

Releasing

YEGOR BUGAYENKO

Lecture #7 out of 8
80 minutes

The slidedeck was presented by the author in this [YouTube Video](#)

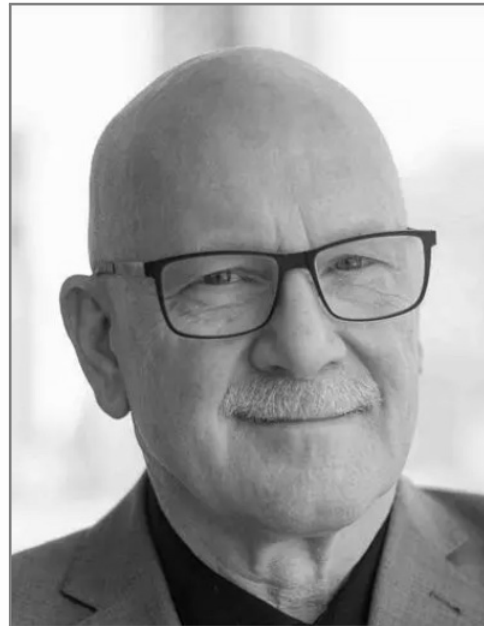
All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as web sites. Copyright belongs to their respected authors.



ANDRE VAN DER HOEK

“Simply making available and retrieving interdependent components individually neither facilitates independent software development nor encourages widespread use of large systems of systems.”

— Andre Van Der Hoek, Richard S. Hall, Dennis Heimbigner, and Alexander L. Wolf. Software Release Management. *ACM SIGSOFT Software Engineering Notes*, 22(6):159–175, 1997. doi:[10.1145/267896.267909](https://doi.org/10.1145/267896.267909)



GÜNTHER RUHE

“Release planning is typically done *ad hoc* and not based on sound models and methodologies. This is even the case when planning involves several hundred features.”

— Günther Ruhe and Moshood Omolade Saliu. The Art and Science of Software Release Planning. *IEEE Software*, 22(6):47–53, 2005. doi:[10.1109/MS.2005.164](https://doi.org/10.1109/MS.2005.164)



1. Release when it's different, not when it's good.



KAZU OKUMOTO

“**Goel-Okumoto model:** An important problem of practical concern is the determination of the point when testing should stop and the system can be considered ready for release, that is, the determination of the software release time. Two criteria are investigated: software reliability and total expected cost.”

— Kazu Okumoto and Amrit L. Goel. Optimum Release Time for Software Systems Based on Reliability and Cost Criteria. *Journal of Systems and Software*, 1(1):315–318, 1979. doi:[10.1016/0164-1212\(79\)90033-5](https://doi.org/10.1016/0164-1212(79)90033-5)



SHIGERU YAMADA

“This paper extends the problem of Okumoto and Goel [1979] by evaluating both criteria simultaneously. We discuss optimal software release policies which minimize a total average software cost under the constraint of satisfying a software reliability requirement.”

— Shigeru Yamada and Shunji Osaki. Cost-Reliability Optimal Release Policies for Software Systems. *IEEE Transactions on Reliability*, 34(5):422–424, 1985.
[doi:10.1109/TR.1985.5222222](https://doi.org/10.1109/TR.1985.5222222)

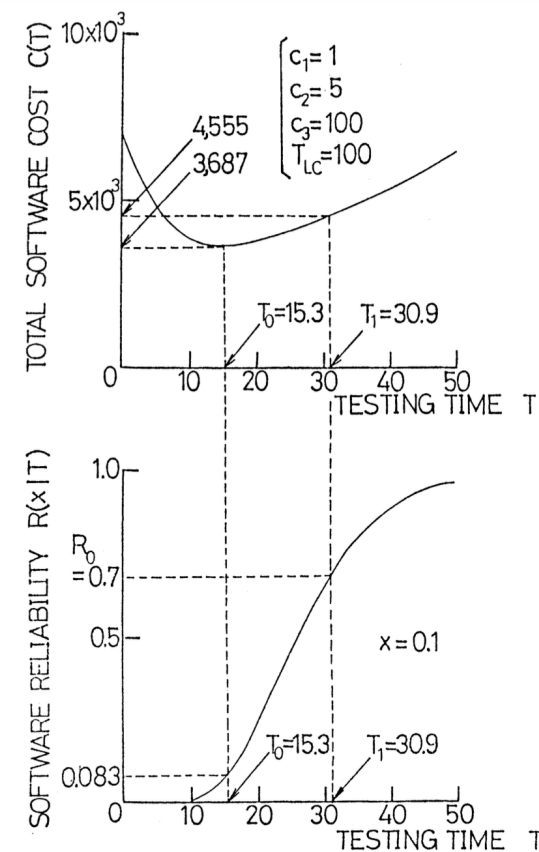


Fig. 1. An illustration of the cost-reliability optimal release problem.

“The optimum software release time is the testing time which comes closest to satisfying some pre-specified software reliability.”

Source: Shigeru Yamada and Shunji Osaki.
Cost-Reliability Optimal Release Policies for Software Systems. *IEEE Transactions on Reliability*, 34 (5):422–424, 1985. doi:[10.1109/TR.1985.5222222](https://doi.org/10.1109/TR.1985.5222222)



STEVE MCCONNELL

“The question of whether to release software is a treacherous one. The answer must teeter on the line between releasing poor quality software early and releasing high quality software late. The questions of ‘Is the software good enough to release now?’ and ‘When will the software be good enough to release?’ can become critical to a company’s survival.”


— Steve McConnell. *Software Project Survival Guide*. Microsoft Press, 1998.
doi:[10.5555/270015](https://doi.org/10.5555/270015)



NASIF IMTIAZ

“We find that the open source packages are typically fast in releasing security fixes, as the median release comes within 4 days of the corresponding security fix. However, 25% of the releases still have a delay of at least 20 days.”

— Nasif Imtiaz, Aniq Khanom, and Laurie Williams. Open or Sneaky? Fast or Slow? Light or Heavy? Investigating Security Releases of Open Source Packages. *IEEE Transactions on Software Engineering*, 49(4):1540–1560, 2022. doi:[10.1109/TSE.2022.3181010](https://doi.org/10.1109/TSE.2022.3181010)



2. Fully automate the release process,
avoiding any manual intervention.



MICHAEL NYGARD

“Releases should be about as big an event as getting a haircut. There’s an added benefit of frequent releases: it forces you to get really good at doing releases and deployments.”

— Michael Nygard. *Release It!: Design and Deploy Production-Ready Software*. The Pragmatic Bookshelf, 2007. doi:[10.5555/1200767](https://doi.org/10.5555/1200767)



JEZ HUMBLE

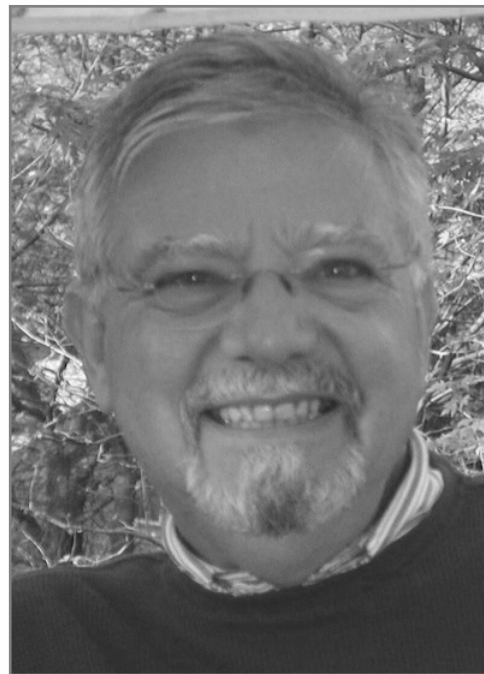
“Over time, deployments should tend towards being fully automated. There should be two tasks for a human being to perform to deploy software into a development, test, or production environment: to pick the version and environment and to press the ‘deploy’ button.”

— Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Pearson Education, 2010.
doi:[10.5555/1869904](https://doi.org/10.5555/1869904)



3. Release frequently.

Respectfully
disagree!



VICTOR BASILI

“If improving productivity is the main concern, then it may be wise to try to avoid scheduling small error correction releases. Instead the manager should try, when possible, to package small error corrections in a release with larger enhancements. ”

— Victor Basili, Lionel Briand, Steven Condon, Yong-Mi Kim, Walcélío L. Melo, and Jon D. Valen. Understanding and Predicting the Process of Software Maintenance Releases. In *Proceedings of the 18th International Conference on Software Engineering*, pages 464–474. IEEE, 1996. doi:[10.1109/ICSE.1996.493441](https://doi.org/10.1109/ICSE.1996.493441)



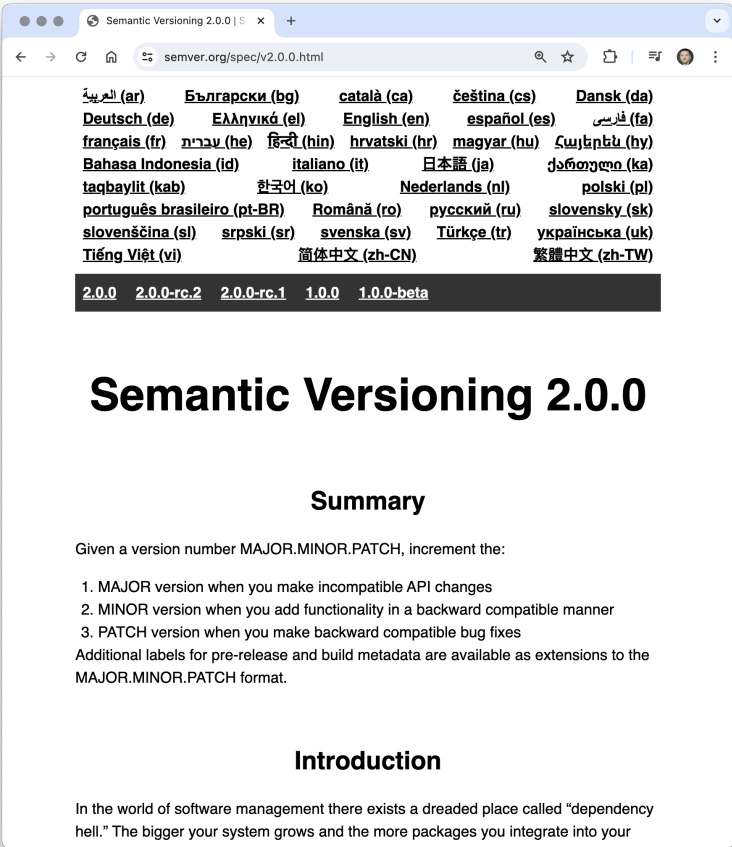
FOUTSE KHOMH

“We found that (1) with shorter release cycles, users do not experience significantly more post-release bugs and (2) bugs are fixed faster, yet (3) users experience these bugs earlier during software execution (the program crashes earlier).”

— Foutse Khomh, Tejinder Dhaliwal, Ying Zou, and Bram Adams. Do Faster Releases Improve Software Quality? An Empirical Case Study of Mozilla Firefox. In *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR)*, pages 179–188. IEEE, 2012. doi:[10.1109/MSR.2012.6224279](https://doi.org/10.1109/MSR.2012.6224279)



4. Use SemVer.

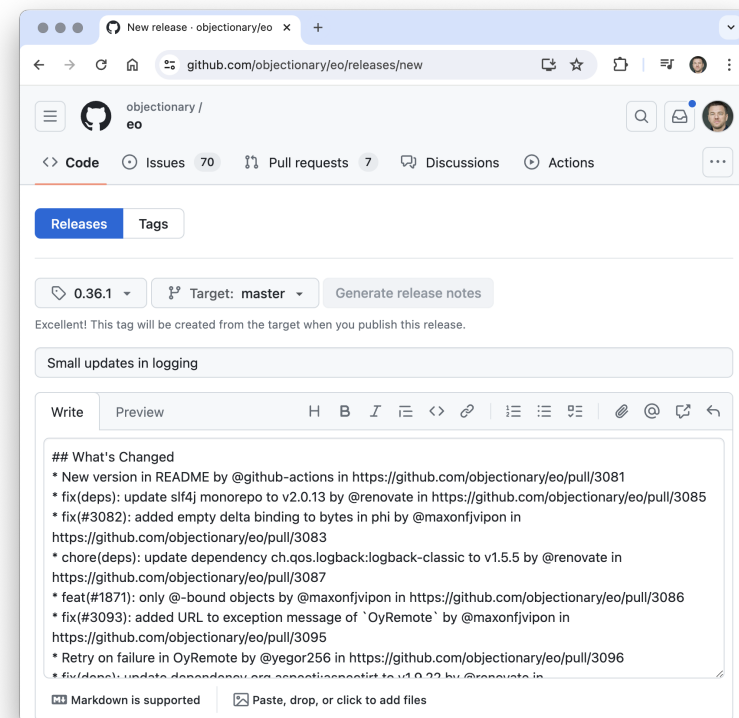


“The implication of semantic versioning is that clients may rely on dependencies subject to flexible version constraints, like 1.2.*. Such a client may safely upgrade to new micro versions (e.g., from 1.2.3 to 1.2.4), fully automated.”

Source: Patrick Lam, Jens Dietrich, and David J. Pearce. Putting the Semantics into Semantic Versioning. In *Proceedings of the SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pages 157–179, 2020. doi:[10.1145/3426428.3426922](https://doi.org/10.1145/3426428.3426922)



5. Generate release notes automatically.



“Automatically generated release notes provide an automated alternative to manually writing release notes for your GitHub releases. With automatically generated release notes, you can quickly generate an overview of the contents of a release.”

<https://docs.github.com/en/repositories/releasing-projects-on-github/automatically-generated-release-notes>

Some Release Notes Generators:

- [ClickUp](#)
- [Taskade](#)
- [Zeda](#)
- [Aha](#)
- [ReleasesNotes](#)
- [ScribeHow](#)
- [Released](#)
- [ai-github-release-notes](#)

Try Google search with “generate release notes with AI”



JIANYU WU

“We find that: 1) RN producers are more likely to miss information than to include incorrect information, especially for breaking changes; 2) improper layout may bury important information and confuse users; 3) many users find RNs inaccessible due to link deterioration, lack of notification, and obfuscate RN locations; 4) automating and regulating RN production remains challenging despite the great needs of RN producers.”

— Jianyu Wu, Hao He, Wenxin Xiao, Kai Gao, and Minghui Zhou.
Demystifying Software Release Note Issues on GitHub. In *Proceedings of the 30th International Conference on Program Comprehension*, pages 602–613, 2022.
[doi:10.1145/3524610.3527919](https://doi.org/10.1145/3524610.3527919)

6. Publish binaries.

Some Artifact Publishing Platforms:

- Maven Central for Java, Kotlin, Scala, Groovy, etc.
- Npm for JavaScript
- PyPi for Python
- RubyGems for Ruby
- Crates for Rust

References

Victor Basili, Lionel Briand, Steven Condon, Yong-Mi Kim, Walcélio L. Melo, and Jon D. Valen. Understanding and Predicting the Process of Software Maintenance Releases. In *Proceedings of the 18th International Conference on Software Engineering*, pages 464–474. IEEE, 1996. doi:[10.1109/ICSE.1996.493441](https://doi.org/10.1109/ICSE.1996.493441).

Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Pearson Education, 2010. doi:[10.5555/1869904](https://doi.org/10.5555/1869904).

Nasif Imtiaz, Anika Khanom, and Laurie Williams. Open or Sneaky? Fast or Slow? Light or Heavy? Investigating Security Releases of Open Source Packages. *IEEE Transactions on Software Engineering*, 49(4):1540–1560, 2022. doi:[10.1109/TSE.2022.3181010](https://doi.org/10.1109/TSE.2022.3181010).

Foutse Khomh, Tejinder Dhaliwal, Ying Zou, and Bram Adams. Do Faster Releases Improve

Software Quality? An Empirical Case Study of Mozilla Firefox. In *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR)*, pages 179–188. IEEE, 2012. doi:[10.1109/MSR.2012.6224279](https://doi.org/10.1109/MSR.2012.6224279).

Patrick Lam, Jens Dietrich, and David J. Pearce. Putting the Semantics into Semantic Versioning. In *Proceedings of the SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pages 157–179, 2020. doi:[10.1145/3426428.3426922](https://doi.org/10.1145/3426428.3426922).

Steve McConnell. *Software Project Survival Guide*. Microsoft Press, 1998. doi:[10.5555/270015](https://doi.org/10.5555/270015).

Michael Nygard. *Release It!: Design and Deploy Production-Ready Software*. The Pragmatic Bookshelf, 2007. doi:[10.5555/1200767](https://doi.org/10.5555/1200767).

Kazu Okumoto and Amrit L. Goel. Optimum Release Time for Software Systems Based on Reliability and Cost Criteria. *Journal of Systems and Software*, 1(1):315–318, 1979. doi:[10.1016/0164-1212\(79\)90033-5](https://doi.org/10.1016/0164-1212(79)90033-5).

Günther Ruhe and Moshood Omolade Saliu. The Art and Science of Software Release Planning. *IEEE Software*, 22(6):47–53, 2005. doi:[10.1109/MS.2005.164](https://doi.org/10.1109/MS.2005.164).

Andre Van Der Hoek, Richard S. Hall, Dennis Heimbigner, and Alexander L. Wolf. Software Release Management. *ACM SIGSOFT Software Engineering Notes*, 22(6):159–175, 1997. doi:[10.1145/267896.267909](https://doi.org/10.1145/267896.267909).

Jianyu Wu, Hao He, Wenxin Xiao, Kai Gao, and Minghui Zhou. Demystifying Software Release Note Issues on GitHub. In *Proceedings of the 30th International Conference on Program Comprehension*, pages 602–613, 2022. doi:[10.1145/3524610.3527919](https://doi.org/10.1145/3524610.3527919).

Shigeru Yamada and Shunji Osaki. Cost-Reliability Optimal Release Policies for Software Systems. *IEEE Transactions on Reliability*, 34(5):422–424, 1985. doi:[10.1109/TR.1985.5222222](https://doi.org/10.1109/TR.1985.5222222).