Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Využití nástrojů umělé inteligence pro podporu tvorby kódu

DIPLOMOVÁ PRÁCE

Studijní program: Informační systémy a technologie

Specializace: Vývoj informačních systémů

Autor: Bc. Tomáš Vlček

Vedoucí diplomové práce: Ing. et Ing. Michal Doležel, Ph.D.

Praha, květen 2025

Poděkování

Děkuji svému vedoucímu práce panu Ing. et Ing. Michalu Doleželovi, Ph.D. za všechny cenné a věcné rady, připomínky k práci a čas věnovaný vedení této práce. V neposlední řadě bych rád poděkoval rodině a všem osobám, které mi poskytly podporu a součinnost při tvorbě práce, zejména jednotlivým účastníkům provedeného kvalitativního výzkumu, bez jejichž ochoty by práce nemohla vzniknout.

Abstrakt

Diplomová práce se zabývá využitím AI asistentů kódování při vývoji softwaru. Hlavním cílem je popsat, jaká jsou očekávání programátorů v České republice při používání AI asistentů kódování, jak tyto nástroje programátoři v České republice nejčastěji využívají, jak dopadá používání této technologie na jejich práci. Dále navrhnout sadu doporučení umožňující co nejefektivnější využití a implementaci těchto nástrojů.

Za tímto účelem bylo provedeno celkem devět rozhovorů s programátory využívajícími AI asistenty kódování při své každodenní práci. Z jejich analýzy vyplynulo, že účastníci rozhovorů využívají asistenty kódování několikrát denně a berou je jako běžnou součást své práce. Od asistentů nejčastěji očekávají zvýšení komfortu při práci a zejména pomoc při řešení rutinních a méně zajímavých úkolů. Výrazně asistenti kódování změnili i způsob vyhledávání informací a podkladů ke zpracovávanému úkolu. Pro vývojáře asistenti kódování nabízí nový způsob, jak rychle získat odpovědi na otázky vzniklé během práce a jak diskutovat možná řešení. Díky tomu v určitých situacích vývojáři řešení úkolů konzultují v dřívějších fázích vývoje, což pomáhá některé chyby odhalovat dříve, jiné díky tomu ani nevzniknou. Po implementaci AI asistentů kódování kleslo i množství dotazů směřujících na kolegy v týmu.

Z provedených rozhovorů vyplynuly i slabé stránky asistentů kódování. Mezi ně patří především omezená možnost integrace s ostatními nástroji uchovávajícími požadavky a zadání, nebo omezená možnost integrace se znalostní bází. V některých případech označili participanti jako slabinu i kvalitu vygenerované odpovědi a schopnost asistentů zpracovávat komplexnější nebo složitější úkoly.

Pro prostředí České republiky tak práce přináší dosud neexistující popis vnímání AI asistentů kódování a způsobů jejich využívání ze strany vývojářů, ale také chybějící doporučení pro co nejefektivnější využití a implementaci těchto nástrojů.

Klíčová slova

AI, umělá inteligence, asistenti kódování, očekávání, použití, dopady, tvorba kódu

Abstract

This diploma thesis focuses on the use of AI coding assistants in software development. The main objective is to describe the expectations of programmers in the Czech Republic when using AI coding assistants, how these tools are most commonly employed by programmers in the Czech Republic, how the use of this technology affects their work, and to propose a set of recommendations for maximizing the effective use and implementation of these tools.

To achieve this, a total of nine interviews were conducted with programmers who use AI coding assistants in their daily work. The analysis of these interviews revealed that the participants use coding assistants several times a day and perceive them as a standard part of their work. Their primary expectations of the assistants include increased work comfort and, in particular, help with routine and less engaging tasks. Coding assistants have also significantly changed the way programmers search for information and materials related to the tasks at hand. For developers, coding assistants offer a new way to quickly obtain answers to questions that arise during work and to discuss possible solutions. As a result, in certain situations, developers consult on task solutions at earlier stages of development, which helps detect some errors sooner, and prevents others from occurring altogether. Following the implementation of AI coding assistants, the number of questions directed at team colleagues has also decreased.

The interviews also revealed weaknesses of coding assistants. These include primarily the limited ability to integrate with other tools that store requirements and task specifications, or the limited possibility of integration with knowledge bases. In some cases, participants also identified the quality of generated responses and the assistants' ability to handle more complex or sophisticated tasks as weaknesses.

In the context of the Czech Republic, this thesis thus provides a previously non-existent description of how AI coding assistants are perceived and used by developers, as well as missing recommendations for the most effective use and implementation of these tools.

Keywords

AI, artificial intelligence, coding assistants, expectations, usage, impact, code creation

Obsah

Úvod

Během posledních měsíců došlo ke značnému a rychlému rozmachu umělé inteligence (AI), pokroku v generování textu a v oblasti velkých jazykových modelů (LLM) (Federal Office for Information Security, 2024). Nástroje využívající generativní umělou inteligenci se staly běžnou součástí pracovního procesu v nejednom oboru. Mezi příklady nejčastěji používaných nástrojů využívajících umělou inteligenci lze zařadit např. ChatGPT, Dall-E aj., ale také ty upravené speciálně pro konkrétní profese. V případě oblasti vývoje softwaru se jedná o tzv. „*AI coding assistants*“. V této práci je užíván český název „asistenti kódování“, nebo „AI asistenti kódování“. Asistenti kódování dokáží podpořit širokou škálu činností, které vývoj softwaru jako komplexní obor zahrnuje. Tyto nástroje se během samotné implementaci informačního systému hojně používají zejména pro generování kódu a jeho testování (Pan et al., 2024).

Podle zahraničních studií jsou asistenti kódování vývojáři vnímáni spíše jako nástroje usnadňující a zrychlující vývojářskou práci, snižující nutnost danému problému vyčleňovat tak velké množství kapacit (Bird et al., 2023). Díky tomu se programátoři mohou místo řešení jednoduchých nebo často se opakujících úkolů soustředit na tvorbu a vývoj složitějších, důležitějších a komplexnějších projektů. Většina studií zmiňuje, že nástroj je velmi dobrým počátečním bodem, který pomáhá programátorům zorientovat se v dané problematice, včetně vyhledávání relevantních zdrojů. Ve velké většině případů asistenti kódování vývojářům nahrazují dosud běžné hledání podkladů na internetu (Vaithilingam et al., 2022). I když se jedná spíše o subjektivní pocity samotných programátorů, práce Asareho et al. (2024) zmiňuje i skutečnost, že jsou si programátoři při používání těchto nástrojů jistější.

Klíčovými tématy mezi vývojáři zůstává bezpečnost, soukromí, problematika duševního vlastnictví a nakládání s použitými daty ze strany poskytovatelů služeb. Jako klíčový problém je vnímána i správnost generovaných odpovědí (Arkheden & Eklund, 2024) a menší porozumění kódu, pokud je kód vytvořen AI asistenty místo samotných vývojářů (Vaithilingam et al., 2022). To může mít paradoxně ve výsledku za následek nutnost věnovat úpravám a ladění nefunkčního kódu vygenerovaného nástrojem více času, než kdyby si vývojáři napsali kód sami (Prather et al., 2023).

Vymezení mezery v poznání a problému

Jak z výše uvedených zjištění vyplývá, popsaný technologický pokrok a dostupnost nástrojů vedly k jejich širokému nasazení v různých fázích vývojového cyklu. Od psaní kódu až po jeho testování a ladění. Nicméně i přes rostoucí popularitu a široké užívání těchto nástrojů zůstává způsob jejich používání v reálných vývojových týmech a organizacích v České republice stále neprozkoumán a nepopsán. V zahraničí začínají být způsoby používání a dopady těchto nástrojů známé, avšak přímo prostředí České republiky se žádná práce nevěnovala. To může vést k nejasnostem a otázkám ohledně toho, jak jsou nástroje nejčastěji používány, jaká jsou očekávání programátorů, jak dopadá používání této

technologie na jejich práci a jak nejefektivněji využít a implementovat tyto nástroje v kontextu prostředí České republiky. Tyto nejasnosti následně mohou způsobovat odmítání využívání asistentů kódování, snížení efektivity práce vývojářů, nebo v případě firem i ztrátu konkurenceschopnosti.

Cíle diplomové práce

Hlavním cílem diplomové práce je charakterizovat očekávání programátorů v České republice při používání tzv. AI asistentů kódování, jak tyto nástroje programátoři v České republice nejčastěji využívají, jak dopadá používání této technologie na jejich práci, a dále navrhnout doporučení umožňující co nejefektivnější využití a implementaci těchto nástrojů.

Výše uvedený hlavní cíl práce se skládá z následujících dílčích cílů.

- **DC1:** Provést rešerši existující literatury věnující se aktuálnímu stavu poznání využití AI asistentů kódování, dopadům používání na vývojáře a jejich práci.
- **DC2:** Sestavit interview protokol zaměřující se na využití AI asistentů kódování, jejich dopady na vývojáře a jejich práci v kontextu České republiky.
- **DC3:** Na základě vytvořeného protokolu provést rozhovory s vybranými vývojáři a tyto rozhovory tematicky analyzovat.
- **DC4:** Vytvořit a ověřit sadu doporučení umožňující co nejefektivnější využití a implementaci AI asistentů kódování.

Výzkumné otázky

Na základě hlavního cíle byly definované následující výzkumné otázky, na které bude práce hledat odpovědi.

- **RQ1:** Jak vývojáři v České republice vnímají AI asistenty kódování a jaká jsou jejich hlavní očekávání a očekávané přínosy při používání AI asistentů kódování?
- **RQ2:** Jak vývojáři v České republice při své práci využívají AI asistenty kódování?
- **RQ3:** Jaký dopad má využití AI asistentů kódování na práci a výkon vývojářů v České republice během vývoje softwaru?

Metody výzkumu

Těžištěm práce bylo provedení kvalitativních polostrukturovaných rozhovorů, které byly vedeny podle předem připraveného interview protokolu, a jejich následná tematická analýza. Interview protokol byl vytvořen na základě systematické rešerše zdrojů. Ta byla provedena primárně nad databází ACM Digital Library. Druhá fáze rešerše obsahovala tradiční rešerši zdrojů. K těmto účelům byl využit vyhledávač Google Scholar, databáze ProQuest Central a databáze vysokoškolských kvalifikačních prací Theses.cz. Finální struktura interview protokolu odpovídá stanoveným rešeršním otázkám a má následující podobu: (1) úvod, (2) očekávání, (3) použití, (4) dopad a (5) závěr. Část otázek byla převzatá ze zahraničních výzkumů, k nim byla navíc přidána i část vlastních otázek vytvořených na základě rešerše existující literatury.

Samotné rozhovory probíhaly podle preferencí jednotlivých participantů buď online nebo v klasické fyzické podobě. Rozhovory s respondenty byly nahrány a po pořízení přepisu byly následně tematicky analyzovány podle metody Braunové a Clarkové (2021). Pro usnadnění provádění tematické analýzy byl využit nástroj MAXQDA.

Na základě finální sady identifikovaných témat pomocí provedené analýzy byla vytvořena sada doporučení umožňující co nejefektivnější využití nástroje reflektující zjištění z jednotlivých rozhovorů. Doporučení nechal autor práce ověřit a připomínkovat vybranými účastníky rozhovorů. Jejich připomínky a zpětnou vazbu následně zapracoval do finální verze doporučení.

Detailní popis všech použitých metod výzkumu včetně průběhu jeho jednotlivých fází obsahuje kapitola 2.

Důvod volby tématu

Téma si autor vybral z důvodu rostoucího významu umělé inteligence, a to nejen v oblasti softwarového inženýrství. Nástroje založené na generativní umělé inteligenci představují v poslední době inovativní prvek, který umožňuje zvýšení efektivity při tvorbě kódu a zároveň může usnadnit práci vývojářům s různou úrovní zkušeností. Sám autor má s nástroji osobní zkušenost a vzhledem k faktu, že jsou v současné době velmi populární a propagované, chtěl zjistit, jak nástroje vnímají vývojáři s různou úrovní znalostí a zkušeností, jak nástroje používají, jak na ně používání dopadá a jak nástroje vnímají. Vzhledem k tomu, že obdobný ucelený popis pro prostředí České republiky dosud neexistuje, autor věří, že tato práce a její výsledky pomohou k zodpovězení popsaných otázek, či inspirují další výzkumníky k realizaci navazujících výzkumů.

1 Stav poznání

Cílem této kapitoly je, navzdory velké dynamice daného prostředí, popsat současný stav na trhu s AI asistenty kódování. Kromě vytvoření přehledu nejpoužívanějších AI asistentů kódování, popisu jejich funkcí a doporučených postupů pro jejich používání, se autor rovněž zaměří na shrnutí klíčových publikací, které se věnují aktuálnímu stavu poznání ve zkoumané oblasti. Všechny zdroje a použité materiály uvedené v následujících podkapitolách byly při tvorbě textu této práce (podzim 2024) aktuální a maximálně relevantní. Je však třeba vzít v úvahu, že od té doby mohlo dojít vzhledem k dynamice trhu k jejich změnám.

1.1 Asistenti kódování

AI asistenti kódování se od jiných produktů využívajících generativní umělou inteligenci liší tím, že jejich jazykové modely byly trénovány speciálně pro poskytování nápovědy a odpovědí zaměřených na vývoj softwaru. Kromě jazykových modelů je zpravidla součástí poskytovaného nástroje i rozšíření do vývojového prostředí (IDE). To umožňuje interakci s asistentem buď přímo v editoru během psaní kódu nebo prostřednictvím samostatného chatovacího okna, přičemž rozšíření vždy zohledňuje aktuálně řešený problém a jeho kontext (Bird et al., 2023). Oba způsoby interakce přinášejí odlišné možnosti a výhody. V praxi se způsoby používání obvykle vzájemně kombinují.

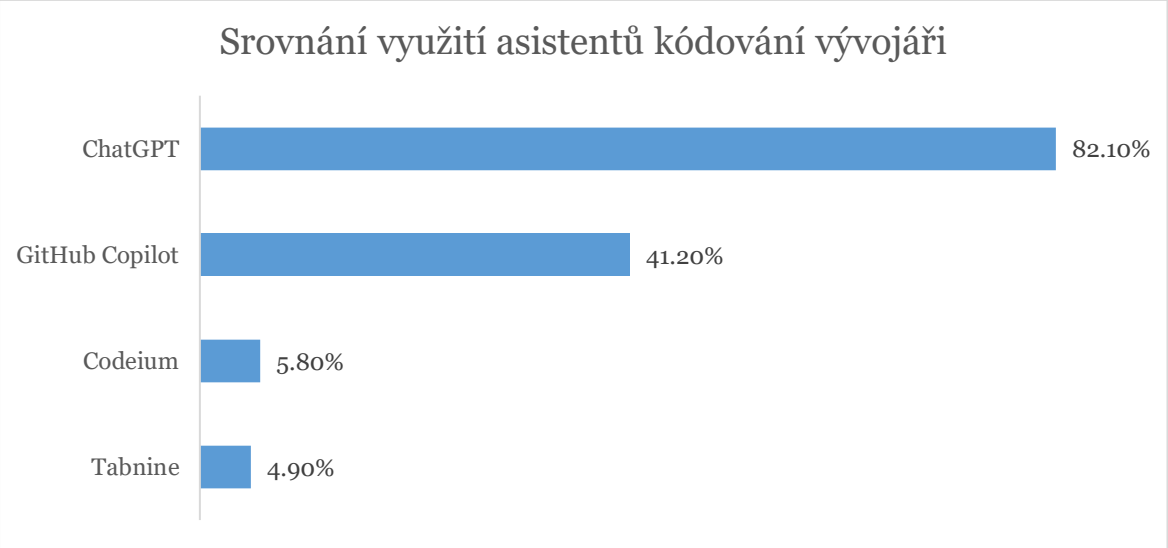
V současnosti nabízí velká část poskytovatelů různé úrovně předplatných lišících se v počtu dostupných funkcí asistenta. Nehledě na vybraný balíček se tvůrci chlubí tím, že jejich řešení nabízejí velmi podobné schopnosti jako programátoři z masa a kostí. Kromě samotných funkcí je možné vybrat si také způsob provozu. Nejčastěji bývají nástroje provozované přímo u jejich tvůrců, existují ale i varianty umožňující provoz na vlastní infrastruktuře. Tuto možnost však obvykle využívají větší podniky, které disponují potřebným zázemím.

V praxi se asistenti kódování používají podobně jako chatboti (Federal Office for Information Security, 2024). Uživatelé zadají modelu instrukci (tzv. prompt), na jejímž základě nástroj vygeneruje odpověď. Prompt může být zadán buď jako popis požadované funkce, nebo jako strukturovaný komentář přímo v kódu. Výstupem obvykle bývá vygenerovaný kód ve zvoleném programovacím jazyce, který odpovídá vývojářovým zadáním. Současní asistenti kódování zpravidla generují více návrhů, díky čemuž si může programátor vybrat ten, který mu do kontextu jeho projektu a práce zapadá nejlépe. Nástroje ovšem neposkytují pouze možnost generování kódu, ale často nabízejí i další funkce, mezi něž patří např. možnosti vyhledávání potřebných informací, oprava a vysvětlení různých částí kódu, generování dokumentace aj.

V posledních měsících je trh s AI asistenty kódování poměrně dynamický. Jen v posledním roce bylo na trh nově uvedeno nebo výrazně vylepšeno několik asistentů kódování. Pro

určení nejpoužívanějších z nich využil autor odpovědí z kategorie „*AI Search and Developer Tools*“ z průzkumu provedeného společností Stack Exchange (2024) v květnu roku 2024 na vzorku 65 000 vývojářů ze 185 zemí světa. Průzkum se opakuje každý rok a jeho cílem je zjistit aktuální trendy v oblasti vývoje softwaru mezi vývojáři. Od roku 2023 průzkum navíc detailněji zjišťuje, jak se v oblasti vývoje softwaru používají nástroje využívající umělou inteligenci.

Na základě odpovědí z tohoto průzkumu vybral autor 3 nejpoužívanější nástroje, které se na trhu pozicují buď přímo jako asistenti kódování, nebo nabízejí velmi podobné funkce. Obecné nástroje jako Google Gemini[1] nebo Bing AI[2] nezahrnoval. Výjimka byla udělána pouze u nástroje ChatGPT[3]. Ten se sice nepozicuje přímo jako vývojářský nástroj, ale vzhledem k jeho dominanci v průzkumu společnosti Stack Exchange Inc. ho autor práce také zahrnul. Celkem tedy autor vybral 4 nástroje – ChatGPT, GitHub Copilot[4], Codeium[5] a Tabnine[6]. Využití zmíněných nástrojů mezi vývojáři je srovnáno v graf 1.



Graf 1: Srovnání využití asistentů kódování vývojáři, upraveno autorem, zdroj: (Stack Exchange Inc., 2024)

Nástroje uvedené v grafu výše autor v následujících podkapitolách popíše a shrne jejich základní funkce. Asistentovi kódování GitHub Copilot bude věnováno více prostoru vzhledem k četnosti jeho využití. Kromě základních funkcí a možností použití bude popsán i princip jeho fungování, integrace s IDE a osvědčené postupy pro jeho efektivní využívání. Ostatní nástroje budou také popsány, ale nebude jim věnována stejná míra detailu. Nicméně principy fungování a osvědčené postupy při využívání GitHub Copilota mohou být do jisté míry aplikovány i na ostatní popisované asistenty kódování.

[1] https://gemini.google.com/
[2] https://copilot.microsoft.com/
[3] https://chatgpt.com/
[4] https://copilot.github.com
[5] https://codeium.com
[6] https://tabnine.com

1.1.1 Chat GPT[7]

Jedná se o nástroj vyvinutý společností OpenAI a je jedním z nejznámějších nástrojů svého druhu. Poprvé byl představen v listopadu 2022 (OpenAI, L.L.C., 2022) a od té doby prošel několika aktualizacemi, které významně zlepšily jeho schopnosti. I když je hlavní specializací ChatGPT práce s textem v přirozeném jazyce, jeho funkce lze využít i v jiných oblastech. Oblibu jeho používání mezi vývojáři dokládá i průzkum uvedený dříve v podkapitole 1.1 této práce. Nástroj není omezen čistě na zadávání vstupů pomocí textu. Uživatel s ním může interagovat i pomocí hlasu, může nahrávat potřebné soubory a interagovat s ním dalšími způsoby. Stejně jako u ostatních nástrojů popisovaných v této práci se jedná o closed-source řešení poskytované čistě jako služba. V základu využívá pro generování odpovědí model GPT-4, který byl do nástroje integrován v březnu 2023 (OpenAI, L.L.C., 2023). Do té doby byl k dispozici model GPT-3.5.

Vedle bezplatné verze nabízí společnost OpenAI svým zákazníkům i placenou verzi. Tato verze poskytuje kromě základních funkcí také vyšší počet dotazů zpracovávaných modelem GPT-4o a možnost generovat obrázky pomocí generátoru DALL-E (OpenAI, L.L.C., b.r.). Na rozdíl od svých předchůdců, modelů GPT-3.5 a GPT-4, dokáže GPT-4o získávat kontext z několika druhů vstupů (text, audio a video) současně, a to v reálném čase.

S nástrojem může uživatel pracovat nejen přes webové rozhraní, ale také prostřednictvím desktopové a mobilní aplikace nebo API, díky kterému lze model používat i v dalších aplikacích. Nenabízí však oficiální možnost přímé integrace do některého z dostupných vývojových prostředí (IDE) na trhu. Na rozdíl od GitHub Copilota neumožňuje nástroji našeptávat přímo při psaní kódu ani poskytovat další funkce spojené s prací v IDE nebo s verzovacími nástroji Git.

1.1.2 GitHub Copilot[8]

Jde o nástroj primárně určený k podpoře a řešení otázek týkajících se vývoje softwaru a souvisejících témat, který byl vyvinut společností GitHub ve spolupráci s OpenAI, autorem populárního modelu GPT. Společnost GitHub jej prezentuje jako „*nejvíce adoptovaný vývojářský nástroj využívající umělou inteligenci*“ (GitHub, Inc., 2022). Na jeho vývoji se pracovalo několik let. První technická verze nástroje byla představena v červnu 2021. O rok později byla veřejnosti zpřístupněna první plná produkční verze (Bird et al., 2023).

Nástroj je v oblasti vývoje softwaru velmi populární. Do svého vývojářského procesu jej zapojilo více než 50 000 společností po celém světě (GitHub, Inc., 2022). Společnost GitHub, tvůrce populárního asistenta GitHub Copilot, dokonce tvrdí, že 88 % programátorů je při používání tohoto nástroje produktivnější a až o 55 % rychlejší (Kalliamvakou, 2022). V době psaní této práce mělo rozšíření, které umožňuje interakci s nástrojem přímo ve Visual Studiu Code, více než 33 milionů stažení (Microsoft Corporation, b.r.-a).

[7] https://chatgpt.com/
[8] https://github.com/features/copilot

Asistent se skládá z vlastního velkého jazykového modelu (LLM) a rozšíření pro vývojová prostředí, které uživatelům umožňuje interagovat s chatem a našeptávat části kódu přímo během jeho psaní. Kromě toho poskytuje další funkce spojené s prací v IDE nebo s verzovacími nástroji Git (Bird et al., 2023). Dále nabízí vlastní chat, který vývojářům umožňuje vyhledávání informací, opravu a vysvětlení částí kódu, generování dokumentace a další funkce. Klíčovou výhodou nástroje je, že jeho jazykový model byl trénován na rozsáhlém množství zdrojového kódu (Bird et al., 2023). Nástroj byl navíc navržen a vyvíjen s cílem poskytnout specializovaný nástroj přizpůsobený specifickým potřebám vývojářů.

Asistent v prvních verzích těžil z modelu OpenAI Codex, který přímo vycházel z modelu GPT-3. Trénink modelu probíhal na datech obsahujících jak přirozený jazyk, tak na milionech veřejně dostupných GitHub repositářů (Alford, 2021). Model OpenAI Codex umožňoval uchovávat až třikrát více kontextových informací než GPT-3 (OpenAI, L.L.C., 2021). Trénování modelu však vyvolalo značnou kritiku, protože obsahovalo použití kódu z veřejně dostupných repositářů bez souhlasu jejich autorů. OpenAI Codex poháněl nástroj až do konce listopadu 2023, kdy společnost GitHub oznámila přechod na novou generaci modelu GPT-4 (GitHub, Inc., 2023).

Nástroj nabízí tři různé typy předplatného: *Individual, Business a Enterprise*. Kromě třicetidenní zkušební doby není možné nástroj využívat bezplatně. Jak název jednotlivých předplatných napovídá, první varianta je určena především jednotlivcům, studentům nebo menším podnikům. Zbývající dvě varianty jsou navrženy pro potřeby větších firem (Salva, b.r.). Každý typ předplatného poskytuje různé úrovně funkcionality a přístupu k funkcím, ale všechna předplatná mají společný základ:

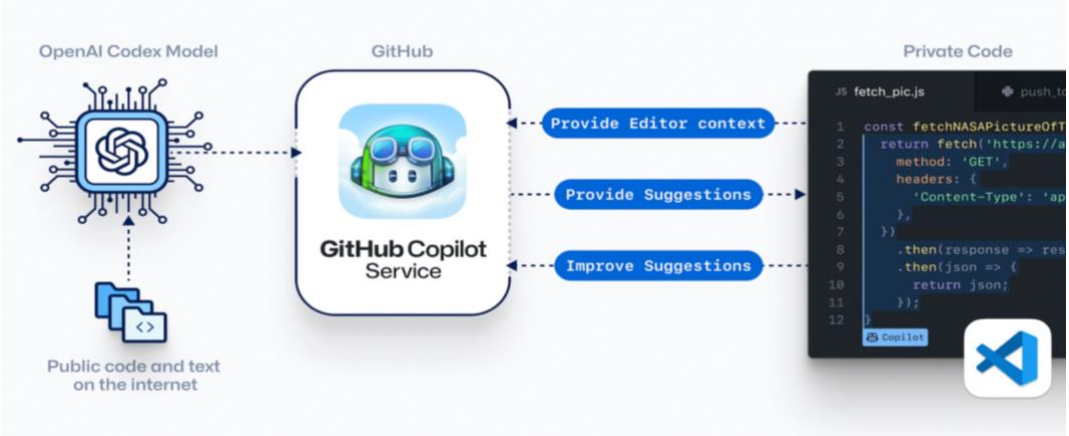
1. generování návrhů kódu v reálném čase při jeho psaní,
2. funkci chatu určenou především k pokládání otázek týkajících se vývoje v kontextu řešeného problému, ke generování kódu na základě zadaných požadavků a k úpravě již existujícího kódu,
3. možnost poskytnout pomoc při opravě nefunkčního nebo chybně napsaného kódu,
4. vysvětlení fungování částí kódu v projektu.

Mimo základní funkce vyšší předplatná nabízejí další možnosti, jako je generování souhrnu vytvářeného pull requestu, commit message a další funkce, určené především větším organizacím pro usnadnění jejich správy a auditovatelnosti používání nástroje.

Princip fungování

Pro zahájení generování odpovědi je nejprve nutné poskytnout nástroji podnět, který spustí tento proces. Podnětem může být zadání promptu při využití GitHub Copilot Chatu, nebo zahájení psaní kódu v otevřeném souboru v IDE při použití funkce dokončování kódu. Požadavek vyvolaný těmito akcemi je obohacen o kontext – například o informace z otevřených souborů v IDE, aktuální pozici kurzoru či jiná relevantní data z projektu (Prokopets, 2024). Následně je požadavek odeslán na server ke zpracování. Po zpracování je vygenerovaná odpověď odeslána zpět do uživatelova IDE. Uživatel může odpověď přijmout, použít ji jako základ pro další úpravy, nebo ji rozšířit. Při následných úpravách se proces popsaný v tomto odstavci znovu opakuje. Jednoduché schéma generování odpovědi je znázorněno na obrázek 1. Z něj je kromě samotné interakce s nástrojem a generování

odpovědí patrné, že veškeré zpracování promptu i generování odpovědi probíhá na serverech poskytovatele.



Obrázek 1: Fungování GitHub Copilot, zdroj: (Prokopets, 2024)

Podporované programovací jazyky a IDE

Nástroj podporuje mnoho populárních programovacích jazyků a frameworků, přičemž konkrétní počet těchto jazyků GitHub oficiálně neuvádí. Mezi nejvyužívanější jazyky v kombinaci s GitHub Copilotem patří JavaScript, Python, C#, Java a C++ (Zhang et al., 2023). Na vrcholu nejčastěji používané technologie se nachází Node.js. Co se týče typu úkolů, nejčastěji jsou asistenti využíváni programátory při implementaci funkcí pracujících s daty, ovládání frontendových prvků, zpracování textových řetězců a psaní testů pro naprogramovaný kód (Zhang et al., 2023).

Nabízené funkce v podporovaných jazycích lze využít v pěti hlavních vývojových prostředích, mezi které patří kromě Visual Studio Code i další prostředí: Visual Studio, Vim/Neovim, Azure Data Studio a vývojová prostředí od společnosti JetBrains - IntelliJ IDEA, WebStorm, PHPStorm, Android Studio a další (GitHub, Inc., b.r.).

Doporučené postupy pro používání

Jak již bylo zmíněno v předchozích kapitolách, společnost GitHub označuje svého asistenta kódování za nejrozšířenější nástroj pomáhající vývojářům doručovat kód rychleji a s lepšími výsledky. Nicméně jako všechny nástroje má i GitHub Copilot své slabé a silné stránky. Oficiální dokumentace nástroje podrobně popisuje různé scénáře, doporučení a způsoby použití, ve kterých by měl Copilot vynikat, a naopak situace, kdy je vhodnější zvolit alternativní přístup. Mezi hlavní přednosti GitHub Copilot patří:

- generování často se opakujícího kódu,
- vysvětlení a popis nejasných částí v kódu,
- psaní testů pro test-driven development,
- vytváření regulárních výrazů.

Na druhou stranu nástroj nedokáže odpovídat na otázky, které nesouvisejí s programováním nebo technologiemi. V potaz je nutné brát také rozdílné určení použití dvou základních částí, z nichž se nástroj obvykle skládá.

První částí je rozšíření pro našeptávání kódu, s nímž programátor interaguje v otevřeném souboru přímo během jeho psaní. Funkce je zvláště vhodná pro generování kratších implementací jednotlivých metod nebo tvorbu kódu na základě dostupných komentářů (GitHub, Inc., 2024).

Druhou částí nástroje je samostatný chat, který lze využít zejména v případech, kdy uživatel potřebuje vytvořit delší fragmenty, u nichž chce nástroji poskytnout větší kontext nebo chce zadání předat v přirozeném jazyce. Chat je rovněž vhodný pro scénáře, kdy programátor dopředu ví, že bude s asistentem výsledky dále diskutovat a upravovat je pomocí dalších promptů. Do chatovacího vlákna lze připojit i různé projektové soubory se zdrojovým kódem, které asistentovi umožní lépe porozumět aktuálnímu kontextu a generovat přesnější odpovědi s ohledem na aktuálně řešený problém. Kromě toho umožňuje chat i práci s tzv. personami. To znamená, že uživateli umožňuje definovat požadavky tak, aby je asistent řešil z pohledu určité osoby (např. seniorního vývojáře jazyka C++ z určitého oboru) (GitHub, Inc., 2024). V obou případech je klíčové formulovat komentáře a prompty co nejpřesněji. Kromě popisu principu fungování kódu se doporučuje doplnit prompt konkrétními příklady vstupů a výstupů, což zvyšuje přesnost a relevanci vygenerovaného řešení (GitHub, Inc., 2024).

Bez ohledu na to, jakou z výše popsaných částí GitHub Copilota programátor využívá, je vhodné rozdělit řešený problém na co nejmenší možné úkoly. Nástroj je výrazně efektivnější při řešení menších úkolů než při zpracovávání komplexních problémů. Tímto přístupem může vývojář navíc u každého dílčího úkolu specifikovat přesnější zadání a poskytnout nástroji detailnější kontext (Microsoft Corporation, 2025).

Pokud programátor stále nezískává uspokojivé odpovědi, další možností je přeformulovat svůj prompt nebo připojit k vláknu další relevantní soubory, které mohou nástroji poskytnout potřebný kontext. Dalším doporučením je zahájit nový chat v situacích, kdy není žádoucí navazovat na kontext předchozích dotazů, nebo když začíná programátor řešit nový problém (GitHub, Inc., 2024). Při práci s funkcí našeptávání kódu přímo během jeho psaní je vhodné mít otevřené pouze soubory související s aktuálním problémem a ostatní zavřít, aby se maximalizovala relevantnost návrhů. Copilot také často nabízí více návrhů řešení, mezi nimiž lze přepínat. Programátor by měl všechny dostupné návrhy pečlivě projít a vybrat ten nejvhodnější pro konkrétní situaci (GitHub, Inc., 2024).

Vývojář by si měl však neustále uvědomovat, že GitHub Copilot je pouze nástroj usnadňující práci, nikoliv nástroj přebírající jeho odpovědnost. Každý vygenerovaný návrh je proto nutné po jeho přijetí důkladně zkontrolovat a případně upravit tak, aby odpovídal aktuálním potřebám a požadavkům projektu.

1.1.3 Codeium[9]

Tento nástroj nabízí velmi podobné funkce jako GitHub Copilot, který byl popsán v předchozí podkapitole. Oproti němu však vyniká lepším zpracováním a uchováním

[9] https://codeium.com/

kontextu dotazů. Tuto vlastnost nevyzdvihují pouze jeho tvůrci, ale potvrzují ji i uživatelé v recenzích na tržišti s rozšířeními pro vývojářské prostředí Visual Studio Code (Microsoft Corporation, b.r.-b). Mnozí recenzenti pocházejí z řad vývojářů, kteří dříve používali GitHub Copilot, ale nebyli spokojeni s jeho schopnostmi. Zejména nebyli spokojení s jeho prací se zadávaným kontextem. Kromě lepšího zpracování kontextu a přesnějšího generování kódu uživatelé oceňují u asistenta Codeium také rychlejší odezvu a menší výskyt „halucinací", tedy situací, kdy nástroj poskytuje buď nesmyslné, nebo zcela smyšlené odpovědi.

Autoři nástroje se navíc zavázali, že základní předplatné zůstane pro jednotlivce vždy zdarma (Windsurf Team, 2023). Díky tomu je nástroj často vnímán vývojáři jako bezplatná a v mnoha ohledech adekvátní alternativa k již zmíněnému GitHub Copilotu. Navíc Codeium ve vyšších úrovních předplatného nabízí možnost provozu na vlastní infrastruktuře, což může být atraktivní volba pro týmy a organizace s vyššími požadavky na kontrolu dat a zabezpečení (Windsurf Team, 2023).

Interakce s nástrojem je v základní verzi možná nejen prostřednictvím rozšíření integrovaného přímo do IDE, ale také přes webové rozhraní. Co se týče podpory vývojářských prostředí, Codeium nabízí velmi podobnou podporu jako GitHub Copilot. Oproti němu však vyniká širší škálou podporovaných programovacích jazyků. Podle dostupných informací zvládá více než 70 programovacích jazyků, mezi které patří například Python, JavaScript, TypeScript, Java, Go nebo PHP (Exafunction, Inc., b.r.).

1.1.4 Tabnine[10]

Oproti dříve popsaným asistentům se Tabnine co do nabízených funkcí, možností interakce s uživatelem, podpory programovacích jazyků a vývojářských prostředí prakticky neliší. Tvůrci asistenta však kladou velký důraz na soukromí a bezpečnost dat svých uživatelů. To potvrzuje i získání certifikace SOC 2 Type 2 compliance (Tabnine Ltd., 2024). Součástí tohoto závazku je i pravidlo „*zero data retention policy*", které společnost zavazuje k nulovému uchovávání analyzovaných dat a zákazu jejich poskytování či sdílení třetím stranám (Kedar, 2024). Pokud by si potenciální zákazník přesto nebyl jistý bezpečností nástroje provozovaného v cloudu, má možnost provozovat vlastní instanci nástroje uvnitř své podnikové infrastruktury, která bude zcela izolována od vnějšího světa.

Kromě toho nástroj umožňuje administrátorům zvolit jazykový model, který bude generovat odpovědi. K dispozici mají správci několik různých modelů včetně dvou speciálně vytrénovaných společností Tabnine (Tabnine Ltd., b.r.). Pokud by tato nabídka nebyla dostatečná, každá společnost s patřičným předplatným si může integrovat upravený velký jazykový model přizpůsobený vlastním potřebám (Tabnine Ltd., 2025). I po napojení tohoto modelu je nástroj schopný plně analyzovat repozitáře uložené u různých poskytovatelů nástrojů pro správu verzí (source control management) bez omezení. Zabezpečení nástroje,

[10] https://www.tabnine.com/

dat a zdrojového kódu lze kromě globální úrovně nastavit i na úrovni týmů. V obou případech se dá specifikovat, jaká data budou použita k analýze a generování odpovědí.

1.2 Klíčové studie

V posledních měsících bylo publikováno velké množství prací zkoumající a zabývající se problematikou generativní umělé inteligence a jejího zapojení do různých pracovních prostředí. Cílem této podkapitoly je shrnout a popsat nejzásadnější myšlenky výsledků provedené rešerše již existujících zdrojů věnující se primárně tématice užití AI asistentů kódování. Rešerše a její výsledky poslouží dále jako jeden ze základů pro tvorbu a sestavování interview protokolu.

1.2.1 Vnímání asistentů kódování

V celosvětovém kontextu jsou nástroje využívající umělou inteligenci vnímány primárně jako asistenti usnadňující práci při vývoji softwaru. Vývojáři je nepovažují za přímou hrozbu pro svou profesi (Pan et al., 2024). Nicméně pro správnou adopci a co nejefektivnější využití těchto nástrojů však sami vývojáři zdůrazňují význam vhodného nastavení firemního prostředí, které podporuje sdílení zkušeností mezi zaměstnanci a motivuje je k využívání technologií umělé inteligence (Li et al., 2024).

Velkými otázkami ve vnímání AI asistentů kódování ze strany vývojářů ovšem stále zůstávají bezpečnost, soukromí, problematika duševního vlastnictví a nakládání s použitými daty ze strany poskytovatelů služeb. Zmíněné otázky často programátoři vnímají jako velmi důležité a jako jeden z důvodů, proč není adopce nástrojů ještě rychlejší (Arkheden & Eklund, 2024). Vývojářské společnosti by podle nich měly nástroje využívající umělou inteligenci aktivně začleňovat do běžné práce vývojářů a nebránit v jejich používání. Na druhou stranu je nutné, aby společnosti proaktivně nastavovaly vnitřní pravidla pro práci a nakládání s firemními daty. Ta by měla definovat povolené pracovní postupy a způsoby, kdy a s jakými daty lze asistenty kódování použít. Vývojáři existenci těchto zásad uvnitř firmy často oceňují a jsou si díky nim při používání nástrojů jistější (Pan et al., 2024). Definice podobných pravidel začíná být mezi většími firmami poměrně běžná. U menších bývá ovšem tato otázka často opomíjená nebo jí není přikládaná taková důležitost (Pan et al., 2024).

1.2.2 Způsoby využití

Zahraniční zdroje označují za hlavní přínos využívání asistentů kódování, zejména usnadnění a zrychlení práce vývojářů, spolu se snížením nároků na množství kapacit věnovaných danému problému (Bird et al., 2023). Díky tomu se programátoři mohou místo řešení opakujících se, či rutinních úkolů zaměřit na tvorbu a vývoj složitějších, klíčovějších a komplexnějších projektů.

Většina studií uvádí, že nástroj slouží jako velmi dobrý výchozí bod, který programátorům pomáhá zorientovat se v problematice, včetně vyhledávání relevantních zdrojů. Ve velké většině případů asistent kódování nahrazuje dosud běžné vyhledávání podkladů na

internetu (Vaithilingam et al., 2022). Arkheden a Eklund (2024) zjistili, že vývojáři asistenty rádi používají při práci se zdroji i z toho důvodu, že jim zdroje předem třídí a podává je v jasnější a přehlednější formě. Nástroje však neslouží pouze při práci se zdroji. Někteří vývojáři je využívají i během psaní kódu pro získání zpětné vazby a pro kontrolu kvality vytvořeného kódu ještě před jeho odesláním kolegům ke schválení (Mendes et al., 2024). Asistenti kódování jsou také často používáni při úpravách kódu, aby byl lépe formátovaný, efektivnější, lépe napsaný a přehlednější (Zhang et al., 2023). Při srovnání těchto postupů a praktik se studenty programování lze pozorovat velmi podobné vzorce chování a používání (Prather et al., 2023).

V kontextu učení s využitím nástrojů založených na generativní umělé inteligenci však programátoři stále zdůrazňují nezbytnost fundamentálních znalostí a principů v dané oblasti (Haque et al., 2024). Asistenti kódování mohou učení usnadnit především tím, že umožňují a usnadňují přepínání mezi syntaxemi různých programovacích jazyků a frameworků, aniž by bylo nutné jejich zdlouhavé studium. Nicméně kvalita a přesnost odpovědí generovaných AI se podle některých názorů výrazně odvíjí od znalostí vývojářů, kteří zadávají prompty (Haque et al., 2024). Znalosti programování a jeho základních principů tak zůstávají podle mnohých klíčovým předpokladem efektivní práce s asistenty kódování. Tento fakt současně vyvrací tvrzení, že by umělá inteligence byla schopná plně nahradit vývojáře, jejich dovednosti a zkušenosti.

1.2.3 Osvědčené postupy používání

V případech potřeby zajistit úspěšnou a fakticky správně vygenerovanou odpověď je vhodné zvážit některé z promptovacích strategií a doporučení pro používání asistentů kódování sestavených přímo jejich tvůrci. Zjištění, ke kterým jednotlivé studie dospěly, se často překrývají s doporučeními tvůrců asistentů kódování uvedenými dříve v této práci.

Vaithilingam et al. (2022) označuje dekompozici složitějších úkolů na menší a jednodušší celky často za klíčovou strategii pro dosažení úspěchu při používání asistentů kódování. Tím potvrzuje doporučení samotných tvůrců nástrojů, že menší úkoly jsou asistenti kódování schopni zpracovat mnohem lépe, což se potom pozitivně odráží na kvalitě výsledné odpovědi. Pan et al. (2024) přidává mezi účinné strategie také tu spočívající v přiřazování různých profesních osobností (person) generativní umělé inteligenci. Tato technika umožňuje vývojářům získat více návrhů řešení z různých úhlů pohledu, což přispívá k větší rozmanitosti výsledných odpovědí. Autoři stejné práce také dodávají, že je vždy dobré provést více iterací dotazu nad řešenou otázkou a zkusit získat více odpovědí.

Mendes et al. (2024) zmiňuje ve své práci poskytnutí dostatečného kontextu při generování odpovědí pomocí asistentů kódování jako jeden z klíčových faktorů ovlivňujících jejich správnost a kvalitu. Pokud nástroj opakovaně generuje nerelevantní odpovědi, doporučuje přehodnotit a upravit poskytovaný kontext. Zmiňuje, že právě nedostatečný nebo nepřesně definovaný kontext je často příčinou chyb a nepřesností v generovaných odpovědích. Kontext lze upravit například poskytnutím přístupu k dalším souborům v repozitáři, uzavřením souborů projektu nesouvisejících s řešeným problémem, nebo přepracováním samotného textu promptu. Mendes et al. (2024) dále upozorňuje, že problémy spojené s nedostatečným kontextem se mohou objevit v různých fázích vývoje. Častější ovšem bývají

v začátcích nového projektu, kdy se stává, že vývojáři nápovědu poskytnutou asistenty kódování kvůli její špatné kvalitě vypínají, neakceptují ji, nebo jim přijde dokonce vyrušující.

1.2.4 Dopady na vývojáře

Dopad AI asistentů kódování na produktivitu lze shrnout tak, že jejich přínos závisí na úrovni zkušeností vývojáře. Čím zkušenější programátor, tím méně mu tyto nástroje pomáhají při vytváření kvalitnějšího kódu (Ziegler et al., 2024). Nicméně i zkušení vývojáři oceňují, že jim nástroje pomáhají lépe se soustředit na podstatné úkoly, zatímco rutinní a repetitivní činnosti zvládnou dokončit rychleji. Výsledný kód pak při používání asistentů bývá čitelnější a efektivnější. Významnou přidanou hodnotou je rovněž pomoc při rychlejším osvojování nových programovacích jazyků (Ziegler et al., 2024). V určitých případech a za poskytnutí dostatečného kontextu komentářů přímo v kódu mohou být nástroje použity i pro odstranění existujícího technického dluhu. Klíčovým faktorem pro úspěšné dosažení požadovaného výsledku je však vhodná strategie tvorby promptů (OBrien et al., 2024).

I když objektivní rozdíly v hodnocení bezpečnosti výsledného implementovaného řešení s využitím asistentů kódování a bez nich nejsou zřejmé, 82 % vývojářů používajících GitHub Copilot označilo svou práci jako správnou a 58 % ji zároveň hodnotilo jako bezpečnou. Tato čísla jsou výrazně vyšší ve srovnání s hodnocením vývojářů pracujících bez asistentů kódování (Asare et al., 2024). Z uvedených údajů vyplývá, že jsou si vývojáři při navrhování řešení s asistenty jistější a mají větší důvěru v kvalitu svého výstupu.

V prvních verzích asistentů kódování vývojáři vnímali jako hlavní překážky v jejich používání několik omezení. Mezi nejčastěji uváděné patřil proces instalace, komplikace spojené s integrací rozšíření umožňujících práci s asistentem přímo ve vývojovém prostředí, snížení funkcionality ostatních rozšíření po instalaci asistenta a omezená délka generovaných odpovědí (Zhang et al., 2023). S novými verzemi asistentů kódování však uvedené problémy postupně ustupují.

1.2.5 Přesnost generovaných odpovědí a pochopení kódu vývojáři

Za klíčový problém vývojáři považují správnost generovaných odpovědí. Tento problém nabývá zásadního významu zejména při generování kódu, kde chybná odpověď na počátku může zásadně ovlivnit velkou část vývoje a může vést k neúspěchu nebo zdržení doručení výsledků vývoje (Arkheden & Eklund, 2024). S tím souvisí i zjištění, že vývojáři často nerozumějí generovanému kódu tak dobře, jako když jej napíší sami (Vaithilingam et al., 2022). Tento aspekt může vývojáře zavést do slepé uličky, kdy je nástroj navede k použití řešení, které ve výsledku nefunguje. Následkem toho tráví značné množství času pochopením, úpravou a laděním nefunkčního kódu, než kdyby si ho od začátku napsali sami (Prather et al., 2023).

K problematice správnosti generovaných odpovědí patří i nesoulad verzí používaných programovacích jazyků, frameworků nebo knihoven v odpovědi a samotném vývojovém projektu. Generované odpovědi mohou být z technického a faktického hlediska správné, ale

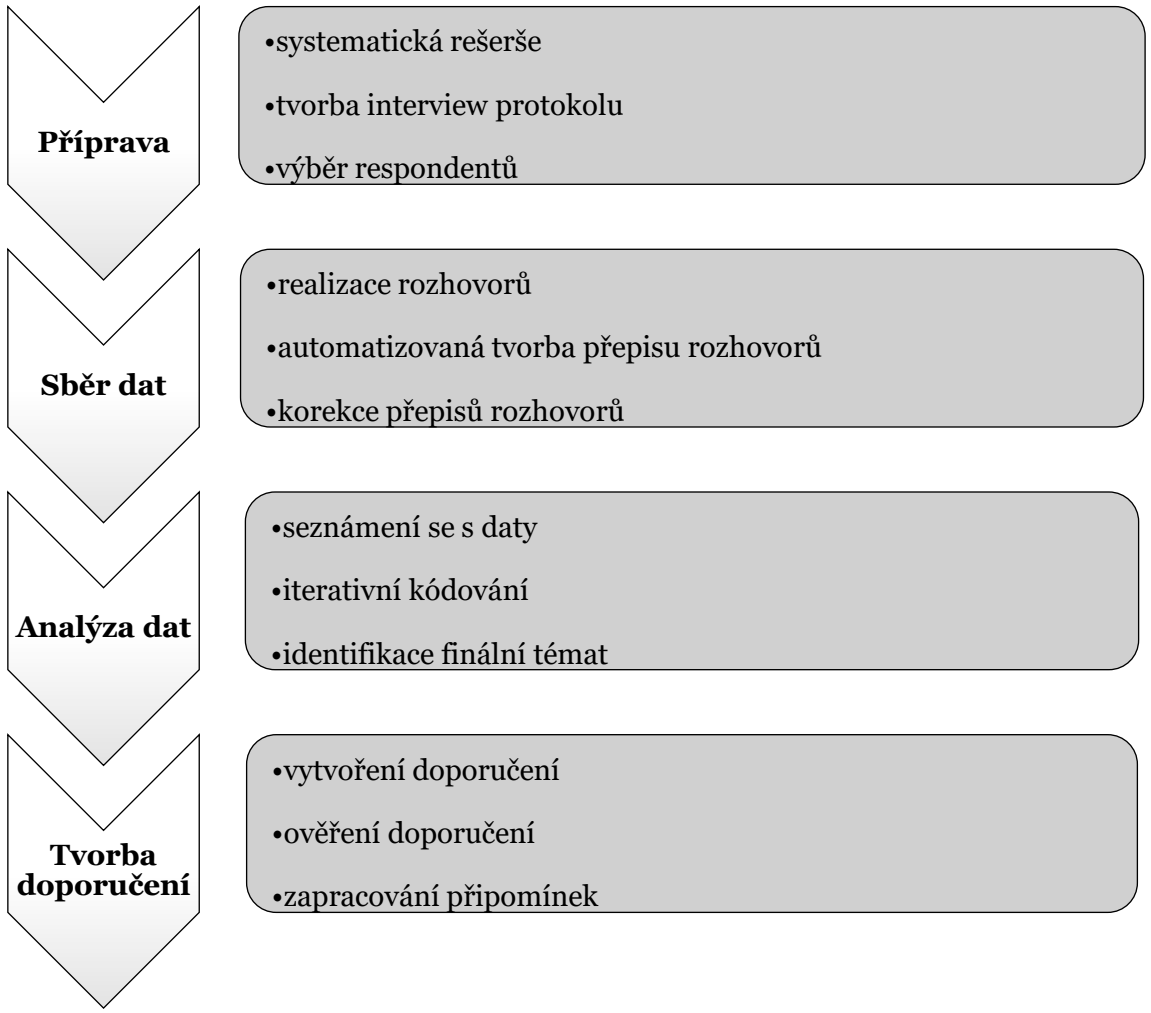
pokud nejsou kompatibilní s verzí používanou v konkrétním projektu, stávají se prakticky nepoužitelnými. Tento nesoulad často vede k nefunkčním řešením a vyžaduje od vývojářů další časově náročné úpravy (Mendes et al., 2024).

1.3 Shrnutí

V kapitole byl vytvořen přehled nejpoužívanějších AI asistentů kódování, popis jejich funkcí a osvědčené způsoby pro jejich používání. Kromě toho autor rovněž shrnul klíčové publikace, které se věnují aktuálnímu stavu poznání ve zkoumané oblasti. Při popisu současného stavu poznání kladl důraz především na vnímání asistentů kódování vývojáři, způsoby jejich využití, doporučené postupy při jejich používání, dopady na vývojáře a přesnost generovaných odpovědí.

2 Metody výzkumu

Cílem kapitoly je popsat jednotlivé metody použité k dosažení stanovených cílů a dílčích cílů diplomové práce. Použité metody lze rozdělit do čtyř fází. První fázi tvoří příprava výzkumu, na ní navazuje fáze sběru dat, následuje fáze jejich analýzy a celý proces uzavírá fáze tvorby doporučení. Podrobný popis jednotlivých metod je uveden v následujících podkapitolách. Pro lepší přehled o průběhu výzkumu připravil autor obrázek 2, na němž je zachycen celý proces výzkumu a jeho jednotlivé kroky.



Obrázek 2: Znázornění postupu výzkumu, zdroj: autor

První část této kapitoly je věnována popisu zvolené strategie systematické rešerše zdrojů, jejíž výsledky byly podrobně popsány a uvedeny v kapitole 1. Dále v kapitole následuje popis kontextu výzkumu. V rámci této podkapitoly autor popisuje kritéria výběru jednotlivých účastníků rozhovorů včetně požadavků na jejich odbornou kvalifikaci a zkušenosti s používáním AI asistentů kódování. Kapitola dále obsahuje postup tvorby interview protokolu, kde je popsána nejen jeho struktura, ale i postup a zdůvodnění výběru studií, z nichž autor při tvorbě interview protokolu vycházel.

Kromě kontextu výzkumu a tvorby interview protokolu je v kapitole popsán i typický průběh jednotlivých interview včetně popisu participantů a jejich zkušeností relevantních pro tuto práci. Následuje popis metod použitých při zpracování a přepisu získaných dat během interview. V této fázi je popsán proces od zpracování samotné nahrávky interview přes její přepis do textové podoby až po její manuální korekci.

Po přepisu získaných nahrávek se kapitola věnuje popisu další práce se získanými daty. Ten zahrnuje základní seznámení se s daty a jejich iterativní kódování podle metody Braunové a Clarkové (2021). Z iterativního kódování získaných dat následně vzešla ústřední témata práce, která jsou prezentovaná a interpretovaná v kapitole 3.

Poslední část této kapitoly se věnuje popisu použitých metod vedoucích k naplnění čtvrtého vytyčeného dílčího cíle. V rámci něj měly být získané poznatky využity pro vytvoření sady doporučení umožňující co nejefektivnější využití a implementaci AI asistentů kódování. Vytvořená doporučení byla nakonec ověřena s vybranými participanty rozhovorů a získaná zpětná vazba byla následně zapracována do finální verze doporučení. Ta je k dispozici v kapitole 5.

2.1 Rešerše zdrojů

V prvé řadě provedl autor systematickou rešerši aktuálního stavu poznání. Ta byla provedena primárně nad databází ACM Digital Library[11]. Pro systematickou rešerši nad databází ACM byl využit výchozí vyhledávací řetězec *„[All: coding assistant] AND [All: developers] AND [[All: work experience] OR [All: methods of use] OR [All: performance impact] OR [All: best practice]] AND [E-Publication Date: Past 2 years]*“ s časovým omezením publikace na poslední dva roky. Spolu s časovým omezením byl použit také filtr „*Research Article*“ pro typ záznamu v databázi. Vzhledem k velkému množství nalezených výsledků byl řetězec dále modifikován tak, aby bylo množství výsledků sníženo.

Snížení celkového počtu výsledků autor dosáhl tak, že řetězec rozdělil na dílčí dotazy spouštěné samostatně. Jejich výsledky se podle očekávání v určitých fázích rešerše částečně překrývaly. Většina z nich byla ovšem pro vyhledávané řetězce unikátní. V této fázi autor také vyřadil duplicity a články, které již podle názvu neodpovídaly požadavkům na téma. Do zúženého výběru se takto dostalo okolo 250 unikátních výsledků. Následovala fáze screeningu. V té byly zbylé články posuzovány vždy minimálně podle abstraktu, u studií kvantifikující určitý jev byl procházen i závěr. Do konečného výběru klíčových zdrojů byly vybrány pouze články, u nichž bylo již z názvu a abstraktu patrné, že mezi vyhledaným výsledkem a vytvářenou diplomovou prací autora existuje tematický průnik. Tedy že se jedná o zdroj zaměřený na asistenty kódování, očekávání vývojářů od těchto nástrojů, způsoby jejich používání nebo na dopady, které mají tyto nástroje na práci vývojářů. V úvahu byly brány pouze zdroje, u nichž se jednalo o hlavní zaměření práce. Články věnující se této problematice pouze okrajově autor do finálního výběru nezahrnoval. Druhá fáze

[11] https://dl.acm.org/

rešerše obsahovala tradiční rešerši zdrojů. K těmto účelům byl využit vyhledávač Google Scholar[12], databáze ProQuest Central[13]a databáze vysokoškolských kvalifikačních prací Theses.cz[14].

2.2 Kontext výzkumu

Pro naplnění definovaného cíle práce autor vybíral výhradně vývojáře, kteří již měli předchozí zkušenosti s AI asistenty kódování v kontextu vývoje softwaru. I když byl při výzkumu kladen důraz primárně na pracovní prostředí, mohli participanti rovněž sdílet zajímavé poznatky z využití asistentů kódování mimo svůj profesní rámec. Autor při výběru účastníků dopředu nekladl žádné podmínky ohledně zkušeností s konkrétním produktem. I přes to měla nakonec velká část vývojářů zkušenosti primárně s nástrojem GitHub Copilot.

S ohledem na dosažení co nejpřesnějších výsledků byli do výzkumu zahrnuti vývojáři z různých věkových skupin, s různou úrovní zkušeností a zaměřením vývoje. Autor vybíral jak nezávislé kontraktory spolupracující se společnostmi na vývoji softwaru, tak i vývojáře na hlavní nebo vedlejší pracovní poměr. U vedlejšího pracovního poměru bylo podmínkou, aby participant věnoval programování alespoň 80 hodin měsíčně, což odpovídá práci na poloviční úvazek.

Vybraní respondenti nebyli ve většině případů součástí stejného vývojového týmu. Velikost týmů se pohybovala přibližně mezi 10 až 15 členy. Do celkového počtu nebyli započítáváni pouze programátoři, ale i ostatní členové týmu. Účastníci rozhovorů pracovali podle klasifikace Evropské komise (2021) pro dva velké podniky (korporáty) působící v oblasti bankovnictví. Obecně lze ovšem říci, že se jednalo o programátory působící v agilních týmech řídících se metodikou Scrum nebo její mírně upravenou verzí.

2.3 Tvorba interview protokolu

Na základě provedené rešerše zdrojů autor připravil interview protokol. Tvorba interview protokolu probíhala v několika krocích. Prvním bylo rozdělení interview do základních tematických celků, které kromě úvodní a závěrečné části odpovídají stanoveným rešeršním otázkám. Část otázek byla převzatá ze zahraničních výzkumů, část přidal autor práce na základě rešerše existující literatury. Následně proběhl výběr a úprava nejvhodnějších otázek. Nakonec autor upravené otázky zařadil do tematických celků. Finální struktura interview protokolu má následující podobu: (1) úvod, (2) očekávání, (3) použití, (4) dopad a (5) závěr.

Jako stěžejní výzkum použitý pro tvorbu interview protokolu autor vybral kvalitativní studii zabývající se využitím AI asistentů kódování ve vztahu k bezpečnosti vyvíjeného softwaru

[12] https://scholar.google.com
[13] https://proquest.com
[14] https://theses.cz

(Klemmer et al., 2024). V té její autoři zkoumali 3 klíčové oblasti: (1) využití generativní AI při vývoji softwaru, (2) dopady generativní AI na bezpečnost vyvíjeného softwaru, (3) budoucnost generativní AI při vývoji softwaru. Otázky ze studie nebyly použity při vytváření vlastního interview protokolu v plném rozsahu. Autor čerpal primárně z oblastí 1 a 3. Otázky týkající se dopadů umělé inteligence na bezpečnost vyvíjeného softwaru nepoužil, jelikož byly velmi podrobné. Místo nich v interview protokolu formuloval jednu obecnější týkající se tohoto tématu. Otázky ohledně bezpečnosti uvedené v původní studii ho ovšem inspirovaly a využil je v upravené formě pro zjištění dopadů umělé inteligence na vývoj softwaru obecně. Protokol doplnil ještě o otázky týkající se očekávání, pocitů a dopadů AI asistentů kódování na práci vývojářů. Tento pohled byl v první studii zkoumán pouze v omezené míře, proto se autor rozhodl použít pro tvorbu protokolu ještě švédskou práci zabývající se užitím umělé inteligence při vývoji softwaru (Arkheden & Eklund, 2024), v níž byl tento druh otázek zastoupen více.

Převzaté otázky z obou studií byly původně v anglickém jazyce, proto je autor pro účely této práce přeložil. Překlad nebyl u všech otázek doslovný a u některých bylo mírně pozměněno původní znění. Kompletní interview protokol je uveden v příloze A. Příloha B pak obsahuje reference jednotlivých otázek na zdrojovou literaturu.

Po navržení hlavních otázek interview protokolu byla na jeho úvod přidána ještě sada otázek, která má za úkol představit samotného účastníka, jeho zkušenosti, kontext pracovního týmu a obecně jeho vnímání zkoumané problematiky. Celý protokol pak uzavírají shrnující otázky dávající účastníkovi možnost zdůraznit klíčové poznatky, které během používání asistentů kódování nabyl a které mu přišly pro jejich správné používání nejdůležitější. Tato závěrečná část následně poslouží jako odrazový můstek pro tvorbu sady doporučení, jež jsou součástí hlavního cíle této práce.

Před zahájením provádění jednotlivých interview byla finální verze protokolu podrobena pilotnímu interview. To bylo provedeno s osobou P0 splňující dříve definované požadavky na participanty. Podrobnější údaje o pilotním participantovi včetně jeho zkušeností obsahuje tabulka 1. Pilotní interview sloužilo čistě k ověření struktury otázek, logičnosti tematických celků, jejich návaznosti, jasnosti pokládaných otázek a délky rozhovoru. Jeho výstupy nebyly do samotné práce zahrnuty. Výsledek ukázal, že je protokol navržený dobře a lze ho bez problémů použít pro další rozhovory. Před prvním ostrým interview provedl autor pouze drobné stylistické úpravy u některých z otázek. Tato změna ovšem neměla dopad na jejich význam nebo znění.

2.4 Průběh sběru dat

Jak vyplývá z předcházejících kapitol, stěžejní metodou práce byl sběr dat pomocí kvalitativních rozhovorů vedených polostrukturovanou formou podle předem připraveného interview protokolu. S ohledem na dosažení co nejpřesnějších výsledků byl kladen důraz na výběr vývojářů napříč různými věkovými kategoriemi, s různou úrovní zkušeností a zaměřením. Kontakty na participanty P1 až P3 byly s jejich svolením získány v rámci pracovního prostředí. Předání kontaktů na další potenciálně vhodné účastníky probíhalo primárně přes participanty, kteří se interview již zúčastnili. Kromě pilotního účastníka, na

němž bylo provedeno ověření vytvořeného protokolu, bylo vybráno dalších devět vhodných účastníků. Informace o nich znázorňuje tabulka 1.

Tabulka 1: Informace o účastnících výzkumu, zdroj: autor

ID účastníka	Zaměření	Zkratka zaměření	Praxe (roky)	Zkušenosti s coding asistenty (roky)	Délka rozhovoru (min.)
P0	Frontend	FE	3	2	77
P1	Frontend	FE	1,5	4	65
P2	Backend	BE	1,5	2	67
P3	Fullstack	FS	6	2	90
P4	Backend	BE	20	1	83
P5	Backend	BE	25	1	65
P6	Backend	BE	11	1,5	75
P7	Fullstack	FS	8	1	70
P8	Frontend	FE	11	2,5	60
P9	Fullstack	FS	5	2	80

Pro snadnější orientaci a odkazování na jednotlivé účastníky výzkumu dále v práci zavedl autor vlastní označení účastníků. To se skládá z dat uvedených v tabulka 1 následujícím způsobem: „*ID účastníka-zkratka zaměření*“. Označení pro účastníka výzkumu P1 by vypadalo následovně: „P1-FE“.

Participanti měli před konáním interview možnost výběru, zdali jim více vyhovuje provést rozhovor na online platformě Zoom[15], nebo preferují klasické osobní setkání. Pokud daný participant neměl preferenci, autor práce upřednostňoval osobní setkání. Tuto možnost volil zejména kvůli lepší interakci s participantem a lepšímu zachycení některých neverbálních reakcí. Online nakonec proběhlo pět interview, zbylé čtyři proběhly osobně. Samotný termín byl s účastníky domlouván individuálně podle jejich potřeb. Před rozhovorem byla participantovi zaslána zpráva s podrobnostmi o schůzce, jejím průběhu a se základními informacemi o vytvářené práci.

[15] https://zoom.us

Na začátku každého rozhovoru autor participantovi zopakoval cíl výzkumu, práva participanta během rozhovoru a představil strukturu interview protokolu. Schůzka byla nehledě na její formu se souhlasem participanta nahrávána. Nahrávání bylo prováděno kvůli možnosti zpětného vytvoření přepisu jednotlivých rozhovorů. Kromě nahrávání si autor práce dělal v průběhu i vlastní poznámky, které následně spolu s přepisy využil jako podpůrný materiál při vyhodnocování rozhovorů.

2.5 Analýza dat

Před samotným zahájením analýzy dat bylo nutné pořízené audionahrávky převést na text. Toho bylo docíleno použitím softwaru pro převod audia do textové podoby. Pro výběr nejvhodnějšího způsobu převodu vyzkoušel autor několik modelů a nástrojů. Mezi ně patří OpenAI Whisper[16], Microsoft Azure AI Speech[17], Beey.io[18], Descript[19], Otter.ai[20], Sonix[21] a Happy Scribe[22]. Při výběru nejvhodnějšího nástroje zohledňoval především kvalitu přepisu audionahrávky do českého jazyka. Kvalitou výsledného přepisu má autor na mysli zejména přesnost vytvářeného přepisu, výsledné formátování textu a množství gramatických nebo stylistických chyb. Dále při výběru zohledňoval i uživatelskou přívětivost, maximální délku přepisované audionahrávky nebo cenu a dostupnost služby. Pro sledování zmíněných parametrů a kvality přepisu využil krátký (přibližně patnáctiminutový) úsek nahrávky z pilotního interview.

Na základě provedených převodů zkušební nahrávky byl vybrán vzhledem k dostupnosti a kvalitě výsledného přepisu open source model OpenAI Whisper s UI nástavbou umožňující práci na vlastním počítači. I přes to, že měl nástroj nejpřesnější přepis záznamu do českého jazyka, vždy bylo nutné vygenerovaný přepis ručně projít a provést korekci. Ta zabrala v průměru přibližně 2 hodiny pro jeden rozhovor.

Odpovědi participantů byly následně tematicky analyzovány podle metody Braunové a Clarkové (2021), která zahrnuje šest klíčových fází. Prvním krokem je *detailní seznámení* se s daty prostřednictvím jejich opakovaného čtení a poznámkování prvních myšlenek. Následuje *proces kódování*, během kterého jsou klíčové charakteristiky systematicky označovány kódy, jenž reflektují relevantní zjištění ve vztahu k výzkumným otázkám. Ve třetí fázi jsou kódy *seskupovány do širších kategorií*, které tvoří základ pro identifikaci potenciálních témat. Potenciální témata jsou následně revidována a upravována tak, aby byla zajištěna jejich logická struktura. Poté je *definována, pojmenována a připravena pro interpretaci finální sada identifikovaných témat*. Posledním krokem je *sepsání analýzy*

[16] https://platform.openai.com/docs/guides/speech-to-text
[17] https://azure.microsoft.com/cs-cz/pricing/details/cognitive-services/speech-services
[18] https://www.beey.io
[19] https://www.descript.com
[20] https://otter.ai
[21] https://sonix.ai
[22] https://www.happyscribe.com

popisující identifikovaná témata a poskytující jejich interpretaci podloženou konkrétními výpověďmi jednotlivých účastníků.

Analýza dat začala prvním krokem, v němž se autor seznamoval se získanými daty. Kromě samotného pročítání probíhalo také zvýrazňování zajímavých částí a tvrzení v rozhovorech spolu s tvorbou poznámek. V tomto kroku se ještě nejednalo o klasické kódování, proto v rámci kroku nebyl použit žádný specializovaný nástroj podporující kódování nebo analýzu dat, ale byly použity nástroje jako Microsoft Word a nástroje pro tvorbu poznámek. Po dokončení prvního kroku již následovala hlavní část analýzy dat. Ta byla pro usnadnění prováděna ve specializovaném nástroji MAXQDA[23].

V první iteraci provedl autor kódování podle oblastí zvýrazněných v minulém kroku. Jednotlivé rozhovory byly analyzovány postupně a průběžně podle pořadí, v němž byly pořízeny – nebylo volené žádné speciální pořadí. V rámci první iterace bylo vytvořeno celkem 78 kódů v celkem šesti oblastech zaměření. Pro lepší přehlednost byla každá oblast v nástroji MAXQDA označená vlastní barevnou kategorií.

V rámci druhé iterace bylo využito poznámek, které byly zaznamenávány během provádění jednotlivých interview. Během této iterace autor již vytvořené kódy průběžně přetvářel. Přitom se podrobněji zaměřoval na části rozhovorů, u nichž měl napsané poznámky. Na konci druhé iterace se pokusil i o první agregaci vytvořených kódů do širších kategorií.

Následovalo ještě několik iterací, během kterých docházelo k průběžnému vytváření, odstraňování, upravování a přeskupování kódů nebo kategorií kódů tak, aby tvořily co nejvíce tematicky soudržné celky. Po poslední iteraci vznikl finální počet kategorií kódů, z nichž vzešla celkem čtyři ústřední témata.

Při použití citací participantů dále v kapitole 3 bylo do původního textu zasahováno co nejméně. Pokud byl text upraven, bylo to z důvodu jeho lepší pochopitelnosti a přehlednosti. Dále byly do textu přidány v hranatých závorkách slova, která byla během rozhovoru zřejmá, ale v samotném textu by nemuselo být jasné, o čem participant mluví. Nespisovné výrazy autor práce v citacích ponechával. V případě potřeby vypuštění řady slov použil autor tři tečky.

2.6 Tvorba a ověření sady doporučení

Na základě zjištění získaných analýzou provedených rozhovorů byla formulována sada doporučení, která reflektuje nejdůležitější identifikovaná témata. První verze byla validována s pěti participanty, kteří se účastnili rozhovorů provedených dříve v rámci výzkumu. Výběr participantů probíhal tak, aby pokryl co nejširší škálu programátorů – od juniorů až po seniory a při tom zohledňoval pestré zaměření vývojářů a délku jejich zkušeností s AI asistenty kódování. Cílem ovšem nebylo zopakovat rozhovory se všemi

[23] https://maxqda.com

devíti participanty. Proto byli vybráni participanti P1, P3, P5, P7, P9. Podrobnosti o délce jejich praxe, zaměření a zkušenostech s AI asistenty kódování jsou popsány v kapitole 2.4.

Vybraní participanti byli osloveni s žádostí o vyjádření a poskytnutí zpětné vazby k vytvořené sadě doporučení. Všichni oslovení s poskytnutím zpětné vazby souhlasili. Po zaslání se seznámili s pracovní verzí a poskytli k ní komentáře a zpětnou vazbu k zapracování. Zpětná vazba byla sbírána formou krátkých (přibližně desetiminutových) rozhovorů. Rozhovory probíhaly dle možností participantů buď online, nebo fyzicky a neměly dopředu stanovenou konkrétní strukturu. Jejich cílem bylo postupně projít všechna doporučení, ke kterým měli participanti komentáře a získat od nich co nejvíce relevantních informací. Získaná zpětná vazba byla následně zapracována a na jejím základě vznikla finální sada doporučení popsaná v kapitole 5.

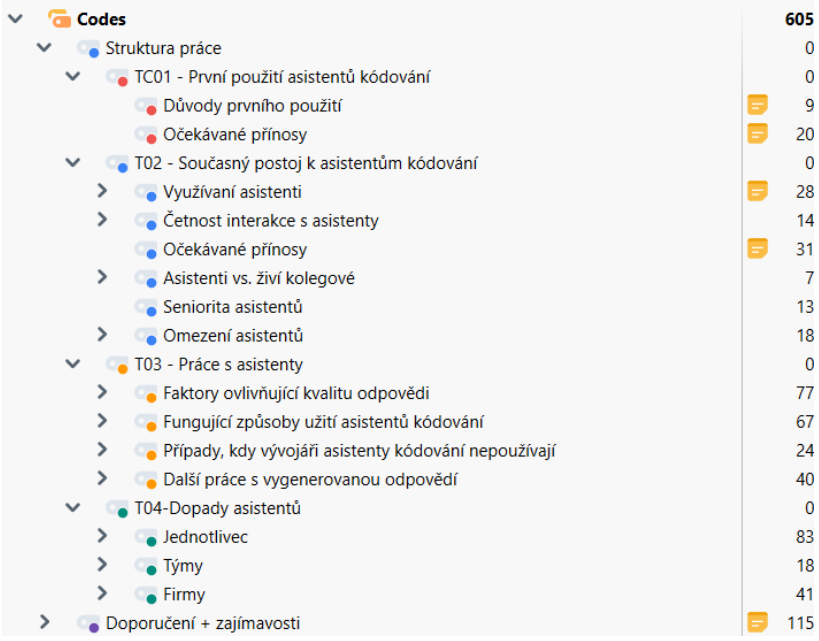
2.7 Shrnutí

V kapitole byl popsán průběh provedeného kvalitativního výzkumu spolu s metodami zvolenými pro naplnění vytyčených cílů diplomové práce. Jednotlivé podkapitoly se postupně zabývaly zvolenou strategií pro systematickou rešerši zdrojů, specifikovaly konkrétní kontext výzkumu a proces tvorby interview protokolu. Dále v nich autor popsal výběr účastníků prováděných rozhovorů, průběh sběru dat a v neposlední řadě následoval detailní popis analýzy získaných dat. Celou kapitolu uzavírá popis způsobu, jakým byla sada doporučení reflektující identifikovaná témata vytvořená, jak ji autor ověřoval a jak zapracovával získanou zpětnou vazbu.

3 Výsledky výzkumu

Cílem kapitoly je prezentovat výsledky a zjištění získané během provedených polostrukturovaných rozhovorů podle interview protokolu, jehož sestavení se věnuje kapitola 2.3., a jejich následné analýzy. Kapitola je strukturována podle identifikovaných témat vzešlých z analýzy získaných dat, jejíž postup je popsán v kapitole 2.5. Témata kromě definovaných otázek v úvodu práce přinášejí také zajímavé informace a zjištění, na které se autor při tvorbě interview protokolu tolik nesoustředil. Vzhledem k jejich relevantnosti a informační hodnotě je ovšem do této kapitoly také zahrnul. Kromě toho se jednalo o informace a zjištění, které se následně promítly do sady doporučení uvedené v kapitole 5. Četnost výskytu jednotlivých ústředních témat a podtémat v provedených rozhovorech pak podrobněji ukazuje podkapitola 3.5.

Obrázek 3 poskytuje základní přehled identifikovaných témat vzešlých z provedených rozhovorů skládající se z dílčích kódů. Kompletní přehled identifikovaných kódů v plném rozsahu je přiložen v příloze C.



Obrázek 3: Přehled identifikovaných témat, zdroj: autor

Z tematické analýzy provedených rozhovorů vzešla celkem 4 hlavní témata. Každému je věnovaná samostatná podkapitola, kde je téma podrobněji popsáno spolu se souvisejícími podtématy.

První identifikované téma se zaměřuje na počáteční postoje a vnímání AI asistentů kódování z pohledu vývojářů. Druhé popisuje, jak programátoři v současné době asistenty kódování vnímají. Navazující téma se věnuje konkrétním způsobům, jak vývojáři nástroje využívají v každodenní praxi. Zahrnuje faktory ovlivňující přesnost a kvalitu vygenerované odpovědi a popis strategií pro získání co nejpřesnějších odpovědí. Zabývá se také úkoly, pro

které použití asistentů kódování není vhodné, a způsoby, jak vývojáři dále nakládají s vygenerovanou odpovědí. Poslední identifikované téma popisuje dopady využívání AI asistentů kódování z pohledu programátorů jako jednotlivců, ale částečně se věnuje i dopadům na jejich týmy a firmy. Mimo jiné zahrnuje přínosy i možná rizika vyplývající z integrace těchto nástrojů do vývojového procesu.

Pro snadnější referenci na jednotlivá témata, na něž nebude chtít autor dále v textu odkazovat celým názvem, bylo každé téma označeno zkratkou T a číslem odpovídajícímu pořadí, v jakém téma vzešlo z provedené analýzy. Pro první identifikované téma tak bude použita reference „*T01*“ nebo první téma. Tabulka 2 obsahuje kompletní výčet identifikovaných témat z provedené analýzy.

Tabulka 2: Identifikace témat vzešlých z analýzy, zdroj: autor

Název identifikovaného tématu	Pořadové číslo tématu	Označení zkratkou
Přijetí asistentů kódování	1	T01
Současný postoj k asistentům kódování	2	T02
Práce s asistenty kódování	3	T03
Dopady asistentů kódování	4	T04

3.1 Přijetí asistentů kódování

Prvním ústředním tématem, které vzešlo z analýzy provedených rozhovorů, se zaměřuje na vnímání asistentů kódování vývojáři při jejich prvním použití. Popisuje důvody, které vedly programátory k rozhodnutí asistenty začít při práci využívat a očekávání spojená se zahájením používání této technologie.

3.1.1 Důvody prvního použití

Všichni účastníci rozhovorů se nezávisle na sobě shodli, že hlavním důvodem k vyzkoušení asistentů kódování byla zvědavost a snaha zjistit, jaké možnosti tyto nástroje nabízejí. Mnozí o nich měli povědomí již dříve, avšak chyběl jim konkrétní podnět nebo důvod k jejich aktivnímu využití. Níže jsou uvedeny klíčové faktory, které účastníky přesvědčily k zahájení práce s asistenty kódování.

Participanta P2-BE zaujala na asistentech schopnost a kvalita našeptávání kódu. Ve srovnání s dosud dostupnými statickými našeptávači poskytoval pro jeho konkrétní potřeby asistent výrazně lepší výsledky.

„Ze začátku to určitě byla zvědavost. Já jsem prostě viděl, že to bylo něco hodně nového. Hlavně oproti již existujícím statickým doplňovačům mi tohle přišlo jako hodně velký skok, tak si to člověk chtěl vyzkoušet.“ P2-BE

Pro P3-FS nebyla generativní umělá inteligence ničím novým, jelikož se s ní již setkal v různých jiných oblastech. Ovšem až jedna z konverzací s obecným AI nástrojem ho příjemně překvapila svou odpovědí natolik, že se rozhodl ji začít využívat i při vývoji softwaru ve formě asistentů kódování.

„Řekl jsem si, tak jdu to vyzkoušet a má první otázka do Chatu GPT byla, jestli zná zásady robotiky od Isaka Asimova, a začala zajímavá filozofická diskuze se strojem. Následně jsem si řekl, když to dokáže takto odpovídat na humanisticko-filozofické otázky, tak to přeci musí velice relevantně poskytovat odpověď pro programování, které je mnohem jednodušší, pravidla, sémantika a syntaxe jazyka je jasně daná.“ P3-FS

Pro participanta P4-BE bylo klíčovým impulzem k zahájení používání asistentů jejich zavedení do firemního prostředí jeho zaměstnavatelem. Před tímto krokem byl ke schopnostem asistentů kódování spíše skeptický. Teprve s postupným experimentováním si začal uvědomovat jejich možnosti a přínosy.

„…nejdřív jsem tomu nevěřil. Potom, jak jsme byli v podstatě vybídnutí [vedením firmy kde pracuje], abychom to vyzkoušeli sami, tak jsem si to nainstaloval, začal jsem to používat a začal jsem zjišťovat, co to všechno umí.“ P4-BE

Participant P5-BE uvádí jako prvotní impuls k používání obecného nástroje ChatGPT běžnou konverzaci s přáteli, kteří se rovněž pohybují v oblasti vývoje softwaru a s nástrojem již měli zkušenosti. Hlavním podnětem k aktivnímu využívání asistentů kódování však pro něj, stejně jako pro participanta P4-BE, bylo až jejich zavedení zaměstnavatelem. Z doslechu se o asistentech kódování dozvěděl i participant P8-FE. V jeho případě se ale nejednalo o informace přímo od známých, ale z videí na platformě YouTube[24].

„Ukázali mi [kamarádi] to [Chat GPT], pak jsem si s tím začal hrát. To samé ten Copilot. Zaměstnavatel ho sem zavedl, udělal nějaké promo, no a začal jsem ho zkoušet a používat tak, aby mi to vyhovovalo. P5-BE

„Já hodně sleduju moderní trendy, jsem velký YouTube konzument, a právě že takoví ti nejznámější TypeScript influenceři prostě říkali, není důvod nepoužívat Copilota, pokud to vlastně dovolí ve firmě. V tu chvíli jsem si říkal, proč to nevyzkoušet?“ P8-FE

P6-BE se rozhodl asistenty vyzkoušet, protože klasické nástroje využívané pro řešení problémů mu nebyly schopné dát uspokojivou a kvalitní odpověď pro jeho specifický problém. Navíc už podle svých slov vyčerpal i ostatní možnosti, které při řešení problémů běžně používal.

[24] https://youtube.com

„Já jsem je začal používat, když jsem už fakt nevěděl kudy kam a Google nebo Stack Overflow mi nebyl schopný pomoct s tak konkrétně zacíleným problémem, který nějakým způsobem nebyl úplně standardní.“ P6-BE

3.1.2 Očekávané přínosy

Mnoho účastníků rozhovorů k nástrojům zpočátku přistupovalo spíše zdrženlivě a neměli od nich velká očekávání. Nejprve si je potřebovali vyzkoušet, prozkoumat jejich možnosti a najít optimální způsob, jak je co nejefektivněji využít při řešení svých problémů. Jakmile se seznámili se schopnostmi asistentů, jejich pohled na tyto nástroje se postupně začal měnit, což dobře shrnují participanti P1-FE a P5-BE. Navzdory tomuto posunu však mezi vývojáři stále přetrvává zdravá míra ostražitosti, jak zmiňuje participant P4-BE. Participant P2-BE se explicitně nevyjadřuje k původním očekáváním, ale zdůrazňuje, že od asistentů neočekával, že by jej v jeho práci nahradili nebo ji vykonávali za něj.

„Na začátku od toho člověk neměl žádné očekávání, protože přišla nová featura, která dokáže doplňovat kód a člověk byl nadšenej, že to doplní jednoduchý for cyklus. Ale teď už ty očekávání mám logicky větší. Zvykla jsem si hlavně na komfort toho, že mám toho Copilota.“ P1-FE

„Očekávání nikdy nebyla velká. Na začátku to byla zvědavost, co to dokáže. Když zjistíš, co to dokáže, tak zkoušíš, kde jsou až jeho hranice, nebo jestli dokáže, co tě zajímá. Pak když zjistíš, že je v něčem dobrej, tak ho začneš používat a hledat si ty svoje cestičky.“ P5-BE

„Na začátku to bylo jako ‚no uvidíme, nevěřím tomu‘, ale teď už je to lepší. Ale samozřejmě ten kód, který dostanu od asistenta, tak se na něj nejdřív musím podívat, vyzkoušet si, jestli to funguje. Prostě slepě tomu nevěřím.“ P4-BE

„…já jsem od toho nikdy upřímně neměl takové velké očekávání. Nejsem schopnej říct, že jsem od toho konkrétně něco očekával, ale povím ti, že jsem určitě nečekal, že to tu práci bude celou dělat za mě.“ P2-BE

Konkrétní očekávání od používání asistentů kódování poté, co si na něho zvykli, vyjádřili participanti P1-FE, P6-BE, P7-FS a P9-FS. Participantka P1-FE očekávala, že jí asistent přinese úsporu v množství psaného kódu. Naopak participant P6-BE očekával spíše větší personalizaci odpovědí s ohledem na aktuálně řešený problém a rozšířené vyhledávání informací a zdrojů. Mezi rozšířené možnosti vyhledávání, které vnímal jako velkou přidanou hodnotu oproti běžným vyhledávačům, patřilo zejména doplňování dodatečných instrukcí a kontextu problému přímo během vyhledávání.

„Velkej potenciál byl v tom, že člověk nemusel toho kódu tolik psát. Hlavně toho opakujícího se.“ P1-FE

„Když dobře instruuješ toho chatbota, tak je schopný ti najít řešení na míru. Hloubka personalizace a přizpůsobení se tomu problému je něco, co ti Google nikdy pořádně neudělá. Mimochodem i možnost dopovídávat [doplňovat] mu další instrukce nebo kontext za běhu je taky docela klíčový, protože to u normálních vyhledávačů uděláš jen těžko.“ P6-BE

Mezi očekávané přínosy participanta P7-FS patřilo generování unit testů, revize kódu a kontrola chyb při jeho psaní. Přestože generování unit testů je podle ostatních účastníků jedním z nejčastějších způsobů využití asistentů, v jeho případě se tento přístup neosvědčil. Stejně tak ani revizi kódu nezařadil mezi své způsoby využívání, jelikož ve firmě mají tento proces pokrytý jinou automatizací během buildu aplikace. Za nejpřínosnější označil nakonec participant využití asistenta pro konzultaci problémů, přičemž tuto roli sám označil jako tzv. ‚roli gumové kachničky'. Tento přístup mu pomáhá lépe si sumarizovat a strukturovat myšlenky a dojít k řešení, které by mu jinak mohlo uniknout. Pokud samotná sumarizace nestačí, často mu pomohou odpovědi nebo zdroje, které asistent poskytne.

> *„Sliboval jsem si od toho, že mi to pomůže psát třeba unitesty nebo něco takového. Na to to nakonec až tak nepoužívám. Očekával jsem i nějakou revizi kódu, dejme tomu, že bude schopen poznat, jestli něco dělám správně nebo špatně, nějaký základní chyby. Ale to vlastně děláme pomocí jiných nástrojů v GitHub Actions. V rámci toho buildu se nám tam kontroluje spoustu věcí a ty mi tohle spíš odchytávají než samotný Copilot." P7-FS*

> *„Nakonec z toho mého zkoušení toho Copilota vyplynulo, že nejlépe mi to pomáhá jako gumová kachnička." P7-FS*

Participant P9-FS viděl v používání asistentů kódování velkou úsporu času. Ta pro něj vzhledem ke způsobu jeho hodnocení znamenala rychlejší zpracování zadaného úkolu případně větší množství odvedené práce. To se následně odrazilo i na jeho celkovém ohodnocení.

> *„Určitě ušetření času. Já jsem v tu dobu pracoval jako OSVČ, tam se platí hodinově, a pokud se vlastně estimuje na nějakou funkcionalitu určitý čas, tak buď máš víc volna nebo můžeš stihnout více práce a dostaneš více zaplaceno." P9-FS*

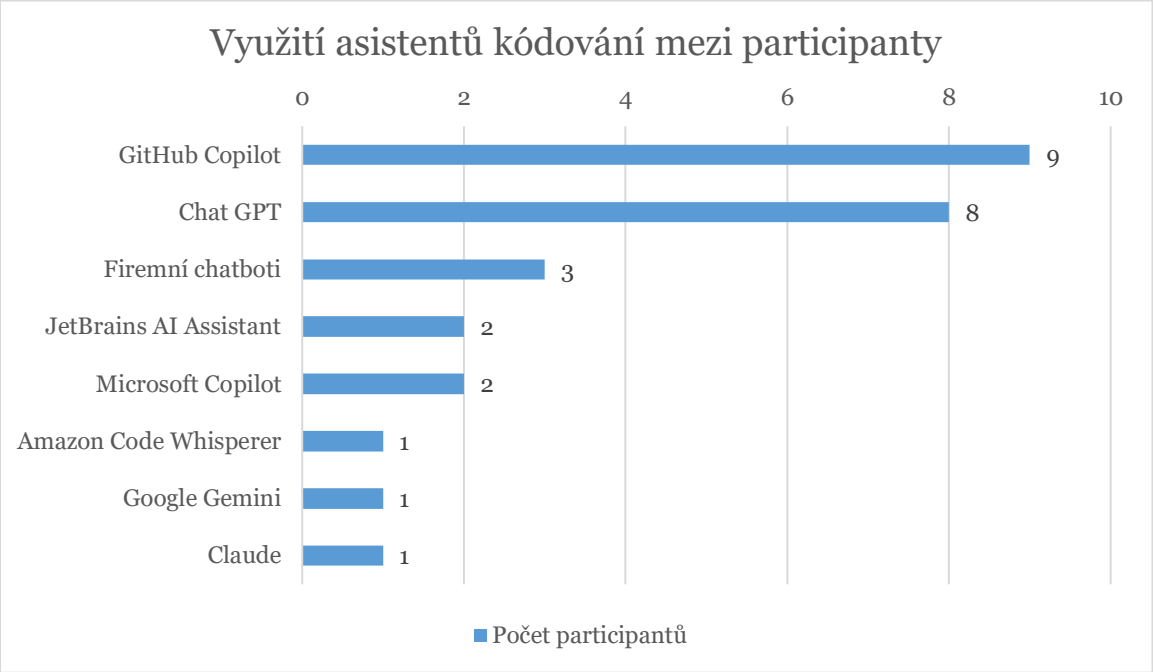
3.2 Současný postoj k asistentům kódování

Podkapitola se zaměřuje na aktuální vnímání asistentů kódování z pohledu vývojářů. Popisuje, jaké AI asistenty kódování využívají programátoři nejčastěji a jak často s nimi během své práce interagují. Dále se zaměřuje na přínosy, které programátoři při jejich používání očekávají. Věnuje se také popisu, jak jednotliví participanti asistenty kódování vnímají v porovnání se živými kolegy, jaká je expertíza asistentů kódování a jaké mají asistenti při používání největší omezení.

3.2.1 Využívaní asistenti

Z provedených rozhovorů vyplývá, že mezi nejčastěji používané AI asistenty kódování v rámci zkoumaného vzorku participantů patří GitHub Copilot, následovaný Chat GPT a firemními chatboty. Do kategorie „firemní chatboti" spadají všichni AI asistenti kódování, kteří fungují na podobném principu jako dva zmíněné nástroje, avšak společnosti, pro které participanti pracují, si je upravily podle vlastních potřeb. Modifikace byly nejčastěji prováděny s cílem zvýšit bezpečnost a ochranu zpracovávaných dat. Kromě zmíněných

asistentů využívají participanti i další nástroje. Graf 2 ukazuje všechny asistenty včetně počtu účastníků rozhovorů, kteří daného asistenta vyzkoušeli.



Graf 2: Využití asistentů kódování mezi participanty, zdroj: autor

3.2.2 Četnost interakce s asistenty kódování

Z provedených interview vyplývá, že s postupem času využití asistentů kódování roste stejně jako důvěra programátorů ve schopnosti asistentů. Při určení četnosti použití je nutné odlišit našeptávání kódu přímo v IDE a funkci chatu.

Funkci našeptávání kódu používají participanti během vývoje softwaru velmi často. Podle participantů P2-BE, P6-BE a P8-FE se stalo našeptávání kódu naprosto běžnou součástí použití asistentů kódování během jejich práce. Oproti jiným nástrojům poskytujícím automatické doplnění kódu vidí u asistentů výhodu v tom, že jsou schopni doplňovat větší a delší celky kódu.

> *„Našeptávání od Copilota mám zapnutý pořád, to už vůbec nevypínám. A využívám ho pravidelně bez ohledu na to, co dělám.“ P2-BE*

> *„Při psaní zdrojového kódu, to je jednoduché. Jakmile začneš psát, tak ti tam už rovnou začne dávat návrhy jako autocomplete. To je poměrně přesný a spolehlivý, dokončuje řádky i delší bloky kódu, takže to používám neustále.“ P6-BE*

> *„Při doplňování kódu v IDE už ani kolikrát nevím, že ho používám. To je reflex. Už to beru jako takovou prodlouženou ruku.“ P8-FE*

Míra používání funkce chatu se odvíjí primárně od řešeného problému a zkušeností programátora s využívanou technologií. Participanti P2-BE, P5-BE, P6-BE a P7-FS se shodují, že čím větší je jejich znalost řešené problematiky, tím méně asistenty kódování do řešení problému zapojují a méně využívají i možnost konzultovat řešený problém.

„U chatu určitě záleží, jestli je ten úkol z principu programátorsky jednodušší a nevyžaduje to napsat hodně kódu, nebo jak moc té řešené problematice rozumíš. Pokud to je něco jednoduchýho nebo něco, co znáš, pak to [nápovědu od asistenta] člověk skoro ani nevyužije.“ P2-BE

„Tam jde o to, že třeba v té Javě vím, jak se píšou všechny rutiny. Ale když je mám [rutiny] v tom Kotlinu, tak někdy nevím, jak to napsat, aby to bylo co nejefektivnější. Protože nemám tolik zkušeností s tím jazykem. Takže u mě se rozhodně míra použití obvíjí od zkušeností a znalostí v tom jazyku.“ P5-BE

„Co se týče nějakého troubleshootingu přes chat, tak hrozně záleží, co člověk dělá. Když děláš něco, co dobře znáš, pak nepotřebuješ konzultovat. Ale když si máš rozjet nějakou technologii, kterou vůbec neznáš, nikdy jsi to neviděl, tak ten chat pochopitelně používám dost často.“ P6-BE

„Za mě to používám na tom frontendu víc, protože v Reactu nedělám tak dlouho, takže nevím zas tolik věcí a asi častěji potřebuji pomoc, nějakou konzultaci nebo pomoc s vygenerováním nějaké komponenty nebo jiného kusu kódu.“ P7-FS

Podle participanta P2-BE závisí míra využití chatovací funkce i na způsobu, jakým se s chatem komunikuje a interaguje. Automatické našeptávání kódu je přímo zakomponováno do editoru, kde vývojář pracuje, a nabízí návrhy přímo v kontextu aktuálně upravovaného kódu. Oproti tomu funkce chatu, i když je integrována do vývojového prostředí, vyžaduje otevření samostatného okna a popsání problému, což vede ke ztrátě koncentrace.

„Já si myslím, že když mám použít to chatovací okno nebo se překliknout do prohlížeče, tam to beru víc selektivně, protože mi to už víc narušuje to workflow. Místo toho, abych nad problémem přemýšlel sám nebo jsem něco sám psal, tak se zastavím, musím si otevřít chatovací okno a začít psát ten dotaz. To je pro mě výrazné narušení soustředění,“ P2-BE

3.2.3 Očekávané přínosy

Na úvod je důležité zdůraznit, že očekávání vývojářů se v čase vyvíjejí. Jak již bylo dříve zmíněno, při prvním setkání s asistenty kódování participanti často neměli vysoká očekávání. Jejich přístup byl spíše experimentální, motivovaný zvědavostí a snahou porozumět fungování nového nástroje. Postupem času se však jejich očekávání výrazně zvýšila, což úzce souvisí s přínosy a výhodami, které v používání asistentů objevili.

V současné době je využívání asistentů kódování ze strany vývojářů spojeno primárně s komfortem, který popisuje participantka P1-FE. Ta očekává, že asistenti budou schopní zanalyzovat přiložené soubory a detailněji se zaměřit na popisovaný problém se snahou ho vyřešit, nebo i pomoci se získáním aktuálních informací z dokumentace.

„Já jsem si zvykla hlavně na komfort toho, že mám toho Copilota. A je to o tom, že očekávám, že nějakým způsobem zanalyzuje soubory, co mu tam přikládám, zaměří se na ten problém, co mu tam popisuju, že je schopnej tahat data z těch novějších dokumentací, takže mi ve výsledku usnadní tu práci.“ P1-FE

Účastníci rozhovorů neočekávají pouze pomoc s analýzou problémů a hledání podkladů k nim, ale lze je použít i při řešení problémů, se kterými se programátor nesetkává tak často. Například participant P4-BE od asistentů očekává, že mu pomohou při řešení méně častých problémů, nebo problémů, kde se používá kód, u něhož si musí osvěžit jeho správnou syntaxi. Stejně tak mu mohou pomoci najít nebo ukázat řešení, které by ho vůbec nenapadlo. Na druhou stranu při používání asistentů participant neočekává, že všechna řešení půjde použít bez jakékoli úpravy. Participanti P5-BE a P8-FE se přidávají s očekáváním, že jim asistent v případě potřeby kdykoli pomůže s rutinními úkoly, čímž jim ve výsledku ušetří čas.

> *„Teďkom od toho očekávám nějaké ulehčení práce, nabídnutí různých možností, které třeba nevidím nebo jsem je mohl třeba zapomenout. Ale obecně, když ho chci použít, tak očekávám, že mi poradí. Rozhodně ale neočekávám, když něco po něm chci, že bych to jenom vzal a zkopíroval.“ P4-BE*

> *„Už se spoléhám hodně často na to, že když dělám právě nějaké rutinní věci, on mi s tím pomůže.“ P5-BE*

> *„Jednoduše očekávám, že mi to ušetří práci a tím pádem čas. Běžně se mi teďka děje, že vytvořím novou komponentu a on ji skoro celou udělá za mě.“ P8-FE*

Očekávání se ovšem u vývojářů liší i podle toho, jakou funkci asistenta kódování používají. Podle participanta P6-BE při využívání chatu vývojáři očekávají primárně nápovědu a poskytnutí rady k předloženému problému. Nutně nečekají přímo vygenerovaný kód, ale spíše popis řešení v přirozeném jazyce. Při využití rozšíření pro generování kódu naopak očekávají, že od asistenta dostanou konkrétní kód zapadající do daného projektu a respektující využívané programovací styly a vzory.

> *„Ve výsledku je to při použití chatu vyřešení nějakého problému bez emocí a poskytnutí popisu, rady a schopnost posunout mě dál v tom problému. Klidně jen klasicky normálním textem.“ P6-BE*

> *„Pokud používám Copilota v IDE, tak tam zase od toho mám jiný očekávání. Tam čekám přímo, že mě vygeneruje kusy kódu, protože zná celý projekt, dokonce má přečtený i celý GitHub. Musím říct, že ta moje očekávání se naplňují.“ P6-BE*

Během výzkumu se objevil i názor od účastníka P7-FS, že asistenti kódování nenaplnili jeho původní očekávání týkající se množství generovaného kódu. Sám ovšem připouští, že to může být způsobené stylem práce a její náplní. Oproti ostatním účastníkům se jeho náplň práce výrazně liší. To zejména v tom ohledu, že kromě samotného psaní kódu se podobně jako P5-BE věnuje vedení lidí v týmu. Na samotné programování mu pak zbývá méně času.

> *„Zase tolik kódu mi ten Copilot nepíše, jako jsem čekal, že bude. Možná je to ale dáno tím, co dělám a jak často ho používám.“ P7-FS*

3.2.4 Asistenti vs. živí kolegové

Porovnání řešení problému s AI asistentem a živým kolegou bylo zmiňováno poměrně často. Vždy se ovšem jednalo spíše o obecné konstatování toho, že asistent je dobrý na zpracování

repetitivních úkolů, které se nikomu nechtějí dělat, nebo na rychlou konzultaci nápadu v prvotní fázi. Nejdetailněji však popsal rozdíl AI asistenta a živého kolegy účastník P7-FS.

Podle jeho názoru má živý kolega zásadní výhodu v tom, že komunikace s ním není omezena pouze na úzký výsek kódu, ale bývá komplexnější a názorově rozmanitější. Klasický kolega se navíc často vyhne zacyklení na stejných odpovědích a dokáže v případě potřeby nahlížet na problém v širším týmovém kontextu. Jedním z klíčových aspektů, které asistentům dosud chybí, je znalost historických i aktuálně řešených problémů, používaných technologií a zavedených postupů v týmu. Právě díky této znalosti může živý kolega nabídnout zcela odlišné a často správnější řešení, které by asistent nebyl schopen navrhnout. Důvodem je absence potřebného kontextu, který by samotného vývojáře mnohdy ani nenapadlo explicitně zmínit.

> *„…když se ptám na něco chatbota, tak se ptám na konkrétní dotaz. Když se ptám kolegy, tak se sice ptám na konkrétní dotaz, ale on rovnou na to může navázat na něco z úplně jiné oblasti a zároveň i to mě může posunout dál. Teď to [AI asistenti] neumí… Navíc kolega zvládá naprosto bravurně diskutovat i nad delším a rozsáhlejším kódem.“ P7-FS*

> *„Prostě [kolega] není zaboxovaný [zahleděný] jenom do toho jednoho dotazu, který jsem napsal, ale má daleko větší přehled o tom, co řešíme, v jakých technologiích to řešíme a tak.“ P7-FS*

Pokud by si mohl vybrat, při návrhu nových částí aplikace by raději preferoval komunikaci se živým kolegou. Naopak při práci na dálku preferuje komunikaci s asistenty kódování, jelikož se občas cítí, že by svého živého kolegu online hovorem nebo zprávou mohl vyrušovat. Zmiňuje také, že tato varianta je pro něj při práci na dálku příjemnější a rychlejší.

> *„A co se týče toho návrhu, tak myslím si, že kdybych seděl nebo mohl mít pět dní v týdnu v kanclu živé kolegy, tak ho [asistenta] tolik nepoužívám, že bych si raději povídal s těmi živými lidmi…“ P7-FS*

> *„…ale když sedím doma, tak prostě někomu napsat nebo zavolat, mám tam takový ten blok, že prostě nevím, jestli teďka má ten člověk čas, jestli mi to vezme a jak to správně formulovat, tak tam radši napíšu tomu Copilotovi a je to pro mě příjemnější a často i rychlejší.“ P7-FS*

Na druhou stranu participant P6-BE poukazuje na výhodu komunikace se strojem místo s živým kolegou v tom, že není třeba s AI asistentem vést zdvořilostní konverzaci před zahájením řešení problému a je možné se ponořit hned do jeho řešení.

> *„Je to stroj, můžeš vynechat jakýkoliv emoce, zdvořilostní fráze nebo nějaký prosím a děkuji, takže prostě soustřeďujeme se na ten problém. Tady je zadání, jak to vypadá, to potřebuju, chci takový a takový výstup.“ P6-BE*

3.2.5 Expertíza asistentů

Úhly pohledu na senioritu asistentů se liší podle řešených problémů. Z pohledu znalosti jednotlivých algoritmů, syntaxe, výjimek, dokumentace a hloubky expertízy v jasně definované problematice participanti hodnotí asistenty velmi kladně. Často padala

hodnocení na úrovni seniorních vývojářů. Participant P3-FS navíc dodává, že až k dosavadním znalostem a schopnostem asistentů přibude schopnost číst analytická zadání, bude se jednat o skutečného seniora.

> *„…v momentě, kdy to jsou takový běžný věci, co by většina programátorů, udělala stejně, tak tam si myslím, že už je to schopné generovat správnou odpověď a člověk si může být jistý, že můj problém vyřeší tak, jak by to vyřešili ostatní.“ P2-BE*

> *„Jde o to, že tyto nástroje mají načtenou dokumentaci. Jestli to budeme brát podle znalosti syntaxe, zákoutí kódu, výjimek ve frameworcích, chybových hlášek a jak je vyřešit, tak jako absolutní senior. A ve chvíli, kdy si dokáže třeba ten Chat GPT přečíst zadání od analytika, tak to bude senior se vším všudy.“ P3-FS*

> *„Jestli se díváš na tu hloubku expertízy, to by mohlo být klidně něco nad seniorem, protože je schopen odpovídat na všechny ty otázky.“ P5-BE*

> *„Nějaký algoritmus, tak na to je častokrát seniornější než já, nebo vytvoření React komponenty. Nebo i prostě ví trochu víc z těch návrhů, z architektury než já.“ P7-FS*

Vždy ovšem záleží na konkrétním typu a komplexitě úkolu. Pokud dostanou asistenti za úkol řešit komplexnější problém, nebo se jedná o nestandardní situaci, která není řešitelná běžnými postupy, nebo již existujícími algoritmy, pak se asistenti často ztrácejí a dostávají se v hodnocení na úroveň juniorů nebo mediorů.

> *„Když mu dáš řešit komplexní úkol, tak se dostane úplně na úroveň juniora. Když mu dáš něco, co napíšu kódem na 10, 20, 30 řádků, tak to bude určitě hodně seniorní, ale když mu dám udělat projekt o 100 souborech, jako 100 třídách, nedovedu si představit, že by to zvládnul.“ P5-BE*

> *„Takže, jak říkám, na těch častejch věcech je to senior a na těch méně častejch je to ulhanej junior.“ P6-BE*

> *„… ale aby to bylo to, co dodá seniorní člověk, teda celý funkční obraz, tak na tom zase rozhodně není. Tam si myslím, že je spíš pod juniorem.“ P7-FS*

> *„Já bych řekl, že je to takový zkušenější medior [kariérní stupeň mezi juniorem a seniorem]. Záleží podle typu úkolu. Ty jednodušší úkoly to zvládá v pohodě, ale jak je něco hodně komplexního, tak se zadrhává, nedokáže pochopit ten kontext a je to s ním fakt na dlouho.“ P9-FS*

Nicméně i u objektivně jednoduchých a nekomplexních úkolů s dostatečným množstvím informací se někdy asistentovi nepodaří vrátit funkční řešení. Ovšem, jak zmiňuje participant P2-BE, nástroje se postupně vyvíjejí a zlepšuje se i úroveň generované odpovědi. Oproti prvním verzím se podle něj kvalita odpovědí výrazně zlepšila. S tímto názorem souhlasí i participanti P4-BE a P6-BE, kteří ovšem dodávají, že kvalita odpovědi bývá kolísavá. Participant P8-FE si ovšem i přes neustálé zlepšování a vývoj jednotlivých používaných modelů myslí, že zejména architektonické znalosti a míra autonomie asistentů se stále nedají plnohodnotně porovnávat s živými programátory. Přednosti asistentů v tuto chvíli vidí hlavně ve schopnostech agregace zdrojů a rychlosti práce.

„Myslím si, že ze začátku, kdyby to byl člověk, asi bych ho ani nezaměstnal. Dneska vzhledem k tomu, jak se nástroje zlepšily, si myslím, že to je vlastně byznysově relevantní věc.“ P2-BE

„Jestli bych to měl hodnotit jako ve škole, tak občas je to jednička, občas je to prostě klidně pětka. Prostě lítá to opravdu z extrému do extrému.“ P4-BE

„Kvalita je kolísavá, ale většinu času bych řekl, že je dost seniorní. Samozřejmě občas to na jednoduchých kusech kódu selže naprosto katastrofálně, protože ti začne doporučovat dokončení řádku, jako úplně nějaký nesmysl, neexistující atributy a podobně.“ P6-BE

„Já bych řekl, že schopnosti zatím nejsou na takové úrovni autonomie, abychom to byli schopni zhodnotit s reálnými lidmi. Já si myslím, že v tuhle chvíli to AI funguje víc jako agregátor informací, protože ve finále, když mu dáš řešit nějaký problém, tak on je schopný pracovat rychleji, ale nemyslím si, že ještě líp než lidi. Zejména tedy co se architektury týče, tam je slabý. Matematika a algoritmy to je stranou, k tomu se nedostanu tak často, ale databáze a architektura tam pořád juniora AI nenahradí.“ P8-FE

3.2.6 Omezení asistentů

Podkapitola se věnuje základním omezením aktuálních verzí asistentů kódování, které jednotliví participanti při používání zaznamenali. Mezi největší omezení patří nemožnost integrace asistentů kódování s ostatními nástroji, jejich schopnost zpracovávat komplexnější úkoly, obecnost jejich odpovědí a originalita generovaných řešení.

Integrace s ostatními nástroji

Vývojáři P1-FE a P3-FS vnímají nemožnost asistentů kódování získávat informace z jiných nástrojů a aplikací jako jedno z velkých omezení. Jedná se o informace ze znalostních bází, ticketovacích nástrojů, ale i dalších aplikací obsahujících relevantní informace pro řešení konkrétního problému. Omezený přístup k těmto informacím může podle účastníků vést ke snížené schopnosti asistenta poskytnout relevantní odpověď.

„Nejhorší je, že člověk mu nemůže najednou dát celou tu Jiru nebo Confluence, celý ten background storky [požadavku ve formě user story], aby pochopil zadání, což pak může vést určitě k nějakým technickým dluhům... Nemusí ta architektura té aplikace být elegantní, nemusí být zas tak přepoužitelná, může se s ní hůř pracovat.“ P1-FE

„Máme tady GitHub Copilota, který je na GitHubu, kde máme repozitář a on tam zná úplně všechno. Pak tady ale máme hromadu dalších informací, které nemáme v žádném repozitáři. Jen namátkou: jak spolu integrují aplikace, jak jsou nastavené servery, jak probíhá build, jaké je flow nasazení. Všechny tyhle informace jdou úplně mimo jakýkoli repozitář a často je nejde ani do nějakého zanést, je to v nějakém settingu aplikace. A on do ní nemá přístup, nemůže to vědět.“ P3-FS

Participant P8-FE dále doplňuje, že v dnešní době má hodně aplikací své vlastní asistenty, ale vzhledem k absenci jakékoli vzájemné integrace přicházejí nástroje o řadu užitečných funkcí s velkým potenciálem a přidanou hodnotou.

„AI už je v GitHubu, ve Figmě, v Notionu, v Jiře atd. Já ale vidím jiný nedostatek. Všechny tyhle tooly pracují jenom sami se sebou. Ten moment, kdy se stanou mnohem silnější je, když si začnou povídat mezi sebou. Až se tohle někomu povede, tak to bude další velký krok dopředu.“ P8-FE

Komplexita úkolu

Účastníci se shodují, že velkým omezením současných asistentů kódování je jejich schopnost zpracovávat komplexnější úkoly. Například participant P4-BE zmiňuje, že i když se opakovaně snažil zadávat asistentům komplexní úkoly s podrobným popisem, i přes to se stávalo, že asistenti na některou část zadání zapomněli. Stejně vnímá komplexitu úkolu i participant P5-BE. Podle něho je lepší rozdělit řešené úkoly na menší části a ty s asistenty řešit. Kvalita výstupů u dílčích úkolů totiž bývá velmi dobrá, ale ve zpracování komplexnějších úkolů nejsou asistenti rozhodně silní. Účastník P7-FS dodává, že i když asistent zvládne vygenerovat odpověď na komplexnější problém, úprava jeho odpovědi do funkční podoby může následně zabrat podobné množství času, jako kdyby úkol udělal vývojář celý od začátku sám. Tuto zkušenost potvrzuje i participant P9-FS, který se asistenty snažil i upozorňovat na konkrétní místa, kde se chyby ve vygenerované odpovědi nacházejí. I když asistent jednu konkrétní chybu opraví, nezřídka vytvoří chybu jinou.

„Zkoušel jsem je používat i pro větší celek, ale tam mu musíš slovně popsat, co vůbec chceš. Jakmile mu zadáš trochu delší prompt, často na něco zapomene. Snad nikdy mi u rozsáhlejšího úkolu nedal správnou odpověď. Vždycky jsem tam musel alespoň nějakou drobnost opravit. V takovém případě je ale potom lepší jít postupně. Necháš si vytvořit nějakou strukturu, potom mu začneš doplňovat postupně větší a větší detaily. Přitom ho kontroluješ a koriguješ ten jeho výstup. Úplně si nedovedu představit, že by to dělalo celý kód za tebe.“ P4-BE

„…spíš asi ta komplexita, kvalita mi přijde dobrá. On jako nedělá chyby. Musíš jít ale postupně a navádět ho. Jakmile dostane něco fakt většího a komplexnějšího, dřív nebo později se ztratí. Protože si vymyslí nějakou blbost. A pak bys ho musel přesně půl den ukecávat, aby ses k dopracoval k rozumnému výsledku.“ P5-BE

„Nemyslím si, že zatím dokáže navrhnout a dodělat komplexní aplikaci. Kdybych řekl, teď potřebuji udělat tuhle a tuhle aplikaci s takovými parametry, nemyslím, že by to Copilot zvládnul udělat dostatečně kvalitně, aby s tím nebylo potom skoro stejně práce jako bez něj. Nebo samozřejmě některé ty dílčí rutinní věci tam dokáže zrychlit, ale že by udělal komplexnější práci za vývojáře, tak na té úrovni ještě rozhodně nejsme.“ P7-FS

„…já jsem už zkoušel to, že mu u komplexnějšího úkolu napíšu, kde je chyba, aby ji mohl opravit. On jednu chybu opraví, vygeneruje odpověď znovu, ale většinou se tam objeví zase nějaká chyba na jiném místě. No a takhle se to pořád opakuje.“ P9-FS

Participant P9-FS vnímá kromě vyšší chybovosti asistentů u komplexnějších úkolů jako jejich další omezení také nižší schopnost uchovávání kontextu v delších chatovacích vláknech. Asistenti sice pracují pořád ve stejném vlákně, ale po několika předchozích promptech zapomínají, jaké požadavky byly na nástroj kladeny dříve.

„Já mám spíš problém, že zapomínají kontext. Mně se něco podobného stávalo ještě bez asistentů, když jsem dělal různé analýzy, modely a korelace. Každopádně vnímám to v určitých momentech i u těch asistentů. Já v rámci jednoho vlákna můžu víceméně furt přidávat nové a nové prompty, ale v určité době on začne zapomínat, co byl ten prompt na začátku. On není schopný si udržet celý kontext toho vlákna. Sice se ti omluví, ale je ti to na houby, protože pak zapomene něco jiného.“ P9-FS

Obecnost odpovědí a originalita řešení

Obecnost odpovědí navrhovaných asistenty kódování je další omezení, které vnímá participant P6-BE. Do obecné roviny odpovědí se nástroje dostávají, pokud si samy s řešeným problémem nedokáží poradit. V takovém případě bývají jednotlivé kroky navrhovaného řešení často nejen obecné, ale i zdlouhavé. Navíc jejich vykonání přesně podle zadání často nezaručuje úspěch a vyřešení problému. Participant by v takovém případě upřednostňoval, aby mu asistent dopředu řekl, že si neví rady. Takové chování by pak ušetřilo mnoho času.

„Když ho požádáš, aby ti s něčím pomohl a on ti navrhne body zaprvé, zadruhé, zatřetí, tak dost často jsou to takový univerzální rady, jako bys použil třeba Microsoft Fórum. Jsou to takové úmorné kroky, kterým se tě snaží spíš odradit, než aby vyřešili tvůj problém. Tak až tam směřuje ten Chat GPT s tím troubleshootingem, když už si teda neví rady. To je trochu odpuzující. Kolikrát to vyzkoušíš, nic ti to nepomůže, ale ztratíš čas. Bylo by prostě lepší, kdyby rovnou řekl, že neví.“ P6-BE

Participant P7-FS zmiňuje další zajímavý poznatek týkající se způsobu, jakým asistent poskytuje odpovědi. Podle jeho zkušenosti asistent nabízí buď relativně obecné rady popisované participantem P6-BE v předchozím odstavci, nebo naopak poskytne velmi podrobné odpovědi. U řešení konkrétních problémů se však často stává, že se asistent pevně drží zadaného kontextu a jen těžko ho dokáže opustit. Participant toto přisuzuje způsobu, jakým byli asistenti trénováni. Ti se podle něho učí na určitém množství dat, kde byl problém řešen určitým způsobem. Chybí jim ovšem schopnost v případě potřeby kontext rychle opustit a navrhnout kreativní řešení, jako umějí lidé.

„Podle mě je schopen navrhnout moc obecné řešení, anebo když už se dojede do něčeho konkrétnějšího, tak je zase zaškatulkovaný do nějakého kontextu. Nicméně, když je třeba, tak člověk z detailu toho problému vyskočí snáz ven. Ale ti asistenti jsou prostě často omezení na nějaké své zadání, které se naučili na již existujících věcech a vymyslet něco nového, co bude jiné, je pro ně opravdu obrovský problém. Možná bych řekl, že to ještě ani neumí.“ P7-FS

3.3 Práce s asistenty kódování

Podkapitola popisuje, jakým způsobem vývojáři používají asistenty kódování během své práce. Zaměřuje se na to, jaké jsou faktory ovlivňující kvalitu vygenerované odpovědi a jaké způsoby používání účastníkům rozhovorů fungují nejlépe. Dále se zaměřuje na případy, kdy je lepší se při řešení úkolu na asistenty kódování nespoléhat. Zabývá se také způsoby, jak participanti dále přistupují k získané odpovědi a jak s ní pracují.

3.3.1 Faktory ovlivňující kvalitu odpovědi

Mezi klíčové faktory ovlivňující kvalitu odpovědi patří podle účastníků rozhovorů zpracovávaný úkol, způsob zadávání promptu, výběr jazykového modelu, dodaný kontext řešené problematiky, ale i chování vývojáře spolu s projektem a využívanými technologiemi. Faktory částečně souvisejí s omezeními asistentů popsanými v kapitole 3.2.6.

Zpracovávaný úkol

Největší vliv na kvalitu vygenerované odpovědi má podle participantů velikost řešeného úkolu. Zdůrazňují, že rozhodně nejsou asistenti kódování ještě na takové úrovni, aby byli schopni zpracovat úkol od začátku do konce pouze podle zadání požadavku. S rostoucí komplexitou řešeného úkolu podle participanta P5-BE úměrně klesá i přesnost odpovědi a ochota programátora využívat asistenty kódování. Participant P6-BE uvádí, že by nemělo zadání úkolu, které má asistent kódování zpracovat, přesáhnout přibližně 10 vět. Po překročení této hranice jsou získané výsledky velmi nepřesné. Participant P8-FE pak potvrzuje, že jeho zkušenosti jsou podobné a klíčem k efektivnímu použití asistentů je rozdělení komplexního úkolu na menší celky, se kterými dokáže následně asistent výrazně lépe pomoci.

> *„Čím je to [úkol] komplexnější, tak tam přestává fungovat. Teda alespoň já ho přestávám využívat, protože mi přijde, že bych ho musel moc opravovat a ten strávený čas by byl neúměrný výsledku.“ P5-BE*

> *„Určitě to funguje tam, kde ten problém jde dobře popsat pár větama. Řekněme fakt maximálně 10 vět. Pokud mu chceš zadat větší úkol, pak s ním bude mít problém. Hlavně pokud je to problém rozlezlý přes několik tříd.“ P6-BE*

> *„Myslím si, že AI v tuhle chvíli není schopný udělat celý úkol od začátku do konce. Ale když mu úkoly rozsekáš na menší části, tak je schopný s nima pomoct. Vyřeší za tebe rychleji třeba research, anebo tě navede na správný způsob řešení.“ P8-FE*

Kromě velikosti úkolu ovlivňuje výrazněji kvalitu odpovědi také jasnost problému, jeho standardizace a dostupnost zdrojů. Participant P4-BE považuje za nejvhodnější úkoly pro asistenty takové, které mají jasně definovaná pravidla nebo využívají zavedené algoritmy. V těchto případech asistenti dokáží nabídnout přesná a efektivní řešení, aniž by bylo nutné je výrazně upravovat. Podle participanta P6-BE platí pravidlo: čím více je dostupných zdrojů o řešeném problému, tím vyšší je pravděpodobnost získání kvalitní odpovědi.

> *„…když by se na ten kód podívalo deset programátorů a devět z nich ten úkol udělá stejně, tak pro takové úkoly je ten asistent jako stvořenej a pomáhá bravurně.“ P2-BE*

> *„Je silnější tam, kde se má čeho chytnout. Ideálně pokud se ten úkol má řešit podle přesně daných algoritmů, jsou tam přesně zavedená pravidla. Rozhodně ale nejde použít na nějaké abstraktnější storky, kde se nemá čeho chytnout a spíš by si vymýšlel.“ P4-BE*

> *„Čím více dostupných zdrojů o té problematice na internetu najdeš ty sám, tím zpravidla je i kvalita odpovědi asistenta lepší, protože má víc podkladů, z čeho čerpat.*

A samozřejmě to platí i obráceně. Pokud je málo zdrojů, pak si velmi pravděpodobně bude vymýšlet.“ P6-BE

Na druhou stranu pořád existují typy úkolů, jež nehledě na dostupnost zdrojů a existenci standardů nejsou asistenti kódování podle participanta P3-FS schopni vyřešit. Podle jeho názoru se jedná zejména o abstraktnější úkoly vyžadující schopnost obsáhnout a pochopit velké množství kontextu. Takové úkoly se typicky řeší např. na úrovni architektury softwaru.

„Samozřejmě jsou otázky, třeba na architekturu, to už je tak abstraktní, tak široce v kontextu, že ty nástroje ani náhodou nemůžou obsáhnout tuhle problematiku. Můžou dát strašně obecné odpovědi, které mi jsou ale k ničemu. Teďka mám na mysli třeba koncepční pohled, jak něco udělat. Nemyslím konkrétní implementaci, ale když se baví třeba vývojář a analytik, jak něco udělat, jak postavit architekturu aplikace.“ P3-FS

Participant P9-FS zdůrazňuje, že při zadávání úkolu asistentovi je důležité nejen specifikovat požadavky a omezení vyplývající přímo z potřeb zákazníka, ale také zohlednit technická omezení a zavedené zvyklosti projektu. Tato pravidla se obvykle do zadání nepíší, protože jsou pro vývojáře samozřejmostí. Pro asistenta jsou ovšem neznámá a jejich uvedení do promptu výrazně zvyšuje kvalitu a relevanci výstupu. Pokud tedy chtějí vývojáři zadání požadavku použít pro tvorbu promptu, pak jej zpravidla musí o tyto doplňující informace obohatit.

„Hodně záleží na kvalitě zadání a popisu těch omezení, pravidel a limitů. Musíš popsat opravdu hodně věcí, i třeba ty pro tebe nebo tvůj tým vývojářů jasný a očividný. Zpravidla nejde použít zadání požadavku, co dostaneš, protože tam není určitě zohledněno všechno, na co třeba my jako developeři myslíme, aniž by to musel analytik napsat, protože je to v týmu nějaký zajetý zvyk.“ P9-FS

Na druhou stranu participant P7-FS upozorňuje, že velké množství definovaných pravidel a omezení kvalitě vygenerované odpovědi také neprospívá. To se týká zejména již existujícího kódu, do kterého je třeba dodatečně přidat požadavek, a asistent se při jeho generování musí pravidlům přizpůsobovat.

„Když má méně těch omezení, kterým se musí přizpůsobovat, nebo množství kódu, kterému se musí přizpůsobovat, tak je to pro něj lepší, že nemusí přemýšlet nad tím kontextem, co už tam vlastně je vytvořené, a jak to do toho napasovat.“ P7-FS

Způsob zadávání promptu

Při zadávání prvního promptu v rámci nové konverzace účastníci P6-BE, P7-FS a P8-FE vyzdvihují množství poskytnutého kontextu asistentovi jako klíč k získání co nejpřesnější odpovědi. Většinou zadávají do prvního promptu kontext projektu, využívané technologie, chybové vstupy, nebo i cíl, kterého chtějí dosáhnout. Podle názoru participanta P8-FE je asistent s větším množstvím kontextu schopen lepšího zdůvodňování, proč zadaný problém řešil vybraným způsobem. Důležitost kontextu pak roste úměrně s logickou náročností řešeného problému.

„Například specifikuju jako první, že mám tady Spring Boot projekt plus Docker Kubernetes a nedaří se toto, chci dosáhnout tohoto a už mu rovnou sypu klidně i stack trace nebo nějaké chybové výstupy. S tím už si prostě ten bot poradí.“ P6-BE

„Úplně na začátku sepíšu třeba takový odstaveček cca na tři věty, co si představuji, že od něj chci dostat.“ P7-FS

„Já většinou preferuju spíš komplexnější zadání toho prvního pokusu. Používám to [asistenty], když mám kód, vím o tom problému hodně, tak mu radši dám dlouhý odstavec toho, kdy popíšu přesně co a jak, protože mám pocit, že když mu dám hodně kontextu na začátku, tak bude mít jednodušší reasoning a díky tomu se ztratí nebo musí něco měnit rozhodně méně často. A jakmile se jedná o nějaký logický problém, tak množství kontextu ovlivňuje kvalitu odpovědi ještě mnohem víc.“ P8-FE

Participant P9-FS ještě přímo zmiňuje, že před jakoukoli interakcí s asistentem, při níž požaduje určitou radu, si zpracovávaný požadavek vždy upravuje a strukturuje do podoby, kterou jsou asistenti schopni lépe zpracovat. Každý dotaz by podle něho měl vždy obsahovat minimálně kontext projektu, cíl úkolu, jeho omezení a speciální podmínky a požadavky. Používáním této struktury se participantovi daří dlouhodobě získávat lepší výsledky než před jejím používáním.

„…protože ty požadavky zpravidla dělá více lidí, tak mají pokaždé trochu jinou strukturu, a ne vždy je tam vše. Nicméně mě se osvědčila jedna struktura toho dotazu, kde dám na začátek kontext celého projektu nebo části projektu, potom zadávám cíl úkolu, což beru z toho požadavku, a nakonec omezení nebo speciální podmínky, které mu [asistentovi] nemusí být na první pohled zřejmé.“ P9-FS

Participanti P3-FS a P5-BE zjistili, že napsání prvního řádku požadovaného kódu přímo vývojářem tak, jak si ho představuje, výrazně zvyšuje kvalitu vygenerovaného výstupu při našeptávání kódu v IDE. Popisovaný přístup umožňuje asistentovi lépe porozumět záměru vývojáře a generovat relevantnější kód. Tak se minimalizuje potřeba dodatečného upřesňování požadavku a zároveň se snižuje riziko, že asistent vygeneruje kód, který nesplňuje očekávání vývojáře. Navíc se jedná o časově efektivnější způsob, než dodatečně slovně doplňovat asistentovi kontext.

„Mně opravdu víc vyhovuje, když mu vlastně napovím tím, že začnu psát. Ti asistenti v IDE se pak mnohem lépe chytají, když jim hodíš na začátku nějakou pomoc. Prostě si pak tolik nevymýšlejí a reagují přímo v tom kontextu, co chceš získat odpověď.“ P3-FS

„Vidí, že převádím telefonní číslo z jednoho formátu na druhý. Má navnímané ty třídy, ale ty začneš najednou převádět něco v jiné třídě, kterou on nemá navnímanou. Předtím si dělal e-mail a dal si getValue, getType. A předpokládá to samé u telefonu. Ale telefon může vypadat jinak, nicméně on si to začne vymýšlet podle toho e-mailu, který si dělal předtím. Což ve výsledku vede k vygenerování úplný blbosti. Abych s tím neztrácel čas, protože vím, že napsat něco takového mi trvá pět minut, ale abych ho ukecával a on mi to udělal za deset minut, tak ho nepoužiju hned od začátku, a napíšu první řádek té rutiny tak, jak má být a on už mi pak začne pomáhat mnohem přesněji.“ P5-BE

Výběr jazykového modelu pro generování odpovědi

Podle participanta P8-FE je pro získání co nejrelevantnější odpovědi klíčový správný výběr modelu pro generování odpovědi. V případech, kdy je vyžadována během zpracování úkolu

hlubší analýza kódu, nebo se jedná o úkoly, kde musí asistent provádět logické úvahy, preferuje participant při využití Chat GPT alespoň modely GPT O1 nebo GPT O1 mini. Jedná se o tzv. reasoning modely, které dokáží oproti běžným modelům jako je např. GPT 4O řešit komplexnější úkoly a porovnávat předpokládané chování se skutečností a vyvodit z toho závěr. Na druhou stranu vygenerování odpovědi pomocí těchto modelů trvá delší dobu. Z toho důvodu není jejich použití vhodné v případech, kde je preferovaná spíše rychlost oproti hloubce přemýšlení.

> *„Jakmile mu dám kód a chci nějakou analýzu, tak vybírám O1 a chytřejší, protože 4O vlastně v porovnání těch modelů neumí logicky uvažovat. V tu chvíli 4O shoří a vyhalucinuje nějaký nerelevantní nesmysl. Proto potřebuji ty reasoning modely. Naopak, když mi stačí jen něco vyhledat, tak ale klidně 4O použiju. Tam vidím naopak jeho výhodu v tom, že on sice tolik nepřemýšlí, ale rychle a dobře hledá informace, což O1 a výš někdy přemýšlí třeba i půl minuty.“ P8-FE*

Pro dokreslení situace připojuje participant i vlastní zkušenost, kde ukazuje, že na správném výběru jazykového modelu skutečně záleží.

> *„Kolega pracoval na projektu, který je hodně starý – Java 1.8, Hibernate, starý Spring, něco, co už hodně pamatuje. Dal to do modelu GPT 4o a odpověď byla úplně vyhalucinovaná, úplné nesmysly. Tak jsme vzali stejnou otázku, dali jsme ji ke mně do GPT O1, dostal jsem úplně jinou odpověď a s minimálníma úpravama to šlo použít.“ P8-FE“*

Tento názor potvrzuje i participantka P1-FE, která si myslí, že s postupem času a s každým novým jazykovým modelem se kvalita a přesnost odpovědi zvyšuje.

> *„Už teďka pociťuji, že ty novější modely umějí odpovědi generovat líp a víc nad tím preprocesují, přemýšlejí. Takže to pak hází lepší výsledky.“ P1-FE*

Kontext

Poskytnutí kontextu problému a jeho rozsah je podle účastníků P4-BE a P6-BE klíčové pro vygenerování relevantní odpovědi. Zároveň je však nutné zohlednit, že nadměrné množství informací může asistenta přetížit, ale nepřispívá ke zlepšení kvality jeho odpovědi.

> *„Záleží strašně moc na tom, jaký má kontext. To znamená, na pozadí je stejně nějaký LLM a ty musíš tomu asistentovi říct to, co by měl znát ohledně toho, co ty potřebuješ, a zároveň záleží i na velikosti toho kontextového okna. To znamená, že když ty chceš po něm strašně moc věcí, tak on začne ztrácet ten kontext, co má na začátku.“ P4-BE*

> *„Je to obrovský autocomplete [automatické doplňování kódu] a potřebuje dostat kontext, aby věděl, ve kterých neuronech má surfovat. Takže samozřejmě klíč k dosažení co nejlepší odpovědi je kontext.“ P6-BE*

Participant P8-FE pak vyzdvihuje, že většina asistentů je schopna zohledňovat styl psaní programátora a generované odpovědi jeho stylu přizpůsobit.

> *„…a hlavně jak to má i tu historii toho, co píšu, tak je to jenom lepší a lepší.“ P8-FE*

Vývojář

Výsledek a úspěšnost generování odpovědi asistentem kódování je dán i zkušeností programátora. Participantka P1-FE je přesvědčena, že programátor lépe se orientující v dané problematice je schopen přesněji definovat svůj požadavek a případné odchylky a chyby v odpovědi včas odhalit a instruovat asistenta, aby je opravil.

> *„Kromě toho samotného úkolu rozhodně záleží na zkušenosti člověka. Jednak junior ti spíš vezme ten vygenerovaný kód a nebude se toho asistenta tolik ptát, kdežto někdo zkušenější ho bude s tím výsledkem více konfrontovat a ptát se více do hloubky, čímž třeba odhalí nějakou nepřesnost, na kterou nemusí ten junior přijít.“ P1-FE*

Participanti P4-BE a P9-FS poukazují na význam pečlivé formulace zadání a popisu řešené situace při práci s asistentem kódování. Podle participanta P9-FS je potřebné předem si rozmyslet prompt včetně jasného cíle a kontextu. S tím souhlasí participant P4-BE, který ovšem ze své vlastní zkušenosti přiznává, že on a vývojáři v jeho okolí mají přirozeně tendenci si zadávané množství informací zkracovat, i když vědí, že toto chování může mít negativní dopady na kvalitu odpovědi.

> *„Když zadám prompt asistentovi, tak už dopředu musím přesně vědět, co od toho očekávám, musím mu popsat co nejlépe tu situaci. Takže není to takové slepé střílení různých nápadů, ale je to opravdu, že zadám prompt, který mám promyšlený, nejlépe s nějakým kontextem, s nějakým cílem, příklady atd.“ P9-FS*

> *„Hodně jde o to, jestli se ti to chce vypisovat nebo nechce. Aby si ho uvedl do problému, co ty vlastně chceš. Já osobně přiznávám, že ten popis požadavku občas zkracuju a doufám, že to bude stačit. Takže tahle moje lenost, že se mi nechce psát jako 20 řádků, ale jenom třeba 3 může ovlivnit tu odpověď. Na druhou stranu jsem viděl hodně lidí, co to dělají podobně.“ P4-BE*

V neposlední řadě může podle participanta P5-BE programátor ovlivnit kvalitu odpovědi asistenta kódování tím, že mu vytvoří a poskytne kvalitní předlohu kódu, podle níž může asistent zpracovávat další části kódu.

> *„Čím líp napíšu první metodu, kterou on se bude inspirovat, tím líp on začne s psaním těch zbylých metod.“ P5-BE*

Projekt a technologie

Participanti P2-BE a P3-FS se shodují, že stále vidí u asistentů kódování rozdíly ve schopnostech generování a zpracovávání úkolů podle jejich typu a využitého programovacího jazyka. P7-FS pak doplňuje, že z jeho zkušeností jsou odpovědi asistentů kódování přesnější při využití na frontendu. Zejména pokud se jedná o starší aplikace s velkým množstvím vlastní firemní logiky. V takových případech lze asistenty použít spíše pro získání obecnějších rad.

> *„Mně pořád přijde, že třeba u některých jazyků to umí radit hůř.“ P2-BE*

> *„Musím říct, že zatím furt vidím rozdíl mezi kvalitou frontendu a backendu, kdy se mi zdá, že na některé témata, na některé jazyky, to funguje lépe, na některé hůře.“ P3-FS*

„Já bych řekl, že na frontendu má daleko větší úspěšnost. Protože na backendu, zvlášť když je to v nějaké staré aplikaci, kde je Java 8, je tam spoustu interních věcí, o kterých on nemůže vědět, tak tam ho fakt používám spíš na hodně obecné rady, aby mě navedl nějakým směrem, ale fakt ho nejde použít, že by ti vygeneroval přímo celý kód.“ P7-FS

Kvalitu vygenerované odpovědi ovlivňuje i využití specifického firemního softwaru a knihoven, jejichž fungování není veřejně nikde popsáno. Využití asistentů kódování pro řešení problémů v kombinaci s interními knihovnami popisuje participant P6-BE jako nefunkční právě kvůli chybějícím informacím, případně jejich nedostupnosti pro asistenty.

„My v korporátu to máme ztížený tím, že máme tuny vlastního softwaru, tuny vlastních knihoven, který obalují jiné public knihovny. Když něco nejde, tak je to dost v háji. Samozřejmě pomůže se zeptat živých kolegů, ale vzhledem k tomu, jak špatně se ty informace shánějí, tak tady se ti asistenti absolutně nechytají.“ P6-BE

Dokonce i zvolená strategie verzování softwaru na projektu může ovlivnit schopnost asistentů kódování vracet relevantní odpověď. Ze zkušeností participanta P8-FE vyplývá, že použití asistentů kódování přímo ve vývojovém prostředí na větších projektech uložených v monorepozitáři[25] se jeví jako nereálné. Hlavním omezením je nedostatek operační paměti pracovní stanice potřebné pro správné fungování asistenta. Participant tuto situaci řeší tak, že asistenta používá ve webovém prohlížeči a nahrává mu pouze relevantní kontext pro řešený problém.

„Důležitý aspekt je, jakým způsobem jsou projekty uložené v Gitu. Teďka dělám na projektu, který je uložený ve velkém monorepu [strategie/způsob správy kódu]. A to je tedy pro asistenta oříšek, protože to monorepo je obrovský, komplikovaný a provázaný. Popravdě se mi zatím nepovedlo najít asistenta, který by mi byl schopen radit přímo v kontextu toho IDE. Většinou v takovém případě si musím ten kód kopírovat vedle do prohlížeče, protože jinak mi napíše, že mi nestačí paměť nebo že na té úrovni, co si platím, to není prostě schopné zvládnout.“ P8-FE

3.3.2 Fungující způsoby užití asistentů kódování

Podkapitola se věnuje popisu fungujících způsobů využití asistentů kódování. Identifikované způsoby lze rozdělit do dvou hlavních skupin. První tvoří využití zaměřené na usnadnění zejména repetitivní práce. Do druhé lze zařadit způsoby užití, které pomáhají programátorům při řešení úkolů, získávání znalostí a procesu učení. Každá skupina bude dále v podkapitolách popsaná včetně konkrétních případů užití.

Usnadnění práce při tvorbě kódu

Z provedených rozhovorů bylo identifikováno celkem šest hlavních způsobů využití asistentů kódování, které vývojáři nejčastěji používají k usnadnění si repetitivní práce. Patří mezi ně tvorba unit testů, mapování objektů, CRUD operace a práce s databází, vykonávání

[25] Strategie správy kódu softwaru, při níž je využit jeden velký repozitář pro více projektů nebo celou firmu (Brousse, 2019).

rutinních a pomocných prací, předpříprava dat, přepis kódu do jiných programovacích jazyků, dokumentace kódu a návrh jednoduchých frontendových komponent.

Tvorba unit testů

Suverénně nejčastěji zmiňovali participanti v kategorii usnadnění práce jako nejlépe fungující způsob užití pomoc s vytvářením unit testů. Právě psaní testů vnímá většina participantů jako zdlouhavou a méně zábavnou část své práce, se kterou jim asistenti velmi dobře pomáhají. Participanti P3-FS a P5-BE se shodují, že například na testování CRUD operací fungují asistenti velmi dobře. Podle P3-FS stačí mít připravené metody k otestování v dané třídě a asistent je schopen testy vygenerovat včetně pozitivních a negativních scénářů. Participant P5-BE navíc doplňuje, že pro zpřesnění a zlepšení výsledků generovaných odpovědí pomáhá, když jim programátor vytvoří první vzorový test.

> *„Generování testů, na to je fakt dost silnej. Řekla bych, že v testech je možná nejsilnější z těch všech oblastí.“ P1-FE*

> *„Naprosto precizně a skvěle funguje dopisování unit testů. Jakmile zná asistent kontext, ví, že mám třídu uživatel, v ní mám funkce uprav, smaž, vyhledej podle jména atd., tak jemu je jasné, jaké budu potřebovat testy, aby protestoval všechny funkce, včetně jejich negativních scénářů, a rovnou je začne generovat.“ P3-FS*

> *„Když chceš udělat test na nějaké CRUD operace, tak tam nemá problém. A ještě je úplně nejlepší, když mu nejdřív uděláš první test, tak ty zbylé už umí nakopírovat jako celkem hezky. Ono obecně je schopný projít success scénář. Nebo když vidí, že zrovna mockuješ, tak ti pomůže s mockováním dalších dílčích elementů. Když mu dáš negativní scénáře, které většinou bublají strašně rychle, tak ty dokáže udělat v podstatě okamžitě.“ P5-BE*

Jako obzvlášť velkou úsporu vnímá P6-BE asistenty v případě, kdy se testují buď velké korporátní knihovny, nebo je třeba mockovat velké množství dat. Se schopností pomáhat s mockováním dat souhlasí i P5-BE.

> *„Za sebe ho nejraději používám na psaní testů. To je zdlouhavá, úmorná činnost. Hlavně když pak testuješ software korporátní, tak tam to jsou desítky až stovky řádků kódu na test. Obzvlášť velká úspora to je, když se mockujou data a responsy, kde už nevoláš jen jednoduché asserty. Tak tam to čas šetří úplně brutálně. S tím asistentem ti kolikrát stačí kód toho testu mnohdy jenom zkontrolovat, protože ten Copilot už ví podle názvu testu, co chceš testovat, ví, jaký na to používáš knihovny v projektu, a nageneruje opravdu stovky řádků kódu, který bych dělal normálně třeba půl dne.“ P6-BE*

Velkým faktorem ovlivňující kvalitu prováděných unit testů je komplexita testované třídy. Podle participanta P5-BE platí pravidlo, že se zvyšující se komplexitou testované třídy klesá přesnost a kvalita vygenerovaných testů. Podle jeho zkušeností vracejí asistenti nejkvalitnější testy k třídám, které mají nižší stovky řádků kódu.

> *„Dokáže testovat i custom logiku. Nemusí to být nutně jen CRUD operace, ukládání do DB atd. Co ale výrazně ovlivňuje kvalitu toho testu, je komplexita testované třídy. My v projektu občas máme metody na 500 až 1000 řádků a tam už to dobře neotestuje,*

neudělá dobré testy. Pro kvalitní testy je určitě lepší s ním testovat o něco kratší třídy. Ideálně v řádu fakt nižších stovek řádků.“ P5-BE

Mapování objektů, CRUD operace a práce s databází

Participanti P3-FS a P7-FS si pochvalují pomoc asistentů kódování při mapování objektů mezi sebou. Podle nich stačí mít vytvořené objekty, ze kterých na které se bude mapovat. Následně stačí napsat ručně první atribut a asistent je dále schopen provést mapování skoro sám. Podle participanta P7-FS se navíc tato schopnost ještě zlepšuje, jedná-li se o nový projekt.

„Poměrně dobře funguje, když mapuju jeden objekt na jiný. Napíšu jeden řádek, dám ENTER a on vlastně předpokládá, že když jsem namapoval jeden řádek, třeba jméno, tak ví, že uživatel má kromě jména ještě příjmení, rodné číslo… A protože je to takhle pod sebou a já psal mapping toho prvního atributu, tak automaticky nabídne ten druhej. Já vlastně místo toho, abych neustále psal, tak dám tabulátor a ty řádky sami naskakujou.“ P3-FS

„Když jsme na jaře psali novou aplikaci, tak tam mi docela pomáhal. V momentě, kdy jsem dělal mapování mezi jednotlivými objekty, tak mi pěkně napovídal, jak vlastně ten mapper udělat, aby novému objektu přiřadil parametry ze starého. Tam jsem prostě potom mačkal jen TAB, TAB, TAB a ono to docela pěkně jelo samo.“ P7-FS

Jako další a poměrně často využívaný způsob užití asistentů kódování zmiňovali vývojáři jejich použití pro vytváření CRUD operací nad určitým objektem. Podle participanta P3-FS a P5-BE stačí podobně jako u mapování objektů napsat libovolnou CRUD operaci a asistent je schopen podle kontextu našeptávat kód se zbytkem potřebných operací. Implementace jednotlivých operací je pak ještě přesnější, pokud v projektu vývojář již psal dotazy do databáze, nebo má ve vývojovém prostředí otevřené relevantní soubory pro vytvářené operace.

„Máš velkou třídu s CRUD operacemi. Mě stačí napsat create user a už podle toho ten asistent ví, že minutu předtím jsem někde v souboru, který mám i otevřený, nakódil dotaz do databáze, tak mě rovnou nabídne obsah té metody. Když pak napíšu hnedka pod to na řádek delete, tak on mi dokáže extrémně přesně vygenerovat zase úplně celou funkci.“ P3-FS

„Když máš otevřenou nějakou třídu uživatel, nebo user service, píšeš k tomu implementaci, uděláš nad tím třeba create a pak začneš psát read operaci, tak on ti ten její vnitřek doplní. Takže mačkáš tabulátor a už nad těmi dalšími metodami nemusíš moc přemýšlet. A i když mu tu operaci customizuješ, že to není jen jednoduchý insert do databáze, ale že tam uděláš ošetření vstupů, případně nějakou návratovou hodnotu, tak on je schopnej to reflektovat i v tý další metodě.“ P5-BE

Participant P5-BE zmiňuje i konkrétní příklad využití asistentů při rozšiřování informací ukládaných do databáze a usnadnění jejich následného získávání na jiném místě v aplikaci.

„Teďka děláme u nás úpravy, kde jsem potřeboval rozšířit základní informace o uživateli ukládané do databáze. Tak jsem rozšířil tu část, kde informace ukládáme. Pak na jiném místě jsem si ty informace potřeboval z databáze zase načíst a plnit je do

odpovědi. No a tam jsem se už fakt spoléhal na to, že budu jen tabelátorovat. On si pamatoval, že pět minut předtím jsem to [informace o uživateli] přidal a už přes ty tabelátory mi to přesně vyfrkal.“ P5-BE

Rutinní a pomocné práce, předpříprava dat

Kromě generování unit testů si participanti P2-BE a P5-BE asistenty kódování chválí i v případech, kdy potřebují buď připravit data pro svůj další vývoj, nebo když potřebují udělat určitou rutinní nebo přípravnou práci. Často se může jednat o kód, který ani nebude součástí finálního dodávaného řešení, ale jeho rychlé vytvoření vývojáři výrazně usnadní nebo urychlí další práci. Participant P5-BE tento typ úkolů popisuje jako úkoly, které se relativně často dávají ke zpracování juniornějším kolegům.

„Já to poměrně dost používám na pomocný práce, když člověk potřebuje třeba připravit si nějaký data. Stačí asistentovi popsat zadání, aby mi vytvořil jednoduchý skriptík na vytvoření dat. Ono si člověk řekne příprava dat, to je jednoduchý. Sice to nejde do finálního kódu, ale fakt mi to pomáhá, že pořád nemusím vymýšlet nějaká data, se kterými budu pracovat. Takže za mě takové pomocné, přípravné věci, tak si myslím, že to docela zvládá a mě to pomáhá.“ P2-BE

„Já bych to obecně shrnul tak, že to dobře zvládá takové ty pomocné a rutinní práce, co dáváš většinou juniorům. Dáš mu jednoduchý úkol – jeden soubor, jedna třída, jeden test, a on [junior] to dělá celý den nebo dva dny. Tady to dáš asistentovi a pokud nemá zrovna nějaký výpadek, že je naprosto mimo, tak ti s tím velmi rychle pomůže. Určitě rychleji než ten junior.“ P5-BE

Přepis kódu do jiných programovacích jazyků

Dalším způsobem využití asistentů kódování pro usnadnění nebo vylepšení práce je přepisování kódu z jednoho programovacího jazyka do druhého. Konkrétní zkušenost s tímto využitím má participant P5-BE, který používá asistenty kódování k úpravě vytvořeného kódu v programovacím jazyce, který není jeho primárním. Cílem tohoto způsobu použití je zajistit, aby výsledný kód co nejvíce odpovídal konvencím a zvyklostem běžně používaným v daném jazyce.

„My na aplikaci nové věci děláme v Kotlinu, ale původní funkce máme v Javě. Já jsem Javista, i když je ten Kotlin docela podobný Javě, ještě ho nemám tolik navnímaný. Napíšu něco v Kotlinu, pak mu napíšu, udělej z toho víc Kotlin. A on ten můj kód vezme a snaží se na něj aplikovat různá ta Kotlin pravidla a zvyklosti.“ P5-BE

Dokumentace kódu

I když participant P7-FS nepoužívá asistenty kódování pro generování celé dokumentace vytvářeného kódu, osvědčilo se mu používání asistentů pro generování JavaDoc anotací před jednotlivými metodami a třídami. Podle participanta se opět jedná o poměrně rutinní činnost, kterou asistent zvládá vzhledem k již hotovému kódu snadno a dobře. Navíc participantovi přináší tento způsob užití i velkou úsporu času.

„Umí to generovat jako celou dokumentaci, to ale moc nepoužívám. Kde je [asistenty], co se týče dokumentace, používám je generování JavaDoc. Na to jsou fakt dobrý, protože to je fakt většinou jen rutinní práce, například vyjmenovat parametry a popsat jejich význam, tam mi to fakt šetří čas.“ P7-FS

Tvorba jednoduchých frontendových komponent

Na frontendu využívají vývojáři asistenty kódování mimo jiné ke generování a stylování jednoduchých, často se opakujících komponent nebo HTML stránek. Konkrétní zkušenosti s tímto způsobem generováním mají participanti P7-FS a P8-FE. Participant P8-FE dokonce v některých případech nechává AI generovat i složitější komponenty.

„Nedávno jsem potřeboval udělat nějakou úvodní jednoduchou stránku. Říkal jsem, nemá smysl na to dělat samotnou React aplikaci, udělám prostě HTML s trochou CSS a možná JavaScriptu. Tak jsem sepsal prompt, aby mi tu stránku vygeneroval. Dostal jsem HTML i CSS kód. Samozřejmě jsem to upravil, ale tady jsem ho využil dost na generování kódu.“ P7-FS

„Zejména, pokud to jsou nějaké jednoduché opakující se komponenty (třeba přihlašovací obrazovka) v Reactu nebo komponenty, co se mi nechtějí psát, tak mu řeknu, ať mi napíše jednoduchou komponentu. Často to jsou věci, co nejdou nijak pokazit, spíš taková nudná rutina. Ale někdy, když se mi nechce přemýšlet, tak ho nechám dělat i složitější projekty.“ P8-FE

S tímto způsobem použití ovšem úplně nesouhlasí participantka P1-FE, která si myslí, že zejména při stylování a vytváření frontendových komponent jsou schopnosti asistentů omezenější. Z toho důvodu asistenty kódování k těmto činnostem tolik nepoužívá.

„Ono obecně, když píšu ty frontendové komponenty, co se uživateli zobrazí, tak vlastně on není moc schopnej odhadnout, který elementy tam chci dát, nebo jak to chci nastylovat, což je logický, takže na to ho používám nejmíň.“ P1-FE

Hledání informací, možných řešení a rozšiřování znalostí

Kromě usnadnění repetitivní práce využívají programátoři asistenty kódování také k vyhledávání potřebných informací, vysvětlování kódu, hledání chyb, návrhu možných řešení, konzultaci problémů, rozšiřování svých znalostí, optimalizaci kódu a osvěžení si méně často používaných jazykových konstrukcí. Právě popisu těchto způsobů užití se budou následující podkapitoly.

Vyhledávání informací

V oblasti vyhledávání informací participanti nejčastěji využívají asistenty kódování jako alternativu k tradičním vyhledávačům. To, co dříve vyhledávali přímo na internetu prostřednictvím vyhledávačů, dnes řeší pomocí asistentů kódování. Podle participanta P2-BE je velkou výhodou, že mají asistenti přístup zpravidla k celému projektu. To pomáhá k získání relevantnějších odpovědí, pokud si programátor není jistý, co přesně má hledat.

„Navíc ten chat má výhodu, že má kontext toho celého projektu. To je třeba výhoda v tom, že občas člověk neví, na co se zeptat, ale ten asistent tohle často odbourává. Když nevíš, na co se zeptat, tak to maximálně popíšeš vlastními slovy a on ti na základě toho, co vidí v projektu, poradí.“ P2-BE

Kromě již popsaných výhod používání asistentů kódování při vyhledávání informací si participant P5-BE vychvaluje schopnost asistentů agregovat a shrnout všechny relevantní zdroje v několika bodech. Díky tomu pak nemusí procházet jednotlivé zdroje, ale má je shrnuté všechny na jednom místě.

„Takže to používám místo toho, abych nemusel psát do Googlu, který ti najde a vypíše 100 odkazů, ty se tím musíš prohrabávat a hledat. Takhle mi Copilot podle mého dotazu všechno zagreguje, všechno máš na jednom místě, a hlavně se nemusíš prohrabávat stovkama odkazů.“ P5-BE

Participant P9-FS navíc rád využívá asistenty kódování k získání dodatečného kontextu prováděné změny. Kromě hledání snadno dostupných zdrojů jako jsou legislativní pravidla a další veřejně dostupné informace týkající se zpracovávaného požadavku, používá asistenty i k identifikaci možných specifických a okrajových způsobů uživatelského chování. Tím nejenže provádí dodatečnou kontrolu kompletnosti zadání, ale také si identifikuje uživatelské chování, které musí aplikace zvládat nebo ho musí programátor ohlídat.

„Kromě takového toho dodatečného hledání informací (třeba zákonů), které se týkají požadavku a které potřebuješ pro správné zpracování, je [asistenty kódování] používám třeba i na získávání limitů aplikace. Takové to chování, co by mohl uživatel chtít dělat, nikoho to nenapadne a ta funkce by mohla chybět, nebo by nám padala aplikace. V takových případech dám asistentovi roli, že je uživatel, který bude používat danou funkcionalitu. Následně se ho ptám na takové řekněme typické vlastnosti požadavku, které by tam takový uživatel dával, jak by ho používal. Vlastně z toho já přicházím na to, že si definuji některé ty vlastnosti a podmínky, které nejsou v zadání explicitně definované, ale ve finálním produktu by měly být ohlídané.“ P9-FS

Vysvětlování kódu a hledání chyb

Mezi časté způsoby užití asistentů kódování ze strany vývojářů patří mimo jiné vysvětlování fungování různých částí kódu. Například participant P4-BE tuto funkci využívá zejména tehdy, pokud pracuje na kódu, na němž již dlouho nepracoval, nebo když se setká s kódem úplně novým. Funkci považuje za obzvláště užitečnou u komplexnějších projektů, kde původní autoři kódu již nepracují, a ani na projektu déle pracující kolegové nevědí, jak daný kód funguje.

„Když děláš na nějakém kódu, na kterém si dělal nedávno, tak toho asistenta de facto ani nepotřebuješ. Když ale děláš na nějaké storce, která je na druhé straně kódu, který jsi nedělal nebo si ho v životě neviděl, tak si nechám od toho asistenta třeba nějakou metodu nebo nějaký řádek vysvětlit, ať mi řekne, co to dělá. I když u nás na projektu jsou lidi, kteří jsou tam o hodně déle než já, tak některé kusy nikdy neviděli, tudíž se nemáš v tu chvíli koho zeptat, jak něco funguje. Protože ti lidi, co to psali, už tam dávno nejsou. A ten Copilot je schopen ti tam vlastně to zanalyzovat a poradit.“ P4-BE

Podobně využívá asistenty kódování i participant P6-BE. Ten ovšem nevztahuje způsob využití pouze na neznámý kód, ale popisovanou funkci využívá v obecné rovině. Navíc se mu líbí, že u jednotlivých částí implementace se může dle potřeby doptávat na různé detaily.

> *„No, když si nejsem někde něčím jistý, co dělá tato funkce, co dělá tato třída, tak se ho zeptám. On je schopný to klidně krok po kroku popsat a já se doptávám, co jednotlivé kroky dělají.“ P6-BE*

Participantka P1-FE využívá asistenty kódování nejen k vysvětlování fungování kódu nebo hledání dodatečných informací, ale také k hledání chyb, které znemožňují správné fungování aplikace. Velmi podobně přistupuje k použití asistentů kódování i P3-FS. Pokud si není jistý příčinou jakékoli chyby, přidává k promptu konkrétní chybovou hlášku a obvykle očekává buď vysvětlení chyby a její příčiny, nebo konkrétní návrh opraveného kódu.

> *„…druhý způsob je, že mu označím části kódu a zeptám se ho, proč tohle nefunguje, jestli tam nevidí nějaký problém.“ P1-FE*

> *„Já asistenty teďka používám pro zjišťování informací a doplňování si třeba kontextu. Ať je to doplnění informace k syntaxi, nebo když vidím nějakou chybu. Samozřejmě mohl bych to dát do Googlu, ale proč bych to dělal? Dneska to dám do asistenta, přidám k tomu rovnou kód a on mi napíše, kde je chyba a rovnou mi nabídne řešení.“ P3-FS*

Hledání možných řešení

Jedním ze způsobů použití asistentů kódování, který využívají P1-FE a P3-FS, je získání řešení konkrétního problému. Přestože by úkol dokázali vyřešit sami, nechtějí nad řešením trávit zbytečně mnoho času, nebo se jim nechce dohledávat potřebné podklady. Občas se při použití asistentů kódování tímto způsobem může stát, že se programátor naučí novou konstrukci jazyka anebo objeví lepší způsob řešení problému, než by sám použil.

> *„S Copilotem obvykle řeším problémy typu, že něčeho chci dosáhnout, ale nechce se mi to vymýšlet nebo hledat, protože to je delší cesta než se ho zeptat. Prostě když nevím přesně, jak na to, tak se ho zeptám, jak bych to mohla řešit. A kromě samotného řešení si často říkám i o odkazy do dokumentace, abych se na to mohla podívat do hloubky.“ P1-FE*

> *„Já se vlastními slovy přímo ptám, jak něco udělat. Většinou to bývají fakt elementární věci, nad kterými se mi třeba nechce přemýšlet. Př.: chci získat uživatele s tím a tím jménem a určitým rokem narození. On mě dá nějaký kód, zkontroluju ho a většinou stačí upravit názvy atributů podle kontextu aplikace. Příjemným doprovodným efektem pak může být, že mi občas ukáže i nějakou novou funkci, kterou jsem do té doby neznal.“ P3-FS*

Kromě využití asistentů kódování k vyřešení zadaného problému se mezi participanty objevují i další přístupy. Například participant P4-BE asistenty využívá nejen k nalezení konkrétního řešení, ale také jako zdroj inspirace pro možná alternativní řešení. Nejprve asistentovi popíše svůj problém a podle toho, jak dobře jej asistent pochopí, se vývojář rozhodne, zda převezme celé řešení, jeho část, nebo pouze základní myšlenku.

„Zeptám se ho na přímo na nějaký postup, ať mi řekne podle toho mého slovního zadání, jak by šlo něco udělat. Někdy je to opravdu, že použiju jenom jeden řádek, někdy použiju jen nápad, jindy použiju celé navrhované řešení i s kódem. Rozhoduju se většinou podle řešeného úkolu a podle toho, jak je schopen [asistent] vnímat ten kontext, protože když ho nechytne, tak tam začne dávat třeba názvy proměnných, které jsou úplně jiné, než máš v kódu.“ P4-BE

Velmi podobně k řešení problému přistupuje i participant P7-FS. Ten navíc přidává ještě další fázi, ve které se na asistenty kódování obrací, když už je rozhodnutý pro implementaci určitým způsobem, ale narazí během ní na problém, se kterým si neví rady, nebo o něm potřebuje zjistit další podrobnosti.

„Hodně ho používám na nabrání těch myšlenek, když si rozmýšlím, jak bude co fungovat. Často si nechám vygenerovat nějaký kus kódu, jak by problém řešil. Pokud je to podle mě správné řešení, pak ho použiju. Pokud mi naopak nevyhovuje, pracuju si na implementaci dál sám. No a pak se k němu vracím v momentě, kdy se dostanu k nějakým problémům, když mi něco nefunguje nebo něco není jasný, tak pak řešíme nějaké ty konkrétní věci, jak se problému zbavit.“ P7-FS

Konzultace problému a porovnání navrhovaných řešení

Mezi další způsoby použití asistentů kódování patří konzultace řešeného problému. Participant P6-BE dává příklad, že s asistenty často konzultuje řešené úkoly ve chvíli, kdy má již vymyšlené a implementované vlastní řešení, ale vyskytne se v něm chyba, s níž si programátor neví rady. Participant oceňuje, že odpovědi asistenta bývají na úrovni ostatních seniorních kolegů. Navíc dostane od asistentů konkrétní seznam kroků, které může vyzkoušet a zkontrolovat.

„…dál ho použiju k tomu, že v projektu něco mám špatně, něco mi nejede, tak tam s ním rád konzultuju. Ty jeho odpovědi mi často připadají, jako kdybych se radil se seniorním kolegou. Tam je za mě fajn, že on řekne zkontroluj to, to a to a kolikrát to jsou fakt věci, který by tě ani nenapadly, ale je v nich problém. Případně víš, čeho se můžeš chytnout.“ P6-BE

Participant P7-FS využívá asistenty kódování nejen k opravě nefunkčního kódu, ale především jako tzv. „gumovou kachničku“. Jedná se o způsob, kdy vývojář s asistenty konzultuje svůj postup. Zpravidla diskutuje již konkrétní návrh nebo zadání. I přes to, že ne vždy dostane participant správnou nebo uspokojivou odpověď, už samotná diskuse problému mu pomáhá utřídit si myšlenky a může mu ukázat řešení, které by ho do té doby nenapadlo. Podle participanta lze v určité míře asistenty použít také na konzultaci architektonických otázek.

„Já je využívám především jako gumovou kachničku. Když řeším nějaký problém, chci si rozvrhnout, jak něco budu dělat, nebo pak mi to z nějakého důvodu nejde, tak napíšu do chatbota. Prostě mu předložím nějaký problém, co řeším, nebo kus kódu a doptávám se. Většinou mi neodpoví na první dobrou správně, kolikrát mi neodpoví správně vůbec, ale to, jak si s ním o tom povídám, tak mě buď on navede na nějakou myšlenku, anebo já si přijdu prostě na řešení sám a v tomhle mi to dost pomůže.“ P7-FS

„Když jsme navrhovali jednu aplikaci úplně od začátku, tak jsem ho třeba používal na návrh jednotlivých vrstev. Jak mezi sebou budou interagovat, nebo i databázového modelu, jak by problém rozdělil do jednotlivých entit. Myslím, že jsem tam byl vždycky docela spokojený s tím, co mi říká. Samozřejmě jsem to nebral, že to musím takhle udělat, ale spíš jenom na to utříbení myšlenek." P7-FS

Velmi podobně asistenty využívá i participant P8-FE. Ten navíc zmiňuje i jednu z velkých výhod tohoto přístupu, kterou je mnohonásobně menší zátěž ostatních kolegů, s nimiž by jinak podobné konzultace probíhaly.

„Mě na začátku mé kariéry naučil jeden senior takovou věc. Jednou na Vánoce všem svým podřízeným přines kachničku. Tím chtěl docílit toho, že má taky svoji práci a není tady jen od toho, aby nám odpovídal na otázky. Říkal, nejdřív se nad problémem zamyslete, něco si najděte, až pak se mě ptejte. Od té doby jsem vždycky nosil do práce svoji kachničku. Nicméně od adopce Chat GPT a Copilota jsem svoji kachničku nepotřeboval. Vlastně tyhle nástroje mi ji plně nahradily, a navíc mi ještě usnadnily samotné vyhledávání informací o problému." P8-FE

Dalším možným způsobem, jak asistenty v rámci zmíněných konzultací využít, je porovnání dvou možných návrhů řešení, mezi nimiž se programátor nemůže rozhodnout. Tento způsob úspěšně využívá P8-FE. Ten vždy navrhne podle požadavků možná řešení a následně požádá asistenta, aby mu je porovnal a podle definovaných kritérií mu pomohl vybrat v kontextu projektu to efektivnější. Přínosem postupu je pro participanta nejen usnadnění rozhodování, ale i získání dalších vlastností řešení, o nichž dosud nevěděl. Je ovšem zásadní, aby programátor asistentovi předložil dvě alespoň částečně připravená řešení. Podle participanta P8-FE by asistent nebyl schopný vymyslet a vybrat vhodné řešení pouze na základě původního zadání. Nevhodnost použití asistenta kódování pro přijímání rozhodnutí na základě úvodního zadání potvrzují i zkušenosti participanta P2-BE, který při tomto způsobu použití odhaduje spolehlivost asistentů okolo 50 %.

„Nedávno jsem ho použil docela zajímavě na jednom projektu, kde jsem měl dvě vymyšlená řešení. Vůbec jsem se nemohl rozhodnou, který z těch návrhů je lepší. Tak jsem mu [asistentovi] dal zadání, aby mi je srovnal, řekl mi jejich pozitiva a negativa a rozhodl se pro efektivnější v kontextu projektu. On mi dal pár informací navíc a ukázal mi věci, které jsem nevěděl nebo neviděl. V tomhle mi jako výrazně pomohl si nakonec vybrat. Nicméně jsem přesvědčenej, že kdybych mu dal zadání toho problému, a nejen dvě vymyšlený řešení, tak by nebyl schopný nic udělat. Já jsem je připravil, ale on mi pomohl se rozhodnout." P8-FE

„Zkoušel jsem asistenty používat i k nějakým návrhovým rozhodnutím, aby se podle dostupného projektu a zadání rozmysleli a navrhli, jakou cestou se mám vydat. Tam si myslím, že doteď je to tak půl na půl. V polovině případů si myslím, že mi to jako pomohlo a že to navrhlo něco, co se ve finále dalo použít. Ve zbytku případů si myslím, že jsem se zbytečně kvůli tomu točil v kruhu, že mě to svedlo na špatnou cestu nebo se to [navrhované řešení] nedalo použít." P2-BE

Rozšiřování znalostí a optimalizace kódu

Asistenty kódování lze podle participantů P1-FE a P3-FS využívat i k rozšiřování svých znalostí a dovedností. Participantka P1-FE zdokonaluje své znalosti pomocí asistentů

kódování tím, že si od nich nechává generovat elegantnější a technicky dokonalejší řešení ke svému kódu. Podle participanta P3-FS lze asistenty využít k naučení se nových dovedností prostřednictvím pokládání doplňujících dotazů při řešení konkrétního problému. Oba postupy lze dobře použít nejen při osvojování si nových znalostí, ale i při zdokonalování se v dovednostech stávajících.

> *„Občas nakódim něco, co si myslím, že by šlo napsat líp, ale nevím přesně jak. Tak mu označím část toho kódu a zeptám se, jestli existuje nějaká elegantnější syntaxe. On mi navrhne nějaké řešení a většinou se naučím něco nového.“ P1-FE*

> *„Občas si zkouším, když mám vyřešit konkrétní problém, nechat pomocí asistenta vygenerovat takovou osnovu, jak by problém řešil. Následně vidím, že je tam třeba rozdíl oproti tomu, co bych použil já. Tak se ho zeptám na doplňující otázky. Třeba jaká je výhoda oproti mému návrhu. Můžu se dál doptávat, od které verze se jím vygenerovaná syntaxe používá atd. Tím si rozšiřuju poměrně příjemným a rychlým způsobem svoje znalosti.“ P3-FS*

Kromě samotného rozšiřování znalostí se asistenti kódování participantům P3-FS a P4-BE osvědčili i pro optimalizaci a zjednodušování vytvořeného kódu nebo algoritmu. Pro vývojáře pak může být přidanou hodnotou tohoto použití možnost se dodatečně doptávat, v jakém ohledu je upravený kód efektivnější.

> *„…pak další použití mám, když chci třeba zoptimalizovat kód, dám dotaz a hned mi vlastně přepíše můj kód a řekne, takhle to je lepší nebo efektivnější. A opět se ho můžu zeptat, proč a v čem.“ P3-FS*

> *„Na nějaké zjednodušování algoritmu to taky občas používám, tam to taky bývá docela silný.“ P4-BE*

Osvěžení si méně často používaných konstrukcí

V neposlední řadě zmiňují účastníci P4-BE a P5-BE jako fungující řešení osvěžení si konstrukcí programovacího jazyka, které nepoužívají tak často. To zpravidla dělají tak, že si nechají vygenerovat základní konstrukci, díky které si práci zopakují a následně s konstrukcí dále bez problémů pracují a upravují si ji podle svých potřeb.

> *„Když jsou to věci, které už jsem třeba dlouho nikde nepoužil a zapomněl jsem je, tak se ho ptám více. To mi hodně pomůže si ten kód osvěžit. Většinou mi nahodí nějakou nápovědu, já si vzpomenu a pak už pracuju sám.“ P4-BE“*

> *„Pak to hodně používám na takové věci, které člověk zapomněl, protože je děláš jednou za kvartál. My na projektu třeba moc nenačítáme data ze souborů. Pořád máme API a někam se připojujeme. Ale načtení dat ze souboru tolik neděláš. Tak napíšeš Copilotovi, v Javě nebo v Kotlinu mi udělej načtení ze souboru, on ti tam hodí tu rutinu, kterou zkopíruješ a pak už si vzpomeneš a jedeš dál sám.“ P5-BE*

3.3.3 Případy, kdy vývojáři asistenty kódování nepoužívají

I přes relativně velkou oblibu asistentů kódování ze strany vývojářů existují způsoby užití, kdy programátoři preferují a volí při své práci jiný nástroj. Tato podkapitola se věnuje nejen

popisu, proč je v některých případech rychlejší si úkol vyřešit sám bez použití asistentů, ale ukazuje také konkrétní příklady řešených úkolů a problémů, kdy se použití asistentů podle účastníků rozhovorů nehodí.

Proč je někdy rychlejší si úkol vyřešit sám

Participanti P3-FS, P5-BE, P6-BE a P7-FS se shodují, že existují takové druhy úkolů, kdy využití asistentů nedává z časového hlediska smysl. Jedná se buď o velmi jednoduché a rychlé úpravy v kódu, nebo naopak o úkoly, kde je třeba asistentovi napsat velké množství kontextu, aby byl schopen úkol správně zpracovat. Nicméně délka zadání, které musí vývojář napsat, aby asistent úkol zpracoval správně, je neúměrně dlouhá množství času potřebného pro dokončení úkolu samotným vývojářem bez použití asistentů kódování.

> *„…než napíšu to zadání, tak já si samotné řešení vymyslím a napíšu rychleji. To znamená, abych mu dal dostatečný kontext, já už vlastně rovnou ten kód píšu a vím, že je dobře a nemusím ho kontrolovat. Protože, kdybych napsal komentář, který by mě napsal odpověď, tak bych si celou tu odpověď musel přečíst a říct si třeba, no já to chci trošku jinak. Znova bych si to musel přečíst, zkontrolovat atd.“ P3-FS*

> *„…abych s tím neztrácel čas, protože vím, že napsat něco takového mi trvá 5 minut, ale abych ho ukecával a on mi to udělal za 10 minut, tak ho raději nevyužiju.“ P5-BE*

> *„Ve spoustě věcí mi práci usnadňují, ale existují i takové úkoly, kde než bys popsal tomu chatbotovi, co vlastně potřebuješ, jaký jsou tam omezení, ať už technický, byznysový, jak vedeš strukturu v tom projektu, tak to je prostě absolutně časově neefektivní.“ P6-BE*

> *„Neefektivní časově je to hlavně u úprav něčeho, kde vím, co je chyba nebo kde ji hledat. V takovém případě mi přijde rychlejší jen přepsat si ručně tu věc, kde vidím, že je tam problém, než vzít CTRL + C, CTRL + V a pak kontrolovat, jestli náhodou se náhodou nezrušilo tou vygenerovanou odpovědí něco jiného v tom původním kódu.“ P7-FS*

Zpracování úkolu pouze na základě slovního popisu

Většina participantů interview (P1-FE, P2-BE, P3-FS, P5-BE a P6-BE) se podle vlastní zkušenosti shodla, že asistenty kódování nelze zatím efektivně použít ke zpracování úkolu pouze na základě zadaného slovního popisu nebo připojení relevantní části technické specifikace požadavku. Pro participanty se jedná o scénář, který asistenti nejsou v tuto chvíli rozhodně schopni pokrýt. Při zpracování požadavku asistentem je tedy skoro vždy nutné, aby do zpracování zasáhl člověk.

> *„On [asistent] fakt není schopen si dotáhnout zadání třeba z Jiry nebo Confluence. Asi neexistuje, že bych mu řekla, tady máš specifikaci, udělej to podle ní. Jako už je schopnej si natáhnout informace z codebaseu a zanalyzovat, co je důležité a co ne. Ale furt to rozhodně nestačí na zpracování celého zadání.“ P1-FE*

> *„Rozhodně si nemyslím, že by to bylo vhodné používat na to, že by člověk jenom v rámci nějakého toho chatovacího okna zadal nebo popsal požadovaný výstup jenom high level nebo businessově a čekal by, že dostane celou hotovou storku.“ P2-BE*

„Když asistenti začínali, tak se říkalo, stačí napsat komentář např. chci vyfiltrovat uživatele, kteří mají takové první písmeno ve jménu atd. a následně ten asistent by to měl být schopný vytvořit. No, popravdě na začátku a ani teďka mě vlastně tohle téměř nikdy nefungovalo dobře.“ P3-FS

„Nedovedu si představit, že bych to použil na nějaký větší projekt, že bych mu jen hodně ze široka popsal, co je třeba. Podle mě je to fakt jenom na takové dílčí úkoly, na to si to dokážu představit. Na komplexních úkolech to hodně rychle havaruje.“ P5-BE

„No tak samozřejmě na tu vyšší úroveň abstrakce nebo tvorbu čistě business logiky to určitě není.“ P6-BE

Konfigurace a DEVOPS

Další oblastí, kde asistenti kódování nejsou podle participantů silní, je pomoc s konfigurací aplikace, webových serverů a otázky týkajících se DEVOPS. Participanti zmiňují, že se nejedná pouze o případy, kdy se v konfiguraci vyskytne problém, ale jde i o případy, kdy potřebují získat obecnou radu. V této oblasti vidí P2-BE velký prostor pro zlepšení. Participant P3-FS dokonce zmiňuje, že u otázek týkajících se DEVOPS je v jeho případě lepší a rychlejší si dojít pro přesnější odpověď přímo za kompetentním kolegou.

„Snažil jsem se s ním prostě v práci řešit různé otázky konfigurace aplikace. Bylo to hlavně v okamžiku, kdy to [aplikace] člověku neběží, ale i v případech, kdy jsem vytvářel nebo spravoval nějaké nové věci. V tomhle ohledu mají [asistenti] dost různou míru úspěchu. A vidím tady velký prostor pro zlepšení.“ P2-BE

„Čím víc DevOps jsou to otázky, tím je [asistent] víc nemožný, tím mě dává méně specifické odpovědi. Vždycky mě dá nějaké obecné fráze, které mi rozhodně moc nepomůžou. Jako určitě mu můžu dávat zase další a další kontext, ale to s ním strávím takového času, že už je lepší dojít přímo za kompetentním člověkem a zeptat se jeho.“ P3-FS

Hledání zranitelností v kódu

Na základě osobní zkušenosti participant P6-BE nedoporučuje využívat u asistentů kódování jejich funkci pro hledání chyb a zranitelností v kódu. Při použití této funkce se participantovi několikrát stalo, že si asistent vymýšlel různé neexistující chyby v kódu. To ve výsledku vedlo k tomu, že participant strávil nemalý čas s kolegy, aby zjistili, že celá identifikovaná chyba je pouze vymyšlená.

„Párkrát jsem plejtval svým časem tím, že jsem zkoušel funkcionalitu, kdy ti to pomáhá hledat slabiny a chyby v kódu. Postupoval jsem tak, že jsem mu dal nějaký kód s dotazem, ať kód zanalyzuje a pokusí se najít případné zranitelnosti. On to zanalyzoval, a i když tam žádný chyby očividně i podle ostatních kolegů nebyly, tak si je vymyslel. Úplně v pohodě si vzal část kódu, ve který si vymyslel, že je chyba. Takže já jsem několik hodin zbytečně hledal, jaká tam může být chyba, abych pak s kolegy zjistil, že si to všechno vymyslel.“ P6-BE

Situace, kdy vývojáři nechtějí asistenty používat

Participantka P1-FE vnímá asistenty kódování jako nevhodnou pomůcku pro učení se úplných základů programování. Zdůrazňuje, že při učení je klíčové osvojit si samotný způsob přemýšlení nad problémem. Pokud tak budoucí vývojář neučiní, asistent kódování mu sice v budoucnu v omezené míře může pomoci, ale dlouhodobě se nejedná o vhodný přístup. Pokud ovšem programátor základy programování již ovládá, pak vidí asistenty jako vhodný podpůrný nástroj, jak si osvojit více programovacích jazyků.

> *„Rozhodně bych nechtěla asistenty používat, a i jsem je odmítala, když jsem se učila programovat, aby on [asistent] programoval za mě. Teďka myslím fakt jako základy programování. Nemyslím jednotlivé jazyky, u nich je to v pohodě, tam ti to pomůže, já díky tomu zvládnu více jazyků najednou. Klíčové je ale spíš to přemýšlení nad problémem. Protože to, když se nenaučíš, tak pak jsi v háji, asistent ne asistent, a ztrácí to trošku smysl.“ P1-FE*

Jak již bylo naznačeno v předcházejících podkapitolách, participant P5-BE využívá asistenty kódování hlavně při rutinní implementaci kódu. Za užitečné nepovažuje asistenty při analýze zadání a vymýšlení architektonického návrhu řešení. V těchto fázích zpracování úkolu asistenty ani nechce používat, protože by ho zbytečně zatěžovali a vyrušovali z jeho myšlenkových procesů.

> *„Když si dělám vlastní analýzu nad zadáním, tak ho [asistenta] vůbec nechci používat. Pak, když začnu přemýšlet nad návrhem, tak tam ho také nevyužiju. A i když začnu dělat architekturu, že si navrhnu nějakou třídu, interface, test, tak když píšu jenom ten high level pohled, nepoužívám ho. Ani nechci, aby mě otravoval a zbytečně zatěžoval a já s ním musel komunikovat. Ale pak ho fakt hodně používám u té rutinní implementace, když už mám všechno rozmyšlené, navržené a nechce se mi to psát.“ P5-BE*

3.3.4 Další práce s vygenerovanou odpovědí

Podkapitola se zaměřuje na to, jak vývojáři pracují s vygenerovanými odpověďmi od asistentů kódování. Popisuje způsoby ověřování správnosti generovaných odpovědí, postupy, které vývojáři volí v případě nespokojenosti s výsledkem, a také se věnuje odhadu chybovosti a celkové kvality vygenerovaných odpovědí.

Ověření správnosti odpovědi

Participanti P1-FE a P9-FS pečlivost ověření vygenerované odpovědi podřizují kritičnosti vytvářeného úkolu. U méně kritických úkolů, mezi které patří jednoduché frontendové úpravy nebo stylování, generovanou odpověď tolik nekontrolují a rovnou ji zkoušejí začlenit do kódu. Pokud ovšem pracují na úkolu s velkým dopadem na uživatelská data nebo fungování aplikace jako takové, přistupují k ověření správnosti mnohem důkladněji.

> *„Záleží, jak moc to ohrozí stabilitu aplikace. Nemyslím jenom třeba bezpečnost jako autentizace, ale jestli se člověku neztratí nějaký data, nebo prostě se mu něco nepřestane zobrazovat. To znamená, pokud je to nějaké CSS, tak ho vezmu a zkusím.*

Pokud je to něco klíčovějšího, pak to nejdřív zkontroluju a následně až zkouším.“ P1-FE

„Když si nechám pomocí asistenta vygenerovat nějakou komplexnější část toho kódu, tak tam to kontroluji důkladně. Vložím vygenerovaný kód na místo, kde by měl být, a potom je to taková práce na půl testera, na půl vývojáře. Zkusím ověřit, jak to funguje, vytvořím si nějaké testovací scénáře ověřující, jak kód zvládá různé extrémy atd. Ale pokud jsou to takové jednodušší úkony typu, že se vytvoří nějaká logika, kterou jasně definuji, nebo je to nějaký styling, tak tam u toho to tolik nekontroluji.“ P9-FS

Druhým poměrně rozšířeným přístupem k ověřování funkčnosti vygenerované odpovědi je provádění code review. Tento způsob volí primárně participanti P2-BE, P5-BE a P6-BE. Ti popisují, že až na výjimečné případy vždy nad vygenerovanou odpovědí provádějí klasické code review, jako by jim kód předložil živý kolega. V takovém případě je pro ně klíčová jasnost a srozumitelnost kódu, jeho správné fungování a efektivita.

„Ten kód pročtu a snažím se nad tím zamyslet, jestli nemá nějaký problém, že by to prostě vedlo k nějaký chybě, jestli je to efektivní řešení. Prostě úplně stejně, jako kdybych v tu chvíli dělal takový code review nad tím, co mi to vlastně doporučilo.“ P2-BE

„Úplně stejnej princip, jako když dělám reviewčko někomu jinému. Chci, aby to splnilo přesně definované věci týkající se funkčnosti, rozuměl jsem tomu a když to odevzdám, tak aby tam nebyla pro mě žádná náhoda, nebo nějaká neznáma. Prostě jako kdyby to byl můj kód. Tohle musí být splněné stůj, co stůj. Ty odpovídáš za to, když to pošleš na produkci a ono to selže, žádné AI. Takže žádná výmluva to jsem vygooglil, nebo to udělalo AI.“ P5-BE

„Pokud jde o generování kódu, tak tam to kontroluji očima, samozřejmě nějaký zkušenosti mám, tak už od pohledu stačí přejet očima a člověk vidí, jestli se asistent alespoň trefil. Když se trefí, a to řešení alespoň na oko funguje, pak si to přečtu důkladně. Dá se to skoro připodobnit klasickému review, které musíme dělat při pull requestech.“ P6-BE

Participant P4-BE zmiňuje, že vygenerovanému kódu asistenty kódování vždy plně nevěří a nepoužívá ho, aniž by si jeho chování zkontroloval a ověřil.

„Vždycky se na ten kód podívám, protože mu nevěřím na 100 %. Takže slepě nekopíruji a nevkládám, aniž bych si ověřil, že to vlastně je v pořádku.“ P4-BE“

Participant P3-FS vygenerovanou odpověď asistenty kódování vzhledem ke stylu jejich používání nepodrobuje tak důkladné kontrole. To si může dovolit zejména vzhledem k tomu, že si nechává generovat velmi krátké útržky kódu, u nichž může hned posoudit jejich správnost. Připouští ovšem, že kdyby se způsob jeho použití změnil a on začal asistenty kódování používat pro generování větších částí kódu, pak by musel tento přístup změnit.

„Já si většinou nechávám generovat krátký kód (cca jeden až pět řádků), u kterého hned vidím, jestli je chybně nebo správně. Právě vzhledem k tomu, že je ten kód fakt krátký, tak nad tím nemusím dělat tak detailní review. Úspěšnost mého aktuálního používání je velmi dobrá, proto se vyhýbám generování velkých kusů kódu, kde jsem

měl špatné zkušenosti s kvalitou. Pokud bych se vrátil zase ke generování větších kusů, tak bych kvalitu odpovědi asi neposuzoval jinak než pomocí code review.“ P3-FS

V případě, kdy si nechává participant P6-BE generovat odpověď na problematiku, která je pro něj nová, důvěřuje asistentům více. Na jejich odpověď se nespoléhá pouze v případech, kdy asistenti participantovi vrátí evidentně nesprávnou nebo nebezpečnou odpověď. V takových případech asistenty pro dořešení problému již nevyužívá.

„Chatovací okna asistentů hodně používám, když se v daný problematice nebo technologii neorientuji, takže po nich žádám postup krok po kroku. No a buď se mi při čtení té odpovědi rozsvítí varovný maják, že tady asi někdo kecá, nebo se mi tohle nestane, pak to řešení vezmu a vyzkouším, jestli bude fungovat nebo ne. Samozřejmě nebudu kopírovat "rm -rf" a podobné srandy, ale to bych si všiml.“ P6-BE

Postup, když není vývojář s odpovědí spokojen

V případě, kdy nejsou participanti P2-BE, P3-FS a P4-BE spokojeni se získanou odpovědí, doplní asistentovi dodatečné informace, které by mohly odpověď zpřesnit a nechávají ji přegenerovat. Podle participanta P3-FS je dobré poukázat také na konkrétní nedostatky, jež programátor v odpovědích nachází. Participant P2-BE zmiňuje, že odpověď nechává přegenerovávat nanejvýš dvakrát. Pokud ani po třetím pokusu není asistent schopný správně odpovědět, hledá participant jiný způsob, jak problém vyřešit.

„Pokud nedostávám relevantní odpovědi, dopovídávám mu kontext, dál s ním komunikuju. Případně pokud vím, že něco nedělá správně nebo porušuje určité zvyklosti, tak ho s tím konfrontuji. On velice hezky dokáže vnímat moje další poznatky a aktualizovat si svoji paměť.“ P3-FS

„Většinou to člověk zkusí přegenerovat nebo popsat nějak jinak, nebo mu doplním další informace. Když to třeba ani podruhé nebo potřetí nefunguje, i když mu člověk přidá nějakou další informaci, pak už to dál nezkouším.“ P2-BE

„Opravdu se snažím pořád do kolečka mu opakovat ten kontext, protože se občas začne ztrácet.“ P4-BE

S výše uvedeným postupem souhlasí i participanti P6-BE a P7-FS. Ti si navíc chválí, že celé doplňování kontextu probíhá postupně formou dialogu, v němž mohou přesně specifikovat, jaké části odpovědi jim fungují a naopak, na které je třeba se více zaměřit.

„Když cítím, že si to fakt vymyslel, tak ho to nechám přegenerovat, upravím víc to zadání nebo ho více specifikuju. Samozřejmě tam probíhá jako dialog, což je hodně super. Takže řeknu, mám tady ještě tuhle informaci a tohle mi nefunguje.“ P6-BE

„Takovým příjemným a postupným iteračním způsobem ho usměrňuji tam, kam chci. Začnu z nějakého relativně obecného hlediska a snažím se mu popsat co nejvíc věcí, co potřebuji. Ale většinou to není jen jedna věta, snažím se tam napsat všechny své obecné potřeby.“ P7-FS

Participant P9-FS pak doporučuje kromě doplnění kontextu a upozornění asistenta na konkrétní nefunkční části kódu také na skutečnost, že asistenti lépe zohledňují zadané nedostatky a požadavky k přepracování, když jim jsou prezentovány v bodech.

„Záleží podle toho, co mi vrátí. Pokud vím, že v tom kódu něco nezohlednil, nebo tam vyloženě něco chybí, tak většinou mu na píšu v bodech, co tam není, a nechám ho to přegenerovat. Navíc mi přijde, že v těch bodech si to pak daleko snáze rozdělí a pak lépe odpoví.“ P9-FS

Chybovost a kvalita odpovědí

Participanti P2-BE a P6-BE se během rozhovorů snažili odhadnout míru chybovosti používaných asistentů. Podle nich lze odhadovat, že 10 až 25 % odpovědí, které od asistentů kódování dostali, nebyly ve výsledku uspokojivé nebo správné a bylo by lepší odpovědi na ně hledat dříve a jiným způsobem.

„U těch složitějších věcí si myslím, že určitě je třeba čtvrtina těch případů kde ve finále, když to člověk zhodnotí zpětně, tak by bývalo bylo lepší, kdyby to člověk řešil jinak nebo to třeba dřív začal hledat klasicky na internetu.“ P2-BE

„Můžu říct, že třeba 10 až 20 % jsou odpovědi, které jsou úplně blbě.“ P6-BE

3.4 Dopady asistentů kódování

Posledním tématem identifikovaným během tematické analýzy provedených rozhovorů jsou dopady asistentů kódování. Téma se věnuje primárně dopadům na vývojáře jako jednotlivce, přičemž popisuje jak pozitivní, tak negativní dopady. Některé identifikované dopady ovšem přesahují pouze úroveň jednotlivců a částečně ovlivňují také pracovní týmy, organizace a jejich kulturu. Z tohoto důvodu byly do vlastních podkapitol zahrnuty i dopady asistentů kódování ovlivňující týmy a firmy.

3.4.1 Jednotlivec

Tato podkapitola se zaměřuje na dopady používání asistentů kódování z pohledu jednotlivce, a to jak v pozitivním smyslu, kdy asistenti přinášejí usnadnění práce a nové možnosti, tak v negativním, kdy jejich využívání může přinášet určitá úskalí a rizika.

Pozitivní dopady

Mezi hlavní pozitivní dopady asistentů kódování řadí účastníci rozhovorů především změnu způsobu a urychlení vyhledávání informací, celkové urychlení vývojářské práce a zlepšení její kvality. Díky asistentům navíc probíhá konzultace řešeného problému v dřívější fázi rozpracovanosti úkolu, než kdyby se programátoři spoléhali pouze na diskusi s kolegy. Dalším významným přínosem je pozitivní vliv na psychiku a celkovou spokojenost vývojářů.

Zdroje informací, znalosti a chápání problému

Největší dopad mají asistenti kódování na zdroj a způsob vyhledávání informací, odkud vývojáři čerpají. Podle participantů P3-FS, P4-BE a P8-FE drtivou většinu dotazů, které vývojáři dříve zadávali do vyhledávačů a hledali na ně odpovědi na různých diskusních

fórech, nyní nahradilo zadávání promptů do asistentů kódování. Velkou přidanou hodnotu v tomto způsobu vyhledávání vidí zejména v rychlosti, přehlednosti práce se zdroji a jejich sumarizaci.

> *„Drasticky se změnil můj primární zdroj informací. Dovolím si tvrdit, že asistenti nahradili třeba 90 % mých zdrojů používaných při programování. Dneska sem tam kouknu do dokumentace, ale na Stack Overflow jsem třeba nebyl od doby, co jsem asistenty začal používat.“ P3-FS*

> *„Jasně, změnili [asistenti] způsob získávání informací už jen tím, že když nevíš, nejdeš na Google, ale ptáš se napřímo asistenta. Ten ti to všechno hezky sumarizuje a nemusíš procházet všechny výsledky vyhledávání.“ P4-BE*

> *„Řeknu to jednoduše. To, co bych dřív dával do Googlu, tak teďka, pokud to není nějaký security threat, dotaz okamžitě sypu do chatbota. Na základě toho, co mi řekne, tak se odvíjí moje další kroky. Musím ale říct, že téměř vždy ta odpověď vede k nějakému produktivnímu zisku.“ P8-FE*

Podle participanta P7-FS probíhá konzultace řešeného problému po zařazení asistentů kódování mezi používané nástroje v mnohem dřívějších fázích, než kdyby byl problém diskutován se živým kolegou. To je dáno především tím, že asistent kódování nemá očekávání, jak by měl dotaz vypadat nebo v jaké fázi rozpracovanosti by se měl nacházet. Participanti se tedy neostýchají asistenta použít dříve.

> *„Myslím, že ta konzultace probíhá s Copilotem daleko dřív než s někým živým. Asi je to kvůli tomu, že když jdeš za kolegou, tak on automaticky očekává, že už to [řešení úkolu] máš v určité pokročilejší fázi, něco si vymyslel. Většinou nad tím problémem už musíš koumat déle. U asistenta prostě zkusím nastřelit první návrh řešení a zeptám se, co si o tom myslí.“ P7-FS*

Po zahájení používání asistentů kódování začali programátoři vnímat získávání informací a učení se jako výrazně jednodušší disciplínu. To zejména kvůli snížení času, námahy a úsilí, které museli k získání nové informace obětovat. Díky tomu např. participant P3-FS věnuje mnohem více času učení se nových věcí a zjišťování si podrobností o fungování různých částí kódu.

> *„Už víc vím, jak některé věci fungují, a mám i větší chuť si to zjišťovat. Jdu víc do hloubky. Hlavně získávání informací je fakt levné. Před tím bych naprogramoval něco, co by fungovalo, ale už jsem neměl chuť si o tom něco dále zjišťovat. Takže se třeba stalo, že podle jména funkce jsem chápal, co kód dělá, ale trochu jsem nerozuměl syntaxi, protože to byl nový programovací jazyk. Dneska stačí jenom trošku chtít, zadat dotaz a hned máš odpověď. Já jsem hladový po informacích a on mě je naservíruje na zlatým podnose.“ P3-FS*

Snížení náročnosti při získávání nových informací zmiňuje i participant P4-BE. Ten tuto skutečnost zmiňuje zejména v kontextu začleňování nováčka do vývojového projektu. Podle jeho zkušeností pomáhá používání asistentů k rychlejšímu začlenění se do nového projektu s mnohem menším počtem otázek směřovaným ke kolegům, kteří mají na projektu více zkušeností.

„Třeba Copilot je schopen ti zanalyzovat celý kód a poradit. Tady vidím velké plus, když jsi někde nováček na projektu a úplně se v tom kódu ještě neorientuješ, tak i z téhle pozice jsi schopen se do toho kódu dostat rychleji, aniž by ses nějak hodně ptal ostatních kolegů.“ P4-BE

Jako další významný dopad vnímají vývojáři usnadnění učení se nových programovacích jazyků a možnost práce s jazykem, s nímž nemají tolik zkušeností. Poslední zmiňovaný přínos popisuje hlavně participantka P1-FE. Participant P3-FS pak oceňuje výhodu možnosti pokládat při učení dotazy vlastními slovy, aniž by musel přesně vědět, co má hledat. Podobnou výhodu shledává i participant P9-FS. Ten oceňuje primárně přínos při učení se syntaxe nového jazyka. Podle něj je využití asistentů v tomto případě mnohem rychlejší a efektivnější než klasické způsoby vyhledávání v dokumentaci.

„Mě to umožňuje psát na různých projektech i v jazyku, který tolik neumím.“ P1-FE

„Když jsem se před těmi asistenty potřeboval něco naučit nebo něco vyřešit, často jsem nevěděl, na co přesně se ptát. Ale tady ten problém vysvětlím vlastními slovy klidně na 10 pokusů, co potřebuju. On mě pochopí a dá mi odpověď. Vysvětlí mě, jak se to jmenuje, na co se ptám a řekne mi to, co potřebuju.“ P3-FS

„Poslední dobou dělám sem tam kód i v Mathlabu. Dá se tam dělat spousta věcí, ale já Matlab neznám. Určitě na vysvětlení syntaxe to neskutečně pomáhá. Mohl bych jít do dokumentace, podívat se, jestli tam bude třeba zrovna kapitola nazvaná cykly, ale než bych na to přišel, tak toto [asistenti] mi odpoví daleko rychleji a efektivněji.“ P9-FS

Rychlost práce

Participanti vnímají úsporu času při psaní kódu, opravě chyb a automatizaci rutinních úkolů jako hlavní výhodu použití asistentů kódování. Podle účastníků P4-BE, P6-BE a P9-FS spočívá úspora především v urychlení samotného psaní kódu, kdy asistenti pomáhají s jeho rychlejším vytvářením. Dopad je vnímán ze strany programátorů tím pozitivněji, čím více se jedná o rutinní kód. P5-BE a P7-FS pak ukazují i na nepřímou úsporu času, kdy jsou asistenti využíváni nejen ke generování kódu, ale i k rychlejšímu získávání a vyhledávání relevantních informací o řešeném problému.

„Je prostě schopen ti dopisovat kusy kódu, které jsou takové manuální a opakující se. Ty jenom zmáčkneš tlačítko a on to tam celé vloží. Takže je to urychlení práce v tom množství kódu, které musíš napsat ručně, případně ti poradí, když něco nevíš.“ P4-BE

„Jsou to většinou kusy kódu, co vymyslíš taky, napíšeš to taky, ale ten Copilot to udělá za zlomek sekundy, zatímco ty bys na tom pálil drahocenný minuty.“ P6-BE

„Já bych teda řekl, že časově mi to velmi pomáhá, že spousta těch věcí se s tím píše rychleji a snadněji.“ P9-FS

„…ten čas a pak ta hluboká expertiza, když něco hledáš. Ale při tom hledání informací je to zase jen o tom, že ti ušetří čas. Buď si to budeš sám zdlouhavě studovat, nebo googlit, nebo ti během chvilinky odpoví on.“ P5-BE

„Vnímám, že jsem o kus rychlejší. Nemusím řešit rutinní věci, které bych stejnak prostě hledal někde přes Stack Overflow a pak to přepisoval do vlastního názvosloví. Takhle

mi to ten Copilot pěkně vygeneruje. Dopad je určitě na rychlost a pohodlnost, že nemusím vyhledávat na internetu a proklikávat jednotlivý diskuze a tak dále.“ P7-FS

K celkovému usnadnění a zjednodušení práce přidává ještě svoji zkušenost i P2-BE, která se zakládá na využití asistentů k rychlejšímu dokončení jednoduchých úkolů pomocí jejich automatizace, díky čemuž je pak nemusí dělat programátor ručně.

„...to [asistenti kódování] člověku dost automatizuje, urychluje a pomáhá mu to s takovýma těma jako jednoduššíma dílčíma úkolama v rámci toho programování nebo obecně takový ten boilerplate kód.“ P2-BE

Úspora a usnadnění práce se pak pozitivně odrážejí nejen na spokojenosti vývojářů, ale podle participanta P6-BE také na množství nově dodaných funkcionalit.

„To se pak podepisuje na efektivitě, protože jsme schopni dodat víc funkcionalit za jednotku času.“ P6-BE

Participant P8-FE se snažil úsporu kvantifikovat na základě jemu dostupných dat. K tomu použil statistiky svého používání generované přímo asistentem kódování GitHub Copilot. Podle nich ušetřil za poslední rok 46 % kódu, který by za normálních okolností musel napsat ručně.

„Pokud bych ten dopad měl nějak kvantifikovat, tak mě jednou za rok Copilot pošle statistiky ušetřeného kódu. Za minulý rok mi to vyhodilo 46 % ušetřenýho kódu. Což osobně potvrzuju, že mi ty nástroje i pocitově zrychlily vývoj.“ P8-FE

Kvalita práce

Participanti se shodují na tom, že používání asistentů kódování má dopady i na kvalitu dodaného kódu. Podle participanta P2-BE asistenti kódování ovlivňují kvalitu softwaru tím, že konzultací nebo vygenerováním části kódu pomáhají vývojáři vyhnout se chybám, které si sám ani neuvědomuje. Bez použití asistenta by vývojář sám na tyto chyby pravděpodobně nepřišel a zahrnul by je do svého výsledného kódu. Participant P6-BE pak připouští, že existují případy, kdy mu asistent kódování poradil doplnit část kódu, na níž během návrhu zapomněl, ale ve výsledném produktu by chyběla a ovlivnila by chování celé aplikace. Kromě toho je podle něj asistent schopen v některých ohledech vygenerovat optimálnější řešení navrhovaného algoritmu.

„Po vygenerování toho kódu nebo potom, co mi [asistent] dá nějakou textovou odpověď, se mi občas stane, že přijdu na to, že bych ten kód dělal jinak a špatně. Vlastně až zpětně si uvědomím, že kdybych to dělal podle sebe, tak tam udělám nějakou chybu, na kterou bych třeba ani nepřišel.“ P2-BE

„...pak to navrhne doplnění parametru, atributu nebo něčeho, co se v tom kódu hodí, akorát jsi na to při tom návrhu softwaru ještě nemyslel nebo si mohl zapomenout, takže ti to rovnou připomene. Pak samozřejmě může generovat kód, který bude lepší/efektivnější, než bys navrhl sám.“ P6-BE

Participant P6-BE dále dodává, že v některých případech asistent kódování zvyšuje kvalitu produkovaného kódu tím, že pomůže ošetřit stavy, které v aplikaci mohou nastat, ale vývojáři se nechtějí vzhledem k jejich nízké pravděpodobnosti výskytu řešit.

> *„Ta kvalita taky dost často obsahuje třeba null-checky, na který bych mohl zapomenout, nebo by se mi nechtěly dělat. Říkal bych, je to moc psaní, tato situace stejně nenastane. Ale tím, že asistent už rovnou produkuje kód s těma null-checkama, tak už to není pro mě žádná práce ten návrh pouze akceptovat.“ P6-BE*

Participanti P5-BE a P9-FS upozorňují, že dopady na kvalitu softwaru mají asistenti kódování hlavně v programovacích jazycích, které vývojáři nepoužívají často, nebo v nich nemají takové odborné znalosti. Při použití asistentů v jazycích, kde jsou si vývojáři jisti, není zvýšení kvality produkovaného kódu tak zřejmé.

> *„Je to určitě kvalitnější. V tom, co neznám, třeba v tom Kotlinu, tak on mi s tím pomůže, aby byla ta práce kvalitnější nebo třeba optimálnější.“ P5-BE*

> *„No, co se týče chybovosti, tak určitě jsem si toho všiml. Když jsem měl projekty, kde byl jazyk, ve kterém jsem nikdy nic neprogramoval, tak jsem díky tomu nasekal daleko méně chyb, ale u jazyků, ve kterých jsem byl zkušenější, tam je to jako nastejno.“ P9-FS*

Workflow

Jak již bylo zmiňováno v předchozích kapitolách, velkým vnímaným benefitem využívání asistentů kódování je převzetí repetitivní práce. To poskytuje participantům P5-BE, P6-BE a P8-FE o mnoho více prostoru a času na přemýšlení nad možnostmi řešení úkolu.

> *„Já se vždycky těším, že [asistent] napíše ty primitivní rutiny a mě pak zbyde více času na přemýšlení a vymýšlení jiných řešení.“ P5-BE*

> *„On vyřeší nudnější práci a díky tomu mi umožní víc se věnovat věcem vyžadujícím nějaké přemýšlení nebo kreativitu. Já chci jako programátor řešit problémy a to, že se pak napíšou jako písmenka, to už je jenom cesta, jak myšlenky hodit do stroje.“ P6-BE*

> *„To, že ten den můžu díky tomu víc přemýšlet a méně času trávit opravdu nějakým researchem nebo psaním kódu jako takovýho, tak mě to vlastně víc baví.“ P8-FE*

Podle zkušeností participantů P1-FE, P2-BE, P8-FE je začlenění asistentů kódování do práce vývojáře výrazné. Popisují, že v případech, kdy mají asistenta buď vypnutého, nebo ho nemají dočasně dostupného, jsou pomalejší a zpracování úkolů jim trvá déle.

> *„Do jistý míry mi to ‚změnilo život‘. Člověk je navyklý, že když píše kód, tak mu to tam ten Copilot doplní. Když má výpadek, tak teď člověk vždycky čeká, kdy se to doplní a ono se nic neděje. V tu chvíli si připadám jako zamrzlá nebo zpomalená.“ P1-FE*

> *„Na ty funkcionality jsem si dost zvyknul. Už je [asistenty] beru jako takovou běžnou součást a samozřejmost té mé práce. Bez nich jsem jak bez ruky.“ P2-BE*

„Strašně jsem si na to zvyknul. Dám takový krátký příběh. Šel jsem na pohovor a měl jsem online coding, začal jsem psát, čekám a nic. Najednou zjistím, že tam není Copilot. Tak si říkám, dobře, to bude trvat o něco dýl. Co se týče rychlosti, je to výrazný rozdíl psát s Copilotem a bez něj." P8-FE

Začlenění asistentů kódování do práce je sice výrazné, ale participanti P4-BE a P5-BE říkají, že samotný postup, úkony prováděné během vývoje softwaru a návaznosti mezi nimi nemuseli měnit. Participant P8-FE souhlasí, že svůj postup při zpracovávání úkolu také výrazněji nezměnil. Osobně ale vnímá, že po začátku používání asistentů kódování výrazně změnil nástroje a způsob související s vyhledáváním informací.

„Neřekl bych, že se změnil můj postup. Stejně jako před tím se podívám na zadání, něco si o tom zjistím a pak začnu dělat na tom kódu." P4-BE

„Co jsem ale nezměnil, je můj postup. V tomhle ohledu se musel přizpůsobit [asistent] on mě. Prostě mám nějaké svoje flow, do kterého ten nástroj integruju a používám ho v těch svých jednotlivých fázích." P5-BE

„Určitě se mi nemění flow, jakým přistupuju k řešení problémů. Spíš bych řekl, že jsem vyměnil Google search za některého z asistentů." P8-FE

Větší změny v samotných krocích procesu vývoje softwaru sice nenastaly, ale asistenti kódování začali být vnímáni jako další instance, s níž mohou vývojáři při své práci komunikovat a řešit problémy. Ze všech rozhovorů tento přínos nejlépe popsala participantka P1-FE.

„Myslím si, že to je spíš další pomůcka k tomu, aby člověk nalezl řešení a neptal se třeba zbytečně někoho dalšího. Takže je to spíš další instance, které se ptáš předtím, než když jdeš za někým dalším." P1-FE

Psychologické dopady

Používání asistentů kódování má i pozitivní dopady na spokojenost vývojářů. Např. podle participanta P8-FE má jejich využívání dopad na jeho spokojenost. To je dáno především tím, že nemusí tolik času věnovat psaní opakujících se konstrukcí programovacího jazyka. S tímto pohledem souhlasí i participant P6-BE. Podle něho nemají asistenti kódování přímý vliv na jeho spokojenost, ale osobně cítí, že mu díky jejich používání zbývá více času na užitečnější a smysluplnější práci.

„Čím míň píšu kódu, ale zároveň dodávám výsledky, tím jsem spokojenější. Protože s postupem času, po nějakých 11 letech práce, už mě nebaví psát elementární věci. Vím, jak to udělat, rád to navrhuji, ale pak to psaní je otrava." P8-FE

„A o těch unit testech ani nemluvím. Nemusím dělat nudnou, zdlouhavou činnost, ale věnuju se tomu, co považuji za přínosnější a zrovna zajímavější." P6-BE

Participant P8-FE zmiňuje, že dopady asistentů kódování na spokojenost fungují i opačně. V případě nefunkčního asistenta kódování, na kterého je zvyklý, sám vnímá, že je méně produktivní, což mívá negativní vliv na jeho spokojenost.

„Mě tohle v podstatě i zlepšuje den. Když má třeba nějaký ten chat výpadek, tak můžu být mnohem méně produktivní a jsem z toho otráven.“ P8-FE

Velkým benefitem využívání asistentů kódování je podle participantů P7-FS a P8-FE skutečnost, že asistenty kódování mohou použít pro konzultaci v jakékoli fázi rozpracovanosti svého úkolu, aniž by měli nebo museli překonávat pocit, že ruší živého kolegu.

„Ať mám nápad v jakékoli fázi rozpracovanosti, tak si nikdy nemyslím, že bych ho [asistenta] rušil, jako to mívám třeba u živých kolegů.“ P7-FS

„Nejdůležitější je, že neotravuju živýho kolegu, nemusím rušit takový ten mentální blok. Já jsem se prostě naučil, že první je chat a pak až kolega.“ P8-FE

Participant P3-FS vnímá dodání pocitu větší svobody jako významného přínosu asistentů kódování. Tento pocit vychází z potlačení obavy týkající se nedostatečné kapacity nebo síly pro zvládnutí určitého problému, ale i z lepší příležitostí a schopností si snadněji osvojovat nové potřebné znalosti.

„Když jsem se dříve [před používáním asistentů] do něčeho pouštěl, musel jsem vypočítat svoji kapacitu, čas, sílu a pozornost v kontextu toho, do čeho se chci pustit. Teď ale nemám problém nebo obavu dělat úplně cokoliv, protože vím, že když se zaseknu, tak se doptám asistenta přesně na to, co potřebuju.“ P3-FS

„Ale vlastně ta schopnost, jak je jednoduché a levné se naučit novou věc, tak mě to dává křídla jako programátorovi. Programátor je schopnější, rychlejší, samostatnější.“ P3-FS

V neposlední řadě zaznamenal participant P9-FS díky používání asistentů kódování i změnu ve vnímání rychlosti a času věnovaného zpracování úkolu. Díky používání asistentů mu nyní některé úkoly připadají zdlouhavé, přestože by bez asistenta jejich dokončení trvalo stejně dlouho nebo dokonce ještě déle.

„Posunula se mi vlastně hranice vnímání času. Teďka nedávno pro mě bylo šokující, že některé věci jsem řešil podle mého až moc dlouho. Když jsem se nad tím ale zamyslel, tak bez toho asistenta by to ve výsledku vyšlo časově stejně nebo možná o něco dýl.“ P9-FS

Negativní dopady a obavy

Kromě pozitivních dopadů zmiňovali participanti během rozhovorů i negativní zkušenosti s používáním asistentů kódování. Tyto zkušenosti se týkají způsobů využití asistentů kódování, které již nyní ovlivňují jejich práci a jejichž dopady by se mohly v budoucnu ještě prohloubit.

Participantka P1-FE zmiňuje nebezpečí, kdy programátor přenechá vytváření kódu primárně na asistentovi kódování a nevěnuje programování plnou pozornost. V takovém případě může asistent udělat ve vytvářeném kódu chybu, která je pro programátora těžko odhalitelná a může se nepozorovaně dostat až do produkčního prostředí. S nižší pozorností podle ní souvisí i další problém, který se projevuje menším chápáním a orientací ve

vygenerovaném kódu. Snížená orientace má za následek, že ruční úpravy v kódu trvají programátorovi o dost déle, než kdyby si kód psal celý sám.

> *„Člověk má tendence nad tím kódováním tolik nepřemýšlet a nechá Copilota programovat sám. A tohle je něco, na co si člověk musí dávat bacha. Nedávno mi někdo řekl, že použil Copilota, nechal na to napsat i testy a pak zjistil až v produkci, že ten kód nedělá, co má. Vygenerovalo to něco, co si myslel, že je takhle správně, ale při bližší inspekci kódu, se našla chyba." P1-FE*

> *„Vede to k tomu, že člověk má hotovej ten kód a objeví se tam nějaká chyba. Člověk furt nechápe, kde je ta chyba. Pořád si myslí, že to ten chat naprogramoval tak, jak si ten programátor myslí. Ve skutečnosti to ale vygeneroval jinak. Než to člověk pochopí, tak si nad tím dlouho láme hlavu, než kdyby si to napsal sám." P1-FE*

S názorem participantky P1-FE částečně souhlasí i participant P6-BE, který upozorňuje na riziko nekritického spoléhání se na vygenerované odpovědi, zejména v oblastech, kde programátor nemá dostatečné znalosti k posouzení jejich správnosti. Pokud si vývojář neověří přesnost odpovědi a nekontroluje její věrohodnost, může dojít k přijetí chybného řešení se závažnými důsledky. Tento dopad je o to závažnější, čím klíčovější je zpracovávaný úkol.

> *„Kdybych o tom měl nějakou znalost, tak se rovnou u toho zasměju, že si úplně to [odpověď] vylhal. Tím, že tu znalost nemám, tak ono mi to zase tak úsměvný nepřijde. Vezmi si, že bych dělal něco fakt důležitýho a on by mi poradil nějakou takovou blbost, kterou bych přijmul, protože bych ji neměl třeba jak jinak ověřit, ale způsobilo by to nějaký problém." P6-BE*

Participant P3-FS vidí v asistentech kódování kromě již popsaného také nebezpečí spočívající v určitém mentálním úpadku v oboru vývoje softwaru, pokud by se přenechalo generování kódu čistě na asistentech. Osobně preferuje, aby se asistenti kromě řešení konkrétních problémů používali také na rozšiřování si vlastních znalostí.

> *„Je dobré využít ty nástroje k tomu, abych se kromě toho, že mi pomůžou vytvořit nějaké dílo, abych se zároveň něčemu naučil. Protože vidím, jak snadno lze dojít do stavu, kdy já jako člověk půjdu vlastně intelektem a znalostmi v tom oboru níž, než jsem. Protože přenechám to myšlení na stroji." P3-FS*

Další obava související s asistenty kódování byla odhalena participantem P8-FE. Ten vidí problém s využíváním asistentů kódování zejména u některých juniornějších kolegů, kteří sice umí výborně ovládat různé programovací jazyky nebo AI nástroje včetně asistentů kódování, ale méně rozumí generovanému obsahu a začínají pomalu ztrácet schopnost řešit problémy samostatně.

> *„Myslím si, že asistenti ještě více prohloubí problém, kdy někteří junioři sice umí perfektně nějaký jazyk nebo framework, ale neumí pořádně přemýšlet nad problémem a jeho řešením. Takovýho člověka můžeš podporovat, ale je to k ničemu, protože jakmile přijde složitější problém, nebude schopnej ho vyřešit. Takže to podle mě AI prohloubí. Lidé budou motivovaní víc se učit AI, a nebudou umět základy toho, co ta AI tvoří. Jakmile bude case, u kterého budou potřebovat něco doprogramovat, tak to nebudou schopni zvládat." P8-FE*

3.4.2 Týmy

S příchodem a větším rozšířením asistentů kódování zatím nedošlo k masivnějším změnám ve fungování týmů v souvislosti s implementací asistentů kódování. Podle participanta P8-FE se v týmech proměnily spíše používané nástroje a částečně způsob komunikace, ale rozhodně nedošlo k úpravám pracovních postupů, nebo používaných metodik pro vývoj.

> *„Zatím jsem neviděl v žádnym týmů, že by byli ti asistenti na tolik adoptovaný, aby to mělo takový dopad, že by tým nějak jako podřizoval svoje flow podle fungování asistentů. Nezažil jsem prostředí na to, aby to ovlivnilo tu společnost a její ceremonie takhle moc… Zatím je to furt na úrovni, že se to používá jako ostatní nástroje.“ P8-FE*

Participanti ovšem shodně tvrdí, že největší změna ve fungování týmu nastala v množství pokládaných otázek. Drtivá většina participantů potvrdila, že se cítí samostatnější a mají menší potřebu se ptát svých kolegů v případech, kdy narazí na problém. Potvrzují i opačnou zkušenost, kdy na jejich osobu směřuje i méně dotazů od jiných kolegů z týmu. Popsané změny nejlépe ilustrují participanti P1-FE, P6-BE a P9-FS.

> *„Když narazím na nějaký problém, tak to nejdřív řeším s tím Copilotem a nejdu hned za kolegou. Určitě to má asi ten dopad, že lidi se méně ptají a jsou víc samostatnější.“ P1-FE*

> *„Co se týče třeba troubleshootingu, tak se s kolegama trochu méně otravujeme. Odpadlo takové to, pojď se mi na něco podívat, nějak mi to nejde.“ P6-BE*

> *„Dříve to bylo, že když někdo něco nevěděl, tak se ti lidé ptali více mezi sebou. Ale osobně jsem vždycky cítil, že kolegu, kterého se ptám, mohu rušit. Takže teďka, když mám tu možnost, raději preferuju se zeptat nějakého asistenta. Zjistíš, jestli ho nenapadne nějaké řešení. Pokud si ani on neví rady, tak se zeptáš kolegy živého.“ P9-FS*

Podle participantů P6-BE a P9-FS způsobují dříve popsané změny v týmové komunikaci fakt, že se pro velkou část vývojářů stal asistent kódování běžnou součástí jejich práce a stal se první instancí, se kterou vzniklé problémy během práce řeší.

> *„Než jdeš otravovat živýho kolegu, tak jdeš otravovat chatbota, trochu s ním pochatuješ a buď z něj ta odpověď vypadne nebo ne.“ P6-BE*

> *„Lidé si víceméně zvykli na to, že mají vyloženě toho asistenta jako takovou svoji sekretářku, která je jim 24/7 dostupná a můžou se jí na cokoliv zeptat. A když si oba dva nejsou jistí, tak teprve potom přecházejí na komunikaci s celým tím týmem.“ P9-FS*

Používání asistentů kódování a snížení množství pokládaných otázek kolegům v týmu má podle participantů P4-BE a P8-FE i kladný vliv na soustředění a zátěž ostatních kolegů. Ovšem i samotní tazatelé se cítí lépe, když nemusejí své kolegy tak často vyrušovat z jejich práce.

> *„Určitě se zmenšilo množství otázek, které pokládám kolegům, protože ten asistent je schopen mi jich spoustu zodpovědět. Takže kolegy třeba nezatěžuji tolik, neberu jim tolik času otázkami, na které dokáže odpovědět asistent.“ P4-BE*

„V týmu je mnohem méně otázek. To znamená, že se tolik navzájem nerušíme, což je obrovskej benefit.“ P8-FE

Asistenti ovšem v tuto chvíli nejsou schopni podle participanta P2-BE nahradit v týmu klasickou komunikaci týkající se rozdělení, organizace nebo způsobu integrace jednotlivých vyvíjených částí softwaru mezi sebou.

„Rozhodně asistenti nenahradí potřebu konsultovat, kdo na čem třeba pracuje, jakým způsobem to integrovat mezi sebou a tak. Může být méně pokládaných otázek, jak co funguje, ale na tyhle jako provozní a organizační otázky to určitě ještě nemá.“ P2-BE

3.4.3 Firmy

Participanti P6-BE a P7-FS vnímají podporu asistentů kódování jako jedno z kritérií, které by v dnešní době zohledňovali při výběru nové pracovní pozice. Používání asistentů jim usnadňuje a zpříjemňuje práci natolik, že kdyby jim potenciální zaměstnavatel zakázal jejich používání, jeho pracovní nabídku by nepřijali.

„Co jsem tak četl, v některých firmách je používání asistentů přísně zakázáno. Já si ale říkám, že to není dobrá vizitka pro tu firmu. Kdybych šel dneska hledat novou pracovní pozici, tak první otázka na pohovoru bude, jestli při práci používají asistenty. Když mi řeknou, že nepoužívají, tak to si neplácneme, protože já tady nebudu psát kód až do zblbnutí, až se ze mě bude čoudit, tahle doba už je pryč.“ P6-BE

„Na druhou stranu, kdyby společnost zakazovala vývojářům používat umělou inteligenci, tak si myslím, že u ní lidi nebudou chtít pracovat a nevím, jestli teď, ale za pár let určitě bude trošku taková firma ztrácet na kapacitách. Nejen personálních, ale i časových. Pro mě je používání asistentů také docela relevantní věc, podle které bych si vybíral zaměstnavatele.“ P7-FS

K předchozímu tvrzení dodává participant P6-BE ještě poznatek, že odmítání asistentů ze strany zaměstnavatele může mít dopad nejen na nedostatek pracovníků, ale také na snížení produktivity vývojářů a tím pádem i konkurenceschopnosti na trhu. S podobným tvrzením přichází i participant P5-BE. Pro něho sice nejsou asistenti kódování důležitým kritériem při výběru zaměstnání, ale také v nich vidí technologický posun a zvýšení produktivity.

„Myslím si, že když [firmy] ty asistenty neadoptují, sníží se jejich konkurenceschopnost, protože nebudou moc dodávat za takovou cenu. Ti vývojáři toho s asistenty stihnou mnohem více. Nejen že to šetří čas, ale já tomu říkám jednotky pozornosti, kterých prostě nemáš neomezeně za ten den. Takže když budeš dělat nějakou práci do zblbnutí, která ti zabere sice ‚jen‘ půlku dne, ale protože došly jednotky pozornosti, tak budeš zbytek dne nepoužitelný.“ P6-BE

„Podle mě to dneska pomůže úplně všem. Kdo to odmítá, tak tohle je další technologický posun nebo zlom v tom vývoji. Musíš se na to adaptovat, jinak si skončil. Je to stejné, jako kdybychom dneska psali kód v poznámkovém bloku. Napsali bychom ho, ale ta produktivita by se rapidně snížila a zaměstnavatelé by koukali.“ P5-BE

V podmínkách České republiky nebývá z pohledu zaměstnavatelů používání asistentů kódování při práci vyžadováno. Podle zkušeností participanta P9-FS se ovšem v zahraničí

začínají objevovat společnosti, které asistenty kódování a využívání AI nejen podporují, ale jejich používání od vývojářů přímo vyžadují.

> *„Co vím třeba od ostatních developerů, kteří žijí v zahraničí, tak je zajímavé, že jim vyloženě šéfové říkají, že pokud nebudou používat AI, tak si najmou někoho jiného. Takže tam je to vyloženě podmínka. Ale že by to tady v Česku bylo až tak vyhrocené, tak s tím jsem se nesetkal.“ P9-FS*

3.5 Četnost výskytu ústředních témat

Cílem podkapitoly je ukázat, jak často byla zastoupena jednotlivá ústřední témata v analyzovaných rozhovorech. Toto zastoupení ukazuje tabulka 3. Četnost výskytu tématu je reprezentována počtem kódů identifikovaných během provedené tematické analýzy. Z přehledu vyplývá, že nejčastěji zastoupeným ústředním tématem je T03, následované tématy T04 a T02. Naopak nejméně frekventovaným tématem v analyzovaných rozhovorech bylo T01.

Tabulka 3: Četnost výskytu ústředních témat, zdroj: autor

Ústřední téma a podtémata	Četnost	Příklad citace
T01 – Přijetí asistentů kódování	**29**	-
1. Důvody prvního použití	9	„Ze začátku to určitě byla zvědavost. Já jsem prostě viděl, že to bylo něco hodně nového. Hlavně oproti již existujícím statickým doplňovačům mi tohle přišlo jako hodně velký skok, tak si to člověk chtěl vyzkoušet.“
2. Očekávané přínosy	20	„Na začátku od toho člověk neměl žádné očekávání, protože přišla nová featura, která dokáže doplňovat kód a člověk byl nadšenej, že to doplní jednoduchý for cyklus. Ale teď už ty očekávání mám logicky větší. Zvykla jsem si hlavně na komfort toho, že mám toho Copilota.“
T02 – Současný postoj k asistentům kódování	**111**	-
1. Využívaní asistenti	28	„Nejvíc používám Chat GPT, zkoušel jsem Gemini a Copilot z těch standardních. Pak jsem také zkoušel i některé jako interní verze, co nám poskytla firma.“
2. Četnost interakce s asistenty kódování	14	„Při doplňování kódu v IDE už ani kolikrát nevím, že ho používám. To je reflex. Už to beru jako takovou prodlouženou ruku.“
3. Očekávané přínosy	31	„Už se spoléhám hodně často na to, že když dělám právě nějaké rutinní věci, on mi s tím pomůže.“
4. Asistenti vs. živí kolegové	7	„„…když se ptám na něco chatbota, tak se ptám na konkrétní dotaz. Když se ptám kolegy, tak se sice ptám na konkrétní dotaz, ale on rovnou na to může navázat na něco z úplně jiné oblasti a zároveň i to mě může posunout dál. Teď to [AI asistenti] neumí… Navíc kolega zvládá naprosto bravurně diskutovat i nad delším a rozsáhlejším kódem.“
5. Expertíza asistentů	13	„Když mu dáš řešit komplexní úkol, tak se dostane úplně na úroveň juniora. Když mu dáš něco, co napíšu kódem na 10, 20, 30 řádků, tak to bude určitě hodně seniorní, ale když mu dám udělat projekt o 100 souborech, jako 100 třídách, nedovedu si představit, že by to zvládnul.“

6.	Omezení asistentů	18	„Máme tady GitHub Copilota, který je na GitHubu, kde máme repozitář a on tam zná úplně všechno. Pak tady ale máme hromadu dalších informací, které nemáme v žádném repozitáři. Jen namátkou: jak spolu integrují aplikace, jak jsou nastavené servery, jak probíhá build, jaké je flow nasazení. Všechny tyhle informace jdou úplně mimo jakýkoli repozitář a často je nejde ani do nějakého zanést, je to v nějakém settingu aplikace. A on do ní nemá přístup, nemůže to vědět.“
T03 – Práce s asistenty kódování		**208**	-
1.	Faktory ovlivňující kvalitu odpovědi	77	„Čím je to [úkol] komplexnější, tak tam přestává fungovat. Teda alespoň já ho přestávám využívat, protože mi přijde, že bych ho musel moc opravovat a ten strávený čas by byl neúměrný výsledku.“
2.	Fungující způsoby užití asistentů kódování	68	„Za sebe ho nejraději používám na psaní testů. To je zdlouhavá, úmorná činnost. Hlavně když pak testuješ software korporátní, tak tam to jsou desítky až stovky řádků kódu na test…“
3.	Případy, kdy vývojáři asistenty kódování nepoužívají	23	„Neefektivní časově je to hlavně u úprav něčeho, kde vím, co je chyba nebo kde ji hledat. V takovém případě mi přijde rychlejší jen přepsat si ručně tu věc, kde vidím, že je tam problém, než vzít CTRL + C, CTRL + V a pak kontrolovat, jestli náhodou se náhodou nezrušilo tou vygenerovanou odpovědí něco jiného v tom původním kódu.“
4.	Další práce s vygenerovanou odpovědí	40	„Ten kód pročtu a snažím se nad tím zamyslet, jestli nemá nějaký problém, že by to prostě vedlo k nějaký chybě, jestli je to efektivní řešení. Prostě úplně stejně, jako kdybych v tu chvíli dělal takový code review nad tím, co mi to vlastně doporučilo.“
T04 – Dopady asistentů kódování		**142**	-
1.	Jednotlivec	83	„Drasticky se změnil můj primární zdroj informací. Dovolím si tvrdit, že asistenti nahradili třeba 90 % mých zdrojů používaných při programování. Dneska sem tam kouknu do dokumentace, ale na Stack Overflow jsem třeba nebyl od doby, co jsem asistenty začal používat.“
2.	Tým	18	„Co se týče třeba troubleshootingu, tak se s kolegama trochu méně otravujeme. Odpadlo takové to, pojď se mi na něco podívat, nějak mi to nejde.“
3.	Firmy	41	„Myslím si, že když [firmy] ty asistenty neadoptují, sníží se jejich konkurenceschopnost, protože nebudou moc dodávat za takovou cenu. Ti vývojáři toho s asistenty stihnou mnohem více. Nejen že to šetří čas, ale já tomu říkám jednotky pozornosti, kterých prostě nemáš neomezeně za ten den. Takže když budeš dělat nějakou práci do zblbnutí, která ti zabere sice ‚jen‘ půlku dne, ale protože došly jednotky pozornosti, tak budeš zbytek dne nepoužitelný.“

4 Diskuse výsledků

Kapitola se věnuje diskusi výsledků získaných během provedeného výzkumu. Autor v ní prezentuje dosažené výsledky, navzájem je propojuje do uceleného pohledu a zasazuje je do kontextu trendů ve vývoji informačních systémů. Současně se snaží výsledky a zjištění propojit nebo naznačit možnou existenci vazby s již existující literaturou. Pro zachování přehlednosti a systematičnosti je kapitola strukturována dle dříve formulovaných výzkumných otázek. Každé otázce je věnována samostatná podkapitola, v níž jsou příslušná zjištění podrobně diskutována.

4.1 RQ1: Jak vývojáři v České republice vnímají AI asistenty kódování a jaká jsou jejich hlavní očekávání a očekávané přínosy při používání AI asistentů kódování?

Z provedených rozhovorů vyplývá, že nejčastěji používaným AI asistentem kódování je Chat GPT. S tímto asistentem měli zkušenosti všichni participanti rozhovorů. Hned za ním se umístil GitHub Copilot, který nevyzkoušel pouze jeden účastník. Časté využívání zmíněných asistentů potvrzuje i společnost Stack Exchange Inc. (2024) ve svém průzkumu používání AI asistentů kódování vývojáři. S určitým odstupem následovali firemní chatboti. Jedná se o skupinu asistentů, jež byli vytvořeni společnostmi, pro které participanti pracovali. Jako hlavní důvody implementace a využívání firemních chatbotů zmiňovali participanti nejčastěji zvýšení bezpečnosti vkládaných dat. Po firemních chatbotech následovali AI asistenti kódování od společností JetBrains, Microsoft a dalších.

Vnímání a očekávání programátorů od AI asistentů kódování se s postupem času mění. Zpočátku byl postoj k asistentům ve většině případů velmi rezervovaný. Programátoři byli při používání nových nástrojů velmi obezřetní, moc jim nedůvěřovali a v první fázi od nich neměli žádná očekávání. Většinou je začali používat ze zvědavosti nebo až na doporučení známých, kolegů, či po výzvě zaměstnavatelů. Ze vzorku participantů a jejich odpovědí také vyplývá, že juniornější vývojáři jsou spíše ochotnější asistenty používat, mají s nimi větší zkušenost a využívají je i po delší dobu. Vyšší míru adopce u juniorů zaznamenal i Dylan Walsh (2024).

Po nabytí zkušeností s používáním AI asistentů kódování se pohled programátorů na asistenty změnil. Dnes účastníci asistenty kódování vnímají jako užitečné pomocníky, kteří dokáží pomoci s různými aspekty během programování. Vnímání asistentů kódování v České republice je velmi podobné jejich vnímání v zahraničí (Pan et al., 2024). Sice při jejich používání stále přetrvává jistá *dávka obezřetnosti*, přesto účastníci rozhovorů shodně potvrzují, že se stali asistenti kódování *běžnou součástí jejich práce*. Dnes vývojáři od jejich užití očekávají především zvýšení komfortu při práci a zejména pomoc při řešení rutinních a méně zajímavých úkolů. Tento způsob pomoci se pak projevuje snížením množství času, který musí vývojář rutinnímu úkolu věnovat. Tyto přínosy zmiňují nejen participanti

provedených rozhovorů v této práci, ale také programátoři ze zahraničí (Pan et al., 2024). Kromě *ušetření času a úsilí věnovanému rutinním úkolům* očekávají vývojáři od asistentů kódování i poskytnutí pomoci při vyhledávání informací a podkladů ke zpracovávanému úkolu, nebo pomoc při řešení problémů, s jejichž řešením si nejsou jistí, nebo tam, kde je třeba použít konstrukci programovacího jazyka, se kterou se nesetkávají tak často. V neposlední řadě očekávají, že jim asistent kódování může buď odhalit chybu v kódu, nebo navrhnout vhodnější řešení problému, které by je jinak nenapadlo.

Očekávání vývojářů od asistentů kódování *se liší i podle toho, jakou funkcionalitu asistentů se vývojáři rozhodnou používat*. U funkce chatu jsou očekávané spíše rady a návrhy v přirozeném jazyce. Takové odpovědi jsou nejčastěji získávané formou konzultací konkrétního problému, nebo formou dialogu mezi vývojářem a asistentem. Oproti tomu při použití funkce našeptávání kódu chtějí programátoři dostat kus kódu řešící zadaný problém a splňující jimi definovaná pravidla. Důležité také je, aby vygenerovaný kód respektoval používaný programátorský styl na projektu.

Do očekávání programátorů se promítají také *omezení*, která jsou z jejich strany v současné době vnímána. Jedná se především o *omezenou možnost integrovat AI asistenty kódování* s ostatními vývojářskými nástroji, ve kterých jsou uchovávané požadavky, jejich zadání nebo znalostní báze. Dále se jedná o menší *schopnost asistentů zpracovávat komplexnější nebo složitější úkoly* bez přímého zásahu a instruování ze strany programátora. Někteří participanti si dokonce myslím, že použití asistenta kódování pro úspěšné zpracování zadaného úkolu je v určitých případech bez jejich zásahu zatím nemožné. Velmi podobné omezení zmiňuje také studie zkoumající interakci programátorů s AI asistenty. Z ní vyplývá, že nejvýznamnějšími obavami programátorů jsou právě nepřesnosti v odpovědi nebo nedostatek kontextu pro její vygenerování (Akhoroz & Yildirim, 2025).

Účastníci rozhovorů této diplomové práce také zmiňovali jako *nevýhodu obecnost generovaných odpovědí* v případech, kdy si asistent neví rady, jak odpovědět. Problém vnímají vývojáři i v kreativitě odpovědí asistentů. Lze vidět, že asistenti generují odpovědi podle materiálů, z nichž se učili, a bývá u nich problém, když vývojáři požadují vyšší míru kreativity.

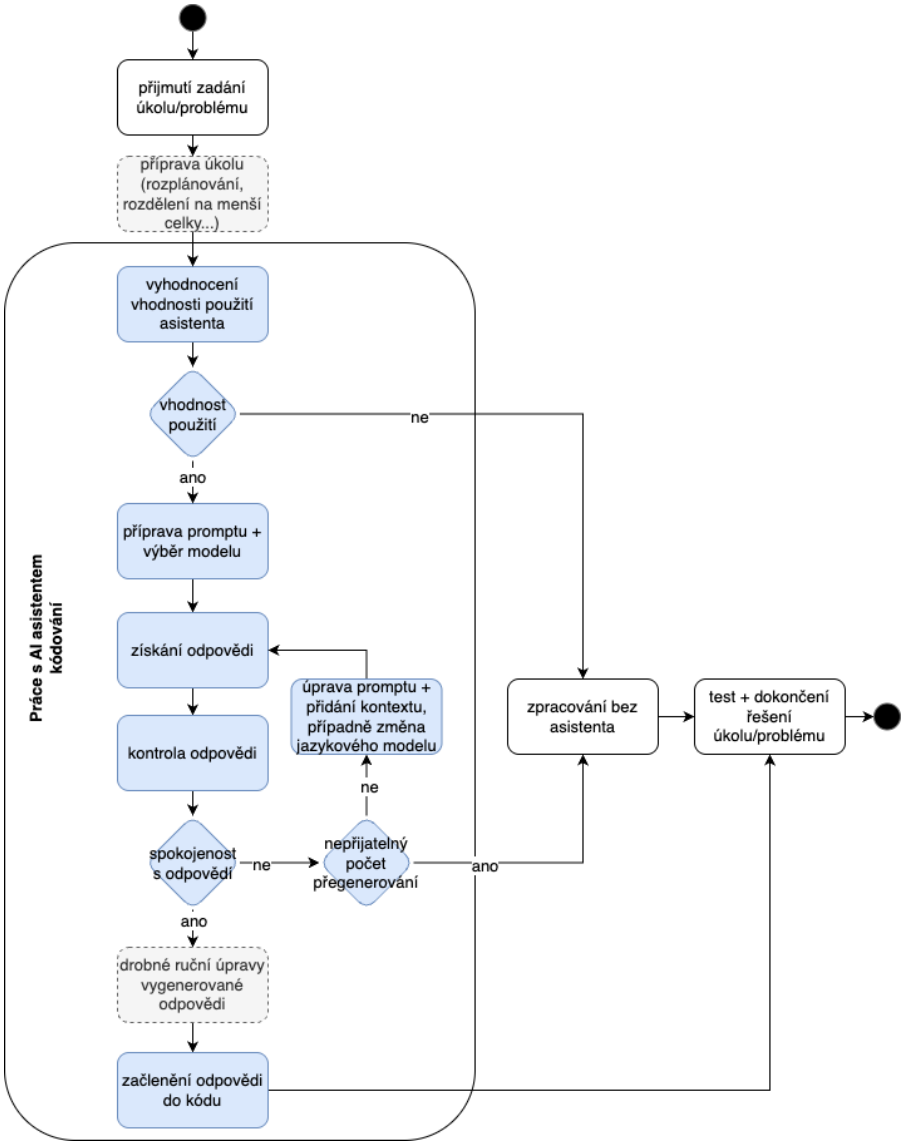
S tím se pojí i vnímání rozdílu mezi AI asistenty kódování a živými kolegy. Podle účastníků má konzultace řešeného problému se živým kolegou výhodu v tom, že kolega má znalost historicky i aktuálně řešených problémů a zavedených postupů v týmu. Díky tomu je schopen při diskusi nabídnout pestřejší a vhodnější řešení, aniž by se mu musela explicitně zmiňovat některá fakta, což může vést k získání naprosto unikátního řešení problému.

Během rozhovorů hodnotili účastníci mimo jiné také *expertízu asistentů*. Z jejich odpovědí vyplývá, že je třeba při hodnocení zohlednit zejména komplexitu řešeného úkolu. Jak již bylo v této kapitole zmíněno, schopnost řešit komplexnější úkoly je jedním z hlavních omezení dnešních asistentů. Tento fakt se projevuje i v samotných hodnoceních. *Čím komplexnější je řešený problém, tím klesá úroveň expertizy*. Naopak ve schopnosti jít u méně komplexního tématu více do detailu se asistenti kódování umí vyrovnat seniorním vývojářům.

Ve světě bývá ze strany vývojářů často zmiňovaná nejistota ohledně *bezpečnosti asistentů* a nakládání se zaslanými daty (Arkheden & Eklund, 2024). V provedených rozhovorech ovšem bylo toto téma zmiňováno spíše okrajově. Vývojáři otázku bezpečnosti nevnímají ze svého současného pohledu jako klíčovou.

4.2 RQ2: Jak vývojáři v České republice při své práci využívají AI asistenty kódování?

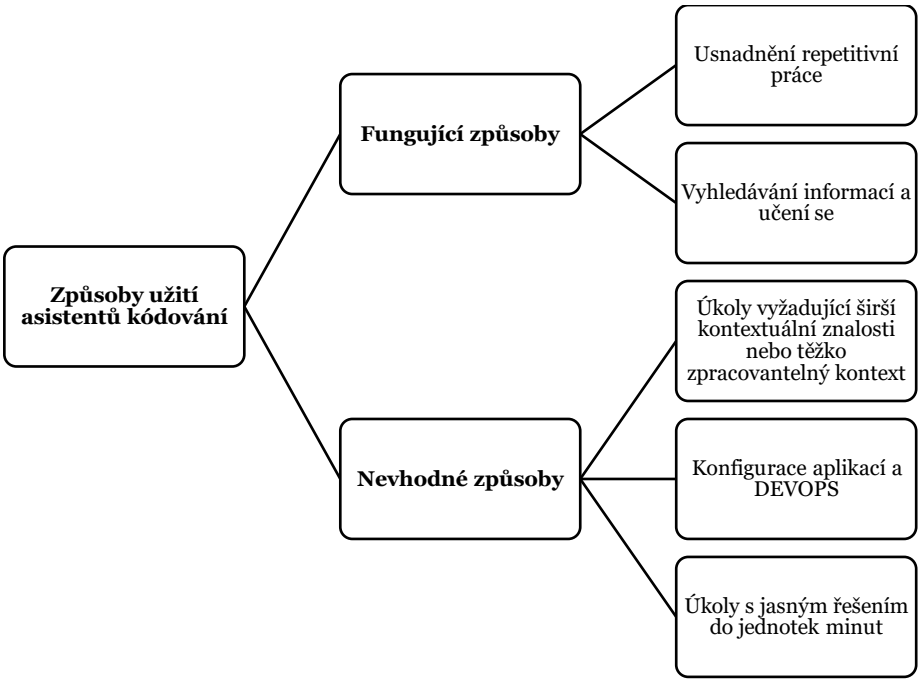
Používání asistentů kódování se dnes stalo běžnou součástí práce vývojářů a je při tvorbě kódu stejně přirozené jako práce s vývojovými prostředími. O něco více spoléhají programátoři na funkci našeptávání kódu. Použití funkce chatu je více závislé na zkušenostech a znalostech vývojáře. Čím více se v řešené problematice orientuje, tím je méně pravděpodobné, že bude chtít asistenty kódování pro zpracování konkrétního požadavku použít. Samotný průběh použití z pohledu vývojářů zachycuje obrázek 4.



Obrázek 4: Workflow používání AI asistentů kódování, zdroj: autor

Obecně lze říci, že při používání funkce chatu jsou programátoři ochotnější s asistentem více interagovat a v případě špatné nebo neuspokojivé odpovědi mu doplňovat kontext řešeného problému a doptávat se ho na podrobnosti. Takové chování ovšem nelze pozorovat při našeptávání kódu, kde participanti dávají asistentům kódování pouze jeden pokus na vygenerování správné odpovědi. To může do jisté míry souviset s tím, že funkci chatu vývojáři využívají častěji při řešení problémů, které jsou pro ně nové. Proto mohou mít tendenci se asistentů více doptávat, nebo pro ně může být pohodlnější a příjemnější zkusit nechat odpověď víckrát přegenerovat, než ručně hledat podklady k řešenému problému na internetu.

Obrázek 5 pak zachycuje základní znázornění fungujících a nefungujících způsobů užití asistentů kódování. Podrobnějšímu popisu i s konkrétními příklady fungujících způsobů využití se věnuje podkapitola 4.2.1. Popisu nevhodných způsobů je pak věnována kapitola 4.2.2..



Obrázek 5: Znázornění fungujících a nefungujících způsobů užití asistentů kódování, zdroj: autor

4.2.1 Fungující způsoby užití asistentů kódování

Identifikované fungující způsoby užití AI asistentů kódování, které zmiňovali účastníci rozhovorů, lze rozdělit do dvou základních kategorií.

První z nich tvoří *úkony* a *činnosti usnadňující repetitivní práci*, kam se řadí psaní rutinního kódu, tvorba unit testů a CRUD operací, pomoc při přepisu kódu z jednoho programovacího jazyka do druhého, generování dokumentace kódu nebo generování jednoduchých kusů kódu a komponent.

Do druhé kategorie spadají především *činnosti týkající se vyhledávání potřebných informací, učení se nových znalostí a konzultace vzniklých problémů nebo vysvětlování kódu*. Zde AI asistenti kódování u mnohých účastníků nahradili klasické vyhledávání

informací pomocí internetových vyhledávačů. To potvrzuje i Vaithilingam et al. (2023). Kromě toho jsou asistenti používáni pro *získávání rychlých nápadů* řešení konkrétního problému. V takovém případě participanti často nepožadují vygenerování konkrétního kódu, ale stačí jim získat nápad na řešení, který si následně implementují podle sebe. Ostatní zdroje kromě již popsaných fungujících způsobů dodávají další způsoby užití. Mezi ty nejpoužívanější patří refactoring kódu, ale také identifikace a klasifikace závažnosti jednotlivých defektů (Sergeyuk et al., 2025).

Podle participantů AI asistenti kódování nabízí *nový způsob, jak rychle získat odpovědi na otázky vzniklé během práce a jak diskutovat možná řešení*. Před jejich používáním bylo nutné, aby si řešený problém a otázky s ním související dopředu připravili a konzultovali je se živým kolegou. Taková konzultace probíhala zpravidla až v pozdějších fázích implementace. K tomu docházelo především z toho důvodu, aby si kolega tazatele nemyslel, že za ním tazatel přichází s nehotovým řešením. Při použití asistentů ovšem tento problém odpadá a účastníci rozhovorů používají asistenty velmi často i pro konzultaci problémů v rané fázi implementace.

4.2.2 Nevhodné způsoby užití asistentů kódování

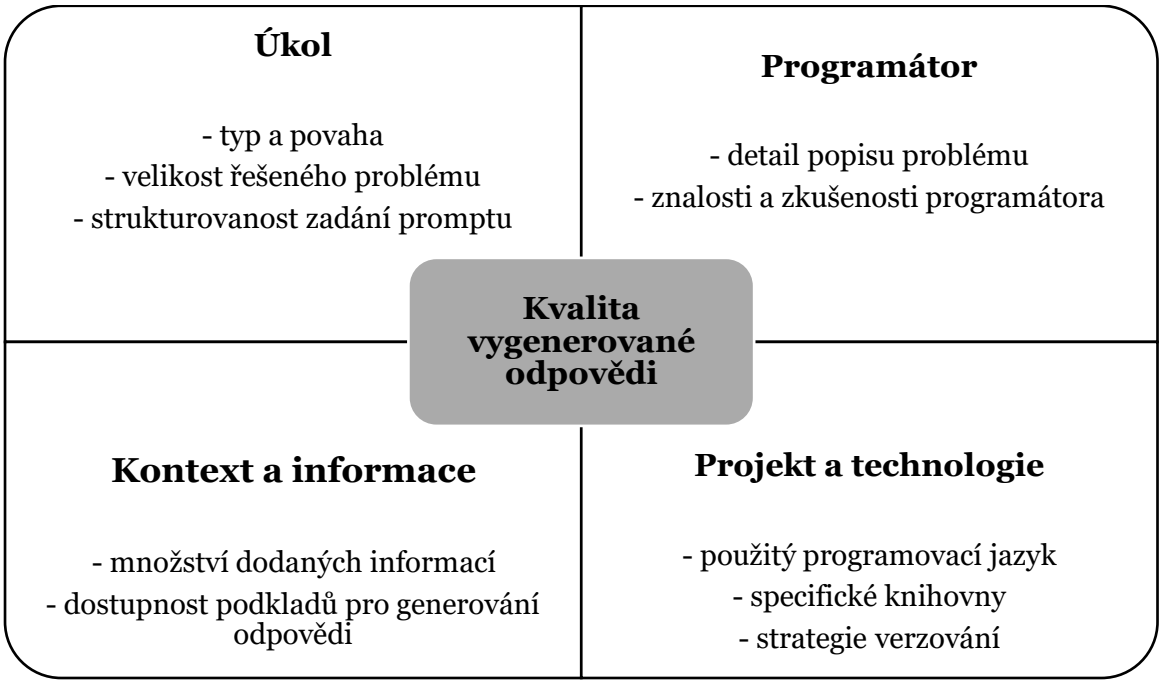
Na druhou stranu existují i typy úkolů, které nemá smysl s asistenty kódování řešit. Jedná se především o *úkoly vyžadující širší kontextuální znalosti*, které se těžko zachycují a přenášejí, nebo jsou uložené na místech a v podobě, jež je pro asistenty těžko zpracovatelná. Typickými příklady mohou být *architektonické návrhy a rozhodnutí, konfigurace aplikace* a úkoly týkající se *DevOps*[26]. Existují i situace, kdy si participanti řešený úkol udělají sami, protože dokončení požadavku vlastní silou je rychlejší, než kdyby požadavek nechávali zpracovávat asistenta kódování. Popsaná situace nastává zejména u *kratších a jednodušších úkolů*. Podle Sergeyuka et al. (2025) ovšem ovlivňuje používání asistentů také nezájem o nové nástroje, nepřesnost jejich odpovědí a nedostatek důvěry v kvalitu vygenerované odpovědi.

Určité odlišnosti ve způsobech použití asistentů kódování lze pozorovat i mezi juniornějšími a seniornějšími vývojáři. *Junioři* mají tendenci *používat asistenty* kódování *častěji* a používají ho nejen k usnadnění repetitivní práce, ale často také k tomu, aby jim asistent pomohl a poradil s řešením konkrétního problému, anebo s odstraněním chyby. Oproti tomu *seniornější vývojáři* asistenty kódování používají *především k usnadnění repetitivní práce*, získání nápadu na řešení problému, které si pak sami implementují, nebo ho používají k osvěžení a rozšíření si svých znalostí.

[26] Přístup k vývoji softwaru, který propojuje vývojářské (Dev) a provozní (Ops) týmy s cílem zrychlit a zefektivnit nasazování softwaru pomocí automatizace, spolupráce a kontinuální integrace a doručování (Kim et al., 2021).

4.2.3 Faktory ovlivňující kvalitu odpovědi

Rozhodování, zdali asistenta kódování programátor použije pro zpracování daného úkolu ovlivňuje zejména výsledná kvalita vygenerované odpovědi, kterou ovlivňuje několik faktorů. Mezi ně patří *povaha úkolu*, *chování programátora, dostupný kontext, ale i projekt a zvolené technologie*. Jednotlivé faktory vizualizuje obrázek 6.



Obrázek 6: Faktory ovlivňující kvalitu vygenerované odpovědi, zdroj: autor

Pokud je to možné, u úkolu preferují vývojáři jeho rozdělení na *menší celky*, které pak s asistentem řeší samostatně. Toto chování koresponduje i se zjištěními ostatních prací, které se zabývaly podobnou tématikou v zahraničí (Vaithilingam et al., 2022). Pokud by vývojáři úkol nerozdělili, pak by asistentům kódování museli poskytnou neúměrně detailní popis, jehož vytvoření by ve výsledku zabralo více času, než si problém vyřešit sám. Mimo to je důležitý i typ a povaha řešeného problému včetně samotné struktury zadávaného promptu. Ten by měl v ideálním případě vždy obsahovat cíl úkolu, jeho omezení a příklady.

Kvalitu vygenerované odpovědi ovlivňuje i samotná *ochota* a vůle vývojářů detailně *popisovat řešený problém* a všechna jeho omezení, požadavky a pravidla, která musí asistent kódování zohlednit. Často se stává, že programátoři uvedou pouze menší množství informací, než které mají dostupné, a doufají, že to bude pro správné vygenerování odpovědi stačit. Dostatečné množství potřebného kontextu je však klíčem k úspěchu i podle Mendese et al. (2024). Podle něho nemá zkracování kontextu rozhodně význam a bývá častou příčinou nesprávných odpovědí.

Pro kvalitní odpověď je klíčový i samotný *kontext úkolu*. Ten spočívá nejen v uvedení dostatečného množství informací o požadovaném výstupu, ale také v dostupnosti podkladů, ze kterých může asistent kódování při generování odpovědi čerpat. Obecně platí pravidlo, že čím více je na internetu dostupných informací o řešeném problému, tím je vyšší

pravděpodobnost, že bude vygenerovaná odpověď kvalitnější. Pan et al. (2024) k tomu navíc doporučuje, že je vždy dobré provést více iterací dotazu nad řešenou problematikou.

Projekt a využívané technologie se také podílejí na výsledné kvalitě. Podle participantů jsou přínosy v použití asistentů kódování rozdílné nejen mezi frontendem a backendem, ale také mezi jednotlivými jazyky používanými pro stejný účel. Dále do kvality generované odpovědi výrazněji vstupuje použití vlastních firemních knihoven, jejichž chování a fungování není nikde specifikováno a asistent si ho nemůže nijak ověřit. Kromě toho může docházet k podobným problémům i při používání starších verzí knihoven (Mendes et al., 2024). Kvalitu odpovědi definuje také využitá strategie verzování. Z rozhovorů vyplývá, že např. při použití AI asistentů kódování ve větších repositářích nejsou asistenti schopni vzhledem k omezením, které se týkají výpočetního výkonu pracovní stanice programátora, odpověď vygenerovat.

4.2.4 Práce s vygenerovanou odpovědí

Dalším důležitým krokem při práci s asistenty kódování je posouzení správnosti vygenerované odpovědi. Důslednost a podrobnost její kontroly ze strany vývojářů *závisí* podle participantů na *kritičnosti zpracovávaného úkolu*. Mezi méně kritické úkoly se řadí například stylování a tvorba frontendových komponent nebo menší změny na backendu s malým dopadem na fungování aplikace. U takových úkolů vývojáři vygenerovanou odpověď zkontrolují pohledem a jdou ji ihned použít v kódu aplikace. K podobné situaci dochází i u úkolů, které mohou být sice pro chod aplikace kritičtější, ale vzhledem k množství generovaného kódu (zpravidla do tří řádků) je snadné, aby vývojář zkontroloval jeho správné fungování na první pohled.

Pokud je ovšem množství generovaného kódu vyšší, nebo je programátorem řešený úkol vyhodnocen jako kritický, pak před samotným použitím vygenerované odpovědi provádějí programátoři její detailnější zkoumání. Kritičnost změny participanti nejčastěji hodnotí podle dopadu na data uložená v databázi. Zaměřují se ovšem také na to, zdali změna výrazněji nezvyšuje riziko pádu aplikace nebo ztráty dat, případně zdali se změna netýká autorizace uživatelů. V takových případech pak programátoři před samotným zařazením nového kódu provádějí detailnější zkoumání. Často se jedná o velmi podobný princip, jako když dělají code review svým kolegům.

Když najdou vývojáři nejasnost, nebo chybu ve vygenerované odpovědi, pak se snaží asistentovi kódování doplnit další relevantní informace a kontext řešeného problému. Následně nechávají odpověď přegenerovat. Podle účastníků rozhovorů nechávají původní odpověď přegenerovat zpravidla dvakrát. Pokud ani po druhém přegenerování nedostanou uspokojivou odpověď, problém řeší jiným způsobem. Při doplňování kontextu a upozorňování na chyby ve vygenerované odpovědi se osvědčilo dávat zpětnou vazbu v bodech, kde každý z nich reflektuje jeden nedostatek v odpovědi. Díky tomu jsou asistenti kódování schopni jednotlivé připomínky lépe zohlednit.

4.3 RQ3: Jaký dopad má využití AI asistentů kódování na práci a výkon vývojářů v České republice během vývoje softwaru?

V rámci provedených interview bylo zjištěno, že začlenění AI asistentů kódování samotný proces vývoje a jeho kroky výrazněji nezměnilo. Jak již bylo naznačeno v předešlých kapitolách zodpovídajících ostatní rešeršní otázky, proměnily se především používané nástroje. K velké změně došlo v oblasti *vyhledávání informací*, kde asistenti kódování *nahradili* v mnoha případech klasické *internetové vyhledávače*. Díky tomu se vyhledávání informací stalo rychlejší a jednodušší. Ke stejným změnám dochází nejen v ČR. Vaithilingam et al. (2022) dokonce zmiňuje, že někteří programátoři již vůbec nepoužívají klasické internetové vyhledávače. Účastníci rozhovorů si také pochvalují, že kromě urychlení jim asistenti vyhledávání i zpříjemňují, jelikož jim z relevantních zdrojů vyberou a shrnou pouze to nejpodstatnější a vývojáři si pak nemusí procházet tolik zdrojů. Arkheden a Eklund (2024) ještě zjistili, že se vývojářům zdá odpověď od asistentů jasnější.

S usnadněním získávání informací se pojí i další dopad, který souvisí s chutí programátorů *vyhledávat si dodatečné informace o řešeném problému* nad rámec toho, co by pro jeho zpracování potřebovali. Vývojáři jsou díky ušetřenému času a úsilí při zpracovávání úkolu ochotnější doplňovat si další znalosti tam, kde určité části zpracovávaného problému tolik nerozumějí. To se ve výsledku pozitivně projevuje na jejich schopnostech, ale i celkovém chápání a řešení problémů.

Pozitivní dopady vnímají účastníci také v případě, kdy se dostanou na nový projekt. Zde jim asistenti pomáhají *rychleji pochopit kód projektu* a jsou díky tomu schopni mnohem dříve a rychleji do něj začít přispívat, aniž by se museli často doptávat kolegů. Podobné výhody vnímají vývojáři i u programovacích jazyků, ve kterých nemají takové zkušenosti. Stejný přínos ovšem neplatí při učení se úplných základů a principů programování. Zde programátoři přínos ve využívání asistentů nevnímají, jelikož je podle nich nutné základní principy pochopit a nespoléhat se pouze na asistenty kódování. Haque et al. (2024) tuto zkušenost jen potvrzuje a zdůrazňuje důležitost fundamentálních znalostí a principů v dané oblasti. Na druhou stranu také zmiňuje, že kvalita a přesnost odpovědí se odvíjejí od znalostí programátorů. Toto tvrzení ovšem výsledky diplomové práce nemohou úplně potvrdit. Podle této diplomové práce zkušenosti vývojářů ovlivňují spíše ochotu asistenty kódování používat, než že by ovlivňovaly kvalitu jejich odpovědí.

Při zodpovídání RQ2 bylo zmíněno, že díky asistentům kódování probíhá konzultace problému v dřívější fázi vývoje. To má dopad nejen na programátora, který nemusí překonávat pocit, že svým dotazem někoho vyrušuje, ale i na kvalitu kódu. Používáním asistentů kódování dochází také *k odhalení některých chyb v mnohem dřívější fázi vývoje*. To často dramaticky snižují náklady na jejich odstranění. V určitých případech se používáním asistentů může vzniku některých chyb dokonce úplně předejít. Nejčastěji bývají odhaleny chyby vzniklé z nepozornosti, opomenutím nebo neošetřením stavu aplikace, který by podle vývojáře neměl nastat.

Největším benefitem používání asistentů kódování je *ušetření času a zvýšení spokojenosti* programátora při práci. Vyšší spokojenost je způsobená jednak tím, že jsou vývojáři schopni

za stejné množství času doručit více funkcionalit, ale i tím, že musí dělat mnohem méně repetitivní činnosti a zbývá jim díky tomu více času na zajímavější a smysluplnější práci. Tento dopad popisuje ve své práci také Bird et al. (2023). Popisovaný pocit bývá o to silnější, čím je vývojář seniornější.

Implementací asistentů kódování došlo podle Birda et al. (2023) a některých účastníků rozhovorů této diplomové práce k *odbourání obav týkajících se nedostatku kapacit*, nebo schopností zpracovávat určité úkoly. Nyní se vývojáři neobávají s AI asistenty kódování kdykoli pustit do jakéhokoli úkolu, i když by se za normálních okolností na jeho zpracování například kvůli únavě necítili. Z rozhovorů také vyplývá, že u některých participantů došlo ke *změně vnímání času stráveného při řešení úkolů*. I když na úkolu stráví programátor stejné, anebo menší množství času, jako by věnoval podobným úkolům před používáním asistentů, strávený čas se mu zdá mnohem delší.

Vývojáři ovšem vnímají i negativní dopady, které AI asistenti kódování přináší. Na jednu stranu skutečně *ulehčují práci a usnadňují vstup do odvětví*, na druhou stranu toto vnímají programátoři částečně jako negativní dopad. Podle nich v některých případech dochází při používání asistentů k *postupnému snižování znalostí programátorů*. Jev je silnější, čím jsou vývojáři juniornější. Podle části účastníků rozhovorů lze jev pozorovat již nyní u vybraných juniorů, kteří umí výborně s asistenty kódování, ale jsou slabší v řešení problémů nebo chápání základů programování. Participanti některých zahraničních prací dokonce přímo tvrdí, že s používáním asistentů kódování se úplný začátečník nic nenaučí (Akhoroz & Yildirim, 2025). Dalším negativním dopadem je *nižší soustředění na vytvářený kód* v případě, kdy ho generuje asistent kódování. To ve výsledku vede k *menšímu chápání kódu*, což může hrát klíčovou roli při kontrole a odhalování chyb. Jev potvrzují i zahraniční studie. Vaithilingam et al. (2022) dokonce zmiňuje, že vývojáři při používání asistentů kódování rozumějí vygenerovanému kódu výrazně méně.

V programátorských týmech má začlenění asistentů kódování největší dopady na četnost interakce jednotlivých členů mezi sebou. Podle většiny účastníků rozhovorů došlo po zahájení používání asistentů kódování ke *snížení celkového množství pokládaných dotazů mezi kolegy*. Díky tomu dochází v týmech k mnohem menšímu vyrušování jednotlivých členů, čímž se zvýšila jejich soustředěnost.

Využití AI asistentů kódování nemá dopad pouze na efektivitu, úsporu času a množství vyprodukovaného kódu jednotlivce, ale i celé firmy. Proto je podle participantů *klíčové pro zajištění konkurenceschopnosti* alespoň v částečné míře implementovat AI asistenty kódování do procesů firem. Tato zjištění jsou v souladu se zahraničními pracemi. Ty mimo jiné v souvislosti s touto otázkou ještě zmiňují, že vývojáři často oceňují, pokud jsou ve firmě jasně nastavená pravidla týkající se používání umělé inteligence (Arkheden & Eklund, 2024).

Někteří participanti rozhovorů zmiňují, že se pro ně stali asistenti kódování naprosto běžnou součástí jejich práce. Možnost používání asistentů při práci je jedním z kritérií, které by zohlednili při výběru nového zaměstnání. Pokud by potenciální zaměstnavatel zakazoval využívání asistentů kódování, pak by někteří participanti u něho nabídku práce nepřijali. Lze tedy říci, že asistenti kódování nemění pouze samotný proces vývoje softwaru, ale částečně i trh práce. Tento trend bude čím dál častěji ovlivňovat firmy v různých odvětvích,

které budou muset kvůli zachování konkurenceschopnosti změnit svou firemní kulturu a přístup k popisované technologii. V opačném případě hrozí, že zůstanou pozadu a budou mít problém přitáhnout a udržet talentované odborníky. Tím se dostanou do nevýhodné pozice vůči konkurenci, což může přímo ohrozit jejich existenci.

5 Doporučení pro používání nástroje

Cílem kapitoly je definovat na základě provedeného výzkumu sadu pravidel, která v praxi pomohou s co nejefektivnějším využitím a implementací asistentů kódování. Doporučení jsou určena nejen programátorům jako jednotlivcům, aby jim asistenti pomohli jejich práci zvládat rychleji a lépe, ale zaměřuje se také na doporučení týkající se implementace asistentů kódování z pohledu samotných firem.

5.1 Zadávání úkolu

Podkapitola popisuje celkem 5 doporučeních týkajících se zadávání úkolu. Doporučení se zaměřují na vhodnou velikost úkolu, množství informací, jaké by se mělo pro generování kvalitní odpovědi zadávat, nebo jak by měla vypadat vhodná struktura promptu. Dále se doporučení zabývají správou chatovacích vláken, ale také výběrem vhodného jazykového modelu pro generování odpovědi.

5.1.1 Velikost řešeného úkolu

Jazykové modely asistentů kódování lépe zpracovávají úkoly rozdělené do menších částí. Pokud je úkol příliš velký, komplexní nebo složitý, roste pravděpodobnost, že nebude asistent schopen odpověď správně vygenerovat nebo se ztratí v kontextu. Postupné řešení menších problémů zvyšuje úspěšnost odpovědí a umožňuje jít v problematice více do hloubky, což vede k přesnějším a kvalitnějším výstupům. Je tedy lepší volit při používání asistentů přístup zpracování menších samostatných úkolů, než řešit s asistenty komplexní celky.

5.1.2 Množství vstupních informací

Kvalita odpovědi závisí na množství a přesnosti vstupních informací. Vývojáři by si proto měli dát práci se sepsáním maximálního množství podkladů a informací, které mají v rámci řešeného úkolu k dispozici, a neměli by spoléhat na to, že menší množství informací bude asistentovi stačit. Čím více relevantních podkladů asistent obdrží, tím lépe porozumí zadání a poskytne hodnotnější výstup. Na druhou stranu by se množství poskytnutých informací v prvním promptu nemělo ani přehánět. Participanti doporučují, aby byly na začátku poskytnuty všechny klíčové informace, ale samotný prompt nepřekračoval cca 10 vět. V případě delšího zadání nebo většího množství informací může docházet k tomu, že nebudou všechny informace v odpovědi zohledněny, nebo se její kvalita sníží.

5.1.3 Struktura a formulace promptu

Pro dosažení přesnějších odpovědí je vhodné používat strukturované prompty. Každý by měl obsahovat tři základní prvky – jasně definovaný cíl, konkrétní omezení a příklady požadovaného výstupu. Cíl specifikuje, co má odpověď obsahovat nebo jakého výsledku má být dosaženo. Omezení zahrnují pravidla, která je nutné při zpracování úkolu dodržet: například omezení vyplývající ze zadání, projektová pravidla a zvyklosti nebo další omezení, která nemusí být na první pohled zřejmá. Příklady pak poskytují asistentovi lepší představu o chtěném výsledku a zvyšují pravděpodobnost, že odpověď bude odpovídat očekáváním.

Pokud je potřeba k již vygenerované odpovědi doplnit více připomínek najednou, je vhodné je vyjadřovat jasně a každou z připomínek vyjádřit pomocí samostatného bodu (odrážky). Použití samostatných bodů totiž asistentům kódování pomáhá v lepším odlišení a zpracování jednotlivých připomínek.

5.1.4 Používání chatovacích vláken

Důležitým faktorem je správné používání chatovacích vláken. Používání dlouhých vláken s vyšším množstvím promptů není efektivní, protože asistent často nedokáže reflektovat všechny informace uvedené na jeho začátku. Z tohoto důvodu je vhodné řešit každý problém v samostatném vlákně. Pokud jde o komplexnější úlohy, které vyžadují desítky různých promptů, je vhodné rozdělit problém na menší části. Tento přístup umožňuje modelu lépe uchovávat a reflektovat předchozí odpovědi a instrukce, čímž se zvyšuje konzistence a kvalita generovaného obsahu.

5.1.5 Výběr jazykového modelu

Neméně důležitým aspektem je správný výběr jazykového modelu, který zásadně ovlivňuje jak kvalitu generovaných odpovědí, tak i rychlost jejich získání. V případech, kdy je vyžadována vysoká přesnost a schopnost logického uvažování, je vhodné použít tzv. *reasoning modely*[27], jež lépe zvládají řešení složitějších úloh. Pokud je naopak prioritou rychlost odpovědí, pak je lepší využít takové jazykové modely, které tuto schopnost nemají. Klíčem k efektivnímu využívání jazykových modelů je tedy správná identifikace potřeb a očekávání v závislosti na konkrétním problému.

5.2 Na jaké úkoly asistenta používat

Podkapitola obsahuje nejen doporučení, na jaké typy úkolů je vhodné asistenty kódování během práce použít, ale také způsoby užití, které během provedených rozhovorů byly

[27] „*Reasoning*“ v umělé inteligenci (AI) označuje způsob využívání dostupných informací k vytváření předpovědí a vyvozování závěrů. Zahrnuje reprezentaci dat ve formě, kterou stroj dokáže zpracovat, pochopit a následně použít logiku k vytvoření rozhodnutí (Caballar & Stryker, 2025).

unikátní a velmi zajímavé a každý programátor využívající asistenty kódování by si je měl alespoň jednou vyzkoušet.

5.2.1 Silný na standardizovaná řešení

Asistenty kódování lze použít na konzultaci problému již v rané fázi jeho řešení. Mimo to jsou vhodné pro řešení jasně definovaného problému, který se řídí přesně definovanými pravidly nebo algoritmy. Pokud chce vývojář získat originálnější řešení, nebo nápad, případně bohatší a rozmanitější diskusi, s největší pravděpodobností bude lepší upřednostnit místo asistenta kódování raději živého kolegu. To zejména z toho důvodu, že jeho pohled i týmová znalost a znalost projektu nebývá tak úzce omezená jako při definici řešeného problému v několika promptech.

5.2.2 Využití asistentů k rozšiřování znalostí

Participanti často v rámci rozhovorů zmiňovali, že se hledání a získávání nových informací stalo díky asistentům kódování mnohem rychlejší a dostupnější. Této příležitosti je dobré využít a v případech, kdy během řešení zadaných úkolů vznikne určitá nejasnost, nebo programátor objeví mezeru ve svých znalostech, je dobré asistenty využít k jejich doplnění. Takové doplňování má navíc výhodu v tom, že se může vývojář ptát na konkrétní otázky při řešení reálného problému, kde může získané znalosti ihned využít, nebo je zkonzultovat se zkušenějším kolegou. Odpadá tedy nejen dlouhé hledání materiálů k prostudování, ale také problém, kdy programátor přesně neví, co si má doplnit za znalosti, co hledat nebo kde potřebné informace najít.

5.2.3 Validace a doplnění požadavků

Zajímavým použitím asistentů kódování je možnost jejich využití k dodatečné validaci zadání. Ta probíhá tak, že se asistentovi kódování nastaví určitá *persona*[28] sdílející stejné rysy a vlastnosti jako zákazník, pro nějž je software vytvářen. Programátor se pak asistenta ptá na doplňující otázky a snaží se zadané požadavky před zahájením jejich zpracování ověřit. Výsledek může sloužit nejen k formální validaci požadavků a ověření pokrytí všech uživatelových potřeb, ale také k identifikaci dodatečných požadavků, na které se mohlo během fáze návrhu zapomenout. Techniku lze využít ve všech vývojových týmech. Užitečnější bude ovšem spíše v týmech, kde na validaci požadavků nezbývá tolik času a kapacit.

[28] Soubor atributů, který formuje identitu jazykového agenta a způsob jeho odpovědí. Zahrnuje např. osobnost, styl komunikace, způsob přemýšlení, znalosti a úroveň odbornosti v řešené problematice. (Tseng et al., 2024)

5.3 Na jaké úkoly asistenta nepoužívat

Podkapitola obsahuje celkem 3 doporučení, na jaké typy úkolů není vhodné asistenty kódování během práce používat.

5.3.1 Nevýhodný na abstraktnější úlohy

Asistenty kódování není vhodné používat na abstraktnější úkoly, kde je třeba získat velké množství kontextuálních informací, které se do zadání špatně přenášejí. Zpravidla se jedná o různé nastavení aplikací, ale i rozhodnutí, která závisí na znalosti architektury a dalších integrací mezi aplikacemi, jejichž dokumentace často nebývá dostatečná a úplná. V těchto případech je rozhodně vhodnější uvažovat o využití jiných nástrojů, jelikož množství věnovaného času pro vytvoření promptu by bylo neúměrně vysoké kvalitě získané odpovědi.

5.3.2 Generování kódu na základě byznysového zadání

Další oblastí, kde není doporučené asistenty kódování použít, je generování kódu pouze na základě obecných slovních popisů byznysového kontextu. V těchto případech obvykle chybí dostatečná analýza a přesnější specifikace problému, což vede k výsledkům, které jsou neúplné nebo nesprávné. Žádný z participantů nepotvrdil, že by mu tento způsob použití fungoval. Naopak pro získání kvalitního výstupu zmiňovali participanti spíše nutnost se na úkol podívat komplexněji a poskytnout detailnější vstupní informace. Z toho důvodu určitě nelze asistenty používat způsobem, jak propagují samotní tvůrci asistentů kódování, kdy tvrdí, že stačí požadovanou funkcionalitu popsat několika laickými větami a na výstupu bude k dispozici kvalitní a fungující kód. Při zpracování jakéhokoli úkolu je tedy vhodné udělat alespoň základní analýzu požadavku.

5.3.3 Učení se programování

Asistenty kódování není vhodné používat při učení se základům a principům programování. Programátoři vyzdvihují při řešení problémů nenahraditelnost chápání základních principů programování. Zejména někteří juniornější participanti zmiňují, že při používání asistentů kódování během učení méně chápali osvojovanou problematiku. Seniornější vývojáři pak potvrzují, že v případech, kdy začínající programátoři používají při učení se programování asistenty kódování, jejich schopnost samostatně řešit problémy klesá. Naopak při učení se pouze syntaxe nového programovacího jazyka není podle nich v principu s využitím asistentů problém. V těchto případech si myslí, že může dokonce dojít k určité pomoci a urychlení učení.

5.4 Kontrola vygenerované odpovědi

Podkapitola obsahuje celkem 3 doporučení týkající se kontroly vygenerované odpovědi. Doporučení nabízejí možné způsoby provedení kontroly nad vygenerovanou odpovědí, ale

také obsahují typy úkolů, při jejichž zpracování je dobré dbát zvýšené pozornosti. Poslední doporučení se zaměřuje na způsoby získání lepší odpovědi.

5.4.1 Odpovědnost za vygenerovaný kód

Je nutné si uvědomit, že v drtivé většině případů nese odpovědnost za vygenerovanou odpověď vždy vývojář. Asistenti kódování mohou vývojáři sice výrazně pomoci, nikdy by jim však neměl plně důvěřovat a zbavovat se tak tím své odpovědnosti za odvedenou práci. I když asistent pomůže s vytvořením kódu, programátor jej musí být stále schopen analyzovat, upravit a ověřit jeho správnost. K ověření vygenerované odpovědi by mělo dojít ve všech případech, jelikož bez kontroly se podle participantů nezřídka stává, že dojde k nasazení kódu s chybou. Úroveň kontroly by pak měla odpovídat kritičnosti daného kódu, přičemž při ní lze využít nebo vyžadovat podobný princip code review, jako se dnes poměrně běžně používá při pull requestech.

5.4.2 Specifické a méně známé úkoly

Při řešení méně častých nebo velmi specificky zaměřených problémů je nutné častěji a intenzivněji ověřovat přesnost generovaných odpovědí. Zvýšená míra obezřetnosti je v těchto případech nutná zejména z toho důvodu, že asistenti kódování mohou častěji produkovat odpovědi, které zní uvěřitelně a věrohodně, ale obsahují chyby a nesprávné, nebo zavádějící informace. Ještě větší pozornost je pak třeba věnovat tématům, která programátor sám tak dobře nezná. V těchto případech roste riziko použití a akceptace zavádějící odpovědi ještě mnohem rychleji. Pro snížení rizik popsaných v této podkapitole je vhodné vygenerované výsledky častěji ověřovat a konzultovat se zkušenějšími kolegy.

5.4.3 Jak zajistit lepší odpověď

Pokud participanti nejsou spokojeni s vygenerovanou odpovědí, nejčastěji jim pomůže buď doplnění dodatečných informací, nebo kontextu o řešeném problému. Pokud ani to nevede ke zlepšení výsledku, alternativním přístupem je začít psát zamýšlený kód ručně a po několika řádcích nechat asistenta odpověď vygenerovat znovu. Kvalitu odpovědi může dále zvýšit poskytnutí konkrétního příkladu požadovaného výstupu nebo existujícího řešení podobného problému jako předlohy pro generování souvisejícího kódu. Poslední zmíněné řešení lze využít například při tvorbě podobné metody či třídy.

5.5 Doporučení pro firmy a zaměstnavatele

Tato podkapitola obsahuje celkem 7 doporučení určených především pro firmy uvažující o zavedení asistentů kódování do svého prostředí. Doporučení se věnují popisu, jak se z pohledu firmy k asistentům kódování stavět, aby byly nástroje vývojáři co nejlépe přijaty a používány. Zaměřují se ale také na faktory, které je dobré zohlednit, aby byly výsledky asistentů kódování v prostředí firmy co nejlepší.

5.5.1 Asistent kódování jako kritérium při výběru zaměstnání

Jak již bylo v práci několikrát zmiňováno, asistenti kódování se pro mnoho vývojářů stali běžnou součástí práce. Trend je natolik silný, že možnost využívat asistenty kódování se stává mezi vývojáři jedním ze zohledňovaných faktorů při výběru zaměstnání. Pokud by potenciální zaměstnavatel používání asistentů omezoval, existuje reálné riziko, že by vývojáři dali přednost jiné pracovní nabídce. Firmy, které chtějí být atraktivní pro programátory, by proto měly zohlednit, že současní vývojáři považují asistenty kódování za běžný nástroj a očekávají jejich podporu v pracovním prostředí.

5.5.2 Podporovat využívání asistentů zaměstnanci

Místo odmítání nebo zákazu asistentů kódování z pozice firmy je tedy vhodné se soustředit spíše na jejich efektivní využití a odbourávání bariér, které ve firemních prostředích mohou vznikat. Díky implementaci asistentů kódování a snížení překážek v jejich používání budou moci vývojáři pracovat rychleji, efektivněji a s větší spokojeností, protože mají více prostoru na zajímavější a složitější úkoly. Výsledkem implementace tedy může být nejen vyšší produktivita, kvalita, množství vytvářených artefaktů za jednotku času, ale také spokojenost vývojářů. Navíc je v souvislosti s podporou a implementací dobré zvážit definici jasných pravidel, jak s umělou inteligencí pracovat. Vývojáři existenci těchto pravidel uvnitř firmy často oceňují a jsou si díky nim při používání nástrojů jistější (Pan et al., 2024).

5.5.3 Podporovat sdílení zkušeností

Přestože asistenti kódování patří mezi běžně používané nástroje, v mnoha vývojářských týmech chybí aktivní sdílení zkušeností s jejich využitím. Zatímco při jejich zavádění docházelo ke sdílení poznatků poměrně často, postupem času, jak si každý vývojář našel svůj vlastní způsob interakce s asistentem, tato výměna informací prakticky přestala. Přitom sdílení zkušeností a způsobů užití asistentů je velmi dobrým zdrojem, jak přijít na nové nebo efektivnější způsoby jejich používání. To potvrzuje i fakt, že někteří participanti se na konci prováděných rozhovorů sami ptali a zajímali o to, jak s asistenty pracují jejich kolegové. Podpora pravidelné výměny zkušeností ze strany zaměstnavatele nebo týmu by tak mohla přispět k vyšší efektivitě práce i lepšímu využití a adopci asistentů do každodenního vývoje.

5.5.4 Komunikace výhod asistentů mezi zaměstnance

Programátoři považují asistenty kódování za užitečné, ne všichni jsou ale ochotni sami aktivně vyhledávat informace o jejich možnostech a správném používání. Někteří vývojáři začali nástroje využívat až na vyzvání zaměstnavatele. To může vést k situacím, kdy nejsou nástroje ve firmě plně využity, protože zaměstnanci neznají jejich funkce nebo si nejsou jisti, jak je zapojit do své práce. Při zavádění asistentů kódování do firemního prostředí je proto zásadní zajistit, aby se informace o schopnostech a výhodách asistentů dostaly ke všem zaměstnancům. Toho lze dosáhnout například prostřednictvím interních workshopů, podporou sdílení zkušeností mezi kolegy nebo sdílením praktických ukázek využití nástrojů.

Aktivní komunikace pomůže nejen zvýšit adopci a povědomí o nástrojích, ale také zajistí, že jejich přínosy budou ve firmě maximálně využity.

5.5.5 Podpora integrace nástrojů mezi sebou

Dále je žádoucí v maximální možné míře podporovat integraci různých nástrojů mezi sebou. V současnosti mají jednotlivé aplikace k dispozici cenné informace, často je ale mezi sebou nedokáží efektivně sdílet. To vede buď k tomu, že některé informace zůstávají při generování odpovědí asistenty kódování nevyužité, nebo je programátor musí vyhledávat a ručně zadávat do promptů, což snižuje jeho rychlost. Lepší propojení nástrojů a sdílení dat mezi nimi by mohlo tento problém eliminovat a dále zvýšit užitečnost AI asistentů kódování ve vývojovém procesu.

5.5.6 Zavádět plně funkční řešení

V případech, kdy se jakákoli firma rozhodne zavést vlastní asistenty kódování, je zásadní, aby do provozu uváděla od počátku plně funkční řešení. Jakékoli nedostatky, omezení nebo neúplné fungování mohou vést k tomu, že programátoři nebudou chtít firemní nástroj využívat a najdou si alternativní asistenty, které budou fungovat podle jejich představ. Jakmile si ovšem zvyknou na používání externího asistenta, nemusí se jim chtít k firemnímu řešení vrátit. Tato situace může nastat i v případech, kdy dojde dodatečně k odstranění všech nedokonalostí ve firemním asistentovi. Aby se podobné situaci předešlo, je důležité ihned po zavedení do firemního prostředí nabídnout plně funkční nástroj, který funguje a nemá oproti jiným běžně dostupným asistentům žádné výrazné kompromisy a omezení. V opačném případě hrozí, že firemní asistent nebude přijat a jeho využití zůstane minimální.

5.5.7 Používat co nejméně privátních knihoven

Z provedeného výzkumu vyplývá, že s ohledem na využití asistentů kódování je při vytváření softwaru nejlepší používat co nejméně proprietárních a soukromých knihoven, jejichž chování není nikde zdokumentováno. Tím se snižuje riziko, že asistent nebude schopen správně generovat a upravovat zadaný kód. Pokud nemá firma vážný důvod k používání vlastního řešení, omezení proprietárních řešení ve prospěch standardních a dobře zdokumentovaných knihoven tak může výrazně přispět ke kvalitě generovaných výsledků.

Závěr

Hlavním cílem diplomové práce bylo popsat, jaká jsou očekávání programátorů v České republice při používání tzv. AI asistentů kódování, jak tyto nástroje programátoři v České republice nejčastěji využívají, jak dopadá používání této technologie na jejich práci, a dále navrhnout sadu doporučení umožňující co nejefektivnější využití a implementaci těchto nástrojů. Hlavní cíl byl rozdělen na 4 dílčí cíle.

V rámci dílčího cíle 1 (DC1) měla být provedená rešerše existující literatury věnující se aktuálnímu stavu poznání využití AI asistentů kódování, jejich dopadům na vývojáře a jejich práci. Rešerše byla provedena a její výsledky jsou popsány v kapitole 1. Tento dílčí cíl práce byl tedy naplněn.

Druhým dílčím cílem (DC2) bylo na základě provedené rešerše a poznatků z již existující literatury vytvořit interview protokol zaměřující se na využití AI asistentů kódování, jejich dopady na vývojáře a jejich práci v kontextu České republiky. Popisu postupu tvorby protokolu se věnuje kapitola 2.3. Vytvořený interview protokol je pak k dispozici v příloze A. Dílčí cíl 2 lze tedy považovat také za splněný.

Podle sestaveného protokolu následně probíhaly rozhovory s vybranými vývojáři. Uskutečnění rozhovorů a jejich následná tematická analýza byla náplní dílčího cíle 3 (DC3). Průběh rozhovorů a celého sběru dat popisuje kapitola 2.4, postup při jejich analýze pak kapitola 2.5. Výsledky provedené analýzy detailně představuje kapitola 3. V kapitole 4 jsou výsledky provedené analýzy diskutovány a jsou v ní zodpovězené všechny tři stanovené výzkumné otázky. Ve zmíněných kapitolách je tak zdokumentováno naplnění třetího dílčího cíle.

Posledním (čtvrtým) dílčím cílem (DC4) bylo vytvořit a ověřit sadu doporučení umožňující co nejefektivnější využití a implementaci AI asistentů kódování. Průběhu tvorby a ověření sady doporučení se věnuje kapitola 2.6. Finální sada doporučení je detailně popsána v kapitole 5. Tím se naplnil i čtvrtý dílčí cíl práce.

Všechny stanovené dílčí cíle práce byly postupně naplněny, což vedlo i k naplnění samotného hlavního stanoveného cíle této diplomové práce.

Význam a přínos

Hlavním přínosem práce je zodpovězení stanovených výzkumných otázek. Zodpovězením došlo k vytvoření uceleného popisu, jaká očekávání od AI asistentů kódování mají programátoři v České republice, jak je vnímají, jak je nejčastěji využívají a jaké mají AI asistenti kódování dopady na práci vývojářů v České republice. Na základě toho vznikla sada doporučení, jak přistupovat k používání a začlenění AI asistentů kódování do procesu vývoje softwaru.

Doporučení je určeno jak samotným vývojářům, tak vedoucím pracovníkům zodpovědným za implementaci tohoto typu nástroje v podniku. Vývojářům pomohou zjistit, jaké zkušenosti mají s AI asistenty kódování jejich kolegové, jaké způsoby a strategie jsou při používání asistentů kódování nejefektivnější. Mimo jiné doporučení ukazují, na co si dát při používání těchto nástrojů pozor. Vedoucím pracovníkům pak doporučení odhalují, na co při zavádění AI asistentů kódování klást důraz, jak správně nastavit firemní prostředí, aby podporovalo využívání těchto nástrojů a aby byli asistenti kódování pro vývojáře přínosným a efektivním pomocníkem při jejich práci a neměli obavy z jejich využívání.

Omezení práce

Jako hlavní omezení této práce vidí autor velikost zkoumaného vzorku participantů. Dále jako omezení vnímá i menší různorodost firem, pro které participanti pracovali. Přístup k používání AI asistentů kódování a regulace jejich používání ve firmě může být ovlivněna nejen oborem, ve kterém působí, ale také velikostí firmy. Různé firmy tak mohou mít různé přístupy k používání asistentů. Postoje firmy pak do jisté míry ovlivňují i ochotu samotných zaměstnanců asistenty používat.

Aby mohl autor výsledky správně generalizovat pro české prostředí, bylo by potřeba rozhovory uskutečnit na větším počtu participantů, případně práci rozšířit ještě o kvantitativní část, která by potvrzovala zjištění získaná v části kvalitativní. Je nutné také zmínit fakt, že u analyzovaných participantů se stále jedná o osobní názory, které mohou být zkreslené a ovlivněné jejich zkušenostmi, osobními preferencemi, znalostmi a schopnostmi v daném oboru.

Částečně může být omezením i validita analyzovaných zjištění. Každý výzkumník na získaná data nahlíží z jiné perspektivy a může klást důraz na konkrétní oblast nebo problematiku a mírně upozadit jiná zjištění. Pro zajištění co největší míry validity a odhalení většího množství zajímavých informací v provedených rozhovorech by tedy pomohlo, pokud by se vyhodnocení věnoval větší tým výzkumníků, kteří by kódování prováděli nezávisle na sobě.

Doporučení pro budoucí práce

Budoucí práce by se mohly zaměřit na dlouhodobý vliv AI asistentů kódování na kvalitu kódu. I když tyto nástroje vývojářům subjektivně zjednodušují a zrychlují vývoj, zatím není dostatečně prozkoumáno, zdali jejich používání v delším časovém horizontu neovlivňuje kvalitu generovaného kódu. Výzkum by tak mohl zahrnovat srovnání kvality kódu u vývojářů používajících asistenty pravidelně a těch, kteří je využívají minimálně nebo vůbec.

Někteří participanti během rozhovorů sami uvedli, že způsob a míra používání AI asistentů kódování se může lišit v závislosti na pracovním prostředí, ve kterém aktuálně pracují. Tento poznatek naznačuje další možný směr budoucího výzkumu, a to zkoumání rozdílů ve využívání AI asistentů kódování v kontextu pracovního prostředí. Zvláštní pozornost by přitom mohla být věnována např. srovnání používání těchto nástrojů při práci z domova a v kancelářském prostředí.

Použitá literatura

Akhoroz, M., & Yildirim, C. (2025). Conversational AI as a Coding Assistant: Understanding Programmers' Interactions with and Expectations from Large Language Models for Coding (No. arXiv:2503.16508). arXiv. https://doi.org/10.48550/arXiv.2503.16508

Alford, A. (2021, srpen 31). OpenAI Announces 12 Billion Parameter Code-Generation AI Codex. InfoQ. https://www.infoq.com/news/2021/08/openai-codex/

Arkheden, L., & Eklund, S. (2024). Evaluating the Use of Generative AI in Software Development Proposing a Tentative Framework. http://hdl.handle.net/20.500.12380/307912

Asare, O., Nagappan, M., & Asokan, N. (2024). A User-centered Security Evaluation of Copilot. Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. https://doi.org/10.1145/3597503.3639154

Bird, C., Ford, D., Zimmermann, T., Forsgren, N., Kalliamvakou, E., Lowdermilk, T., & Gazit, I. (2023). Taking Flight with Copilot. Communications of the ACM, 66(6), 56–62. https://doi.org/10.1145/3589996

Braun, V., & Clarke, V. (2021). Thematic Analysis: A Practical Guide. Sage Publications Ltd.

Brousse, N. (2019). The issue of monorepo and polyrepo in large enterprises. Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming, 1–4. https://doi.org/10.1145/3328433.3328435

Caballar, R. D., & Stryker, C. (2025, březen 14). What Is Reasoning in AI? | IBM. https://www.ibm.com/think/topics/ai-reasoning

Dylan Walsh. (2024, listopad 4). How generative AI affects highly skilled workers | MIT Sloan. https://mitsloan.mit.edu/ideas-made-to-matter/how-generative-ai-affects-highly-skilled-workers

Evropská komise. (2021). SME definition—European Commission. https://single-market-economy.ec.europa.eu/smes/sme-fundamentals/sme-definition_en

Exafunction, Inc. (b.r.). Codeium vs Github Copilot | Windsurf (formerly Codeium). Získáno 4. duben 2025, z https://windsurf.com/compare/comparison-copilot-codeium

Federal Office for Information Security. (2024). AI Coding Assistants. Bundesamt für Sicherheit in der Informationstechnik. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/KI/ANSSI_BSI_AI_Coding_Assistants.pdf?__blob=publicationFile&v=7

GitHub, Inc. (b.r.). Getting code suggestions in your IDE with GitHub Copilot—GitHub Enterprise Cloud Docs. GitHub Docs. Získáno 21. březen 2025, z https://docs.github.com/en/enterprise-cloud@latest/copilot/using-github-copilot/getting-code-suggestions-in-your-ide-with-github-copilot

GitHub, Inc. (2022). GitHub Copilot · Your AI pair programmer. GitHub. https://github.com/features/copilot

GitHub, Inc. (2023, listopad 30). GitHub Copilot – November 30th Update · GitHub Changelog. The GitHub Blog. https://github.blog/changelog/2023-11-30-github-copilot-november-30th-update/

GitHub, Inc. (2024). Best practices for using GitHub Copilot. GitHub Docs. https://docs.github.com/en/copilot/using-github-copilot/best-practices-for-using-github-copilot

Haque, E. A., Brown, C., LaToza, T. D., & Johnson, B. (2024). Information Seeking Using AI Assistants (No. arXiv:2408.04032). arXiv. https://doi.org/10.48550/arXiv.2408.04032

Kalliamvakou, E. (2022, září 7). Research: Quantifying GitHub Copilot's impact on developer productivity and happiness [Blog]. The GitHub Blog. https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/

Kedar, S. (2024, červen 17). Tabnine vs. GitHub Copilot. Tabnine. https://www.tabnine.com/blog/tabnine-versus-github-copilot/

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations. IT Revolution.

Klemmer, J. H., Horstmann, S. A., Patnaik, N., Ludden, C., Burton Jr, C., Powers, C., Massacci, F., Rahman, A., Votipka, D., Lipford, H. R., Rashid, A., Naiakshina, A., & Fahl, S. (2024). Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns (No. arXiv:2405.06371). arXiv. https://doi.org/10.48550/arXiv.2405.06371

Li, Z. S., Arony, N. N., Awon, A. M., Damian, D., & Xu, B. (2024, červen 25). AI Tool Use and Adoption in Software Development by Individuals and Organizations: A Grounded Theory Study. arXiv.Org. https://arxiv.org/abs/2406.17325v1

Mendes, W., Souza, S., & De Souza, C. (2024). „You're on a bicycle with a little motor": Benefits and Challenges of Using AI Code Assistants. Proceedings of the 2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering, 144–152. https://doi.org/10.1145/3641822.3641882

Microsoft Corporation. (b.r.-a). GitHub Copilot—Visual Studio Marketplace. Získáno 4. duben 2025, z https://marketplace.visualstudio.com/items?itemName=GitHub.copilot

Microsoft Corporation. (b.r.-b). Windsurf Plugin (formerly Codeium): AI Coding Autocomplete and Chat for Python, JavaScript, TypeScript, and more - Visual Studio Marketplace. Získáno 4. duben 2025, z https://marketplace.visualstudio.com/items?itemName=Codeium.codeium

Microsoft Corporation. (2025, únor 6). Tips and tricks for Copilot in VS Code. https://code.visualstudio.com/docs/copilot/copilot-tips-and-tricks

OBrien, D., Biswas, S., Imtiaz, S. M., Abdalkareem, R., Shihab, E., & Rajan, H. (2024). Are Prompt Engineering and TODO Comments Friends or Foes? An Evaluation on GitHub Copilot. Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. https://doi.org/10.1145/3597503.3639176

OpenAI, L.L.C. (b.r.). ChatGPT. Získáno 21. březen 2025, z https://openai.com/chatgpt/overview/

OpenAI, L.L.C. (2021, srpen 10). OpenAI Codex. https://openai.com/index/openai-codex/

OpenAI, L.L.C. (2022, listopad 30). Introducing ChatGPT. https://openai.com/index/chatgpt/

OpenAI, L.L.C. (2023). GPT-4. https://openai.com/index/gpt-4/

Pan, S., Wang, L., Zhang, T., Xing, Z., Zhao, Y., Lu, Q., & Sun, X. (2024). „I Don't Use AI for Everything": Exploring Utility, Attitude, and Responsibility of AI-empowered Tools in Software Development (No. arXiv:2409.13343). arXiv. https://doi.org/10.48550/arXiv.2409.13343

Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., & Santos, E. A. (2023). "It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. ACM Trans. Comput.-Hum. Interact., 31(1). https://doi.org/10.1145/3617367

Prokopets, M. (2024, duben 5). The Ultimate Manual to GitHub Copilot. Nira. https://nira.com/github-copilot/

Salva, R. (b.r.). Essentials of GitHub Copilot. GitHub Resources. Získáno 24. listopad 2024, z https://resources.github.com/learn/pathways/copilot/essentials/essentials-of-github-copilot/

Sergeyuk, A., Golubev, Y., Bryksin, T., & Ahmed, I. (2025). Using AI-Based Coding Assistants in Practice: State of Affairs, Perceptions, and Ways Forward. Information and Software Technology, 178, 107610. https://doi.org/10.1016/j.infsof.2024.107610

Stack Exchange Inc. (2024). Stack Overflow Developer Survey 2024. https://survey.stackoverflow.co/2024/

Tabnine Ltd. (b.r.). Plans & Pricing | Tabnine: The AI code assistant that you control. Tabnine. Získáno 4. duben 2025, z https://www.tabnine.com/pricing/

Tabnine Ltd. (2024, květen 30). Security | Tabnine Docs. https://docs.tabnine.com/main/welcome/readme/security

Tabnine Ltd. (2025, březen 31). AI Models | Tabnine Docs. https://docs.tabnine.com/main/welcome/readme/ai-models

Tseng, Y.-M., Huang, Y.-C., Hsiao, T.-Y., Chen, W.-L., Huang, C.-W., Meng, Y., & Chen, Y.-N. (2024). Two Tales of Persona in LLMs: A Survey of Role-Playing and Personalization (No. arXiv:2406.01171). arXiv. https://doi.org/10.48550/arXiv.2406.01171

Vaithilingam, P., Glassman, E. L., Groenwegen, P., Gulwani, S., Henley, A. Z., Malpani, R., Pugh, D., Radhakrishna, A., Soares, G., Wang, J., & Yim, A. (2023). Towards More Effective AI-Assisted Programming: A Systematic Design Exploration to Improve Visual Studio IntelliCode's User Experience. Proceedings of the 45th International Conference on Software Engineering: Software Engineering in Practice, 185–195. https://doi.org/10.1109/ICSE-SEIP58684.2023.00022

Vaithilingam, P., Zhang, T., & Glassman, E. L. (2022). Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems, 1–7. https://doi.org/10.1145/3491101.3519665

Windsurf Team. (2023, březen 24). How is Codeium Free? https://windsurf.com/blog/how-is-codeium-free

Zhang, B., Liang, P., Zhou, X., Ahmad, A., & Waseem, M. (2023, březen 15). Practices and Challenges of Using GitHub Copilot: An Empirical Study. Proceedings of the 35th International Conference on Software Engineering and Knowledge Engineering (SEKE 2023). https://doi.org/10.18293/SEKE2023-077

Ziegler, A., Kalliamvakou, E., Li, X. A., Rice, A., Rifkin, D., Simister, S., Sittampalam, G., & Aftandilian, E. (2024). Measuring GitHub Copilot's Impact on Productivity. Commun. ACM, 67(3), 54–63. https://doi.org/10.1145/3633453

Přílohy

Příloha A: Interview protokol

Kategorie	Otázka	Doplňující otázky
Úvod	V krátkosti se, prosím, představte.	• typ vývoje (FE, BE, fullstack, mobilní…) • používané technologie frameworky • délka vývojářské praxe
	V krátkosti, prosím, představte svůj projekt a tým.	• hlavní myšlenka • firma • velikost týmu • počet vývojářů • aktuální role v týmu
	Zkuste popsat, jak AI coding asistenty vnímáte Vy osobně.	
	Jak dlouho coding asistenty používáte?	
	Jaké coding asistenty jste v souvislosti s vývojem softwaru již použil(a)?	• četnost používání nástroje • účely použití • spokojenost s nástroji • jejich (krátké) srovnání • důvody k přechodu ke konkurenci

Očekávání	Proč jste se rozhodl(a) coding asistenty začít používat a pracovat s nimi?	• důvod
		• jejich potenciál
		• dopady na produktivitu nebo jiný aspekt práce
		  ○ fáze vývoje
		  ○ rychlost vývoje
		  ○ kvalita kódu
		  ○ méně chyb

	Jaké konkrétní dopady jste očekával(a), že bude mít využití coding asistentů na vaši	• fáze vývoje • rychlost vývoje • kvalita kódu • méně chyb

	produktivitu nebo jiný aspekt vaší práce?	

	Jaké jsou nyní Vaše hlavní důvody a očekávání, proč coding asistenty používáte?	• očekávaný druh pomoci
		• přínosy, hodnota
		• dopady
		  ○ fáze vývoje
		  ○ rychlost vývoje
		  ○ kvalita kódu
		  ○ méně chyb
		• očekávané výstupy
		• způsoby interakce (IDE, mobilní aplikace, jinak…)

	Jak vnímáte coding asistenty z hlediska bezpečnosti a soukromí?	• očekávání • práce s osobními údaji • nakládání s kódem • nakládání s citlivými daty organizace • využití pro zlepšení bezpečnosti vytvářeného kódu
Používání	Jak často coding asistenty při práci využíváte?	
	Popište, prosím, jak coding asistenty používáte během Vaší práce.	• typ nejčastěji řešeného úkolu • nehodící se typ úkolu/limity • specifika jednotlivých typů úkolů • vyhledávání informací • větší vs. menší úkoly/celky • vymýšlení promptů, promptovací strategie • důvěra v asistenty

	Jakým způsobem dále pracujete s vygenerovanou odpovědí/návrhem?	• Probíhá kontrola správnosti?
		  ○ Jak?
		  ○ časová náročnost
		• Musíte generované návrhy upravovat?
		  ○ Proč?
		  ○ Jak často?
		  ○ V jakých případech?

	Jakým způsobem se Vaše společnost staví k používání AI při práci?	• vlastní modely • podpora vs. zákaz • kultura ohledně AI

		• (ne)povolené scénáře použití AI
		• existence pravidel a směrnic
		  ○ struktura směrnic
		  ○ obsah směrnic
		  ○ její užitečnost

	Jak se osobně cítíte při používání coding asistentů?	• obavy • pozitivní pocity • očekávání s ohledem na budoucí vývoj nástroje • limity nástrojů
Dopad	Jak přesně na Vaši práci dopadá používání coding asistentů? Zvažte kladné i záporné.	• spokojenost • kvalita práce • výkonnost • efektivita • výhody/nevýhody jeho využívání • změna pracovních návyků (charakter a rozsah změny, rychlost přijetí změny)
	Jak ovlivnilo používání coding asistentů dynamiku a spolupráci ve vašem týmu?	• pozitivní i negativní • dopad na komunikaci, pomoc a např. na code review • sdílíte si poznatky o využívání nástrojů
	Jak z Vašeho pohledu používání coding asistentů změnilo podobu a způsob práce v oboru vývoje softwaru?	• pozitivně • negativně

	Jak si myslíte, že se v budoucnu dále změní přístup k vývoji softwaru díky používání coding asistentů?	• primární změny
		  ○ náplň práce
		  ○ best practices
		  ○ procesy
		  ○ metodiky atd.?
		  ○ úroveň potřebných znalostí
		• horizont změn
		• dopad nástrojů na kvalitu
		  ○ zvýšení vs. snížení?

	Jak si myslíte, že Vámi dříve popsané změny ovlivní konkrétně Vaši práci?	• méně psaní kódu, více review • plné nahrazení coding asistenty pro určité úkoly

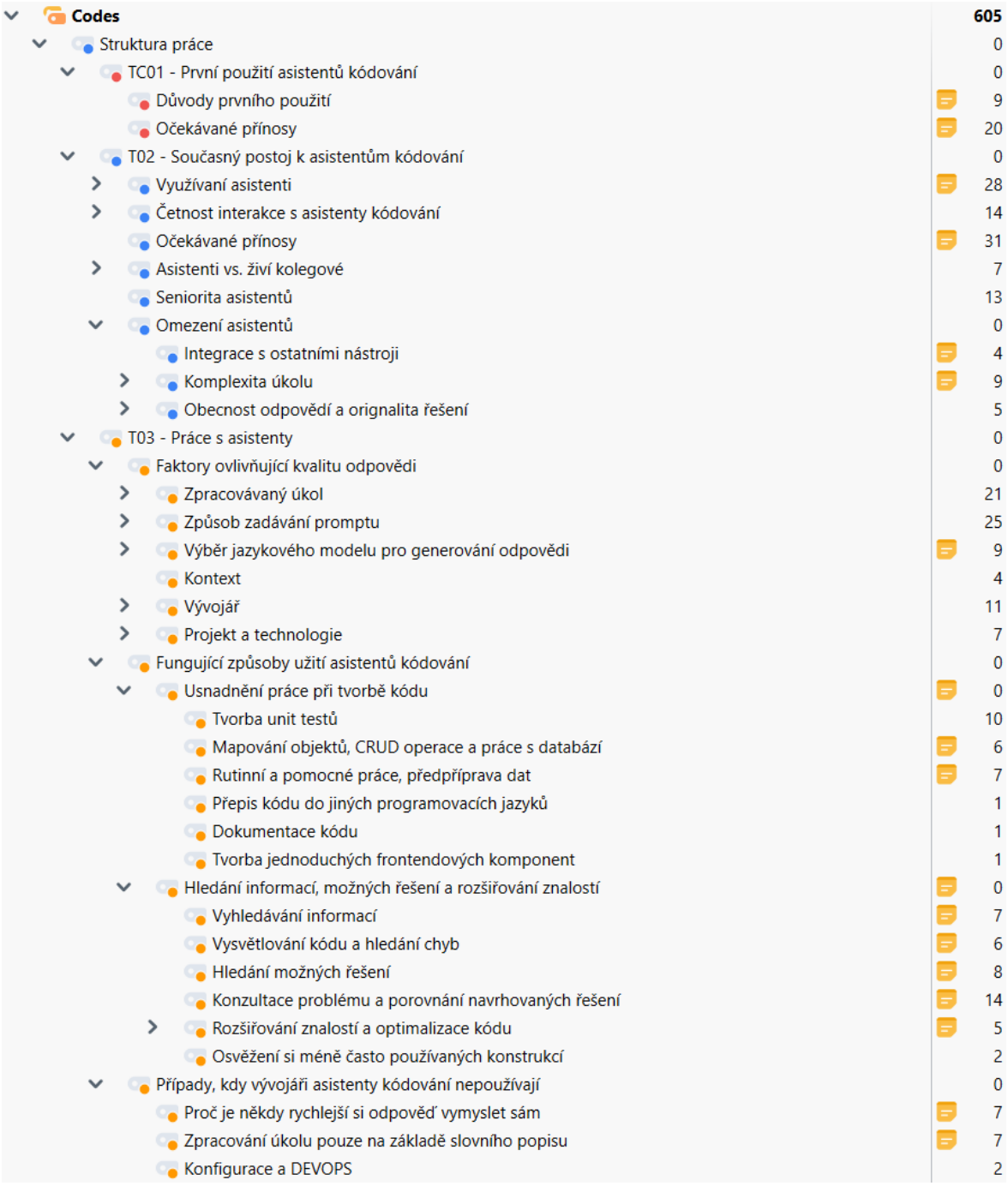
Závěr	Jaká doporučení byste dal(a) týmům/jednotlivcům, které zatím coding asistenty neimplementovaly, ale v nejbližší době se na implementaci chystají?	• Co by Vám pomohlo pro lepší práci s asistenty? • jednotlivec • tým • organizace a její kultura • co pomáhá • co nepomáhá
	Máte další poznatky nebo poznámky, které jste neměl(a) možnost během rozhovoru sdělit, ale myslíte si, že by měly ještě zaznít?	

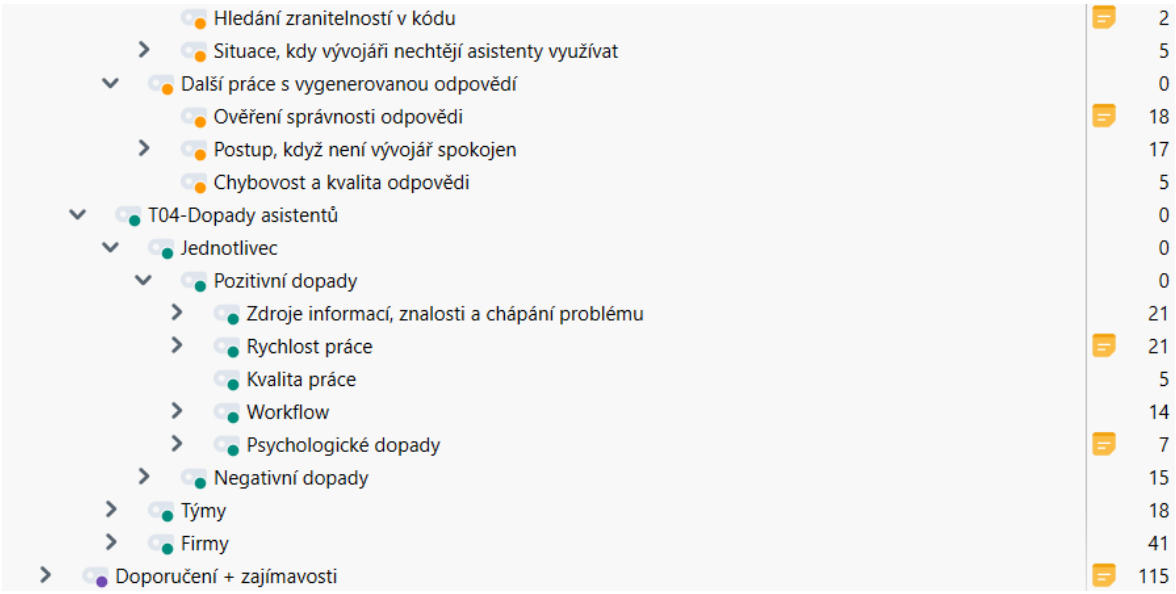
Příloha B: Mapování otázek interview protokolu na literaturu

Kategorie	Otázka	Zdroj
Úvod	V krátkosti se, prosím, představte.	autor
	V krátkosti, prosím, představte svůj projekt a tým.	autor
	Zkuste popsat, jak AI coding asistenty vnímáte Vy osobně.	autor
	Jak dlouho coding asistenty používáte?	autor
	Jaké coding asistenty jste v souvislosti s vývojem softwaru již použil(a)?	autor
Očekávání	Proč jste se rozhodl(a) coding asistenty začít používat a pracovat s nimi?	autor
	Jaké konkrétní dopady jste očekával(a), že bude mít využití coding asistentů na vaši produktivitu nebo jiný aspekt vaší práce?	autor
	Jaké jsou nyní Vaše hlavní důvody a očekávání, proč coding asistenty používáte?	*Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns* (Klemmer et al., 2024)
	Jak vnímáte coding asistenty z hlediska bezpečnosti a soukromí?	autor
Používání	Jak často coding asistenty při práci využíváte?	*Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns* (Klemmer et al., 2024)
	Popište, prosím, jak coding asistenty používáte během Vaší práce.	*Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns* (Klemmer et al., 2024)

	Jakým způsobem dále pracujete s vygenerovanou odpovědí/návrhem?	*Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns* (Klemmer et al., 2024)
	Jakým způsobem se Vaše společnost staví k používání AI při práci?	*Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns* (Klemmer et al., 2024)
	Jak se osobně cítíte při používání coding asistentů?	*Evaluating the Use of Generative AI in Software Development Proposing a Tentative Framework (Arkheden & Eklund, 2024)*
Dopad	Jak přesně na Vaši práci dopadá používání coding asistentů? Zvažte kladné i záporné.	*Evaluating the Use of Generative AI in Software Development Proposing a Tentative Framework (Arkheden & Eklund, 2024)*
	Jak ovlivnilo používání coding asistentů dynamiku a spolupráci ve vašem týmu?	*Evaluating the Use of Generative AI in Software Development Proposing a Tentative Framework (Arkheden & Eklund, 2024)*
	Jak z Vašeho pohledu používání coding asistentů změnilo podobu a způsob práce v oboru vývoje softwaru?	autor
	Jak si myslíte, že se v budoucnu dále změní přístup k vývoji softwaru díky používání coding asistentů?	*Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns* (Klemmer et al., 2024)
	Jak si myslíte, že Vámi dříve popsané změny ovlivní konkrétně Vaši práci?	*Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns* (Klemmer et al., 2024)
Závěr	Jaká doporučení byste dal(a) týmům/jednotlivcům, které zatím coding asistenty neimplementovaly, ale v nejbližší době se na implementaci chystají?	autor
	Máte další poznatky nebo poznámky, které jste neměl(a) možnost během rozhovoru sdělit, ale myslíte si, že by měly ještě zaznít?	autor

Příloha C: Výstup tematické analýzy




Hledání zranitelností v kódu 2
Situace, kdy vývojáři nechtějí asistenty využívat 5
Další práce s vygenerovanou odpovědí 0
Ověření správnosti odpovědi 18
Postup, když není vývojář spokojen 17
Chybovost a kvalita odpovědi 5
T04-Dopady asistentů 0
Jednotlivec 0
Pozitivní dopady 0
Zdroje informací, znalosti a chápání problému 21
Rychlost práce 21
Kvalita práce 5
Workflow 14
Psychologické dopady 7
Negativní dopady 15
Týmy 18
Firmy 41
Doporučení + zajímavosti 115