux in the embedded industries [29–31].

With the correct hardware it is possible to run a general purpose Linux distribution, for example Debian Linux (Desktop) needs at least 128 megabytes of RAM and 5 gigabytes of hard drive space [6]. There also exists Linux distributions for Raspberry Pi, one of these distributions is Tiny Core Linux, which only need 46 megabytes RAM and 12 megabytes hard drive space [25]. The last and smallest Linux variant is the embedded one, an example being uClinux, this embedded Linux version uses, in its minimal configuration less then 300 kilobytes of hard disk space, thus it does not really need a hard drive, but can instead be loaded into the bootloader of the embedded product [20]. If the embedded system does not need real-time pre-emption embedded Linux could be a good choice.

4.4 Proprietary to Open Source Migration

Existing literature describes that there are a lot of relevant discussions and reports of open source software, and from actual migrations from proprietary to open source software reported. Below are some views and cases mentioned.

According to Ven et al. [28] there are many different claims and counter-claims when it comes to using open source software. In their case study they show that there are different ways of thinking about advantages of open source software in different organizations [28]. They provided an analysis, which shows, organizations interpret the usefulness of open source software depending on how and if they are using open source themselves. Furthermore, there exists an organizational and contextual factor that has to be taken into consideration when deciding upon adopting open source software. The authors emphasize on that organizations should not adopt open source software because others are doing it, or because of different claims in literature [28]. Also, Ven et al. mentions that different advantages and disadvantages, which are widely claimed, should not be taken for granted, but instead be investigated based on 'organization-specific context' [28].

In the sample discussed by Ven et al., six organizations pointed out that a migration from Unix to Linux is easier to perform then a migration from Microsoft Windows to Linux [28], this because Linux is Unix-like. Many administration tools are shared between these two systems which makes a data transition easier [28]. 'Organizations using proprietary standards, however, might face

significant costs during data migration. Hence, the installed base will largely determine whether open source software can easily be deployed within the organizations' [28]. According to Morgan and Finnegan [17], the technical benefits of open source software, for example quality and flexibility, overcomes the disadvantages making open source software interesting for business solutions. A case is mentioned where a organization decided on adopting open source software for their product because of demands from the customers. The customers simply wanted to have Linux running on their machine because of the benefits open source software gives [17]. The above-mentioned case is a good example of how open source software can affect the market and fulfil customer needs. Another benefit of adopting open source software is, as mentioned before, reduced costs. Nagy et al. mentions a case where an organization managed to lower their expenses [19]. This case is about replacing a '$100 million mainframe system with a $2.5 million system running on 144 Linux servers'.

It is important to understand that there are barriers in adoption of open source software, Nagy et al. present five of these and propose remedies for each one [19]. For example, they list 'Legacy integration' as a barrier, which means that there might be a problem connecting software to an old system [19]. If this is an issue the authors proposes usage of middleware solutions. Another barrier is about 'Sunk costs' with the description that proprietary software is invested in as a prior solution as a case [19]. A proposed remedy for the 'Sunk cost' barrier is to consider an open source software in places where proprietary software is not used [19]. Also, it is proposed to compare future costs between proprietary software and open source software in order to conclude the worthiness of a migration [19].

4.5 Open Source Adoption Model for Hospitals

Munoz-Cornejo et al. [18] conducted a survey of 30 hospitals, along with 5 interviews from the same population. From the results of these interviews as well as the survey results, Munoz-Cornejo et al. proposed the 'Empirical model for open source software in hospitals'. The model is depicted in Fig 1, and it is built on the paradigms from Staruss and Corbin [5]. Munoz-Cornejo et al. hoped this model would help hospitals in their decision to adopt open source software in their organizations or not. The model describes a set of factors in different categories to consider when conducting a migration from proprietary systems to an open source system in a hospital setting. The categories are, 'Casual conditions', 'Core categories', 'Strategic actions', 'Intervening conditions', 'Contextual factors', and 'Consequences'.

Casual conditions are according to Munoz-Cornejo et al. factors that influence the core categories, thus influencing whether or not a hospital is open to a technology solutions which includes open source software. Some of these conditions are, lack of in-house software developer, lack of IT personnel, the view that open source software is lacking in quality.

Core category, are set from a mix of casual conditions, core category contains the main stakeholder, thus this is where the main choice whether or not to adopt

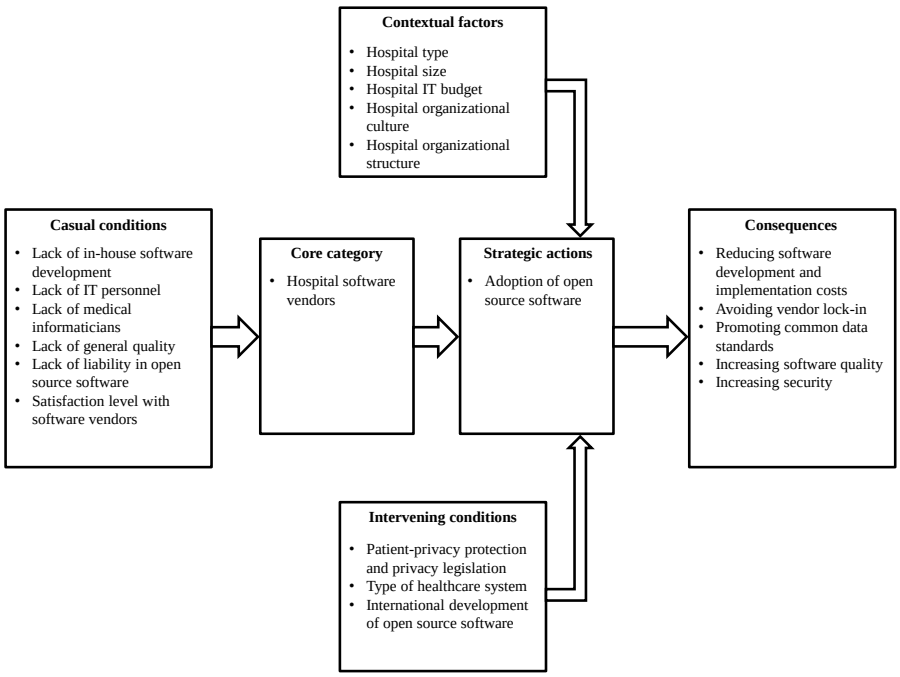

Fig. 1. Empirical model for adoption of open source software in hospitals

open source. In the hospital setting, since they do not have any in-house software development, this decision is often left to the software vendors of the hospitals.

Contextual factors, are more general and static than casual conditions. They do not form any specific attitude towards whether to adopt or not adopt open source software in a hospital setting. Hospital type has big role here, for example, university hospitals are usually more open to new unproven software, while private hospitals usually are less prone to the same unproven software. The culture and power structure of the organization also plays a large role here, an organization which has a manager that is open to open source software is more prone to adopting new open source software than a organization that has a manager that is not open to open source.

Intervening factors, are external factors that needs to be mitigated, otherwise they will impact the adoption of open source in a negative way. In a hospital there are several of these factors. The most important factor that needs to be mitigated, is the patient-privacy protection and privacy legislation. There was a general view in this study that open source software was a threat to patient-privacy protection and privacy legislation.

Strategic actions, this is the interaction outcome of core categories, contextual factors, and intervening conditions. Depending on the outcome of these, a

decision must be made whether to adopt or not adopt open source software in the hospital setting.

Consequences, this is the outcome of the decision to adopt open source, the model presented in the paper by Munoz-Cornejo et al. is closely aligned with the benefits of open source that the most literature about adopting open source claims.

5 Method

As data collection method for this case-study [23] we used interviews.

We conducted the interviews in a semi-structured way with the pyramid strategy [23] during the interviews themselves. This strategy starts with specific questions, then drifts towards more open ended questions [23]. The number of interviews conducted were five and they were conducted on four different companies within the embedded industry. These companies were decided in collaboration with HiQ.

The interviews themselves took roughly one hour to conduct. We decided that if more time is spent there is a risk of the interviewee being bored with the questions and not answering to the best of their knowledge.

The choice of semi-structured interviews was for us the best way to conduct this study, since new questions might have been arising during the initial interviews, questions we have not thought to add in our line of reasoning for embedded systems. Also semi-structured interviews is a good way to get a holistic research view of the companies we interview, this is needed to give the research more validity [23].

The interviews were conducted by both researchers of this case-study [23] and were recorded when permitted. Also, notes were taken during the interviews. Interviews started out with a privacy consent agreement between the interviewee and the researchers. After consent was given the interview was conducted, asking some simple background questions:

1. How long have you been working within the company?
2. How long have you been working within the embedded industry?
3. How much knowledge do you have about embedded architecture?

When these start-up questions had been asked we continued the interview according to the pyramid strategy [23] using these interview questions:

- What kind of system are you using in your products today?
- What are your companies main challenges when using RTOS/Embedded Linux?
- How do you see on open source solutions instead of RTOS?
  - Does your company have any specific position on open source solutions in general?
- Functional completeness:
  - Which functions do you require in your system?

    - Are there any specific tasks that need to be completed in your system?
- Time behaviour:
    - Does your system have any specific time constraint needs?
- Fault tolerance:
    - How do you handle hardware/software faults in your applications?
- Recoverability:
    - Does your system have any way to recover from faults/interrupts?
- Modifiability:
    - How do your company handle modifications in your systems?
- Does your company have the ability to run your target software in a host environment?

The quality attributes in these questions were decided upon by our understanding of what is important for RTOS/embedded Linux, and the questions were used to answer all the research questions.

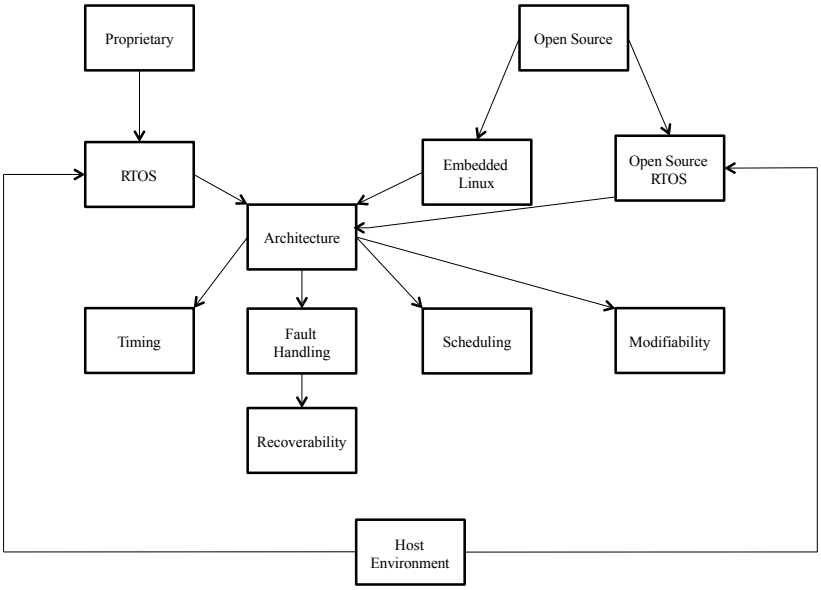

Fig. 2. Codecs

Next step in the process was to transcribe the interviews and send the transcribed records to the interviewee for approval. After the interviews were transcribed, analysis of the collected data begun.

Analysis of the data was conducted using an editing approach [23]. This allowed us to ensure the quality of the data assembled during the interviews. After the data was transcribed a preliminary set of codes was found. These codecs were deliberately derived from the interview questions. Depicted in Fig 2 are the codecs that were determined by the researchers, these codes were (Proprietary, Open Source, RTOS, Embedded Linux, Open Source RTOS, Host Environment, Architecture, Timing, Scheduling, Modifiability, Fault Handling, Recoverability). The model in Fig 2 is to be read as a tree structure were the parent nodes are proprietary and open source. Both parent nodes have two shared children, architecture and host environment, and architecture itself has several children. The last codec in the list is 'Host Environment', this is used if any of the systems in question has the ability to run in some sort of hosted environment. For embedded Linux this is Linux. For RTOS these could be a simulator environment used to test the code without having to run it on the target hardware.

By using these codes we conducted an edited analysis of the data and sorted it such as it corresponds with the codecs derived from the transcription.

Next step in the analysis was to interpret the findings based on what was discovered during the editing phase of the analysis. To validate our results and findings we were using four different companies from different sectors within the embedded industry, thus making our findings more general.

6 Results

This section describes the results of our research. It is divided after the research questions, and at the end our 'Embedded Linux Adoption Model' is described.

6.1 Research Question 1

In answer to research question 1: *'What are the perceived quality gains of migrating from RTOS to embedded Linux?'*.

We found that the quality gains in a migration from RTOS to embedded Linux are, embedded Linux is free, thus there is no licensing cost, this is the main advantage with embedded Linux according to all our interview subjects. Also, one thing that arose during the interviews were, when using embedded Linux it is easier to develop on a host machine running Linux, thus it is simpler to change the hardware drivers if it is needed. Or as one of our interviewees said 'When using embedded Linux, it is easier to run the target software on a host with some kind of interface or acceleration. It is less expensive to port a system if it is built on embedded Linux, the development can easy be made on a host and if a customization has to be made the drivers can be changed'.

One other thing that we found was, if a custom solution of embedded Linux should be implemented, this could be done quite easy with the help of Yocto [22], by just choosing the packages that is suitable for the situation the company is trying to solve.

6.2 Research Question 2

In answer to research question 2: *'In what cases are RTOS needed?'*.

We found that in some cases RTOS is required, for example in time critical software solutions where the software needs to be interrupted down to milliseconds, this is something that Linux has gotten better at but at its current state it cannot compete with RTOS [10]. If the system in question is a large system with many different modules that are correlating with each other the best solution to this problem is to run a hybrid RTOS/Linux system, these are available as both proprietary and open source solutions. Sometimes the best solutions is to create an own RTOS solution, this can be cheaper but with the trade-off that it can be more difficult to maintain, since you have to have in-house specialists who main the system.

Three out of five interviewees, at company 1, 2, and 3 thought that real-time attributes would be lost if a migration to embedded Linux was done. While two out of five interviewees at company 2 and 4 did not find this as an issue. It may also be simpler to maintain due to the fact that everyone can learn it without having to pay a licence fee to use the product, people sitting at home can learn it faster with the help of the vast Linux community. This makes it easier to find expertise which already know how to program in Linux and thus the company does not have to educate them in their proprietary RTOS.

In many cases the needs for RTOS is just perceived, since the companies using RTOS does not actually need the timing abilities of RTOS. Furthermore, we have not found any prejudice towards open source solutions itself, although, there seems to be a prejudices towards the timing requirements overall in embedded systems, since two out of five of our interviewees at company 1 and 3 said that they needed microsecond timing requirements. This is something that needs to be researched further since our research did not include the timing constraints of embedded systems.

6.3 Trade-offs

When it comes to trade-offs between RTOS and embedded Linux we found a few points to consider while conducting a migration. When switching from RTOS to embedded Linux you may loose some timing accuracy of your product as well as real-time scheduling and kernel pre-emption, another loss could be support, since this is something you pay for in most RTOS.

The gains from a migration, in addition to the cost perspective, are maintainability and higher flexibility. As one of our interviewees at company 1 said about flexibility, 'The problems occurs when we want to extend our product, the big distributors of closed solutions are very slow'.

6.4 Embedded Linux Adoption Model

To conclude our findings about migrating from RTOS to embedded Linux we created a model based on the paradigms by Corbin and Strauss [5]. This model

was adopted for a hospital setting by Munoz-Cornejo et al. [18]. These empirical models helped us develop the 'Embedded Linux Adoption Model' in Fig 3.

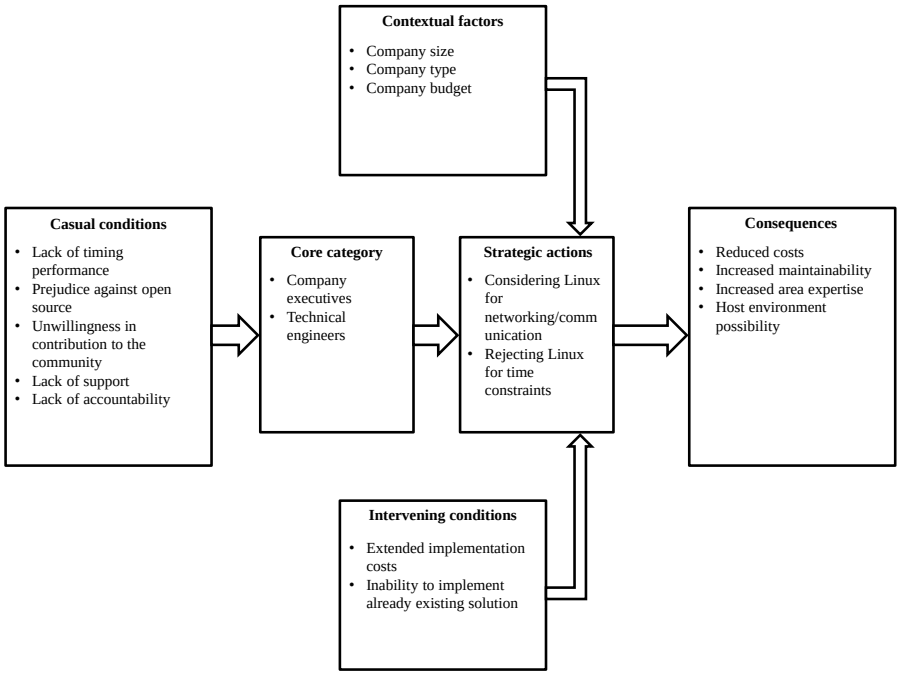

Fig. 3. Embedded Linux Adoption Model

Casual Conditions are 'factors that are identified as influencing the core category' [18]. Casual conditions according to Munoz-Cornejo et al. have an influence on whether companies are open minded to open source software, in our case Linux. The casual conditions we found in our research are: 1) Perceived lack of timing performance, this correlates to scheduling algorithms and time from interrupts to actual execution of code. 2) Prejudices towards open source, such as timing constraints, etc. 3) Unwillingness to contribute to the community; some companies does not want to contribute with their results and source code to the community because of competition issues. If this is the case a solutions has to be found in order to protect the code from being exposed, for example by integrating the code into external scripts. 4) Lack of support, many community editions of Linux lack official support, there is however many different forums where help can be supplied. As one of our interviewees said 'We do not have anything against Open-source software and we could definitely share changes in the code. On the other side, a problem could be support; we would need some kind of agreement for paid maintenance of the Open-source code.' 5) Lack of

accountability, who is responsible if something goes wrong with the open source software? This is something that needs to be taken into consideration when thinking of a possible migration, as this cannot be mitigated.

Core Category, these are the main stakeholders of the company adopting embedded Linux [5]. These have not been addressed in this study, but a more general approach towards open source software in general has been taken. The main stakeholders according to Hammouda [24] are: 1) Company Executives, and 2) Technical Engineers.

Strategic Actions are 'purposeful or deliberated acts that are taken to resolve a specific problem' [5]. This interaction is the outcome of core categories, contextual factors and intervening conditions combined [18]. For our model the strategic actions are: 1) Consider Linux for network/communication, these are parts of the system where timing constraints does not need to be taken into account, at least not down to the millisecond level. 2) Reject Linux for time constraints, where hard real-time is actually needed, Linux is a poor choice, here it is better to either keep running the RTOS as it is, or migrate to an RTOS running a Linux as a subsystem.

Contextual Factors are 'specific set of conditions (patterns of conditions) that intersect dimensionally at this time and place to create a set of circumstances or problems to which persons respond through actions/interactions' [5]. These factors are general in comparison to 'casual conditions' listed above. We have not identified any new contextual factor from the research done by Munoz-Cornejo et al., only adapted them to suit a more general view of our model. The contextual factors are: 1) Company size, large companies are more and more willing to both contribute and use open source. 2) Company type, companies that deal with hard real-time constraints are not as willing to adopt embedded Linux due to the perceived timing loss. 3) Company budget, a company with a large budget is more willing to create their own RTOS solution for real-time constraints and use embedded Linux for everything else. As one of our interviewees said, 'We cannot just pick something from the shelf and expect it to work as we want, we need to meet the microsecond requirements.'

Intervening Conditions are the conditions 'mitigate or otherwise impact causal conditions' [5]. The intervening conditions we have identified as important for embedded systems are: 1) Extended implementation cost, if functionality is missing in Linux that is needed for the company, there may be a need to develop this functionality in Linux in order to make the migration possible. 2) Inability to implement already existing solutions, some companies have already working solutions. These can be hard to port to an embedded Linux solution.

Consequences are 'the outcomes of the interaction of the core category with the contextual factors, intervening conditions and the strategic actions' [18]. The benefits of this model is aligned with the actual benefits of a more general propriety to open source migration, however this is something we can only speculate in since this is not part of this research. 1) Reduced cost, the licensing cost is removed due to it being an open source software. 2) Increased maintainability, with a vast community and the ability to correct errors without having to wait for

vendors to do it. 3) Increased area expertise, everyone that have the interest can download and educate themselves in open source software. 4) Host environment possibility, the target software can be run and tested in a host environment.

7 Discussion

Since the amount of studies made about this topic is quite low, whilst the usage of RTOS in embedded products is constantly growing, we found this subject important to study. The research questions were aimed to answer what kind of possibilities or barriers there exists when conducting a migration from RTOS to embedded Linux. Also, in what cases a migration from RTOS to embedded Linux is not worth considering. We managed to answer our research questions and conclude that a migration in many cases is possible. As with all other kind of migrations there are trade-offs, we concluded that one of the biggest trade-off in a migration from RTOS to embedded Linux, has to do with loss of timing accuracy and other real-time characteristics such as scheduling and kernel pre-emption as well as loss of support in favour for free software, with higher maintainability and flexibility. Although, there are many embedded products that do not have any hard real-time constraints, such as refrigerators, door locks, and washing machines. Where, in many cases embedded Linux could serve very well as a substitute to RTOS.

One important fact is that there are cases RTOS is needed and cannot be replaced with embedded Linux. These cases are when the hard real-time aspects needs to be taken into consideration. During our interviews we got the impression that there might be prejudice towards the timing abilities that are needed in embedded systems, this is something that could be investigated further. We think this study can contribute to all companies that think about migrating from RTOS to embedded Linux. Although no actual implementation was done we managed to provide an adoption model that we think might be helpful in the decision-making.

7.1 Validity Threats

Since the main source of information for this case study was interviews there is a risk that the interviewed participants were biased. We identified possible threats before every interview in order to avoid confirmation bias and inconsistent questioning. Another threat to validity is about perceived views of the interviewed participants. Also, the size of the interviewed companies might affect the result since only large and medium sized companies were interviewed, see Section 3.2. The fact that we only did interviews at large and medium size companies may give some bias to the adoption model when it comes to small size companies. The last identified threat to validity in this paper is the amount of interviews, since we only had the chance to do five interviews in total this could mean that our result may not be generalized.

8 Conclusion

We concluded that Linux can substitute a RTOS if there are no needs for hard time constraints. It cannot outperform an RTOS when it comes to the core properties of real-time but can still be a very good substitute in many cases with increased functionality as a bonus. It is shown that the needs of an RTOS in many cases are perceived.

Our hopes are that the model we presented will be to a help while considering a migration from RTOS to embedded Linux.

The result shows that open source solutions definitely is something that is worth to be taken into consideration and that many prejudices are based on old thinking, and in many cases are incorrect. We also concluded that the opinions about open source solutions are very positive.

8.1 Future Work

Our main recommendation for future work is to extend this study with an actual migration from an RTOS to embedded Linux following the Embedded Linux Adoption Model we presented. One, or more Linux builds could advantageously be made with the Yocto Project [22] to suit the system and cover all the needs. Different tests could be performed on real-time performance, maintainability, usability, visibility etc. with comparison to a proprietary RTOS; this could preferably be done directly on a target. Also, the study could be extended with more focus on real-time properties in an experiment setting, answering the following question. 'In what cases does RTOS need microsecond timing?'

The biggest limitation of this study is that no practical testing has been performed; if we had the possibility to extend the study we would definitely make an actual migration with experimental testing.

Acknowledgments

Authors would like to thank HiQ for all their help and collaboration in this paper. Especially Martin Bergek for all the helpful input as our sounding board. We would also like to thank our Professor Imed Hammouda. And last but not least, we would like to thank all the people within the industry that took the time to let us interview them.

References

1. Open source initiative (May 2014), http://opensource.org/
2. Barabanov, M.: A linux-based real-time operating system. Ph.D. thesis, New Mexico Institute of Mining and Technology (1997)
3. Bonaccorsi, A., Rossi, C.: Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. Knowledge, Technology & Policy 18(4), 40–64 (2006)

4. BSI, B.: Iso/iec 25010: 2011 systems and software engineeringsystems and software quality requirements and evaluation (square)system and software quality models (2011)
5. Corbin, J., Strauss, A.: Basics of qualitative research: Techniques and procedures for developing grounded theory. Sage (2008)
6. Debian: Debian system requirements (June 2014), https://www.debian.org/releases/stable/i386/ch03s04.html.en
7. Dinkelacker, J., Garg, P.: Corporate source: Applying open source concepts to a corporate environment (position paper). 1st WS Open Source SE (2001)
8. Fox, R., Kasten, E., Orji, K., Bolen, C., Maurice, C., Venema, J.: Real-time results without real-time systems. Nuclear Science, IEEE Transactions on 51(3), 571–575 (2004)
9. GNU: Gnu general public licence (May 2014), https://www.gnu.org/copyleft/gpl.html
10. Gupta, R.: Linux 2.6 for embedded systems-closing in on real-time. RTC Magazine 12(11) (2003)
11. Hauge, Ø., Ayala, C., Conradi, R.: Adoption of open source software in software-intensive organizations–a systematic literature review. Information and Software Technology 52(11), 1133–1154 (2010)
12. HiQ: Hiq website (May 2014), www.hiq.se/en
13. Kvist, H., Larsson, M.: Migration from a real-time operating system to embedded linux (2005)
14. Li, Y., Potkonjak, M., Wolf, W.: Real-time operating systems for embedded computing. In: Computer Design: VLSI in Computers and Processors, 1997. ICCD'97. Proceedings., 1997 IEEE International Conference on. pp. 388–392. IEEE (1997)
15. Marchesin, A.: Using linux for real-time applications. Software, IEEE 21(5), 18–20 (2004)
16. Morgan, L., Finnegan, P.: Benefits and drawbacks of open source software: an exploratory study of secondary software firms. In: Open Source Development, Adoption and Innovation, pp. 307–312. Springer (2007)
17. Morgan, L., Finnegan, P.: Open innovation in secondary software firms: an exploration of managers' perceptions of open source software. ACM SIGMIS Database 41(1), 76–95 (2010)
18. Munoz-Cornejo, G., Seaman, C.B., Koru, A.G.: An empirical investigation into the adoption of open source software in hospitals. ProQuest (2007)
19. Nagy, D., Yassin, A.M., Bhattacherjee, A.: Organizational adoption of open source software: barriers and remedies. Communications of the ACM 53(3), 148–151 (2010)
20. Opdenacker, M.: Introduction to uclinux (June 2014), free-electrons.com/doc/uclinux_introduction.odp
21. of Oxford, U.: Oss watch (May 2014), http://oss-watch.ac.uk/
22. Project, Y.: Yocto project (May 2014), https://www.yoctoproject.org/
23. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 14(2), 131–164 (2009)
24. Sowe, S.K., Parayil, G., Sunami, A.: Free and Open Source Software and Technology for Sustainable Development. United Nations University Press (2012)
25. TinyCore: Tiny core system requirements (June 2014), http://distro.ibiblio.org/tinycorelinux/
26. Top500.org: Top 500 super computers (May 2014), http://www.top500.org/statistics/details/osfam/1#.U3ze_Pl5OGE

27. Torvalds, L.: Release notes for linux v0.12 (May 2014), http://web.archive.org/web/20110721105526/http://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.12
28. Ven, K., Verelst, J., Mannaert, H.: Should you adopt open source software? Software, IEEE 25(3), 54–59 (2008)
29. Weinberg, B., CESATI, M.: Moving from a proprietary rtos to embedded linux. Montavista Software (2001)
30. Weinberg, B., Pundit, L.: Migrating legacy vxworks applications to linux
31. Weinberg, W.: Moving legacy applications to linux: Rtos migration revisited (2008)