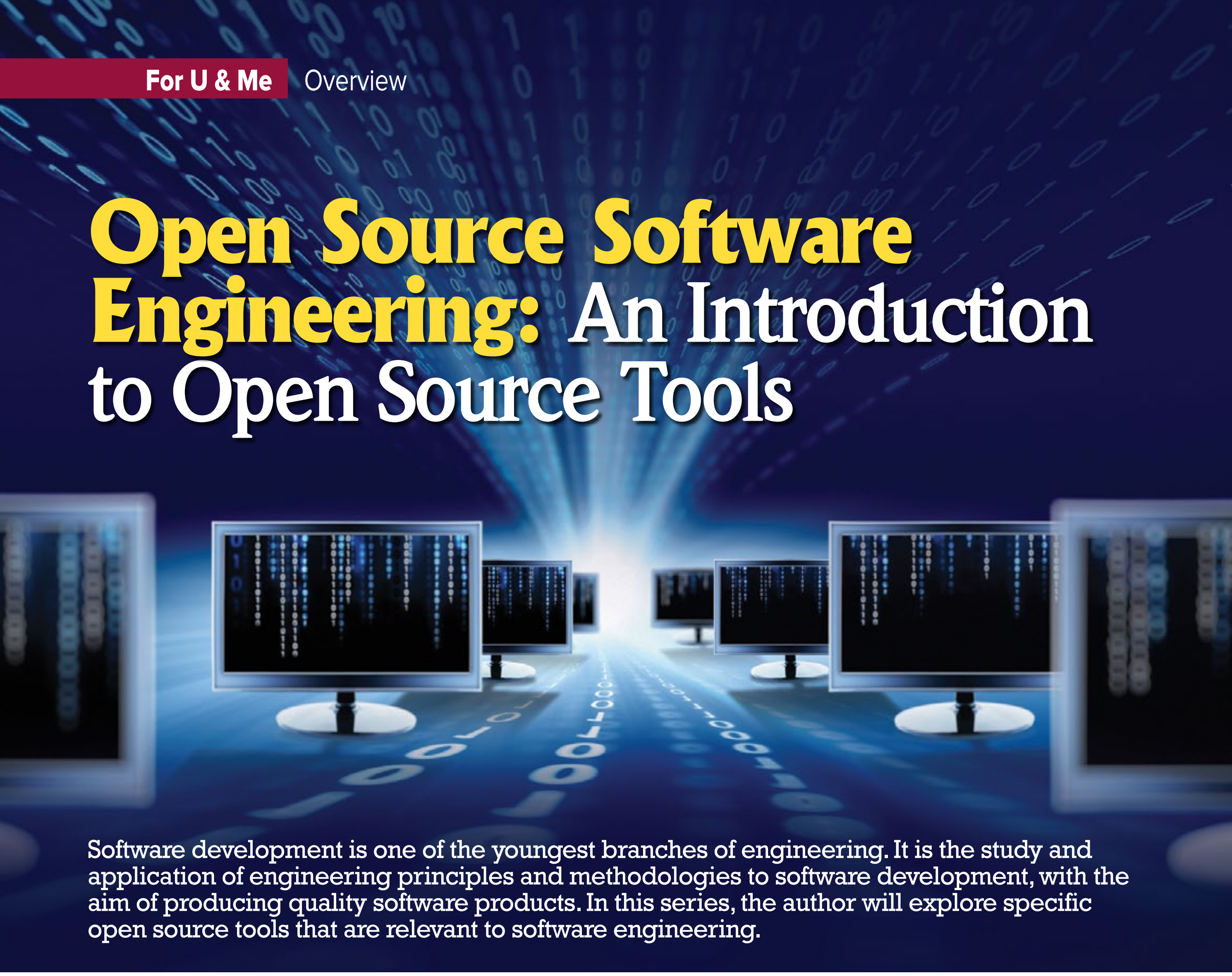

Open Source Software Engineering: An Introduction to Open Source Tools

Software development is one of the youngest branches of engineering. It is the study and application of engineering principles and methodologies to software development, with the aim of producing quality software products. In this series, the author will explore specific open source tools that are relevant to software engineering.

A long time back, I remember reading a popular white paper on software engineering. It said that software engineers could be compared to a cobbler's barefoot children: *"They make tools and applications that enable users in many domains to perform their work more effectively and efficiently, yet frequently, they do not use those tools themselves."* To some extent this remains true even today, because software product engineering still requires a lot of support, from the tools point of view. Thanks to open source, we not only get the source code for development, but also get a bunch of tools to deliver high quality products.

The relevance of open source software engineering

Now, one may ask, "Why do we need to have such tools? What is the importance of following software engineering processes?" Well, the answer is simple. Based on experience, we know that building a product inside a lab and getting a momentary high is quite different from deploying a commercial quality product to a customer who pays for it. In order to achieve the later, project managers, product engineers, architects and quality professionals face multiple challenges. To a larger extent, these challenges are overcome by implementing various processes and adopting different lifecycles like Waterfall, Agile, etc.

In each of these lifecycles, there are engineering activities that are defined, like requirement analysis, design, coding, customer demos, etc. These engineering activities help software teams to set up activities that are benchmarks and repeatable. This eventually builds quality in each cycle and ensures predictability in software delivery. Quality needs to be controlled and managed throughout the software development lifecycle, or it could lead to customer dissatisfaction or even major disasters, when projects fail on a large scale. The schedule is another critical element that needs to be managed throughout the software development process.

There are innumerable organisations that have released various expensive tools to manage software products, but these are beyond the budget of start-ups and entrepreneurial ventures. Due to constantly

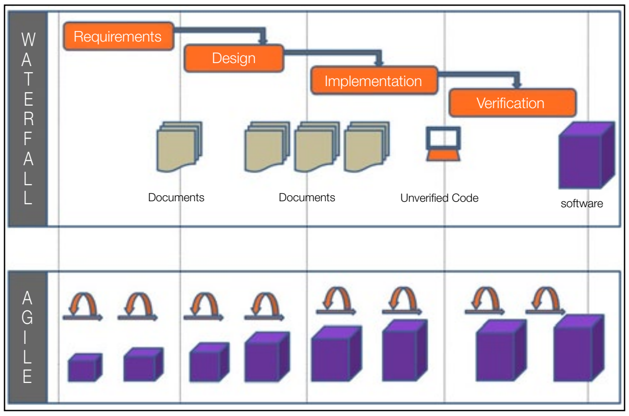

Figure 1: Waterfall and Agile models

Table 1

Functions	Tools
Code and build	Gvim, Yocto, QEMU, lxr
Check code quality	Cpp check, lcov, Code striker, Sparse
Security and scalability	Wireshark, Phoronix, Nmap
Qualification	Linux Test Project
Automation	Cruise control, Auto test
Diagnostics	Oprofile, Kprobe, LTTng
Project management	OpenProj, Xplanner
Defect tracking	Bugzilla
Team collaboration	Alfresco, Mediawiki

changing customer demands, fewer resources and shorter timelines, smaller organisations end up adopting an ad hoc approach to software product development. The concept of 'perpetual beta', which is often misunderstood as the engineering approach to product building, takes a back seat due to such challenges. While the practice of using open source software as the source code is already popular, using open source to 'engineer open source software products' is not yet popular or is relatively less known to many in the engineering community.

Along with the engineering approach, having a strong management framework for planning, controlling and monitoring software development is equally important. At every point of development, certain metrics or measurements need to be captured for monitoring, in order to drive improvements in software project teams. If you can't measure, you can't improve. Thanks to the cross-functional nature of software product development, information flow among team members should happen in a seamless manner. Rather than having lengthy and formal communication methods, quick and easy-to-access communication and collaboration modes should be adopted for seamless information sharing and communication. In short, the *engineering approach*, *management framework* and *communication framework* are three critical elements of software product development today. Fortunately, there are umpteen options available in open source itself, which can help teams to build great software products.

From the engineers' point of view, they may be associated with various activities like coding, code review, unit testing, defect fixing, estimation, etc, all of which are very critical elements that ensure a quality product is delivered to the market. Having the necessary tools becomes important to ensure quality is integrated in each of these activities. For example, for an engineer who has written 10K lines of a C program for an embedded system, answering the following questions is very important:

- How can I ensure I am always building on a stable base of software?
- How quickly will I come to know when I do a wrong code check-in?
- How can I ensure my code does not have any major issues like memory leaks?
- How can I ensure I am not missing any semantic aspects?
- How do I author unit test cases and then automate them?
- How do I ensure my unit testing covers the maximum lines of code I have written?

All the above actions cannot be left to an individual's capability, but need to be systematically and meticulously tracked and followed up. By ensuring each of them is done to the best possible extent, the quality of the code itself will be so high that it will prevent further issues getting into subsequent phases of software development.

Open source has very simple and effective tools that can be used by both the engineering and the management communities to track the above list of actions. A simple snapshot of all these tools, along with their various functions, is provided in Table 1.

Each of these tools, which are managed as individual open source projects, can be deployed at various stages of product development, yielding specific benefits. While writing on each of these tools would become too lengthy, we have chosen a specific set of tools to write about in this series on open source software engineering. Each of these tools has been tried out and been found to be very useful.

On a concluding note, using open source tools in software engineering is not only cost effective, but also very productive. We sincerely hope this series will help product engineers, product managers, product architects and entrepreneurs, and enable them to build great software products that stand for long lasting quality. END

By: Jayakumar Balasubramanian

The author is a director at Emertxe Information Technologies *(http://www.emertxe.com)*, and has been associated with building Linux-based products for over a decade. His interest lies in building innovative models around open source. He can be reached at *b.jayakumar@emertxe.com.*