



Faculty Publications

2009-01-01

Author Entropy vs. File Size in the GNOME Suite of Applications

Jason R. Casebolt
caseb106@gmail.com

Daniel P. Delorey
routey@deloreyfamily.org

Charles D. Knutson
knutson@cs.byu.edu

Jonathan Krein

Alexander C. MacLean

Follow this and additional works at: <https://scholarsarchive.byu.edu/facpub>



Part of the [Computer Sciences Commons](#)

Original Publication Citation

Jason Casebolt, Jonathan L. Krein, Alexander C. MacLean, Daniel P. Delorey, Charles D. Knutson. Challenge Paper: "Author Entropy vs. File Size in the GNOME Suite of Applications." Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories (MSR 9), Vancouver, British Columbia, Canada, May 23, 29.

BYU ScholarsArchive Citation

Casebolt, Jason R.; Delorey, Daniel P.; Knutson, Charles D.; Krein, Jonathan; and MacLean, Alexander C., "Author Entropy vs. File Size in the GNOME Suite of Applications" (2009). *Faculty Publications*. 862.
<https://scholarsarchive.byu.edu/facpub/862>

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.

Author Entropy vs. File Size in the GNOME Suite of Applications

Jason R. Casebolt, Jonathan L. Krein, Alexander C. MacLean, Charles D. Knutson

SEQuOIA Lab, Brigham Young University

caseb106@gmail.com, jonathankrein@byu.net, amaclean@byu.net, knutson@cs.byu.edu

Daniel P. Delorey

Google, Inc.

dandelorey@gmail.com

Abstract

We present the results of a study in which author entropy was used to characterize author contributions per file. Our analysis reveals three patterns: banding in the data, uneven distribution of data across bands, and file size dependent distributions within bands. Our results suggest that when two authors contribute to a file, large files are more likely to have a dominant author than smaller files.

1 Introduction

As software systems evolve and grow they become more complex [1]. One measure of system complexity is author entropy, which characterizes software authorship patterns. Author entropy is a summary statistic that quantifies the contributions of authors to files. Files with dominant authors have low entropy; files without dominant authors have high entropy.

In this paper we present our investigation of the relationship between author entropy and file size in the GNOME projects suite. Specifically, we focus on the two-author case. This study continues the research initiated by Taylor et al [3].

2 Methods

We selected projects from the GNOME suite of desktop applications in response to the 2009 MSR Data Mining Challenge. In this study we filtered and calculated metrics (as described in section 2.1) for all of the 576 projects in the GNOME suite, as of February 2009. We report results from a visual analysis of ten of these projects.

2.1 Producing the Data Sample for Visual Analysis

We manually selected 10 projects for visualization based on maturity and size. Mature projects are more likely to display long-term project patterns, and large projects provide more data for visualization.

Each project was filtered to exclude all non-source-code files. The filtering compared the extension of each file to 107 known source-code file extensions [2]. To focus our results on the two-author entropy case, we further filtered the projects to exclude all files composed by either a single author or by more than two authors.

After filtering, we computed two metrics for each file in each project: file size (measured in lines of code, LOC) and author entropy (described in section 2.2).

2.2 Author Entropy

We use *author entropy* to characterize the distribution of author contributions for each file. In this section we briefly describe author entropy. For a more detailed explanation, see [3].

Entropy is a measure of chaos or disorder in a system¹. The concept of entropy originated in thermodynamics but has been borrowed by information theory. For our purposes, we use author entropy as a measure of the distribution of author contributions within a file.

The general form for entropy, shown in Equation 1, states that for c authors each author's contribution to the total entropy is $p_i \cdot \log_2 p_i$, where p_i is the proportion of lines in the file written by author i .

$$E(S) = \sum_{i=1}^c (p_i \cdot \log_2 p_i) \quad (1)$$

¹Low entropy does not necessarily indicate software quality or "goodness."

The total entropy is directly proportional to the evenness of the contributions of the authors. If one author dominates, the entropy is low; if all authors contribute evenly, the entropy is maximized for that number of authors. Equations 2 and 3 give the general forms for calculating maximum and minimum entropy based on the number of authors, c . Figure 1 demonstrates entropy as a function of the p_i values for the two-author case.

$$E_{max} = \log_2(c) \quad (2)$$

$$E_{min} = 0 \quad (3)$$

We calculate author entropy at the granularity of a single line of code. Specifically, if an author changes any part of a line (based upon blame data from Subversion), the entire line is attributed to that author.

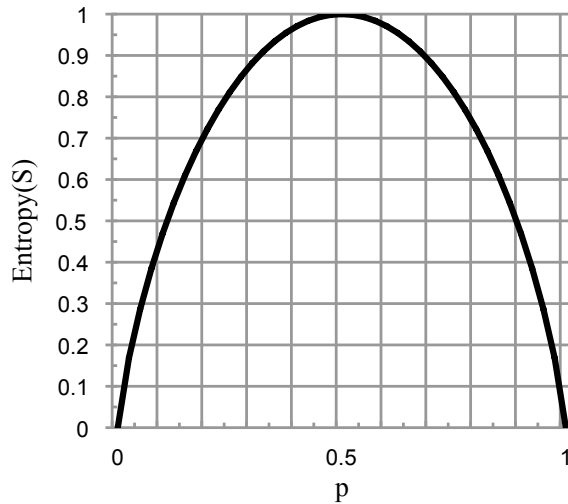


Figure 1. 2nd-Order Entropy Curve

2.3 Tools

In order to calculate author entropy for each file we used the original Python script implemented by Taylor et al. in their study, which introduced author entropy [3]. The Python script uses the Subversion *blame* command to extract author contribution information about each file. Further, since different source-code line counting utilities produce slightly different results (e.g., not all utilities count empty lines), we implemented a custom Java program to aggregate the output of each Python script execution, from which we obtained file size values identical to those used in the author entropy calculations. Our aggregate tool, therefore, consists of a Java program that executes the Python script on each

GNOME project, pipes the Python output into a routine that calculates the file sizes, and then links the file sizes to their corresponding author entropy scores.

In order to prevent excessive calls to the GNOME servers, we developed an automated Java process using *svn-sync* to clone each of the 576 projects (running the Subversion *blame* command on a checked-out copy of a project generates a call to the remote repository). Thus, we were able to test and perfect our processes, as well as run our calculations, without stressing the GNOME servers.

3 Analysis

Using the filtered results for only the two-author case we discovered several interesting patterns. We chose to focus on the two-author case because it is common and lays a foundation for studying the effects of small changes to files that were previously “owned” by a single author.

We constructed separate plots for each of the 10 projects selected for visualization. However, since the results were similar across the projects, we report the results of one representative project, *Evolution*.

3.1 Patterns

Figure 2 shows a graph of author entropy versus file size for almost all revisions of all files² that have exactly two authors in the GNOME project, *Evolution*. Three interesting patterns are immediately apparent: (1) banding in the data; (2) the distribution of data points across the bands; and (3) the distribution of data points within the bands.

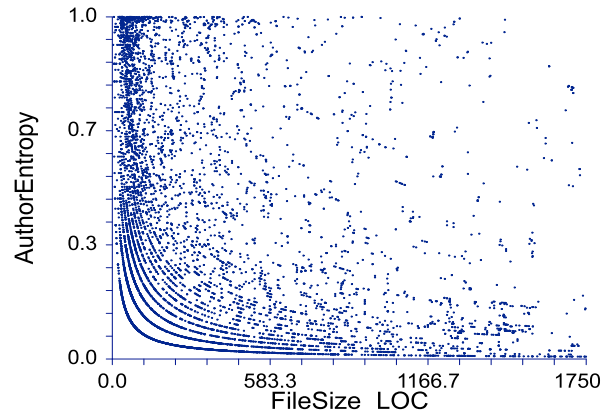


Figure 2. Author Entropy vs. File Size (LOC)

²Several outliers between 1,750 and 5,000 LOC have been excluded to provide a clearer view of the main body of data.

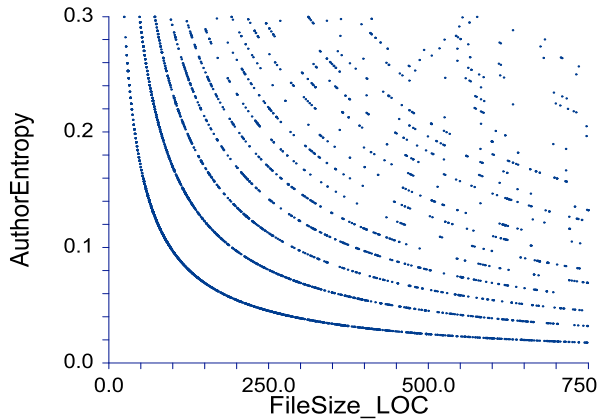


Figure 3. Author Entropy vs. File Size (LOC) (Zoomed)

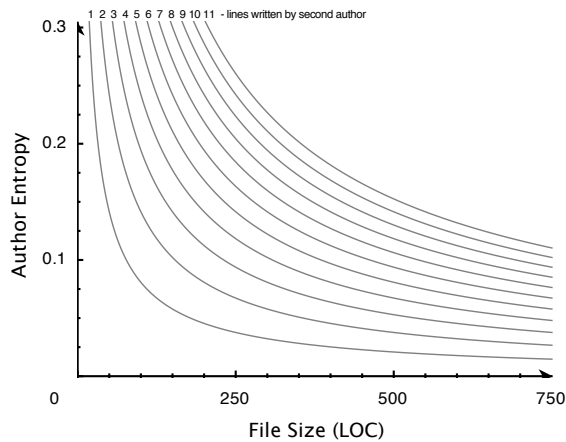


Figure 4. Author Entropy vs. File Size (Full Distribution)

3.1.1 Pattern 1: Data Bands

The author entropy vs. file size data points lie along distinct bands, which are clearly visible in Figure 3. The bands curve through the line $y=x$ and asymptotically approach limits that run parallel to both the x and y axes. The banding pattern is due to both the nature of the author entropy calculation and the file size metric. Specifically, the two metrics divide the data points into equivalence classes, one for each band on the graph.

Data points in the first equivalence class—the outer band closest to the axes—correspond to files in which all lines but one were written by the same author. Data points in the second equivalence class correspond to files in which one author wrote two lines and the co-author wrote the rest. Thus, the equivalence class to

which a data point belongs depends on the number of lines in the smaller of the two authors' contributions. If the smaller contribution is three, four, or five lines, then the file will belong to the third, fourth, or fifth equivalence class, respectively.

Figure 4 shows a graph of the first 11 equivalence classes. Lines are used to represent every possible file size in each equivalence class. Note that the minimum file size possible in the first equivalence class is 2, in the second, 3, in the third, 4, and so forth.

3.1.2 Pattern 2: Distribution of Data Points Across Bands

Data points are not evenly distributed across the bands. The majority of the data points concentrate in the lower order equivalence classes (outer bands), which correspond to the lowest possible entropy values for each file size. To understand the distribution differences, we manually inspected files that contribute data points to the outer bands. We found many changes associated with white space formatting and output message transformations, as well as multiple one line modifications to files by authors who had not previously edited the file.

Of the small, one-time modifications, two classes of changes are particularly interesting. In one case, a secondary author slightly changes a file to update the interface between the file being changed and another file to which that same author is the primary contributor. In the other case, the secondary author made small changes which are seemingly unrelated to any other file to which the author made a significant contribution. We hypothesize that the unrelated changes may be bug fixes which cause exceptional contribution patterns.

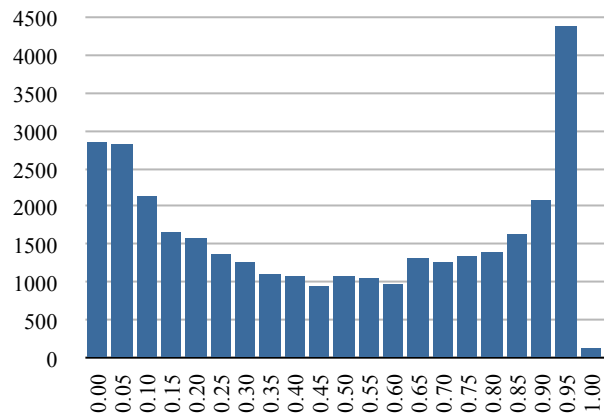


Figure 5. 2-Author Entropy Distribution by File Count for 28,955 Files from 33 SourceForge Projects

3.1.3 Pattern 3: Distribution of Data Points Within Bands

The data points within each band are not uniformly distributed. Figure 3 demonstrates that as file size increases, the data points cluster on the lower order bands (lower entropy), and as file size decreases, the data points increasingly tend to lie on the higher order bands (higher entropy).

Our previous work has shown that the distribution of author entropy values clusters around the upper and lower bounds and is sparse in the middle range (see Figure 5 taken from [3]). This study further separates the high and low entropy clusters to reveal the patterns for small and large file sizes independently.

For the projects we considered, our results also demonstrate that it is less common for two authors to contribute approximately equally to a file. It is much more likely for one author to have greater share in the per-line ownership as a file grows in size.

4 Conclusions

Our study of multiple GNOME projects reveals a roughly inverse relationship between author entropy and file size. This result suggests that when two authors contribute to a file, large files are more likely to have a dominant author than small files. Our investigation characterizes multiple causes of small author contributions. These contributions include white space formatting changes, output message changes, interface modifications, and possible bug fixes.

References

- [1] Victor R. Basili and Barry T. Perricone. Software errors and complexity: an empirical investigation. *Commun. ACM*, 27(1):42–52, 1984.
- [2] Daniel P. Delorey. Observational studies of software engineering using data from software repositories. Master’s thesis, Brigham Young University, April 2007.
- [3] Quinn C. Taylor, James E. Stevenson, Daniel P. Delorey, and Charles D. Knutson. Author entropy: A metric for characterization of software authorship patterns. In *Third International Workshop on Public Data about Software Development (WoPDaSD ’08)*, page 6, September 2008.