An Exploratory Study on Library Aging by Monitoring Client Usage in a Software Ecosystem

Conference Paper · February 2017

DOI: 10.1109/SANER.2017.7884643

5 authors, including:

Raula Gaikovina Kula
Nara Institute of Science and Technology
165 PUBLICATIONS **2,076** CITATIONS
SEE PROFILE

Takashi Ishio
Future University Hakodate
156 PUBLICATIONS **1,408** CITATIONS
SEE PROFILE

Ali Ouni
Osaka University
159 PUBLICATIONS **4,005** CITATIONS
SEE PROFILE

Katsuro Inoue
Nanzan University
385 PUBLICATIONS **6,968** CITATIONS
SEE PROFILE

An Exploratory Study on Library Aging by Monitoring Client Usage in a Software Ecosystem

Raula Gaikovina Kula*, Daniel M. German‡, Takashi Ishio*, Ali Ouni*† and Katsuro Inoue*

*Osaka University, Japan
{raula-k, ishio, ali, inoue}@ist.osaka-u.ac.jp

‡ University of Victoria, Canada
dmg@uvic.ca

†UAE University, UAE
ouniaali@gmail.com

***Abstract*—In recent times, use of third-party libraries has become prevalent practice in contemporary software development. Much like other code components, unmaintained libraries are a cause for concern, especially when it risks code degradation over time. Therefore, awareness of when a library should be updated is important. With the emergence of large libraries hosting repositories such as Maven Central, we can leverage the dynamics of these ecosystems to understand and estimate when a library is due for an update. In this paper, based on the concepts of software aging, we empirically explore library usage as a means to describe its age. The study covers about 1,500 libraries belonging to the Maven software ecosystem. Results show that library usage changes are not random, with 81.7% of the popular libraries fitting typical polynomial models. Further analysis show that ecosystem factors such as emerging rivals has an effect on aging characteristics. Our preliminary findings demonstrate that awareness of library aging and its characteristics is a promising step towards aiding client systems in the maintenance of their libraries.**

I. Introduction

In software development, a third-party library is a reusable software component that provides a certain functionality for client systems. With promises of quality implementations, libraries offer an effective and efficient means to software development [11]. Reuse of these popular libraries have lead to the rise of large-scale library hosting sites, thus creating an ecosystem between software that are either library providers (systems) or library users (clients). For instance, in 2010, Sonatype reported that the Maven Central [3], one of the largest online ecosystem of Java OSS libraries contained over 260,000 maven libraries and served over 70 million downloads every week [2]. By November 2016, this ecosystem had multiplied almost five times over, with more than 1,680,822 maven libraries available for client usage[1]. Other notable library ecosystems include the NPM [4] JavaScript package ecosystem for node.js, RubyGems [5] for Ruby and CPAN [1] for Perl packages.

Software reliability is widely regarded as the prediction of whether a software system functions as expected without failure. There are theories of system failure that are related to aging components. Parna's software aging theory [15] states that as time passes, aged components are faced with a higher likelihood to fail. Similarly, Eick's code decay principle [10] states that unless an ongoing process of maintenance is in place, changes in external environments creates an:

> *'unanimous feeling among developers of the software that code degrades through time and maintenance becomes increasingly difficult and expensive'*

Mitigation of code decay, which prolongs the lifespan of a software is regarded as code rejuvenation.

In this paper, inspired by reliability concepts of software aging, we model library aging by monitoring client-usage of a software library within its software ecosystem. Specifically, we represent library aging characteristics such as library *decay* or *rejuvenation*. Since library age depicts ecosystem changes, it may prove useful in guiding client developers on whether they should update, especially if a library is about to reach its end–of–life.

In an exploratory study of over 4,500 client projects that adopt and use over 1,500 different java library versions, we first investigate whether this phenomena can be modeled as a mathematical function. Analyzing library usage data collected from clients, we identify typical polynomial mathematical curve models (i.e., First-order model, Second-order model, or Higher-order model) that 'best fits' library aging. Results show that 93.87% of the tested library versions can fitted to any of the models. Furthermore, we found that 81.7% fitted the Higher-order model. To investigate usefulness of the curves, we took a more qualitative analysis of Higher-order model, by manual examination of the curve shape and aging characteristics of two case studies. Interestingly, we found empirical evidence that rivals do have an effect on the aging of popular libraries.

II. Background & Definitions

Software ecosystems are defined as *'a collection of software systems, which are developed and co-evolve in the same environment'* [13]. In the context of library ecosystems, this consists of a plethora of library units that share and compete for clients that share a commonality, such as the same technological platform. For instance, the Maven Central [3] ecosystem host a multitude of library packages that can be integrated into any java client system.

In this study, we are interested in the relationship between a client system and its dependent libraries. Therefore, we use the population count of library dependents (i.e., library usage) for a certain library, to determine its age. We propose the

[1] statistics taken from https://search.maven.org/#stats on Nov, 2016

978-1-5090-5501-2/17/$31.00 © 2017 IEEE

SANER 2017, Klagenfurt, Austria
Early Research Achievements

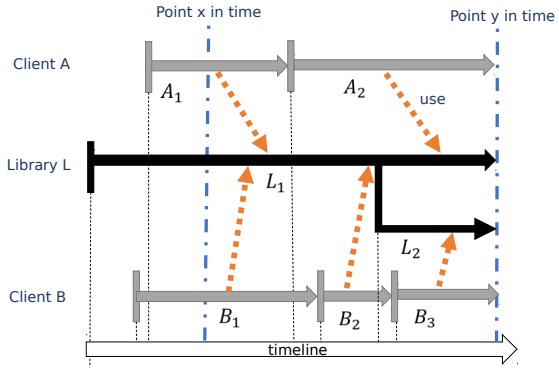


Fig. 1. Library migration between library and systems. The orange arrow represents a dependency between Systems A , B and Library L.

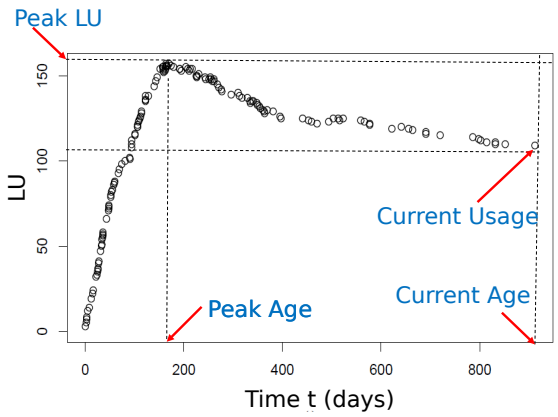


Fig. 2. Modeling Library Aging as a mathematical curve equation.

following aging process: (1) younger attractive libraries will experience a growing number of clients that adopt the library. Later, (2) the library reaches maturity with a maximum peak of clients. Once the library reaches its peak and cannot attract any more new users or clients abandoning the library, (3) it falls into library decay. Finally, (4) a library is said to reach its end of life when its not attracting any more clients or no more clients remain. Since aging is temporal, from the universe of known client systems we can determine the usage of a library, which we denote as *Library Usage (LU): is a population count of client systems that use a certain library at a specific point in time.*

Figure 1 shows how LU changes through that adoption and migration of library versions by clients in the ecosystem. Suppose we represent L_v as a library and C_v as a client system. In this example, we have a library L with two versions (L_1 and L_2) and two clients: A (A_1 and A_2) and B (B_1, B_2 and B_3). Library migration is shown at point x in time, where the LU of L_1 is two (system A and B); however at point y, since B_3 adopted L_2, the LU of L_1 is one (client A) while the LU of L_2 is now one (client B).

Library migration is sometimes caused when a better alternative to the current library emerges in the ecosystem. In this context, we define these competing libraries as *Library Rival: is a competing library that causes library migration from other libraries*. The most common rival to any library version is the newly released version, especially if it fixes bugs or includes needed features that the previous library version lacks. In Figure 1, we observe that evidence that L_2 is indeed the rival of L_1, thus causing future versions of client B to use it instead of L_1.

III. Library Aging based on LU

In this paper, we propose that LU can be interpreted the aging of a library over time. In Figure 2, we show that this relationship between LU and time can be plotted visually as a curve. We see that for the shown GUAVA$_{14.0.1}$ library, this relationship is represented with as a mathematical equation or curve. From Figure 2, a curve may exhibit the following distinguishable feature characteristics:

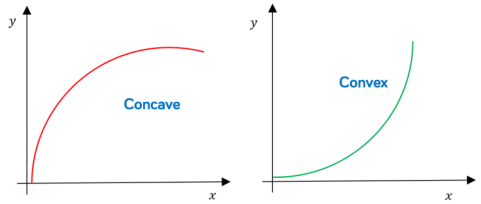


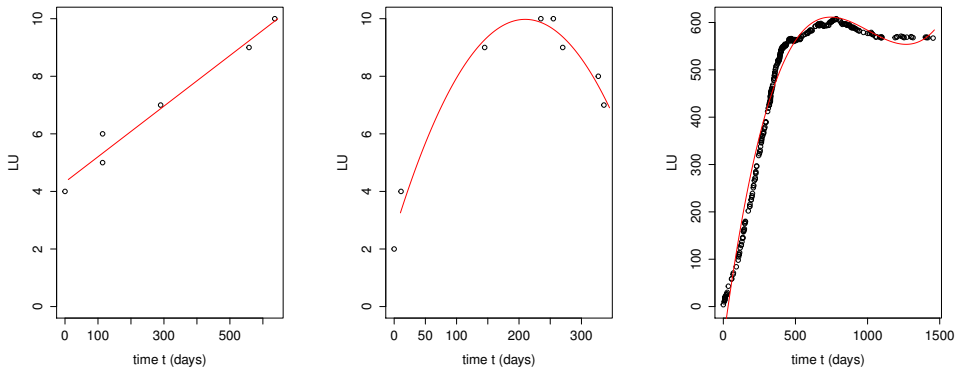
Fig. 3. Shapes of the curves showing (i) a concave or (ii) a convex aging curve

- **Current Age** - is measured by the time from first usage by a client till its most recent last seen LU by a system. (i.e., library age = 912 days).
- **Current LU** - is the LU count at the oldest age. (i.e., current LU = 102 at day 912).
- **Peak LU** - is the point in which the maximum LU has been reached. (i.e., peak usage =157 LU).
- **Peak Age** - is the number of days point in which the Peak LU has been reached. (i.e., peak age was reached in 164 days).

As shown in Figure 3, the LU can either increase (i.e., rejuvenation) or decrease (i.e., decay) to form a upward or downward curve, which is either concaving or convexing. Hence, we define these curve shapes:

- **Library Decay** - starts once the LU peak has been reached. Visually, decay which is evident by the decreasing rate of LU over time with a concave aging curve. (i.e., Figure 3 (i))
- **Library Rejuvenation** - is the time period in which the rate of LU is steady or increases over time with a convex aging curve. (i.e., Figure 3 (ii))

Mathematical models are commonplace in many fields of biology, medicine, economics and the social sciences [8]. They describe and predict the estimation of a quantitative measure over time. In software engineering, different models have been applied primarily in the context of software reliability (e.g., [7], [16]). In this study, we use fundamental polynomial equations are shown in Figure 4 and Table I for our curve fitting. Thus, for the relationship between library usage LU



(a) First-order model (b) Second-order model (c) Higher-order model

Fig. 4. Library Aging for three libraries representing the proposed three models. In each plot, a dot is one system that uses it; and the full line is the model that fits the curve best. Note that Figure 4(a) represents the First-order model (FINDBUGS-ANNOTATIONS$_{2.0.1}$), Figure 4(b) the Second-order model (CGLIB$_{3.0}$)and Figure 4(c) illustrates the best-fits for Higher-order model (JUNIT$_{4.10}$).

TABLE I
SUMMARY STATISTICS OF THE COLLECTED DATASET

Model	Equation
First-order model	$LU = at + b$
Second-order model	$LU = at^2 + b$
Higher-order model	$LU = at^y + b + c$

and time t, key characteristics of each model are described below:

- **First-order model**. The first degree polynomial equation is depicted in Figure 4(a) as having single linear line and mathematically where variables a and b are constants with no convex or concave curve. The linear relationship typically indicates that LU for a library is steady increasing over time, with no signs of slowing down.
- **Second-order model**. The second degree polynomial equation is depicted in Figure 4(b) as having a bell-shaped curve and mathematically where variables a and b are constants. This characteristic is typical of libraries dropped immediately, due to an exposed vulnerability.
- **Higher-order model** . The third and higher degree polynomial equation is depicted in Figure 4(c) as having multiple inflections points in its curve and is expressed mathematically where variables a, b and c are constants and $y = [3, 4]$. Higher-order model have multiple inflection points of decay or rejuvenation in the aging curve.

IV. EMPIRICAL EVALUATION

A. *(RQ1)* Do popular libraries share common aging characteristics of their usage? If so, what are some of these distinctive characteristics?

1) Research Method: Our research method comprises of three steps. In the first step, we need to collect empirical data that represents library usage for a particular software ecosystem. Table II shows a summary of collected Open Source

TABLE II
SUMMARY STATISTICS OF THE COLLECTED DATASET

	Dataset statistics
projects creation dates	2004-Oct to 2009-Jan
projects last update	2015-Jan onwards
# unique systems (projects)	48,495 (4,659)
total size of projects	630 GB
# commits related to `pom.xml`	4,892,770
# library migrations	852,322
# library versions curves	9,197

TABLE III
BEST FITTING RESULTS FOR THE 1,503 POPULAR LIBRARY VERSIONS. INDECISIVE IS WHEN ALL THREE MODELS SCORE THE SAME AIC.

Model	# Fitted	% of Studied Libraries
First-order model	40	2.66%
Second-order model	143	9.51%
Higher-order model	1228	81.70%
Indecisive	63	4.19%
No Fit	29	1.93%
Total	1,503	100%

Source (OSS) client projects used libraries from the Maven ecosystem. To ensure validity and quality of the collected data, we (i) selected projects showed activity (commits to the project) from January 2015 onwards and (ii) removed projects that had less than 1,000 commits to its codebase. Similar to our prior work [12], we used our `PomWalker`[2]) tool and `jgit`[3] to extract a history of library dependency changes. As a result, we were left with 852,322 dependency facts between the system and a maven library. Additionally, we set a minimum threshold of 8 LU with a lifespan of at least more than a day. For the second step, we then run experiments by which the library aging data is fitted against the proposed models. We employ the widely-used Akaike

[2]https://github.com/raux/PomWalker
[3]https://eclipse.org/jgit/

TABLE IV
SUMMARY STATISTICS OF LIBRARY AGING CHARACTERISTICS FOR THE 1,474 FITTED LIBRARY VERSIONS. HIGHLIGHTED VALUES SHOWS THAT THE CHARACTERISTICS WERE DEEMED STATISTICALLY SIGNIFICANT (TUKEY HSD TEST).

	Fitting Model	Min.	1st Qu.	$\bar{x} = median$	$\mu = mean$	3rd Qu.	Max.
Current Age (# days)	First-order model	1.00	149.50	756.00	838.00	1,361.00	3,106.00
	Second-order model	1.00	239.00	520.00	712.90	983.00	2,812.00
	Higher-order model	1.00	181.00	500.50	753.80	1,138.00	3745.00
Peak Age (# days)	First-order model	8.00	487.50	980.00	1003.00	1482.00	3151.00
	Second-order model	15.00	487.00	837.00	1052.00	1482.00	3053.00
	Higher-order model	8.00	544.00	967.50	1112.00	1528.00	3745.00
Peak LU (LU)	First-order model	8.00	8.00	10.00	10.78	11.25	20.00
	Second-order model	8.00	10.00	12.00	14.85	17.00	63.00
	Higher-order model	8.00	10.00	14.00	27.69	24.00	779.00
Current LU (LU)	First-order model	1.00	8.00	9.50	9.68	11.00	20.00
	Second-order model	1.00	8.00	10.00	13.13	15.00	63.00
	Higher-order model	1.00	8.00	12.00	25.01	22.00	746.00

TABLE V
TOP 10 HIGHER-ORDER MODEL LIBRARIES BY PEAK LU.

ID	Library	Version	Peak LU (LU)	Current LU (LU)	Peak Age (#days)	Current Age (#days)	# Rivals (succeeding releases)
L1	junit	4.8.2	373	340	959	1809	8 (ver.4.9∼4.12)
L2	commons-collections	3.2.1	268	266	2508	2570	2 (ver.3.2.2 & ver.4.0)

Information Criterion (AIC) [6] to evaluate the goodness of fit of each model. Specifically, the model with the lowest AIC score is deemed the best fit. This criterion provides us with a measure of the relative quality of the models that accounts for the trade-off between their goodness of fit and their complexity (see [14] for a thorough treatise of model fitting). For the curve-fitting, we use the R[4] statistical environment to produce each library model. We rely on R's (`grofit`)[5] package, which implements the AIC score, to fit against First-order model, Second-order model and Higher-order model. Finally, in the third step, we examine our proposed feature characteristics. Specifically, we compare the *Current Age, Peak Age, Peak LU* and *Current LU* characteristics of the different models. To this end, we use the Tukey HSD [14] test to find significance differences $(p > 0.05)$ between the libraries that fitted each model.

2) Findings: Table III lists the percentages of library aging that are best-fit by each model. The most important finding is that 93.87% of the 1,503 eligible library versions are best-fitted by one of the models (i.e.,1.93% of libraries did not fit any model, while 4.19% as indecisive). A practical example of a no fit was libraries that had a constant LU, thus reflecting a straight line. The result shows that library aging is not random, but can be modeled. Furthermore, looking at the Table III, we observe that 81.7% of the library versions fitted the Higher-order model. Based on the nature of Higher-order model, we can assume that most library versions show multiple inflection points of either decay or rejuvenation during its lifespan.

Table IV shows the statistics of library characteristics that are specific to the fitted library versions. As highlighted in the Table, using the Tukey HSD test, we found significant differences for Peak LU (*p=0.008*) and Current LU (*p=0.005*) between the Second-order model and Higher-order model. Hence, the results indicates that most rejuvenated and popular libraries (i.e.,indicated by high Peak and Current LU) fit the Higher-order model. Importantly, it shows that many libraries do not fit the Second-order model, not immediately dropped. Hence, we answer RQ1:

> We found 81.7% of the popular libraries (i.e., with high Peak LU and Peak Age) best fitted the Higher-order model.

B. *(RQ2)* What is the effect of ecosystem factors such as library rivals to popular library aging characteristics?

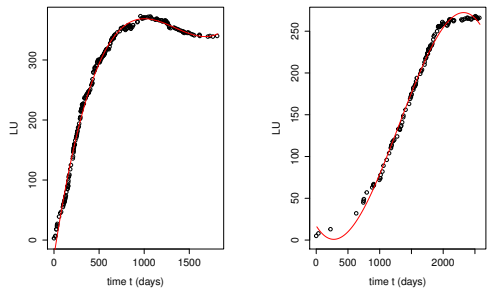
Results from RQ1 indicate that library versions tend to fit the Higher-order model, especially towards the more popular libraries. Hence our motivation for RQ2, is to take a more qualitative approach to manually examine the visual aspects of the library curve with the effect of ecosystem factors such as rivals.

1) Research Method: In order to address RQ2, we randomly select two popular libraries that best fitted the Higher-order model in RQ1. Our research method has two steps to understand the usefulness of our curve characteristics. In the first step, we visually examine the library aging curves to draw some interesting observations. Such observations include identifying the Peak LU and determining if the shape of the curve. In the second step, we identify rival library versions.

2) Findings: Figure 5 and Table V shows the two selected libraries and their rivals. In the analysis we make two main observations. First, in Figure 5(a), we observe that L1 is still a library that has reached Peak LU. Visually, we identify that the

[4]http://cran.r-project.org/

[5]https://cran.r-project.org/web/packages/grofit/grofit.pdf. grofit relies on glm https://stat.ethz.ch/R-manual/R-devel/library/stats/html/glm.html to fit a function



(a) L1 (JUNIT$_{4.8.2}$) (b) L2 (COMMONS-COLLECTIONS$_{3.2.1}$)

Fig. 5. Library Aging for the selected three popular libraries. In each plot, a dot is one system that uses it; and the full line (red) is the fitted Higher-order model.

library has a concave aging curve. This characteristic suggests that since the library has reached Peak LU, it is now in library decay. Manually inspecting the release dates of the rivals, we find that the cause of the decay may be attributed to a rival rival $junit_{4.9}$, which was released at the same time (i.e., 1000 days (2.7 years)). However, although the library is decaying, there is no significant reason for library migration. In fact, since reaching its Peak LU, LU has only dropped to 53. We conjecture that due to the nature of `junit` as a testing library, developers may not consider upgrading to rival libraries as a priority. However, we can recommend that new systems should consider other rival libraries.

Second, Figure 5(b), we observe L2 as having a concave aging curve. The curve suggests that L2 took a longer time to reach its Peak LU and the absence of rivals during this period. Currently, its Peak Age is at 2,500 (i.e., 6 years). Coincidently, at this same time, its rival of the next version ($commons-collections_{4.0}$) was released, possibly explaining the Peak LU. It is interesting to note that also the next rival ($commons-collections_{3.2.2}$) was later released in 2015, to accommodate existing users of the older version 3 systems users. This is because $commons-collections_{4.0}$ is a major upgrade to the new JDK from 4 to 5. In this case study, we also recommend new users to consider other rival libraries as this library has reached its peak, but due to the long convex age curve, we can see that it was a very popular and reliable library, with users hardly moving away from this library to other rivals in the ecosystem. Hence, we answer RQ2:

> Emergence or absence of rival libraries in the ecosystem have an effect on library decay or rejuvenation, especially causing a library to reach its Peak LU.

V. Challenges and Future Work

In this paper, we modeling library usage as aging curves, using them as a means to guide client developers on whether a library version should be either adopted or migrated away. In this study, we found that that 81.7% of 1,503 popular libraries fit a curve model. Furthermore, using two case studies, we manually were able to explain the curve characteristics to ecosystem factors such as library rivals.

Although results show promise, many challenges exist. In the study, we only considered rivals as the superseding library versions. There may be unrelated rival libraries in the ecosystem that will age a library. As stated by Bavota et al. [9], other global factors such as security vulnerabilities or a change in the environment platform may affect the aging of a library. We consider this as future work. Currently, the curves are only historic monitoring of client usage reporting the current state. As the mathematical functions have predictive properties, future work includes further investigation of prediction about the potential lifespan a library. Also, the current experiment is only performed within the Maven JVM ecosystem. It would be interesting to extend the research to different ecosystems such as the JavaScript npm, RubyGems and CRAN R ecosystems to investigate this phenomena.

VI. Acknowledgments

This work is supported by JSPS KANENHI (Grant Numbers JP25220003 and JP26280021) and the "Osaka University Program for Promoting International Joint Research."

References

[1] Comprehensive perl archive network,accessed 2016-07-20. http://www.cpan.org/.
[2] Maven central statistics,accessed 2016-07-20. http://blog.sonatype.com/2010/12/now-available-central-download-statistics-for-oss-projects/#.Vmgw77_2O7g.
[3] The maven central,accessed 2016-07-20. http://search.maven.org/.
[4] Npm package manager for javascript,accessed 2016-07-20. https://www.npmjs.com/.
[5] Rubygems community gems,accessed 2016-07-20. https://www.rubygems.org/.
[6] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *2nd Intrnl Symp. on Info. Theo.*, pages 267–281, Budapest, 1973.
[7] V. Almering, M. van Genuchten, G. Cloudt, and P. Sonnemans. Using software reliability growth models in practice. *Software, IEEE*, 24(6):82–88, Nov 2007.
[8] G. Annadurai, S. Rajesh Babu, and V. R. Srinivasamoorthy. Development of mathematical models (logistic, gompertz and richards models) describing the growth pattern of pseudomonas putida (nicm 2174). *Bioprocess Engineering*, 23(6):607–612, 2000.
[9] G. Bavota, G. Canfora, M. Di Penta, R. Oliveto, and S. Panichella. How the apache community upgrades dependencies: An evolutionary study. *Emp. Softw. Eng.*, 20(5):1275–1317, Oct. 2015.
[10] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus. Does code decay? assessing the evidence from change management data. *IEEE Transactions on Software Engineering*, 27(1):1–12, Jan 2001.
[11] W. Frakes and K. Kang. Software reuse research: status and future. *Softw. Eng., IEEE Trans. on*, 31(7):529–536, July 2005.
[12] R. G. Kula, D. M. German, T. Ishio, and K. Inoue. Trusting a library: A study of the latency to adopt the latest maven release. In *22nd IEEE Intrnl Conf. on Soft. Ana., Evol., and Reeng., SANER 2015, Montreal, Canada, March 2-6, 2015*, 2015.
[13] M. Lungu. Towards reverse engineering software ecosystems. In *ICSM*, pages 428–431. IEEE Computer Society, 2008.
[14] H. Motulsky and A. Christopoulos. *Fitting models to biological data using linear and nonlinear regression: a practical guide to curve fitting*. Oxford University Press, 2004.
[15] D. L. Parnas. Software aging. In *Software Engineering, 1994. Proceedings. ICSE-16., 16th Intnl. Conf. on*, pages 279–287, May 1994.
[16] S. Yamada, M. Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. *Reliability, IEEE Trans.*, R-32(5):475–484, Dec 1983.