# Looking Back on Prediction: A Retrospective Evaluation of Bug-Prediction Techniques

Eirik Aune[1], Adrian Bachmann[2], Abraham Bernstein[2], Christian Bird[1], Premkumar Devanbu[1]

[1]University of California, Davis, USA
[2]University of Zurich, Switzerland
{*emaune,cabird,ptdevanbu*}*@ucdavis.edu* {*bachman,bernstein*}*@ifi.uzh.ch*

## Abstract

The heavy economic toll taken by poor software quality has sparked wide interest in bug prediction systems. If likely bug locality could be predicted early and well, targeted inspection, testing, etc. could be effectively deployed. Researchers in bug prediction systems have used historical data from open-source projects such as Eclipse. Typically, developers record how, when, where, and by whom bugs are fixed in code version histories (e.g., CVS) and bug tracking databases (e.g., Bugzilla). However, there is no standard or enforced practice to link a particular source code change to the corresponding entry in the bug database. Linkages are only provided sometimes and are not always indicated the same way. Consequently, inferring them is inexact and unreliable[1].

Many recent bug prediction efforts (e.g. [4, 6, 2, 5]) rely on corpora such as Zimmerman's [7, 8] dataset linking bugzilla bugs with source code commits in Eclipse. This corpus is created by establishing links between the repository information and the bug database information through inexact heuristics (e.g., the bug number occurs in the commit message). As a result, this corpus accounts for only about 10% of all the bugs in the database. Therefore, what we have here is a sample of bug occurrences, rather than the population of all known bug occurrences. Still, this corpus represents a store of valuable information concerning the location of bug fixes (and thus bug occurrences). Consequently, this type of data is at the core of much work on bug prediction. Typically one trains predictors on historically earlier data and then evaluates them on later data in
the commit message). As a result, this corpus accounts for only about 10% of all the bugs in the database. What we have here, therefore, is a *sample* of bug occurrences, rather than the population of all known bug occurrences. Still this corpus represents a store of valuable information concerning the location of bug fixes (and thus bug occurrences); therefore this type of data is at the core of much work on bug prediction. Typically one trains predictors on historically earlier data, and evaluates them on later data (both being drawn from the same sample).

This leads to our core research question: Is this sample of links biased in some way or does it accurately represent the entire population of bug occurrences? Moreover, if there is such bias how does it affect the performance of the bug prediction systems?

We studied 3 kinds of bias in the Eclipse corpus. First, are only certain types of bugs linked to source code? In fact, we have found the "most" severe bugs are the "least" likely to be linked. Second, we ask, are some bug fixers more likely to link bugs to source code? Indeed, we find that more experienced people are more likely to link bugs. Third, we ask does the way a bug is fixed matter?. We find a strong correlation between a change being linked and the bug being verified in the bugzilla database. Other possible sample biases are under study, We studied 3 kinds of bias in the Eclipse corpus. First, *are only certain types of bugs linked to source code?* In fact, we have found the most severe bugs are the least likely to be linked. Second, *are some bug fixers more likely to link bugs to source code?* Indeed, we find that more experienced people are more likely to link bugs. Third, *does the way a bug is fixed matter?*. We find that bugs that are verified after fixing are more likely to be linked. Other possible sample biases are under study, including *where* and *how* the bugs are fixed.

We have also looked at whether the existence of bias affects the performance of bug prediction system. Preliminary work, using the BugCache[3] bug prediction system, indicates that indeed, that the bias towards the least severe bugs indicates that a *proportionately much larger number of low-severity bugs are predicted*. This result raises a nagging, important question: does the sample bias affect the ability of bug prediction systems to direct quality-control efforts towards the most severe bugs? We are now evaluating other

prediction techniques.

## References

[1] M. Fischer, M. Pinzger, and H. Gall. Populating a release history database from version control and bug tracking systems. In *ICSM '03: Proceedings of the International Conference on Software Maintenance*, page 23, Washington, DC, USA, 2003. IEEE Computer Society.

[2] S. Kim, K. Pan, and J. E. E. James Whitehead. Memories of bug fixes. In *SIGSOFT '06/FSE-14: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 35–45, New York, NY, USA, 2006. ACM.

[3] S. Kim, T. Zimmermann, E. J. W. Jr., and A. Zeller. Predicting faults from cached history. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, pages 489–498, Washington, DC, USA, 2007. IEEE Computer Society.

[4] R. Moser, W. Pedrycz, and G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 181–190, New York, NY, USA, 2008. ACM.

[5] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller. Predicting vulnerable software components. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 529–540, New York, NY, USA, 2007. ACM.

[6] A. Schröter, T. Zimmermann, and A. Zeller. Predicting component failures at design time. In *Proceedings of the 5th International Symposium on Empirical Software Engineering*, pages 18–27, September 2006.

[7] J. Śliwerski, T. Zimmermann, and A. Zeller. When do changes induce fixes? In *MSR '05: Proceedings of the 2005 international workshop on Mining software repositories*, pages 1–5, New York, NY, USA, 2005. ACM.

[8] T. Zimmermann, R. Premraj, and A. Zeller. Predicting defects for eclipse. In *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, May 2007.

# Looking Back on Prediction

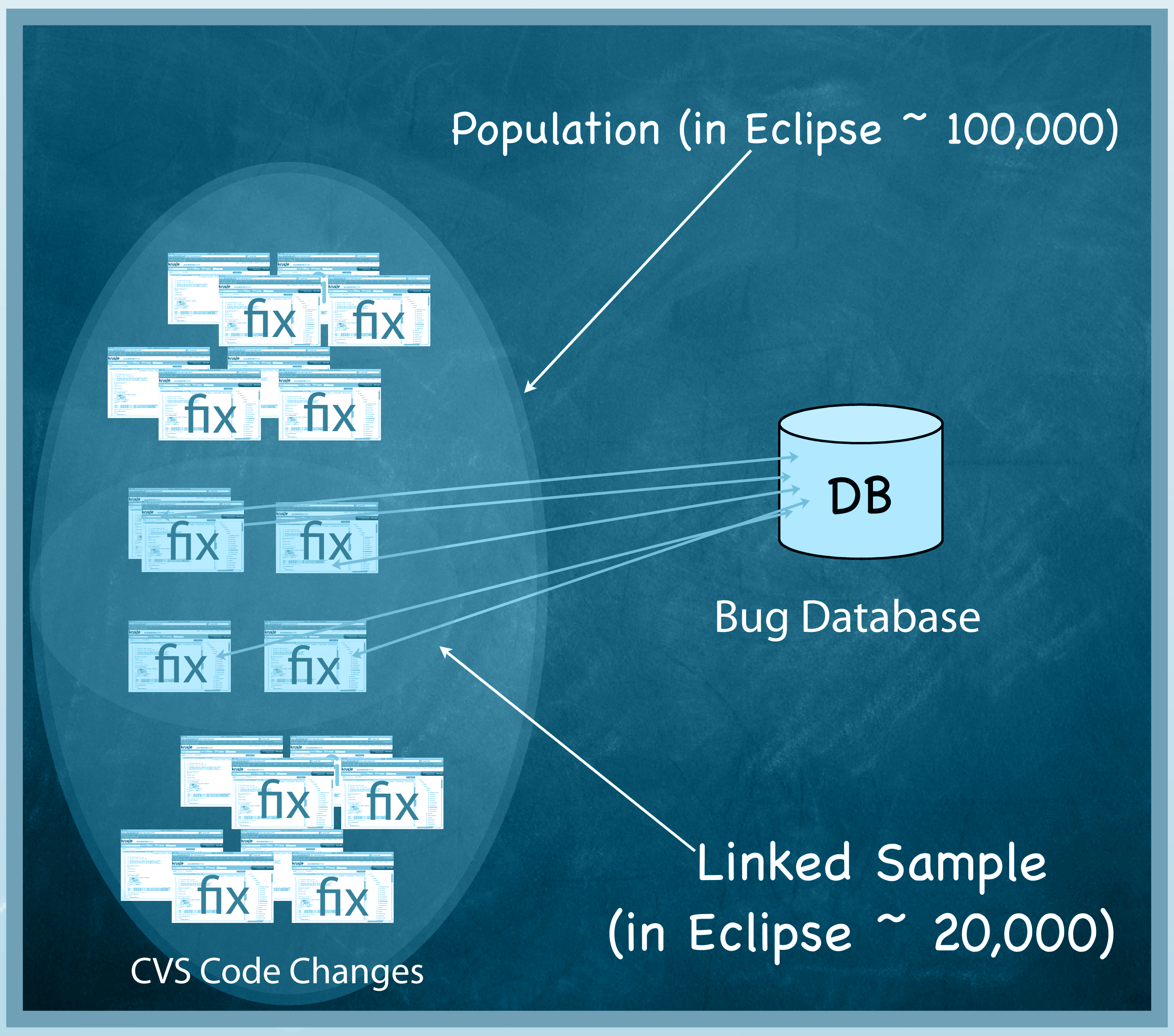
## A Retrospective Evaluation of Bug-Prediction Techniques

Eirik Aune[1], Adrian Bachmann[2], Abraham Bernstein[2], Christian Bird[1], Premkumar Devanbu[1]

[1]University of California, Davis, [2]University of Zurich, Switzerland

## Background

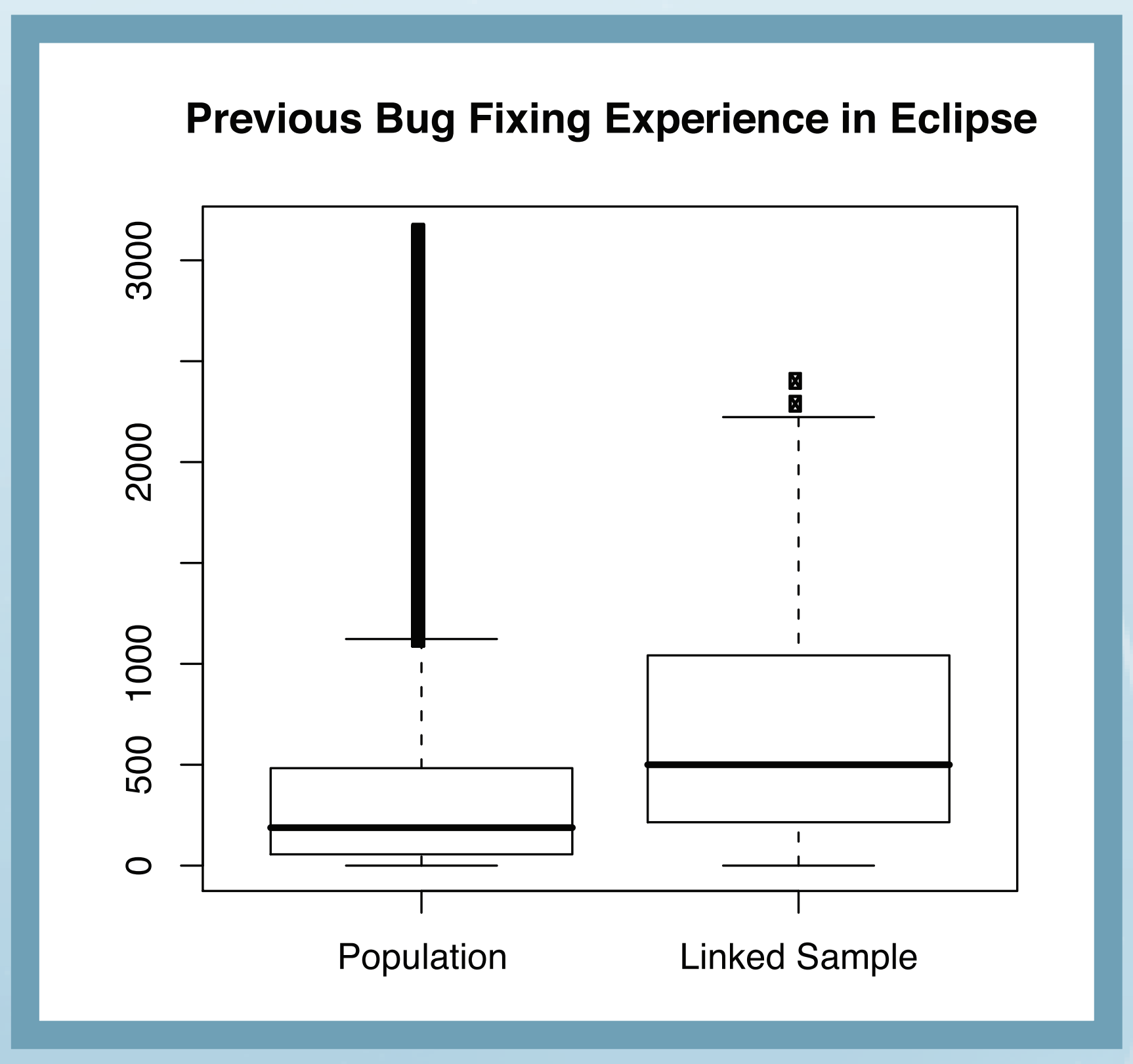
- Many Open Source Projects keep track of bugs with bug databases (Eg. BUGZILLA)
- Some commit logs mention bug numbers, thus **linking** bug fixes
- The sample of linked fixes has been used in bug prediction research
- **But...linked bugs may be a biased sample of the population!**

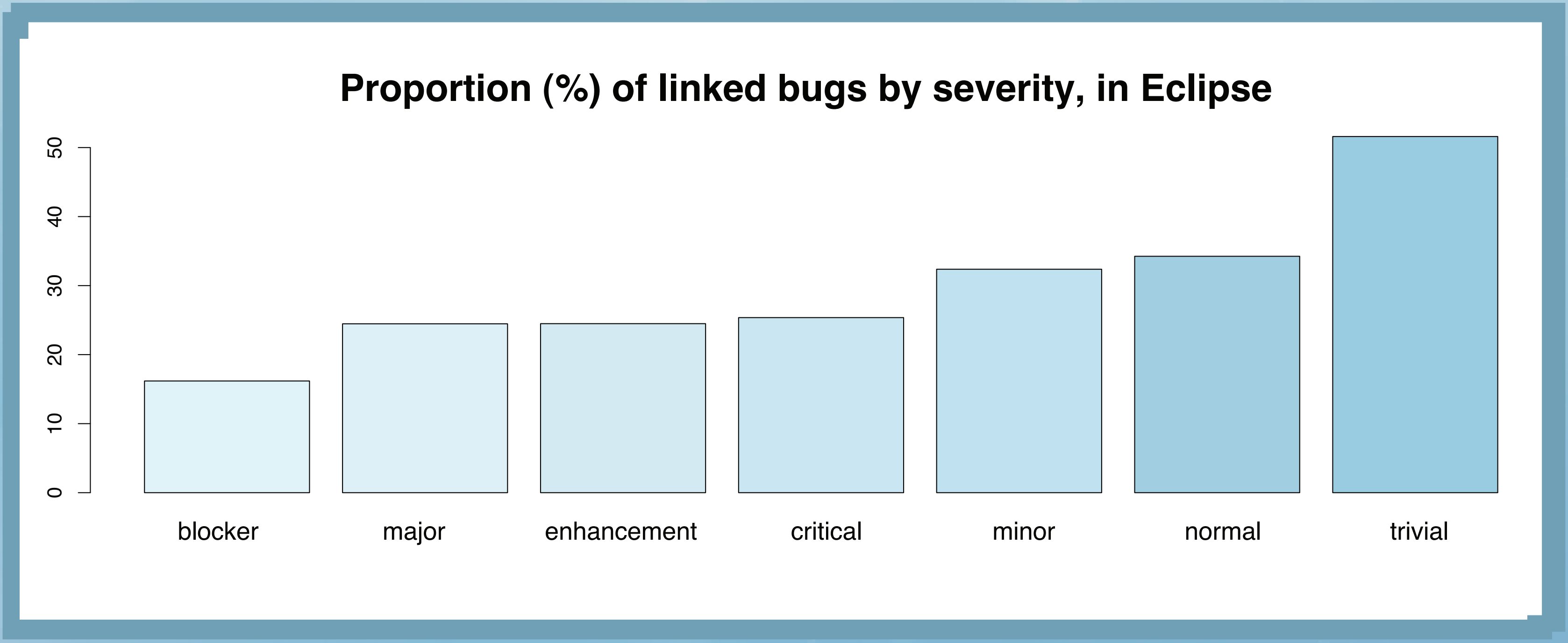


**Is this sample representative? How will bug detection systems be affected if we train on this sample?**

## Results & Discussion

- The sample is NOT representative
- Linked bugs are fixed by more experienced developers
- **Is this why more experienced developers seem to have more bugs?**





- More severe bugs are LESS likely to be linked!
- Experiments with Bug Cache prediction maintain severity discrepancy
- **Training on this sample may miss the most severe bugs**