Dead Code

Yegor Bugayenko

Lecture #12 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video

Motivating Example

Before (wrong):

```
1 class Book
2   private int id;
3   public Book(int it)
4     this.id = i;
5   public int getId()
6     return this.id;
7
8   private int setId(int i)
9     this.id = i;
```

After (better):

```
1 class Book
2   private final int id;
3   public Book(int it)
4     this.id = i;
5   public int getId()
6     return this.id;
```

Dead Code Elimination (Compiler Optimization)

Dead code is here:

```
1 void main(int x) {
2   int a = 42;
3   if (x > 0) {
4     a = 256;
5   }
6   a = 7;
7   print(a);
8 }
```

"Dead code refers to computations whose results are never used. Code that is dead can be eliminated without affecting the behavior of the program."

Source: *Compiler Techniques for Code Compaction*, Saumya K. Debray, William Evans, Robert Muth, Bjorn De Sutter, ACM Transactions on Programming languages and Systems (TOPLAS), 22(2), 2000

Mika Mäntylä

“Dead code is code that has been used in the past, but is currently never executed. Dead code hinders code comprehension and makes the current program structure less obvious.”

— M. Mantyla, J. Vanhanen, and C. Lassenius. A Taxonomy and an Initial Empirical Study of Bad Smells in Code. In *Proceedings of the International Conference on Software Maintenance*, 2003. doi:10.1109/icsm.2003.1235447

Sebastian Eder

"We conducted the study on the level of methods in the sense of object oriented programming. The systems contains 25,390 methods. We found that 25% of all methods were never used during the complete period."

— Sebastian Eder, Maximilian Junker, Elmar Jurgens, Benedikt Hauptmann, Rudolf Vaas, and Karl-Heinz Prommer. How Much Does Unused Code Matter for Maintenance? In *Proceedings of the 34th International Conference on Software Engineering (ICSE)*, 2012. doi:10.1109/icse.2012.6227109

Unreachable/Dead Methods in Java

Table 1: Dataset Information

Software	LOCs	#Types	#Meth.	#Un. Meth.	%Un. Meth.
ArtOfIllusion 2.4.1	79,383	600	5,426	545	10%
LaTeXDraw 2.0.8	65,334	252	3,130	212	7%
aTunes 1.10.1	42,357	778	4,067	240	6%
MediaPesata 1.0	1,580	31	162	8	5%

Source: Simone Romano, Giuseppe Scanniello, Carlo Sartiani, and Michele Risi. A Graph-Based Approach to Detect Unreachable Methods in Java Software. In *Proceedings of the 31st Annual Symposium on Applied Computing*, pages 1538–1541, 2016. doi:10.1145/2851613.2851968

Simone Romano

"Although there is some consensus on the fact that dead code is a common phenomenon, it could be harmful, and it seems to matter to software professionals; surprisingly, dead code has received very little empirical attention from the software engineering research community."

— Simone Romano, Christopher Vendome, Giuseppe Scanniello, and Denys Poshyvanyk. A Multi-Study Investigation into Dead Code. *IEEE Transactions on Software Engineering*, 46(1), 2018. doi:10.1109/tse.2018.2842781

Table 5: Results regarding the relative number of dead methods born dead and became dead.

Application	*%DeadBornMethods*	*%DeadBecameMethods*
4HWC Autonomous Car	92.593	7.407
8_TheWeather	100	0
BankApplication	69.259	30.741
bitbox	98.095	1.905
Density Converter	95.139	4.861
Deobfuscator-GUI	98.718	1.282
graphics-tablet	88.698	11.302
JavaANPR	99.578	0.422
javaman	84.507	15.493
JDM	100	0
JPass	76.623	23.377
MBot	100	0
SMV APP	77.049	22.951
Mean	90.789	9.211
SD	10.673	10.673
Median	95.139	4.861

“i) ...; ii) dead methods generally survive for a long time, in terms of commits, before being buried or revived; iii) dead methods are rarely revived; and iv) most dead methods are dead since the creation of the corresponding methods.”

Source: Danilo Caivano, Pietro Cassieri, Simone Romano, and Giuseppe Scanniello. An Exploratory Study on Dead Methods in Open-Source Java Desktop Applications. In *Proceedings of the 15th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11, 2021. doi:10.1145/3475716.3475773

Subject	# Dead Functions
In-the-lab subjects	
angularjs require	32
backbone	542
canjs	492
dijon	410
dojo	411
enyo backbone	6
gwt	17
jquery	420
jsblocks	459
knockoutjs require	35
mithril	55
polymer	6
reagent	3,357
vanillajs	59
vue	266

"The elimination of JavaScript dead code leads to noticeable (and statistically significant) differences in terms of the number of performed HTTP requests only for in-the-lab subjects."

Source: Ivano Malavolta, Kishan Nirghin, Gian Luca Scoccia, Simone Romano, Salvatore Lombardi, Giuseppe Scanniello, and Patricia Lago. JavaScript Dead Code Identification, Elimination, and Empirical Assessment. *IEEE Transactions on Software Engineering*, 49(7):3692–3714, 2023. doi:10.1109/TSE.2023.3267848

Katriel Cohn-Gordon

"At Meta, in the last year alone, we removed petabytes of data across 12.8 million distinct assets, and deleted over 104 million lines of code."

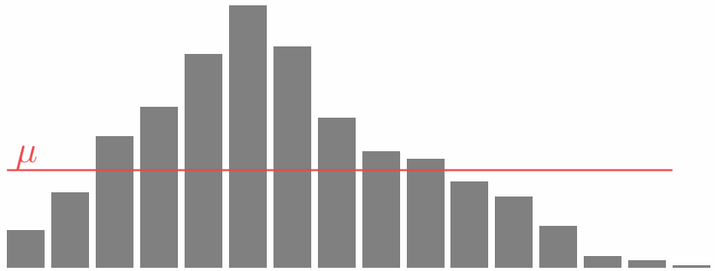
— Will Shackleton, Katriel Cohn-Gordon, Peter C. Rigby, Rui Abreu, James Gill, Nachiappan Nagappan, Karim Nakad, Ioannis Papagiannis, Luke Petre, Giorgi Megreli, et al. Dead Code Removal at Meta: Automatically Deleting Millions of Lines of Code and Petabytes of Deprecated Data. In *Proceedings of the 31st Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1705–1715, 2023. doi:10.1145/3611643.3613871

Pietro Cassieri

"The results indicate that, after removing dead methods, the internal structure of the source code significantly improves, while the space to store executable code significantly decreases along with the time to compile source code."

— Simone Romano, Giovanni Toriello, Pietro Cassieri, Rita Francese, and Giuseppe Scanniello. A Folklore Confirmation on the Removal of Dead Code. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pages 333–338, 2024. doi:10.1145/3661167.3661188

Volatility Metric



"The variance Var(g) is the **Volatility** of the source code. The smaller the Volatility the more *cohesive* is the repository and the smaller the amount of the abandoned code inside it."

Then, the mean μ is calculated as:

$$\mu = \frac{1}{Z}\sum_{j=1}^{Z} g_j \tag{5}$$

Finally, the variance is calculated as:

$$\mathrm{Var}(g) = \frac{1}{Z}\sum_{j=1}^{Z} |g_j - \mu|^2 \tag{6}$$

The variance Var(g) is the Volatility of the source code.

Source: Yegor Bugayenko. Volatility Metric to Detect Anomalies in Source Code Repositories. In *Proceedings of the 1st SIGPLAN International Workshop on Beyond Code: No Code*, pages 1–4, 2021. doi:10.1145/3486949.3486961

Volatility vs. Number of Files in a Repo

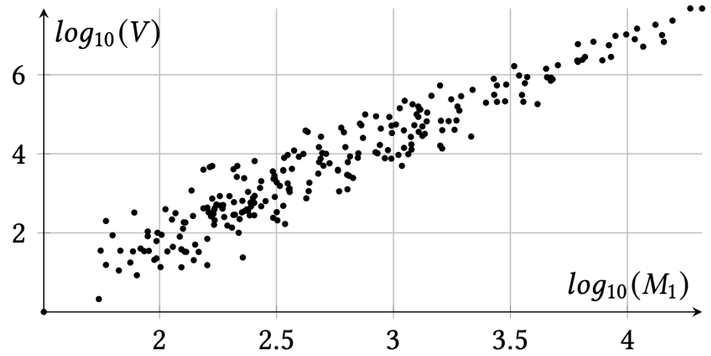


Figure 3. The relationship between the number of files in the repository and its Volatility (Z is set to 32)

Source: Yegor Bugayenko. Volatility Metric to Detect Anomalies in Source Code Repositories. In *Proceedings of the 1st SIGPLAN International Workshop on Beyond Code: No Code*, pages 1–4, 2021. doi:10.1145/3486949.3486961

Ciera Jaspan

"Our survey results show that engineers at Google strongly prefer our monolithic repo, and that visibility of the codebase and simple dependency management were the primary factors for this preference."

— Ciera Jaspan, Matthew Jorde, Andrea Knight, Caitlin Sadowski, Edward K. Smith, Collin Winter, and Emerson Murphy-Hill. Advantages and Disadvantages of a Monolithic Repository: A Case Study at Google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, pages 225–234, 2018. doi:10.1145/3183519.3183550

Monolithic Repositories

Centralization The codebase is contained in a single repo encompassing multiple projects.

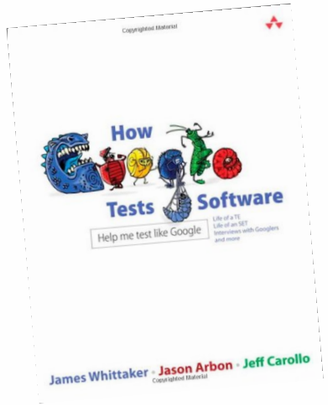
Visibility Code is viewable and searchable by all engineers in the organization.

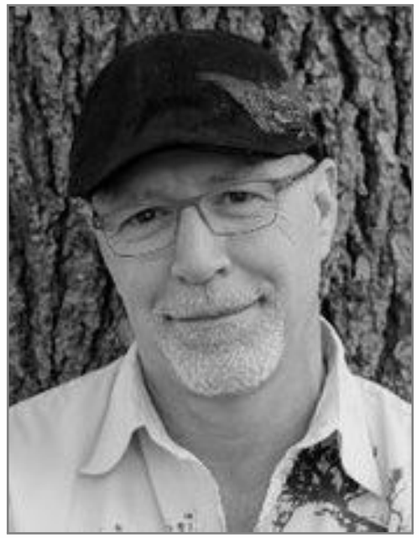
Synchronization: The development process is trunk-based; engineers commit to the head of the repo.

Completeness Any project in the repo can be built only from dependencies also checked into the repo. Dependencies are unversioned; projects must use whatever version of their dependency is at the repo head.

Standardization A shared set of tooling governs how engineers interact with the code, including building, testing, browsing, and reviewing code.

Source: Ciera Jaspan, Matthew Jorde, Andrea Knight, Caitlin Sadowski, Edward K. Smith, Collin Winter, and Emerson Murphy-Hill. Advantages and Disadvantages of a Monolithic Repository: A Case Study at Google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, pages 225–234, 2018. doi:10.1145/3183519.3183550

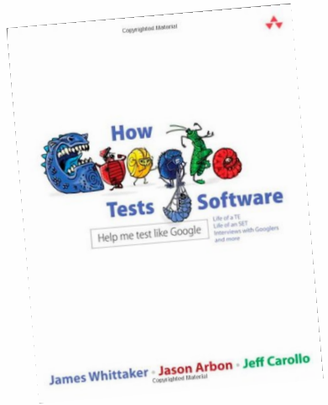




James Whittaker

"Most code at Google shares a single repository and common tool chain. This single repository makes a great deal of sense as engineers moving from project to project have little to relearn."

— James A. Whittaker, Jason Arbon, and Jeff Carollo. *How Google Tests Software.* Addison-Wesley, 2012. doi:10.1145/1852666.1852675



John Penix

“Software development at Google happens quickly and at scale. The Google codebase receives over 20 changes per minute and 50 percent of the files change every month!”

— James A. Whittaker, Jason Arbon, and Jeff Carollo. *How Google Tests Software.* Addison-Wesley, 2012. doi:10.1145/1852666.1852675

Durham Goode

"Facebook's main source repository is enormous—many times larger than even the Linux kernel, which checked in at 17 million lines of code and 44,000 files in 2013."

— Rain Durham Goode. Scaling Mercurial at Facebook. https://jttu.net/goode2014, 2014. [Online; accessed 15-03-2024]

Rachel Potvin

"The Google codebase includes approximately one billion files and has a history of approximately 35 million commits spanning Google's entire 18-year existence. The repository contains 86TBa of data, including approximately two billion lines of code in nine million unique source files."

— Rachel Potvin and Josh Levenberg. Why Google Stores Billions of Lines of Code in a Single Repository. *Communications of the ACM*, 59(7), 2016. doi:10.1145/2854146

Tomas Votruba

“Before monorepo, I had to upgrade every package manually, which resulted in dissonance: one package used Symfony\Console 3.2, but other only 2.8 and it got messy for no reason.”

— Tomas Votruba. How Monolithic Repository in Open Source Saved My Laziness. `https://jttu.net/votruba2017`, 2017. [Online; accessed 15-03-2024]

What About Yandex?

Сейчас большая часть исходного кода проектов Яндекса хранится в едином репозитории, либо находится в процессе переезда в него:

- над проектами трудятся более 2000 разработчиков.
- более 50 000 проектов и библиотек.
- размер репозитория превышает 25 Гб.
- в репозиторий уже совершено более 3 000 000 коммитов.

Source: Alexey Kruglov. Continuous Integration in Yandex. `https://jttu.net/kruglov2018yandex`, nov 2018. [Online; accessed 15-12-2024]

Benefits of “Manyrepo” Approach

Encapsulation Each repo encapsulates and hides its details from everybody else.

Fast Builds When a repo is small, the time its automated build takes is small.

Accurate Metrics Calculating LoC for a large repository doesn’t make any sense.

Homogeneous Tasks It’s easier to make tasks similar in size and complexity.

Single Coding Standard Smaller repositories look more beautiful.

Short Names Smaller namespaces mean better maintainability.

Simple Tests More dependencies are difficult to mock and test.

Source: Yegor Bugayenko. Monolithic Repos Are Evil. `https://www.yegor256.com/180905.html`, sep 2018. [Online; accessed 15-12-2024]

References

Yegor Bugayenko. Monolithic Repos Are Evil. https://www.yegor256.com/180905.html, sep 2018. [Online; accessed 15-12-2024].

Yegor Bugayenko. Volatility Metric to Detect Anomalies in Source Code Repositories. In *Proceedings of the 1st SIGPLAN International Workshop on Beyond Code: No Code*, pages 1–4, 2021. doi:10.1145/3486949.3486961.

Danilo Caivano, Pietro Cassieri, Simone Romano, and Giuseppe Scanniello. An Exploratory Study on Dead Methods in Open-Source Java Desktop Applications. In *Proceedings of the 15th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11, 2021. doi:10.1145/3475716.3475773.

Rain Durham Goode. Scaling Mercurial at Facebook. https://jttu.net/goode2014, 2014. [Online; accessed 15-03-2024].

Sebastian Eder, Maximilian Junker, Elmar Jurgens, Benedikt Hauptmann, Rudolf Vaas, and Karl-Heinz Prommer. How Much Does Unused Code Matter for Maintenance? In *Proceedings of the 34th International Conference on Software Engineering (ICSE)*, 2012. doi:10.1109/icse.2012.6227109.

Ciera Jaspan, Matthew Jorde, Andrea Knight, Caitlin Sadowski, Edward K. Smith, Collin Winter, and Emerson Murphy-Hill. Advantages and Disadvantages of a Monolithic Repository: A Case Study at Google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, pages 225–234, 2018. doi:10.1145/3183519.3183550.

Alexey Kruglov. Continuous Integration in Yandex. https://jttu.net/kruglov2018yandex, nov 2018. [Online; accessed 15-12-2024].

Ivano Malavolta, Kishan Nirghin, Gian Luca Scoccia, Simone Romano, Salvatore Lombardi, Giuseppe Scanniello, and Patricia Lago. JavaScript Dead Code Identification, Elimination, and Empirical Assessment. *IEEE Transactions on Software*

Engineering, 49(7):3692–3714, 2023. doi:10.1109/TSE.2023.3267848.

M. Mantyla, J. Vanhanen, and C. Lassenius. A Taxonomy and an Initial Empirical Study of Bad Smells in Code. In *Proceedings of the International Conference on Software Maintenance*, 2003. doi:10.1109/icsm.2003.1235447.

Rachel Potvin and Josh Levenberg. Why Google Stores Billions of Lines of Code in a Single Repository. *Communications of the ACM*, 59(7), 2016. doi:10.1145/2854146.

Simone Romano, Giuseppe Scanniello, Carlo Sartiani, and Michele Risi. A Graph-Based Approach to Detect Unreachable Methods in Java Software. In *Proceedings of the 31st Annual Symposium on Applied Computing*, pages 1538–1541, 2016. doi:10.1145/2851613.2851968.

Simone Romano, Christopher Vendome, Giuseppe Scanniello, and Denys Poshyvanyk. A Multi-Study Investigation into Dead Code. *IEEE Transactions on Software Engineering*, 46(1), 2018. doi:10.1109/tse.2018.2842781.

Simone Romano, Giovanni Toriello, Pietro Cassieri, Rita Francese, and Giuseppe Scanniello. A Folklore Confirmation on the Removal of Dead Code. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pages 333–338, 2024. doi:10.1145/3661167.3661188.

Will Shackleton, Katriel Cohn-Gordon, Peter C. Rigby, Rui Abreu, James Gill, Nachiappan Nagappan, Karim Nakad, Ioannis Papagiannis, Luke Petre, Giorgi Megreli, et al. Dead Code Removal at Meta: Automatically Deleting Millions of Lines of Code and Petabytes of Deprecated Data. In *Proceedings of the 31st Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1705–1715, 2023. doi:10.1145/3611643.3613871.

Tomas Votruba. How Monolithic Repository in Open Source Saved My Laziness. `https://jttu.net/votruba2017`, 2017. [Online; accessed 15-03-2024].

James A. Whittaker, Jason Arbon, and Jeff Carollo. *How Google Tests Software*. Addison-Wesley, 2012. doi:10.1145/1852666.1852675.