Evaluating the usability of integrated development environments

Evaluación de la usabilidad de los entornos de desarrollo integrados

Jenny Morales[1], Fabián Silva-Aravena[1], Paula Sáez[1]

[1] Universidad Católica del Maule. Departamento de Economía y Administración. Talca, Chile.

DOI: 10.64966/ingeniare.v34.05

Abstract

Programming can be a difficult task for both students and professionals, requiring programming environments that include editors, compilers, and debugging tools. This study assessed the usability of three commonly used programming environments in higher education, NetBeans, Eclipse, and Dev-C++, through a System Usability Scale (SUS) survey, with participation from 143 computer engineering students. NetBeans received the highest usability perception, but all three IDEs scored below the usability threshold (SUS score of 68), particularly struggling in learnability. A consultation with experts was also conducted to gather additional information. Expert feedback from seven professionals highlighted NetBeans, Eclipse, and Visual Studio as the most commonly used IDEs, emphasizing simplicity and rich features as their key strengths. Key areas for improvement include usability, customizable interfaces, and integration of AI-assisted coding. Future work will include evaluating Visual Studio Code and the usability of AI code generation tools.

Keywords: Usability, programming environment, usability perception.

Resumen

Programar puede ser una tarea difícil tanto para estudiantes como para profesionales, ya que requiere entornos de programación que incluyan editores, compiladores y herramientas de depuración. Este estudio evaluó la usabilidad de tres entornos de programación comúnmente utilizados en la educación superior (NetBeans, Eclipse y Dev-C++) mediante una encuesta utilizando la Escala de Usabilidad del Sistema (SUS), con la participación de 143 estudiantes de ingeniería informática. NetBeans recibió la mejor percepción de usabilidad, pero los tres IDE obtuvieron puntuaciones por debajo del umbral de usabilidad (puntuación SUS de 68), con especial dificultad para aprender. También realizamos una consulta a expertos para recopilar información complementaria. Las opiniones de siete profesionales destacaron NetBeans, Eclipse y Visual Studio como los IDE más utilizados, destacando la simplicidad y la gran cantidad de funciones como sus principales fortalezas. Las áreas clave de mejora incluyen la usabilidad, las interfaces personalizables y la integración de la codificación asistida por IA. El trabajo futuro incluirá la evaluación de Visual Studio Code y la usabilidad de las herramientas de generación de código con IA.

Palabras clave: Usabilidad, entornos de programación, percepción de usabilidad.

Autor de correspondencia: jmoralesb@ucm.cl

Received: 5 September 2025 | Accepted: 25 May 2026

INTRODUCTION

Programming is a challenging endeavor for students and professionals alike. Programmers need a programming environment to write their code. Programming environments typically contain, among other elements, editors for different languages, compilation tools, and debugging tools. Various programming environments exist to support different types of users, ranging from children to adults and technology professionals to non-professionals. In this paper, we focus on analyzing Integrated Programming Environments (IDEs) used in Computer Science degree programs. The objective of this study is to determine whether the IDEs students use during their undergraduate training meet minimum usability requirements that support learning and tool use. This study can also serve as input for those who teach programming and help determine which IDEs students use.

A survey tailored to programming environments using the System Usability Scale (SUS) was conducted to gather perceptions of usability across Dev-C++, Eclipse, and NetBeans. Also, expert programmers were asked to gather further information about the programming environment they use and their perceptions of usability.

The System Usability Scale (SUS) results show that NetBeans achieved the highest score among the assessed IDEs. However, the low overall scores suggest that NetBeans, Eclipse, and Dev-C ++ are not fully usable. Expert consultations further emphasize the significance of usability and identify the incorporation of AI for code generation as a desirable feature.

This work is organized as follows: the background section provides relevant information about the programming environments, usability, user experience, and programmer experience; the Methodology section presents the questionnaires used to collect usability information on the programming environments; the results section presents the analysis of the collected data; and finally, we present the conclusion and future work.

BACKGROUND

Learning to program is a challenging activity for beginners, which explains the difficulties that students face when they start programming courses [1,2]. This has generated a great variety of programming environments that aim to facilitate the incorporation of programming concepts through graphical interfaces, thereby avoiding the complexity of the syntax [3,4]. Then the concern arises about the usability of these environments, to determine if they really meet the declared goals. Below are explained the concepts of programming environments, usability, and user experience, which are based on the programmer's experience, on which this work is based.

Programming environments

A programming environment provides programmers with several tools for developing software. It is also known as a development environment or an integrated development environment (IDE). An IDE can support multiple programming languages. This includes a compiler, interpreter, or both. It features a source code editor, a debugger, and tools such as code auto-completion, language keyword recognition, and auto-indentation to maintain program order through proper

indentation. A programming environment is one of the most used tools by programmers. Several articles about IDEs were found, such as articles that propose improvements to IDEs used by professional programmers, such as visibility of code [5], [6], navigation of code [7], [8], improving the process of refactoring [9], and making a more understandable error message [10], [11].

IDEs are explicitly created for developing programming skills in beginners [12], as well as for non-programmers and children [13].

Usability

Usability is defined as follows: "The Extent to which specified users can use a system, product, or service to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use"[14]. Usability is composed for attributes such as: Learning is a vital attribute for novice users, related to easy to learn; Efficiency is the speed with which the user can reach their objectives, or how much steps the user needs to complete a task; Memorability is the ability to be remembered for sporadic users or frequent users; Errors, a good indicator in this attribute is a low number of errors committed by the user during the development of a task; and Subjective satisfaction, a subjective attribute that measures the impression subjective that the user has of the system [15].

User experience

User eXperience (UX) is defined as follows: “person's perceptions and responses resulting from the use and anticipated use of a product, system or service” [16]. UX is unique for different people, given the emotional factors involved in how they uses a product, service, or system, the satisfaction derived from use, and the time period of use.

One of the models used to explain UX aspects is the Honeycomb model, a website user experience model proposed by Morville [17]. This model comprises seven aspects: Valuable, Usable, Useful, Desirable, Findable, Accessible, and Credible.

Programmer experience

Programmers are the people who write and maintain code. This work focuses on this specific role in software development.

The programmers are considered users of specific software, such as programming environments or design documents. In this way, the PX was investigated from the perspective of programmers, specifically in the case of UX.

In previous studies, we found several works on developers and the impact of happiness and unhappiness on their work [18] - [21]. Another study focuses on programmer behavior and indicates the activities they carried out in their work, providing information about the tools used [22]. It also examines the effort programmers invest in implementing code changes [23].

All of these studies provide interesting insights into the programmers. After analysis it was found that PX is composed of intrinsic motivation, technical skills, and social skills of the programmers.

In addition, PX can be influenced by the tools programmers use in their work.

Usability in programming environments

Various usability studies have been carried out across different programming environments, for example, those that provide graphical programming interfaces [24,25], intended to be simple and friendly for non-programmer end users.

Specifically, there are studies comparing block programming environments and hybrid environments that, in addition to blocks, support data-flow programming. In their study, the authors found that users prefer hybrid environments and found high usability for block programming [26].

On the one hand, some studies in programming environments for programmers (not end users) show that perceived usability, ease of learning, and navigability can affect the adoption of a technology [27]. On the other hand, studies on the usability of programming environments have evaluated perceived usability, and some have identified various issues [28].

Therefore, usability in programming environments can be seen as a crucial element of study for both end users and programmers, who must use them to program. If these programming environments are usable, they could promote their adoption, facilitate learning, and help them achieve their goals.

METHODOLOGY

The System Usability Scale (SUS) questionnaire was used to conduct this work, as shown in Table 1. This questionnaire is well-known and widely used. This questionnaire contains five positive (uneven) and five negative (even) statements. Respondents' level of agreement is graded on a 5-point Likert scale, with 5 indicating "strongly agree" and 1 indicating "strongly disagree." Scores are calculated for odd-numbered questions as the sum of the scores minus 5, and for even-numbered questions, 25 minus the sum of the scores. Finally, the sum of the above results is multiplied by 2.5 to give the final score [29]. The questionnaire was administered to undergraduate computer engineering students over a four-month period. All of the participants freely and voluntarily shared their perceptions. Regarding the participants, they regularly used these programming environments for their programming assignments in the various courses they took in their undergraduate program. The usability assessment in this study was based on the students' accumulated experience working with these environments. The questionnaire was administered through a classroom intervention, during which the study's objective was explained, and guidance was provided. Those who freely chose to participate were given a maximum of 20 minutes to complete the questionnaire.

Table 1. System usablity scale.

n°	Item
1	I think that I would like to use this system frequently.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I think that I would need the support of a technical person to be able to use this system.
5	I found the various functions in this system were well integrated
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
9	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.

Source: Brooke, [29].

The questionnaire, detailed in Table 2, was sent to expert programmers with more than five years of experience to supplement the survey. The questionnaire was distributed via email. The responses were collected between November and December 2024.

Table 2. Questions for expert programmers.

n°	Item
1	What programming environments do you frequently use?
2	If you had to recommend a programming environment, which one would it be and why?
3	Indicate one or more positive aspects of the programming environments you have used.
4	What would you like to improve about the programming environments?

RESULTS

Initially, the questionnaire contained information relevant to consent to participate and the confidentiality with which the data would be treated.

Participants were surveyed about their perception of usability across three programming environments: Dev-C++, NetBeans, and Eclipse.

A total of 143 responses were obtained. Not all respondents responded to all three programming environments; the distribution of responses is as follows: Dev-C++: 132; NetBeans: 102; and Eclipse: 112.

The age distribution shows that the majority of participants are men, a common occurrence in computer engineering programs, 13 participants were women, representing 9%; 118 men, representing 83%; and 12 participants of other genders, representing 8%.

Regarding age group, the majority of participants were between 20 and 25 years old, totaling 126 participants, or 88%. Those over 25 were 17, representing 12% of the total. The average age was 23,5 years.

In Table 3, the data show that the highest frequency value (mode) for Dev-C++ is 62,5, while for NetBeans and Eclipse it is 50. This indicates that Dev-C++ has a higher frequency value than NetBeans and Eclipse.

The mean indicates that the best average was obtained by NetBeans (60,64), followed by Dev-C++ (58,58), and finally, the lowest average was achieved by Eclipse (57,61).

The median indicates the central value of the data; in this case, NetBeans has the highest median, 61,25 points. Meanwhile, for Dev-C++ and Eclipse, at least half of the data are lower than the obtained values of 60 and 57,5 points, respectively.

Regarding standard deviation, the lowest data dispersion is Dev-C++ with 11.99 points, suggesting fewer variable results, followed by NetBeans with 12,64 points, and finally Eclipse with the most varied responses at 13,05 points.

Finally, NetBeans shows better means and medians, albeit with moderate variability. Dev-C++ is more consistent, with less dispersion, although its mean is slightly lower than NetBeans. Eclipse shows more variability and a lower mean score.

Considering the SUS level scale [30], all the programming environments evaluated would be in grade D, with scores between 51,7 and 62,6, which are relatively low. This means that they are far from meeting the expected usability requirement of 68 points.

Table 3. Results of system usablity scale.

	Dev-C++	NetBeans	Eclipse
Mode	62.5	50	50
Mean	58.58	60.64	57.61
Median	60	61.25	57.5
Standard deviation	11.99	12.64	13.05

The Shapiro-Wilk test (a more reliable test for small samples) was used to determine whether the results obtained from the three IDEs follow a normal distribution.

It is defined:

Null Hypothesis (H_0): The data have a normal distribution.

Alternative Hypothesis (H_1): The data do not have a normal distribution.

It was found that all of them had p-values > 0,05, with Dev-C++ (p-value = 0,106), NetBeans (p-value = 0,111), and Eclipse (p-value = 0,165). This indicates that the results from all three IDEs follow a normal distribution, thus confirming that the null hypothesis is fulfilled (Table 4)

Table 4. Shapiro-Wilk test.

	Shapiro-Wilk		
	Statistic	df	Sig.
Dev-C++	0,979	102	0,106
NetBeans	0,979	102	0,111
Eclipse	0,982	102	0,165

The one-way ANOVA test was then used to determine whether there were significant differences between the groups. The following analyses were performed to conduct the test.

Levene's test was performed, establishing:

Null hypothesis (H_0): the variances are equal across groups.

Alternative hypothesis (H_1): the variances are not equal across groups.

The results shown in Table 5 indicate that equality of variances (H_0) is confirmed across groups (p-value = 0,807).

The ANOVA test was then performed, establishing the following:

The null hypothesis (H_0) establishes that there are no significant differences between the group means. The alternative hypothesis (H_1) states that at least two of the group means are different.

The results shown in Table 6 indicate that H_0 is true (p-value = 0,201), meaning there are no significant differences between the means of the groups analyzed.

Table 5. Test of homogeneity of variances.

SUS			
Levene Statistic	df1	df2	Sig.
0.215	2	343	0.807

Table 6. One-Way ANOVA.

SUS					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	507.057	2	253.528	1.613	0,201
Within Groups	53915.848	343	157.189		
Total	54422.905	345			

An analysis of each item shows that scores are generally low (Table 7), with a target score of 68. However, Item 1 (I think I would like to use this system frequently) and Item 5 (I found the various features of this system to be well-integrated) achieved high scores, with a target score of 68.

For Item 1, Eclipse achieves a score of 3,69, which is a low score, with a target score of 80, which is higher (with greater perceived usability). This same situation occurs for Item 5 for NetBeans, with a score of 3.92. This means that NetBeans integration is perceived as very well-integrated.

Furthermore, the lowest scores are observed in items 4 and 10, which are precisely those related to the learning dimension [31]. Therefore, according to the results, the three environments would show a low perception in this area.

Table 7. Results of system usablity scale.

Item SUS	Dev-C++	Eclipse	NetBeans
I1	3,03	3,69	3,51
I2	2,53	2,79	2,57
I3	3,57	3,36	3,31
I4	2,64	2,89	2,94
I5	3,48	3,75	3,92
I6	2,57	2,59	2,33
I7	3,26	3,11	3,42
I8	2,58	2,74	2,47
I9	3,48	3,52	3,56
I10	3,08	3,36	3,16

Regarding the experts' responses, we got answers from seven experts, as follows:

1.- What programming environments do you frequently use? The most frequently mentioned programming environments were NetBeans (five times), Eclipse (four times), and Visual Studio (four times). These results converge with those obtained in the evaluation of the SUS questionnaire, where NetBeans obtained the highest usability score among the three evaluated. Visual Studio is worth considering for future evaluations because it is frequently mentioned. Figure 1 shows the cloud of words mentioned by the experts.

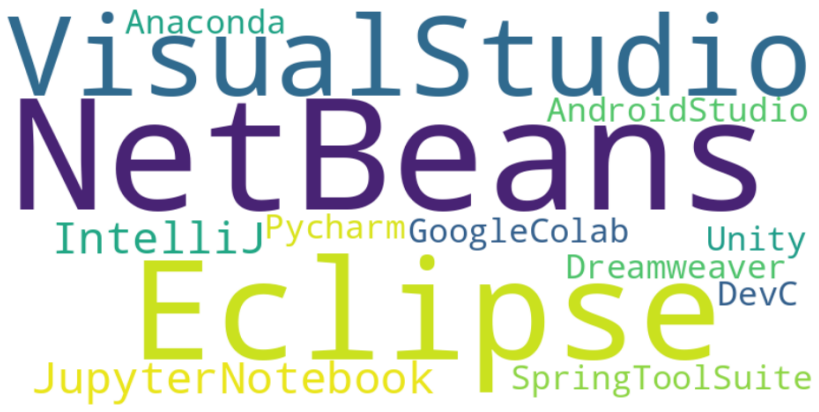

Figure 1. Programmer environments mentioned by experts.

2.- If you had to recommend a programming environment, which one would it be and why? The recommended programming environments were Eclipse, Dreamweaver, IntelliJ, NetBeans, and Visual Studio, primarily due to their simplicity and varied functionality.

Expert: "*Eclipse for its ease of learning and the fact that it doesn't generate code for the tool itself, but for the developer.*"

Expert: "*Dreamweaver, for its dynamism and simplicity for working with local servers.*"

Expert: "*IntelliJ because it offers so many features and is easy to use.*"

Expert: "*Apache NetBeans. It's quite versatile.*"

Expert: "*Visual Studio, I would recommend it because it's one of the few I still use from time to time. It's also compatible with many programming languages and has an exquisite interface.*"

3.- Indicate one or more positive aspects of the programming environments you have used. The most frequently mentioned favorable aspects are “ease of learning” (cited by three experts), “variety of functionalities” (mentioned by three experts), “usability” (mentioned by one expert), and “the use of few computing resources” (cited by one expert). Expert comment: "*Multiple features, easy to use, and not too resource-intensive.*"

4.- What would you like to improve about the programming environments? Among these responses, they highlight what they would like to improve in programming environments: that they be cross-platform, offering the same options regardless of the operating system; from a usability perspective, that error messages be more explicit, which points to ease of use, as explained in [32]; and that they be easy to use with more intuitive icons and interfaces, like authors consider in the heuristics for programming environments [33]. Resource optimization was also mentioned, as certain features consume significant machine resources. Among the comments, the inclusion of artificial intelligence, which is widely used today, stands out. An expert: "*A very good improvement would be to natively implement AI in programming environments to develop more robust systems, mitigate errors, reduce lines of code, and optimize execution times.*" This turns out to be a differentiating element from what the authors analyzed in similar previous studies [28].

CONCLUSION

Considering the importance of usability in systems and the difficulty of programming code, this work conducted a usability perception survey using the System Usability Scale (SUS) across three programming environments commonly used in higher education. A total of 143 undergraduate students from computer engineering programs participated in the survey. The results show that NetBeans was the most highly rated; however, the three IDEs evaluated have many areas for improvement, so their scores were low, with a minimum average of 68, considered usable. Furthermore, when analyzing the results by item, it was found that the integration of NetBeans and Eclipse functions was well received, and the intention to use them was high. The low scores per item are consistent with the learnability aspect, with the three IDEs having the weakest results

in items 4 and 10. Using one-way ANOVA, no statistically significant difference was found among the three IDEs, indicating that the groups are homogeneous and that there is no real difference in user-perceived usability between the analyzed IDEs.

Responses from seven experts, who determined that the most widely used IDEs are NetBeans, Eclipse, and Visual Studio, were received. On the one hand, their recommendations for programming environments and their positive aspects relate to ease of use, simplicity, and feature variety. On the other hand, the factors they would most like to improve in a programming environment include usability related to error messages, configurable interfaces, and icon recognition, as well as a new element: the incorporation of AI-powered code.

Some limitations of this work include that the IDEs analyzed are used with different approaches and scopes. However, in this case, we are examining the programmer's perception of usability in the specific context in which it is used, rather than in every context. Furthermore, it is necessary to increase the sample size to obtain results that are more representative and generalizable.

In future work, it is expected to evaluate Visual Studio Code as one of the most frequently mentioned environments and to analyze the usability of code generation using AI assistants.

Funding

The authors declare that they received no funding for this work.

REFERENCIAS

[1] V. Ramalingam, D. LaBelle, and S. Wiedenbeck, "Self-efficacy and mental models in learning to program, " in *ACM SIGCSE Bulletin*, vol. 36, no. 3, pp. 171-175, 2004, https://doi.org/10.1145/1007996.1008042 [↑]

[2] C. Bravo, R. Duque, and J. Gallardo, "A groupware system to support collaborative programming: Design and experiences, " *Journal of Systems and Software*, vol. 86, no. 7, pp. 1759-1771, 2013, https://doi.org/10.1016/j.jss.2012.08.039 [↑]

[3] K. Chawla, M. Chiou, A. Sandes, and P. Blikstein, "Dr. Wagon: a 'stretchable' toolkit for tangible computer programming, " in *Proceedings of the 12th International Conference on Interaction Design and Children*, Jun. 2013, pp. 561-564, https://doi.org/10.1145/2485760.2485865 [↑]

[4] J. Good and K. Howland, "Natural language and programming: Designing effective environments for novices, " in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Atlanta, GA, USA, Oct. 2015, pp. 225-233, https://doi.org/10.1109/VLHCC.2015.7357221 [↑]

[5] M. Harward, W. Irwin, and N. Churcher, "In situ software visualization, " in *2010 21st Australian Software Engineering Conference*, Auckland, New Zealand, Apr. 2010, pp. 171-180, https://doi.org/10.1109/ASWEC.2010.18 [↑]

[6] M. R. Jakobsen and K. Hornbæk, "Transient or permanent fisheye views: A comparative evaluation of source code interfaces, " *Information Visualization*, vol. 11, no. 2, pp.151-167, 2012,

https://doi.org/10.1177/1473871611405643 [↑]

[7] J. Smith, C. Brown, and E. Murphy-Hill, “Flower: Navigating program flow in the IDE, ” in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Raleigh, NC, USA, Oct. 2017, pp. 19-23, https://doi.org/10.1109/VLHCC.2017.8103445 [↑]

[8] T. Karrer, J.P. Krämer, J. Diehl, B. Hartmann, and J. Borchers, “Stacksplorer: call graph navigation helps increasing code maintenance efficiency, ” in *Proceedings of the 24th Annual ACM symposium on User Interface Software and Technology*, Oct. 2011, pp. 217-224, https://doi.org/10.1145/2047196.2047225 [↑]

[9] E. Murphy-Hill and A. P. Black, “Programmer-friendly refactoring errors, ” *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1417-1431, 2012, https://doi.org/10.1109/TSE.2011.110 [↑]

[10] B. A. Becker, “*An exploration of the effects of enhanced compiler error messages for computer programming novices*, ” M.S. Dissertation, Dublin Institute of Technology, Dublin, Ireland, 2015. [↑]

[11] T. Barik, K. Lubick, S. Christie, and E. Murphy-Hill, “How developers visualize compiler messages: A foundational approach to notification construction, ” in *2014 Second IEEE Working Conference on Software Visualization*, Victoria, BC, Canada Sep. 2014, pp. 87-96, https://doi.org/10.1109/VISSOFT.2014.24 [↑]

[12] E. McArdle, J. Holdsworth, and I. Lee, “Assessing the usability of students object-oriented language with first-year IT students: A case study, ” in *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, Adelaide, Australia, Nov. 2013, pp. 181-188, https://doi.org/10.1145/2541016.2541036 [↑]

[13] M. Ramoğlu, Ç. Genç, and K. Rızvanoğlu, “Programming a robotic toy with a block coding application: A usability study with non-programmer adults, ” in International Conference of Design, User Experience, and Usability: Theory, Methodology, and Management (DUXU 2017). Lecture Notes in Computer Science, A. Marcus and W. Wang *, Eds.* vol. 10288, Jul. 2017, pp. 652-666, https://doi.org/10.1007/978-3-319-58634-2_47 [↑]

[14] International Standard Organization, *Ergonomics of human-system interaction-Part 11: Usability: Definitions and concepts*, International Standardization Organization (ISO). ISO 9241-11, 2018. [Online.] Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en [↑]

[15] J. Nielsen, *Usability Engineering*, San Francisco, USA: Morgan Kaufmann, 1993. [↑]

[16] International Standard Organization, Ergonomics of human system interaction-Part 210: Human-centred design for interactive system, ISO 9241-210. International Standardization Organization (ISO). ISO 9241-210, 2010. [Online.] Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-1:v1:en [↑]

[17] P. Morville, “User experience honeycomb, ” semanticstudios.com. Accessed: Mar. 12, 2026 [Online.] Available: http://semanticstudios.com/user_experience_design/ [↑]

[18] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, “Unhappy developers: Bad for themselves, bad for process, and bad for software product, ” in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Buenos Aires, Argentina May. 2017, pp. 362-364, https://doi.org/10.1109/ICSE-C.2017.104 [↑]

[19] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, “On the unhappiness of software developers, ” in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, Jun. 2017, pp. 324-333, https://doi.org/10.1145/3084226.3084242 [↑]

[20] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, “What happens when software developers are (un) happy, ” *Journal of Systems and Software*, vol. 140, pp. 32-47, 2017, https://doi.org/10.1016/j.jss.2018.02.041 [↑]

[21] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, “Consequences of unhappiness while developing software, ” in *Proceedings of the 2nd International Workshop on Emotion Awareness in Software Engineering*, Buenos Aires, Argentina, May. 2017, pp. 42-47, https://doi.org/10.1109/SEmotion.2017.5 [↑]

[22] S. Astromskis, G. Bavota, A. Janes, B. Russo, and M. Di Penta, “Patterns of developers behaviour: A 1000-hour industrial study, ” *Journal of Systems and Software*, vol. 132, pp. 85-97, 2017, https://doi.org/10.1016/j.jss.2017.06.072 [↑]

[23] M. Vakilian, N. Chen, R. Zilouchian Moghaddam, S. Negara, and R. E. Johnson, “A compositional paradigm of automating refactorings, ” in *European Conference on Object-Oriented Programming(ECOOP 2013). Lecture Notes in Computer Science*, G. Castagna, Ed. vol. 7920, Jul. 2013, pp. 527-551, https://doi.org/10.1007/978-3-642-39038-8_22 [↑]

[24] K. Żyła, K. Chwaleba, and D. Choma, “Evaluating usability and accessibility of visual programming tools for novice programmers—the case of app inventor, Scratch, and StarLogo, ” *Applied Sciences*, vol. 14, no. 21, Art. no. 9887, pp. 9887-9902, 2024, https://doi.org/10.3390/app14219887 [↑]

[25] J. Morales and C. Rusu, “Usability perception of visual programming language: A case study, ” in *Proceedings of the VI Iberoamerican Conference of Computer Human Interaction*. vol. 2747, Arequipa, Perú, Sep. 16-18, 2020, pp. 83-88. [↑]

[26] N. Ritschel, F. Fronchetti, R. Holmes, R. Garcia, and D. C. Shepherd, “Blocks? graphs? Why not both? Designing and evaluating a hybrid programming environment for end-users, ” in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, Apr. 2024, pp. 326-327, https://doi.org/10.1145/3639478.3643101 [↑]

[27] M. Jönsson and M. Andersson, “How do students perceive usability and usefulness of the Eclipse Integrated Development Environment? A survey study on students’ perspectives at Lund University, ” *Bachelor Thesis, School of Economics and Management, Department of Informatics, Lund University*, Lund, Sweden. [Online.] Available: https://lup.lub.lu.se/student-papers/search/publication/9119308 [↑]

[28] J. Morales, F. Botella, C. Rusu, and D. Quiñones, “How “friendly” integrated development environments are?, ” in *Social Computing and Social Media. Design, Human Behavior and Analytics (HCII 2019)*. Lecture Notes in Computer Science, G. Meiselwitz, Ed. vol. 11578, Jun. 2019, pp. 80-91, https://doi.org/10.1007/978-3-030-21902-4_7 [↑]

[29] J. Brooke, “SUS: A ‘quick and dirty’ usability scale, ” in *Usability Evaluation in Industry*, London, UK: CRC Press, 1996, pp. 207–212, https://doi.org/10.1201/9781498710411-35 [↑]

[30] J. R. Lewis and J. Sauro, “Item benchmarks for the system usability scale, ” *Journal of Usability Studies*, vol 13, no. 3, pp. 158-167, 2018. [↑]

[31] J. R. Lewis and J. Sauro, “The factor structure of the system usability scale, ” in *Human Centered Design: First International Conference (HCD 2009). Held as Part of HCI International.*, M. Kurosu, Ed. vol. 5619, San Diego, CA, USA, July 19-24, 2009, pp. 94-103, https://doi.org/10.1007/978-3-642-02806-9_12 [↑]

[32] J. Morales, C. Rusu, F. Botella, and D. Quiñones, “Programmer eXperience: A systematic literature review, ” *IEEE Access*, vol. 7, pp. 71079-71094, 2019, https://doi.org/10.1109/ACCESS.2019.2920124 [↑]

[33] J. Morales, C. Rusu, F. Botella, and D. Quiñones, “Programmer experience: a set of heuristics for programming environments, ” in *Social Computing and Social Media. Participation, User Experience, Consumer Experience, and Applications of Social Computing*, Gabriele Meiselwitz, Ed. Cham, Switzerland: Springer, 2020, pp. 205-216, https://doi.org/10.1007/978-3-030-49576-3_15 [↑]