Uncovering and Exploring the Intersection of Low-Code Development and Sustainability

Bernhard Schenkenfelder

Software Competence Center Hagenberg GmbH, Softwarepark 32a, 4232 Hagenberg, Austria

Abstract

Sustainability has become a cross-disciplinary research topic in the face of pressing, complex global challenges. DIY (Do-It-Yourself) Software Development, End-User Development (EUD), and Low-Code Development (LCD) are empowering non-programmers to design, develop, and maintain software. These paradigms use techniques such as visual programming or natural language programming to abstract beyond text-based program code. Beginning with a case study that draws a connection between DIY software development and sustainability through digital transformation, this position paper examines some of the broader sustainability implications in the areas of both DIY and software development in general, and argues whether or not they are applicable or adaptable to LCD. The preliminary conclusion is that, despite some challenges, LCD can draw from both fields, and that LCD can be used as a lever to promote sustainability.

Keywords

Low-Code Development (LCD), End-User Development (EUD), DIY (Do-It-Yourself) Software Development, Sustainability

1. Introduction

In a recent case study on the digital transformation of the global energy company Shell, Carroll and Maher [1] discuss the notion of **DIY** (Do-It-Yourself) **Software Development**[1]. They report a common argument in the End-User Development (EUD) community: A massive global shortage of skilled software developers, followed by giving business people more tools to increase their productivity and the development of a shadow IT that introduces security risks. Citizen development mitigates these concerns through ongoing IT support, but still hides the complexity of the program code while empowering domain experts (citizen developers) to design, develop, and deploy software into production. Guided by its mission to "empower every employee to digitize work processes to improve productivity, increase agility, and create more value for customers," Shell went through four phases to successfully normalize the citizen development program: (i) Sensemaking for Shell's citizen development program, (ii) stakeholder participation to build momentum for the DIY program, (iii) collective action to embrace and enact the DIY program, and (iv) evaluating progress to review the impact of the DIY program. Carroll and Maher conclude the case study with seven recommendations and requirements for a successful DIY software development program, including being highly user-centric, introducing an element of fun, leveraging democratization, growing the DIY community, prioritizing necessary changes, and empowering employees. Interestingly, in the context of DIY software development, they see the need to upskill employees in different technologies and data, which at first glance is not consistent with other reports that focus on lowering the barrier to entry into software development for non-programmers. However, this argument is reasonable given that citizen developers (need to) learn and use new tools once they are part of the DIY program.

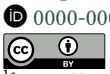
Notably, in this case study, there is a link between sustainability and DIY software development. Shell recognizes that digitization "will be key for decarbonizing the energy supply," and that the company's

Joint Proceedings of IS-EUD 2025: 10th International Symposium on End-User Development, 16-18 June 2025, Munich, Germany.

Bernhard.Schenkenfelder@scch.at (B. Schenkenfelder)

https://www.scch.at/team/bernhard.schenkenfelder (B. Schenkenfelder)

0000-0001-5129-6268 (B. Schenkenfelder)



[1] https://www.shell.com/what-we-do/digitalisation/digitalisation-in-action/do-it-yourself-software-development-the-power-is-in-your-hand.html, last accessed: 2025-04-01

citizen development program plays an important role in digital transformation. **This raises the question of whether there are other implications between sustainability and DIY software development (or EUD/LCD, see below), where to uncover them, and how to explore them.**

In an attempt to explore the similarities and differences between **End-User Development** (EUD) and **Low-Code Development** (LCD), Schenkenfelder et al. [2] point out that while there are–sometimes subtle–differences in the scientific communities in which they are rooted, the user groups supported, the techniques used, the scope of tools and infrastructure, and the application domains, these are outweighed by similarities[2] in that both approaches empower non-programmers to create their own software using tools with a low barrier to entry.

The above descriptions confirm a certain similarity or overlap between the concepts of EUD, LCD, and DIY software development. And since the term Low-Code is used in the industry-academia collaborations in which the author is involved, he will also use it in the context of this paper, recognizing that for practical work in such collaborations, both EUD, LCD, and DIY software development, either alone or in combination, can help achieve the goal of empowering non-programmers.

This position paper aims to uncover and explore the intersection between LCD and sustainability by reviewing relevant areas and how they relate to sustainability in general, and then to argue whether or not these implications can be applied or adapted to LCD. The remainder of this paper is organized as follows. Related areas and their implications are presented in Section 2 (DIY) and Section 3 (Software Development). Section 4 discusses the results, namely whether and how these implications apply to LCD, followed by conclusions and an outline of future work (Section 5). And since this position paper is intended as a contribution to an interactive and engaging session to further elaborate on the topic, some of the implications may require more imagination to follow, and others less.

2. DIY and Sustainability

The area of DIY in general is considered relevant because of the notion of DIY software development described in Section 1. The idea is to uncover and explore sustainability implications from this area that also apply to the more specific concept of DIY software development. The author is not a DIY expert, nor does he have any knowledge of the relevant scientific communities, conferences, or journals. An LLM prompt for the five most frequently cited conference papers or journal articles on DIY and sustainability returned only a list of hypothetical references (see Appendix A.1). A search was finally conducted on Google Scholar and ResearchGate, using the keywords *DIY, do-it-yourself, sustainability*, and combinations thereof.

In 15 interviews with DIY practitioners, Salvia [4] seeks to explore how the product design profession can have a positive environmental impact through DIY practices. Salvia describes DIY as a "window of opportunity" for both sustainable consumption and production, especially in terms of saving and extending the life of products through reuse, repair, repurposing, and reappropriation. Interestingly, respondents were categorized as doers, adapters, makers, and creators, depending on their level of interest and motivation. While the doers are primarily motivated by the need to solve problems, the creators, on the other hand, are driven by their passion and desire to innovate. Even though DIY is generally perceived as environmentally friendly, this could be undermined by the higher cost per item or the potential for error compared to mass production. Salvia also recognizes the contribution of designers in potentially expanding the range of DIY practitioners and classifies them along two axes: Designers can play the role of a facilitator or collaborator, on a global or local scale.

Bafarasat and Oliveira [5] examine the role of DIY vs. SIY (Start-It-Yourself) urbanism for social sustainability, which is part of sustainable development along with environmental and economic sustainability. In this context, both DIY and SIY are bottom-up development paths, and in detail, DIY

[2]In the discussion following the presentation of the paper, the workshop participants suggested an even greater overlap between End-User Programming (EUP) and LCD. According to Barricelli et al. [3], EUP is more narrowly defined than EUD and is about empowering end users to write programs using special-purpose programming languages, including programming by demonstration and visual programming. On the other hand, EUD also focuses on the adaptation of systems during their actual use, thus covering the entire software life cycle.

refers to the fight of communities against poverty and poverty-generating mechanisms without the permission and resources of public institutions, while SIY aims to create pressure on these institutions after starting actions. They argue that both approaches have benefits and limitations, and propose two different entities with checks and balances: The development entity meets the needs of the community, and the organizing entity ensures that it serves the community and does not act as a business entity, which requires a mature political culture.

In his review of the literature and actual developments, Hoftijzer [6] notes that DIY, which he sees as a convergence of production and consumption, has been made possible by technological developments that now empower end users to design and develop products. He emphasizes the positive nature of user design as a sustainable contrast to the use and disposal of consumer products. However, Hoftijzer also points out that the manufacturing technologies for unique user products are not yet mature in terms of their use of natural resources and the potential abundance of product iterations.

De Waal and Smal [7] present DIY as a pathway to sustainability through the case study of a Johannesburg-based clothing label. Their framework of DIY motivators is based on two main categories, namely "Product and Service Evaluation" and "Identity Enhancement." The first category, including customization, diversity, and quality of a product as well as economic benefits, deals with the production itself, while the second, including empowerment and fulfillment by creating, belonging to a community, and uniqueness in identity, deals with the DIYers themselves, and how they enhance their identity. Expanding on the idea of DIY, they propose a new approach called DI4Y^2 (Doing-It-For-Yourself-And-Others), which not only meets the needs of the individual, but also contributes to "the greater good of the environment and all living beings," further promoting sustainability.

3. Software Development and Sustainability

The literature reviewed in this section is part of an ongoing research initiative[3] focused on how software engineering practice addresses sustainability, with the goal of identifying influencing factors, deficiencies, and best practices. These papers are considered relevant because LCD is a subset of software development. This allows the author to speculate on whether or not some of the general ideas are applicable or adaptable to LCD.

Paech et al. [8] recognize that sustainability is becoming an increasingly relevant aspect of software development, but identifying sustainability effects during requirements engineering is still a challenging task. In an exploratory study, they use Generative Pretrained Transformers (GPTs) to automatically identify sustainability effects and conclude that GPT-4o is "capable of generating relevant sustainability effects." In addition, these effects can be generated in seconds, but at the cost of high power consumption. These generated effects were found to be valid and plausible, hallucinations were not observed. Limitations in the use of GPT-4o are its non-determinism and the inability to rank effects by relevance.

With the adoption of the Corporate Sustainability Reporting Directive (CSRD)[4], the EU requires companies to report on what they consider to be the risks and opportunities associated with social and environmental issues, as well as the impact of their activities in these areas. Mazak-Huemer et al. [9] present SustainScrum to systematically address sustainability throughout the development process. Specifically, SustainScrum is a customized Scrum process that considers sustainability in the backlog, user stories, and development phases, and provides quantitative sustainability indicators, including a Sustainability Sprint KPI and a Sustainability Product KPI report. Interestingly, when evaluating SustainScrum with open-source data, they found a number of challenges that need further attention. While answering sustainability-related questions (SusAF, see below) is time-consuming in itself, it is often necessary to train the development team to understand sustainability in the first place, including EU regulations. Automating this sustainability assessment is another complex and laborious task.

[3] https://se.jku.at/sustainability-in-software-engineering/, last accessed: 2025-04-01

[4] https://finance.ec.europa.eu/capital-markets-union-and-financial-markets/company-reporting-and-auditing/company-reporting/corporate-sustainability-reporting_en, last accessed: 2025-04-01

Bambazek and Groher [10] report on the integration of the Sustainability Awareness Framework (SusAF) into an undergraduate software engineering course and how the students were able to identify sustainability effects. The SusAF is a guideline with questions to identify potential sustainability effects with social, individual, environmental, economic, and technical dimensions. In addition, each identified effect is categorized as immediate, enabling, or structural to indicate its magnitude. The authors observed that all teams found potential sustainability effects, most of which were positive, resulting in a variety of issues and illustrating the impact of software on an organization's sustainability goals. Bambazek and Groher also pointed out that the students did this without any prior theoretical or practical knowledge of SusAF.

Groher and Weinreich [11] conducted an interview study with 10 software project team leaders to understand how sustainability is perceived by practitioners. Specifically, the participants were asked about their definition of sustainability in software development and its importance, potential influencing factors, current problems, and measures to improve sustainability. They found that technical and organizational sustainability issues are more important than economic ones, while environmental aspects are not considered at all. They identified business goals, requirements, technologies, and people as influencing factors. In terms of shortcomings such as lack of documentation and knowledge sharing, the identified sustainability strategies are very general and difficult to distinguish from general quality assurance measures.

4. Results and Discussion

Low-Code Development Platforms (LCDPs), which provide the tools to build software using the Low-Code approach, offer a low barrier to entry for reuse and repair compared to traditional software development, and therefore could be a great lever to extend the life of software while addressing other aspects of sustainability. Categorizing Low-Code users by motivation and designing systems accordingly to support a sustainable balance between problem-solving capabilities (doers) and facilitating passion (creators) could be another way to promote sustainability [4].

The idea of DIY vs. SIY also applies to LCD [5]. LCD initiatives that do not evolve into shadow IT, but continue to receive IT support, avoid long-term side effects that undermine their original purpose [1]. Only by carefully designing LCD initiatives can a sustainable path be found between IT involvement (organizing entity) and end-user flexibility (development entity). And when you do, LCD can deliver on its promises. DI4Y^2 [7] is another small but significant extension to the DIY approach. LCD literature is typically concerned with empowering individual non-programmers. By implementing the DI4Y^2 approach, the (positive) sustainability effects could be multiplied by the number of users. Different users may have different sustainability effects, using AI to create adaptive systems would further maximize the sustainability effects by addressing users individually within the DI4Y^2 approach.

Again, using AI to automate certain aspects of sustainability [8] requires careful consideration due to its high power consumption. However, incorporating such an AI capability into the design of an LCDP would only consume energy once, rather than several times when building software from scratch. On the other hand, GPTs have remarkable capabilities that could also help improve sustainability. Whether automated or manual, systematically addressing and capturing sustainability helps identify its effects [9, 10, 11]. With regard to SustainScrum, it remains unclear whether this process model can be applied directly to LCD or whether it needs to be adapted. However, applying a development process to LCD may be contradictory in the first place, since Low-Code promises a shallow learning curve.

It is interesting to note that papers from both areas discuss the fact that sustainability can be associated with certain pitfalls that undermine its original intention, such as higher cost per item [4, 6] or the administrative burden it imposes [9]. Since the initial cost of building an LCDP is generally higher than that of building custom software from scratch, careful consideration is required to ensure that the platform is sustainable in the sense that many software instances will be derived from it.

5. Conclusions and Future Work

This position paper first reflects on a case study of DIY (Do-It-Yourself) software development and how it relates to sustainability through digital transformation. The author then argues that DIY software development, EUD and LCD have a large overlap. Having established this common ground, the paper continues to uncover and explore whether there are other or more detailed sustainability implications than digital transformation. This is done by examining the related fields of DIY and software development in general and how they address sustainability. Finally, the author speculates on whether or not these general ideas and approaches can be applied or adapted to LCD, finding that LCD can draw from both fields and, more importantly, that LCD can be used to leverage sustainability.

Immediate future work aims to confirm these preliminary findings. First, by aligning the measures found in the literature and the ideas presented above with the United Nations Sustainable Development Goals, a more comprehensive and generalizable baseline could be established (see Appendix A.2, for example). Then, a systematic extension of the literature reviewed would provide more insights and a broader range of answers to the original question. Finally, an empirical validation of the claims made with respect to the intersection of sustainability and LCD through their use in actual LCD projects could strengthen the existing body of knowledge.

Acknowledgments

The research reported in this paper has been partly funded by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Federal Ministry for Labour and Economy (BMAW), and the State of Upper Austria in the frame of the SCCH competence center INTEGRATE (FFG grant no. 892418) in the COMET - Competence Centers for Excellent Technologies Programme managed by Austrian Research Promotion Agency FFG.

Declaration on Generative AI

During the preparation of this work, the author used DeepL Write and DeepL Translate in order to: Text Translation, Improve writing style, and Grammar and spelling check. Further, the author used Llama 3.3 70B (hosted in-house) in order to: Generate literature review (results discarded, see Appendix A.1) and Drafting content (see Appendix A.2). After using these tools/services, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] N. Carroll, M. Maher, How shell fueled digital transformation by establishing diy software development, MIS Quarterly Executive (2023) 99–127. doi:10.17705/2msqe.00076.

[2] B. Schenkenfelder, U. Brandstätter, H. Kirchtag, M. Wimmer, Low-code development and end-user development: (how) are they different?, in: D. Tetteroo, S. Feger, D. Fogli, A. Mørch, A. Piccinno, M. U. Modén (Eds.), Proceedings of the First International Workshop on Participatory Design & End-User Development - Building Bridges co-located with the International Nordic Conference on Human-Computer Interaction (NordiCHI 2024), number 3778 in CEUR Workshop Proceedings, Aachen, 2024. URL: https://ceur-ws.org/Vol-3778/short5.pdf.

[3] B. R. Barricelli, F. Cassano, D. Fogli, A. Piccinno, End-user development, end-user programming and end-user software engineering: A systematic mapping study, Journal of Systems and Software 149 (2019) 101–137. doi:10.1016/j.jss.2018.11.041.

[4] G. Salvia, The satisfactory and (possibly) sustainable practice of do-it-yourself: the catalyst role of design, J. of Design Research 14 (2016) 22. doi:10.1504/JDR.2016.074782.

[5] A. Ziafati Bafarasat, E. Oliveira, Social sustainability: Do-it-yourself urbanism, start-it-yourself urbanism, Geoforum 141 (2023) 103726. doi:10.1016/j.geoforum.2023.103726.

[6] J. Hoftijzer, Sustainability by do-it-yourself product design: User design opposing mass consumption, in: P. Israsena, J. Tangsantikul, D. Durling (Eds.), Research: Uncertainty Contradiction Value - DRS International Conference 2012, Bangkok, Thailand, 2012. URL: https://dl.designresearchsociety.org/drs-conference-papers/drs2012/researchpapers/52.

[7] D. De Waal, D. Smal, Re-thinking do-it-yourself: An ontological approach to sustainability in fashion design praxis, DIY, Alternative Cultures & Society 2 (2024) 191–206. doi:10.1177/27538702241238775.

[8] B. Paech, P. Kaiser, P. Bambazek, I. Groher, N. Seyff, Exploring generative pretrained transformers to support sustainability effect identification - a research preview, in: A. Hess, A. Susi (Eds.), Requirements Engineering: Foundation for Software Quality, Springer Nature Switzerland, Cham, 2025, pp. 226–234.

[9] A. Mazak-Huemer, M. Vierhauser, I. Groher, Sustainscrum: integrating sustainability assessment in a tailored scrum process for computing quantitative sustainability indicators, Software and Systems Modeling (2025). URL: https://link.springer.com/10.1007/s10270-025-01266-5. doi:10.1007/s10270-025-01266-5.

[10] P. Bambazek, I. Groher, Integrating the sustainability awareness framework in undergraduate software engineering education, in: D. Mendez, A. Moreira, J. Horkoff, T. Weyer, M. Daneva, M. Unterkalmsteiner, S. Bühne, J. Hehn, B. Penzenstadler, N. Condori-Fernández, O. Dieste, R. Guizzardi, K. M. Habibullah, A. Perini, A. Susi, S. Abualhaija, C. Arora, D. Dell'Anna, A. Ferrari, S. Ghanavati, F. Dalpiaz, J. Steghöfer, A. Rachmann, J. Gulden, A. Müller, M. Beck, D. Birkmeier, A. Herrmann, P. Mennig, K. Schneider (Eds.), Joint Proceedings of REFSQ-2024 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Colocated with REFSQ 2024, number 3672 in CEUR Workshop Proceedings, Aachen, 2024. URL: https://ceur-ws.org/Vol-3672/EuT-paper1.pdf.

[11] I. Groher, R. Weinreich, An interview study on sustainability concerns in software development projects, in: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, Vienna, Austria, 2017, p. 350–358. URL: http://ieeexplore.ieee.org/document/8051370/. doi:10.1109/SEAA.2017.70.

A. LLM Prompts and Responses on DIY and Sustainability

A.1. Literature Referernces

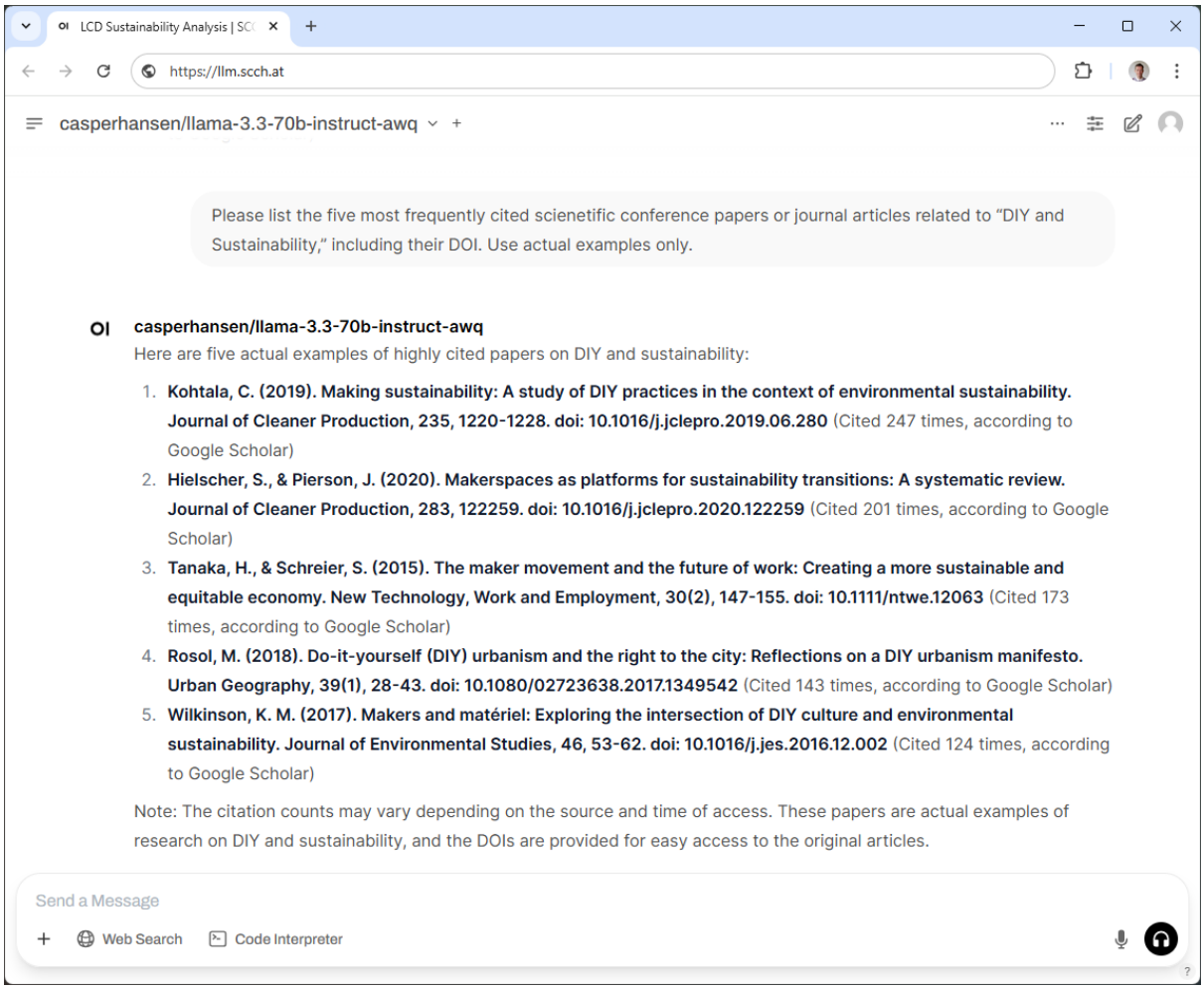


Figure 1: Llama 3.3 70B generates hypothetical examples of what a list of references on DIY and sustainability could look like. The examples are not real, and DOIs are not valid.

A.2. Low-Code Development (LCD) and Sustainability Development Goals (SDGs)

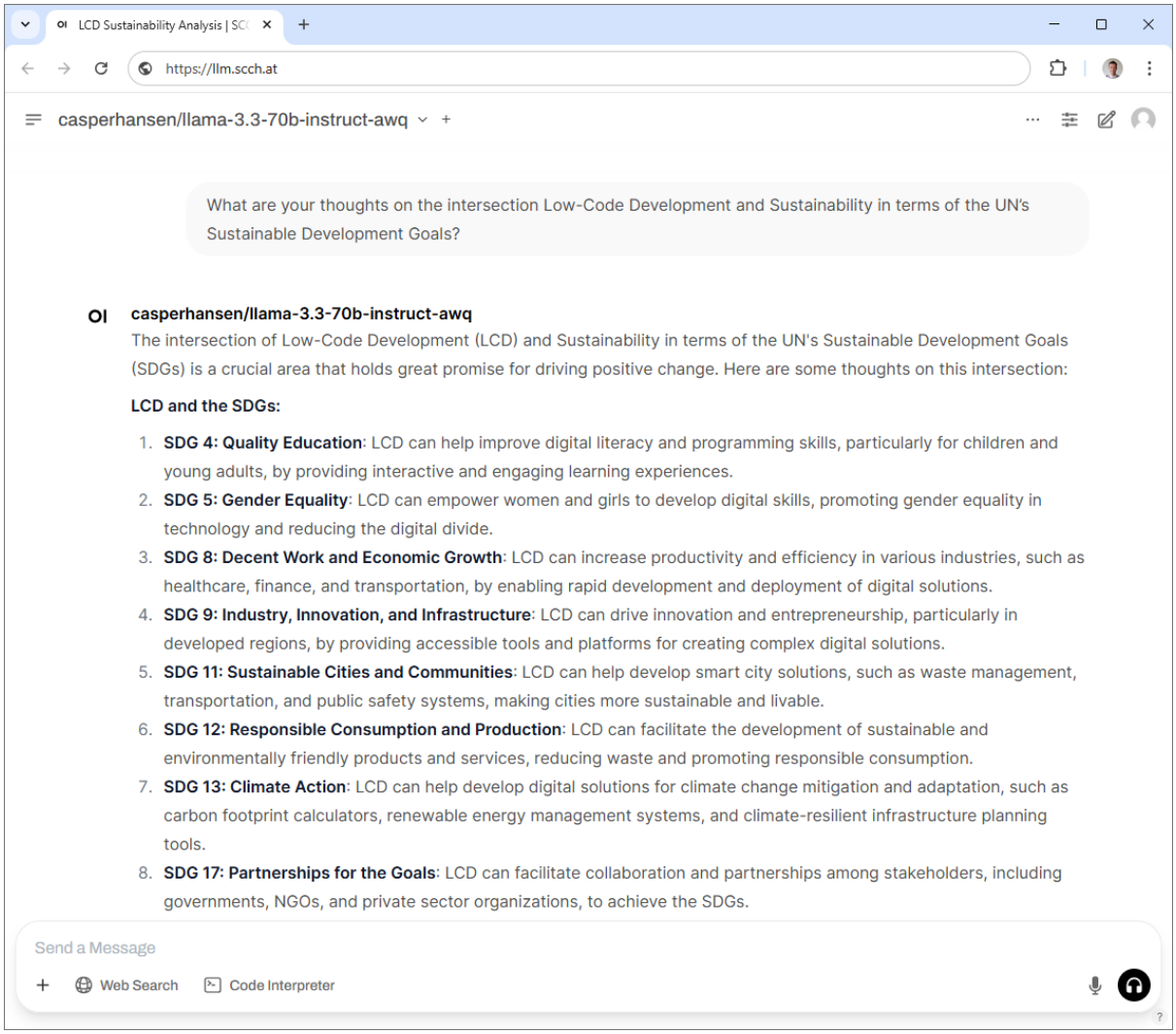


Figure 2: The response generated by Llama 3.3 70B on the intersection of LCD and the SDGs.