Release Engineering Processes, Models, and Metrics

Hyrum K. Wright
Empirical Software Engineering Laboratory
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712
hyrum_wright@mail.utexas.edu

ABSTRACT

No matter the development process or methodology, a software product must ultimately be released to a user in a readily consumable form. Different software products, from desktop applications to web services, may require different release processes, but each process must produce an artifact which meets an expected level of quality, and is relatively bug-free. We describe current research to model and quantify existing release processes, and an effort to prescribe improvements to those processes.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*software process models*; D.2.7 [**Software Engineering**]: Distribution, Maintenance, Enhancement—*release creation and distribution*

General Terms

Management, Standardization

Keywords

release engineering, release management, software processes

1. INTRODUCTION

A software release is the most prominent aspect of a software deployment. Whether a web service, an open source project, a commercial system, or an internally developed application, unreleased software is nonexistent software. Often, significant external resources, such as marketing campaigns, are dependent upon a successful release process. Development teams must ensure they have a sufficiently high-quality release process, to create low-fault and high-frequency releases.

Often, the release process changes as a organization grows. A small startup company will release software much differently than a multinational vendor. While developers and architects focus their energies on the design of the software itself, the release process tends to gradually evolve, with very little comprehensive design. Instead, companies and projects add release engineering techniques as required, with little forethought or structured design of the process.

The primary goal of this research is to develop techniques, methods, metrics and processes whereby the release engineering process can be quantitatively measured, reasoned about, and improved. Such techniques will benefit the software industry by reducing release process overhead, and improving the quality of the release artifact. This paper gives a brief background of release engineering as a practice, the existing research in this area, and how we intend to proceed toward our goals.

2. BACKGROUND

Release engineering is the part of the software engineering process during which the release artifact, usually an executable, installer, library 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 deployment and maintenance phases [1, 8]. Many proprietary and open source software projects employ dedicated release teams which are tasked with building the final shipping product, very literally "engineering the release."

For many development teams, release engineering can usually be broken into several phases: stabilization, validation, and publication. During the entire process, the release is overseen by one or many *release engineers*. This individual or team may coordinate release activities, determine schedule and make binding decisions regarding releases. Depending on the size of the development team, release management may be a dedicated assignment, or may rotate among team members.

2.1 Related Work

Despite the ubiquitous nature of the release process in the software development cycle, very little research has focused primarily on the release processes itself. Some of the existing literature looks at the distribution and maintenance of the release artifacts, but not the process involved in creating them.

The open source world has provided a particularly fertile source of data for examining release processes. Erenkrantz outlined release processes in several open source projects, but the work is quite dated and each of the projects surveyed has changed their process in the interim [3]. Michlmayr examined release engineering processes in open source projects, and problems that the open environment presents to the process [6, 7]. Additional researchers have also examined how release managers fit into the social fabric of open source projects [2].

Other authors have looked at where the release process fits in the software development cycle. The problem of distributing the release artifact is addressed by van der Hoek, et. al. [9], but the work is more aimed at dependency management of releases. Other work describes the Software Dock [4], a system for configuration, deployment and maintenance of software installations.

3. GOALS AND STATUS

Part of this work will be simply defining the scope of the release engineering process, and how that applies in different organizations. Many organizations task a "release team" with managing release engineering within the project, but the charter of such a team can vary widely. While the needs of every organization are different, many commonalities exist and creating a flexible, yet useful, description of the process is an important part of our work.

Our primary goal, though, is to observe and model existing release processes, and to eventually prescribe changes to those processes in an effort to improve them. Metrics exist for nearly every other aspect of the software engineering cycle. These metrics help researchers and developers alike, by quantifying and describing the process, so they can then be improved upon [10]. We aim to introduce a similar framework for use with release engineering.

In previous work on this subject, we examined details of the Subversion open source project's release cycle for version 1.5 [11]. We hope to expand this work to include lessons learned from other commercial and open source projects, particularly the problems they have encountered in their own release engineering processes, and how they addressed (or did not address) them.

We are currently working on creating a uniform database of release information for a variety of open source projects. This is in a similar spirit as the FLOSSmole data mining project [5], but focuses on only the release information for the selected projects. In the process of creating this database, we are also creating rough methods of normalizing release information to allow inter-project comparison. In addition, we are developing tools to support the maintenance and improvement of the collected data.

Using this data, and our proposed models and metrics, we will then prescribe improvements to the projects' release processes. This approach is not limited to just open source projects, and we also plan on examining selected proprietary release engineering practices.

We also plan to examine the current tooling infrastructure surrounding release engineering. Anecdotal evidence seems to suggest that what tools exist are homegrown, and not uniform throughout the software industry. These tools could help standardize release artifact creation and notation, making releases and release histories easier to reason about. By developing a common tool infrastructure, release engineering processes could be streamlined and improved.

4. CONCLUSION

Release engineering for most teams is, at best, an ad hoc, homegrown conglomeration of scripts, tests, and processes which have grown up over the course of many years. Tool support is often hit-or-miss, and little research which addresses the issues faced by teams creating the large software projects common in development environments today. In addition, very few standards exist for release engineering processes and artifacts. We hope to address these issues through the above outlined research.

Every software team must eventually release the end result of its labors. The release process is a critical part of the software development cycle, and by better understanding the release process, software engineers, as well as development teams, will be better able to create high-quality software releases.

5. REFERENCES

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

[2] K. Crowston and J. Howison. The social structure of free and open source software development. *First Monday*, 10(2), 2005.

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

[4] R. Hall, D. Heimbigner, A. van der Hoek, and A. Wolf. The Software Dock: A Distributed, Agent-based Software Deployment System. Technical Report CU-CS-832-97, University of Colorado, Dept. of Computer Science, February 1997.

[5] 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.

[6] M. Michlmayr. Quality Improvement in Volunteer Free Software Projects: Exploring the Impact of Release Management. In *Proceedings of the First International Conference on Open Source Systems*, pages 309–10, 2005.

[7] M. Michlmayr, F. Hunt, and D. Probert. Release Management in Free Software Projects: Practices and Problems. *International Federation for Information Processing*, 234:295, 2007.

[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.

[11] H. K. Wright and D. E. Perry. Subversion 1.5: A Case Study in Open Source Release Mismanagement. In *Proceedings of the ICSE 2nd Emerging Trends in FLOSS Research and Development Workshop*, May 2009.