Choosing Between Commercial and Roll Your Own (RYO) Embedded Linux OSes

Long-Term ROI Considerations

Jerry Krasner, Ph.D., MBA

September 2009

Embedded Market Forecasters

American Technology International, Inc.

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 has extensive clinical research and medical industrial experience, including the successful filing of more than a dozen 510k submissions.

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. He has been the principal investigator of several NIH funded clinical research programs.

Dr. Krasner was formerly Chairman of Biomedical Engineering at Boston University, and Chairman of Electrical and Computer Engineering at Wentworth Institute of Technology.

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.

Copyright 2009 by American Technology International, Inc, 1257 Worcester Road #500, Framingham, MA 01701. All rights reserved. No part of this book covered by copyright hereon may be reproduced or copied in any manner whatsoever. 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.

I. 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).

In December 2007 EMF published a paper entitled "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".

The purpose of that 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 the 2007 analysis was the extensive survey data that EMF compiles year-over-year from embedded developers. The findings and analysis were based on factual data. In 2007 the EMF report on embedded Linux generated angst among some of the RTOS vendors who thought that Linux was an inferior product. The 2007 data was revisited and updated in 2009 showing that Linux remains a viable OS choice for most embedded developments and a preferred choice for many.

The 2007/2009 EMF conclusions were:

- 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 roll your own (RYO) 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.
- The use of Linux for embedded designs is no longer restricted to a few applications. Furthermore, EMF design outcome data demonstrated that developers can consider using a Linux OS without concern that their end product will be inferior to end products produced with a proprietary RTOS.

Now, EMF is taking a look at the ROI between commercial Linux and "roll your own (RYO)" Linux developments. It's a natural assumption that RYO Linux development is the most cost effective option because there are no subscriptions or support costs. You simply download the code and begin development, but is this really true? Let's determine if these Linux developments are really "free".

The use of Linux for embedded development is an accepted fact for most 32 and 64-bit processors. What continues to be debated is whether RYO Linux developments produce a comparable ROI to developments that use a commercial Linux product. Much has been written on both sides of the debate. They can't both be true.

For more than a decade, EMF has written reports and presented statistically valid data for the benefit of embedded developers. Embedded vendors have used EMF reports to

better understand and serve the needs of embedded developers. The EMF leadership has now included CEOs and CFOs looking to make better strategic economic decisions.

The purpose of this paper is to present today's developers with an analysis of design outcomes using in-house Linux versions versus commercial Linux offerings based on factual data derived from embedded developer responses to detailed EMF surveys.

In this report, factual embedded developer data is used to present an ROI analysis for Linux designs based on the use of commercial Linux products and those based on free downloaded Linux with in-house development tools.

II. Calculating ROI from the EMF Database

There are different ways of assigning cost; some direct, some indirect and some that are speculative as they depend upon unforeseen events that may or may not occur.

These are:

- Direct costs
- Intrinsic costs
- Extrinsic costs
- Risk costs

Direct costs are easy to determine once they are measured. From the extensive EMF database from surveys of embedded developers, the following data can be determined:

- Average number of software developers per project
- Average number of months required to go from design start to product shipment

Note that this can be calculated for any product or combinations of products (e.g., total commercial Linux developments and total RYO Linux designs – or Monta Vista/Blue Cat/Wind River Linux products). By multiplying these numbers together one gets the total average number of software developers per project.

This is a direct cost. By multiplying the estimated monthly loaded cost per developer by the number of developers per project and the number of months it takes to ship a product, one derives the Direct Cost.

Intrinsic costs are easy to determine once they are measured. From the EMF database the following information can be determined:

- Percent of designs cancelled and the number of months the project has run before cancellation
- Percent of designs completed behind schedule and the number of months that the project is behind

These are examples of Intrinsic Costs and they can be calculated by multiplying the number of developers per project by the percent of design cancellations (or by the percent of behind schedule completions) and by the number of months either behind schedule or before cancellation.

Extrinsic costs can be conceptually considered. However, they cannot be directly quantified. From the EMF database, the percent of final designs that are within 30% of the pre-design expectation can be determined. EMF has compiled year-over-year data regarding pre-design expectations with final design outcomes. EMF believes that final designs that are within 20% of pre-design expectations are very good outcomes, whereas designs that are within 30% of pre-design expectations represent a satisfactory design outcome.

Developers were asked to report how close their final design approximated their pre-design expectation for Performance, Systems Functionality, and Features & Schedule. For each of these expectations, they could respond:

Within 10%
Within 20%
Within 30%
Within 40%
Within 50%
Not within 50%

These are the extrinsic costs. They provide a comfort level – or lack of one – and might suggest that requirements might have to be changed or that features might have to be removed. Another extrinsic cost would involve the unanticipated expense incurred by a product restart.

This might occur with a RYO Linux development wherein the company must decide to either restart the project from scratch – or seek an alternative solution. In any case there is a cost that can be estimated.

Risk costs are important to consider, but are speculative in nature. For example, the risk of a product recall might be financially devastating. Shipping a product too early or with fewer features than planned in order to meet a targeted window of opportunity would also constitute a risk cost. Finding a critical bug late in the design process, or worse after product shipment, also constitutes a risk cost.

EMF suggests that the Risk cost could be approximated by estimating the actual monetary cost of the risk and multiplying by the probability of its occurrence. So a risk that involves a potential cost of $10 million that has a 5% chance of occurring would have a projected risk cost of $500,000. This is a consideration that company executives – not embedded developers – need to consider. While not a part of the ROI calculation, risk mitigation is certainly a factor for consideration in any planning process.

III. Looking at the Numbers

Table III-1 presents data derived from the 2009 EMF Survey of Embedded Developers.

2009 EMF Data	Commercial Linux	RYO Linux
Average # SW developers/project	4.4	4.9
Total lines of code/project x1000	658.5	362.7
Months from design start to shipment	13.0	12.8
Designs completed ahead of schedule	28.8%	27.7%
Percent of designs cancelled	14.9%	14.0%
Months before cancellation	4.1	4.3
Designs completed behind schedule	37.4%	47.9%
Months behind schedule	3.5	3.4

Table III-1: Design Outcome Data

It is interesting to note that the average number of lines of code is significantly different between the commercial and RYO Linux developments. So when we look at such comparative factors as time-to-market, number of developers per project and percent of designs behind schedule, it is important to remember that commercial Linux developments, on average, involve many more lines of code than RYO.

From the data presented in Table III-1, we can calculate the Direct Cost and the Intrinsic Cost as shown in Table III-2. These costs are measured in software developer man-months.

	Commercial Linux	RYO
Software Developer Months to ship product	57.2	62.7
SW Months Wasted to Cancellation	2.2	2.4
SW Months Wasted to Behind Schedule	5.4	10.1
Total Average SW Development Months/Project	64.8	75.2

Table III-2: Direct and Intrinsic Costs

Assuming that the loaded costs of commercial and RYO developers are the same, Table III-2 shows that commercial Linux developments enjoy a 16.1% ROI advantage. One can argue that an ROI based on the average total number of lines of code per project developer would provide a better comparison. EMF data shows that, on average, the total lines of code for developers using commercial Linux offerings are 44.9% larger than for RYO developments. One can speculate as to why there is this difference in code size. One point of view would suggest that commercial developments are able to add more functionality and do cooler things since they are spending less time just getting things to work. One can further argue that commercial Linux offerings enjoy a more

efficient development process. Later in this report, Table III-4 examines how final design outcomes compare with pre-design expectations for Performance, Systems Functionality, and Features & Schedule. The fact that *all* major commercial Linux products outperform RYO (as well as the industry average) suggests that the development process produces better design outcomes notwithstanding the larger average lines of code.

EMF suggests that mid to upper level management and executives give this considerable thought.

Table III-3 summarizes the Direct and Intrinsic Cost analysis with corresponding ROIs.

	Commercial Linux	RYO
Software Developer Months to ship product	57.2	62.7
SW Months Wasted to Cancellation	2.2	2.4
SW Months Wasted to Behind Schedule	5.4	10.1
Total Average SW Development Months/Project	64.8	75.2
Assuming loaded cost/ developer = $10,000/mo		
Average SW developer cost/project	$647,572	$752,116
ROI Advantage	16.1%	
Average Total Lines of Code x1000	658.5	362.7
Average Lines of Code per developer	10162.0	4823.1
ROI Advantage - size of project	52.5%	

Table III-3: Summary of Direct and Intrinsic Costs with ROI Calculation

Based on a lines-of-code per developer analysis, commercial Linux developers would enjoy a 52.5% ROI advantage. Table III-3 also calculates the cost differential based on an average cost loading of $10,000/developer month.

Let's now consider the Extrinsic Cost based on final design outcome as compared with pre-design expectation.

Table III-4 presents the percent of final designs that are within 30% of pre-design expectations.

	Industry Ave	Monta Vista	Blue Cat	WRS Linux	RYO
Performance	74.5%	84.4%	83.6%	79.4%	71.1%
Systems Functionality	74.2%	88.3%	85.5%	79.3%	65.9%
Features & Schedule	69.6%	86.9%	76.4%	73.9%	65.9%

Table III-4: Comparative Design Outcomes within 30% of Pre-Design Expectation

It is interesting – and important – to note that each of the most popular commercial Linux products outperform RYO in this analysis. As explained previously, extrinsic costs cannot be quantified easily, and is subject to interpretation by the user. It is clear, however, that the use of a commercial Linux product provides better design outcomes than RYO.

IV. Addressing the Broader Need – Minimizing the Cost of Restarts

It is surprising to EMF that the leading commercial Linux vendors chose a marketing strategy to compete head-to-head with RYO Linux. To EMF, the appropriate product strategy would be to include a “lifeline” to RYO developers that are stuck in their development with little “wiggle room” other than to either cancel the project or restart. It would not only make marketing sense for commercial vendors, but they would have a captive market for the next internal development project.

Following this approach, let’s look at those situations in which a RYO project could run into “the wall”.

There is nothing intrinsically wrong with open source software. There must be some explanation as to why, year-over-year, commercial Linux products deliver better design outcomes. The EMF data presented in this paper shows this to be the case. So why do we get these results?

The fact that *all* major commercial Linux products outperform RYO (as well as the industry average) suggests that the development process produces better design outcomes notwithstanding the larger average lines of code. EMF suggests that mid to upper level management and executives give this considerable thought.

Starting new design projects with semiconductor or open source Linux technology can be very appealing. It can provide a quick start to development. Later in the development process, when software limitations, defects, and difficulties arise, the development team can find themselves constrained and locked in. For systems engineers wishing to customize their distribution, there is no easy access to source code, patches, and information on how to build binaries. It's difficult to integrate code from various open source communities, outside contractors, and even team members, while at the same time insulating the project from errors.

Furthermore, RYO development teams are required to keep up with the pace of open source development. This can be a burden when the team needs to deliver a new embedded product on time. With RYO, the team is responsible for porting or backporting any missing features to the kernel version, integrating the tools, creating and then maintaining the build system, etc. With RYO there is a significant amount of work to be done just to get an initial build done so development can begin. This, of course, is the promise of commercial Linux to ease this burden. There is an intrinsic cost for time and effort lost when a build fails after some 8-10 hours due to some missing source file.

It is not a secret that commercial embedded Linux vendors can offer tools, technical support, and bug fixes proven to improve design outcomes. The problem is that it can be difficult to transition midstream from a RYO approach to a commercial solution.

Recently, MontaVista addressed this need and opportunity with the release of MontaVista Linux 6 (MVL6) to address the needs of embedded developers where they

are in their development cycle with a complete embedded Linux distribution and developer tools. EMF expects that LynuxWorks (Blue Cat) and Wind River/Intel (Linux) will take similar steps. Mentor Graphics already recognizes this as a need.

EMF believes that this will revolutionize Linux developments and reduce the disparity between commercial and RYO design outcomes.

Let's take a brief look at MVL6 as this should become the model for commercial Linux products. By using the de-facto standard open source build system, developers can leverage their existing work and move to MVL6 with the assurance of feature compatibility and a clear migration path. MVL6 makes it easy to relocate and maintain local build environments in order to get teams developing sooner while leveraging open source software and insulating projects from community changes.

MVL6, for example, is feature compatible with the Linux semiconductor providers and at a current or higher kernel rev. Other commercial Linux offerings currently require backporting of needed features with their products. EMF believes that this will change and others will follow the MontaVista model.

The MontaVista Integration Platform (MVIP) is built on BitBake, an open source technology that is used as the basis of the Open Embedded project. It's non-proprietary and supports all Open Embedded recipes so that developers can easily leverage a very broad collection of open source software without major effort.

With the MontaVista Zone Content Server, which acts as a source mirror for all the MVL6 content, developers are assured that what they need to conduct initial and subsequent builds is always accessible. Hence, no lost servers, missing code, or builds that die late at night due to missing source code.

MontaVista has taken the commercial embedded Linux marketplace in a new and appropriate direction. EMF believes that other commercial Linux vendors will need to move in this direction.

V. Summary

By offering a lifeline to RYO Linux developers, who may find themselves in an impossible or intractable development /design position, to move to a commercial solution (without needing to restart the development) and by assuming the responsibility to keep up with the pace of open source development (and to test and integrate the new developments that are worthwhile), the industry has achieved an integration between the advantages inherent in both embedded RYO and commercial Linux developments.

Linux developments based on commercial Linux products continues, year-over-year, to outperform RYO Linux developments. All commercial Linux offerings outperformed RYO Linux developments.

In addition, final design outcomes as compared with pre-design expectations were far better with commercial Linux developments than with RYO developments.

As the use of Linux in embedded developments continues to grow, the ability to unite RYO with commercial Linux with a common support package should facilitate these opportunities.

Markets grow in leaps and bounds as new tools and capabilities enable developers to rethink their development concepts. It is to the advantage of CFOs to understand these occurrences and to look to the long term benefits that can accrue.