Tvorba softvéru pomocou umelej inteligencie *

Ľubomír Novotný
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xnovotnyl1@stuba.sk

1. október 2021

Abstrakt

Tento článok sa zaoberá využitím umelej inteligencie pri testovaní softvéru, a najmä používateľského rozhrania. Používateľské rozhranie testované softvérom riadeným udalosťami (event driven software testing) môže ťažiť z využitia AI riešení. Umelá inteligencia výrazne pomohla procesu automatizácie rôznych softvérových procesov. Riešenia pomocou umelej inteligencie znižujú cenu a garantujú lepšiu kvalitu a dôkladné testovanie. Testovanie používateľského rozhrania sa môže pokladať za najnáročnejšiu časť testovania softvéru. Hoci výsledky sú dosť predbežné, ale aplikovanie rôznych AI techník pre testovanie používateľského rozhrania preukázalo prínos ideálnych výsledkov. Pri tvorbe a udržiavaní softvéru nám umelá inteligencia môže podstatne pomôcť.

Kľúčové slová: testovanie softvéru, umelá inteligencia, AI, grafické používateľské rozhranie, GUI

1 Úvod

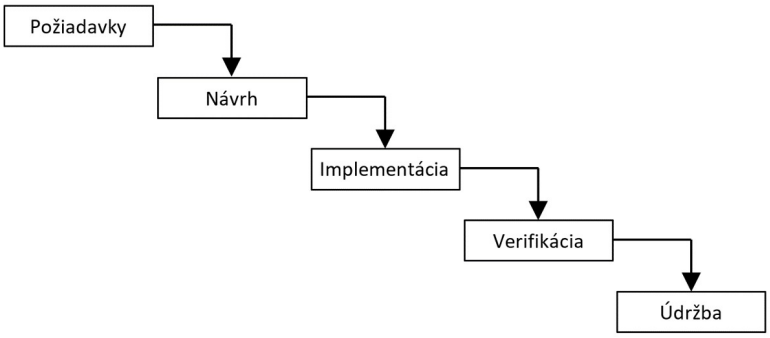
Umelá inteligencia je nový prelom v oblasti informatiky 1.3. Umelá inteligencia má veľký vplyv v posledných rokoch na postupy pri riešení problémov, ale aj na naše každodenné životy. AI používa počítače na riešenie inak neriešiteľných problémov a zlepšenie kvality dočasných počítačových systémov. Dôležitou otázkou je, či môžeme priamo použiť AI na problémy softvérového inžinierstva 1.1, a či procesy softvérového inžinierstva sú schopné využiť umelú inteligenciu 2. Testovanie a podobné aktivity si vyžadujú veľa pozornosti počas celej fázy tvorby softvéru, aby sme mohli vytvoriť požadovaný softvér. Na tento zdĺhavý a komplikovaný problém môžeme použiť umelú inteligenciu. Umelá inteligencia je schopná vykonávať širokú škálu činností, ktorú by inak nebolo možné naprogramovať konvenčným spôsobom. Testovanie používateľského rozhrania je podobné rôznym iným činnostiam, ktoré sme dokázali vyriešiť pomocou umelej inteligencie. V posledných rokoch boli možnosti umelej inteligencie vyšetrované, a boli použité na automatizáciu testovania softvéru.

*Semestrálny projekt v predmete Metódy inžinierskej práce, ak. rok 2021/22, vedenie: Ing. Vladimír Mlynarovič

1.1 Softvérové inžinierstvo

Softvérové inžinierstvo [1] sa zaoberá tvorbou softvérových systémov ale aj efektivitou softvérových systémov. Efektivita je dôležitá, pretože v reálnom svete máme na vývoj a údržbu ohraničené zdroje, stanovené požiadavky a časový limit. Preto môžeme povedať, že softvérové inžinierstvo sa zaoberá tvorbou väčších softvérových systémov, čo najefektívnejším spôsobom. Pri tvorbe sa sústreďujeme na tvorbu čo najspoľahlivejšieho softvéru, pričom sa snažíme čo najlepšie využiť schopnosti softvérových inžinierov. Tvorbu softvéru môžeme realizovať rôznymi spôsobmi. Všetky procesy tvorby softvéru sa vykonávajú v jednotlivých etapách. Tie môžeme nazvať aj ako životný cyklus softvéru. Poznáme viacero takýchto modelov alebo životných cyklusov, ktorými sa môžeme riadiť pri tvorbe softvéru [5]:

- **Vodopádový model** je založený na jednoznačnom vymedzení činností, ktoré na seba nadväzujú a neprelínajú sa. Je vymedzená postupnosť krokov, ktoré za sebou nasledujú. Po poslednom kroku máme náš projekt dokončený.
- **Agilný model** je protikladom vodopádového modelu. Uplatňuje sa v projektoch v ktorých je jasný cieľ, ale nedokážeme presne určiť detailný plán projektu bez priebežných prototypov. Agilný model je interaktívny, pružný a prírastkový. Je závislý na častej spätnej väzbe počas procesu tvorby prototypovania. Nastavujeme si stále nové požiadavky a parametre na základe a skúseností z jednotlivých iterácií.



Obr. 1 Vodopádový model tvorby softvéru podľa [5].

1.2 Testovanie softvéru

Podľa [1]: "testovateľnosť je úsilie, ktoré treba vynaložiť na testovanie vlastností výrobku, napr. či vykazuje požadované správanie."Testovanie programu [3] začína už na začiatku vývoja, keď samotní programátori testujú funkčnosť počas programovania. Konečné testovanie časti softvéru, by nemal vykonávať tvorca softvéru, a malo by sa vykonávať systematicky, aby sa testovaním podrobila čo najväčšia časť programu. Testovanie je pomerne zložitá ale dôležitá oblasť, preto sa testovaniu venuje odborník, a nie programátor. Pri samotnom procese testovania sa softvéroví inžinieri sústreďujú na celý životný cyklus testovania, ktorý začína testovaním funkcií a končí preberacím testovaním a zahŕňa [2]:

- Testy funkcií a modulov - vykonávané prevažne softvérovými inžiniermi počas implementácie.
- Integračné testy - testovanie viacerých modulov súčasne.
- Alfa a beta testovanie - testovanie systému v prostredí skutočnej prevádzky.
- Validačné testovanie - overovanie splnení špecifických požiadavok určených zákazníkom.
- Preberacie testovanie - posledné testovanie pred odovzdaním produktu.

	Kde boli chyby nájdené					
Kde vznikli chyby	Zhromažďovanie požiadaviek, analýza a návrh	Kódovanie a testovanie modulov	Integračné a systémové testovanie	testovanie zákazníkom a Beta testy	Vyexpedovaný produkt	Spolu
Zhromažďovanie požiadaviek, analýza a návrh	**3,5**	**10,5**	**35**	**6**	**15**	**70**
Kódovanie a testovanie modulov		**6**	**9**	**2**	**3**	**20**
Integračné a systémové testovanie			**6,5**	**1**	**2,5**	**10**
Spolu	**3,5**	**16,5**	**50,5**	**9**	**20,5**	100%

Obr. 2 Tabuľka rozloženia chýb a ich objavenia počas vývoja aplikácie podľa [2].

Chyby môžu vzniknúť v každom štádiu vývoja aplikácie [2], vrátane testovania. Vodopádový spôsob vývoja softvéru [3] sa spolieha na testovanie softvéru až po dokončení dizajnu, kódovania a splnenia podmienok. Agilný a iteratívny prístup tvorby softvéru predstavili prototypovanie, ktoré je použité na overenie požiadavok a verifikáciu minulej iterácie. Neustála verifikácia sa snaží využiť formálne metódy a inšpekcie počas procesu tvorby, na rozdiel od testovacej fázy na konci vývojovej fázy.

1.3 Umelá inteligencia

Pod umelou inteligenciou (AI) chápeme softvér, ktorý napodobňuje ľudské kognitívne funkcie tým, že má schopnosť učiť sa a riešiť problémy. Program sa správa tak, že ak by to robil človek, považovali by sme ho za inteligentného. Okrem rozpoznávania reči, obrazov, prekladu do jazykov dokáže imitovať rozmýšľanie ľudí - učiť sa, uvažovať, riešiť problémy, sám hľadať spôsoby, ako sa dostať k cieľu. Je to aj schopnosť počítačov používať algoritmus na učenie sa z dát a získané vedomosti použiť na rozhodovanie podobné ľudskému. Umelá inteligencia ako celok stavia vo veľkej miere na základoch mnohých ďalších vedných odborov, a to predovšetkým na informatike, matematike, štatistike, logike, lingvistike či

neurovedách. Podobnosť umelej a ľudskej inteligencie je ale iba povrchná a nie je správne ich stotožňovať. [7]

2 Používanie AI pri tvorbe a testovaní SW

Umelá Inteligencia môže byť použitá pri vývoji a testovaní nového softvéru, alebo jeho údržbe. Použitie umelej inteligencie má veľa výhod [6]:

- Zredukovanie cyklov tvorby softvéru a testovania, umožňujúc priniesť produkt na trh rýchlejšie.
- Vyššia efektivita testov, plné využitie hardvérových zdrojov.
- Šetrenie na pracovnej sile, zníženie počtu ľudí podieľajúcich sa na teste a znížiť cenu testovania.
- Vyššia stabilita a efektivita testu a testovacieho procesu.
- Zlepšenie presnosti a precíznosti testu, zvýšenie dôveryhodnosti v softvér.
- Softvérové testovacie nástroje sú relatívne jednoduché a môžu priniesť kvalitnejšie výsledky.
- Úlohy ktoré nie je možné vykonať ručne, ale môže to urobiť automatizované testovanie, ako je testovanie záťaže a výkonu.

Za súčasných okolností, automatické testovanie softvéru nedokáže vyriešiť všetky problémy. Automatické testovanie softvéru má nasledujúce obmedzenia a problémy [6]:

- Nedokáže nahradiť manuálne testovanie.
- Manuálne testovanie nachádza viacero defektov ako automatizované testovanie.
- Manuálne testovanie je vysoko závislé od kvality testu.
- Automatizácia testu môže obmedziť vývoj softvéru.
- Samotný nástroj nemá imagináciu, nedokáže sa snažiť nájsť chyby ako človek.

2.1 Rola AI v testovaní softvéru

Rôzne nástroje AI sa používajú na generovanie testovacích údajov , výskum vhodnosti údajov, optimalizáciu a analýzu pokrytia, ako aj správu testov. Na účely testovania softvéru sa použilo mnoho ďalších techník založených na AI [6]. ABC (Artificial Bee Colony Optimization) [4] je optimalizačný algoritmus založený na inteligentnom správaní sa včelieho roja pri hľadaní potravy. ABC algoritmus sa zameriava na miesta v kóde, kde je definovaná alebo použitá premenná, ktorá predstavuje zdroj potravy. V modeli sa kolónia skladá z troch skupín včiel: zamestnaných včiel, pozorovateľov a prieskumníkov. Predpokladá sa, že pre každý zdroj potravy existuje len jedna umelo zamestnaná včela. Zamestnané včely idú k svojmu zdroju potravy a tancujú na tejto ploche. Zamestnaná včela, ktorej

zdroj potravy bol opustený, sa stane prieskumníkom a začne hľadať nový zdroj potravy. Pozorovatelia sledujú tance zamestnaných včiel a vyberajú si zdroje potravy v závislosti od tancov. Vybraný zdroj sa následne otestuje vygenerovanými vstupnými údajmi. Testovanie modelom ABC dosahuje v porovnaní s inými technikami takmer globálnu optimalizáciu s niekoľkými vykonanými testami.

2.2 Rola AI v testovaní GUI

Grafické používateľské rozhrania (GUI) sú jedným z najpotrebnejších prvkov v dnešnom vývoji softvéru. Priebeh testovania GUI aplikácie si vyžaduje obrovské úsilie kvôli zložitosti týchto aplikácií. GUI môže mať aj v malej aplikácii problém obrovského počtu stavov, pretože sa môže generovať príliš veľa testovacích prípadov. Testovanie zahŕňa vypracovanie súboru úloh, vykonanie týchto úloh a porovnanie skutočných výsledkov s očakávanými výsledkami. Technika zahŕňa detekciu reakcie aplikácie na udalosti myši a klávesnice, komponentov, ako sú tlačidlá, lišty ponúk, obrázky, lišty nástrojov, textové polia, reakcie na vstup používateľa atď. Testovanie GUI sa môže vykonávať manuálne aj automatizovane, pričom obe metódy majú svoje výhody a nevýhody [6]. Techniky na automatizované testovanie zahŕňajú generovanie GUI na základe modelu, generovanie testov na základe modelu a automatizáciu generovania testovacích prípadov, aby bolo možné automaticky regenerovať testy pri každej zmene GUI, ktoré modelujú správanie používateľského rozhrania. Mnoho činností v testovaní GUI je potrebné automatizovať, aby boli dané postupy ekonomicky vhodné. V optimálnom prístupe sa použijú inteligentný vyhľadávací agenti (Intelligent Search Agent) (ISA) pre optimálne generovanie testovacej sekvencie a inteligentný agent na optimalizáciu testovacích prípadov (Intelligent Test Case Optimization Agent) (ITOA) pre optimálne generovanie testovacích prípadov. [6]

3 Reakcia na témy

Spoločenské súvislosti. V dnešnej dobe si musíme uvedomiť, ako veľmi je náš svet závislý od výpočtovej technológie. Od obyčajného prehliadania internetu, až po dôležité ekonomické a bezpečnostné úlohy každej krajiny. Chyba v systéme banky môže znamenať, že sa ľudia nedostanú k svojim peniazom, alebo neočakávané správanie softvéru burzy môže vytvoriť ekonomickú depresiu. Preto je testovanie správnosti funkcie daných programov dôležite pre správne fungovanie našej spoločnosti. V posledných rokoch môžeme zaznamenať trend okupovania sa elektronikou. Takzvaný IOT (internet of things) začína vstupovať do našich životov a do našich domovov. Inteligentný asistenti, inteligentné osvetlenie a mnoho ďalších technológií sa dostalo do našich domovov za účelom lepšej kontroly a starania sa o náš domov. Správnosť týchto zariadení je každým dňom dôležitejšia, keďže ich prítomnosť sa neustále zväčšuje.

Historické súvislosti. Vývoj softvéru sa počas histórie informatiky menil. Od samotných programovacích jazykov, písania kódu, funkcionalite programov, až po samotné prístroje pre ktorý bol softvér tvorený. Každá zmena v tomto prostredí nám pomohla tvoriť lepšie a rýchlejšie programy a aplikácie. Jednou

z týchto zmien bolo to, ako testujeme softvér. Softvér je citlivý na každý vonkajší a vnútorný atribút od spôsobu, ako je naprogramovaný, až po zariadenie na ktorom má vykonávať jeho danú činnosť. Donedávna testovanie prebiehalo pomerne konzervatívne. Programátor vyberie časť kódu, a testuje na ňom rôzne vstupy, alebo manuálne testuje používateľské rozhranie, ktoré sa testuje veľmi neefektívne. Postupom času sa pohľad na testovanie zmenil. Začali sa generovať vstupné údaje, používanie softvérov na testovanie, čím sa šetrilo na pracovnej sile, zdĺhavé testovanie sa prenieslo na počítač, aplikovanie rôznych algoritmov na optimalizáciu testov, neskôr sa začala používať aj umelá inteligencia atď. Hoci stále existujú chyby, ktoré nájde len človek, inovácie v testovaní nám taktiež pomohli pri tvorbe softvéru.

Technológia a ľudia. Pri vývoji softvéru programátori neustále kontrolujú ich vlastný kód. Je priam nemožné napísať kód bez chýb, ktoré si programátor neuvedomil pri jeho tvorbe. Preto sa kód neustále testuje celým jeho vývojom, aj po tom, čo už bol dokončený a proces tvorby softvéru sa posunul na inú fázu. Počas písania kódu programátor strávi veľa svojho času hľadaním chýb v kóde, čo je vysoko neefektívne, stojí to veľa nákladov, ale aj psychickej sily samotného programátora. Chyby v kóde sú niekedy také závažné, že viacero programátorov sa musí sústrediť na nájdenie a opravenie chyby v kóde. Zapojenie nástrojov ktoré pomôžu programátorom pri vývoji softvéru sa zefektívni postup a zlepšia sa pracovné podmienky pre programátorov, ktorí môžu vytvoriť lepší produkt pre ľudí, ktorý ho budú používať.

Udržateľnosť a etika. Po vytvorení aplikácie alebo daného softvéru sa práca programátorov nekončí. Väčšina softvérov má po vystavení na trh určitú dobu údržby, v ktorej je daný softvér podporovaný spoločnosťou, ktorá ho vytvorila. Podpora softvéru zahŕňa najmä bezpečnosť softvéru, stabilitu, prípadne aj nové funkcionality. Spomenuté spôsoby testovania môžu pomôcť nie len pri vývoji, ale aj pri údržbe softvéru a jeho častí, pretože pri každej aktualizácií sa softvér musí znova testovať. Často sa chyby objavia aj na miestach, kde v predošlých verziách neboli, preto testovanie pri údržbe softvéru by nemalo byť podcenené a limitované len na zmenené časti softvéru. Lepšie testovacie nástroje nám môžu pomôcť udržiavať softvér s menšími nákladmi, čo by sa malo pozitívne odzrkadliť na dnešnom a budúcom softvéri.

4 Záver

Testovanie je prítomné v každej fáze tvorby softvéru, aj preto je veľmi nákladné a časovo náročné. Hľadať možné zlepšenia a inovácie v tomto smere má teda rozhodne zmysel. Použitie techník AI a rôznych algoritmov sa ukázalo byť vhodným prostriedkom pre vývoj a testovanie softvéru a teda aj GUI. Hoci automatické testovanie softvéru pomocou rôznych nástrojov má aj svoje nevýhody a slabé stránky, vývoj softvéru bude prosperovať pri zavedení týchto nástrojov a techník. Môžeme ho aplikovať v mnohých častiach vývoja, kde zlepšuje kvalitu testovania nášho softvéru. Techniky testovania softvéru obohatené o AI a algoritmy by určite mali byť zahrnuté do procesu testovania.

Literatúra

[1] Mária Bieliková. Softvérové inžinierstvo - princípy a manažment, 2000. ISBN 80-227-1322-8, pages 3-9, Slovenská technická univerzita v Bratislave Fakulta elektrotechniky a informatiky, Ilkovičova 3, 812 19 Bratislava.

[2] Doc. Ing. Pavol Tanuška PhD. Doc. Ing. Peter Schreiber CSc. Proces testovania softvérových produktov the software products testing process. `https://www.mtf.stuba.sk/buxus/docs/internetovy_casopis/2005/5/tanuska.pdf`, 2005. pages 1-2, Katedra aplikovanej informatiky a automatizácie, Materiálovotechnologická fakulta STU, Paulínska 16, 917 24 Trnava.

[3] Brian Fitzgerald and Klaas-Jan Stol. Continuous software engineering: A roadmap and agenda. `https://www.sciencedirect.com/science/article/pii/S0164121215001430`, 2017. pages 176-189, Journal of Systems and Software.

[4] Disha Garg and Abhishek Singhal. A critical review of artificial bee colony optimizing technique in software testing. `https://ieeexplore.ieee.org/document/7542311`, 2016. pages 240-244, 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH).

[5] Dávid Jánosfalvi. Nasadenie nástrojov softvérového inžinierstva pre vývoj softvérového produktu. `http://www.kiwiki.info/index.php/Nasadenie_n%C3%A1strojov_softv%C3%A9rov%C3%A9ho_in%C5%BEinierstva_pre_v%C3%BDvoj_softv%C3%A9rov%C3%A9ho_produktu`, 2020. pages 1-4, Slovenská technická univerzita Materiálovotechnologická fakulta so sídlom v Trnave.

[6] Abdul Rauf and Mohammad N. Alanazi. Using artificial intelligence to automatically test gui. In *2014 9th International Conference on Computer Science Education*, pages 3–5, 2014.

[7] Štefan Trenkler. Umelá inteligencia v medicíne. `http://www.lf.upjs.sk/ceea/doc5/texty/19%20Trenkler%20Umela%20inteligencia%20v%20medicine%20%20CEEA%202019.pdf`, 2019. pages 229-230, CEEA I. klinika anestéziológie a intenzívnej medicíny.