

Embedded Linux Total Cost of Development Analyzed

**An Evaluation of the Costs Associated with Embedded
Linux Development as Compared with Commercial RTOSes
and Non-Commercial Linux**

Jerry Krasner, Ph.D., MBA

December 2007

**Embedded Market Forecasters
American Technology International, Inc.**

Embedded Market Forecasters
Research and Consulting
for Embedded Products,
Markets and Channels



About EMF: www.embeddedforecast.com 508-881-1850

EMF is the premier market intelligence and advisory firm in the embedded technology industry. Embedded technology refers to the ubiquitous class of products which use some type of processor as a controller. These products include guided missiles, radars, and avionics as well as robots, automobiles, telecom gear, and medical electronics.

Embedded Market Forecasters (EMF) is the market research division of American Technology International, Inc. EMF clients range from startups to Global 100 companies worldwide. Founded by Dr. Jerry Krasner, a recognized authority on electronics markets, product development and channel distribution, EMF is headquartered in Framingham, Mass.

About the author:

Jerry Krasner, Ph.D., MBA is Vice President of Embedded Market Forecasters and its parent company, American Technology International. A recognized authority with over 30 years of embedded industry experience, Dr. Krasner was formerly Chairman of Biomedical Engineering at Boston University, and Chairman of Electrical and Computer Engineering at Wentworth Institute of Technology and Bunker Hill Community College. In addition to his academic appointments, Dr. Krasner served as President of Biocybernetics, Inc. and CLINCO, Inc., Executive Vice President of Plasmedics, Inc. and Clinical Development Corporation, and Director of Medical Sciences for the Carnegie-Mellon Institute of Research. Earlier, he was Senior Engineer at the MIT Instrumentation Laboratory. Dr. Krasner earned BSEE and MSEE degrees from Washington University, a Ph.D. in Medical Physiology / Biophysics from Boston University and an MBA from Nichols College. He is a visiting professor at the Universidad de Las Palmas (Spain) where he is recognized for his work in neurosciences and computer technology.

Copyright 2007 by Embedded Market Forecasters, a division of American Technology International, Inc, 1257 Worcester Road #500, Framingham, MA 01701. All rights reserved. No part of this document covered by copyright hereon may be reproduced or copied without expressed permission. Every effort has been made to provide accurate data. To the best of the editor's knowledge, data is reliable and complete, but no warranty is made for this.

Introduction

In 2003 EMF produced a seminal report on the comparative development costs associated with Linux developers as compared with Microsoft Windows Embedded XP and with Windows CE. (“Total Cost of Development, a Comprehensive Cost Evaluation Framework for Evaluating Embedded Development Platforms” – available for free download at www.embeddedforecast.com.)

This report is still the most frequently downloaded and quoted EMF research report.

The purpose of this report is to re-evaluate the 2003 findings in light of the many changes in the embedded marketplace and to set the record straight for 2007.

Four years after the publication of the original study, Linux has become a trusted alternative to commercial RTOSes for many applications. In this paper we use current detailed survey data to compare Linux with established commercial RTOSes for embedded design activities and design outcomes. A discussion of the original survey methodology and data collection is presented in Appendix A.

It is worthwhile noting that, since 2003, notwithstanding the documented problems encountered with Linux development, Linux as an embedded operating system (OS) has gained dramatic market share. Much of that growth can be ascribed to both the demise of the Embedded Linux Consortium and the efforts from Linux vendors to provide an impressive array of development tools to address the needs of the 40 percent of Linux developers who chose at that time to use in-house tools. Following the clear evidence that embedded Linux was here to stay, commercial RTOS vendors such as Green Hills Software, Wind River and Mentor Graphics expanded their IDEs to encompass Linux development.

In this report, EMF presents comparative data derived from the past two years of extensive research about embedded developers to look at design processes and design outcomes comparing developers using established embedded RTOSes versus embedded Linux.

The 2006 and 2007 EMF data show that since 2003, Linux has achieved design parity with commercial RTOSes for many applications.

Design Undertakings and Design Outcomes: Embedded Linux versus Commercial RTOSes

Analysis Parameters

EMF uses a selection of data parameters that best reflect design undertakings and design outcomes. These include:

- Time from design start to product shipment (design time in months)
- Total lines of new and modified source lines of written code--excluding reuse, commercial off-the-shelf and open source software
- Total number of source lines of code written--including reuse, commercial off-the-shelf and open source software
- Percentage of designs that are delivered behind schedule
- Number of months behind schedule
- Once the integration cycle starts, developers are asked to grade the accuracy of final design hardware and software versus pre-design expectations. Expectations in performance, systems functionality, and features and schedule are examined. EMF records the percentage of responses for which the final design achieved within 20% of pre-design expectations and within 30% of pre-design expectations.

EMF also records the percentages of designs that are cancelled, the number of months from design start to cancellation and the percentage of designs that are outsourced. However, the data parameters listed above provide for an accurate comparative assessment between designs using Linux as compared with the leading commercial RTOSes.

Vendors Compared: Linux and commercial OSes

Linux

FSM Labs (now Wind River)
LynuxWorks BlueCat Linux
CodeWarrior for Linux
MontaVista Linux
Red Hat Linux (Enterprise)
Wind River Linux

Commercial RTOSes

Green Hills Integrity
LynuxWorks LynxOS
Mentor Graphics Nucleus
Microsoft Windows XPE
Microsoft Windows CE
Wind River VxWorks

Design comparisons

Table I presents comparative design undertakings between commercial RTOSes and Linux. The data representing written and total lines of code indicate the complexity of the undertaking. The average number of software developers per project as well as the percentage of designs behind schedule and the number of months behind schedule reflect associated costs of design.

The intent here is to determine whether Linux-based designs have improved since 2003, and whether there are substantive differences (aside from security and mission critical requirements that will be discussed later). This data is taken from EMF's 2006 and 2007 surveys of more than 1300 embedded developers.

	RTOSes	Linux	Industry Avg.
Avg # SW developers per project	40.5	47.5	30.9
Avg lines of code written x1000	198.5	252.6	186.3
Avg total lines of code x1000	704.4	1000	749.3
Total design time (mo)	15.5	14.4	15.5
% behind schedule	43.0%	35.8%	41.4%
Avg months off-target for behind-schedule projects	3.8	3.9	4

Table I: Project scope and schedule adherence: RTOSes vs. Linux

Importance of time to market

The data presented here represent an aggregate of different vendors' OSEs as deployed by developers of different vertical market applications. Although beyond the intended scope of this paper, EMF data can be used to compare the cost of development for developers using any OS and designing for any market segment. The following is an example of the calculations that can be made.

Time to market significantly affects the overall cost of a project. Extensive EMF surveys show that the average number of software developers per project is 30.9. If the average cost per engineer (salary, benefits and supporting technical staffing) is only \$150,000 per year, then the average cost for each month of delay is \$386,250.

Knowing that on average 41.4% of embedded designs are behind schedule by an average of 4.0 months, one can calculate the average expected loss due to design delays per project as $(\$386,250) \cdot 0.414 \cdot 4 = \$639,630$.

Notably, these calculations don't take into account the cost of lost revenues because the product was delivered late. Time to market is important in every industry segment, but in short product lifecycle segments such as consumer products, it often makes the difference between profit and loss.

Clearly, a few tens of thousands of dollars difference in up-front costs to purchase a superior time-to-market OS is very small compared to the expenditures experienced as a consequence of late design completion, and the benefits of getting to market ahead of, rather than behind, competitors. Given the implications, it makes sense that such analysis should become part of every company's best practices.

Clearly, a few tens of thousands of dollars difference in up-front costs to purchase a superior time-to-market OS is very small compared to the expenditures experienced as a consequence of late design completion, and the benefits of getting to market ahead of, rather than behind, competitors. Given the implications, it makes sense that such analysis should become part of every company's best practices.

By comparison, the 2006 data showed very similar findings to 2007 with the number of software developers per project for Linux projects being seven more (on average) than for commercial RTOSes, while average design time for Linux projects was 1.1 months less.

It should be noted that this data for both the operating systems from Linux vendors and the systems from commercial RTOS vendors was equal to or better than industry averages. The industry average was determined from an aggregate of 46 different OSES, whereas the comparison between embedded Linux and commercial RTOSes was derived from the previously listed OS vendors.

In examining this data one has to be cognizant that it is broadly gathered across 10 different vertical markets as well as across the operating systems of a number of different vendors. Hence we can look to whether significant differences in design outcomes exist between designs based on commercial RTOSes and Linux.

In examining data derived from over 1300 respondents during a two-year period and from the information presented in Table I, it is clear that the design outcomes are virtually the same for Linux and non-Linux OSES.

An important aspect of the analysis can be found in the data regarding the design outcomes in which developers were asked to report how close their pre-design expectations met their final design results. If significant differences exist between Linux and commercial RTOS design outcomes, they will be seen in Table II.

Table II presents the results of the inquiry by reporting the percentage of design results that were within 20% of pre-design expectations and within 30% of pre-design expectations. Designs not falling within these ranges would be considered problematic. EMF has not looked at whether those designs not within 30% of pre-design expectations were associated with design delays.

	RTOSes	Linux	Industry Avg.
<20% of expectations			
Performance	66.8%	70.0%	64.0%
Sys functionality	68.4%	71.3%	65.5%
Features & schedule	55.6%	59.5%	54.3%
<30% of expectations			
Performance	74.6%	75.6%	71.8%
Sys functionality	75.4%	76.2%	72.1%
Features & schedule	68.1%	67.9%	64.7%

Table II: Meeting project expectations: RTOSes vs. Linux

Table II is instructive as it demonstrates that embedded Linux design outcomes are consistent with design outcomes of projects using OSes from the established commercial RTOS vendors, and that both are ahead of the industry average.

Analysis

It is fair to ask why results for embedded Linux-based designs have improved since 2003. The answer is tools. In 2003 tools were limited and 40% of Linux designs involved in-house tools sets or limited tools that were available for free download.

MontaVista Software, for example, developed a comprehensive set of design tools and have partnered with Model-Driven-Design (MDD) vendors to enhance deployment of embedded designs. LynuxWorks enabled its BlueCat Linux to interoperate with its LynxOS for military applications, whereas Wind River deployed its own version of Linux with its set of development tools (and acquired the assets of FSM Labs).

Linux also gained acceptance within the embedded community based on its use in industrial automation applications, telecom/datacom and in server applications across the board. Furthermore Linux is broadly used for applications in TVs, set top boxes, DVRs such as TiVo, and especially in mobile phones like the Motorola RAZR2 V8.

The emergence of such multi-target OS IDEs as Green Hills' MULTI, Wind River Workbench and Mentor Graphic's EDGE allowed Linux development to be integrated with another OS or pursued as a standalone design using the same IDE.

Linux also benefited from the mission critical capabilities of POSIX conformant and secure (EAL 5+ or greater) OSES such as Integrity and LynxOS in which a non-secure and non-conformant Linux application could be safely run within that certified OS in a secure memory-protected environment. This removes any perceived barrier to entry for such stringent application requirements.

In summary, Linux not only continues to gain market share across the embedded spectrum, but the definitive EMF data set forth herein demonstrates that developers using Linux have the same design outcomes compared with traditional RTOSes. In addition, there is every expectation that Linux vendors will continue to offer more advanced tools as part of their product offerings, which should become a consideration in a developer's OS selection.

The emergence of this ability has greatly benefited the embedded marketplace at large. A stable and application proven Linux design could be directly incorporated within a mission critical application that required MILS or EAL certification.

In summary, Linux not only continues to gain market share across the embedded spectrum, but the definitive EMF data set forth herein demonstrates that developers using Linux have the same design outcomes compared with traditional RTOSes. In addition, there is every expectation that Linux vendors will continue to offer more advanced tools as part of their product offerings, which should become a consideration in a developer's OS selection.

Tools, Tools and More Tools

Clearly, Linux has thrived due to the availability and use of appropriate tool sets. MontaVista and Wind River have an impressive arsenal of general and application specific tools to enable designs that are on-time and close to pre-design expectations – however they have taken different approaches to the marketplace. But not all Linux developers make use of such tool sets; some chose to develop their own. This is not what one would consider an effective use of time and resources – given the availability not only of Linux-specific tool sets but the additional availability of IDEs for embedded development from a number of different vendors. However, a similar mentality exists within the RTOS industry (16.1% of embedded designs use an in-house RTOS and tool sets – down from 30% in 2005).

Recent EMF data clearly shows that time-to-market and final design results play a huge role in the financial success of a design effort. Being late to market and/or having to remove design features can reduce the market potential of a design by 50% or more. (Consumer electronic designs that miss the Christmas window of opportunity, for example, face an even gloomier future.)

When EMF looked at comparative data between established Linux vendors and the in-house (roll your own – EMF calls these “in-house Linux”), our analysis

showed that the in-house developers wrote fewer lines of code and fared poorly when it came to final design outcomes as compared with their pre-design expectations.

Figure I provides an insight to why this might be expected.

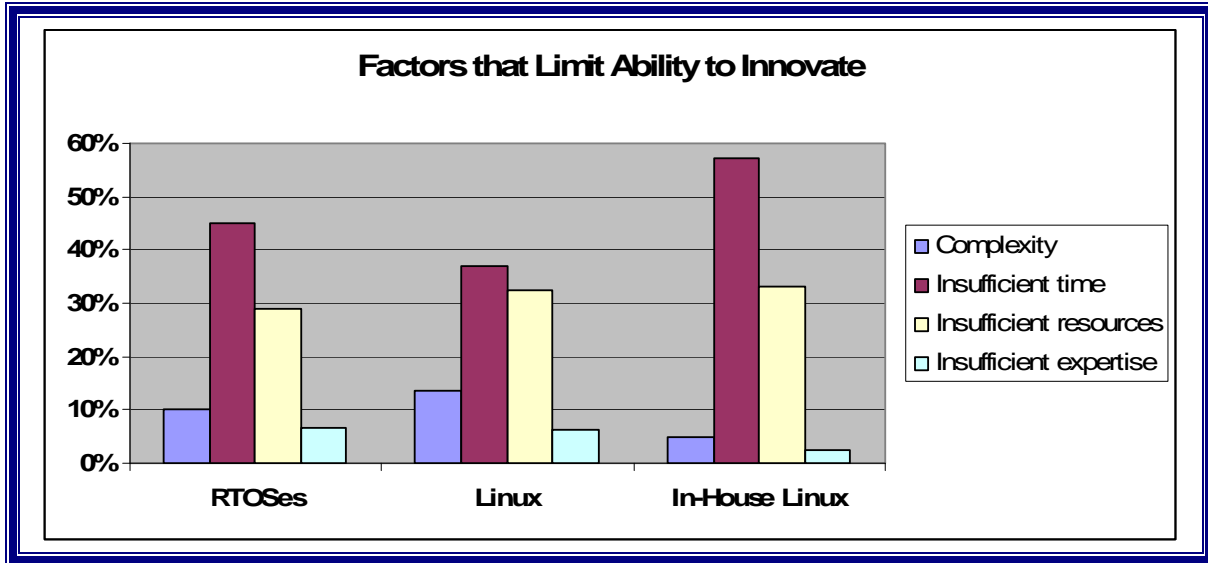


Figure I: Perceived limitations: RTOSes vs. Linux

What we see here is that designs undertaken by in-house Linux developers tend to be well within their levels of expertise and that their resources are similar to developers using commercial Linux products. EMF data shows that these undertakings involve many fewer lines of code written and are thereby less complex. One might assume that developers using in-house Linux would not be stressed by on-time completions. Figure 1 shows that more than 50% of these developers claim to lack sufficient time to develop their concepts and bring them to market.

Further consider the following data that refers to the relationship between pre-design expectations and final design results.

Final design results within 30% of pre-design expectation for features and schedule:

Commercial Linux = 70.2%
 Other Linux = 59.0%

From EMF survey data, in-house Linux developments involve less complexity (and also result in fewer shipments) and fall short of meeting pre-design expectations when compared to commercial Linux or RTOSes. Furthermore (as

seen in Figure 1), in-house Linux developers report a major problem with having sufficient time to complete their designs.

EMF believes that these factors are consistent and point to the risk of using valuable resources and time to maintain an in-house capability when more affordable and effective resources are available.

Commercial OSES versus Open Source Software

Embedded markets change and the market for realtime operating systems is no different. Developers today are confronted by greater design complexities and limited windows of opportunity. They don't have the time to research microprocessors and OSES individually, write firmware code, compile, debug, test and integrate hardware and software. It has become increasingly clear that "time-to-market" is a major factor in the successful outcome of a product release – and distractions or unnecessary design undertakings that put product delivery at risk should be avoided. New vertical market application processors are emerging weekly, and the need for appropriate operating systems to address these products is placing increasing demands upon embedded developers. They are increasingly seeking integrated solutions, rather than components that require assembly and integration. Successful vendors are responding to this need.

Some vendors have partnered with Model Driven Design (MDD) vendors such as Telelogic, the provider of Rhapsody. Others, such as Mentor Graphics, have incorporated Mellor's UML modeling tool into their development environment. MontaVista introduced a "TestDrive" tool that permits a developer to start evaluating MontaVista Linux on the target processor and reference board of their choice so that developers can develop and run their own test applications to quickly determine the technical fit of MontaVista products for their projects before they make a purchasing decision.

The OS market segment has become interdependent with other segments of the embedded software tools market. In particular the availability of appropriate software development tools and integrated development environments (IDEs) has become a driving factor in the OS selection process.

IDEs, along with other supporting tools and components, have become a deciding consideration in determining which OS is best for a project. Over the years, it has become increasingly difficult to technically distinguish between different OS offerings. In order for vendors to get developers to prefer their OS over others, they have differentiated their products not by technical considerations, but rather by providing tools and support that reduce time-to-market and enhance the probability that a final design will closely approximate pre-design expectations.

If the open source organizations are to be successful, they must provide for these considerations. The open source mantra has been that royalty free software represents the most economical and efficacious way to develop embedded applications. As it pertains to embedded software, can software really be free, or is there an intrinsic hidden cost?

When pricing out development costs of an embedded device, software remains the most expensive and hard-to-forecast systems component. When open source dangles the lure of zero cost for an RTOS and associated device drivers, there is a real temptation to jump at the opportunity.

But is open source really free or does it create other backside costs? This question has advocates and detractors in heated debate that might resemble a crusade of sorts. Clearly, the best way to look at this issue is through the hard data, such as the data that the EMF surveys provide.

On the surface, the proprietary approach (no access to source code in addition to paying a license fee, runtime royalties and a maintenance contract), seems quite expensive compared to the opportunity of receiving free software. At first glance, the ability to get a huge amount of free source code and have the freedom to sift through it and pick out the sections of most use to a developer seems very desirable. The embedded developer, without having to pay a fee, gets access to code developed by some of the best minds in the industry.

The counter-argument is that there is no free lunch in the embedded software marketplace. It takes a lot of engineering horsepower to sift through the mountain of code to extract those pieces that best fit a project's requirements. Vendors that provide a royalty-based or subscription based operating system, bereft of the source code, also take on the responsibility for supporting and enhancing the OS in addition to fixing bugs. This pertains to both Linux and non-Linux OSes. When given the choice of having to assume the responsibility of managing an RTOS that is otherwise managed publicly (by a distributed base of programmers), many developers choose to have a proprietary OS vendor assume those responsibilities.

Once a developer has provisioned himself with the selected source code, he still has to get it all onto the host. It requires a considerable amount of work to turn code into a platform appropriate for embedded development. So there is a cost in engineering time and a potential opportunity cost regarding time-to-market. Andrew Morton, speaking at a Linux Foundation meeting, made it clear that free Linux downloaded from kernel.org isn't production quality. So companies who choose to use such code have to do the "productization" themselves, including bug fixes and testing, release and documentation, and porting to other architectures.

An embedded developer has three alternatives in selecting an OS:

1. a proprietary RTOS,
2. an open source free download Linux,

3. a commercial Linux OS that provides the benefits of open source while delivering a commercial software platform.

As such, one cannot consider all versions of Linux monotonically as a single item, nor can one expect a generic “open source” solution. To illustrate the point Table III compares the design outcomes of Linux: industry average, Wind River’s Linux product and Monta Vista’s Linux product.

	Industry Avg (Linux)	Wind River Linux	MontaVista Linux
Avg. # SW developers per project	47.5	29.8	13.7
Avg. lines of written code x1000	252.6	52	97
Avg. total lines of code x 1000	1000	136	908
Total design time (mo)	14.4	13.9	11.2
% behind schedule	35.8%	37.3%	36.4%
Avg. months off target for behind-schedule projects	3.9	7.7	4.1
Performance within 20% of expectation	70.0%	58.9%	80.7%
Sys functionality within 20% of expectation	71.3%	58.8%	84.6%

Table III: Comparative design outcomes for different Linux OSes

Also at issue is whether embedded developers really need (or even want) to see their operating system source code. When thousands of embedded developers have built successful applications on Microsoft Windows without ever seeing a single line of its source, is it essential (or even efficient) to know how the operating system works? Is source code of importance to embedded developers or do they only need to know how it supports the application?

What we hear from customers is that many systems and platform developers absolutely want or need the source code. However, application developers don’t – they usually get a configured system from the system developers and then have access only to its APIs. With Linux, source code is a given, and therefore customers can make use of it in the context of their design methodology (either using source code or not). For many companies, having source code also provides insurance – they can control their own destiny if they have to – and don’t have to rely on a vendor who may not help them in the way they need it.

Certainly, source code provides an opportunity to learn by example how to better develop complex applications. Aside from the educational opportunity afforded by the source code, open source permits the developer to work around bugs in code that they don’t own. It’s assumed that open source software permits embedded

developers to regain control of the design process, fix a problem and move on. Some developers claim that in the wee hours of the morning it's best to have access to the source code rather than wait for commercial vendor's technical support lines to open.

The assumption herein is that by having the source code one can isolate a given problem and fix it in a reliable manner. This also assumes that the developer can imagine what the source code designer had in mind when it was originally written. Certainly the need for detailed and comprehensive documentation is a given and should be requested when developers are evaluating their Linux decision. Furthermore, once the source code is modified the developer has diverged from the original source code, usually at his own risk.

Embedded software differs from host software in that the developer is adapting software to run on a particular device. With different device driver requirements and peripheral part considerations, open source vendors argue that the availability of source code is essential. Closed source vendors can argue that their support tools are much more efficient in addressing these needs and that their products have been proven over many years of use.

As the value propositions change for embedded designs (namely time to market and final design outcomes) the need for effective tools, documentation and support become paramount. Furthermore developers need to minimize the costs related directly to the operating system – developers should invest in the applications rather than in the OS. The data presented herein clearly illustrates the financial implications associated with in-house developments.

Successful embedded Linux developers have learned this.

Summary

The purpose of this report was to re-evaluate EMF's 2003 findings regarding embedded Linux developments in light of the many changes in the embedded marketplace and to update the record for 2007.

The basis of our analysis is the extensive survey data that EMF compiles year-over-year from embedded developers. The findings and analysis are based on factual data.

Our conclusions are:

- Designing with a Linux OS is no longer an expensive and risky undertaking
- Using a supported commercial Linux OS is more cost effective to an in-house Linux development undertaking

- Linux can be used in a mission-critical environment that requires MILS, EAL certification or POSIX conformance, when used in protected memory under a certified RTOS. Also, the use of Linux for embedded designs is no longer restricted to a few applications and that the design outcome data demonstrate that developers can consider using a Linux OS without concern that their end product will be inferior to end products produced for a proprietary RTOS.

Appendix A: 2003 survey methodology and findings

EMF's 2003 paper was the first study of its type that compared development costs across numerous vertical market applications in which virtually identical designs were undertaken by developers using Linux and Microsoft Windows. Great care was taken to insure that the comparative analysis was made according to similar undertakings.

The results were significant and resulted – as might have been expected – in a huge and emotional response on both sides of the issue.

Notwithstanding the heated debate that ensued, the results of the study made clear that comparable developments between embedded engineers that used Linux or Windows OSes demonstrated the following:

In 2003:

- Linux developments used 75% more developers per comparable project than Microsoft Windows developments.
- Total time to market was 1.95 times greater for Linux developments compared to similar Windows developments.
- At the time of the study, Linux developers were paid substantially more than Windows Embedded developers. Even with the assumption that Linux developers were paid the same as Windows Embedded developers, comparable Linux design costs were three times greater than those for Windows Embedded.

Moreover in 2003, 40% of responding embedded Linux developers used free Linux kernel downloads and in-house developed tools. These respondents refused to use tools provided by embedded Linux vendors and promoted by the Embedded Linux Consortium – which advertised that Linux provided faster productivity.

Many undocumented and unproven claims existed in the marketplace, which formed the reason for the EMF undertaking. The results of the EMF study were as shocking to EMF as they were to the embedded community. In 2003, the EMF analysis concluded that the substantial differences in design outcomes might be the result of the following:

- One is the maturity, integration and usability of the tool chain environment as well as documentation. Both Windows Embedded operating systems include integrated development environments and application development tools such as Visual Studio .NET as well as rich API and architecture documentation. There were no tool chains available on embedded Linux in 2003 that would have a comparable level of features or integration and documentation of comparable quality.

- Second would be the componentized form of the Windows Embedded operating systems. Both Windows Embedded operating systems were delivered as a set of pre-tested footprint-reduced components that an OEM individually selects to construct, and an operating system image. For Windows CE .NET, the same operating system components and functionality were available for the four processor architectures Windows CE .NET supported at that time. In 2003 embedded Linux was not delivered as a set of components but as a source-configurable operating system whose kernel was originally designed for PC hardware architecture desktop and server computers. As a result, Linux developers spent large amounts of time to reduce footprints, integrate new functionality and port or develop embedded-specific components and drivers to non-PC architecture processors.
- Third would be the development sequence and process itself. Windows Embedded enabled a development team to build the operating system, application software, and hardware largely in parallel, due to the integrated availability of emulation, Software Development Kit export, and application development tools. While many good individual tools existed in 2003, embedded Linux vendors lacked these tools chains and a platform level SDK in an integrated form, making serial development more the rule than the exception and potentially lengthening development times.

These were the conditions that existed in 2003. Of the many companies that were then part of the Embedded Linux Consortium (no longer in existence), only a few remain and only LynuxWorks and MontaVista have been successful.

Appendix B: 2006 and 2007 survey methodology

EMF interviewed embedded development engineers via a comprehensive survey designed to elicit information regarding current and anticipated tool usage, design starts, completions and cancellations, development (host) and target platforms, microprocessors used, desirable and undesirable product features, vendor evaluation criteria and purchasing decision processes, among other important information. Responses from 606 embedded developers comprised the 2007 comprehensive survey results, which explored the attitudes, preferences and values of embedded developers regarding their current and projected use of embedded technologies usages and best practices. The survey was constructed so responses could be evaluated from many perspectives, including total response, specific job title of the respondent, architecture employed in embedded design, processor family, and each of ten embedded vertical market applications (e.g., telecom, industrial controls, etc.).

The survey questionnaire was developed jointly by Embedded Market Forecasters (EMF) and Wilson Research Group and programmed for a web-based survey deployment by Wilson Research Group. A base of nth-name selected U.S. subscribers to *RTC* magazine, subscribers to *RTC's COTS Journal*, and subscribers to *Military and Aerospace Electronics*, who indicated on a qualification card a professional area of interest in embedded development, were used as the sample. In addition, Wilson Research/EMF worked with e-Rewards, a company that has access to 19,000 embedded developers, beginning with a statistical base of 2,000 respondents. Starting on May 15, 2007, each member of each sample was sent an email explaining the purpose of the study that they were statistically selected, and that their participation would help embedded vendors better service the developers. Each individual in the database was identifiable by a PIN number that respondents used when they went on-line. This served to insure that the appropriate individual sent responded to the survey and that the response was one-time only. One follow-up email was sent to all non-responding members of the sample two weeks after the original mailing.

Six hundred and six developers responded to the 2007 online survey, of which 43 were hardware engineers, 231 were software engineers, 131 were systems engineers, 54 were firmware engineers and 113 were engineering managers. In addition, 34 developers gave titles other than these listed. This provided an excellent distribution of experiences and viewpoints from which to draw inferences and conclusions. Statistically, the response is at a 95% confidence level, plus or minus 4.1%.

EMF's 2006 survey of embedded developers had 705 respondents. This represented a statistical confidence level of 95% plus or minus 4.1%.

Appendix C: Vendor contact information

LynuxWorks

855 Embedded Way
San Jose, CA 95138

+1 (408) 979-3900

<http://www.lynuxworks.com>

MontaVista Software

2929 Patrick Henry Drive
Santa Clara, CA 95054

+1 (408) 572-8000

<http://www.mvista.com>

Red Hat

1801 Varsity Drive
Raleigh, NC 27606

+1 (919) 754-3700

<http://www.redhat.com>

Wind River

500 Wind River Way
Alameda CA 94501

+1 (510) 748-4100

<http://www.windriver.com>