Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra aplikované lingvistiky

Digitální technologie ve vzdělávání

Dopad umělé inteligence na obor programování

Diplomová práce

Autor: František Bílek
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Mgr. et Mgr. Marcel Pikhart, Ph.D.

Hradec Králové duben 2024

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 22.4.2024

František Bílek

Poděkování:

Děkuji vedoucímu diplomové práce doc. Mgr. et Mgr. Marcelu Pikhartovi, Ph.D. za metodické vedení, vstřícnost a ochotu při zpracování této práce.

Abstrakt

Tato diplomová práce se zaměřuje na porovnání schopností programátorů a běžných uživatelů s přístupem k nástrojům umělé inteligence v řešení objektově orientovaných programovacích úloh. V rámci pilotního výzkumu byly testovány dvě skupiny: programátoři bez přístupu k AI a uživatelé bez technických znalostí v oboru programování využívající nástroj ChatGPT. Hlavním cílem bylo zjistit, zda může AI v rukou nezkušeného běžného uživatele efektivně nahradit programátory v konkrétních programovacích úlohách. Výsledky ukázaly, že skupina běžných uživatelů, využívající AI, byla v řešení úloh výrazně rychlejší bez znatelného snížení kvality kódu, což naznačuje, že pro zpracování těchto úloh nejsou zapotřebí pokročilé technické znalosti. Tato zjištění prokazují možnost technologické nezaměstnanosti v oblasti programování. Metodologie této práce je značně omezená a doporučuje rozšiřující výzkum. Práce identifikuje několik omezení, a navrhuje způsob, jak tyto problémy v budoucím výzkumu vyřešit. Diskusní část práce kromě těchto omezení poukazuje na nutnost aktualizace právního a vzdělávacího systému tak, aby byly připraveny na možnosti a hrozby umělé inteligence.

Abstract

Title: Digital technologies in education

This diploma thesis focuses on comparing the abilities of programmers and ordinary users with access to artificial intelligence tools in solving object-oriented programming tasks. During the pilot study, two groups were tested: programmers without AI access and users without technical knowledge in programming using the ChatGPT tool. The main objective was to determine whether AI, in the hands of an inexperienced ordinary user, can effectively replace programmers in specific programming tasks. Results showed that the group of ordinary users, utilizing AI, was significantly faster in solving tasks without a noticeable decrease in code quality, suggesting that advanced technical knowledge is not required

for processing these tasks. These findings demonstrate the potential for technological unemployment in the programming field. The methodology of this work is considerably limited and recommends further research. The work identifies several limitations and suggests how these issues can be resolved in future research. The discussion section of the paper, besides these limitations, highlights the need to update the legal and educational systems to be prepared for the possibilities and threats of artificial intelligence.

Klíčová slova: umělá inteligence, chatgpt, technologická nezaměstnanost, velké jazykové modely, vzdělávání, generativní

Key words: artificial intelligence, chatgpt, technological unemployment, large language models, education, generative

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Pojem umělá inteligence (Artifical Intelligence, dále v textu jen jako AI) je už ustálený a obecně známý, ale v posledních letech se v této oblasti objevil velký pokrok ve formě generativní umělé inteligence, která je schopná vytvářet nový obsah nebo data na základě předchozích vzorů. Tento pokrok se bude pravděpodobně dál rozvíjet (Dean et al., 2023). Možnosti dnešní implementace umělé inteligence jsou velmi rozsáhlé a není možné přesně určit jejich limit. Je i možné že za pár let bude umělá inteligence na kompletně jiné úrovni (Sachdeva, 2024). Práce podrobně analyzuje fundamentální principy, evoluční trajektorii a aplikace umělé inteligence v období od neuronových sítí až po aktuální vrcholové modely. Zabývá se komplexním vývojem těchto technologických inovací a jejich náročným technickým výzkumem, přičemž zkoumá klíčové aspekty v oblasti umělé inteligence a vzdělávání, zejména v oboru aplikované informatiky.

Výzkumná část práce je zaměřena na porovnání schopnosti programátora, a běžného uživatele s přístupem k nástrojům umělé inteligence. Jedná se o pilotní výzkum v této oblasti. Pilotní výzkum hraje zásadní roli v mnoha oblastech vědeckého bádání. Tato počáteční fáze výzkumu je zaměřena na ověření metodiky, sběr předběžných dat a identifikaci potenciálních problémů, které by mohly ovlivnit hlavní studii. Význam výzkumu nelze podceňovat, jelikož poskytuje cenný pohled do proveditelnosti, nákladové efektivity a potřebného časového rámce pro rozsáhlejší výzkum. Díky pilotnímu výzkumu je možné upravit návrh hlavní studie tak, aby se zvýšila její validita a spolehlivost. Tato fáze nejenže napomáhá v optimalizaci výzkumných nástrojů a postupů, ale také vede k lepšímu pochopení dynamiky výzkumného tématu. Pilotní výzkum představuje cenný předpoklad pro úspěšné provedení komplexnějšího a detailnějšího zkoumání zvoleného problému, a vytváří základ pro budoucí objevy a inovace.

S velkým pokrokem v oblasti AI se také začínají objevovat různé problémy, zejména možnost nahrazení lidských zaměstnanců nějakým AI systémem. Nejvíce zasažené oblasti jsou zdravotnictví, finance, maloobchod a elektronický obchod, výroba, lidské zdroje, zákaznická podpora, média a tvorba obsahu, a vzdělávání (George et al., 2023).

Dalšími sektory jsou transport a logistika, zemědělství, energie, marketing a reklamy (George et al., 2023). Dalším problémem je etické používání a tvorba AI systémů. AI je trénováno na datech od několika autorů, kteří za to většinou nejsou nijak kompenzováni. Trénink je také ve většině případů proveden bez souhlasu těchto autorů (Vynck, 2023). Následně hrozí že AI pak tyto autory nahradí. Jedná se tedy o problém regulace a kompenzace trénování AI, a případně autorských práv z pohledu použití dat pro trénování. Také existuje etické dilema, zda je nahrazování lidských zaměstnanců AI systémem morálně přijatelné (Laker, 2023).
U generativních AI může docházet k vytváření obsahu, který je velmi podobný existujícím dílům. To může být výzvou z hlediska ochrany duševního vlastnictví a etiky. Zneužití AI také představuje reálné riziko a může mít různé formy, jako například šíření dezinformací a sociální inženýrství. Následkem je problematika zabezpečení AI a právní regulace. Posledním často diskutovaným problémem je vliv a výskyt stereotypů v AI (Tiku et al., 2023).

Součástí práce je diskuse o etickém používání umělé inteligence a problému zodpovědnosti za vygenerovaný obsah. Diskuse se nezaměřuje pouze na technické hledisko, ale také na širší kontext sociálních a právních aspektů, které vznikají v důsledku rozšíření umělé inteligence do různých oblastí společnosti. V rámci této analýzy se zkoumá etický rámec, který by měl být uplatňován při vývoji, nasazování a využívání systémů umělé inteligence. Identifikují se klíčové otázky, které se týkají zachování lidských práv, spravedlnosti a transparentnosti při implementaci těchto technologií. Kromě toho je věnována pozornost problematickým aspektům spojeným s odpovědností za výsledky generované umělou inteligencí a tomu, jakým způsobem lze zabezpečit, aby subjekty, které používají tyto systémy, nesly odpovědnost za možná rizika a nežádoucí důsledky, které mohou vzniknout v důsledku jejich činnosti.

V souladu s tématem práce byla použita umělá inteligence pro navržení základních bodů teoretického podkladu. Veškeré takto získané informace byly zkontrolovány a ověřeny. Vygenerované body sloužily jako základ teoretického podkladu, který byl následně doplněn kompletně lidsky psaným textem, který je více specifický

a odkazuje se na další odborné práce. AI je také v práci použito pro generování různých obrázků. U všech vygenerovaných obrázků je uvedeno že obrázek byl vygenerován a jako zdroj je uveden nástroj, který byl pro generování použit. AI není použito pro generování diskuse, doporučení, a závěru. Pro tyto potřeby byla použita kombinace nástrojů Google Gemini, základní verze Microsoft Copilot, a základní i placená plus verze OpenAI ChatGPT. Nástroje byly použity v rozmezí od 1.11.2023 do 20.4.2024.

1.1 Umělá inteligence

Umělá inteligence představuje multidisciplinární obor informačních technologií, který se zaměřuje na vývoj a implementaci algoritmů a systémů schopných simulovat kognitivní schopnosti lidského myšlení. Tato disciplína se opírá o metodologie a koncepty z oblastí počítačové vědy, statistiky, matematiky, kybernetiky a kognitivní psychologie. Hlavním cílem umělé inteligence je vytvářet počítačové systémy schopné samostatného učení, adaptace k novým situacím a autonomního rozhodování na základě vstupních dat. Jedná se o realizaci inteligentního chování v kontextu strojového zpracování informací, kde jsou klíčovými prvky rozpoznávání vzorů, rozvoj znalostí, plánování, předvídání a interakce s prostředím (*What is AI?*, 2023).

Umělá inteligence může být rozdělena do několika kategorií, včetně slabé a silné umělé inteligence. Slabá umělá inteligence se specializuje na řešení konkrétních úkolů a není schopna samostatného myšlení mimo daný rámec, zatímco silná umělá inteligence aspiruje na dosažení úrovně lidské inteligence s možností obecného učení a flexibilního přizpůsobení se různorodým úkolům (Mills, 2018). Aktuálně neexistuje žádná silná umělá inteligence ve smyslu, jak je někdy prezentována ve sci-fi literatuře nebo filmech (Altman, 2023). Silná umělá inteligence by byla schopná chápat kontext širokého spektra úkolů na úrovni lidské inteligence a byla by flexibilní v řešení nových problémů bez nutnosti předchozího specifického programování.

V oblasti umělé inteligence jsou v současné době významně využívány metody hlubokého učení (deep learning), které se opírají o neuronové sítě s několika vrstvami a schopností automatické extrakce a abstrakce vzorů. AI nachází aplikaci v různých odvětvích, včetně autonomních vozidel, diagnostiky nemocí, obchodní analýzy, jazykového zpracování, a mnoha dalších, přičemž její vývoj je stále v průběhu intenzivního zkoumání a inovací (*What Is Deep Learning?*, b.r.). Za posledních pár let se vyskytl velký pokrok v oblasti generativní umělé inteligence (Zewe, 2023). Generativní umělá inteligence představuje typ umělé inteligence, který využívá generativní modely k vytváření obsahu, data nebo informací. Tato paradigmatická přístupová metoda využívá neuronové sítě, konkrétně generativní modely založené na hlubokém učení, k produkci nových a autentických dat, která jsou v souladu s vzory a charakteristikami, které byly natrénovány na předchozích datech. Generativní modely vytvářejí obsah prostřednictvím učení ze statistických struktur a vzorů v datech, která byla použita k jejich tréninku (Zewe, 2023). Mezi příklady generativních modelů patří variabilní autoenkodéry (variational autoencoder) generativní kontrastní sítě (generative adversarial network) nebo rekurentní neuronové sítě (recurrent neural network). Generativní umělá inteligence má široké uplatnění v různých oblastech, včetně tvorby uměleckých děl, syntézy obrazů, rozpoznávání vzorů, tvorby textu a dalších úloh spojených s produkci obsahu či dat (Marr, 2023b).

1.2 Strojové učení a neuronové sítě

Strojové učení (machine learning) je disciplína umělé inteligence, která se zabývá výzkumem a vývojem algoritmů a modelů, umožňujících počítačům získávat schopnost učit se a zlepšovat své výkony na základě zkušeností získaných ze vstupních dat. Klíčovým cílem strojového učení je vytvořit systémy, které jsou schopny automaticky odhalovat vzory, identifikovat trendy a adaptovat se na nové informace bez explicitního programování (Brown, 2021). Strojové učení se dělí do několika kategorií, jako jsou učení s učitelem (supervised learning), učení bez učitele (unsupervised learning) a učení s částečným učitelem (semi-supervised learning). V učení s učitelem model přijímá označená trénovací data, zatímco u učení bez učitele se model učí z neoznačených dat a hledá skryté struktury (Brown, 2021).

Vektory hrají klíčovou roli jako reprezentace dat, které umožňuje efektivní analýzu a manipulaci informací. Vektorová reprezentace dat je základem mnoha pokročilých technik strojového učení a AI, přičemž vektory slouží k zakódování informací o objektech, jevech či konceptech do multidimenzionálních prostorů (Mikolov et al., 2013). Vektorové reprezentace mají v oblasti databází umělé inteligence několik klíčových funkcí. První z nich spočívá v efektivním zachycení sémantických vztahů mezi datovými entitami. Vektory umožňují převést abstraktní a kontextuální informace do matematické formy, což usnadňuje porovnávání a analýzu v rámci algoritmů strojového učení (Mikolov et al., 2013). Dále jsou vektory využívány pro reprezentaci výsledků předtrénovaných modelů. Například vektorové modely jako BERT nebo Word2Vec od českého výzkumníka Ing. Tomáše Mikolova, Ph.D. jsou schopny transformovat slova nebo fráze do vektorových reprezentací, což umožňuje modelům lépe porozumět sémantickým vztahům mezi slovy a větami (Rong, 2016). Tyto vektorové reprezentace jsou následně integrovány do databází umělé inteligence pro zlepšení schopnosti modelů porozumět a odpovídat na dotazy.

Vektory také usnadňují práci s velkým množstvím dat a umožňují rychlejší vyhledávání, kategorizaci a shlukování informací (Mikolov et al., 2013). Díky tomu jsou v databázích umělé inteligence vektory klíčovým prvkem pro optimalizaci operací spojených s analýzou dat a umožňují efektivní zpracování a interpretaci informací ze složitých a rozsáhlých datasetů. S tím souvisí tensor, což je matematický objekt, který generalizuje koncepci vektoru a matice na vyšší dimenze. V oblasti strojového učení učení jsou tenzory základní datovou strukturou pro reprezentaci dat v různých podobách. Základní typy tenzorů zahrnují skaláry (tenzory řádu 0), vektory (tenzory řádu 1), matice (tenzory řádu 2) a vyšší dimenze tenzorů (Aslam, 2022).

V rámci strojového učení zaujímají významné místo neuronové sítě (neural networks), což jsou matematické modely inspirované strukturou a fungováním lidského mozku. Tyto sítě se skládají z umělých neuronů propojených váhami, kde každý neuron přijímá vstupy, provádí váhovou sumaci a aplikuje aktivační funkci pro generování výstupu. Tento proces umožňuje síti generalizovat a adaptovat se na nová data (Fumo,

2017). Neurony tvoří vrstvy, a sice vstupní vrstvu, skryté vrstvy a výstupní vrstvu. Vstupní vrstva přijímá externí data, zatímco skryté vrstvy provádějí komplexní výpočty a výstupní vrstva produkuje konečné výsledky. Skrze iterativní proces trénování jsou váhy v síti aktualizovány na základě rozdílu mezi predikcemi modelu a skutečnými hodnotami v trénovacích datech (Hardesty, 2017). Oblast neuronových sítí se ve strojovém učení je často označována jako hluboké učení (deep learning). Tento název vychází z počtu skrytých vrstev neuronových sítí, kde vysoký počet těchto vrstev naznačuje hloubku neuronové sítě.

Neuronové sítě se v praxi uplatňují v širokém spektru aplikací, včetně rozpoznávání obrazu, přirozeného jazykového zpracování, automatizovaného rozhodování a mnoha dalších úloh, kde je vyžadována schopnost komplexního učení a vzorového rozpoznávání (Fumo, 2017). Jejich vývoj a zdokonalování představují důležitou oblast výzkumu v rámci umělé inteligence. Struktura neuronových sítí je rozdělena na 3 části, a to vstupní vrstvu, skryté vrstvy a výstupní vrstvu. Vstupní vrstva přijímá vstupní data do neuronové sítě. Každý neuron v této vrstvě reprezentuje jednu dimenzi vstupního datasetu. Výstupní vrstva generuje výsledné informace nebo predikce. Každý neuron v této vrstvě odpovídá jednomu výstupu modelu. Skryté vrstvy jsou prostřední vrstvy mezi vstupní a výstupní vrstvou. Tyto vrstvy provádějí transformace na vstupní data pomocí vah a aktivačních funkcí. Čím více skrytých vrstev, tím složitější vzory a hierarchie mohou být modelovány (Hardesty, 2017). Nejběžnější skryté vrstvy jsou plně propojené vrstvy, konvoluční vrstvy a rekurentní vrstvy.

V plně propojených vrstvách jsou všechny neurony propojeny s každým neuronem ve vrstvě předchozí a následující, což umožňuje efektivní učení vzorů v datech. Konvoluční vrstvy jsou často používány ve zpracování obrazu. Tyto vrstvy pracují s malými regiony vstupních dat, které se přesouvají přes celý obrázek pomocí filtrů. Tím umožňují modelu zachytit vzory a vlastnosti, jako jsou hrany a textury. Rekurentní vrstvy jsou navrženy pro práci se sekvencemi dat, jako jsou časové řady nebo přirozený jazyk. Tyto vrstvy mají schopnost udržovat vnitřní stav, což jim umožňuje pracovat s historií vstupů (Fumo, 2017). Existuje mnoho dalších typů vrstev, které

většinou podporují již zmíněné vrstvy slouží například k transformaci dat do jiného formátu nebo zabraňují přeučení. Aktivační funkce jsou typicky aplikovány na vážený součet vstupů neuronu. Tato transformace přidává nelinearitu do sítě, což umožňuje síti aproximovat složité, nelineární vztahy v datech. V případě hlubokých neuronových sítí je existence aktivačních funkcí klíčová pro schopnost modelu efektivně reprezentovat abstraktní a hierarchické rysy v datech.

1.3 Velké jazykové modely

Velké jazykové modely (Large Language Models, dále v textu jen LLM) je termín v oblasti umělé inteligence, konkrétně v kontextu s pojmem zpracování přirozeného jazyka (natural language processing, zkrácně NLP), který označuje model, jehož rozměry a parametry jsou výrazně nadstandardní a obvykle představují velký počet parametrů. LLM slouží k modelování a porozumění jazykových struktur a vzorů v obrovském rozsahu dat. Tyto modely jsou často postaveny na hlubokých neuronových sítích, které mají schopnost efektivně zpracovávat složité jazykové vzory a zachycovat významy v kontextu (Kerner, 2023). Základem LLM je takzvaný základní model (foundation model), což je koncept v oblasti umělé inteligence, který označuje velký, silně předškolený model, který slouží jako základ pro řadu úkolů a aplikací. Tento model je trénován na obrovském množství dat a má schopnost zachytit obecné vzory a struktury v informacích (Kerner, 2023).

Významným rysem LLM je jeho schopnost generovat lidsky podobný text, rozumět kontextu a adaptovat se na různé jazykové úlohy, jako jsou strojový překlad, automatické generování textu nebo odpovídání na otázky. Tyto modely jsou často předtrénovány na rozsáhlých korpusech jazykových dat, což umožňuje přenos jejich schopností na specifické úlohy bez potřeby detailního ručního ladění. Využití LLM se nachází v široké škále aplikací, včetně tvorby obsahu, chatbotů, analýzy sentimentu a dalších oblastí, kde je potřeba sofistikovaného porozumění a generace přirozeného jazyka (Kerner, 2023). Přestože LLM přinášejí významný pokrok v oblasti NLP, současně se s nimi pojí otázky týkající se etiky, transparentnosti a bezpečnosti, které vyžadují pozornost při jejich implementaci a využití.

S velkými jazykovými modely také souvisí pojem transformer, neboli transformátor, což je inovativní model v oblasti umělé inteligence, který představuje zlomový pokrok v oblasti zpracování přirozeného jazyka a dalších úloh spojených s sekvencemi dat. Byl poprvé představen v článku s názvem "Attention is All You Need" publikovaném v roce 2017 autory Ashish Vaswani a kolegy (Vaswani et al., 2023). Tento model vyniká svou schopností efektivně zpracovávat dlouhé sekvence a zachovávat vzájemné vztahy mezi vzdálenými částmi vstupních dat. Transformátor využívá mechanismus tzv. attention, neboli pozornosti, který umožňuje modelu selektivně se zaměřit na různé části vstupu při generování výstupu. Tím se dosahuje větší flexibility a schopnosti zachycovat kontextuální informace na delší vzdálenosti, což představuje zlepšení oproti předchozím rekurentním a konvolučním architekturám (Vaswani et al., 2023).

Transformer má hierarchickou strukturu skládající se z enkodéru a dekodéru, přičemž každý z těchto bloků obsahuje několik vrstev. Každá vrstva zahrnuje mechanismus attention a takzvanou feedforward síť, která zpracovává informace na každém úrovni. Tato struktura umožňuje transformeru dosahovat vysoké paralelizace a efektivně zpracovávat i velmi rozsáhlá data (Vaswani et al., 2023). Transformer se rychle stal klíčovým prvkem v mnoha pokročilých systémech pro zpracování přirozeného jazyka, strojového překladu, generativního modelování a další úlohy spojené se sekvencemi. Jeho inovativní architektura otevřela cestu k řadě úspěšných aplikací v oblasti umělé inteligence a posílila výzkumné úsilí směřující k dalšímu zdokonalování modelů založených na principu transformeru (Toews, 2023).

Generativní předtrénovaný transformátor (Generative Pre-trained Transformer, dále v textu jen jako GPT) představují rodinu špičkových jazykových modelů, založených na transformační architektuře neuronových sítí. Tato rodina modelů, vyvinutá společností OpenAI, se vyznačuje výraznou schopností porozumění a generace přirozeného jazyka na základě obecného předtrénování na rozsáhlých korpusech textových dat. Jak už vyplývá z názvu, GPT využívá mechanismu transformeru, který umožňuje efektivní zpracování dlouhých sekvencí dat

a zachycení vzorů v kontextu (Radford et al., 2018). Předtrénování modelů GPT spočívá v expozičním učení, kdy jsou modely vystaveny rozsáhlým textovým datům bez specifického zadání úlohy. Tato fáze předtrénování umožňuje modelům naučit se jazykové vzory, syntaktické struktury a významové vztahy v textu (Shree, 2020).

Před zahájením předtrénování jsou váhy modelu náhodně inicializovány. Následně je model vystaven rozsáhlým textovým sadám, kde je v každém kroku výcviku vyzván k odhadu následujícího slova nebo fragmentu textu vycházejícího z kontextu. Tato úloha, známá jako maskovaná jazyková modelace, obvykle zahrnuje náhodné maskování určitých slov věty a model je vyzýván k odhadu jejich původní podoby. Během tohoto předtrénování se model postupně adaptuje na strukturu a vzory obsažené v datech. To mu umožňuje vyvinout schopnost porozumět gramatice, slovníkům a kontextu v širokém rozsahu jazyků a témat. Samotný proces předtrénování může vyžadovat několik dní, s nutností využití výkonných výpočetních zdrojů (Ghosh, 2023). Po dokončení fáze předtrénování model může podstoupit další ladění nebo modifikace pro specifické úlohy či oblasti. Tento postup, označovaný jako jemné naladění, vytváří konkrétní využití modelu pro konkrétní aplikace, čímž se získává schopnost generovat lidsky srozumitelný text, odpovídat na otázky a zvládat různé jazykové úkoly (Ghosh, 2023).

Modely GPT jsou schopny generovat koherentní a kontextově relevantní odpovědi na základě zadaných vstupních otázek nebo kontextu. Díky své flexibilitě a obecnosti jsou GPT vhodné pro širokou škálu úloh v přirozeném jazykovém zpracování, včetně strojového překladu, generativního psaní, dialogových systémů a dalších aplikací (Waqas, 2023). Klíčovou vlastností modelů GPT je schopnost přenášet své jazykové schopnosti na specifické úlohy bez potřeby velkého množství trénovacích dat pro každou konkrétní úlohu (Radford et al., 2018). To činí GPT významným hráčem v oblasti přirozeného jazykového zpracování, přinášejícím inovace a dosahujícím výrazného pokroku v oblasti strojové inteligence. GPT modely jsou schopny přenášet své schopnosti na různé úlohy v přirozeném jazykovém zpracování.

Neuronové sítě a modely jako GPT vyžadují výpočetní sílu pro trénování a odvozování. Hardware používaný pro tyto účely závisí na velikosti modelu, rozsahu dat a potřebách konkrétního úkolu (Ibrahim, 2021). Pro trénování neuronových sítí jsou často používány grafické karty (Graphics Processing Unit, dále v textu jen jako GPU). Moderní GPU jsou schopny paralelního zpracování a jsou velmi efektivní pro operace s velkými tenzory, což umožňuje rychlé trénování modelů (Ibrahim, 2021). Takzvané TPU (Tensor Processing Unit) jsou specializované čipy vyvinuté společností Google speciálně pro práci s tenzory a urychlení operací souvisejících s hlubokým učením. Jsou široce využívány pro trénování a inference modelů, včetně GPT (Ibrahim, 2021). Některé organizace vyvíjejí vlastní čipy Zákaznický integrovaný obvod (Application-Specific Integrated Circuit, dále v textu jen jako ASIC), které jsou optimalizovány pro konkrétní úkoly v oblasti strojového učení. Tyto čipy mohou být navrženy tak, aby byly ještě efektivnější než obecné GPU nebo TPU (Hui, 2020). Programovatelná hradlová pole (Field-Programmable Gate Array, dále v textu jen jako FPGA) jsou programovatelné čipy, které mohou být přizpůsobeny pro konkrétní úlohy. I když nejsou tak efektivní jako speciálně navržené ASIC nebo TPU, FPGA mohou být použity pro různé úlohy včetně strojového učení. Mnoho organizací a výzkumných institucí využívá cloudové služby pro výpočetní potřeby spojené s trénováním a inference modelů. Cloudoví poskytovatelé nabízejí širokou škálu instancí s různým hardwarem, včetně GPU a TPU (Hui, 2020).

1.4 Využití AI

Uplatnění umělé inteligence je v dnešní době rozsáhlé a rozmanité. Tyto technologie, založené na pokrocích v oblasti strojového učení a zpracování přirozeného jazyka, nacházejí aplikace v různých odvětvích a disciplínách. V sektoru zákaznického servisu a podpory AI umožňuje automatizované reakce na běžné dotazy zákazníků, řeší problémy a naviguje uživatele k relevantním informacím, čímž významně zlepšuje efektivitu zákaznického servisu (Marr, 2023b). V oblasti e-obchodu a marketingu se AI využívá k personalizaci zákaznických zkušeností, doporučování produktů na základě individuálních preferencí a historie prohlížení, a také k interaktivní komunikaci s potenciálními zákazníky. AI také pomáhá v diagnostice

a léčbě nemocí. Například, algoritmy strojového učení mohou analyzovat obrazová data z lékařských skenů a identifikovat známky nemoci, které by mohly být pro lidské oči obtížně rozpoznatelné (Marr, 2023b). V oblasti vzdělávání a školení je AI využívána k interaktivnímu výukovému prostředí, které pomáhá studentům porozumět složitým konceptům a poskytuje rychlé odpovědi na jejich otázky. V bankovnictví a finančních službách se AI stává nástrojem pro finanční poradenství, sledování tržních trendů a provádění transakcí na základě instrukcí uživatele (Marr, 2023b). Průmyslové aplikace umělé inteligence zahrnují monitorování a řízení výrobních procesů, sledování výrobních parametrů a detekci možných poruch. Ve vědeckém výzkumu AI slouží k analýze dat, tvorbě modelů a automatizaci rutinních úkolů, což vede k zvýšení produktivity a objevování nových poznatků (Marr, 2023b).

AI může vytvářet nové skladby v různých hudebních žánrech. Algoritmy strojového učení mohou být natrénovány na velkých datasetech existující hudby a poté generovat nové melodie, harmonie a rytmy. Tato technologie může být použita například pro vytváření filmové hudby nebo pro generování unikátních zvukových stop pro videohry (Henkin, 2023). AI může také generovat vizuální umělecká díla. Například, technologie jako GAN mohou vytvářet realistické obrazy, které jsou těžko rozeznatelné od skutečných fotografií. AI může také vytvářet abstraktní umění nebo dokonce napodobovat styly známých umělců (Roose, 2022). AI dokáže generovat texty, včetně básní, povídek, a dokonce celých románů. Tyto systémy mohou být natrénovány na textech konkrétních autorů nebo literárních žánrů a poté generovat nové texty, které napodobují daný styl (Marr, 2023b).

ChatGPT je model umělé inteligence vyvinutý firmou OpenAI, který spadá do rodiny GPT. Jedná se o konkrétní variantu, která je navržena pro dialogové interakce s uživateli. Model byl zveřejněn v různých iteracích s postupujícím vylepšováním schopností a výkonu. ChatGPT využívá architekturu Transformer, která umožňuje efektivní zpracování a generování textu na základě kontextuálních informací. Díky předtrénování na velkém množství internetových dat dokázal model zachytit různorodé jazykové vzory, syntaxi a sémantiku. Jeho schopnost generovat plynulé a kontextuálně relevantní odpovědi ho činí vhodným pro různé úlohy dialogového

systému (Marr, 2023a). Při interakci s uživatelem model analyzuje předchozí text a generuje odpovědi s ohledem na kontext a relevanci informací. Vlivem svého předtrénování je schopen obsahově a stylisticky odpovídat na různé dotazy a komentáře. Nicméně je třeba zdůraznit, že ChatGPT má své omezení, jako například sklon k produkci někdy nepřesných nebo fantazijních informací, což vyplývá z charakteru dat použitých při předtrénování. OpenAI se snaží neustále zdokonalovat modely, aby dosáhly vyšší přesnosti a použitelnosti v reálném světě (OpenAI, 2022).

V generativním modelu, jako je ChatGPT, je následující slovo generováno na základě předchozího kontextu, který zahrnuje jak vstup od uživatele, tak i předchozí slova vygenerovaná modelem. To znamená, že při generování každého nového slova model bere v úvahu celý kontext dialogu a všechny předchozí slova, která sám vygeneroval. Tento přístup umožňuje modelu vytvářet plynulé a souvislé odpovědi, které jsou v souladu s kontextem dialogu a předchozími větami (Ruby, 2023). Zjednodušeně by se dalo říct, že tento typ AI myslí nahlas. Prémiová verze využívají novější model GPT-4 je takzvaně multimodální, což umožňuje zpracovat kontext v různých formátech jako je text, obraz nebo i zvuk. Tato verze také zahrnuje další nástroje jako je DALL-E pro generování obrazu a umožňuje chatbotu přistupovat na internet.

Aktuálně k datu 01.03.2024 jsou dvě největší alternativy k ChatGPT Microsoft Copilot od společnosti Microsoft a Gemini od společnosti Google. Microsoft Copilot je inzerován spíše jako digitální asistent a je propojen s ostatním kancelářským softwarem od Microsoftu. Copilot je založený na modelech GPT a pracuje podobně jako ChatGPT. Na rozdíl od ChatGPT ale Copilot neexceluje v generování textu přirozeného jazyka a jeho odpovědi jsou tak více stručné. Stejně jako ChatGPT dokáže Microsoft Copilot generovat obraz pomocí DALL-E (Ciornei, 2024). Podoba nástrojů není náhodná, jelikož Microsoft vlastní 49% společnosti OpenAI (Weise, 2023). Gemini, dříve nazývaný Bard, je také multimodální model, který však není založený na modelech GPT ale na interně vyvinutém modelu od společnosti Google. Podobně jako Copilot má Gemini přístup na internet. Svojí funkcionalitou se ale spíš podobá nástroji ChatGPT (Ciornei, 2024). Existuje mnoho dalších nástrojů jako například

Perplexity AI, které využívá několik různých modelů jako GPT-4 se svým vlastním modelem.

Nejrozšířenější modely pro generování obrazu jsou Stable Diffusion, Midjourney a DALL-E. Kromě generování nového obrazu jsou tyto modely také schopné upravovat existující obrazy. Stable Diffusion je open-source model založený na principu difuze, DALL-E od OpenAI je založen na modelu GPT-3 a je dostupný jakou součást ChatGPT a Microsoft Copilot, a Midjourney je nezávislý projekt, který je stále v beta-verzi a dostupný pomocí komunitní platformy Discord (McFarland, 2024). Všechny tři modely jsou schopné generovat obraz až v rozlišení 1024x1024 (Hanna, 2023). Obrázek 1 je příklad vygenerovaného obrázku pomocí modelu DALL-E v nástroji Microsoft Copilot. Sora je nejnovější generativní AI model od OpenAI, který umožňuje generovat realistická a kreativní videa z textových popisů. Model se stále vyvíjí, ale již teď je schopen vytvářet videa s rozlišením 1024x768 pixelů a délkou až 60 sekund (OpenAI, 2024). Tento model zatím není dostupný pro veřejnost. Pro generování zvuku a hudby také existuje mnoho nástrojů jako například Jukebox od OpenAI.

Obrázek 1 Příklad vygenerovaného obrázku

Zdroj: vlastní, vygenerováno pomocí nástroje DALL-E

Generativní AI také představuje revoluční nástroj v oboru informačních technologií, obzvláště v kontextu softwarového inženýrství. Tato technologie je schopna nejenom generovat zdrojový kód, ale i dotazy optimalizované pro databázové systémy a příkazy pro efektivní ovládání operačních systémů. Tímto způsobem může výrazně usnadnit proces vývoje softwaru a správu IT infrastruktury (Wong et al., 2023). V oblasti softwarového inženýrství umožňuje generativní AI automatizovat části vývojového procesu, čímž zkracuje dobu potřebnou k implementaci nových funkcí nebo opravám chyb. Generování zdrojového kódu podle specifikací umožňuje rychle vytvářet části aplikací a snižovat zátěž

na vývojáře. Podobně, generování optimalizovaných dotazů pro databáze pomáhá zlepšovat výkon aplikací pracujících s velkými datovými sadami (Wong et al., 2023).

V oblasti správy IT infrastruktury může generativní AI generovat příkazy pro konfiguraci a monitorování systémů, což zvyšuje efektivitu provozu a snižuje riziko lidských chyb. Například v cloudovém prostředí může generování příkazů pro nasazení a škálování aplikací zjednodušit proces správy cloudových služeb. Nicméně, přes všechny tyto výhody, je třeba zdůraznit výzvy spojené s používáním generativní AI v praxi. Jednou z klíčových otázek je zajištění správnosti a bezpečnosti generovaného kódu či příkazů, aby nedošlo k vytvoření zranitelných či nefunkčních systémů. Další výzvou je správná interpretace a využití generovaných výstupů v kontextu konkrétních požadavků a specifikací projektu (Kelly, 2023).

S obřím růstem v oblasti AI tak vznikla nová disciplína nazývaná prompt inženýrství (prompt engineering). Jedná se o metodologii navrhování efektivních požadavků, tzv. promptů, pro komunikaci s velkými jazykovými modely. LLM disponují impozantní schopností generovat text, překládat jazyky, psát různé druhy kreativního obsahu a zodpovídat otázky komplexním způsobem. Nicméně, jejich plný potenciál se rozvíjí až s precizním zadáním úkolu v podobě propracovaného promptu (Pawlan, 2023). Cílem prompt inženýrství je vytvořit prompt, který jasně a srozumitelně definuje požadovaný úkol a maximalizuje šanci na dosažení požadovaného výstupu. Prompt by měl detailně specifikovat požadovaný úkol, včetně požadovaného formátu a stylu výstupu. Poskytnutí relevantních příkladů a referencí LLM usnadňuje pochopení zadaného úkolu a požadovaného typu výstupu (Pawlan, 2023). Další metodou může být předladění daného nástroje. Tato strategie spočívá v tom, že LLM nejprve zpracuje sérii ladících promptů, které ho připraví na zpracování finálního promptu. Ladící prompty by měly být relevantní k finálnímu promptu a cíli.

1.5 Objektově orientované programování a AI

Objektově orientované programování (Object Oriented Programming, dále v textu jen jako OOP) je paradigmatický přístup k softwarovému návrhu a implementaci, který se

zaměřuje na modelování a organizaci systémů pomocí konceptu objektů. Objekty představují základní stavební bloky, které sdružují data a metody, jež s nimi pracují. Tato paradigmatická metoda umožňuje zacházet s komplexními systémy prostřednictvím konceptuálních entit, které odpovídají reálným objektům či abstraktním konceptům v daném doménovém kontextu (Gillis & Lewis, 2021). Základními principy objektového programování jsou zapouzdření, dědičnost a polymorfismus. Zapouzdření umožňuje skrýt interní implementaci objektu a odkrývat pouze nezbytně nutné rozhraní. Dědičnost umožňuje vytvářet hierarchie tříd, kde odvozené třídy mohou přebírat zděděné vlastnosti a metody z předka. Polymorfismus pak umožňuje používat stejné rozhraní pro různé typy objektů, což zvyšuje flexibilitu a znovupoužitelnost kódu. V objektovém programování je důraz kladen na návrhové vzory, které poskytují osvědčené postupy pro řešení častých problémů. Tyto vzory napomáhají k vytváření modulárního a udržitelného kódu (Gillis & Lewis, 2021).

Generování zdrojového kódu pomocí umělé inteligence je proces, ve kterém strojový model, jako například GPT-4, vytváří programový kód na základě daných vstupů nebo instrukcí. Tento postup vychází z rozsáhlého tréninku modelu na obrovském množství existujícího kódu, což mu umožňuje pochopit syntaxi, sémantiku a strukturu programovacích jazyků. Při generování zdrojového kódu se model snaží vytvořit kód, který odpovídá specifikacím daným uživatelem. Model kombinuje své znalosti o programovacích paradigmatech, algoritmech a datových strukturách s cílem produkovat funkční a efektivní kód. V průběhu tohoto procesu model nejenom reprodukuje syntaktické struktury, ale také se snaží zachytit zamýšlenou funkcionalitu (Ruiz, 2024).

Základní krok generování zdrojového kódu zahrnuje interpretaci vstupních instrukcí, které mohou být poskytnuty v textové formě. Model následně kombinuje tuto informaci s jeho vnitřním lexikonem a znalostmi o konkrétním programovacím jazyku. Během tohoto procesu se uplatňují metody generativních mechanismů, jako jsou rekurentní neuronové sítě a transformery, které umožňují modelu efektivně predikovat následující kroky v kódu na základě již vygenerované části. Výstupem

generování kódu je pak soubor, který obsahuje kód, přizpůsobený specifikacím a požadavkům uživatele. Je však třeba poznamenat, že navzdory schopnosti generovat kvalitní kód má model omezenou schopnost rozumět obecným konceptům a kontextům. Je důležité provádět pečlivou revizi a testování vygenerovaného kódu, aby byla zajištěna jeho správnost a bezpečnost (Ruiz, 2024).

ChatGPT je běžně používaný nástroj pro generování programového kódu, SQL dotazů a různých technických příkazů. Placený model GPT-4 i bezplatný model GPT-3.5 jsou schopné podle textových instrukcí vygenerovat kód v jakémkoliv programovacím jazyce včetně komentářů a vysvětlení funkce jednotlivých částí kódu (Bala & Ilic, 2023). Výhoda modelu GPT-4 je multimodalita, která umožňuje zadávat požadavky ve více formátech, zejména pomocí obrázků. To je efektivní například pro generování uživatelského rozhraní podle jednoduchého náčrtu. GPT-4 má také více parametrů a je to celkově modernější model, takže i bez využití multimodality by měly být vygenerované výsledky programového kódu kvalitnější (Coello et al., 2024). Podobně je tomu i u konkurenčních nástrojů jako Google Gemini a Microsoft Copilot. Oba nástroje jsou multimodální a schopné generovat zdrojový kód v několika programovacích jazycích. V současné době nejsou ale využívány ve stejné míře jako ChatGPT (Coello et al., 2024).

Codex je model umělé inteligence vyvinutý společností OpenAI, který se zaměřuje na analýzu přirozeného jazyka a generování kódu v reakci na zadané instrukce. Model byl natrénovaný na veřejně dostupném kódu z platformy GitHub a je založený na modelu GPT (Chen et al., 2021). Codex je dostupný v rámci platformy OpenAI API a lze ho integrovat do různých vývojových prostředí a nástrojů. Největším nástrojem založeným na modelu Codex je GitHub Copilot od OpenAI a společnosti GitHub, který pomáhá softwarovým vývojářům psát kód. Narozdíl od obecných AI nástrojů jako ChatGPT je GitHub Copilot speciálně navržený pro programovaní. Funguje jako zásuvný modul pro editory kódu, jako je Visual Studio Code, a nabízí návrhy kódu v reálném čase na základě kontextu vašeho kódu. Podle provedené studie dokázala skupina programátorů s přístupem k tomuto nástroji dokončit úkol o 55,8% rychleji (Peng et al., 2023). Dalším populárním nástrojem je Tabnine, který je založený

na vlastním proprietárním modelu. Výhodou tohoto nástroje je částečná off-line funkcionalita a víc možností přizpůsobení pro různé požadavky velkých podniků (Roshelova, 2023). Amazon CodeWhisperer je další populární nástroj, který je silně integrován do Amazon Web Services cloud computing ecosystému. Jeho použití mimo tento ecosystém jsou ale znatelně omezené (Pandey, 2023).

1.6 Technologická nezaměstnanost

Technologická nezaměstnanost (technological unemployment) je termín pro ztrátu zaměstnání z důvodů technologického pokroku (Campa, 2018). Před vynálezem tiskového lisu museli lidé ručně vyrábět knihy a kopírovat texty. Tiskový lis, vynalezený Johannesem Gutenbergem v 15. století, revolučně zvýšil rychlost a efektivitu produkce knih a dokumentů. S rozvojem telegrafu a poté telefonu byla komunikace usnadněna a zrychlena. Moderní technologie, jako jsou e-maily a mobilní telefony, postupně nahradily potřebu fyzické distribuce zpráv. Před rozvojem motorových vozidel a železnic byla doprava závislá na kočárech tažených zvířaty a lidské síle. Automobily, vlaky a letadla postupně převzaly dopravní potřeby, čímž změnily povahu práce v této oblasti. Přesnější definice slova počítač (computer) se začala formovat v raném 17. století, kdy bylo používáno pro popis lidí, kteří prováděli matematické výpočty nebo matematické operace. Později, v průběhu 20. století, se termín počítač začal používat pro označení elektronických zařízení, která provádějí programovatelné výpočty. S nástupem počítačů a digitálních technologií se mnoho manuálních úloh, jako je ruční počítání a kalkulace, stalo zbytečným. Elektronické tabulky a programy pro správu dat nahradily potřebu lidského počtu. Moderní výrobní technologie, jako je 3D tisk a automatizace, změnily výrobní procesy a vedly ke snížení potřeby ruční práce v některých odvětvích. Před příchodem automatických telefonních systémů museli telefonní operátoři ručně přepojovat hovory. Digitalizace a automatizace telefonních sítí způsobily, že tato profese ztratila svůj význam. Digitální fotografie a digitální zpracování obrazu postupně nahradily filmovou fotografii a ruční zpracování fotografií.

Generativní umělá inteligence má možnost částečně nebo kompletně nahradit několik typů zaměstnání a je možné že programování bude jednou z nich (Marr, 2023c). Z toho plynou otázky: jak umělá inteligence ovlivní vzdělávání v oboru aplikované informatiky a programování, jak efektivně dokáže umělá inteligence nahradit lidský kód, jaký bude dopad umělé inteligence na vzdělávací systém a zda je studium tohoto oboru s příchodem umělé inteligence nepotřebné.

1.7 Výzkumná otázka

Cílem práce je otestovat efektivitu nástroje ChatGPT s modelem GPT-4 při generování zdrojového kódu pro objektově orientované programovací úlohy. GPT-4 je aktuálně jeden z nejnovějších generativních modelů který je široce používány v několika oblastech včetně programování. V praxi je ale používán odborníky jako nástroj, který rozšiřuje jejich dosavadní schopnosti a usnadňuje jejich práci (Roller, 2023). Možnou otázkou tedy je, zda jsou technické znalosti v oblasti programování opravdu potřebné pro využití tohoto nástroje pro programování. Výsledkem tak bude zjištění, zda je ChatGPT nástroj pro programátora, nebo jeho náhrada.

2 Cíl a metodika práce

Cílem této práce je stanovit, jak efektivně dokáže model GPT-4 řešit objektově orientované programovací úlohy na vysokoškolské úrovni a na základě těchto poznatků a teoretického podkladu tak určit možný dopad umělé inteligence na budoucnost oboru aplikované informatiky a celého vzdělávacího systému. Hlavní výzkumnou otázkou této práce je tedy zjistit, zda dokáže nástroj ChatGPT s modelem GPT-4 adekvátně splnit základní objektově orientované programovací úlohy a zda jsou k tomu nutné odborné znalosti uživatele, který s modelem komunikuje.

První předpoklad je, že model GPT-4 dokáže efektivně vygenerovat zdrojový kód dle daného zadaní. Předpoklad je založený na výsledků testování předchozích verzí modelů GPT (Cipriano & Alves, 2023). Druhý předpoklad je, že přestože existují úlohy, pro které je nutná odborná komunikace, aby byl model nasměrován správným směrem, tak pro většinu úloh nejsou tyto znalosti nutné a je postačující pouze kopírování chybových hlášek do konverzace s chatbotem a obecná neodborná navigace modelu. Předpoklad je založený na předchozím testováním nástroje ChatGPT (Khan et al., 2023). Tento přístup byl částečně inspirován podobným experimentem, který byl proveden na portugalské univerzitě v portugalském jazyce a pomocí modelu GPT-3 (Cipriano & Alves, 2023). Výzkumná část této práce se tak snaží z části replikovat výsledky tohoto článku, ale pomocí novějšího modelu GPT-4. Přestože je model GPT-3 stále k dispozici, je mnohem smysluplnější testovat nový model, jelikož je po všech stránkách lepší. Nejedná se o přesnou replikaci, jelikož v článku nejsou publikovány detaily provedeného výzkumu.

Výzkum práce je založený na testování efektivity umělé inteligence, a tak je AI v této části použito pro generování zdrojového kódu, který je následně analyzován a porovnáván s lidským kódem. Výsledek práce je částečně založen na výstupu AI, ale získané informace byly změřeny, zhodnoceny a popsány bez využití AI. Pro potřeby výzkumné části práce byla použita placená verze plus nástroje OpenAI ChatGPT s modelem GPT-4. Nástroj byl použit v rozmezí od 1.3.2023 do 7.4.2024.

2.1 Metodologie

Výzkum byl řešen kvalitativním charakterem porovnáním dvou způsobů objektivně orientovaného programování v jazyce Java. První způsob zahrnoval skupinu programátorů bez přístupu k nástrojům umělé inteligence. Tato skupina dostala zadání pro tři jednoduché objektivně orientované programovací úlohy a na základě těchto zadání jednotlivě programovala zadané úlohy. Zadání pro všechny úlohy bylo napsané v českém jazyce. Druhý způsob zahrnoval skupinu lidí, která nemá zkušenost s programováním. Úkolem druhé skupiny bylo vložení stejných programovacích zadání do modelu GPT-4 a vygenerování funkčního kódu. Následně byl zaznamenáván počet dalších potřebných upřesnění při komunikaci s chatbotem. Tyto upřesnění nevycházela z odborného technického vzdělání, ale pouze směrovala systém správným směrem na základě běžných observací. V případě vyskytnutí chyby při spuštění kódu byla modelu poskytnuta chybová hláška z vývojového prostředí. Každý jednotlivec z obou skupin odevzdal zdrojový kód pro každou úlohu. Také byl zaznamenán čas, který strávil nad každou úlohou. Čas byl měřen na sekundy od vytvoření projektu do jeho odevzdání. Členové druhé skupiny také odevzdali kompletní historii komunikace s chatbotem v rámci jednotlivých úloh. Členové obou skupin zpracovali všechny úlohy po sobě ve stejný den.

Pro komunikaci s modelem GPT-4 byl použit český jazyk, přestože je model efektivnější při komunikaci v anglickém jazyce než v jazyce českém. Předpoklad byl, že rozdíly budou minimální či dokonce nepřítomné. Tento předpoklad byl založený na technické zprávě modelu GPT-4 od společnosti OpenAi (OpenAI et al., 2023). Podle této technické zprávy došlo k výraznému zlepšení u modelu GPT-4 vůči starším modelům co se týče porozumění méně využívaných jazyků. Model GPT-4 by tak měl být více efektivní v českém jazyce než předešlé modely. Vliv vybraného jazyka by také neměl mít velký vliv na vygenerovaný kód. Při generování běžného textu může volba jazyka výrazně snížit kvalitu vygenerovaného textu.
Při generování zdrojového kódu by ale nemělo dojít k výrazným změnám v kvalitě, jelikož obecně je kód a jeho struktura převážně totožná ve všech přirozených jazycích. Také by nemělo dojít k chybám v oblasti struktury vět a skloňování, jelikož se ve zdrojovém kódu obecně nevyskytují věty. Tyto chyby by se mohly vyskytnout

v komentářích, které ale nemají efekt na funkcionalitu kódu. Komentáře i tak byly pozorovány a jejich kvalita byla poznamenána. Úvaha tedy je, že použitý jazyk při komunikaci s ChatGPT má vliv pouze na zadaní kontextu pro generování kódu, rozdíly by ale měly být minimální, pokud je zadaní úlohy správně přeloženo. Hlavním důvod volby českého jazyku byl, že je to rodný jazyk všech členů obou výzkumných skupin. Pokud by bylo zadání a komunikace s chatbotem provedena v anglickém jazyce, tak by úroveň anglického jazyka jednotlivých členů skupin hrála roli v efektivitě jejich práce a nástroje ChatGPT.

První skupina byla složená ze dvou členů, průměrný věk byl 25 let. Druhá skupina byla také složena ze dvou členů, průměrný vek byl 26 let. Pouze členové první skupiny měli odborné vzdělání v oboru programování. Členové druhé skupiny v minulosti již použily nástroj ChatGPT a jsou s ním seznámeni. Všichni členové druhé skupiny používaly během výzkumné práce nástroj ChatGPT pod stejným účtem. To by nemělo ovlivnit výsledky, jelikož si každý člen založil pro každou úlohu novou konverzaci a nástroj nebere v potaz minulé konverzace. Programování úloh bylo provedeno v jazyce Java a jako vývojové prostředí bylo použité prostředí IntelliJ IDEA od společnosti JetBrains které nabízí několik vlastností pro usnadnění práce při manuálním objektovém programování jako generace konstruktorů a metod pro získání a nastavení atributů. To je relevantní pro porovnání času mezi skupinami. Jako SDK (Software development kit) bylo použito Oracle OpenJDK 22 s výchozí úrovní jazyka. Nebyla použita žádná dodatečná rozšíření a pluginy. Jako model pro ChatGPT bude použit GPT4 ve výchozím nastavení.

První skupina pracovala s jednoduchými pravidly. Členové první skupiny měli za úkol manuálně naprogramovat jednotlivé úlohy bez použití jakýkoliv nástrojů umělé inteligence. Skupina měla přístup k internetu, a kromě nástrojů umělé inteligence mohli členové této skupiny použít cokoliv co běžně používají k programování. Dalším pravidlem bylo pracovat samostatně. Členové této skupiny nebyli úzce monitorováni, jelikož se nepředpokládalo že by potřebovali dodatečné instrukce pro dokončení výzkumné práce.

Přesné instrukce první skupiny byly následující:

Naprogramujte jednotlivé úlohy podle daných zadání bez využití jakéhokoliv nástroje umělé inteligence a bez pomoci jiné osoby v reálném čase. Stopkami měřte vyžadovaný čas pro každou úlohu od vytvoření projektu. Odevzdávat budete vytvořený zdrojový kód a čas potřebný pro zpracování každé úlohy. Jako vývojové prostředí použijte IntelliJ Idea a jako SDK použijte OpenJDK 22. Můžete používat internet. Po dokončení úlohy můžete začít pracovat na následující úloze. Všechny úlohy dokončete ve stejný den.

Před zahájením testování byla skupina neprogramátorů seznámena s vývojovým prostředí IntellIJ IDEA pro potřeby tohoto výzkumu. To zahrnovalo vytvoření nového projektu, spouštění programu, vytvoření nové třídy, a práci s konzolí. Konfigurace vývojového prostředí jako výběr SDK byla provedena předem. Tato skupina byla instruována vyřešit jednotlivá zadání výhradně pomocí nástroje ChatGPT a případnými chybovými hláškami z vývojového prostředí. Stejně jako u první skupiny bylo nutné úlohy vypracovat samostatně bez pomoci v reálném čase od jakékoliv jiné osoby. Členové této skupiny byli při práci úzce monitorováni, jelikož nejsou zvyklí na používání testovaných technologií a nedošlo tak k nejasnostem během výzkumu. Veškerá asistence, kterou členové této skupiny dostali během práce, zahrnovala převážně upřesnění popisu jejich práce. Skupina tedy nedostala žádnou nápovědu v oblasti samotného programování.

Přesné instrukce druhé skupiny byly následující:

Vygenerujte pomocí nástroje ChatGPT programový kód podle daných zadání. Použijte verzi GPT-4 a pro každou úlohu vytvořte novou konverzaci. Vygenerovaný kód zkopírujte do vývojové prostředí InteliJ Idea a pokuste se ho spustit. Zároveň stopkami měřte čas nutný pro zpracování jednotlivých úloh od založení projektu či zahájení konverzace s ChatGPT. Jakmile shledáte kód funkční a splňující dané požadavky podle výpisu do konzole, zaznamenejte naměřený čas a můžete se přesunout na další úlohu. Všechny úlohy dokončete ve stejný den. Odevzdat je nutné vygenerovaný zdrojový kód a čas strávený nad každou úlohou. Vaše konverzace s ChatGPT bude také analyzována.

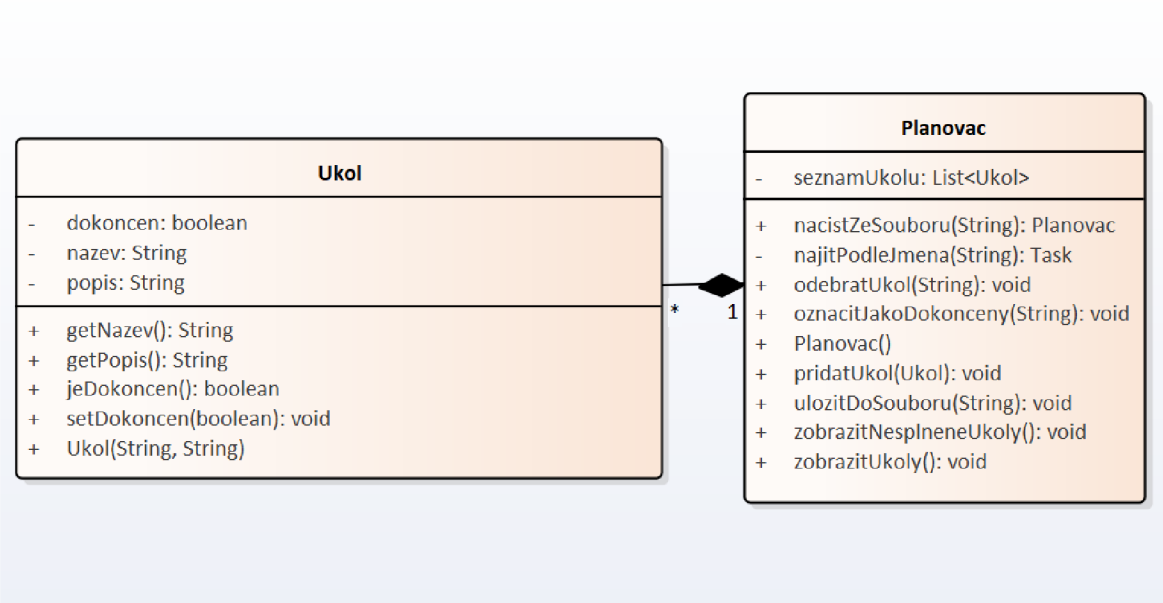
U každé úlohy byl porovnán minimální, maximální a průměrný čas nutný pro manuální programování dané úlohy ve skupině programátorů s minimálním, maximálním a průměrným časem, který byl nutný pro vygenerování kódu členy druhé skupiny pomocí ChatGPT. Je nutné poznamenat že čas není exaktní, jelikož je na každém jedinci rozhodnout, kdy je úloha dokončena. Dále byla měřena správná funkcionalita, struktura objektů, optimalizace a přehlednost u každého zdrojového kódu. Kromě kódu a času byla také analyzována komunikace členů druhé skupiny s nástrojem ChatGPT. Z pohledu principů OOP byl klad důraz na využití principů zapouzdření a abstrakce. Také byl poznamenán počet chyb v kódu. Funkcionalita, struktura, optimalizace a přehlednost byly jednotlivě hodnoceny čísly od 1 do 10, s tím že 10 bylo výchozí hodnocení, které bylo následně snižováno při určitých nedostatkách. Větší nedostatky mohly odpovídat většímu snížení hodnocení. Funkcionalitou byl hodnocen výsledek kódu po spuštění a zda splňuje zadané požadavky. Hodnocení struktury odpovídalo návrhu jednotlivých tříd, metod a využití principů OOP. Optimalizace značila rychlost a efektivitu kódu. Přehlednost zahrnovala názvosloví metod a atributů, kvalitu komentářů, a adekvátní počet metod. Tento způsob hodnocení byl zvolen jako nejvíce objektivní možnost pro prostředky této práce. Nejedná se však o perfektně a objektivně exaktní hodnocení.

2.2 Popis testovaných úloh

Testované programovací úlohy byly navrženy speciálně pro potřeby této práce. Jedná se o obecné a relativně jednoduché objektově orientované programovací úlohy. Záměrem bylo navrhnout úlohy, které nejsou zbytečně komplikované, ale zároveň jsou dostatečně rozsáhlé pro znatelné výsledky při porovnávání a analýzy kódu. Úlohy jsou zaměřené převážně na oblast objektového programování. Nevyžadují tak další komponenty jako je uživatelské rozhraní, nebo propojení s databází. Úlohy byly navrženy tak,aby výsledné aplikace byly funkcionálně kompletní a potenciálně využitelné v praxi. Úlohy byly navrženy tak, že by jejich jednotlivé splnění nemělo trvat déle než hodinu.

2.2.1 Úloha 1 – Plánovač aktivit

Zadání pro první úlohu s názvem Plánovač aktivit bylo navrženo jako jednoduché zadání s cílem určit, zda dokáže ChatGPT v rukou neprogramátora vygenerovat prostou aplikaci s malým počtem tříd a základní funkcionalitou. Výsledkem by měl být jednoduchý program pro ukládání, vypisování a označování denních úkolů a aktivit jako splněných. Cílem bylo určit, zda je možné pro členy druhé skupiny vytvořit program s minimální složitostí. Načítání ze souboru bylo vyžadováno pro zajištění reálné využitelnosti tohoto programu a zajištění správně navrženého veřejného rozhraní tříd. Příklad možné struktury je zobrazen na obrázku 2. Žádná skupina neměla tento obrázek k dispozici, jelikož by to mohlo příliš sjednotit výsledky.



Obrázek 2 Návrh struktury první úlohy v UML

Zdroj: vlastní, vytvořeno pomocí nástroje Enterprise Architect

Zadání této úlohy bylo následující:

V jazyce Java vytvořte jednoduchý konzolový program pro správu denního plánu, který umožní uživatelům zaznamenávat a sledovat své úkoly a aktivity. Žádné uživatelské rozhraní není nutné, důležitý je výpis správné funkcionality v konzoli. V programu by mělo být možné přidat a odebrat úkoly, zobrazit všechny úkoly, zobrazit pouze nesplněné úkoly a uložit a načíst úkoly ze souboru.

Při zapnutí programu by se měl do konzole vypsat výstup, který prokazuje správnou funkcionalitu programu. Důležitá je funkce ukládání a načítání ze souboru, zobrazení všech úloh, označení úkolů jako splněných, a zobrazení pouze nesplněných úkolů. Pokud bude program splňovat požadovanou funkcionalitu, ale nebude přítomný ukázkový výstup do konzole, tak to bude považováno za částečné nesplnění zadání.

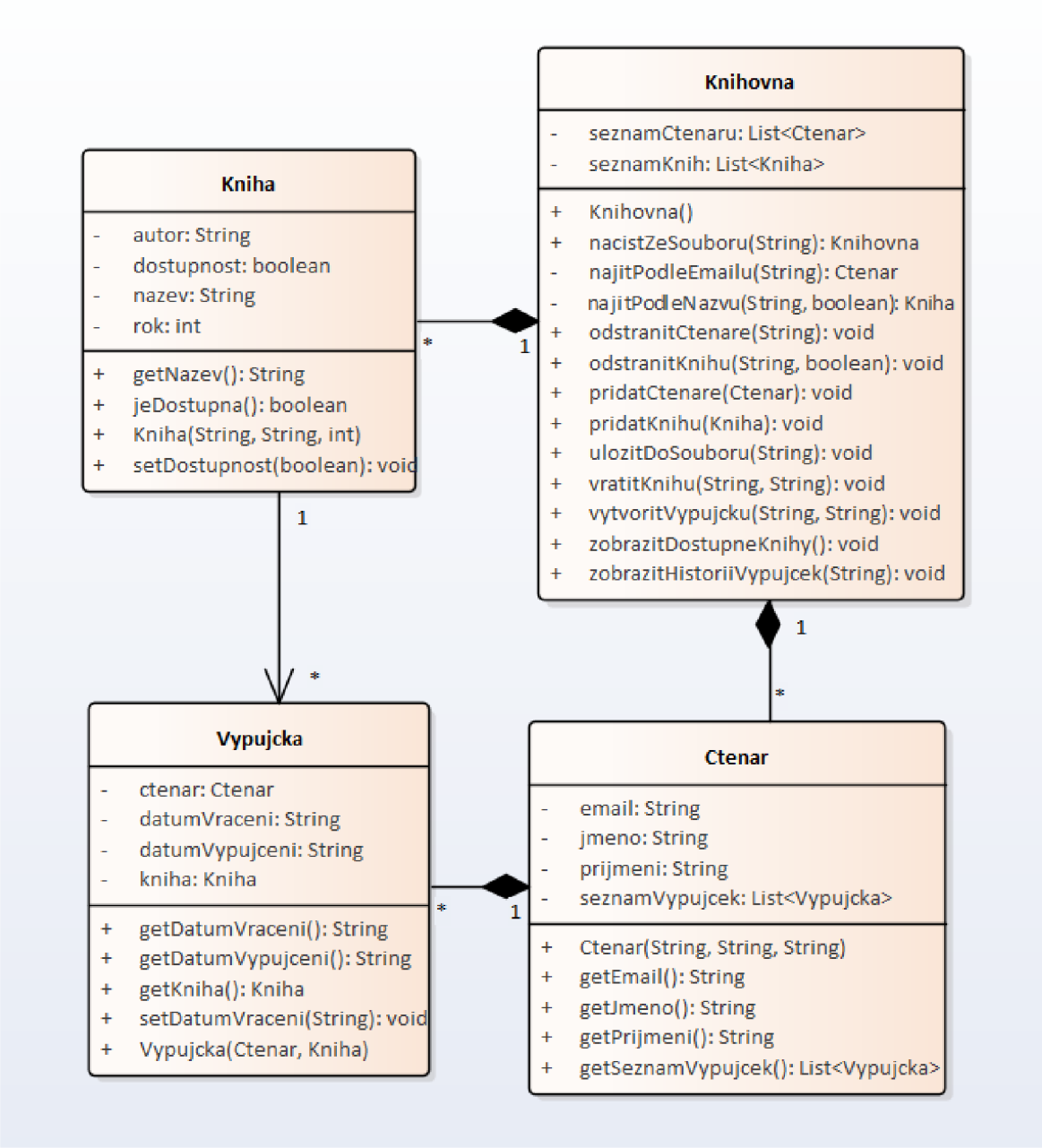
Požadované funkcionality:

1. Třída Úkol:
    a. Atributy: název úkolu, popis úkolu, označení pro splněný úkol.
    b. Metody: konstruktor, nutné metody pro získání a nastavení atributů.
2. Třída Plánovač:
    a. Atributy: seznam úkolů.
    b. Metody: přidání úkolu do plánu, odstranění úkolu, vyhledání úkolu podle názvu, označení úkolu jako splněného, zobrazení všech úkolů, zobrazení pouze nesplněných úkolů.
3. Dodatečné požadavky:
    a. Umožnit ukládání a načítání dat do/z externího souboru (Serializace).
4. Vstupní bod programu (Main):
    a. Vytvoření instance plánovače a načtení dat ze souboru (pokud existuje).
    a. Zobrazení celého seznamu úkolů.
    b. Přidání dalších úkolů do seznamu.
    c. Označení jednoho z těchto úkolů podle jeho názvu jako splněný.
    d. Zobrazení pouze nesplněních úkolů.
    e. Uložení do souboru.

2.2.2 Úloha 2 - Správa knih v knihovně

Zadání pro druhou úlohu s názvem Správa knih v knihovně bylo navrženo jako složitější zadání než zadání pro první úlohu. Komplikovanější zadání zvyšuje šanci výskytu chyb v kódu a snižuje tak pravděpodobnost, že ChatGPT dokáže vygenerovat funkční kód bez odborné navigace programátora. Na rozdíl od první úlohy toto zadání obsahuje vyšší počet tříd a komplikovanější komunikaci mezi nimi. Cílem tohoto

zadání je ověřit správnou funkcionalitu základních mechanismů zahrnujících více tříd. Stejně jako u prvního zadaní bylo také požadováno načítaní ze souboru ze stejných důvodů. Příklad možné struktury je zobrazen na obrázku 3. Stejně jako u první úlohy, neměla ani jedna skupina tento obrázek k dispozici, jelikož by to mohlo příliš sjednotit výsledky.



Obrázek 3 Návrh struktury druhé úlohy v UML

Zdroj: vlastní, vytvořeno pomocí nástroje Enterprise Architect

Zadání této úlohy bylo následující:

V jazyce Java vytvořte objektově orientovaný systém pro správu knih v knihovně. Uživatelské rozhraní není nutné, důležitý je výpis správné funkcionality v konzoli. Systém by měl umožňovat přidávání a odebírání knih z knihovny, zobrazit všechny dostupné knihy, přidávání a odebírání čtenářů z knihovny, a vytváření výpůjček knih. Vypůjčením a vrácením knih by mělo dojít ke změně dostupnosti dané knihy. Mělo by také být možné zobrazit historii všech knih, které si daný čtenář vypůjčil. Program by mělo být možné uložit a načíst ze souboru.

Při zapnutí programu by se měl do konzole vypsat výstup, který prokazuje správnou funkcionalitu systému. Důležitá je funkce ukládání a načítání ze souboru, zobrazení všech dostupných knih, vypůjčení a vrácení knihy, a zobrazení historie výpůjček daného uživatele. Pokud bude program splňovat požadovanou funkcionalitu, ale nebude přítomný ukázkový výstup do konzole, tak to bude považováno za částečné nesplnění zadání.

Požadované funkcionality:

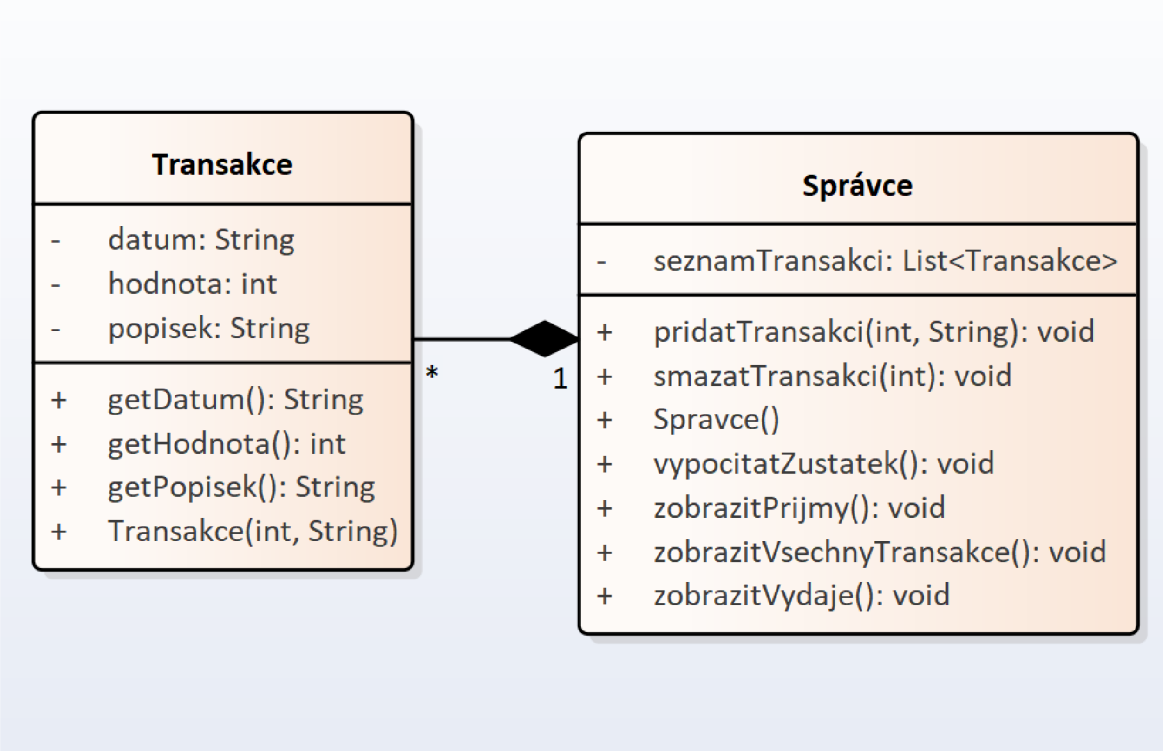
1. Třída Kniha:
    a. Atributy: název knihy, autor, rok vydání, dostupnost.
    b. Metody: konstruktor, nutné metody pro získání a nastavení atributů
2. Třída Čtenář:
    a. Atributy: jméno, příjmení, email, seznam výpůjček.
    b. Metody: konstruktor, nutné metody pro získání a nastavení atributů.
3. Třída Výpůjčka:
    a. Atributy: čtenář, kniha, datum výpůjčky, datum vrácení.
    b. Metody: konstruktor, nutné metody pro získání a nastavení atributů.

4. Třída Knihovna:
    a. Atributy: seznam knih, seznam čtenářů.
    b. Metody: přidání knihy do knihovny, odstranění knihy, přidání a odebrání čtenáře, vytvoření nové výpůjčky, vrácení knihy, zobrazení historie výpůjček čtenáře, vyhledání čtenáře podle emailu, vyhledání knihy podle názvu, zobrazení dostupných knih.
5. Dodatečné požadavky:
    a. Umožnit ukládání a načítání dat do/z externího souboru (Serializace).
    b. Implementovat mechanismus pro vypůjčení a vrácení knih (změna dostupnosti knihy při vypůjčení a vrácení).
6. Vstupní bod programu (Main):
    a. Vytvoření instance knihovny a načtení dat ze souboru (pokud existuje).
    b. Zobrazení dostupných knih.
    c. Vytvoření a přidání knih do knihovny.
    d. Zobrazení dostupných knih.
    e. Vytvoření a přidání čtenáře do knihovny.
    f. Vytvoření a přidání výpůjčky do knihovny.
    g. Zobrazení dostupných knih.
    h. Zobrazit všechny knihy které má daný uživatel vypůjčené.
    i. Vrácení knihy.
    j. Zobrazení dostupných knih.
    k. Uložení do souboru.

2.2.3 Úloha 3 – Správa financí

Zadání pro třetí úlohu s názvem Správa financí bylo navrženo jako jednoduché zadání s cílem určit, jak se budou výsledky lišit při méně jednoznačném zadání. V tomto zadání proto není detailně popsaná požadovaná funkcionalita formou odrážek, které detailně popisovaly strukturu požadovaných tříd u obou předchozích zadaní. Výsledkem by měl být jednoduchý program pro správu příjmů a výdajů. Příklad možné struktury je zobrazen na obrázku 4. Opět neměla žádná skupina tento obrázek k dispozici. Ukládání do souboru nebylo v této úloze nutné, protože cílem tohoto

zadání bylo určit, jak se bude lišit struktura tříd bez přesného zadání. Samotné řešení této funkcionality by také bylo totožné jako u přechozích úloh.



Obrázek 4 Návrh struktury třetí úlohy v UML

Zdroj: vlastní, vytvořeno pomocí nástroje Enterprise Architect

Zadání této úlohy bylo následující:

V jazyce Java vytvořte jednoduchý konzolový program pro správu finančních transakcí, který umožní uživatelům zaznamenávat a sledovat své výdaje a příjmy. Žádné uživatelské rozhraní není nutné, důležitý je výpis správné funkcionality v konzoli. V programu by mělo být možné přidat a odebrat transakce, zobrazit všechny transakce, zobrazit pouze výdaje nebo pouze příjmy, a zobrazit celkový finanční zůstatek. Součástí každé položky je kromě finanční částky také datum a krátký popis. Ukládání do souboru není nutné.

Při zapnutí programu by se měl do konzole vypsat výstup, který prokazuje správnou funkcionalitu programu. Důležitá je funkce zobrazení všech transakcí, zobrazení pouze příjmů, zobrazení pouze výdajů, výpočet celkového zůstatku. Pokud bude

program splňovat požadovanou funkcionalitu, ale nebude přítomný ukázkový výstup do konzole, tak to bude považováno za částečné nesplnění zadání.

3 Výsledky

Výsledky byly zobrazeny do dvou typů tabulek. První typ zobrazuje hodnocení daného zdrojového kódu. V tomto typu tabulky je zobrazen každý člen výzkumu jednotlivě. Programátor je v těchto tabulkách označení pro člena první skupiny a ChatGPT je označení pro člena druhé skupiny. V druhé tabulce jsou zobrazeny časy jednotlivých skupin a vypočítané navýšení efektivity pracovníků těchto skupin vyjádřené v procentech a časová redukce rychlejší skupiny také vyjádřena v procentech. Čas je zobrazen ve formátu „x m y s“ kde x je počet minut, m je zkratka pro minuty, y je počet sekund, a s je zkratka pro sekundy.

3.1 Úloha 1 – Plánovač aktivit

Porovnání jednotlivých programů pro první úlohu je zobrazené v tabulce 1. Počet chyb v tabulce odpovídá vážným chybám, které ovlivňují funkcionalitu kódu. Oba vygenerované programy a jeden manuálně naprogramovaný program splňovaly požadovanou funkcionalitu. U druhého manuálně naprogramovaného programu se vyskytla chyba při porovnávání hodnot řetězců. Pro porovnání obsahu řetězců v jazyce Java je nutné použít metodu „.equals“. V chybném kódu byly řetězce porovnány pomocí operátoru „==“, který porovnává pouze referenci těchto řetězců a nikoliv jejich obsah. To vedlo k problému při práci s načtenými daty. Funkcionalita programu nebyla omezena při práci s nenačtenými daty ze souboru, což pravděpodobně vedlo k zanedbání této chyby. Kvalita vypisování dat do konzole byla různorodá u všech programů, a nelze přesně určit jaká skupina je v tomto ohledu lepší.

Tabulka 1 Hodnocení první úlohy

	Funkcionalita	Struktura	Optimalizace	Přehlednost	Počet chyb
Programátor 1	10	10	10	9	0
Programátor 2	8	10	10	7	1
ChatGPT 1	10	9	9	10	0
ChatGPT 2	10	9	9	10	0

Zdroj: vlastní

Struktura všech programů byla na kvalitní úrovni. Oba vygenerované programy však měly podobný nedostatek, co se týče metod pro ukládání a načítání ze souboru. Tyto metody byly v těchto programech nekompletní, co se týče výjimek. Pro jejich použití tedy bylo nutné obalit volání těchto metod výjimkami ve třídě Main. Na obrázku 5 je zobrazeno jedno takové použití této metody ve třídě Main. Bylo by vhodnější zakomponovat tyto výjimky do samotné metody.

```
// Uložení do souboru
try {
    planovac.ulozitDoSouboru("ukoly.dat");
} catch (IOException e) {
    System.out.println("Nepodařilo se uložit úkoly do souboru.");
}
```

Obrázek 5 Nevhodně strukturovaná metoda

Zdroj: vlastní

Optimalizace všech programů byla také na kvalitní úrovni. Každý program byl napsán trochu jiným stylem, ale jejich logika byla podobná. Vygenerovaný kód byl napsán úspornějším a modernějším stylem. U vygenerovaných programů se však vyskytoval přebytečný import ve třídě Plánovač. Toto nemá efekt na funkci či náročnost programu, a mohlo by to být považováno za chybu v oblasti přehlednosti kódu. Jelikož se ale technicky jednalo o zbytečnou část kódu, bylo to považováno za nedostatek v oblasti optimalizace. Vygenerovaný kód byl celkově přehlednější a lépe okomentovaný než manuálně naprogramovaný kód. Druhý naprogramovaný kód také obsahoval přebytečné a nevyužité metody pro práci s atributy, pravděpodobně vygenerované pomocí vývojového prostředí.

V tabulce 2 je zobrazen výsledný čas zpracování první úlohy mezi oběma skupinami. Skupina pracující s nástrojem ChatGPT dokázala úlohu dokončit daleko rychleji než první skupina. Maximální čas první skupiny byl 21 minut a 23 sekund. Minimální čas byl 17 minut a 48 sekund. Průměrný čas byl 19 minut a 36 sekund. U druhé skupiny byl maximální čas pouze 12 minut a 15 sekund. Minimální čas byly pouhé 4 minuty 37 sekund. Průměrný čas druhé skupiny tak odpovídal 8 minutám a 26 sekundám. Časová

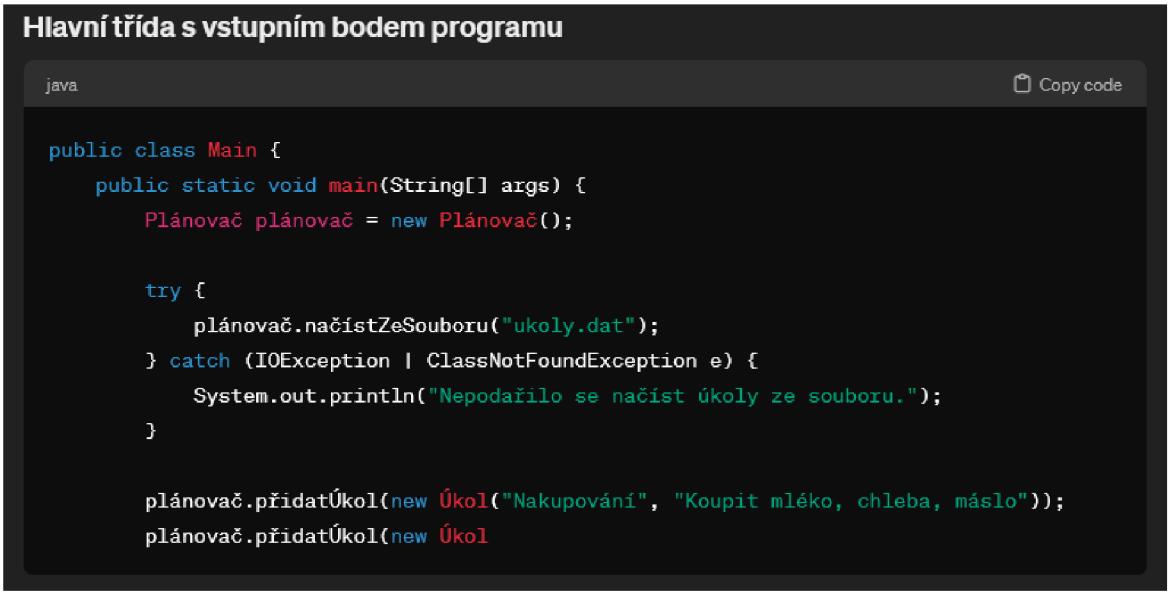
redukce u maximálního času byla 42,71 %, 74,06 % u minimálního času a 56,97 % u průměrného času. Nejrychlejší čas byl u druhé skupiny o 285,56 % efektivnější než u skupiny první. Maximální čas ale byl u druhé skupiny efektivnější pouze o 42,71 %. Průměrný nárůst efektivity u druhé skupiny tak byl 132,31 %.

Tabulka 2 Porovnání času první úlohy

	Průměrný čas	Maximální čas	Minimální čas
První skupina	19 m 36 s	21 m 23 s	17 m 48 s
Druhá skupina	8 m 26 s	12 m 15 s	4 m 37 s
Redukce času	56,97 %	42,71 %	74,06 %
Zvýšení efektivity	132,31 %	74,56 %	285,56 %

Zdroj: vlastní

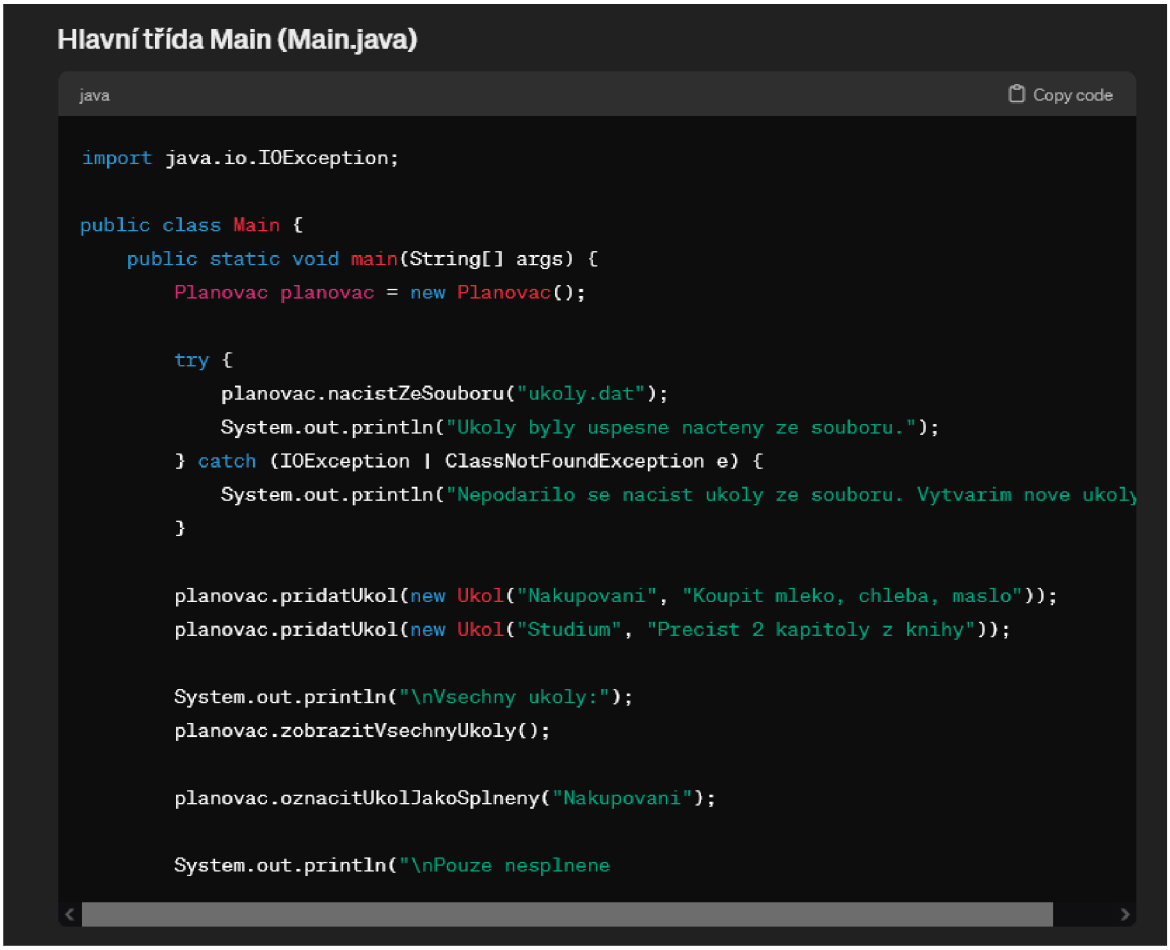
Poměrně vysoká diskrepance u časů druhé skupiny může být vysvětlena analýzou konverzace s ChatGPT. Zatímco u rychlejšího času dokázal nástroj vygenerovat požadovaný zdrojový kód ihned, u pomalejšího času se vyskytl problém, kdy nástroj vygeneroval nedokončený a tím pádem nespustitelný kód. Tento problém je zobrazen na obrázku 6.



Obrázek 6 Nedokončené generování třídy Main

Zdroj: vlastní

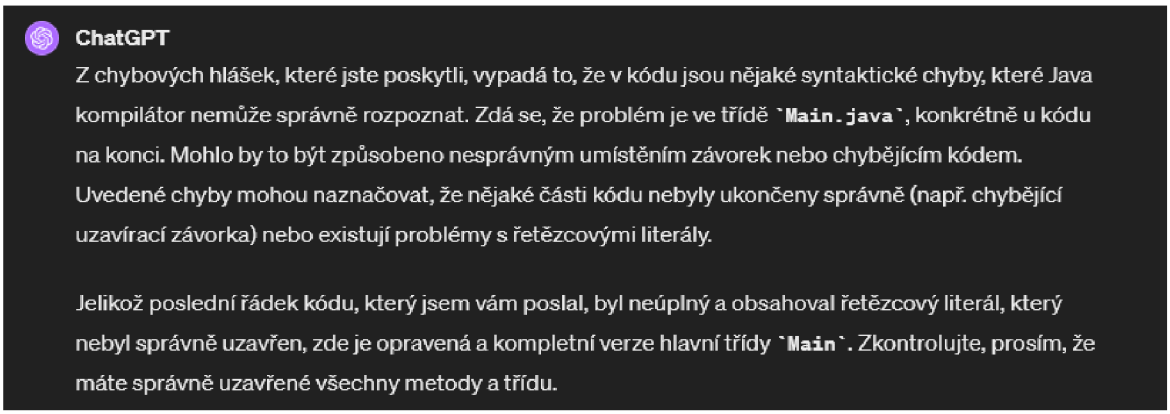
Člen výzkumu následně zaslal chybové hlášky z vývojového prostředí a instruoval nástroj o opětovné vygenerování kódu. Nově vygenerovaný kód měl ovšem totožný problém zobrazen na obrázku 7.



Obrázek 7 Opětovné nedokončení generování třídy Main

Zdroj: vlastní

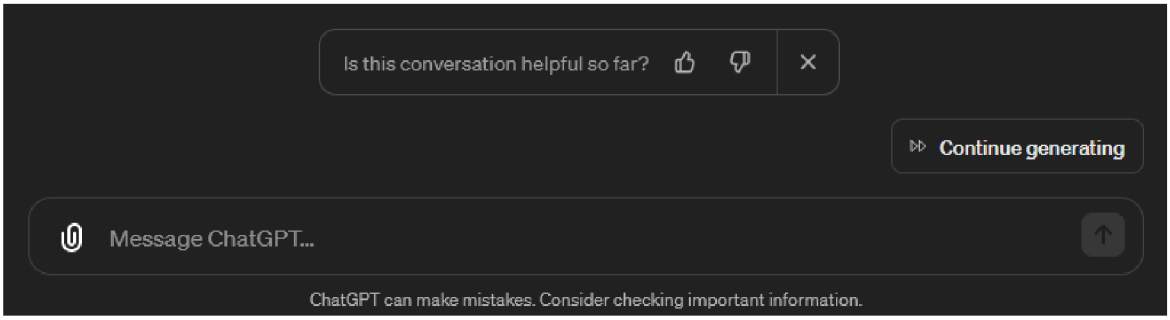
Po opětovném zaslání chybových hlášek si však nástroj sám uvědomil problém a dokázal ho opravit a vygenerovat kompletní kód, což je zobrazeno na obrázku 8. Tento kód stále nebyl plně funkční, jelikož neobsahoval import požadovaných knihoven, což bylo následně opraveno opětovném zaslání chybových hlášek z vývojového prostředí.



Obrázek 8 Opravení nedokončeného generování třídy Main

Zdroj: vlastní

Je možné, že se tato komplikace vyskytla z důvodu pozastavení generace výsledku. Nástroj má omezený počet znaků, který je schopný vygenerovat a pokud je tento limit dosažen, zastaví generaci výsledků. V generování výsledků je možné pokračovat kliknutím na tlačítko „Continue generating“ zobrazené na obrázku 9, nebo odesláním zprávy typu „pokračuj“. Je možné že uživatel toto tlačítko přehlédl a neuvědomil si, že kód není kompletní. Dalším vysvětlením je síťová chyba nebo serverový výpadek ze strany OpenAI. Chyba pravděpodobně nebyla způsobená původní zprávou konverzace, jelikož oba uživatelé poslali téměř totožnou zprávu obsahující kompletní zadání této úlohy.



Obrázek 9 Pozastavené generování

Zdroj: vlastní

Jedná se tedy o poměrně odlišné výsledky, co se týče schopnosti generování zdrojového kódu. Přesto ale byli oba členové druhé skupiny rychlejší než členové první skupiny. Vygenerované a naprogramované programy byly přibližně na stejné úrovni.

Celkově lze tak stanovit že v rámci první úlohy byla efektivnější druhá skupina a nástroj ChatGPT.

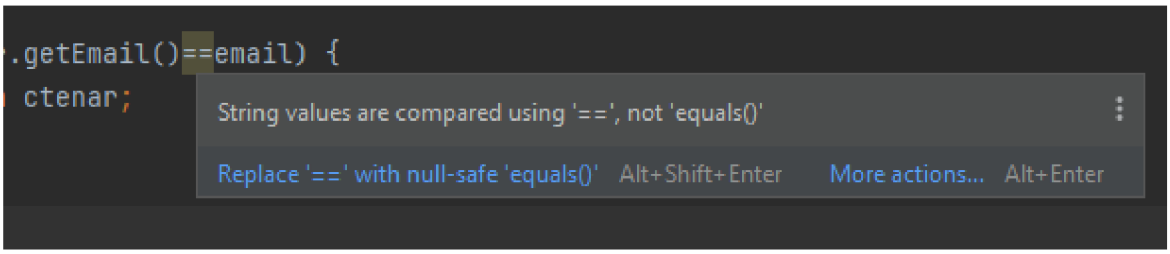
3.2 Úloha 2 – Správa knih v knihovně

Porovnání jednotlivých programů pro druhou úlohu je zobrazené v tabulce 3. U druhého programu první skupiny se vyskytl stejný problém jako u předchozí úlohy, kdy jsou hodnoty řetězců porovnány pomocí operátorů, a nikoliv pomocí metody. Jelikož se tento problém vyskytl v tomto kódu dvakrát, bylo to počítáno jako dvě chyby. Následek těchto chyb je stejný jako u přechozí úlohy. Problém tedy nastane při vyhledávání objektů načtených ze souboru. Tato chyba dá lehce přehlédnout, jelikož jenom částečně omezuje funkcionalitu za určitých podmínek. Vývojové prostředí ovšem na tuto chybu upozorňuje, jak je zobrazeno na obrázku 10. Je možné, že autor chybného kóduobvykle pracuje v jiném programovacím jazyce, kde k této chybě nemůže dojít, nebo se jedná o čistou nepozornost. Nicméně je to typ chyby, kterou by nástroj umělé inteligence pravděpodobně neudělal. Kvalita vypisování do konzole byla opět dostatečná u všech programů.

Tabulka 3 Hodnocení druhé úlohy

	Funkcionalita	Struktura	Optimalizace	Přehlednost	Počet chyb
Programátor 1	10	10	10	9	0
Programátor 2	8	10	10	7	2
ChatGPT 1	10	9	9	10	0
ChatGPT 2	8	10	9	10	1

Zdroj: vlastní



Obrázek 10 Upozornění od vývojového prostředí na možnou chybu

Zdroj: vlastní

Další chyba a omezení funkcionality byla ve druhém vygenerovaném programu. Chyba byla v mechanismu vrácení knih, kdy při vrácení knihy nedošlo k nastavení datumu vrácení u instance objektu výpůjčka. To způsobilo že přestože byly knihy vráceny do knihovny a jejich dostupnost byla správně změněna, tak u historie výpůjček čtenářů byly vždy výpůjčky zobrazeny jako nevrácené. Nástroj ChatGPT si podle komentářů logiku tohoto mechanismu uvědomil v konstruktoru třídy Výpůjčka, jak je zobrazeno na obrázku 11. Tato logika však nebyla nikdy implementována, jak je zobrazeno v následujícím obrázku 12. První vygenerovaný program má v této části velmi podobnou strukturu, ale bez této chyby. U prvního vygenerovaného programu je tedy datum vrácení správně zaznamenáno při vrácení knihy.

```
1 usage
public Vypujcka(Ctenar ctenar, Kniha kniha, Date datumVypujcky) {
    this.ctenar = ctenar;
    this.kniha = kniha;
    this.datumVypujcky = datumVypujcky;
    this.datumVraceni = null; // Datum vrácení se nastaví při vrácení knihy
}
```

Obrázek 11 Konstruktor třídy Výpůjčka

Zdroj: vlastní

```
// Vrácení knihy
1 usage
public void vratitKnihu(String nazevKnihy) {
    knihy.stream().filter(k -> k.getNazev().equals(nazevKnihy))
            .findFirst().ifPresent(kniha -> kniha.setDostupnost(true));
}
```

Obrázek 12 Chybějící logika úpravy datumu vrácení

Zdroj: vlastní

Struktura všech programů byla opět na kvalitní úrovni. Pouze jeden vygenerovaný program však měl obdobný nedostatek jako oba vygenerované programy u předešlé úlohy. U druhého vygenerovaného programu tedy nebyly nutné výjimky ve třídě Main při načítání dat ze souboru.

Optimalizace všech programů byla opět na kvalitní úrovni. Opětovně byl každý program napsán trochu jiným stylem, ale jejich logika byla podobná. Vygenerovaný kód měl opět úspornější a modernější styl. U vygenerovaných programů se opětovně vyskytoval přebytečný import ve třídě Plánovač. Tento nedostatek je popsán u minulé úlohy. Dopad na hodnocení byl totožný jako u předchozí úlohy. Vygenerovaný kód byl opět celkově přehlednější a lépe okomentovaný než manuálně naprogramovaný kód. Druhý naprogramovaný kód opětovně obsahoval přebytečné a nevyužité metody pro práci s atributy, pravděpodobně vygenerované pomocí vývojového prostředí.

V tabulce 4 je zobrazen výsledný čas zpracování druhé úlohy mezi oběma skupinami. Tato úloha byla komplikovanější, a tedy i časově náročnější. Je ale důležité si uvědomit, že členové obou skupin byli po splnění přechozí úlohy zkušenější a bylo očekávané zlepšení efektivity při zpracování této a následující úlohy. Skupina pracující s nástrojem ChatGPT opět dokázala úlohu dokončit daleko rychleji než první skupina. Časový rozdíl a zrychlení je u této úlohy větší než u minulé úlohy. Maximální čas první skupiny byl 56 minut a 13 sekund. Minimální čas byl 47 minut a 40 sekund. Průměrný čas byl 51 minut a 57 sekund. U druhé skupiny byl maximální čas pouze 24 minut a 7 sekund. Minimální čas byl 8 minut 36 sekund.

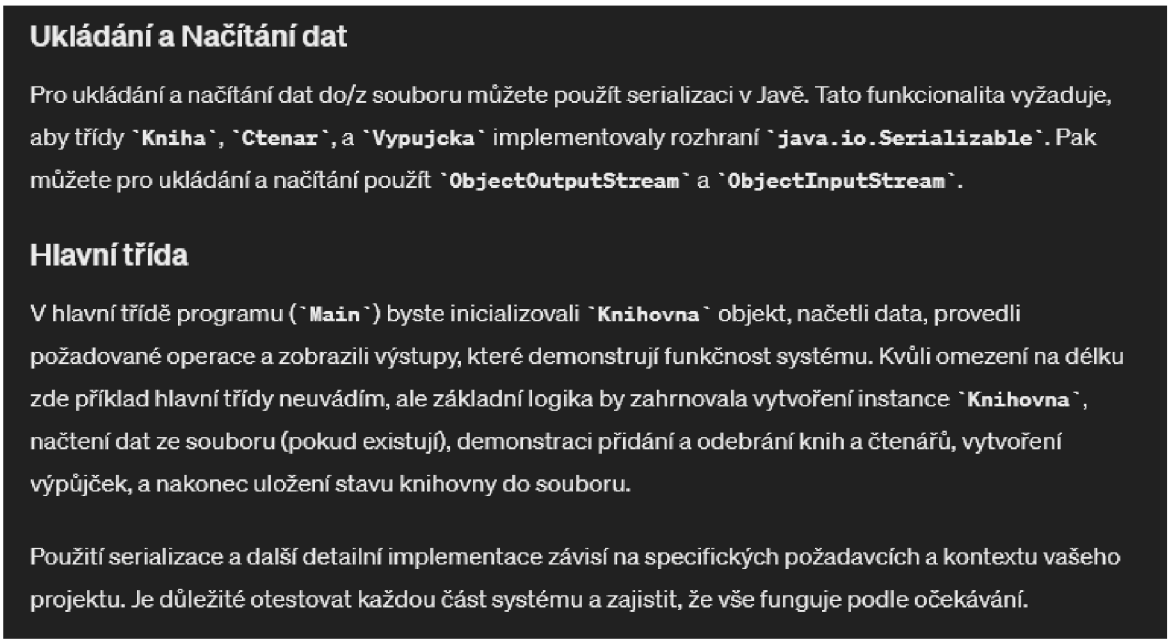
Průměrný čas druhé skupiny byl 16 minut a 22 sekund. To odpovídá 57,10 % časové redukci u maximálního času, 81,96 % časové redukci u minimálního času a 68,50 % časové redukci u průměrného času. U nejrychlejšího času byla druhá skupina o 454,26 % efektivnější. U maximálního času ale byla druhá skupina efektivnější o 133,10 %. Průměrný nárůst efektivity u druhé skupiny byl 217,52 %.

Tabulka 4 Porovnání času druhé úlohy

	Průměrný čas	**Maximální čas**	**Minimální čas**
První skupina	51 m 57 s	56 m 13 s	47 m 40 s
Druhá skupina	16 m 22 s	24 m 7 s	8 m 36 s
Redukce času	68,50 %	57,10 %	81,96 %
Zvýšení efektivity	217,52 %	133,10 %	454,26 %

Zdroj: vlastní

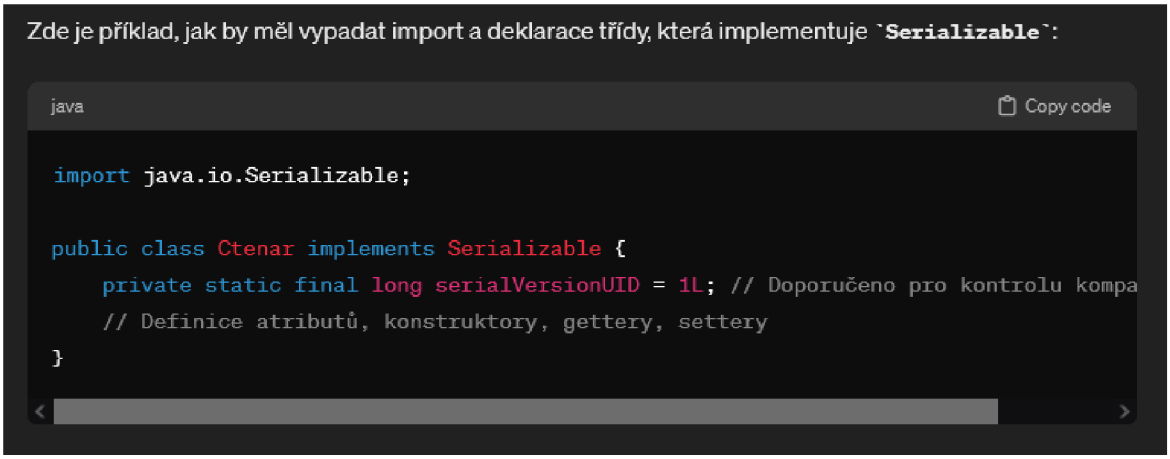
Zatímco časy první skupiny jsou opět relativně podobné, u druhé skupiny je opět znatelný rozdíl. Zatímco u první konverzace byl funkční program vygenerován ihned, u druhé konverzace se vyskytl problém v oblasti serializace. Nástroj původně nevygeneroval program s mechanismem ukládání a načítání ze souboru, ale pouze instruoval uživatel, jak tento mechanismus implementovat. Také z důvodů omezení délky vůbec nevygeneroval třídu Main. Tento výstup je zobrazen na obrázku 13.



Obrázek 13 Nevygenerovaná serializace a třída Main

Zdroj: vlastní

Člen výzkumu si díky této zprávě uvědomil nedostatek vygenerovaného kódu a následovala delší konverzace o přidání mechanismu ukládání a načítání ze souboru do vygenerovaného kódu. Nakonec ještě bylo nutné vyřešit problém chybějící implementace rozhraní „serializable“ a absence importu potřebných knihoven podle obrázku 14. Pro uživatele nezkušeného v oblasti programování by bylo výhodnější nechat si celý kód vygenerovat znova i s mechanismem serializace.



Obrázek 14 Část instrukce pro zakomponování mechanismu serializace

Zdroj: vlastní

Opět se tedy jako u přechozí úlohy vyskytl znatelný rozdíl při generování požadovaného kódu. Oba členové výzkumu druhé skupiny ale zadali téměř totožnou první zprávu do konverzace, která odpovídala celému zadání s krátkou instrukcí o vygenerování zadaného kódu. U delší konverzace, kde se vyskytl problém absence implementace mechanismu ukládání a načítání ze souboru dokonce uživatel jako součást první zprávy stanovil, že není programátor. Přesto ho nástroj místo vygenerování kompletního kódu vedl k manuální implementaci této funkcionality. I tak ale byli oba členové druhé skupiny opětovně rychlejší než členové první skupiny. Vygenerované a naprogramované programy byly opět přibližně na stejné úrovni. Přestože jeden vygenerovaný program obsahoval chybu, která znatelně ovlivňovala funkcionalitu program a jednalo se o problematičtější chybu, než která se vyskytla u druhého naprogramovaného programu, tak byla opět efektivnější druhá skupina a nástroj ChatGPT.

3.3 Úloha 3 – Správa financí

Porovnání jednotlivých programů pro poslední úlohu je zobrazené v tabulce 5. V žádném programu nebyla nalezena podstatná chyba nebo chybějící požadovaná funkcionalita. Přestože bylo zadání méně detailně popsané než zadání pro předchozí úlohy, jsou všechny výsledné kódy podobné. Vygenerované programy používaly výčtový typ (enumerated type) pro nastavení atributu „typ“, který rozlišuje mezi výdaji a příjmy. Oba manuálně naprogramované programy rozlišovaly výdaje a příjmy podle toho, zda byla hodnota transakce kladná či záporná. Oba způsoby jsou v pořádku a nejsou v rozporu s daným zadáním. Jelikož se jednalo o nejjednodušší úlohu s cílem zjistit, zda příliš detailní zadání ovlivňuje efektivitu druhé skupiny, tak je výsledný kód velmi stručný, dobře strukturovaný a přehledný. Druhý vygenerovaný program byl celý umístěn do jednoho souboru, což částečně zhoršilo přehlednost kódu. V druhém manuálně naprogramovaném kódu se opět vyskytly přebytečné metody pro nastavení atributů. V žádném programu se nevyskytovaly komentáře.

Tabulka 5 Hodnocení třetí úlohy

	Funkcionalita	Struktura	Optimalizace	Přehlednost	Počet chyb
Programátor 1	10	10	10	10	0
Programátor 2	10	10	10	9	0
ChatGPT 1	10	10	10	9	0
ChatGPT 2	10	10	10	10	0

Zdroj: vlastní

V tabulce 6 je zobrazen výsledný čas zpracování třetí úlohy mezi oběma skupinami. Tato úloha byla nejjednodušší a časově nejméně náročná. Skupina pracující s nástrojem ChatGPT opět dokázala dokončit úlohu rychleji než první skupina. Časový rozdíl a zrychlení je u této úlohy byl nejmenší ze všech úloh. Maximální čas první skupiny byl 8 minut a 49 sekund. Minimální čas byl 7 minut a 55 sekund. Průměrný čas byl 8 minut a 22 sekund. U druhé skupiny byl maximální čas 8 minut a 43 sekund. Minimální čas byl 3 minuty 17 sekund. Průměrný čas druhé skupiny byl přesně 6 minut. To odpovídá 1,13 % časové redukci u maximálního času, 58,53 % časové redukci u minimálního času a průměrné 28,29 % časové redukci. U nejrychlejšího času byla druhá skupina o 141,12 % efektivnější. U maximálního času ale byla druhá skupina efektivnější pouze o 1,15 %. Průměrný nárůst efektivity u druhé skupiny byl 39,44 %.

Tabulka 6 Porovnání času třetí úlohy

	Průměrný čas	Maximální čas	Minimální čas
První skupina	8 m 22 s	8 m 49 s	7 m 55 s
Druhá skupina	6 m	8 m 43 s	3 m 17 s
Redukce času	28,29 %	1,13 %	58,53 %
Zvýšení efektivity	39,44 %	1,15 %	141,12 %

Zdroj: vlastní

Zatímco časy první skupiny byly přibližné, opět se vyskytl značný rozdíl u časů druhé skupiny. Rychlejší čas druhé skupiny byl opět vygenerován bezchybně jako první výstup, zatímco u pomalejšího času došlo ke komplikacím. Nástroj ChatGPT vygeneroval kód správně, ale jako vstupní bod programu použil vlastní třídu FinanceApp, a nikoliv výchozí třídu Main. Třída je zobrazena na obrázku 15. To je naprosto správný výstup, který ovšem pravděpodobně zmátl uživatele, který nebyl zkušený v programování. Po zaslání chybové hlášky nástroj ChatGPT upravil kód do jednoho souboru a popsal, jak soubor spustit.

```
public class FinanceApp {
    public static void main(String[] args) {
        FinanceManager manager = new FinanceManager();

        manager.addTransaction(new Transaction(Transaction.Type.INCOME, 500.00, "2023
        manager.addTransaction(new Transaction(Transaction.Type.EXPENSE, 50.00, "2023
        manager.addTransaction(new Transaction(Transaction.Type.EXPENSE, 200.00, "202
        manager.addTransaction(new Transaction(Transaction.Type.INCOME, 150.00, "2023

        System.out.println("All Transactions:");
        manager.showAllTransactions();

        System.out.println("\nIncome Transactions:");
        manager.showTransactionsByType(Transaction.Type.INCOME);

        System.out.println("\nExpense Transactions:");
        manager.showTransactionsByType(Transaction.Type.EXPENSE);

        System.out.println("\nTotal Balance:");
        manager.showBalance();
    }
}
```

Obrázek 15 Spouštěcí třída FinanceApp

Zdroj: vlastní

Hodnocení této úlohy bylo téměř totožné u všech programů a časově byla opět efektivnější druhá skupina. I u této úlohy tak byla celkově efektivnější druhá skupina a nástroj ChatGPT. Méně jasné zadání také nemělo podstatný vliv na kvalitu výsledků jakékoliv skupiny. Zajímavé taky je, že rozdíl maximálních časů obou skupin byl pouze 6 sekund. Celkový nárůst efektivity byl také menší než

u předchozích úloh. Z toho vyplývá že druhá skupina byla více efektivní u náročnějších úloh než jednoduchých úloh, což je částečně v rozporu se základním předpokladem, že nástroj ChatGPT je lepší pro jednodušší úlohy.

3.4 Celkové porovnání sady úloh

V tabulce 7 jsou zobrazeny průměrné výsledky hodnocení všech úloh. Součástí tabulky je také průměrné hodnocení obou skupiny a číslo vedoucí skupiny v dané oblasti. Zatímco první skupina měla lepší výsledky v oblastech struktury kódu a optimalizace, druhá skupina měla lepší výsledky v oblastech funkcionality programu a přehlednosti kódu. Ve druhé skupině se také vyskytl nižší průměrný počet chyb. Ze samotných hodnot je ale patrné, že obě skupiny byly na podobné úrovni ve všech oblastech. Největší rozdíl byl v oblasti přehlednosti kódu, kde byla lepší druhá skupina. Přestože byl rozdíl těchto hodnot poměrně malý, je možné stanovit, že podle těchto hodnot byla celkově lepší druhá skupina a nástroj ChatGPT.

Tabulka 7 Průměrné hodnocení všech úloh

	Funkcionalita	Struktura	Optimalizace	Přehlednost	Počet chyb
Programátor 1	10	10	10	9,33	0
Programátor 2	8,67	10	10	7,67	1
ChatGPT 1	10	9,33	9,33	9.67	0
ChatGPT 2	9,33	9,67	9.33	10	0,33
První skupina	9,33	10	10	8,5	0,5
Druhá skupina	9,67	9,5	9,33	9,83	0,17
Vedoucí skupina	2	1	1	2	2

Zdroj: vlastní

V tabulce 8 je zobrazen výsledný čas zpracování celkové sady úloh obou skupin. Celkový maximální čas první skupiny byl 86 minut a 25 sekund. Celkový minimální čas byl 73 minut a 23 sekund. Celkový průměrný čas byl 79 minut a 55 sekund.

U druhé skupiny byl celkový maximální čas 45 minut a 5 sekund. Celkový minimální čas byl 16 minuty 30 sekund. Celkový průměrný čas druhé skupiny byl přesně 30 minut a 48 sekund. To odpovídá 47,83 % časové redukci u maximálního času, 77,52 % časové redukci u minimálního času a průměrné 61,46 % časové redukci. U nejrychlejšího času byla druhá skupina o 344,75 % efektivnější. U maximálního času byla druhá skupina efektivnější o 91,68 %. Průměrný nárůst efektivity u druhé skupiny byl 159,47 %.

Tabulka 8 Porovnání času sady úloh

	Průměrný čas	**Maximální čas**	**Minimální čas**
První skupina	79 m 55 s	86 m 25 s	73 m 23 s
Druhá skupina	30 m 48 s	45 m 5 s	16 m 30 s
Redukce času	61,46 %	47,83 %	77,52 %
Zvýšení efektivity	159,47 %	91,68 %	344,75 %

Zdroj: vlastní

Jelikož byla druhá skupina vždy rychlejší u každé úlohy, je tedy jasné že i celkově byla časově efektivnější druhá skupina a nástroj ChatGPT. Významné je, že průměrná časová redukce celkové sady úloh byla u druhé skupiny 61,46 %, tedy to znamená, že druhá skupiny byla obecně více než dvakrát rychlejší, něž skupina první. Přestože měly skupiny srovnatelné hodnocení, co se týče zpracování kódu, tak byla druhá skupina znatelně časově efektivnější ve zpracování těchto úloh. Na základě těchto výsledků byla ve zpracování zadané sady úloh celkově lepší druhá skupina a nástroj ChatGPT.

4 Shrnutí a diskuse výsledků

Jelikož byly všechny úlohy zpracované druhou skupinou na kvalitní úrovni, je možné stanovit, že nástroj ChatGPT nejen dokáže vygenerovat funkční zdrojový kód podle testované sady zadání, ale také že pro zpracování této sady úloh pomocí nástroje ChatGPT nejsou vyžadovány odborné znalosti v oboru programování. Ze získaných výsledků je patrné, že v rámci sady testovaných úloh byl nástroj ChatGPT v rukou osob nezkušených v oboru programování efektivní náhradou zkušených programátorů. Vygenerované zdrojové kódy byly u všech úloh na relativně téměř totožné úrovni z pohledu funkcionality, optimalizace, struktury a přehlednosti kódu jako manuálně naprogramované programy.

Druhá skupina pracující s nástrojem ChatGPT byla však znatelně rychlejší ve zpracování všech zadaných úloh. Nejmenší zaznamenaný průměrný nárůst efektivity byl přibližně 39 % u poslední a nejjednodušší úlohy. Nárůst efektivity zpracování těchto úloh byl ale vyšší u komplikovanějších úloh. Zejména u druhé úlohy byla druhá skupina o přibližně 217 % efektivnější než skupina první při porovnání průměrných časů jednotlivých skupin. U první úlohy byl průměrný nárůst efektivity přibližně 132 %. Při porovnání průměrných času celé sady úloh byla druhá skupina o přibližně 159 % efektivnější. Jedná se o velmi znatelné zvýšení efektivity rychlosti zpracování, zejména vzhledem ke skutečnosti, že členové druhé skupiny neměli zkušenosti ani odborné znalosti v oboru programování. Zjištěním z těchto výsledků také je, že využití AI je znatelně efektivnější u složitějších úloh. Pravděpodobně ale existuje limit složitosti, kdy bude AI neschopné danou úlohu korektně vygenerovat. Dokud tento limit ale není překročen, efektivita AI roste se složitostí zadané úlohy. Výzkum a jeho výsledky je však nutné posuzovat v kontextu omezených možností této práce.

Z výsledků je patrné, že AI má potenciál kompletně nahradit programátory. Pro zjištění, zda programátorům opravdu hrozí technologická nezaměstnanost je nutné provést dodatečný a rozsáhlý výzkum, který kompletně otestuje efektivitu nástrojů umělé inteligence ve všech oblastech programování. Výsledky této práce prokazují, že takový výzkum v této oblasti je smysluplný a potenciálně užitečný.

4.1 Omezení výzkumu

Jako pilotní výzkum má zvolený výzkum práce několik omezení, problémů a nedostatků. To neznamená, že jsou výsledky výzkumu irelevantní, ale je nutné brát v potaz tato omezení při interpretaci výsledků. Tato zjištění poskytují cenné informace pro budoucí výzkum v této oblasti. Součástí popisu daného problému je často také jeho možné řešení, a odůvodnění proč toto řešení nebylo zakomponováno do této práce.

Zvolený přístup nedokáže zcela zhodnotit kompletní schopnosti modelu GPT-4 co se týče generování kódu. Jedná se o velmi cílený a skromný test, který je zaměřen na malou část programovacích problémů v jednom programovacím jazyce a s velmi malou výzkumnou skupinou osob. Programátor se může setkat s mnohonásobně komplikovanějšími úkoly, než jsou testovány v této práci, pro které je nutné unikátní řešení podle daných okolností. Lepší přístup by vyžadoval více osob, jak zkušené, tak nezkušené s procesem programování, které by pracovaly na několika programovacích úlohách. Osoby by byly rozděleny do většího počtu skupin, které používají a nepoužívají generativní AI a na základě jejich programovacích zkušeností. Úlohy by byly mnohem komplikovanější a více různorodé co se týče zadání, programovacího jazyka a jiných použitých technologií. Takový přístup by byl vysoce časově náročný a vyžadoval poměrně velké množství osob, které by pravděpodobně vyžadovaly finanční odměnu za jejich čas. Výsledkem takového výzkumu by ale byly přesnější a obecnější výsledky a také možnost změřit, jak úroveň technických znalostí v oboru programování ovlivňuje efektivitu nástroje umělé inteligence jako je ChatGPT.

Další možností by bylo testovat přímo nástroj ChatGPT pod dohledem odborníka, který by testoval postup modelu na základě specifických dotazů a výsledky porovnával například s ukázkovým řešením daných úloh. Jinou možností testování tohoto nástroje by bylo testovat efektivitu programátorů při práci s a bez přístupu k ChatGPT. Pro přesnější výsledky by bylo vhodné programátory promíchávat, aby nebyly výsledky ovlivněny osobními dovednostmi jednotlivých programátorů. Potenciálně zajímavou možnosti testování nástrojů umělé inteligence v rámci programování by bylo testovat tyto nástroje mezi sebou a zjistit který nástroj je nejlepší. Výsledky by se mohly

odlišovat u různých typů programovacích úloh a programovacích jazyků.

Potenciálním problémem zvolené metodiky je přesnost zadání jednotlivých úloh. Zadání úloh byla velmi detailně popsaná a byla přesně stanovena požadovaná funkcionalita a struktura jednotlivých programů. Úlohy byly takto zadané z důvodu jednoznačnosti zadání a jednotnosti výsledků pro analýzu. Předpoklad je, že i pokud by úlohy byly nepřesněji zadané, tak by je obě skupiny dokázaly zpracovat. Výsledky by však byly příliš odlišné pro přímé porovnání, a i funkcionalita by se pravděpodobně odlišovala. Nevýhodou takto přesného zadání je potenciální zkreslení výsledků. V praxi je více očekáváné méně přesné zadání pro programovaní, zejména v oblasti struktury objektů. Pokud by zadání bylo méně přesné, může to negativně ovlivnit efektivitu skupiny neprogramátorů více než skupinu programátorů. Z tohoto důvodu byla navrženo zadání pro třetí úlohu s názvem Správa financí tak, aby bylo méně přesné, zejména co se týče struktury tříd. I toto zadání je ale potenciálně přesnější než běžné zadání, které by mohla osoba nezkušená s programováním v praxi zadat do nástroje umělé inteligence. Pro potřeby porovnání kódu je ale jistá míra jednoznačného zadání nutná. Dalším nedostatkem výzkumu této práce byla absence testování dalších principů OOP jako polymorfismus a dědičnost. Pro důsledné testování těchto principů by bylo potřeba testovat složitější úlohy, což by opět vedlo k problému časové náročnosti pro účastníky výzkumu.

Je také nutné si uvědomit, že jednotlivé úlohy jsou si částečně podobné, což může ovlivnit rychlost dokončení následujících úloh. Například metody ukládání a načítání ze souboru mohou být identické u první i druhé úlohy, což vede k rychlejšímu splnění druhé úlohy. Tedy dá se očekávat, že čas potřebný pro splnění druhé úlohy bude kratší než čas, který by osoba potřebovala pro splnění této úlohy bez předchozího splnění první úlohy. Tento efekt také pravděpodobně pomůže skupině programátorů více než skupině pracující s ChatGPT. Pro realističtější časové porovnání je tedy lepší porovnávat dobu potřebnou pro splnění všech úloh dohromady.

Dalším problém je spojen s multimodalitou modelu GPT-4. Přestože je ChatGPT s modelem GPT-4 multimodální a dokáže tak zpracovat kromě textu i obrázky, nebyla tato funkcionalita testována během výzkumu této práce. Při výzkumu tak není testován kompletní nástroj ChatGPT, ale pouze jeho schopnosti generování kódu na základě zpracování přirozeného jazyka. Multimodalita by mohla být využita pro nahrání obrázku diagramu tříd. Je ale málo pravděpodobné že osoba, která se snaží něco naprogramovat pomocí nástroje ChatGPT bez technických znalostí programátora bude mít k dispozici diagram tříd navrhovaného programu. Diagramy tříd pro jednotlivé úlohy tak nebyly k dispozici pro ani jednu skupinu. Multimodalita by také mohla být využita při generování uživatelského rozhraní. To by ale vyžadovalo komplikovanější a časově náročnější zadání.

Dalším nedostatkem je, že práce porovnává pouze schopnost neprogramátora s nástrojem ChatGPT a schopnost programátora bez jakéhokoliv nástroje umělé inteligence. Bylo by zajímavé do porovnání také zařadit schopnost programátora s přístupem k ChatGPT nebo jinému nástroji umělé inteligence. Pro takové porovnání by ale byly vhodnější složitější úlohy. Celkový výzkum by také byl náročnější kvůli vyššímu počtu osob. Zlepšováním metody by ale časem byla stejně dosažena metoda navrhovaná na začátku této podkapitoly, jejíž realizace byla v rámci této práce nemožná.

Možným nedostatkem zvolené metodologie byla volba začátku měření času. Čas byl v rámci výzkumu měřen od vytvoření projektu pro danou úlohu nebo začátku konverzace s nástrojem ChatGPT. Díky tomu měl člen výzkumu možnost připravit se a rozmyslet si úlohu před začátkem měření času. To neznamená že k tomuto zkreslení došlo, pouze že je to možné. Lepší alternativou by bylo začít měřit čas od obdržení zadání. Další alternativou je začít měřit čas od přečtení zadání. To ovšem vede k podobnému problému, kdy si člen výzkumu může rozmyslet část řešení, než přečte celé zadání. Malou nevýhodou tohoto způsobu je, že pro zahájení měření času od doby obdržení zadání je nutné se předem domluvit na začátku měření mezi zadavatelem a členem výzkumu.

Hlavní omezení výzkumu by se dalo shrnout jako malé množství členů výzkumu, malé množství výzkumných skupin, nízký počet testovaných nástrojů umělé inteligence a málo rozsáhlé a rozmanité testovací úlohy. Zejména poslední z těchto nedostatků je velmi obtížné eliminovat, jelikož je oblast programování velice rozsáhlá. Existuje několik programovacích jazyků, způsobů programování a typů programů a aplikací, které by bylo nutné otestovat velkým množstvím osob. Takový výzkum by zcela jistě byl vysoce náročný a kompletní test nástrojů umělé inteligence je tak velmi obtížný. Výzkum podobného typu by se také mohl provést i v jiných oblastech než v oblasti programování, což by zřejmě ukázalo na podobné problémy, ale také by se možná objevily problémy nové, dle dané situace a oblasti.

4.2 Zodpovědnost, plagiatorismus a jiná kritika

Zodpovědnost v oblasti AI se stává stále palčivějším tématem, jak technologie postupuje a její aplikace zasahují do širšího spektra lidských aktivit. Jedním z klíčových aspektů této diskuse je otázka přisuzování odpovědnosti za selhání nebo škody způsobené AI systémy. Toto dilema vyvstává z obtížnosti určit, kdo by měl nést odpovědnost. Zda jsou to vývojáři, kteří systém navrhli a implementovali, uživatelé, kteří AI nasadili v konkrétních situacích, nebo dokonce samotný autonomní systém. Výzvou je, že tradiční právní a etické rámce často nezohledňují specifika AI, jako je její schopnost učení a adaptace, což komplikuje určení zodpovědnosti za její akce (Gill, 2023).

Transparentnost a vysvětlitelnost AI systémů jsou nezbytné pro budování důvěry veřejnosti a zajištění jejich kontrolu. Klíčovým prvkem je, aby byly AI systémy navrženy a vyvinuty tak, aby bylo možné jejich rozhodovací procesy pochopit a vysvětlit lidem, kteří tyto systémy používají nebo jsou jimi ovlivněni (Gill, 2023). To zahrnuje, jak schopnost vysvětlit jednotlivá rozhodnutí po jejich provedení, tak i poskytnutí ucelenějšího vhledu do fungování AI na obecné úrovni. Transparentnost systémů AI nejenže usnadňuje identifikaci a opravu chyb, ale také umožňuje hlubší porozumění potenciálním biasům a omezením těchto systémů. Důležitá je zejména ve chvílích, kdy jsou rozhodnutí AI kritická nebo mají značný dopad na lidské životy, jako je diagnóza v medicíně nebo rozhodování v právních systémech.

Generativní AI systémy, jako je GPT-4, představují průlom v možnostech počítačového generování textů, obrázků a dalších forem obsahu. S těmito pokročilými schopnostmi však přicházejí i zásadní etické a právní otázky, zejména v oblasti autorských práv a originality. Když AI generuje obsah, který je značně podobný existujícím dílům nebo který je zcela nový a unikátní, vyvstává otázka, kdo nebo co by mělo být považováno za autora a jak jsou chráněna práva původních tvůrců. Problém se stává ještě složitější, pokud AI vytváří díla inspirovaná chráněnými díly, což může vést k nejasnostem ohledně porušování autorských práv a nároků na vlastnictví. A i pokud AI nevytváří obsah podobný již existujícím dílům, tak bylo na těchto dílech vytrénováno, což je často provedeno bez kompenzace původních autorů (Vynck, 2023). Zpětná kompenzace těchto autorů je však nepravděpodobná.

Dalším kritickým bodem je využívání generativních AI systémů k manipulaci a šíření dezinformací. S rostoucí schopností AI vytvářet obsah, který je nerozeznatelný od obsahu vytvořeného člověkem, se otevírá prostor pro zneužití těchto technologií k šíření falešných nebo zavádějících informací (Gill, 2023). Tento potenciál pro manipulaci se může dotýkat různých oblastí, od politiky až po sociální média, a může mít dalekosáhlé důsledky pro veřejné mínění a demokracii. Otázka bezpečnosti a integrity informací se stává stále naléhavější, s narůstající potřebou vytvářet robustní mechanismy k ověřování pravosti a zdroje obsahu.
Tyto problémy nelze ignorovat a bude potřeba je komplexně řešit formou vytvoření nových zákonů a upravení existujících zákonů. Problémem je, že oblast AI je v tuto chvíli stále mladá a aktuální zákony na ni nejsou připraveny. Dalším problémem je, že se oblast AI aktuálně rozvíjí rychleji, než dokáže jakákoliv zákonodárná organizace zpracovat. Jedná se o velmi komplikovaný problém, který není jednoduché vyřešit a je patrné, že řešení těchto problémů potrvá řadu let.

Téměř deset let po založení OpenAI jako neziskové organizace s cílem udržet AI bezpečnou a široce dostupnou se spoluzakladatelé Elon Musk a Sam Altman ocitli uprostřed veřejného a právního sporu (Robins-Early, 2024). Musk podal na OpenAI žalobu s tvrzením, že společnost porušila svou zakládající dohodu tím, že místo původní mise prospěchu lidstva začala usilovat o komerční úspěch. Přestože OpenAI začala jako open-source nezisková společnost, její dceřiná společnost má dnes

hodnotu přes 80 miliard dolarů a je kompletně closed-source (Robins-Early, 2024). Polovina společnosti je také vlastněná společností Microsoft.

OpenAI reagovalo na Muskovo obvinění s tím, že sám Musk podpořil přesun k ziskovému modelu před odchodem z představenstva v roce 2018 (Robins-Early, 2024). Také obvinili Muska z profesionální žárlivosti, přičemž poukázali na pokusy sloučit OpenAI s Teslou a založit vlastní generativní AI společnost. Přes právní bitvu oba, Musk i Altman občas hovořili s úctou o přínosech druhého pro technologii a AI. Nicméně Musk kritizoval směr OpenAI pod vedením Altmana, naznačuje odchylku od jeho základních cílů a vyjádřil obavy nad etickým vývojem a kontrolou AI technologie.

I když tedy není jisté, jak se tento právní spor bude vyvíjet dál, je patrné, že technologie AI je čím dál cennější a přechod společnosti OpenAI od neziskového formátu bude mít velký vliv na budoucnost oblasti informační technologie a potenciálně i celého lidstva. S přechodem společnosti OpenAI od neziskového formátu jsou všechny hlavní hráči v oblasti AI obrovské soukromé korporace s hodnotou několik miliard dolarů, jako například Google, Microsoft a Amazon. Do nedávna bylo také využití technologie od OpenAI implicitně zakázáno pro vojenské využití v zásadách použití. Tento zákaz byl ovšem ze zásad použití nedávno nenápadně odstraněn (Biddle, 2024).

S růstem oblasti AI také roste možnost hromadné dezinformace ze strany provozovatele. Jak už bylo zmíněno, už v tuto chvíli je možné použít generativní AI pro šíření falešných informací, například generováním nereálných obrázků. Pokud v budoucnu budou AI systémy běžně využívány pro získávání a ověřování informací, tak hrozí možnost alternace těchto informací ze strany provozovatelů těchto systémů. Tato alternace mohou, ale i nemusí být ze zlého úmyslu. Mohou vzniknout například jako vedlejší produkt relativně dobrého záměru. Náznak takového problému se stal nedávno, když společnost Google spustila možnost generování obrázků pomocí jejich AI nástroje Gemini. Společnost Google se snažila eliminovat obecný rasový bias ve vygenerovaných výsledcích. Tato korekce byla ale příliš silná, nepromyšlená a hrubě navrženáa vznikl tak mnohem větší problém. Nástroj Gemini totiž začal

generovat historicky nepřesné a často i nechtěně ofensivní obrázky. Příkladem byla generace osob afroamerického původu jako členů německé nacistické armády v roce 1939, nebo osob domorodého amerického původy jako jednoho z Otců zakladatelů Spojených států amerických. Problém byl natolik diskutovaný, že se společnost Google rozhodla pozastavit funkci generování obrázků pomocí nástroje Gemini (Pequeño, 2024). Podobné méně znatelné alternace ve výsledcích mohou vést k postupné hromadné dezinformaci, zejména pokud se AI stane běžným způsobem pro získání a ověřování informací.

4.3 Dopad na obor aplikované informatiky a vzdělávací systém

Podle Jensena Huanga, ředitele obchodní společnosti Nvidia, se v dnešní době nemá smysl učit programovat, a místo toho by se měli lidé zaměřit na zemědělství, biologii, výrobu a vzdělávání (Collins, 2024). Tvrdí, že díky nástrojům AI bude programování prováděno v přirozeném jazyce, což znamená že každý člověk je rázem programátor. Přestože transformace programování do přirozeného jazyka je možná, tak stále není jisté, že běžná osoba bez technického vzdělání v oboru programování dokáže plně nahradit programátora. Výzkum této práce prokazuje, že potenciál pro tuto možnost existuje, ale pro její potvrzení je potřeba dodatečný výzkum.

Smysluplnější by bylo zakomponovat schopnost práce s nástroji umělé inteligence do běžného vzdělání, a zejména vzdělání v oboru aplikované informatiky a programování. AI se pravděpodobně v blízké době stane součástí velkého počtu profesí, a osoby schopné AI nástroje a systémy efektivně využít tak budou mít velkou výhodu na trhu práce. Je také zcela jasné, že AI nikam nezmizí a není tak důvod ho nevyužít. AI je nástroj a stejně jako jsou kalkulačky používány pro vzdělávání v oboru matematiky, měly by AI nástroje být používány jako součást vzdělávání kdekoliv jsou efektivní, jako například u programování. AI nástroje se už teď stávají součástí vývojových prostředí, takže jejich zakomponování u programování je nezbytné. V jiných oblastech to ale nebude tak přímočaré a bude nutná značná reorganizace oboru. Zakomponování AI systémů do vzdělávacího

systému také potrvá, zejména u oborů, které nejsou přímo spojeny s informačními technologiemi, a proto by bylo dobré začít co nejdříve.

S příchodem nových AI technologií je také nutné upravit způsob vzdělávání a hodnocení výsledků. Je nutné uvědomit si schopnosti AI a zaměřit se na vzdělávání v oblasti, kde AI není efektivní. Například zaměření na tvůrčí myšlení a logickou architekturu a plánování projektů. Také je důležité zadávat studentům úkoly, které AI samo nedokáže plně zpracovat. Naopak schopnosti, které je AI schopné bezchybně provést, už nebudou tak důležité, jako například schopnost psát databázové dotazy. Velká část studentů také bude AI nástroje používat i pokud k této reformě nedojde, což by vedlo k celkovému poklesu efektivity vzdělávacího systému. Proto je nezbytné AI systémy a nástroje co nejdříve zakomponovat do celého vzdělávacího systému a upravit systém tak, aby počítal s existencí AI a tím z části i limitovat efektivitu AI při zpracování vzdělávacích úloh.

I pokud AI kompletně nenahradí programátory, zcela jistě dojde ke zvýšení jejich efektivity při využití těchto nástrojů. S nárůstem této efektivity může dojít ke celkovému poklesu počtu potřebných programátorů, což může stále vést k částečné technologické nezaměstnanosti v tomto oboru. Podobně tomu tak může být u několika jiných oborů, kde AI prokázalo zlepšení efektivity. AI se také vyvíjí a zlepšuje s vysokou rychlostí a je možné, že i pokud v tuto chvíli není reálné, aby AI kompletně nahradilo programátora, tak je jen otázkou času, než se tak opravdu stane.

4.4 Predikce do budoucna

Nelze přesně určit, jak se bude oblast umělé inteligence dál vyvíjet. Je ale patrné, že AI se bude dále zlepšovat a expandovat do nových oborů. Nástrojů AI je čím dál více a jejich soutěž dále navyšuje rychlost jejich vývoje. AI se brzy stane nedílnou součástí každodenního života, ať přímo či nepřímo, a jeho využití je tak nevyhnutelné. Je nutné vzdělávat veřejnost o možnostech využití AI, ale také o jejím nebezpečí, zejména o možnosti šíření falešných informací. Podstatné je zakomponovat nástroje a systémy umělé inteligence jak do právního systému, tak do systému vzdělávacího.

5 Závěry a doporučení

Výzkumem práce byl potvrzeno potenciální riziko technologické nezaměstnanosti v oblasti programování z důvodů náhlého růstu technologií generativní umělé inteligence. Všechny testované úlohy v tomto výzkumu byly efektivněji splněny skupinou neprogramátorů s přístupem k nástroji ChatGPT. Skupina programátorů bez přístupu k nástrojům umělé inteligence odevzdala zdrojový kód na srovnatelné úrovni. Skupina pracující s nástrojem ChatGPT byla však průměrně o 159 % efektivnější co se týče rychlosti zpracování celé sady úloh. To ukazuje na smysl dodatečného výzkumu v této oblasti. Podle výsledků práce tedy existuje možnost nahrazení povolání programátora technologiemi generativní umělé inteligence.

Jedná se o pilotní výzkum v této oblasti, a proto je třeba brát výsledky v kontextu použité omezené metodologie. Výzkum poukazuje na všechny hlavní nedostatky v této metodologii a jak je případně eliminovat. Pro pokročilejší výzkum by bylo nutné rozšířit oblast zkoumání na širší oblasti programování a důkladně tak otestovat více programovacích jazyků, různé typy úloh a různé generativní AI nástroje při větším počtu osob rozdělených do více skupin. Budoucí výzkum v tohoto typu by se také mohl zaměřit i na jiné oblasti než obor programování.

S náhlým růstem oblasti AI je nutné aktualizovat právní a vzdělávací systém a zakomponovat nástroje a systémy AI do vzdělávacího systému. Zásadní je vzdělávat studenty o možnostech a hrozbách umělé inteligence, zaměřit se při vzdělávání na schopnosti které AI nedokáže plně replikovat, a upravit celý vzdělávací systém tak, aby byl připraven na existenci generativních AI nástrojů jako je ChatGPT. Pokud se tak nestane, dojde k celkovému snížení kvality ve vzdělání. Z pohledu právního systému je nutné přesně stanovit zodpovědnost při zneužití umělé inteligence. Kvůli rapidnímu vývoji v oblasti AI je však celkové zakomponování těchto nástrojů do vzdělávacího systému problematické. Tato technologie ale už existuje a nikam nezmizí, a proto je nutné na ní reagovat co nejdříve.

6 Seznam použité literatury

Altman, S. (24. února 2023). Planning for AGI and beyond. https://openai.com/blog/planning-for-agi-and-beyond

Aslam, N. (25. července 2022). Data Representation in Neural Networks- Tensor. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/07/data-representation-in-neural-networks-tensor/

Bala, K., & Ilic, V. (20. prosince 2023). AI-Powered Programming: Using ChatGPT for Code Generation Support.

Biddle, S. (12. ledna 2024). OpenAI Quietly Deletes Ban on Using ChatGPT for "Military and Warfare". The Intercept. https://theintercept.com/2024/01/12/open-ai-military-ban-chatgpt/

Brown, S. (21. dubna 2021). Machine learning, explained | MIT Sloan. https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

Campa, R. (2018). Technological Unemployment. A Brief History of an Idea. 6, 57–79.

Ciornei, S. (12. února 2024). ChatGPT vs. Microsoft Copilot vs. Gemini: Which Chatbot Should You Get? Thereach.Ai. https://medium.com/thereach-ai/chatgpt-vs-microsoft-copilot-vs-gemini-which-chatbot-should-you-get-bb9f2d7fd948

Cipriano, B. P., & Alves, P. (2023). GPT-3 vs Object Oriented Programming Assignments: An Experience Report. PROCEEDINGS OF THE 2023 CONFERENCE ON INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION, ITICSE 2023, VOL 1, 61–67. https://doi.org/10.1145/3587102.3588814

Coello, C. E. A., Alimam, M. N., & Kouatly, R. (2024). Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models. Digital, 4(1), Article 1. https://doi.org/10.3390/digital4010005

Collins, B. (26. února 2024). Nvidia CEO predicts the death of coding—Jensen Huang says AI will do the work, so kids don't need to learn. TechRadar. https://www.techradar.com/pro/nvidia-ceo-predicts-the-death-of-coding-jensen-huang-says-ai-will-do-the-work-so-kids-dont-need-to-learn

Dean, J., Hassabis, D., & Manyika, J. (22. prosince 2023). 2023: A year of groundbreaking advances in AI and computing.

https://blog.research.google/2023/12/2023-year-of-groundbreaking-advances-in.html

Fumo, J. (22. října 2017). A Gentle Introduction To Neural Networks Series—Part 1. Medium. https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

George, D. A. S., George, A. S. H., & Martin, A. S. G. (2023). ChatGPT and the Future of Work: A Comprehensive Analysis of AI's Impact on Jobs and Employment. Partners Universal International Innovation Journal, 1(3), Article 3. https://doi.org/10.5281/zenodo.8076921

Ghosh, B. (12. června 2023). Empowering Language Models: Pre-training, Fine-Tuning, and In-Context Learning. Medium. https://medium.com/@bijit211987/the-evolution-of-language-models-pre-training-fine-tuning-and-in-context-learning-b63d4c161e49

Gill, P. (19. října 2023). Responsible AI: The Art Of Balancing Power With Responsibility. Forbes. https://www.forbes.com/sites/forbesbusinessdevelopmentcouncil/2023/10/19/responsible-ai-the-art-of-balancing-power-with-responsibility/

Gillis, A. S., & Lewis, S. (červenec 2021). What is Object-Oriented Programming (OOP)? App Architecture. https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP

Hanna, D. (2023). The Use of Artificial Intelligence Art Generator „Midjourney" in Artistic and Advertising Creativity. 4, 42–58. https://doi.org/10.21608/jdsaa.2023.169144.1231

Hardesty, L. (14. dubna 2017). Explained: Neural networks. MIT News | Massachusetts Institute of Technology. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

Henkin, D. (2023). Orchestrating The Future—AI In The Music Industry. Forbes. https://www.forbes.com/sites/davidhenkin/2023/12/05/orchestrating-the-future-ai-in-the-music-industry/

Hui, J. (26. října 2020). AI Chips Technology Trends & Landscape (GPU + TPU + FPGA + Startups). Medium. https://jonathan-hui.medium.com/ai-chips-technology-trends-landscape-gpu-tpu-fpga-startups-4798bfad78a2

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021). Evaluating Large Language Models Trained on Code (arXiv:2107.03374). arXiv. https://doi.org/10.48550/arXiv.2107.03374

Ibrahim, M. (28. dubna 2021). What is a Tensor Processing Unit (TPU)? And how does it work? Medium. https://towardsdatascience.com/what-is-a-tensor-processing-unit-tpu-and-how-does-it-work-dbbe6ecbd8ad

Kelly, W. (6. prosince 2023). How to implement AI into cloud management and operations | TechTarget. Cloud Computing. https://www.techtarget.com/searchcloudcomputing/tip/How-to-implement-AI-into-cloud-management-and-operations

Kerner, S. M. (září 2023). What are Large Language Models? WhatIs. https://www.techtarget.com/whatis/definition/large-language-model-LLM

Khan, M. F. A., Ramsdell, M., Falor, E., & Karimi, H. (2023). Assessing the Promise and Pitfalls of ChatGPT for Automated Code Generation (arXiv:2311.02640). arXiv. https://doi.org/10.48550/arXiv.2311.02640

Laker, B. (9. května 2023). AI At The Crossroads: Navigating Job Displacement, Ethical Concerns, And The Future Of Work. Forbes. https://www.forbes.com/sites/benjaminlaker/2023/05/09/ai-at-the-crossroads-navigating-job-displacement-ethical-concerns-and-the-future-of-work/

Marr, B. (19. května 2023a). A Short History Of ChatGPT: How We Got To Where We Are Today. Forbes. https://www.forbes.com/sites/bernardmarr/2023/05/19/a-short-history-of-chatgpt-how-we-got-to-where-we-are-today/

Marr, B. (10. května 2023b). 15 Amazing Real-World Applications Of AI Everyone Should Know About. Forbes. https://www.forbes.com/sites/bernardmarr/2023/05/10/15-amazing-real-world-applications-of-ai-everyone-should-know-about/

Marr, B. (27. prosince 2023c). How Generative AI Will Change All Of Our Jobs In 2024. Forbes. https://www.forbes.com/sites/bernardmarr/2023/12/27/how-generative-ai-will-change-all-of-our-jobs-in-2024/

McFarland, A. (24. února 2024). Midjourney vs. DALL-E: Which AI Image Generator is Better? Techopedia. https://www.techopedia.com/midjourney-vs-dalle

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space (arXiv:1301.3781). arXiv. https://doi.org/10.48550/arXiv.1301.3781

Mills, T. (17. září 2018). AI Vs AGI: What's The Difference? Forbes. https://www.forbes.com/sites/forbestechcouncil/2018/09/17/ai-vs-agi-whats-the-difference/

OpenAI. (30. listopadu 2022). Introducing ChatGPT. OpenAI. https://openai.com/blog/chatgpt

OpenAI. (15. února 2024). Video generation models as world simulators. https://openai.com/research/video-generation-models-as-world-simulators

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., … Zoph, B. (2023). GPT-4 Technical Report (arXiv:2303.08774). arXiv. https://doi.org/10.48550/arXiv.2303.08774

Pandey, A. (17. srpna 2023). GitHub Copilot vs. AWS CodeWhisperer: A Comparative Analysis. Medium. https://medium.com/@pandeyarpit88/github-copilot-vs-aws-codewhisperer-a-comparative-analysis-b8264a12124e

Pawlan, D. (22. prosince 2023). Prompt Engineering: The Next Wave Of Skillsets. Forbes. https://www.forbes.com/sites/forbestechcouncil/2023/12/22/prompt-engineering-the-next-wave-of-skillsets/

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The Impact of AI on Developer Productivity: Evidence from GitHub Copilot (arXiv:2302.06590). arXiv. https://doi.org/10.48550/arXiv.2302.06590

Pequeño, A. (26. února 2024). Google's Gemini Controversy Explained: AI Model Criticized By Musk And Others Over Alleged Bias. Forbes. https://www.forbes.com/sites/antoniopequenoiv/2024/02/26/googles-gemini-controversy-explained-ai-model-criticized-by-musk-and-others-over-alleged-bias/

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training.

Robins-Early, N. (9. března 2024). The feud between Elon Musk and Sam Altman – explained. The Guardian. https://www.theguardian.com/technology/2024/mar/09/why-is-elon-musk-suing-sam-altman-openai

Roller, J. (24. října 2023). How to Use ChatGPT for Coding and Programming. IEEE Computer Society. https://www.computer.org/publications/tech-news/build-your-career/chatgpt-for-coding/

Rong, X. (2016). Word2vec Parameter Learning Explained (arXiv:1411.2738). arXiv. https://doi.org/10.48550/arXiv.1411.2738

Roose, K. (21. října 2022). A.I.-Generated Art Is Already Transforming Creative Work. The New York Times. https://www.nytimes.com/2022/10/21/technology/ai-generated-art-jobs-dall-e-2.html

Roshelova, A. (27. března 2023). Revolutionizing Development: Exploring AI-Powered Code Tools — Copilot and Tabnine. Medium. https://medium.com/@aroshelova.tech/revolutionizing-development-exploring-ai-powered-code-tools-copilot-and-tabnine-6e1a88f1a2d7

Ruby, M. (15. prosince 2023). How ChatGPT Works: The Models Behind The Bot. Medium. https://towardsdatascience.com/how-chatgpt-works-the-models-behind-the-bot-1ce5fca96286

Ruiz, J. (22. února 2024). How AI code generation works. The GitHub Blog. https://github.blog/2024-02-22-how-ai-code-generation-works/

Sachdeva, K. (9. ledna 2024). Top 6 predictions for AI advancements and trends in 2024. IBM Blog. https://www.ibm.com/blog/top-6-predictions-for-ai-advancements-and-trends-in-2024/

Shree, P. (10. listopadu 2020). The Journey of Open AI GPT models. Walmart Global Tech Blog. https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2

Tiku, N., Schaul, K., & Chen, S. Y. (1. listopadu 2023). These fake images reveal how AI amplifies our worst stereotypes. Washington Post. https://www.washingtonpost.com/technology/interactive/2023/ai-generated-images-bias-racism-sexism-stereotypes/

Toews, R. (3. září 2023). Transformers Revolutionized AI. What Will Replace Them? Forbes. https://www.forbes.com/sites/robtoews/2023/09/03/transformers-revolutionized-ai-what-will-replace-them/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention Is All You Need (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762

Vynck, G. D. (17. července 2023). AI learned from their work. Now they want compensation. Washington Post. https://www.washingtonpost.com/technology/2023/07/16/ai-programs-training-lawsuits-fair-use/

Waqas, U. (11. června 2023). Benefits of Using GPT: Advantages and Disadvantages of Chat GPT for Businesses. Medium. https://medium.com/@umerwaqas/benefits-of-using-gpt-advantages-and-disadvantages-of-chat-gpt-for-businesses-696b118bd2d9

Weise, K. (20. listopadu 2023). How Microsoft's Satya Nadella Kept the 'Best Bromance in Tech' Alive. The New York Times. https://www.nytimes.com/2023/11/20/technology/openai-microsoft-altman-nadella.html

What is AI? (24. dubna 2023). McKinsey. https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-ai

What is Deep Learning? (b.r.). IBM. Retrieved 23. února 2024, from https://www.ibm.com/id-en/topics/deep-learning

Wong, M.-F., Guo, S., Hang, C.-N., Ho, S.-W., & Tan, C.-W. (2023). Natural Language Generation and Understanding of Big Code for AI-Assisted Programming: A Review. ENTROPY, 25(6), 888. https://doi.org/10.3390/e25060888

Zewe, A. (9. listopadu 2023). Explained: Generative AI. MIT News | Massachusetts Institute of Technology. https://news.mit.edu/2023/explained-generative-ai-1109

7 Přílohy

Příloha 1

7.1 Seznam zkratek

AI	Artifical Intelligence
LLM	Large Language Model
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
ASIC	Application Specific Integrated Circuit
FPGA	Field-Programmable Gate Array
OOP	Object Orriented Programming

Příloha 2

7.2 Odkaz na cloudový repositář

https://github.com/Arrotanis/thesis

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. František Bílek**
Studium:	I2200593
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Digitální technologie ve vzdělávání**
Název diplomové práce AJ:	Digital technologies in education

Cíl, metody, literatura, předpoklady:

1. Zásady pro vypracování: Cílem práce je popsat nejnovější digitální technologie, přesněji generativní umělou inteligenci, otestovat její možnosti v oblasti programování a posoudit možný dopad těchto technologií na vzdělávací systém a na obor aplikované informatiky. Osnova: 1. Úvod 2. Cíl a metodika práce 3. Výsledky 4. Shrnutí a diskuse výsledků 5. Závěry a doporučení 2. Literatura: Práce bude vypracována za použití vědeckých a žurnalistických článků ze zkoumané oblasti.

Zadávající pracoviště:	Katedra aplikované lingvistiky, Fakulta informatiky a managementu
Vedoucí práce:	doc. Mgr. et Mgr. Marcel Pikhart, Ph.D.
Datum zadání závěrečné práce:	15.10.2023