Release Engineering Processes in Open Source Projects

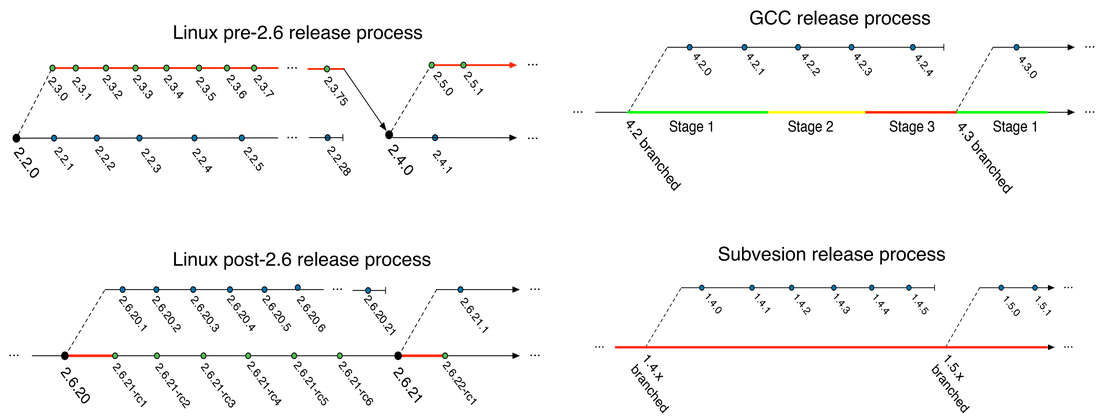
Hyrum K. Wright and Dewayne E. Perry
Emperical Software Engineering Laboratory
Department of Electrical and Computer Engineering
The University of Texas at Austin
{hwright,perry}@ece.utexas.edu

Overview

Release engineering is the part of the software engineering process during which the release artifact, usually an executable, installer, or source code package, is produced. In traditional software development methodologies, such as the spiral or waterfall models, release engineering comes as part of the release and maintenance phases. In recent years, commercial and open source software projects have begun to employ dedicated release teams. These groups are tasked with building the final shipping product.

As projects and organizations mature, the release process changes over time. Observing the changes to project releases projects can help predict and identify trends, and areas for improvement. We seek to analyze the release process evolution of open source projects, and then draw conclusions regarding the trends in release strategies observed. W e also discuss the lessons learned from the Subversion 1.5 release process. We also seek to learn how the lessons learned from this analysis can assist open source projects avoid common pitfalls.

Project Release Processes



Methodology

We have begun a qualitative study the evolution of the release process for three specific open source projects: the Linux Kernel; the Subversion version control system; and the Gnu Compiler Collection (GCC). Each project organization has significantly changed the release management process during their history, allowing us to study how process changes affected the release artifact. We can model the releases and correlate the artifacts with process events on a time line.

To carry the work forward, we will need to create a formal process model for each of the six processes under consideration and perform an analysis of each. We will use data in the form of process deliberation from email logs and process measurement data to identify the factors that necessitated the process changes. Because open source projects have greater transparency and lower institutional inertia, they provide a good opportunity for observing changing trends in software development.

Subversion 1.5 Study

In June 2008, the Subversion development team released Subversion 1.5.0. This release contained a number of new features, but arrived only after a long and painful development, test and release cycle. This protracted process confused and frustrated both users and developers. Some of the problems faced by this release include:

- A failure to learn from the past mistakes of other open and closed source projects.
- Defining a release by feature set, instead of letting features slide and releasing existing completed features.
- Developing the release-defining feature on a branch for an extended period of time.
- Not following established processes. Although processes could be adapted, the Subversion developers invented processes as the release cycle progressed.
- A lack of clear project direction.

Release Process Meta-model

Release engineering can usually be broken into several phases: stabilization, verification, and publication. We propose to create a formal process meta-model of the release engineering process, and categorize each project's sub-phases with respect to process controls, tool support, and other project factors.

Stabilization

An ideal stabilization method includes a dedicated stabilization branch and period where the feature set for the release is set and potential changes are closely reviewed. Only changes meeting established criterial, such as size or importance, are allowed into a release. For example, a large change which xes a ma jor regression may be allowed, whereas a fundamental change to the system architecture would not be.

Verification

Each project estabilishes its own guidelines for release quality, and release verication allows the project members to assure the release meets their expectations. This step also allows community members to verify the release contains the expected code, and has not been altered by the release manager. Community members may provide digital signatures for the release candidate, certifying it to be authentic.

Publication

When a release is deemed to be of sufficient quality, the release manager follows several steps to make the release widely available. These steps may include: verifying signatures provided by the community; announcing the release via email or web page; and corrdinating the availability of source and binary code in multiple distribution channels, such as web page mirrors. The release manager may also work with down stream packagers or distributors to coordinate additional user-oriented packages.

References

[1] B. W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988.

[2] J. R. Erenkrantz. Release Management Within Open Source Projects. In *Proceedings of the ICSE 3rd Workshop on Open Source Software Engineering*, May 2003.

[3] J. Estublier, D. Leblang, A. van der Hoek, G. Clemm, W. Tichy, and D. Wiborg-Weber. Impact of software engineering research on the practice of software configuration management. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 14(4):383–430, 2005.

[4] J. Howison, M. Conklin, and K. Crowston. FLOSSmole: A Collaborative Repository for FLOSS Research Data and Analyses. *International Journal of Information Technology and Web Engineering*, 1(3):17–26, 2006.

[5] J. Howison and K. Crowston. The perils and pitfalls of mining SourceForge. *Proceedings of the International Workshop on Mining Software Repositories (MSR 2004)*, pages 7–11, 2004.

[6] J. Y. Moon and L. Sproull. Essence of Distributed Work: The Case of the Linux Kernel. *Distributed Work*, pages 381–404, 2002.

[7] M. Ramakrishnan. Software Release Management. *Bell Labs Technical Journal*, 9(1):205–210, 2004.

[8] W. W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, pages 328–338. IEEE Computer Society Press Los Alamitos, CA, USA, 1987.

[9] A. van der Hoek, R. S. Hall, D. Heimbigner, and A. L. Wolf. Software release management. *ACM SIGSOFT Software Engineering Notes*, 22(6):159–175, 1997.

[10] A. L. Wolf and D. S. Rosenblum. A study in software process data capture and analysis. In *Second International Conference on the Software Process*, pages 115–124, 1993.

THE UNIVERSITY OF TEXAS AT AUSTIN

UT ECE

esel Empirical Software Engineering Laboratory