The 𝔘nit Testing Harness: Achieving Source Code Street Cred

Nels E. Beckman

***Abstract*— In any large software organization, the street cred of source code is of principle concern. Often-times we find that a given code base can talk a mean game, but when time comes to "throw down," that code base is nowhere to be seen. While intuitively we as software developers come to gain a sense of which code bases are trust-worthy (or, "will ball 'till they fall"), some systematic method for measuring and reporting this is necessary. In this paper we present the 𝔘nit testing harness (pronounced, "Gee Unit"), which does just that.**

***Index Terms*— Software Engineering, Credibility Improvement, Street Life, Hennessy, Testing, Java**

I. INTRODUCTION

Slingin' code is a ride-or-die way of life, where every day it's just another homicide. It is important to stay TRU to tha game, or you risk being played for a chump. But the question is, how does one go about evaluating (and eventually increasing) the TRU-ness of the code that they sling? This is especially important at large software organizations where code often becomes soft from spending too much time in cushy source-code repositories.

In this paper we present the 𝔘nit Testing harness, a tool created explicitly for the purposes of evaluating the street cred of a code base. 𝔘nit is the fruit of a rare cross-discplinary collaboration. Researchers from Carnegie Mellon University's Institute for Software Research have joined together with the Interscope Records' Gorilla Unit, lead by world-renown scholar Dr. Curtis James Jackson III, who goes by the pseudonym 50-Cent.

While 𝔘nit currently only works on Java code bases (an inherrently 'soft' programming language, when compared with other, 'harder' languages such a C and z80 assembly [5]) we believe that our underlying street cred evaluation methodology can easily be extended to other languages.

This paper proceeds as follows. In Section II, we discuss our underlying methodology. How does one take any piece of source code and evaluate its worth in the inner-city? In Section III, we describe implementation and usage of the 𝔘nit tool. Most helpful to the end-user is Section III-A, which describes the classifications into which we place evaluated code. Section IV discusses the evaluation of 𝔘nit and ultimately, and mercifully, Section VI concludes.

II. METHODOLOGY

The 𝔘nit testing harness allows Java developers to build streed cred tests, and then automatically run and evaluate their results at the touch of a button. Our methods are based on earlier work by Clifford Smith, AKA the Method Man [6]. These methods are robust to programmer error and scale to large code bases as well as large quantities of Cristal. Unfortunately, these methods are proprietary and cannot be described in full technical detail.

N. Beckman is an average to below-average PhD student in the Institute for Software Research, *the* most thugged-out department in the School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 nbeckman@cs.cmu.edu

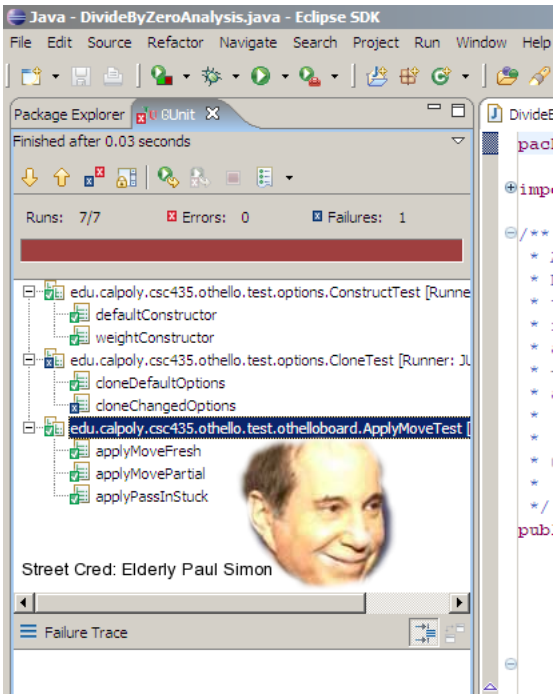

Fig. 1. Screenshot of 𝔘nit: This source code has a street cred rating of "Elderly Paul Simon."

III. THE 𝔘nit TOOL

The 𝔘nit street cred testing harness has been implemented as an Eclipse plug-in [1]. Eclipse is gradually replacing Emacs as the text-editor-that-acts-like-an-operating-system of choice for modern software developers.

A. Street Cred Strata

In order to better convey the the end-user programmer the actual level of street cred of the code base under test, we have developed the following scientific classification scheme. It includes ten discreet levels, listed here in order from furthest to nearest to the street. We also briefly describe each these gradations.

1) **Clay Aiken**—At the lowest end of the street cred spectrum is the rating of "Clay Aiken." Clay's Wikipedia

Fig. 2. Hpnotiq is a 35 proof French fruit liqueur made from vodka, cognac, and tropical fruit juices [7]. Sounds good to me!

entry describes him as "the most successful second-place finisher" in the history of American Idol. If that's not a dubious destinction for your software, then what is?

2) **CMU Graduate Student**—Being a graduate student at Carnegie Mellon University doesn't get you very far; it won't get you into the club, and it won't get your court-side seats. Sometimes the truth hurts.
3) **Elderly Paul Simon**—Most thuggish software doesn't live to see 30. Elderly Paul Simon shows us why this is a good thing. Current best-practices state that one should stay young, fly, and flashy 'till the day that one dies.
4) **P-Diddy**—This is a good indication that your source code was relatively well respected during the past, possibly because it played some pivotal role in the career of a more successful product. But when the more successful product was killed in a drive-by shooting in L.A., everyone kind of realized that your product was more about marketing than quality. Then your software did a cover of a Led Zepplin song for the Godzilla remake, which only made things worse…
5) **Young Paul Simon**—Have you ever see that movie *Almost Famous*? He *always* had big pupils…
6) **Hpnotiq**—Just like the liquor of the same name, your source code is most frequently seen at the clubs, in the VIP section. If you seem him out, remember: Don't start no stuff, won't be no stuff.
7) **Chry$ler 300, with Navigation**—Seriously, that car is dope. It's kinda like the cars that the bad guys drove on Batman: The Animated Series.
8) **Ja Rule**—If you feel the need to scream "Holla, Holla!" when your code base receives this rating, that is a perfectly acceptable reaction. If only it weren't for that memory leak, the software equivalent of a J-Lo collaboration, you'd have more respect.
9) **Dead Prez**—While your source code is not, strictly, as popular as some of the other big names on this list, it is authentic and has the respect of its peers, due to a prolonged life spend on the south side of Chicago, and its refusal to sell out. The authors themselves said it best in their seminal work [2]: *That's why I'm in the dojo, not just for the video. Really though, we really got beef with the popo. Never know when they gonna put you in a choke hold.*
10) **Hova**—The Michaelangelo of flow, your source code paints pictures with poems. When source code achieves a street cred rating of "Hova," this indicates and extremely high degree of credibility. At the same time, this rating indicates that a given piece of source code has achieved a rare feat, that of becoming commercially successful while still maintaining its reputation of being true to its roots. Much like Jay-Z, the J-Hova of hip hop, a code base achieving this rating has flow that is religious.

IV. Results of Tool Deployment

We attempted to try out our tool on a user base of approximately 500 Java developers. However, due to the fact that we were constantly screaming, "Wu Tang!" and, "Don't bring those weak generics into my house!" most of the developers dropped out. Therefore our current results are inconclusive.

V. Related Work

A. CMMI

Currently, the most well-known metric of street credibility in existence is the Stuntin' Engineers' Institute's Crunkability/Make Money Index (CMMI) [3]. CMMI has long been the de-facto standard hype metric for military and government software contractors. As its name suggests, the CMMI ranks the performance of a software development organization in two specific area: Its "crunkability" (its ability to have fun, drink, and spit game at the ladies) and its ability to "Make Money" (get paid by any means necessary). Based on these qualifications, a software organization is given a rank from one, being the fakest, to five, being the realist. The primary difficulty with the CMMI system is the large amount of overhead necessary in having an organization certified. While this is appropriate for large government contracters where the crunk-ness of the entire nation is at stake, for smaller, more agile software organizations, CMMI is more of a burden than anything else. JUnit, on the other hand, is a small and lightweight evaluation framework. So small, in fact, that when the cops ever search you or your vehicle, the chances of them busting you for it are pretty low. Of course they'll probably just plant a copy of JUnit on you anyway…

VI. Conclusion

Maintaining the street cred of a source code base has always been a high priority for software developers. However, up until this point there has been no framework for

automatically creating, running, and evaluating this particular metric. Therefore in order to fill this gap we have developed the GUnit testing harness. Source code is currently available online [4].

VII. ACKNOWLEDGMENTS

The author would like to thank you, dear reader, for making it this far. The metaphors never quite worked, but he kept with it the whole time, and that's got to count for something. The author would also like to thank the SIGBOVIK program commitee for graciously extending the submission deadline. If he had actually taken advantage of that extra time, this paper would no doubt be much funnier.

REFERENCES

[1] Website for the Eclipse SDK. `http://www.eclipse.org/`

[2] stic.man, M-1. Revolutionary but Gangsta. *Journal of the American Underground.* March 30, 2004. pp. 1045-1102.

[3] S. Radamenton, A. Robertson, J. J. Walker. *CMMI: Guidelines for Crunkability Improvement.* Addison-Wesley, 2003.

[4] GUnit Testing Harness Website. `http://www.g-unitsoldier.com/`

[5] W. C. Roeslisberger. z80 Assembly: Fad or In it to Win it? *4th Intl. Working Conference on 10x Programmers: Habits, Moods and Whims Track.* February 13, 1989.

[6] C. Smith. M. J. Blige. All That I Need. *Journal of the American Underground.* February 1, 1996.

[7] Wikipedia entry, "Hpnotiq." `http://en.wikipedia.org/wiki/Hpnotiq`