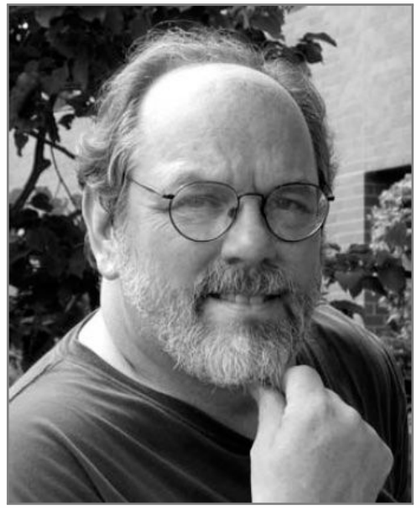
Tech Debt

Yegor Bugayenko

Lecture #14 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video

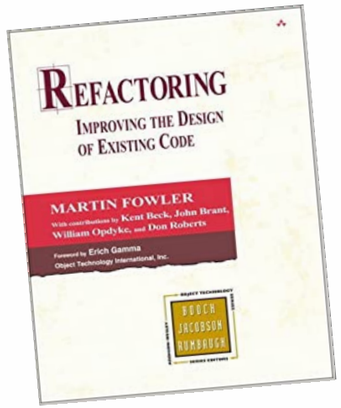


Ward Cunningham

"Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt."

— Ward Cunningham. Experience Report — The WyCash Portfolio Management System. In *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 29–30, 1992. doi:10.1145/157710.157715

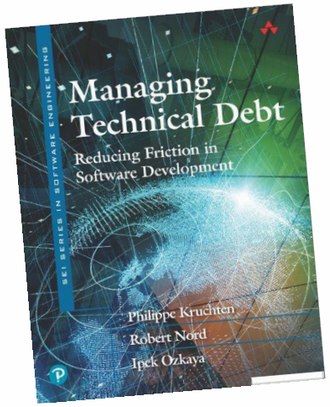
1. Technical debt may hurt if it's not managed earlier rather than never.



Martin Fowler

“You can bear some interest payments, but if the payments become too great, you will be overwhelmed. It is important to manage your debt, paying parts of it off by means of refactoring.”

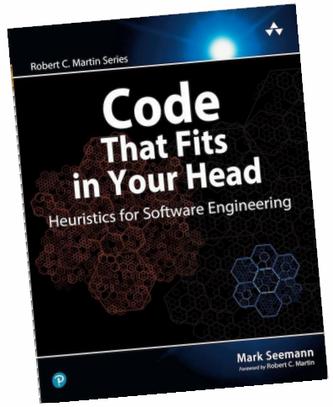
— Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999. doi:10.5555/311424

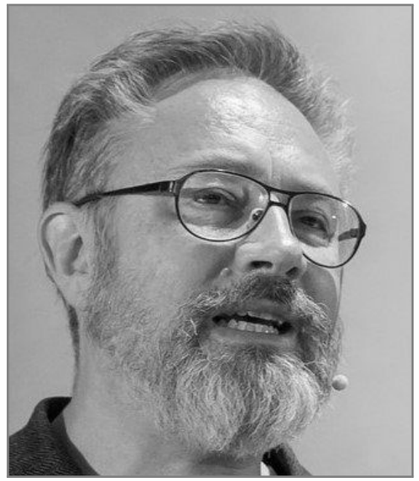


Philippe Kruchten

"By identifying concrete items of technical debt, considering their impact over time, evaluating the lifecycle costs associated with them, and introducing mechanisms for expressing technical debt and estimating its impact, an organization can help everyone better understand the pains of software evolution."

— Philippe Kruchten, Robert Nord, and Ipek Ozkaya. *Managing Technical Debt: Reducing Friction in Software Development.* Addison-Wesley, 2019. doi:10.5555/3364312

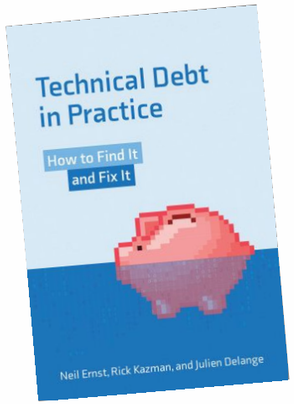


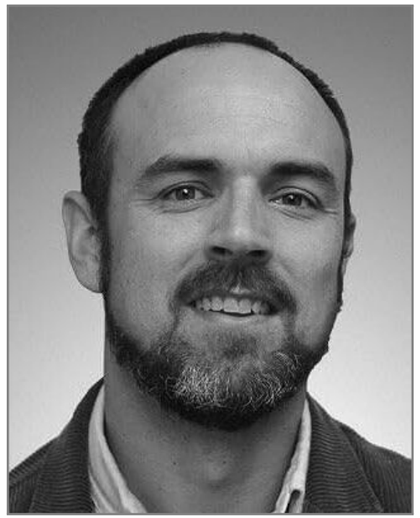


Mark Seemann

“I like the gardening metaphor’s emphasis on activities that combat disorder. Just as you must prune and weed a garden, you must refactor and pay off technical debt in your code bases.”

— Mark Seemann. Code That Fits in Your Head: Heuristics for Software Engineering, 2021





Neil Ernst

"Debt is not, by itself, the problem. The problem is not identifying and acknowledging the debt, not measuring the debt, and not paying the debt back."

— Neil Ernst, Rick Kazman, and Julien Delange. *Technical Debt in Practice: How to Find It and Fix It.* MIT Press, 2021. doi:10.7551/mitpress/12440.001.0001

2. Scientifically, it’s proven too.

Emad Shihab

"We find that developers with higher experience tend to introduce most of the self-admitted technical debt (SATD) and that time pressures and complexity of the code do not correlate with the amount of self-admitted technical debt."

— Aniket Potdar and Emad Shihab. An Exploratory Study on Self-Admitted Technical Debt. In *Proceedings of the International Conference on Software Maintenance and Evolution*, pages 91–100. IEEE, 2014. doi:10.1109/ICSME.2014.31

Gabriele Bavota

"57% of fixed instances shows a very long survivability, by staying in the system for over 1,000 commits, on average."

— Gabriele Bavota and Barbara Russo. A Large-Scale Empirical Study on Self-Admitted Technical Debt. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 315–326, 2016. doi:10.1145/2901739.2901742

Terese Besker

"Our results indicate that technical debt reduces developers' morale since it hinders them from performing their development tasks, making progress, and achieving their goals."

— Terese Besker, Hadi Ghanbari, Antonio Martini, and Jan Bosch. The Influence of Technical Debt on Software Developer Morale. *Journal of Systems and Software*, 167(1), 2020. doi:10.1016/j.jss.2020.110586

Paris Avgeriou

> “There is no commonly agreed on and validated set of rules and metrics for measuring technical debt. Instead, each tool uses its own set of rules and metrics without detailed explanation or motivation.”

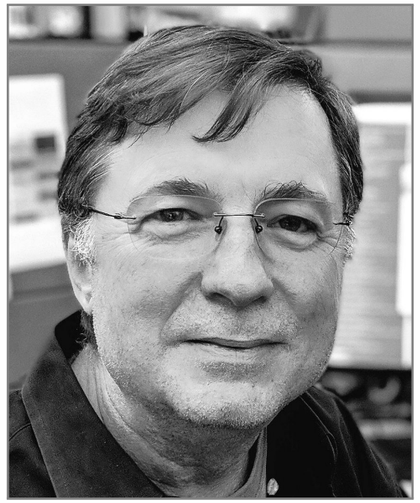
— Paris C. Avgeriou, Davide Taibi, Apostolos Ampatzoglou, Francesca Arcelli Fontana, Terese Besker, Alexander Chatzigeorgiou, Valentina Lenarduzzi, Antonio Martini, Athanasia Moschou, Ilaria Pigazzini, et al. An Overview and Comparison of Technical Debt Measurement Tools. *IEEE Software*, 38(3):61–71, 2020. doi:10.1109/MS.2020.3024958

Yutaro Kashiwa

"We found that changes involving SATD are 6–7% less likely to be accepted by reviews than those not involving SATD. About 28–48% of SATD comments are introduced during reviewing. We found that 20% of SATD comments are introduced due to reviewer request."

— Yutaro Kashiwa, Ryoma Nishikawa, Yasutaka Kamei, Masanari Kondo, Emad Shihab, Ryosuke Sato, and Naoyasu Ubayashi. An Empirical Study on Self-Admitted Technical Debt in Modern Code Review. *Information and Software Technology*, 146(1), 2022. doi:10.1016/j.infsof.2022.106855

3. Now, the Puzzle Driven Development!



Eric Allman

“Technical debt is inevitable. The issue is not eliminating debt, but rather managing it. When a project starts, the team almost never has a full grasp on the totality of the problem.”

— Eric Allman. Managing Technical Debt. *Communications of the ACM*, 55(5): 50–55, 2012. doi:10.1145/2160718.2160733

Puzzle Driven Development: Motivating Example

Commit #1:

```
int fibonacci(int n) {
  if (n <= 2) {
    return 1;
  }
  // @todo I don't know
  // what to do when "n"
  // is larger than "2".
  // Implement it and uncomment
  // the assertion below.
  return 0;
}
assert fibonacci(0) == 1;
assert fibonacci(2) == 1;
// assert fibonacci(9) == 34;
```

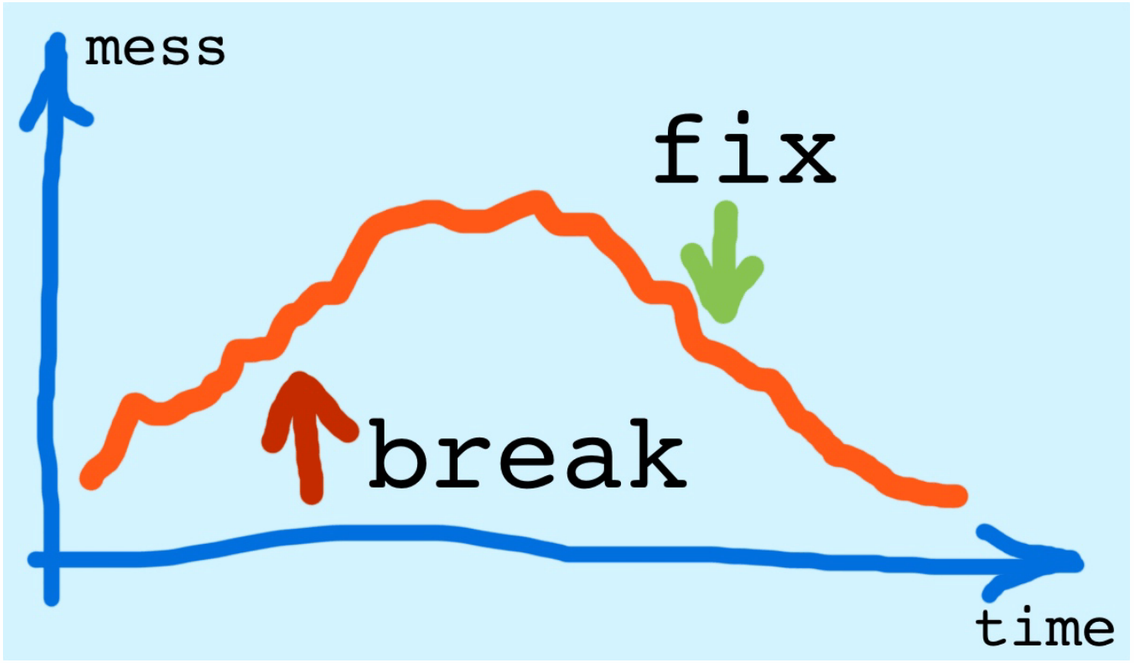
Commit #2:

```
int fibonacci(int n) {
  if (n <= 2) {
    return 1;
  }
  if (n == 9) {
    return 34;
  }
  // @todo Implement others
  // too, but I don't know
  // how to do it right.
  return 0;
}
assert fibonacci(2) == 1;
assert fibonacci(9) == 34;
// assert fibonacci(10) == 55;
```

Commit #3:

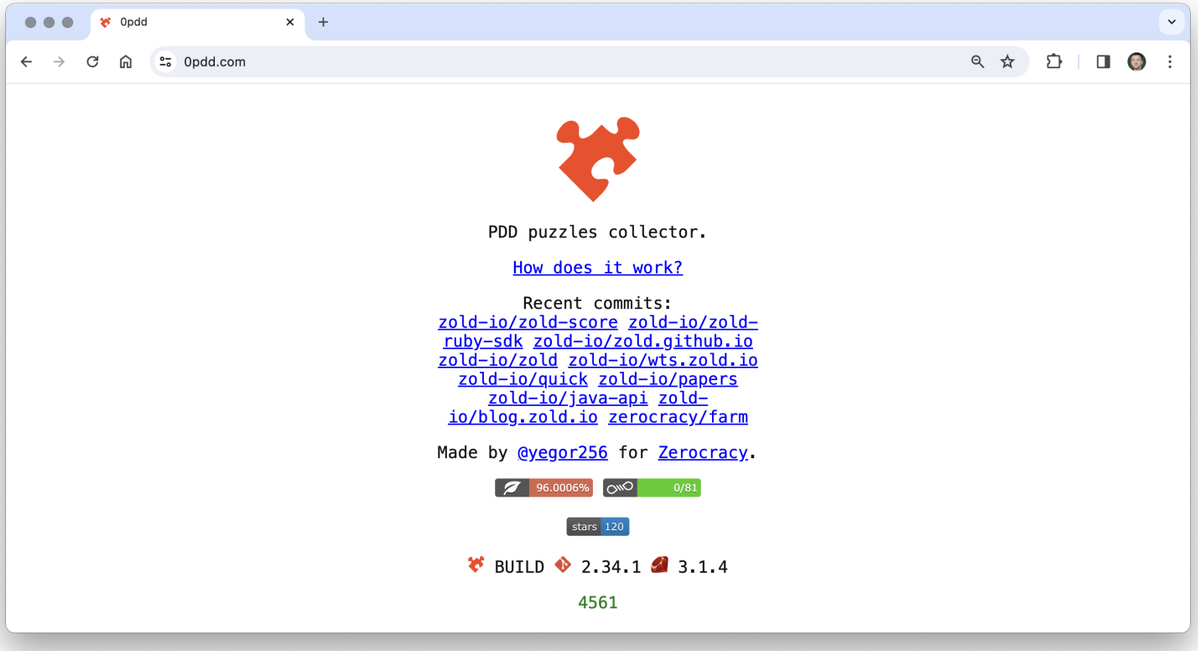
```
int fibonacci(int n) {
  if (n <= 2) {
    return 1;
  }
  return fibonacci(n-1)
    + fibonacci(n-2);
}
assert fibonacci(0) == 1;
assert fibonacci(2) == 1;
assert fibonacci(9) == 34;
assert fibonacci(10) == 55;
```

Break-and-Fix Cycle

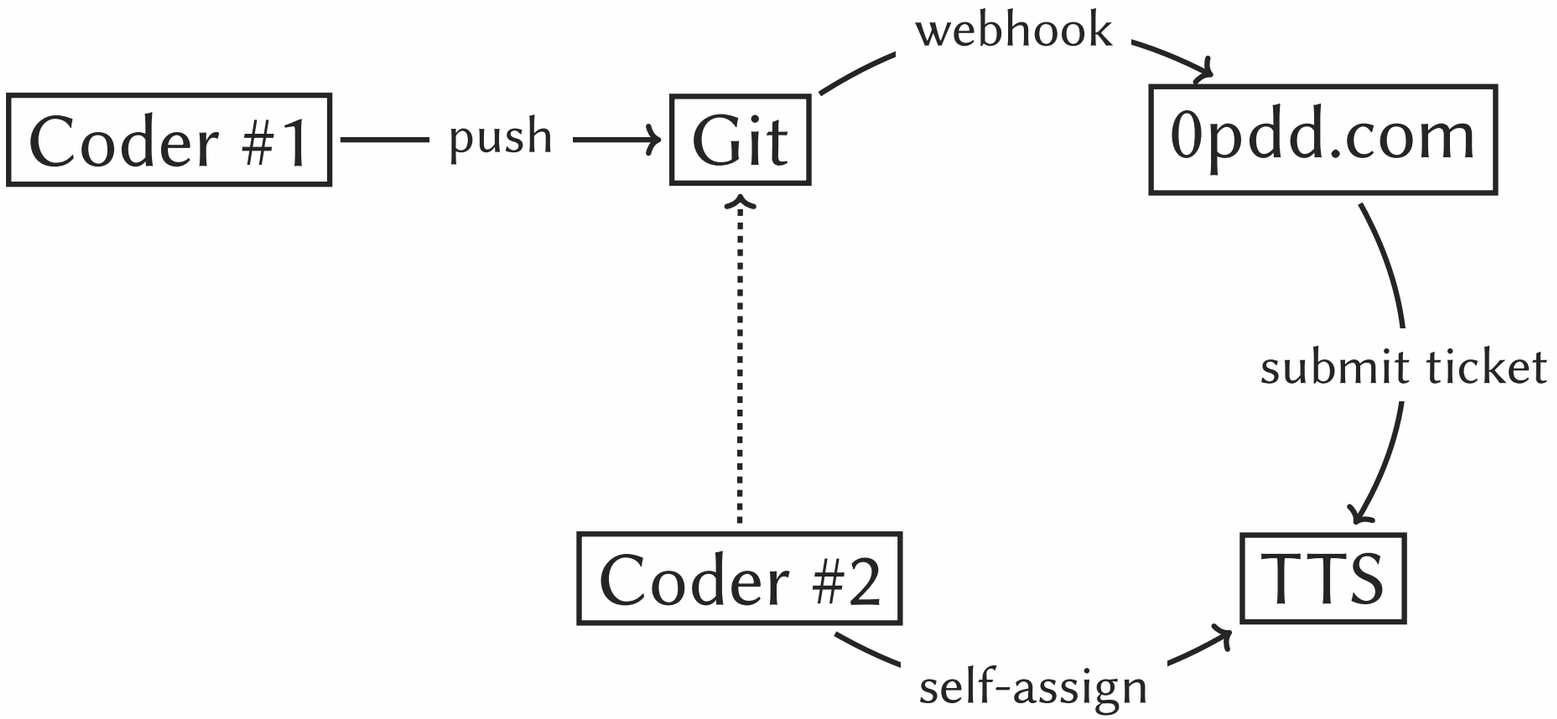


Source: Yegor Bugayenko. PDD by Roles. https://www.yegor256.com/140412.html, 4 2014. [Online; accessed 15-12-2024]

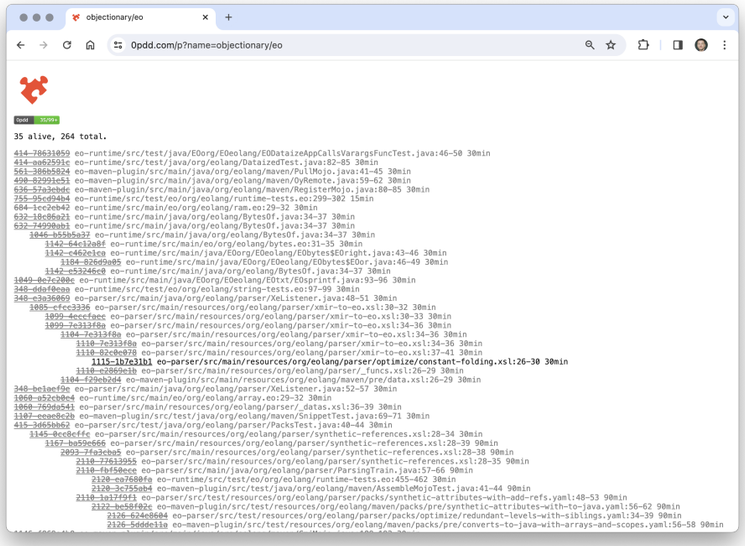
www.0pdd.com


0pdd
0pdd.com
PDD puzzles collector.
How does it work?
Recent commits:
zold-io/zold-score zold-io/zold-ruby-sdk zold-io/zold.github.io zold-io/zold zold-io/wts.zold.io zold-io/quick zold-io/papers zold-io/java-api zold-io/blog.zold.io zerocracy/farm
Made by @yegor256 for Zerocracy.
96.0006%
0/81
stars 120
BUILD 2.34.1 3.1.4
4561

PDD Pipeline

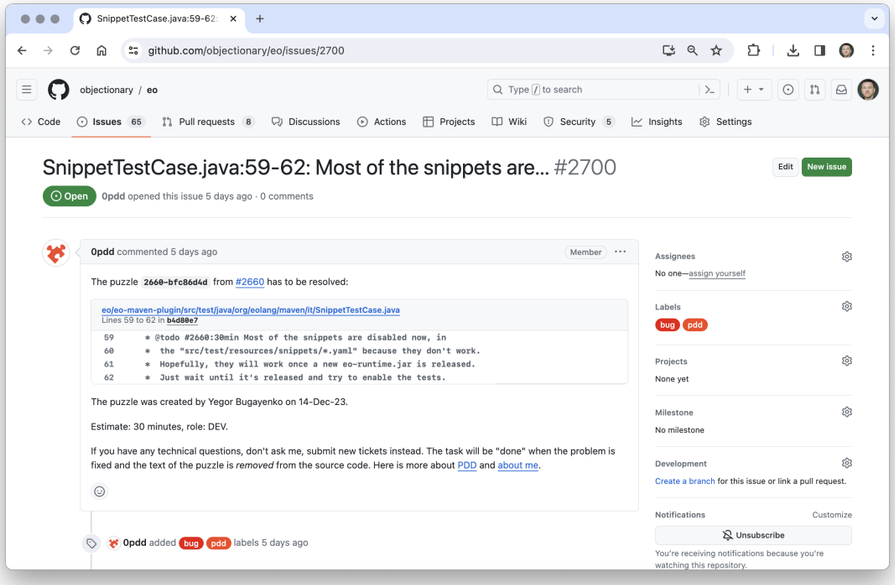


250+ Puzzles in objectionary/eo



Source: https://www.0pdd.com/p?name=objectionary/eo

Sample Ticket Submitted by 0pdd.com to objectionary/eo



Source: https://github.com/objectionary/eo/issues/2700

4. Next, we AI to help us manage long backlog with puzzles.

Giancarlo Succi

“This paper presents the benefits of considering the entire backlog when prioritizing tasks. We employ an iterative approach using Particle Swarm Optimization (PSO) to optimize a linear model with various preprocessing methods to determine the optimal model for task prioritization within a backlog.”

— Yegor Bugayenko, Mirko Farina, Artem Kruglov, Witold Pedrycz, Yaroslav Plaksin, and Giancarlo Succi. Automatically Prioritizing Tasks in Software Development. *IEEE Access*, 11(1), 2023. doi:10.1109/access.2023.3305249

Bibliography

Eric Allman. Managing Technical Debt. *Communications of the ACM*, 55(5):50–55, 2012. doi:10.1145/2160718.2160733.

Paris C. Avgeriou, Davide Taibi, Apostolos Ampatzoglou, Francesca Arcelli Fontana, Terese Besker, Alexander Chatzigeorgiou, Valentina Lenarduzzi, Antonio Martini, Athanasia Moschou, Ilaria Pigazzini, et al. An Overview and Comparison of Technical Debt Measurement Tools. *IEEE Software*, 38(3):61–71, 2020. doi:10.1109/MS.2020.3024958.

Gabriele Bavota and Barbara Russo. A Large-Scale Empirical Study on Self-Admitted Technical Debt. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 315–326, 2016. doi:10.1145/2901739.2901742.

Terese Besker, Hadi Ghanbari, Antonio Martini, and Jan Bosch. The Influence of Technical Debt on Software Developer Morale. *Journal of Systems and Software*, 167 (1), 2020. doi:10.1016/j.jss.2020.110586.

Yegor Bugayenko. PDD by Roles. `https://www.yegor256.com/140412.html`, 4 2014. [Online; accessed 15-12-2024].

Yegor Bugayenko, Mirko Farina, Artem Kruglov, Witold Pedrycz, Yaroslav Plaksin, and Giancarlo Succi. Automatically Prioritizing Tasks in Software Development. *IEEE Access*, 11(1), 2023. doi:10.1109/access.2023.3305249.

Ward Cunningham. Experience Report — The WyCash Portfolio Management System. In *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 29–30, 1992. doi:10.1145/157710.157715.

Neil Ernst, Rick Kazman, and Julien Delange. *Technical Debt in Practice: How to Find It and Fix It*. MIT Press, 2021. doi:10.7551/mitpress/12440.001.0001.

Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999. doi:10.5555/311424.

Yutaro Kashiwa, Ryoma Nishikawa, Yasutaka Kamei, Masanari Kondo, Emad Shihab, Ryosuke Sato, and Naoyasu Ubayashi. An Empirical Study on Self-Admitted Technical Debt in Modern Code Review. *Information and Software Technology*, 146(1), 2022. doi:10.1016/j.infsof.2022.106855.

Philippe Kruchten, Robert Nord, and Ipek Ozkaya. *Managing Technical Debt: Reducing Friction in Software Development*. Addison-Wesley, 2019. doi:10.5555/3364312.

Aniket Potdar and Emad Shihab. An Exploratory Study on Self-Admitted Technical Debt. In *Proceedings of the International Conference on Software Maintenance and Evolution*, pages 91–100. IEEE, 2014. doi:10.1109/ICSME.2014.31.

Mark Seemann. Code That Fits in Your Head: Heuristics for Software Engineering, 2021.