> *And the decision is not just about costs: maintaining a home-grown Linux environment entails schedule, quality, and compliance risks that are eliminated by the use of a guaranteed commercial-grade distribution.*

Beyond the Yocto™ Project:

The key requirements for a stable, production-grade Linux

AUTHORS

Jim Gallagher
Senior Marketing Team Lead, MontaVista Software, LLC.

Tim Weeks
Senior Consultant, TKO Marketing Consultants Ltd.

Contents

EXECUTIVE SUMMARY 3

SOLUTION OVERVIEW..........5

UNDERSTANDING THE LONG-TERM COSTS OF PRODUCTION-GRADE SOFTWARE QUALITY 5

CONCLUSIONS 10

MIGRATION TO SUPPORTED LINUX 10

Executive Summary

Home-grown or commercial Linux: a binary choice?

Are you a home-grown Linux user?

Or do you use a commercial-grade Linux distro provided by a third party such as MontaVista?

It's customary to present this as a binary choice. But in the real world of the embedded developer, it's becoming increasingly common for development teams to be both home-grown and commercial Linux users. And there's one very good reason for this: the Yocto Project.

The Yocto Project has given an extraordinary boost to the early-stage productivity of developers. By using a Yocto Project-compatible distro provided by an SoC manufacturer, or by using the Yocto Project to build their own Linux distro, developers can massively reduce the interval between creating a product idea and starting development of the application.

So a Yocto Project Linux is a great way to start an embedded development.

But is it a great way to finish it?

It's easy to underestimate or even overlook the costs and risks associated with maintenance and support of a production-grade Linux distribution. When a hardware product is shipped, the Linux operating system on which it runs has to be stable, bug-free, and resistant to malware and other security threats.

Now in a new white paper, MontaVista has itemized the costs of maintenance and support over the lifetime of a typical commercial or industrial embedded device. The calculation depends to some extent on a project's complexity, but it shows that there is typically very little cost advantage in maintaining an in-house Linux OS compared to the cost of licensing a commercial-grade Linux distro from a provider such as MontaVista.

And the decision is not just about costs: maintaining a home-grown Linux environment entails schedule, quality, and compliance risks that are eliminated by the use of a guaranteed commercial-grade distro.

On the grounds of stability, security and reliability, there is a clear argument in favour of commercial Linux for production implementations of a product design. For prototyping, however, many developers still find a Yocto Project distro provides the quickest, easiest route to first hardware.

Which is why the old binary choice – home-grown or commercial Linux? - is binary no longer: more and more development teams will in fact use each at the appropriate stage of development.

yocto™
PROJECT
COMPATIBLE
MontaVista
Leveraging unique abilities of Yocto Project to deliver robust and scalable embedded Linux products

Fig 1: Commercializing embedded Linux™ on Open Yocto™ framework

Solution Overview

From its inception, Linux has been an open operating system platform which provides a broad range of optional features and extensions. In the embedded version of Linux, this modularity gives developers the potential to modify and tailor their OS to meet precisely the functional requirements of their platform and application, while excluding unwanted and unnecessary features to keep the OS image as small and simple as possible.

It is this very modularity which explains the rising popularity among embedded developers of the Yocto Project: it provides a set of templates, tools and methods which enable users to select from the myriad drivers, interfaces and other elements available in Linux to put together a custom distribution. Quickly and easily, users can build a Linux platform, and use the Yocto Project and open-source code to implement the features and functions that their end product requires.

While the Yocto Project provides an excellent platform for making quick and productive progress with prototyping on x86, ARM or MIPS hardware, at some point in the development of a commercial end product the OEM has to make a long-term plan for the product which ensures its stability, security and reliability over its production lifetime.

Once a development team has started a prototype on the Yocto Project, it is natural to want to stay on the same, familiar platform. But developers need to apply a rational and realistic method for assessing the relative costs and risks associated with a production-grade product, either by maintaining and supporting their Linux platform themselves via the Yocto Project, or by using a Yocto-compatible commercially maintained Linux package. This article discusses this issue from the point of view of a commercial Linux provider, MontaVista, which is also an enthusiastic supporter of the Yocto Project and a member of its governing Advisory Board.

Understanding the long-term costs of production-grade software quality

The genius of the Yocto Project is to provide a coherent and readily understood framework for selecting and sourcing the elements of a custom Linux distribution, and to give users a set of build tools that enable them to quickly and reliably bring up their tailored Linux on a target processor board. It also provides a standard way to share Linux build projects with the community.

Many embedded engineers first encounter the Yocto Project via the reference board supplied by the manufacturer of their chosen microprocessor, companies such as Cavium, Intel, NXP or Xilinx. These semiconductor manufacturers have been eager to fund and support the Yocto Project, seeing it as a way to simplify and accelerate the process of implementing a large, full-featured OS on their large and complex devices. Many processor makers as a matter of course now make their own Linux distributions, optimized for their own Systems-on-Chip (SoCs) and compatible with the Yocto Project specifications.

At some point after board bring-up, however, every development team will face the need to seek support. Typical support scenarios could include:

- The developer wishes to modify the Linux distribution that a semiconductor vendor bundled with its SoC reference board to support its custom hardware. The semiconductor vendor's bundled Linux distribution, which worked perfectly on its own board, does not work perfectly, or even at all, on the new custom hardware.
- The OEM decides it needs added Linux packages such as advanced security (for instance Secure Boot), virtualization (container or KVM), or carrier-grade packages such as Java or Qt, that are not supplied or supported by the semiconductor vendor.

Fig 2: *Testing boards in a development engineering lab. Engineering support is not a core function of the Yocto Project.*
(Image credit: MontaVista Board Farm)

It is common to assume that support and help with getting a Yocto Project-compatible Linux to work will be forthcoming either from the processor vendor itself, or from the Yocto Project community. In reality, however, processor vendors are not software vendors: they provide a free Linux distribution as an enabler for their product, but they do not make resources available to support users of it. They also do not expect to maintain each processor's Linux distribution beyond its first release. Even if they do release patches for a legacy distribution, they are not willing to spend the time and resource necessary to have it integrated into the official Linux code base.

This reflects the fact that these distributions are not intended to be production-grade software: they are not tested and verified to production-grade quality, and they cannot guarantee compatibility with device drivers and other code which are integrated into the official Linux code base.

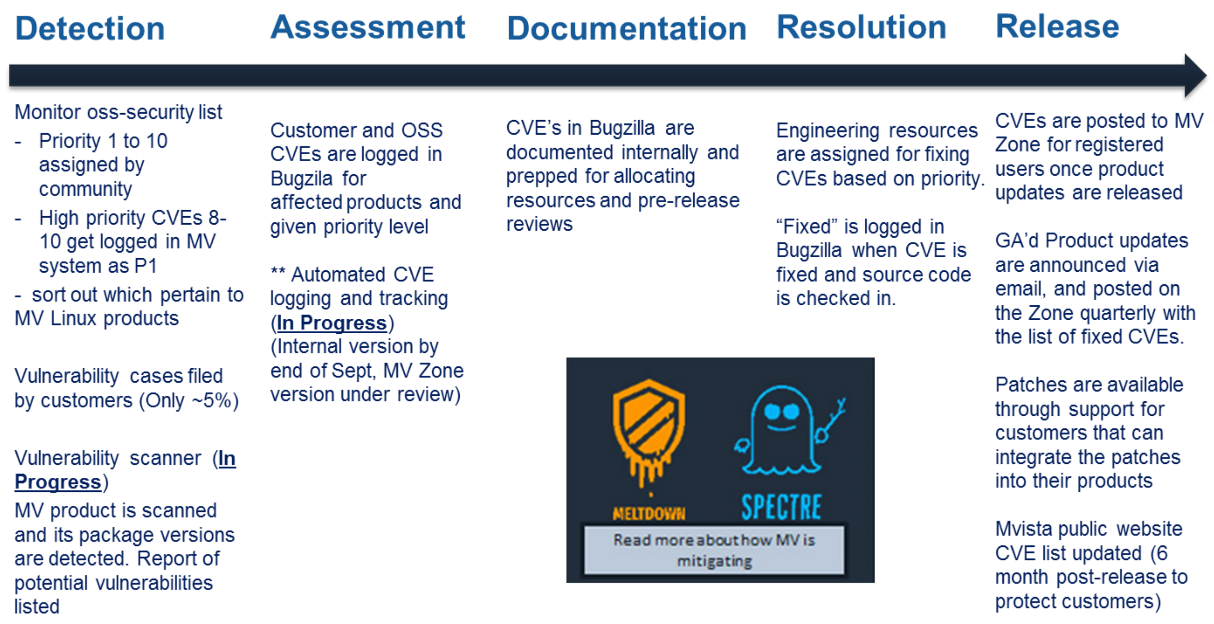
So a processor vendor's Linux helps a project get started quickly: once started, however, the developer is mostly on their own and unsupported.

How different is the situation with the Yocto Project? The project itself is an open-source collaboration under the aegis of the Linux Foundation, and supported by large and well-resourced companies. But there is no support organization within the Yocto Project. There is a community; but the response from the community to requests for help varies in a more or less unpredictable way. If a developer posts a query, a member of the community might see it, might have encountered the problem, might know the solution and might then post a helpful answer.

But if a developer posts a query that is not interesting or pertinent to members of the community, or that is too specialized or application-specific to have been experienced by others, or that is unclearly or confusingly described, or that does not readily yield to attempts at resolution, the community involvement will be provided on a best-effort basis and with no guarantee that the issue will be resolved. And of course it is impossible to know in advance which outcome the developer is going to meet.

So if support comes from the Yocto Project community at all, it will be on a haphazard, chance basis, and it might or might not produce an input that helps to resolve the developer's problem.

What this highlights is that, by choosing to use the Yocto Project, a developer is working on her or his own. This has many advantages: the platform is free, it provides for easier customization, and its direction is tuned to the needs of users rather than of a proprietor. But equally this means that there is no organization standing behind users to ensure the quality and stability of the software on which their products depend. To be clear, the released Yocto versions are vetted and tested to some degree. But as the Yocto developers and maintainers move forward to the next release, their focus is on the future, not on previous versions. The burden then falls on the developer to continue to maintain their chosen Yocto Project product with bug fixes and Common Vulnerabilities and Exposures (CVEs).



Detection

Monitor oss-security list
- Priority 1 to 10 assigned by community
- High priority CVEs 8-10 get logged in MV system as P1
- sort out which pertain to MV Linux products

Vulnerability cases filed by customers (Only ~5%)

Vulnerability scanner (**In Progress**)
MV product is scanned and its package versions are detected. Report of potential vulnerabilities listed

Assessment

Customer and OSS CVEs are logged in Bugzila for affected products and given priority level

** Automated CVE logging and tracking (**In Progress**)
(Internal version by end of Sept, MV Zone version under review)

Documentation

CVE's in Bugzilla are documented internally and prepped for allocating resources and pre-release reviews

Resolution

Engineering resources are assigned for fixing CVEs based on priority.

"Fixed" is logged in Bugzilla when CVE is fixed and source code is checked in.

Release

CVEs are posted to MV Zone for registered users once product updates are released

GA'd Product updates are announced via email, and posted on the Zone quarterly with the list of fixed CVEs.

Patches are available through support for customers that can integrate the patches into their products

Mvista public website CVE list updated (6 month post-release to protect customers)

Fig 3: CVE Resolution Process

The same argument applies to quality assurance and maintenance. For testing and software quality verification, the Yocto Project provides a platform to facilitate the development of test cases. But this is not the same thing as providing the actual test cases themselves. There are almost no ready-made, off-the-shelf test cases for embedded software available in the Yocto Project.

Again, this is hardly worth worrying about at the early prototyping stage, when the emphasis is on creating a viable product which implements the features specified by marketing. But in the design's transition from prototype to production-grade, manufactured unit, comprehensive testing is required to verify that it works without a glitch under all rated operating conditions. This requires that the OEM develop or adapt community tests. For this later phase of the product development cycle, the Yocto Project framework again leaves the developer on his or her own to verify and maintain the quality of their Linux platform.

This analysis extends beyond the first market release of the product to the remainder of its production lifecycle. The Linux on which any embedded product design is built requires periodic updates, to take account of emerging security flaws, to provide new device drivers and so on. Who is going to perform these updates?

Semiconductor vendors do not as a rule update their Yocto Project-compatible Linux distributions – they are optimized one-time-only for a specific part, and are not subsequently updated. To incorporate new features or bug fixes, a developer must move to the next version of the Linux distribution.

If an OEM uses its own, custom Linux built with Yocto Project tools, it falls to that OEM to update it. A quick calculation of the potential cost of such maintenance is often misleading. There is a common perception that the scope of Linux maintenance is relatively limited. Experience shows it is not. The table below gives a calculation of the typical commitment of time required to support a home-grown Linux distro in a relatively simple end product, a low-end edge networking device.

Engineering function	**Full-time equivalent engineer time (years)**
Development phase:	
BSP bring-up and customization	0.75
Software build	1.5
QA and test content generation	2.5
Sub-total	**4.75**
Customer trial and FCS	
BSP maintenance	0.25
Software build	1
CVEs and bug fixes	2
Continuous QA	1.5
Export compliance	0.5
Sub-total	**5.25**
Lifetime maintenance (annual time commitment)	
Software build	0.5
Maintenance	1
Continuous QA	0.5
Sub-total	**2**

The lifetime maintenance value of two full-time equivalent engineer years is a recurring annual commitment once the device has gone into production.

The cost and risk associated with the engineering workload described in the table – a reasonable estimate drawn from real-world experience – should be borne in mind when making a cost and return-on-investment decision about whether to pay for a commercial Linux subscription. The commercial Linux offering takes care of all the functions listed in the table, eliminating entirely the cost of providing and managing the engineer time listed above.

Conclusions

Migration to supported Linux

None of the above is to argue against use of the Yocto Project – quite the contrary. MontaVista is a member of the Yocto Project Advisory Board precisely because it is such a productive tool for early-stage embedded Linux development. The Yocto Project will enable embedded Linux developments to see the light of day that would before have been no more than a developer's idea: that's good for the world at large, and good generally for the embedded Linux community, of which MontaVista is part.

But when a proof-of-concept or early prototype are ready to become fully-fledged products, a supported Linux distribution such as MontaVista addresses all of the problems described in this article. A commercial Linux distribution such as MontaVista is:

- Fully supported by a full-time team of dedicated support engineers. These engineers know the distribution intimately, and can quickly and accurately diagnose a user's problem and draw on the company's knowledge bank to implement an effective solution.
- Fully tested and verified. MontaVista runs a board farm of more than 50 processors on which it runs a comprehensive test suite for every upgrade and change to its Linux product. A Linux vendor which offers 'carrier-grade' Linux, as MontaVista does, has to be able to guarantee stability and availability. This requires minutely detailed quality assurance, and every user of MontaVista Linux benefits from the stable, bug-free performance that results from it.
- Permanently maintained for guaranteed lifetimes. MontaVista Linux distributions are constantly updated, using patches that are regularly incorporated into the official Linux code base. In addition, sometimes patches are applied by backporting from a new Linux kernel to an older version. While this is not rocket science, it can be tricky and convoluted to have a patch backported successfully.

And if a commercial Linux vendor offers products that are Yocto Project-compatible, then developers can be sure that software developed on a Yocto Project-based custom Linux distribution may be readily ported to the compatible commercial

Linux, thus preserving the value of development work done in the Yocto Project environment.

While the Yocto Project approach accelerates early-stage development, then, migration to a commercial Linux distribution as the base for production-grade software saves engineering time and cost, reduces development and maintenance risk, and assures OEMs of a stable and secure platform for products that they attach their valuable brands to.

The Yocto Project will and should continue to grow in popularity. The mistake that an OEM should avoid making is to underestimate the cost and risk associated with basing a production part on Yocto Project-based software.

This White Paper is for informational purposes only. MONTAVISTA MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS WHITE PAPER. MontaVista cannot be responsible for errors in typography or photography.

Information in this document is subject to change without notice.

MontaVista Software, LLC | 2315 North 1st Street San Jose, CA, 95131 | www.mvista.com

DOC. ID. (MVTP-BeyondYocto-031518)