RQ1: What are the perceived advantages and disadvantages of developers using LLM-based tools for the implementation and coding phase of software development?		
ID	**Finding**	**Articles**
Advantages		
	Productivity	
F01	**Using LLMs for generating code increases productivity.**	**30, 42, 48, 63, 69, 91**
	Code generation tools increase productivity, since the programmer does not need to type as much and does not need to look up syntax or documentation.	30
	One benefit highlighted in the interviews is to speed up the process of creating code, resulting in faster development.	42
	Developers can focus on more complex or strategic aspects, thanks to the faster development.	42
	LLM based code techniques have great potential and can greatly boost productivity.	48
	ChatGPT produced a good starting example of solving the task, boosting efficiency.	63
	Code generated by LLMs can speed up the coding process and serve as a foundation for developers to build from.	69
	Integrating LLM into (Flutter) development can accelerate the coding process and improve app quality.	91
F02	**LLMs can be used to complement or complete tasks automatically.**	**42, 71, 141**
	The developers also used assistants to complement or to auto-complete some coding tasks.	42
	Assistants can be used as a tool for enhancing the code quality of developer-written code.	42
	Copilot's code had values of lower Cyclomatic Complexity.	71
	For algorithmic tasks, ChatGPT performs on par with the average human.	141
F03	**LLM generated code is useful to find a starting point or example.**	**42, 63, 69**
	Used to help developers to find a starting point when working on a task, language or API they had no or little knowledge of to solve.	42
	ChatGPT produced a good starting example of solving the task, boosting efficiency.	63
	Code generated by LLMs can speed up the coding process and serve as a foundation for developers to build from.	69
F04	**Using LLMs to generate code is beneficial when learning something new.**	**42**
	Used to help developers to find a starting point when working on a task, language or API they had no or little knowledge of to solve.	42
F05	**LLM generated code is easier to fix compared to code written by a developer.**	**71**

	The cost of repairing code generated by Copilot was lower in terms of repair time & patch site.	71
F06	**LLMs can be used to enhance code security.**	**15, 44, 50, 64**
	ChatGPT produced code containing 20% less CWE vulnerabilities compared to code snippets found on StackOverflow.	15
	Code revised through ChatGPT addressed some security issues, but improvements are required for optimized security.	44
	There are potential benefits to security when using code generation tools, particularly when solving harder or more complex tasks.	50
	No significant difference of introducing CWE vulnerabilities when using Copilot	50
	Repaired 100% of the security bugs for the scenarios.	64
	While many models provided fixes to the program, they introduced new bugs.	64
Prompting		
F07	**You are able to adapt the code generated by LLMs.**	**88**
	The solutions are customizable.	88
Disadvantages		
Productivity		
F08	**Repeatedly switching between prompt and code disrupts programming flow and cause cognitive overload.**	**5**
	Repeatedly switching between prompt and code disrupts programming flow.	5
	Experiencing cognitive overload caused by context switching.	5
F09	**Incorrect results may lead to developers getting stuck in a debugging cycle.**	**88**
	Participants found themselves getting stuck in a debugging cycle when receiving incorrect results	88
F10	**LLM generated code requires fine-tuning before implementation.**	**63, 90**
	Fine-tuning to the generated code from the developer is often beneficial or important to achieve production quality.	63
	ChatGPT may generate code not as precise and efficient as experienced programmers would produce, requiring additional fine-tuning of the code.	90
Prompting		
F11	**The limitations regarding the length of inputs and generated output may require developers to break down tasks**	**64, 90, 124**
	Repairs were restricted to a single place in a single file.	64

	The token limit limits the code length generated in a single prompt, which may require developers to break down a task.	90
	Due to computational complexity, LLMs have a limitation of how long generated outputs can be.	124
F12	**The volume of code suggestions generated from several prompt can sometimes be distracting or interrupt the development process.**	**5, 42**
	Difficult to go back and forth between code and prompt, due to the several prompt iterations.	5
	The volume of code suggestions can sometimes be distracting or interrupt the development process.	42
F13	**A programmer may have to refine their prompt, since providing more context to LLM increases code quality and correctness.**	**7, 42, 57, 142**
	Enriching prompts with UML info increased validity of OCL constraints generated by Codex	7
	Code generated by LLMs is sometimes not accurate or good enough, which may require the programmer to refine their prompt.	42
	Detailed inputs, such as prompts with code snippets, generate more accurate code since the model has more context	57
	If prompts are vague or complex, it becomes harder for the models to generate code of high quality.	142
F14	**It can sometimes be hard to express intentions to LLMs.**	**5, 90**
	The linear prompt makes it challenging to externalize intention for generated code.	5
	The model may sometimes not understand the developers intent.	90
F15	**Identical prompts can result in different code outputs and vary in quality.**	**69**
	Code outputs are inconsistent despite using the same prompt and the code can vary in quality.	69
F16	**Lack of voting and community discussion makes it harder to evaluate generated code.**	**30**
	No voting mechanism or discussion for LLM generated code, compared to solutions posted on stack overflow.	30
	Users are not able to see how or why the AI came up with the suggestion, therefore they may find it difficult to evaluate its validity.	30
	Beginner programmers rely on resources generated by communities to learn about the vulnerabilities that AI might introduce, while experienced programmers may already be aware of these.	30
RQ2: What are the limitations and / or risks of using LLM-based tools in the implementation and coding phase of software development?		
ID	**Finding**	**Articles**

Risks		
Security		
F17	**LLMs generates code containing vulnerabilities.**	**65, 66, 92, 101, 104**
	Some findings found that AI produces easy-to-fix bugs, while others found well-documented vulnerabilities and malware.	65
	Copilot generated vulnerabilities for 40% of the total amount of programs.	66
	Generated code often includes relevant vulnerabilities.	92
	Code generated (Copilot) had substandard coding and security code smells.	101
	Security concerns regarding LLM generated code include security risks and leaking information, especially in industrial applications.	104
F18	**To make LLMs generate code that avoid vulnerabilities, developers have to explicitly prompt this to the model.**	**7, 62**
	Basic prompts in Codex resulted in a low validity score for generated (OCL) constraints	7
	ChatGPT may not know which vulnerability to address, unless the type of vulnerability is explicitly specified in prompts.	62
	ChatGPT did not address vulnerabilities unless told to do so.	62
F19	**Developers should test and analyze the generated code before implementation.**	**35, 44**
	Developers should apply automatic and manual security analysis before integrating code suggestions generated by Copilot.	35
	Following guidelines and incorporating best practices reduces common programming errors and vulnerabilities of AI-generated code.	44
	Robust processes of code reviews and security testing can help identify and address vulnerabilities of generated code early in the development.	44
	Improving security knowledge and skills of employees reduces the risk of introducing vulnerable code.	44
F20	**LLM generated code should not be trusted blindly.**	**15, 30, 35, 65, 69**
	Recommendation: Code produced by both LLM or human sources should not be blindly trusted, and to reduce the risk good software practices are required.	15
	It is hard to evaluate the validity of the code, since users are not able to see how the LLM came up with the code.	30
	For one scenario of SQL injection, the vulnerability ratio had worsened. The recommendation for developers is therefore to be cautious when using tools for code generation, based on the simplicity of the scenarios and the fact that Copilot still generates vulnerable code.	35

	Some articles reported that developers have an unjustified trust in LLM generated code.	65
	Interview participants were concerned about how trustworthy LLMs are regarding code generation.	69
	Generated code needs to be reviewed in terms of correctness & security.	69
Limitations		
	Correctness	
F21	**Insufficient or absent exception handling is a common issue in LLM generated code.**	**9, 63, 100**
	Incomplete exception handling, incorrect exception handling and abuse of try-catch statements are the three challenges of LLM generated code.	9
	Incomplete exception handling is a common phenomenon, occuring in almost all the code generated by ChatGPT.	9
	The code did not include error handling or large data optimization.	63
	Generated code contains various types of execution errors.	100
F22	**The correctness of LLM generated code is lower compared to code written by developers.**	**71, 103**
	Copilot Correctness Ratio was lower compared to students code. This was increased to 98,5% when leveraging a Program repair tool.	71
	The code generated by the LLM performed on par or worse than human code.	103
F23	**A limitation for LLMs is to generate unique solutions**	**71**
	Out of 28 correct solutions generated by Copilot, only 2 were unique.	71
F24	**LLMs are prone to hallucinate, affecting the generated code.**	**117, 124, 144**
	LLMs often hallucinate, and generated code often includes random facts or links that are irrelevant to the topic.	117
	LLMs often exhibited unwanted behaviour, such as making things up.	124
	Copilot sometimes hallucinates inaccessible elements not specified in the requirements.	144
	Copilot was dependent on accessing current elements of a website, as it kept producing inaccessible elements for web pages already lacking them.	144
F25	**Generating code that adheres to constraints or non-functional requirements is a challenge for LLMs.**	**100, 124, 142, 144**
	The most common issues of code generate by ChatGPT were Wrong Outputs and Code Style & Maintainability	100
	It can be challenging for LLM to generate output that adheres to non-functional requirements.	124

	Generated code sometimes lacks crucial information about constraints or formatting requirements from the prompt.	142
	Copilot can not surpass the accessibility standards set by professional developers yet.	144
Complexity		
F26	**LLM generated code often lacks versatility and only solves specific tasks.**	**9**
	A limitation of ChatGPT is to generate versatile code that can perform complex tasks.	9
F27	**The performance of LLMs decreases with task complexity.**	**64, 100**
	LLMs performed best when generating small code samples and worse in complex contexts.	64
	The difficulty has a significant impact on performance to generate code.	100
	The percentage of clean code decreased with task difficulty.	100
	The size of the program also affects performance, as the model performed worse as code length grew.	100
F28	**LLM generated code includes unnecessary nested loops and variable declarations.**	**103**
	The code included unnecessary nested loops and variable declarations.	103
RQ3: Does the availability of LLM support influence developers' choice of programming languages?		
ID	**Finding**	**Articles**
Language & Tools		
F29	**The performance and ability of generating code varies a lot between models.**	**44, 55, 102**
	The quality and security of LLM generated code can vary significantly based on what model is used.	44
	Domain-specific models outperform larger, widely used models when generating MPI-based programs, which suffers from performance when generating these compared to general-purpose programs.	55
	The models performed differently for different languages in terms of correctness and amount of executable code.	102
F30	**Newer models generates code of better quality and improved security.**	**35, 98**
	The percentage of vulnerable code generated has, since improvements to Copilot, decreased from 36.54% to 27.25% when compared to the original study.	35
	Newer models, while slower, have a higher accuracy and better code quality.	98
F31	**LLMs are not always up-to-date with current versions of languages, libraries and other technologies.**	**42, 90, 91, 100**

	Versioning was also reported being a challenge, since the suggestions were not always up-to-date.	42
	ChatGPT is not always up-to-date on specific libraries and other tools, which some programs may rely on.	90
	Although LLM are good Tools for flutter development, improvements can still be made, especially for more complex tasks to fulfill requirements	91
	The types and frequency of compilation and runtime errors differed between Java and Python.	100
F32	**The proficiency of LLMs to generate code varies between programming languages due to varying amount of available training data.**	**64, 65, 66, 74, 90, 92**
	LLMs are less proficient when producing Verilog code compared to C.	64
	Languages with smaller data sets with lower quality for the model to be trained on produced less secure code.	65
	Due to the smaller amount of available training data, Copilot struggled more to generate correct code syntax compared to Python and C.	66
	LLMs perform really well when generating code in languages prevalent in their datasets, the quality drops significantly when generating code in lesser represented languages in niche domains.	74
	Only a few models are able to generate code written in P4, which is underrepresented in open datasets. The ones who can often generate syntax errors and require changes by human developers.	74
	ChatGPT may not be as reliable when generating code for niche languages, due to lack of training.	90
	The probability of generating correct code varies between languages.	92
	ChatGPT may have problems generating code for unfamiliar problems from its training set.	92

Discarded findings		
	Learning about the constraints of AI forms reasonable expectations, as unrealistic expectations could undermine developers' trust.	30
	ChatGPT performánce is low in underrepresented languages.	141