Can Some Programming Languages Be Considered Harmful?

Sabine Janssens
StressLabo
Loonbeek, Belgium
sabine@stresslabo.be

Ulrik Pagh Schultz
University of Southern Denmark
Odense, Denmark
ups@mmmi.sdu.dk

Vadim Zaytsev
Raincode Labs
Brussels, Belgium
vadim@grammarware.net

Abstract

Programming languages are not uniformly perceived by the software engineering community: some are respected, others are hated, some have a devoted following, others are viewed as a necessary evil. However, is it true that using some languages on a regular basis or even being exposed to them to a significant extent, can harm coders' minds? Can a programming language be blamed for limiting a software engineer's ability to form proper abstractions, communicate them to other members of the programming social unit, choose correctly among tools, techniques and constructs within the solution spaces, and altogether for boosting or ruining an engineer's career? If we knew, we could ensure language designers use methods that are least harmful when making new libraries, protocols and domain-specific languages. In this paper we raise a number of questions concerning language safety and design sociotechnical experiments to answer them.

CCS Concepts • **Human-centered computing** → *Computer supported cooperative work*; • **Software and its engineering** → **Software development techniques**; *Programming teams*; • **Social and professional topics** → *Computing literacy*;

Keywords programming languages, mental state, mental harm, teaching programming

1 Introduction

In robotics and similar domains, people often focus on the concept of *safety*, defined as avoiding software systems doing **harm to humans** [33]. Most of the time the harm is understood as physical harm: robots crushing human bodies, falling on people, electrocuting, etc. Such safety considerations are a big part of research and development in many areas such as smart houses [53] or agriculture [34].

However, harm can also be mental: confusing humans, deceiving them, manipulating cannot and should not be considered appropriate behaviour for a robot, a program or any cyberphysical system. Some concerns are already being voiced, such as the Turing test (i.e., the goal of artificial intelligence defined as teaching computers to impersonate humans) being considered harmful for the future of artificial intelligence [30] and should be reframed and rephrased as amplification and complementation of human intelligence [8].

We port the concept of safety to the domain of software languages (mainly programming languages [3, 46, 77] and domain-specific languages [51, 66, 76]). Specifically, we would like to know, **can software languages cause *mental harm* in some form to their users**, and if yes, what factors influence it and how they can be avoided? By mental harm we will mean "more than minor or temporary impairment of mental faculties [such as] the inflicting of strong fear or terror, intimidation or threat" [32, §110], including disorders, disabilities and other lasting effects on one's mental health beyond momentary fright or minor anxiety.

This paper presents a *research agenda*, formulates several research questions and sketches how experiments to answer them, will be set up. Studies on psychological and anthropological aspects of programming are known to be challenging, some even argue whether programming is too complex to be studied in an experimental context altogether [75].

Technically, safety for robots can be implemented by adding a graceful degradation scheme to account for partial failures [1], which unfortunately does not for software languages, since having someone monitor programmers' work and intervening whenever necessary, is highly unusual. We do not seriously count the role of the Scrum Master in some agile software development methodologies, whose work is to actively seek out and eliminate any impediments [68], since it is technically infeasible most of the time for a Scrum Master to change the software language in the middle of the project even if the entire team develops a mental illness.

2 Research Questions

The programming language community has already thoroughly studied questions like "what makes programs bad?", beyond the obvious demands of functionality, adaptability and efficiency [75], covering subtle symptoms of possible present of future problems, called "smells" [24]. Smells in code are known to annoy programmers [52, 79] and make programs worse [24, 54, 78]. Smell detection projects are usually limited to one language or one comparison between two

PLATEAU'17 Workshop on Evaluation and Usability of Programming Languages and Tools, October 23, 2017, Vancouver, CA

chosen languages. There are studies on Swift [39], Java [19], JavaScript [21], Python [72], PHP [26]; extension languages like Java annotations [61], Excel formulae [31] or CSS stylesheets [48, 56]; natural languages used to describe software [29] or IDE usage patterns [13]. Yet, we formulate our first research question more sharply, since this particular issue has not been studied before:

RQ0: does using a particular software language make programmers write bad programs?

(The question has the number zero since in the following sections we do not plan to answer it: the experimental setup would be rather straightforward and can be purely technical and not involve any inter-disciplinary challenges. This puts it outside the scope of our project).

Next, it is important for us to establish that mental health and even mental state of a programmer is related to the quality of code the programmer produces. If there is no link, then striving to avoid causing mental harm by improper language design is a purely ethical question, otherwise it is also a financial one.

RQ1: what are noticeable differences between the code written by programmers in different mental states (e.g., a happy programmer and a sad programmer)?

How an experiment can be set up to answer RQ1, will be shown in § 3.1. Even if the null hypothesis dictates otherwise, our expectation is that the programmer's state of mind indeed influences the quality of code (s)he is producing, since reading through bad code reinforces sadness and depression [5].

Next, we would like to be able to link languages with the mental state of programmers in those languages. It would be natural to assume that the more a programming language looks like a natural language, the stronger are the effects it is having on the mental state of its users that we can expect. Besides that, we know that "the folk language is related to symptom formation" [20], and the idioms people use to express themselves, are known to have direct consequence on their mental state. Hence, the same is expected for software languages, since each of those proposes special "idioms" to represent commonly encountered situations (branching, parallel execution, composite data structures, user authentication, visual layout, etc).

RQ2: is working in a particular language capable of making a programmer less happy or even depressed?

It has been postulated since the very early stages of computer science that "the tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities" [16]. Nowadays there are some that go as far as to claim that all programming is inherently depressing [81]. There is neither general understanding nor estimation of how much exposure to a language is needed for the effects to be noticeable and observable, which is an obvious threat to be accounted for in the experimental setup, since most modern programming involves several languages.

PHP is one of the languages commonly targeted by the programming community as the bad language of bad programmers who write bad code—dozens of testimonies as well as constructive accusations can be found on Quora [38, 67]. The hatred and frustration is so deep that there are entire websites, updated regularly over the course of many years, primarily dedicated to explaining "things in PHP which make me sad" [74]. Admittedly, the situation causes some programmers to compare themselves to peers programming in other mainstream languages like Java and finding their prospects and career paths suboptimal in comparison, which can lead to frustration and depression [36]. Parts of the solution include finding alternative role models online (virtually meeting senior PHP developers), changing the work environment, as well as actually learning new languages and frameworks [36]. Interestingly, the last element is widely considered to be the responsibility of the programmer [65], which is unlike other professions (e.g., construction workers are not expected to learn how to use new equipment in their free time and make effort to convince their employer to purchase it; software construction workers are).

As a more positive example, there are people who first become diagnosed with psychological or psychiatric disorders due to non-programming issues in their lives, and then learn a new language which helps them overcome the illness. As an example case: a person with some C++ knowledge developed signs of depression, learnt programming JavaScript at Free Code Camp (online) and was entertained enough to eventually find a way to recover and address other issues in his life [4].

RQ3: does knowing a particular language cause direct harm in the sense of making a person a worse programmer?

Many years ago FORTRAN was claimed to "mentally chain" programmers to mistakes of past language designers [15]. Another well-known claim of the same era was that "it is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration" [16]. Both claims are interesting in the sense that they seem to assume that once a language has been properly learnt, it is impossible (remains to be guessed: literally or figuratively) to "unlearn" its harmful patterns. If this is found to be true, it would mean exactly what we set to find out from the start: that languages can cause lasting mental harm and "break" programmers beyond repair. More recently, the focus has shifted to languages like PHP, which, reportedly, not only make people write bad code in that language, but leaves a mark when they "redeem themselves"

and switch to other languages and frameworks—even though there is no strong community consensus on this [38, 67].

RQ4: does knowing a particular language cause indirect harm in the sense of making a person worse in communicating ideas and collaborating with others in the context of software creation?

Programming is not a solitary activity and must be studied in the context of a programming social unit [75]. Programmers need to participate in sprints [68], pair programming sessions [7], code reviews [6], sometimes even discuss requirements with the clients and present a prototype solution to them. All these can be frustrating and mentally harmful, especially when reinforced by work and life habits of isolation and perfectionism [40, 45, 47] and can "dramatically increase the likelihood of clinical depression" [45].

There is some anecdotal evidence to be found in informal primary sources such as programmers' blogs, and there are plenty of cases when programmers find out the hard way that the classic claim that learning more languages will make one a better programmer, is not necessarily true. For instance, learning Haskell can make a programmer demotivated since (s)he will start to notice potential errors everywhere in imperative and/or untyped code in other languages [55]. One can also find out painfully that nice good idioms learnt in one language, do not necessarily port well to other languages, making the code hard to explain to your peers and making it look obfuscated to them [55, 80].

Another aspect is that knowing languages that you consider good, and at the same time programming daily in languages you consider bad, can in itself be a source of sadness, frustration and depression. A question was asked at some point on Quora: "As a serious programmer, you taught yourself Haskell, Erlang and Prolog and loads more besides because you care. Do you get very depressed that most big jobs are forced into using lowest common denominator technologies because it is easier to hire people?" [35]. All the highly upvoted answers are variations of "yes" or "yes, but...".

RQ5: can the first programming language learnt by a programmer, have any long-term effects like preventing the programmer to learn and effectively use new constructions and abstractions?

On one hand, the *firsts* are known to be extremely important: the first natural language learnt by a child shapes the way (s)he thinks about the world even if new languages are acquired later; the age of the first sexual intercourse determines to some extent the levels of relationship dissatisfaction later in life [28]; the first job (or the second one, if the first one is considered superfluous [18]) is the most important for skill acquisition and the subsequent career [73], etc. Thus, it is sensible to assume that the first language used to learn basic programming concepts, has lasting consequences for the programmer's career and personal development.

On the other hand, it has been widely observed that experienced programmers have a completely different approach to problem than junior programmers, they organise information in a fundamentally different way [49]; in linguistics it is known and commonly measured that learning a second language influences the usage of the first [11] up to the point of semantic (not syntactic) language attrition [69]; and even the assumption that first love is imprinting expectations for the rest of adult live, is often challenged and found untrue [37]. Hence, the entire list of languages known by a programmer, may weigh much more heavily on their skills and mental state, than the first language per se.

There are, again, two well-known claims about mental harm caused by BASIC and COBOL used as first languages to learn programming:

- "The teaching of BASIC should be rated as a criminal offence: it *mutilates the mind* beyond recovery." [17]
- "The use of COBOL *cripples the mind*; its teaching should, therefore, be regarded as a criminal offence." [16]

Rather obviously, the claims were metaphorical exaggerations at the moment of their utterance, and were completely unsupported by any sort of evidence beyond the confidence of the author. Nevertheless, we claim that it is useful to investigate if there is any degree of harmful mental effects of the first language on its users, and that question has never been raised so far.

Nowadays many people recommend scripting languages like Python or Ruby that allow anyone to start with one-liners and slowly move towards building up more serious applications. However, the assumption that this helps, is unsupported by psychological research that states that it is better to have a relatively stable (even if high) line count per solution than have a mix of snappy one-lines and lengthy boilerplate [75].

3 Experiments

3.1 Experimental Setup 1

To answer RQ1 and link the mental state of a programmer with the quality of the code produced, we will use controlled isolated environments and use mood priming and construct activation to introduce the bias, for which there are many known methods to choose from [63].

For example, group A will be welcomed warmly, their skills will be complimented and unexpected positive reinforcement given. Group B will receive a colder welcome and a demotivating introduction speech complaining about their skill level, and in general their expectations will deliberately be unmet. Then, programming tasks such as writing a piece of code to solve a problem or adjusting existing code to fix a bug, will be given, identical to each group, timed and collected for further investigation by code analysis techniques.

Other mental states can be measured similarly. To compensate for the (a priori unknown) initial state of mind of

experiment participants and to calibrate their self-focus [58], we will provide standardised writing assignments before the start of the experiment [22].

3.2 Experimental Setup 2

To answer RQ2, we will combine techniques from the domains of mining software repositories and natural language processing. For each of the languages chosen for the experiment, we will collect corpora of texts representative of that language: documentation for the language itself as well as its libraries and utilities; posts from thematic discussion boards and StackOverflow tags; developer discussions in internal communication (email, slack), whenever available, as well as external communication (issue comments on the tracker); vocabularies used within the codebase for variable and method names; comments within the code; etc. These corpora should be prepared as isolated as technically possible—some languages such as HTML, CSS and JavaScript will be a challenge: their use is intertwined but typical representatives of developers in individual languages have vastly different background and self-perception. Having these corpora collected, we will apply known automated text-based emotion detection techniques [9, 43] to them and compare results.

Pyszczynski et al. show that internal focus is linked with depression [58]. In our context it means that in programming language communities where it is more common to link programs and bugs to their authors, developers may be more self-focused and thus prone to depression.

3.3 Experimental Setup 3

To answer RQ3, we will rely on the large body of code in FLOSS (Free and Libre Open Source Systems) projects [59, 60], available in highly annotated form with information on authorship and collaboration on a daily basis spanning over decades. We will need to collect additional information from open source developers about what languages they know, and correlate that information with results of analysis of the code they create. Previously similar research initiatives were successful in seeking impact of developer turnover [23], social diversity [71], gender diversity [70], programming social unit size increases [50], prior social links [10], package popularity [62] and linguistic antipatterns [2] on software quality.

3.4 Experimental Setup 4

To answer RQ4, a similar setup can be used to research collaboration habits and collaboration patterns within clusters formed by projects of same/similar languages. Prior possibly related work in this direction includes cross-cultural collaborative learning [41] and research on social tension between what is better for the team versus what is better for the individual programmer [25]. To the best of our knowledge, the influence of the programming language has not yet been studied, but the influence of paradigms (e.g., Agile [42]) has.

It is known from psychological research that depressed people tend to be more negative towards people from their close circles and can behave in a default or even less negative way towards people from outside [64]. Thus, our expectations are to see similar negativity spikes within longer existing projects than across communities.

3.5 Experimental Setup 5

To answer RQ5, we will collect information with questionnaires on first languages learnt by programmers which code is available for analysis and whose careers are also measurable. As usual in such cases, the exact setup will feature blind assessors that will not know the first language when assessing the code and careers, and inter-assessor reliability will have to be calculated to rule out possible biases.

4 Threats to Validity

Defining and measuring the confounding factors will be a challenge. For instance, what if being depressed or having in a particular state of mind, has direct influence on the choice of the language?

Another interesting challenge is that, *if* programming is almost always depressing anyway [81], then we cannot simply trust opinions of randomly sampled programmers without diagnosing them. As an example, consider a case reported at the Open Sourcing Mental Illness Help boards [44]. A clinically depressed person with a schizophrenic parent anonymously asked how to learn programming. The first answers were, not unexpectedly, Python and Ruby. After receiving those, the original poster commented that (s)he already finished Codecademy's courses on PHP, JavaScript, jQuery, CSS and HTML [44]. However, most mentally healthy and optimistic people who had done the same, would have already started to consider themselves programmers, and would not ask for more help to get there. In general, it is known that depressed people tend to put the blame of failures on themselves (e.g., "I have come to realize that I'm a terrible programmer"[12], or "my brain reminds me that I'm worthless and I end up giving up" [14]) and the credit for success on others [27, 57] (including external factors, such as the choice of the language).

5 Conclusion

We have combined expertise of the three authors on cognitive psychology, software safety and language design, respectively, to describe the problem of linking software languages with possibly harmful changes in the mental state of language users. Several research questions were formulated and motivated in § 2. Experimental setups were sketched in § 3 by leveraging prior research on related topics. We believe the problem stated above is relevant for team productivity, retention and turnover, as well as for the DSL creators.

References

[1] Sorin Adam, Marco Kuhrmann, and Ulrik Pagh Schultz. 2016. Towards a Virtual Machine Approach to Resilient and Safe Mobile Robots. In *Proceedings of the 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1–8. https://doi.org/10.1109/ETFA.2016.7733545

[2] Venera Arnaoudova. 2010. Improving Source Code Quality through the Definition of Linguistic Antipatterns. In *Proceedings of the 17th Working Conference on Reverse Engineering (WCRE)*, Giuliano Antoniol, Martin Pinzger, and Elliot J. Chikofsky (Eds.). IEEE Computer Society, 285–288. https://doi.org/10.1109/WCRE.2010.41

[3] Henri E. Bal and Dick Grune. 1994. *Programming Languages Essentials*. Addison-Wesley. https://dickgrune.com/Books/PLE/

[4] Adham El Banhawy. 2017. How Coding Helps Treat My Depression. Medium. (Jan. 2017). https://medium.com/@the.benhawy/how-coding-helps-treat-my-depression-f8d016f7e98e

[5] Santiago Basulto. 2015. The Depressed Programmer. Medium. (Dec. 2015). https://medium.com/@santiagobasulto/the-depressed-programmer-49076d8b33f0

[6] Gabriele Bavota and Barbara Russo. 2015. Four Eyes are Better than Two: on the Impact of Code Reviews on Software Quality. In *Proceedings of the 31st International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 81–90. https://doi.org/10.1109/ICSM.2015.7332454

[7] Kent Beck. 1998. Extreme Programming: A Humanistic Discipline of Software Development. In *Proceedings of the First International Conference on Fundamental Approaches to Software Engineering (FASE) (LNCS)*, Vol. 1382. Springer, 1–6. https://doi.org/10.1007/BFb0053579

[8] Daniel Berrar, Akihiko Konagaya, and Alfons Schuster. 2013. Turing Test Considered Mostly Harmless. *New Generation Computing* 31, 4 (Oct. 2013), 241–263. https://doi.org/10.1007/s00354-013-0401-2

[9] Lars Buitinck, Jesse van Amerongen, Ed Tan, and Maarten de Rijke. 2015. Multi-emotion Detection in User-Generated Reviews. In *Proceedings of the 37th European Conference on Information Retrieval Research: Advances in Information Retrieval (ECIR) (LNCS)*, Vol. 9022. Springer, 43–48. https://doi.org/10.1007/978-3-319-16354-3_5

[10] Casey Casalnuovo, Bogdan Vasilescu, Premkumar T. Devanbu, and Vladimir Filkov. 2015. Developer Onboarding in GitHub: The Role of Prior Social Links and Language Experience. In *Proceedings of the 10th Joint Meeting of the 15th European Software Engineering Conference and the 23rd Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 817–828. https://doi.org/10.1145/2786805.2786854

[11] Vivian Cook. 2003. *Effects of the Second Language on the First*. Multilingual Matters.

[12] csthrowtoss. 2016. I'm Terrible At Programming And It's Making Me Depressed. Please Help. Reddit. (2016). https://www.reddit.com/r/cscareerquestions/comments/4urv9w/im_terrible_at_programming_and_its_making_me/

[13] Kostadin Damevski, David C. Shepherd, Johannes Schneider, and Lori L. Pollock. 2017. Mining Sequences of Developer Interactions in Visual Studio for Usage Smells. *IEEE Transactions on Software Engineering* 43, 4 (2017), 359–371. https://doi.org/10.1109/TSE.2016.2592905

[14] Depressed Developer. 2015. I'm a developer, and the community scares me... Medium. (Nov. 2015). https://medium.com/@depressed.developer/i-m-a-developer-and-the-community-scares-me-91b92e1d982c

[15] Edsger W. Dijkstra. 1972. The Humble Programmer. *Commun. ACM* 15, 10 (1972), 859–866. Turing Award lecture.

[16] Edsger W. Dijkstra. 1982. How Do We Tell Truths That Might Hurt? In *Selected Writings on Computing: A Personal Perspective*. Springer, 129–131.

[17] Edsger W. Dijkstra. 1984. The Threats to Computing Science. (Nov. 1984). http://www.cs.utexas.edu/users/EWD/ewd08xx/EWD898.PDF

[18] Early-Career Psychology. 2005. How to Land Your Second Job. *Monitor on Psychology* 36, 9 (Oct. 2005), 58.

[19] Eva Van Emden and Leon Moonen. 2002. Java Quality Assurance by Detecting Code Smells. In *Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE)*, Arie van Deursen and Elizabeth Burd (Eds.). IEEE Computer Society, 97–108.

[20] Milton H. Erickson and Ernest L. Rossi. 1979. *Hypnotherapy: An Exploratory Casebook*. Irvington Publishers.

[21] Amin Milani Fard and Ali Mesbah. 2013. JSNOSE: Detecting JavaScript Code Smells. In *Proceedings of the 13th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 116–125. https://doi.org/10.1109/SCAM.2013.6648192

[22] Allan Fenigstein and Michael P. Levine. 1984. Self-attention, Concept Activation, and the Causal Self. *Journal of Experimental Social Psychology* 20, 3 (1984), 231–245. https://doi.org/10.1016/0022-1031(84)90049-0

[23] Matthieu Foucault, Marc Palyart, Xavier Blanc, Gail C. Murphy, and Jean-Rémy Falleri. 2015. Impact of Developer Turnover on Quality in Open-source Software. In *Proceedings of the 10th Joint Meeting of the 15th European Software Engineering Conference and the 23rd Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 829–841. https://doi.org/10.1145/2786805.2786870

[24] Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. 1999. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.

[25] Gabriele Frankl, Sofie Bitter, and Bonifaz Kaufmann. 2014. Win-for-All in Software Engineering Education: Balancing Social Dilemmas to Foster Collaboration. In *Proceedings of the 27th Conference on Software Engineering Education and Training (CSEE& T)*. IEEE, 163–167. https://doi.org/10.1109/CSEET.2014.6816795

[26] François Gauthier and Ettore Merlo. 2013. Semantic Smells and Errors in Access Control Models: A Case Study in PHP. In *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, David Notkin, Betty H. C. Cheng, and Klaus Pohl (Eds.). IEEE / ACM, 1169–1172.

[27] Jeff Greenberg and Tom Pyszczynski. 1986. Persistent High Self-focus After Failure and Low Self-focus After Success: The Depressive Self-Focusing Style. *Journal of Personality and Social Psychology* 50, 5 (1986), 1039–1044.

[28] K. Paige Harden. 2012. True Love Waits? A Sibling-Comparison Study of Age at First Sexual Intercourse and Romantic Relationships in Young Adulthood. *Psychological Science* 23, 11 (2012), 1324–1336. https://doi.org/10.1177/0956797612442550

[29] Benedikt Hauptmann, Maximilian Junker, Sebastian Eder, Lars Heinemann, Rudolf Vaas, and Peter Braun. 2013. Hunting for smells in natural language tests. In *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, David Notkin, Betty H. C. Cheng, and Klaus Pohl (Eds.). IEEE / ACM, 1217–1220.

[30] Patrick Hayes and Kenneth Ford. 1995. Turing Test Considered Harmful. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, 972–977.

[31] Felienne Hermans, Martin Pinzger, and Arie van Deursen. 2012. Detecting Code Smells in Spreadsheet Formulas. In *Proceedings of the 28th International Conference on Software Maintenance (ICSM)*. IEEE Computer Society, 409–418. https://doi.org/10.1109/ICSM.2012.6405300

[32] ICTR-95-1 1999. Kayishema and Ruzindana, Trial Judgement. United Nations, Mechanism for International Criminal Tribunals. (May 1999). http://unictr.unmict.org/en/cases/ictr-95-1

[33] Johann Thor Mogensen Ingibergsson, Dirk Kraft, and Ulrik Pagh Schultz. 2017. Safety Computer Vision Rules for Improved Sensor Certification. In *Proceedings of the First IEEE International Conference on Robotic Computing (IRC)*. IEEE, 89–92. https://doi.org/10.1109/IRC.2017.27

[34] Johann Thor Mogensen Ingibergsson, Ulrik Pagh Schultz, and Dirk Kraft. 2016. Towards Declarative Safety Rules for Perception Specification Architectures. *CoRR* abs/1601.02778 (2016), 4. http://arxiv.org/abs/1601.02778

[35] Tikhon Jelvis, Michael O. Church, Debasish Ghosh, Jeff Nelson, Anne Ogborn, Alex Jouravlev, Dima Korolev, Matt Wartell, Jesse Armand, and Nicolae Marasoiu. 2011. As a "serious" programmer, you taught yourself Haskell, Erlang and Prolog and loads more besides because you care. Do you get very depressed that most "big jobs" are forced into using lowest common denominator technologies because it is easier to hire people? Quora. (2011). https://www.quora.com/As-a-serious-programmer-you-taught-yourself-Haskell-Erlang-and-Prolog-and-loads-more-besides-because-you-care-Do-you-get-very-depressed-that-most-big-jobs-are-forced-into-using-lowest-common-denominator-technologies-because-it-is-easier-to-hire-people

[36] Pratik C. Joshi. 2015. Am I failed in life by choosing PHP development as a career? The Laravel Community Portal. (2015). https://laravel.io/forum/am-i-failed-in-life-by-choosing-php-development-as-a-career

[37] Nancy Kalish. 2010. First Love, Lost Love: Is It Imprinting? *Psychology Today* (July 2010). https://www.psychologytoday.com/blog/sticky-bonds/201007/first-love-lost-love-is-it-imprinting

[38] Anders Kaseorg, Thiago Campezzi, Jon Moter, Beekey Cheung, Adrian Nenu, Joel Rivera, Ryan Willis, Josh Begleiter, Anders Borg, and Lee Griffin. 2011. Why is PHP Hated by So Many Developers? Quora. (2011). https://www.quora.com/PHP-programming-language-1/Why-is-PHP-hated-by-so-many-developers

[39] Marouane Kessentini and Ali Ouni. 2017. Detecting Android Smells Using Multi-Objective Genetic Programming. In *Proceedings of the Fourth IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE, 122–132. https://doi.org/10.1109/MOBILESoft.2017.29

[40] Timothy King. 2009. Depression and the Software Developer. Software Development: A Love-Hate Relationship. (April 2009). http://sd.jtimothyking.com/2009/04/17/depression-and-the-software-developer/

[41] Kathrin Kirchner and Liana Razmerita. 2015. Collaborative Learning in the Cloud: A Cross-Cultural Perspective of Collaboration. In *Proceedings of the 26th Conference on Hypertext and Social Media (HT)*. ACM, 333–336. https://doi.org/10.1145/2700171.2804452

[42] Martin Kropp, Andreas Meier, Magdalena Mateescu, and Carmen G. Zahn. 2014. Teaching and Learning Agile Collaboration. In *Proceedings of the 27th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 139–148. https://doi.org/10.1109/CSEET.2014.6816791

[43] Adel Lablack, Taner Danisman, Ioan Marius Bilasco, and Chabane Djeraba. 2014. A Local Approach for Negative Emotion Detection. In *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR)*. IEEE, 417–420. https://doi.org/10.1109/ICPR.2014.80

[44] LordByron. 2013. How to Learn to Program/Code/Script While Depressed. Open Sourcing Mental Illness. (June 2013). https://forums.osmihelp.org/t/how-to-learn-to-program-code-script-while-depressed/86

[45] Jason Lowenthal. 2016. The Programmer's Great Clinical Depression. Simple Programmer. (Oct. 2016). https://simpleprogrammer.com/2016/10/07/programmers-great-clinical-depression/

[46] Bruce J. MacLennan. 1983. *Principles of Programming Languages: Design, Evaluation and Implementation.* Holt-Saunders.

[47] Lauren Maffeo. 2012. Developer Depression: Isolation is the Biggest Problem. The Next Web. (Oct. 2012). https://thenextweb.com/insider/2012/10/20/are-developers-depressed/

[48] Davood Mazinanian, Nikolaos Tsantalis, and Ali Mesbah. 2014. Discovering Refactoring Opportunities in Cascading Style Sheets. In *Proceedings of the 22nd Symposium on the Foundations of Software Engineering (FSE)*. ACM, 496–506. https://doi.org/10.1145/2635868.2635879

[49] Katherine B. McKeithen, Judith S. Reitman, Henry H. Rueter, and Stephen C. Hirtle. 1981. Knowledge Organization and Skill Differences in Computer Programmers. *Cognitive Psychology* 13, 3 (1981), 307 – 325. https://doi.org/10.1016/0010-0285(81)90012-8

[50] Andrew Meneely, Pete Rotella, and Laurie Williams. 2011. Does Adding Manpower also Affect Quality? An Empirical, Longitudinal Analysis. In *Proceedings of the 19th Symposium on the Foundations of Software Engineering and the 13th European Software Engineering Conference (ESEC-FSE)*, Tibor Gyimóthy and Andreas Zeller (Eds.). ACM, 81–90. https://doi.org/10.1145/2025113.2025128

[51] Marjan Mernik, Jan Heering, and Anthony M. Sloane. 2005. When and How to Develop Domain-Specific Languages. *Comput. Surveys* 37, 4 (2005), 316–344. https://doi.org/10.1145/1118890.1118892

[52] Mika Mäntylä, Jari Vanhanen, and Casper Lassenius. 2004. Bad Smells — Humans as Code Critics. In *Proceedings of the 20th International Conference on Software Maintenance (ICSM)*. IEEE Computer Society, 399–408. https://doi.org/10.1109/ICSM.2004.1357825

[53] Kjeld Høyer Mortensen, Kari R. Schougaard, and Ulrik Pagh Schultz. 2004. Distance-Based Access Modifiers Applied to Safety in Home Networks. In *Proceedings of the Second European Symposium on Ambient Intelligence (EUSAI) (LNCS)*, Panos Markopoulos, Berry Eggen, Emile H. L. Aarts, and James L. Crowley (Eds.), Vol. 3295. Springer, 315–326. https://doi.org/10.1007/978-3-540-30473-9_30

[54] Steffen M. Olbrich, Daniela Cruzes, and Dag I. K. Sjøberg. 2010. Are All Code Smells Harmful? A Study of God Classes and Brain Classes in the Evolution of Three Open Source Systems. In *Proceedings of the 26th International Conference on Software Maintenance (ICSM)*. IEEE Computer Society, 1–10. https://doi.org/10.1109/ICSM.2010.5609564

[55] Luke Plant. 2006. Why Learning Haskell/Python Makes You a Worse Programmer. Luke Plant's home page. (Aug. 2006). https://lukeplant.me.uk/blog/posts/why-learning-haskell-python-makes-you-a-worse-programmer/

[56] Leonard Punt, Sjoerd Visscher, and Vadim Zaytsev. 2016. The A?B*A Pattern: Undoing Style in CSS and Refactoring Opportunities it Presents. In *Proceedings of the 32nd International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 67–77. https://doi.org/10.1109/ICSME.2016.73

[57] Tom Pyszczynski and Jeff Greenberg. 1986. Evidence for a Depressive Self-Focusing Style. *Journal of Research in Personality* 20, 1 (1986), 95 – 106. https://doi.org/10.1016/0092-6566(86)90112-1

[58] Tom Pyszczynski, James C. Hamilton, Fred H. Herring, and Jeff Greenberg. 1989. Depression, Self-Focused Attention, and the Negative Memory Bias. *Journal of Personality and Social Psychology* 57, 2 (8 1989), 351–357.

[59] Filippo Ricca and Alessandro Marchetto. 2010. Heroes in FLOSS Projects: An Explorative Study. In *Proceedings of the 17th Working Conference on Reverse Engineering (WCRE)*, Giuliano Antoniol, Martin Pinzger, and Elliot J. Chikofsky (Eds.). IEEE Computer Society, 155–159. https://doi.org/10.1109/WCRE.2010.25

[60] Gregorio Robles, Laura Arjona Reina, Alexander Serebrenik, Bogdan Vasilescu, and Jesús M. González-Barahona. 2014. FLOSS 2013: A Survey Dataset about Free Software Contributors: Challenges for Curating, Sharing, and Combining. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*. ACM, 396–399. https://doi.org/10.1145/2597073.2597129

[61] Serguei A. Roubtsov, Alexander Serebrenik, and Mark van den Brand. 2010. Detecting Modularity "Smells" in Dependencies Injected with Java Annotations. In *Proceedings of the 14th European Conference on Software Maintenance and Reengineering (CSMR)*, Rafael Capilla, Rudolf Ferenc, and Juan C. Dueñas (Eds.). IEEE, 244–247. https://doi.org/10.1109/CSMR.2010.45

[62] Hitesh Sajnani, Vaibhav Saini, Joel Ossher, and Cristina Videira Lopes. 2014. Is Popularity a Measure of Quality? An Analysis of Maven

Components. In *Proceedings of the 30th International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 231–240. https://doi.org/10.1109/ICSME.2014.45

[63] Zindel V. Segal and Rick E. Ingram. 1994. Mood Priming and Construct Activation in Tests of Cognitive Vulnerability to Unipolar Depression. *Clinical Psychology Review* 14, 7 (1994), 663–695. https://doi.org/10.1016/0272-7358(94)90003-5

[64] Chris Segrin and Jeanne Flora. 1998. Depression and Verbal Behavior in Conversations with Friends and Strangers. *Journal of Language and Social Psychology* 17, 4 (1998), 492–503. https://doi.org/10.1177/0261927X980174005

[65] Josh Sherman. 2015. PHP is Stupid and So Are You. https://joshtronic.com/2015/08/23/php-is-stupid-and-so-are-you/. (Aug. 2015).

[66] Diomidis Spinellis. 2001. Notable Design Patterns for Domain-specific Languages. *The Journal of Systems and Software* 56, 1 (2001), 91–99. https://doi.org/10.1016/S0164-1212(00)00089-3

[67] Phil Sturgeon, Ben Werdmuller, Michael Haeuslmann, Yishan Wong, Les Orchard, Rick Harding, Shane Rifles, Alex Feinberg, Irakli Nadareishvili, and Koushik Ramachandra. 2010. Do a Large Majority of People Hate PHP Solely Because Other People Do So? Quora. (2010). https://www.quora.com/Do-a-large-majority-of-people-hate-PHP-solely-because-other-people-do-so

[68] J. Sutherland and J. J. Sutherland. 2014. *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Publishing Group.

[69] Ianthi Tsimpli, Antonella Sorace, Caroline Heycock, and Francesca Filiaci. 2004. First Language Attrition and Syntactic Subjects: A Study of Greek and Italian Near-native Speakers of English. *International Journal of Bilingualism* 8, 3 (2004), 257–277. https://doi.org/10.1177/13670069040080030601

[70] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark van den Brand, Alexander Serebrenik, Premkumar T. Devanbu, and Vladimir Filkov. 2015. Gender and Tenure Diversity in GitHub Teams. In *Proceedings of the 33rd ACM SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3789–3798. https://doi.org/10.1145/2702123.2702549

[71] Bogdan Vasilescu, Alexander Serebrenik, and Vladimir Filkov. 2015. A Data Set for Social Diversity Studies of GitHub Teams. In *Proceedings of the 12th IEEE/ACM Working Conference on Mining Software Repositories (MSR)*. IEEE, 514–517. https://doi.org/10.1109/MSR.2015.77

[72] Nicole Vavrová and Vadim Zaytsev. 2017. Does Python Smell Like Java? *The Art, Science and Engineering of Programming (‹Programming›)* 1 (April 2017), 11–1–11–29. Issue 2. https://doi.org/10.22152/programming-journal.org/2017/1/11

[73] Dieter Verhaest and Eddy Omey. 2010. The Measurement and Determinants of Skill Acquisition in Young Workers' First Job. *Economic and Industrial Democracy* 31, 1 (2010), 116–149. https://doi.org/10.1177/0143831X09343994

[74] Eric Wastl. 2011. PHP Sadness. http://www.phpsadness.com. (May 2011).

[75] Gerald M. Weinberg. 1998. *The Psychology of Computer Programming* (silver anniversary ed.). Dorset House Publishing.

[76] David S. Wile. 2001. Supporting the DSL Spectrum. *Journal of Computing and Information Technology* 9, 4 (2001), 263–287.

[77] Leslie B. Wilson and Robert G. Clark. 1993. *Comparative Programming Languages* (second ed.). Addison-Wesley.

[78] Aiko Fallas Yamashita and Leon Moonen. 2012. Do Code Smells Reflect Important Maintainability Aspects?. In *Proceedings of the 28th International Conference on Software Maintenance (ICSM)*. IEEE Computer Society, 306–315. https://doi.org/10.1109/ICSM.2012.6405287

[79] Aiko Fallas Yamashita and Leon Moonen. 2013. Do Developers Care about Code Smells? An Exploratory Survey. In *Proceedings of the 20th Working Conference on Reverse Engineering (WCRE)*, Ralf Lämmel, Rocco Oliveto, and Romain Robbes (Eds.). IEEE, 242–251. https://doi.org/10.1109/WCRE.2013.6671299

[80] Jiyin Yiyong. 2017. Some Depressions Being a Functional Language Programmer. Medium. (Jan. 2017). https://medium.com/@jiyinyiyong/some-depressions-being-a-functional-language-programmer-df4048879050

[81] Zarar. 2013. Depressed Programmer. https://depressedprogrammer.wordpress.com. (2013).