Digital Skills & Jobs Platform

IN BRIEF

Coding and Programming Skills in the age of Generative AI

THIS PAPER IS PART OF OUR DIGITAL BRIEF SERIES: A SET OF EASY-TO-GRASP DEEP DIVES TACKLING THE LATEST KEY TOPICS AROUND DIGITAL SKILLS AND JOBS. PRODUCED IN COLLABORATION WITH SOME OF EUROPE'S BEST EXPERTS IN THE FIELD.

The Digital Skills and Jobs Platform is funded by the Connecting Europe Facility of the European Union. Opinions expressed do not necessarily reflect the contracting authority's official position.

Table of Contents

Subtitle 3
Summary 3
Keywords 3
Introduction 4
Software Developers for the EU labour market 4
Code, code, code: some terminology and basic concepts 5
Support for the EU labour market 5
EU actions to promote programming/coding skills 6
Will GenAI improve the skills shortage in the labour market? 6
Some Context 6
Position of coding in the software engineering process 7
Proliferation of coding in knowledge-intensive jobs 8
Technological support vs our general understanding 8
Implications for skills development in coding & programming in an age of GenAI 11
Future avenues 12
Bibliography 13

Subtitle

Is Generative AI the answer to tackling the digital skills shortage in the labour market?

Summary

Generative AI, a sub-discipline of Artificial Intelligence is concerned with applications where a computer simulates the human ability to create and produce – whether this is human language, or human creative expressions (like drawing, composing music, etc.). The best example of a Generative AI that has been making waves recently is ChatGPT, a Large-Language Model (LLM) that users can interact with in natural language. One area in which Generative AI is especially promising hides in the way we write our software code. This article delves into the possibilities that Generative AI creates to support this software development process - and its potential to open avenues to write software code to more people than ever before. Together with the opportunities that open up, this paper also looks at the risks of using GenAI-supported tools with diminishing human oversight, and points to potential horizons to explore if we want to get to a more balanced use that takes into account our core EU values.

Keywords

Generative AI, ChatGPT, skills shortage, coding, programming, software development

About the Author

Dr. Kamakshi Rajagopal is an interdisciplinary researcher and freelance consultant in educational design and technology, with extensive experience in networked learning and social learning formats, supported by innovative technologies. She holds a Masters in Linguistics (2003) and Artificial Intelligence (2004) from KU Leuven (BE). She completed her doctoral research at the Open Universiteit (NL) in 2013, investigating personal learning networks and their value for continuous professional development. Her current research is on studying the complexity of learning environments and more specifically on how teachers and learners can be supported in dealing with this complexity. Dr. Rajagopal has developed multiple (nationally-funded and European) collaborative research projects in primary, secondary and higher education with partners from the public sector, industry and civil society. Some examples of her projects are about the role of teacher networks in educational innovation, thesis circles in higher education, the multimodal measurement in collaborative hybrid learning spaces, and mainstreaming Virtual Mobility at higher educational institutions. Since 2023, she has been working on Learning and Development in IT & business consultancy.

Introduction

Generative AI, a sub-discipline of Artificial Intelligence (AI), is concerned with applications where the computer simulates, or *tries to mimic*, the human ability to create and produce in human language and human creative expression – like drawing, or composing music (Stokel-Walker & Van Noorden, 2023). One example of Generative AI that has been making waves recently is ChatGPT, a Large-Language Model (LLM) that users can interact with by typing. The performance of this LLM has astonished many researchers and the general public, as to how *"real"* the interactions can be (Maslej et al, 2023) .

Interest in this form of AI has been growing for several decades already, with foundational research well-developed (Reiter & Dale, 1997; Kandhasamy & Xie, 2004). However, applications remained limited, with small proof-of-concepts algorithm primarily in research – largely due to the lack of computational power or insufficient quantities of training data, necessary to bring to life quality technology (Maslej et al, 2023). This changed in recent years. The period from 2015 to 2016 saw movements to pool resources and build systematic large data sets and models, with coordinated endeavours in industry such as OpenAI (founded in 2015) and HuggingFace (founded in 2016). Since the end of 2022, the maturity of these technologies has undergone a giant boost due to the alignment between computing power, mature algorithms, and sufficient data.

One of the areas where Generative AI is very promising is in the area of writing software code. GenAI is already being employed by many engineers to help them in their coding (Sharma, 2021). Simple human text allows them to write complex code in seconds, potentially speeding up their coding and programming (Brady, 2023). In Generative AI tools, textual instructions (prompts) can be given to instruct the AI to write code to create certain functionalities. As a result, the user of such a tool can communicate in more specific natural language prompts, without needing to know the details of the coding language. **In a limited way and with relevant prompts, Generative AI - could also generate the steps in the logic behind an application, defining the general structure of an algorithm.**

Naturally, this brings up *many questions on how we will write our software in the future*, and *what will be the skills necessary* for this. This next section of this article dives deeper into this question.

Software Developers for the EU labour market

For many years now, there has been a shortage of skilled workers in the digital industry. A study by the European Software Skills Alliance published in 2021, identified the role of *"developer"* as the most in-demand software role, and indicated the continuously changing

nature of this role as the reason behind this shortage. The key skills of developers are programming and coding, two disciplines in which the job content keeps changing, posing a problem for organisations and making it hard to keep employees' skills up-to-date. The growing need for ICT experts across sectors poses risks to the European Digital Decade target of reaching 20 million ICT specialists in Europe by 2030.

Code, code, code: some terminology and basic concepts

Some explanatory terminology is useful here. Software development refers to computer science based activities, i.e. the process of creating, designing, deploying and supporting software (IBM, 2023). The development phase in software development happens when the programmer starts defining and writing the instructions for the computer to follow, in the desired computer language.

So what is programming? Programming refers to the activity developers perform to define the logic of the computer program, including the entities the program needs to manipulate and the algorithms it needs to implement. **Programming skills are thus closely linked to logical and analytical, conceptual and abstract, as well as computational thinking**. When developers code, they write the instructions for the computer to implement certain actions that the computer needs to perform. Developers write in various computer languages, that provide them with different capabilities depending on their structure or purpose. What is more, **computer languages keep evolving, making it a complex task for developers to keep up to date with the latest styles and trends.** When we talk about skills shortages in the labour market, we can refer both to the skills related to programming, as well as to the ones related to coding.

Support for the EU labour market

Since these skills shortages have been of concern for several years, policy makers and researchers have invested much energy in developing support for labour market actors to identify and develop these skills in their (potential) employees. Some key initiatives include standardised competence frameworks specifically for the digital context. The Digital Competence Framework for citizens, DigComp 2.2, (Vuorikari, Kluzer and Punie, 2022) is the main commonly-accepted framework in the EU that identifies and structures the digital skills needed by citizens to participate in a modern digital society. The European e-Competence framework (e-CF) is a standardised reference framework of 41 competences relevant and required in an IT professional work context. The framework also identifies 30 ICT professional role profiles spanning different aspect of the digital industry landscape. In the EU, it is a flagship framework that offers a common language between IT recruiters and IT experts.

There are also more generic frameworks with relevant profiles. Sector-specific skills frameworks through the Pact for Skills bring various industry actors together in various sector-specific alliances to identify and structure the skills needed to develop their sector further. For a large part, these also include complex digital skills as the digital industry is affecting all sectors. For entrepreneurship, the EntreComp framework (McCallum, Weicht, McMullan and Price,

2018) includes competences considering effective use of resources, that also include assessment and insight into the needs of digital resources and development of digital outcomes. The changing technological context is continuously integrated into these frameworks, with its latest versions including skills on working with AI-technology. For example, DigComp in its latest edition, 2.2., now also includes knowledge of AI-related systems.

EU actions to promote programming/coding skills

Apart from providing support in the naming, identification and development of skills related to software development, policy initiatives also focus on making careers in the digital sector more popular and known. **To entice more interest in programming and coding, there are several recurring EU actions**. EU Code Week promotes grassroots initiatives to encourage more citizens to use and develop their programming skills. Working with volunteers, activities include awareness campaigns, workshops, conferences, etc. held across Europe, in local languages and catered to regional needs and contexts. In 2021, more than 4 million people in 80+ countries from all over the world took part in the initiative, with most of the activities taking place in schools (European Commission, 2023).

The European Institute of Technology and Innovation offers a platform for industry actors to come together to innovate in relevant industries in a safe and open environment. EIT Digital focusses specifically on supporting innovation and entrepreneurship in the digital industry (EIT Digital, 2020). Additionally, many EU funded programmes focus on improving the digital skills of the European workforce for the digital industry: just take a look at the massive investment behind the DIGITAL Europe Programme (€7.5 billion going towards the digital transformation of European economy and society). Another clue is the 20% threshold in digital investment in Recovery and Resilience Plans of EU Member States, which has been overshot.

Will GenAI improve the skills shortage in the labour market?

The arrival of Generative AI opens up a world of opportunities for programming and coding in the context of the labour market. On the one hand, GenAI could make more advanced skills (like coding) more accessible to people: and 2023 could prove to be watershed moment – one that sees AI boost people's coding skills, fast, allowing us to create digital tools in a more efficient and effective way. On the other hand, GenAI could also diminish proficiency in coding skills, as it reduces deep knowledge-building and expertise on this. The paragraphs below explore the context in which GenAI could have an impact in software development, and the implications it holds for skill development.

Some Context

Before exploring which skills will be relevant in this new space, it is good to consider some context of which aspects of coding can be affected by Generative AI.

Position of coding in the software engineering process

Software engineering – like other forms of engineering – is a design science, and broadly follows a process with research, ideation, design, development and testing phases to move from conceptual idea to tangible product (Davis, Bersoff & Comer, 1988). By going through this process, knowledge and know-how is gained and through iteration – going through the process multiple times – the problem definition and requirements can be refined, further background research can be done, etc. Although, as Despa, 2014 outlines, there may be different approaches (incremental, waterfall, spiral, prototyping, etc.) to how these phases take shape (e.g. differences in duration of phases, or expected intermediate outcomes, etc.). Each design process has essential components. See ***Figure 1*** for an illustration of the key components of a software design process.

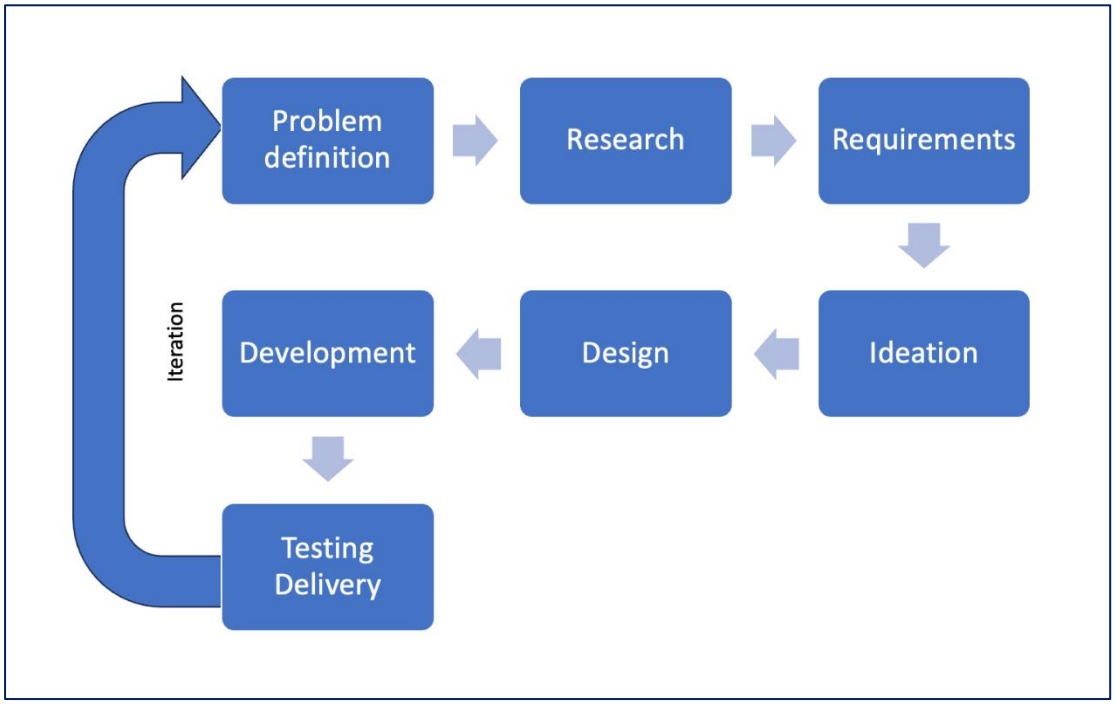

Figure 1 - Software design process, abstracted from Humphrey (1988).

Source: https://dl.acm.org/doi/pdf/10.1145/75111.75122

In this process, coding – *the process of writing actual software code* – primarily takes place in the development and testing phase. At this point in the design process, the problem and potential solution have already been defined, and high-level design decisions have been taken. The software is then produced in the development phase, and tested in the testing stage. Compared to other engineering disciplines, production costs in software are relatively low. Here too, we have the *"industrialisation"* of software production, where efforts are continuously made to be even quicker and more time-efficient in development, whilst ensuring the produced material is of certain quality (Humphrey, 1988). Some ways of doing this are, for example, through efficient sharing of blocks of code (e.g. GitHub repository-style platforms), and more recently, low-code/no-code platforms, where reusable building blocks are used as a starting point to

approach design in a more abstract way, speeding up the whole process (No-Code.tech, 2023; Adalo, 2023). This essentially opens up coding to people, who are not skilled (nor possibly have the interest to be skilled) in the mechanics of writing code.

On the opposite side of this dimension is "software craftmanship", where the technical skill or *craft* of creating code becomes important once again. Approaching software creation through the eyes of an apprentice, this movement stimulates pride in the creation of software, values tradition in software creation, and places the focus on the individual genius (Software Craftmanship Manifesto, 2009). You could even compare it to how we approach cooking at home and cooking in a professional kitchen. In our homes, we look primarily at cooking as the process to produce good food. When cooking becomes a "*craft*," it involves discipline, quality, time-efficiency, and a focused effort to reach heights in quality products. *"Craftmanship"* in cooking involves honing the production skills and resources available to reach this level of excellence (like one would hone their cooking skills - by using quality ingredients, perfecting knife skills, creating new techniques to achieve different flavours and experiences, etc.)

We can take these very same ideas, and extrapolate them to the craftsmanship involved in the writing and production of quality code.

Proliferation of coding in knowledge-intensive jobs

As complex as it sounds, coding is not confined to the spaces of the professional software or ICT industry. One understated fact is that some form of coding is required for many specialised professional technologies in knowledge work. For example, if you are doing some statistical analysis, or even using basic Excel functions, you already need to understand writing code in some minimal form at least. In this context, coding has become like other competences and literacies (think: human language and mathematics), where everyone needs to have basic comprehension and production proficiencies to participate in society (Burke, O'Byrne and Kafai, 2016). The difference lies in the fact that our understanding of coding as a literacy is not yet as advanced, nor as embedded in our schooling system as these other literacies (Rea, 2022). Primary schooling system is built around these literacies and they are essential to build on top of in later years. Yet, the digital transition is changing this fast, making coding and computational thinking just as needed from an early age as other literacies.

Technological support vs our general understanding

So, this creates a mismatch: on the one hand, we have not yet understood the intricacies of dealing with coding as a literacy, nor have we it embedded it fully within our education systems (Vee, 2017). On the other hand, we now have very sophisticated tools at our disposal, enabling us to create, design, and write code. Moreover, the nature of the digital space is such that all these tools are mostly accessible (against a fee) to a wide public.

To illustrate this mismatch, consider applications of ChatGPT and how they are perceived by people:

When you ask ChatGPT to ***write a piece of text***, ***nearly everyone will be able to say if it is a well-written piece of text*** (not considering the content of it for the moment, but purely the form of the text in natural human language).

When you ask ChatGPT to ***complete a mathematical calculation***, **many in the general public** will be able to say *if it was correctly done* or at least if the *correct approach to solve the question was chosen*.

But…when you ask ChatGPT to write ***a piece of code***, **not many in the general public will be able (yet) to say if it is correct.**

How GenAI can alleviate the skills shortage in the labour market

The paragraph above shows one of the areas where Generative AI could prove to be a game changer. **We will now dive deeper into the current issues of the skills shortage and look at the potential of GenAI to alleviate them**.

A first issue concerns **the large differences in the level of proficiency in coding skills between people.** Different people have different levels of proficiency in writing code, and there is no generally accepted basic level in coding proficiency. There are several reasons for this:

1) As coding and programming is not positioned as a literacy, it is not embedded in formal education throughout the curriculum. This means there is *no general access to coding and programming opportunities*, *nor a general expectation towards every individual to develop a basic level of proficiency.*
2) *The development of the skill largely depends on personal individual interest and personal discipline*. If you want to develop your skills in this area, you need to seek out formal and non-formal opportunities online and offline through workshops, bootcamps etc. As this is largely outside of formal education, it requires extra effort and time to invest in this form of skill development.
3) The bottom-line matters here: *skill development in coding depends on investing time and money in relevant opportunities to develop coding skills*. This investment may not be possible generally for everyone in society, as there is a need for infrastructure (hardware and internet), time (outside of formal education time) and potential extra costs for relevant software or licenses.

As in many issues on digital inclusion, there is a large gap in society between the haves and have-nots in this area, with vulnerable people, women, the elderly and people from lower socio-economic backgrounds highly effected (United Nations, 2023).

Generative AI for coding can largely reduce the threshold for people to invest in learning to code. As it allows for interacting with the computer in natural language, anyone who can formulate their ideas for software in human language, could in theory write the code to make this software. However, the issues of accessibility (in time and efforts) remain: here too, there is a possibility that GenAI widens the gap between those who can and those who cannot code.

A second issue concerns the immediacy of the skills-need for people in employment. The labour market needs these digital skills today and in the near future. **Cedefop predicts a need for nearly 979,600 ICT technicians and another 2,977,600 ICT professionals in the period 2022-2035** (Cedefop, 2023). However, building in-depth knowledge and developing these skills to higher levels of proficiency takes time, which is also potentially a barrier to taking up coding in the first place. Bootcamps and other short-term but intensive types of learning are effective but require time and resources that not everyone necessarily has (Thayer & Ko, 2017). Moreover, employees do not only need to achieve a basic level of proficiency – they also need to depend on themselves for continuous improvement in developing these skills.

GenAI has the potential to alleviate this issue to some extent. The accessibility of the technology in natural language can *lower the threshold* for many and provide just-in-time support to find what is needed, thereby making coding and code generation more efficient, and possibly motivating people to use it. **The technology could support self-efficacy,** making it easier and quicker to find answers when they are stuck, and creating an environment in which learners want to continuously develop their skills in coding and programming. However, here too, there is the possibility that these technologies will widen the gap between those who already can and those who cannot. Moreover, in this context, a GenAI-based support tool effectively acts an expert system (*i.e. the human asks, the GenAI creates*). It is important to consider here what level of human oversight is necessary.

A third issue concerns how skills development in coding and programming is currently organised. In formal contexts, coding skills are fit into workshops or bootcamps – potentially around specific coding languages or more generic problem-based activities. Non-formal contexts depend on individuals creating their own learning paths through various online resources, mixed with alternative off-line experiences.

GenAI tools can augment these activities with low-threshold just-in-time support to find what is needed thereby making code generation more efficient and more effective. However, the use of these tools still rely on the human determining correctness of the answers and acceptability of the software outcome. The risk here is that GenAI tools could become a go-to system, where the human learner only uses it to achieve an immediate goal, without developing in-depth knowledge of coding.

Implications for skills development in coding & programming in an age of GenAI

It is clear that GenAI offers many opportunities to lower the threshold towards many in the general public to start not only designing software, but effectively putting software to use in day-to-day contexts. In fact, for many learners, the low threshold may also be a motivating factor to engage with coding in the first place. However, to ensure that humans still develop in-depth knowledge about coding – and do not only see the GenAI as a tool that can take over the messy job of coding – some precautions still need to be taken.

Firstly, it is necessary to approach coding as a literacy that can be embedded throughout the formal educational system. Although efforts are already in place to incorporate coding into lessons plans and curricula, a more systemic change is needed (Rea, 2022; Vee, 2017). The key frameworks on digital skills listed above give more insight into the complexity of these skills and offer ways for a more nuanced approach towards developing them.

Secondly, people also need to be aware of the limits of AI – what it can and cannot do, and even more, what it should and should not do. By its nature, AI technology takes over some decision-making activities from people. Although this creative many positive opportunities (for example, being able to deal with much more data than a human), it also carries severe risks, that people need to be aware of when using these systems. On a policy level, this is the route that is emphasized. The latest version of DigComp includes knowledge of AI systems as a competency for every citizen. The EU AI Act is positioning the guardrails in this environment and clarifying what are low, high and unacceptable risks in working with AI tools.

Thirdly, for coding specifically, it is useful to explore some best practices and guidelines. GenAI will undoubtedly have a massive impact on the practice of coding. In theory, this lowers the threshold and opens up an opportunity for more people to learn coding – and especially those that may have lost motivation due to the technical details and mechanics around writing code. **To use these GenAI tools effectively then, we must first establish proper guides and agreements of how and when these tools can be used**.

In any use of GenAI for coding, it is clear that a computer will write code that people will use. *But how do we make sure we can trust this code?* In this scenario, there is no absolute human oversight anymore. Furthermore, what are the quality control and quality processes put in place to ensure that the GenAI code remains trustworthy? Are new testing procedures needed in this new context, and what would they look like? *It is equally useful to open up the debate of what trust entails, and what are markers to indicate this trustworthiness of the AI.* A possibility is the use

of new quality labels, or specifying use of AI in ethical codes such as in the journalism sector where the use of GenAI writing tools is increasing (Raad van Journalistiek, 2023).

Finally, it is important to continue to recognise and value craftmanship in software coding and programming. For many creators, coding is the language of their craft which gives them intellectual challenges and cognitive satisfaction when they achieve their designs, but their accomplishments are largely hidden. How do you make these more visible and acknowledge their achievements? In these cases, we should not limit our vision to the utilitarian value of coding, but appreciate the skill and craftmanship in it as well. Additionally, it is through engagement with the practice of writing code that people develop in-depth knowledge of coding, which gives them the foundation to improve the language of coding and create new and improved languages for coding. **In other words, if we want to continue to make software coding better, we need sufficient people who write code themselves. A re-appreciation of the human skill here is important.**

Future avenues

Generative AI offers many opportunities in the areas of software coding and programming, potentially making it easier for more people to engage with coding and create their own applications. This is a huge potential for democratisation of software creation, i.e. opening it up to those who may not have been motivated to learn or seen it as a possibility for themselves. GenAI can also make the process of writing code more time- and effort-efficient. However, a downside is that relying too much on the automated development of code without sufficient human oversight can mean loss of quality, and missing opportunities for improving coding practice.

Looking towards the future, some general avenues for exploration can be seen:

Fit-for-purpose software comes from human insight, that aligns perceived needs, technical requirements, with technological design, application and eventual use. Software still needs to work in a complex human environment, and humans are most capable to deal with this complexity. However, it is essential that we find a balance between automated coding possibilities with GenAI and human writing of code or other forms of oversight, that will ensure sufficient human insight into this process and allow for improving it effectively in the future. Quality Assurance processes in software development are already becoming more prolific, with a move to "Shift Left", i.e. engage with quality assurance earlier in the software development process and continue this throughout the process. In other words, many assurances for quality could be foreseen before the stage of effective code-writing is reached. GenAI-support for writing code is then embedded in a wider qualitative development process, where its strengths are optimally used.

Bibliography

Adalo (2023) *Build Custom Responsive Apps & Publish Anywhere — No Coding Required.* Available at: https://www.adalo.com/?via=nile

Brady, D. (2023) 'How generative AI is changing the way developers work', *GitHub Blog*. Available at: https://github.blog/2023-04-14-how-generative-ai-is-changing-the-way-developers-work/ [Last accessed: 25 July 2023].

Burke, Q., O'Byrne, W.I. and Kafai, Y.B., 2016. 'Computational participation: Understanding coding as an extension of literacy instruction' in *Journal of adolescent & adult literacy*, vol. 59(4), pp.371-375

CEDEFOP (2023) Skills Intelligence – Future Employment Needs In EU27 in 2022-2035 Available at: https://www.cedefop.europa.eu/en/tools/skills-intelligence/future-jobs?year=2022-2035&country=EU27#5 [Last accessed: 7 August 2023].

CEN (2019) CEN/TC 428: e-Competence Framework (e-CF) - A common European Framework for ICT Professionals in all sectors - Part 1: Framework Available at: https://standards.cencenelec.eu/dyn/www/f?p=205:110:0::::FSP_PROJECT:67073&cs=15E62ED24D608A5F10D6BEE8E6D50FA10 (Accessed 7 August 2023).

Davis, A.M., Bersoff, E.H. and Comer, E.R., 1988. A strategy for comparing alternative software development life cycle models. IEEE Transactions on software Engineering, 14(10), pp.1453-1461. Available at: https://www.semanticscholar.org/paper/A-Strategy-for-Comparing-Alternative-Software-Life-Davis-Bersoff/bd9a8731723cc88bde1a7639

Despa, M.L., 2014. 'Comparative study on software development methodologies' in *Database Systems Journal*, *5*(3). Available at: https://www.semanticscholar.org/paper/Comparative-study-on-software-development-Despa/28d32b9375e9125624ec614a2cec85c9f0716b13

EIT Digital (2020) 10 Years EIT Digital 2010-2020 – Celebrate Innovation. EIT Digital. Available at: https://www.eitdigital.eu/our-messages/history-strategy-annual-reports/celebrate-innovation-the-eit-digital-10th-anniversary-book/

European Software Skills Analysis - ESSA (2021). *Europe's Most Needed Software Roles and Skills. Needs Analysis Report 2021*. Available at: https://www.softwareskills.eu/wp-content/uploads/2022/10/D.4_ESSA_Europea%CC%82__s-Most-Needed-Software-Roles-and-Skills.-Needs-Analysis-Report_FINAL-draft.pdf [Last accessed 7 August 2023].

Hugging Face (2023) Hugging Face – The AI Community building the Future Available at: https://huggingface.co/huggingface [Last accessed 7 August 2023].

Humphrey, W.S., 1988, April. The software engineering process: definition and scope. In Proceedings of the 4th international software process workshop on Representing and enacting the software process (pp. 82-83) Available at: https://dl.acm.org/doi/pdf/10.1145/75111.75122

IBM (2023) Software Development. Available at: https://www.ibm.com/topics/software-development [Last accessed 25 July 2023].

Kandhasamy, J.S. and Xie, M., 2004. 'From text to images through meanings' in *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. September 2004. pp. 222 - 227.

Nestor Maslej, Loredana Fattorini, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons,James Manyika, Helen Ngo, Juan Carlos Niebles, Vanessa Parli, Yoav Shoham, Russell Wald, Jack Clark, and Raymond Perrault, "*The AI Index 2023 Annual Report*," AI Index Steering Committee, Institute for Human-Centered AI, Stanford University: Stanford. Available at: https://aiindex.stanford.edu/report/

Mccallum, E., Weicht, R., Mcmullan, L. and Price, A. 2018. *EntreComp into Action - Get inspired, make it happen: A user guide to the European Entrepreneurship Competence Framework*, Bacigalupo, M. and O`Keeffe, W. editor(s). Publications Office of the European Union: Luxembourg. Available at: https://doi:10.2760/574864.

No-Code.tech (2023) Academy Learn No-Code Available at: https://www.nocode.tech/academy [Last accessed 25 July 2023].

OpenAI (2023) OpenAI Available at: https://openai.com/about [Last accessed 25 July 2023].

Raad Van Journalistiek (2023) Nieuwe richtlijn over het gebruik van artificiële intelligentie in de journalistiek. Available at: https://www.rvdj.be/nieuws/nieuwe-richtlijn-over-het-gebruik-van-artificiele-intelligentie-de-journalistiek [Last accessed 7 August 2023].

Rea, A., 2022. Coding Equity: Social Justice and Computer Programming Literacy Education. IEEE Transactions on Professional Communication, 65(1), pp.87-103. Available at: https://doi.org/10.1109/TPC.2022.3143965

Reiter, E. and Dale, R., 1997. Building applied natural language generation systems. Natural Language Engineering, 3(1), pp.57-87. Available at: https://www.cambridge.org/core/journals/natural-language-engineering/article/abs/building-applied-natural-language-generation-systems/FEB374A3FF652F06D8567A6FAB2EF36E

Sharma, G. 2021. How artificial intelligence and machine learning are revolutionizing software development, IEEE Computer Society. Available at: https://www.computer.org/publications/tech-news/trends/ai-is-changing-software-development [Last accessed: 25 July 2023].

Software Craftmanship Manifesto. 2009. *Manifesto for Software Craftmanship – Raising the bar*. Available at: https://manifesto.softwarecraftsmanship.org [Last accessed 25 July 2023].

Stokel-Walker, C. and Van Noorden, R. 2023. *What CHATGPT and Generative AI mean for science*, *Nature News*. Available at: https://www.nature.com/articles/d41586-023-00340-6 [Last accessed 24 July 2023].

Thayer, K. and Ko, A.J., 2017, August. Barriers faced by coding bootcamp students. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 245-253).

United Nations (2023) Digital Inclusion. Available at: https://www.un.org/techenvoy/content/digital-inclusion [Last accessed: 24 July 2023].

Vee, A., 2017. Coding literacy: How computer programming is changing writing. Mit Press.

Vuorikari, R., Kluzer, S. and Punie, Y., (2022) DigComp 2.2: The Digital Competence Framework for Citizens - With new examples of knowledge, skills and attitudes, EUR 31006 EN, Publications Office of the European Union, Luxembourg. Available at: https://publications.jrc.ec.europa.eu/repository/handle/JRC128415 [Last accessed: 25 July 2023].