Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Ondrej Špánik

Porovnanie jednotlivých skriptovacích a programovacích jazykov pre urýchlenie práce s GUI

Informačné vzdelávanie

Študijný program: B-INFO4

Vyučujúca: Mgr. Lucia Falbová

December 2019

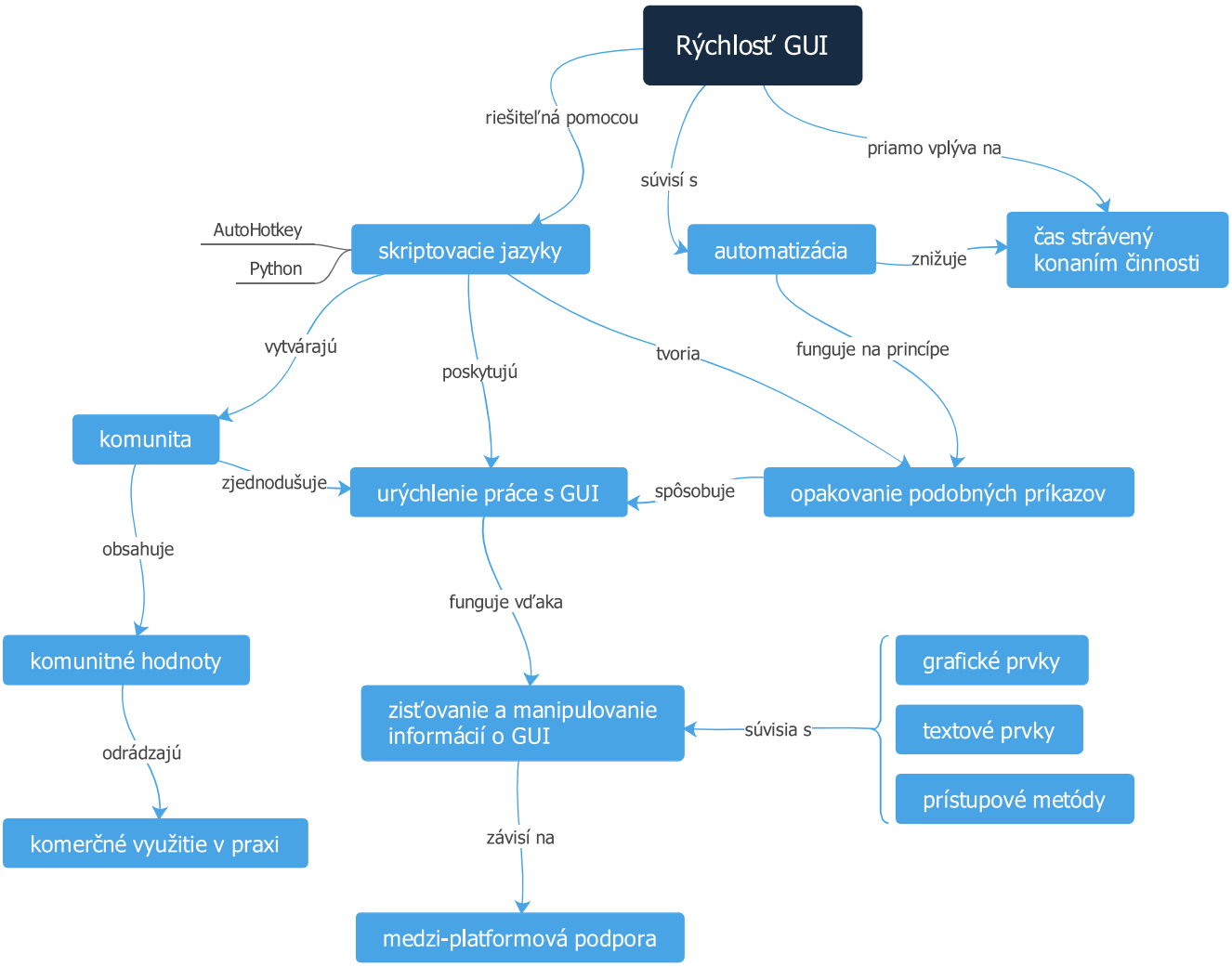
Abstrakt

Spôsoby, ktorými sa dá pozmeniť funkcionalita grafických rozhraní (GUI) pre rýchlejšiu prácu sú limitované a podľa výberu jazyka viac komplexné. Predstavené sú dôvody, kvôli ktorým je automatizácia a úprava GUI prvkov vhodná. Cieľom tejto seminárnej práce je preukázať na reálnych príkladoch, že manipulácia GUI kombinovaná so správnym skriptovacím jazykom môže byť ideálnym nástrojom na dosiahnutie digitálnej automatizácie bez podpory zo strany vývojárov daného GUI.

Kľúčové slová

skriptovacie jazyky, grafické rozhranie, GUI, rýchlosť, efektivita, klávesové skratky, prístupnosť, automatizácia, rozšíriteľnosť, AutoHotkey

Pojmová mapa



Obsah

Zoznam tabuliek a grafov

Zoznam skriptov

Zoznam príloh

Úvod

Porovnanie jednotlivých skriptovacích a programovacích jazykov pre urýchlenie práce s GUI som si zvolil ako tému seminárnej práce preto, lebo po viac ako desiatich rokoch rozvíjania svojich digitálnych schopností vidím, že spôsoby, ktoré sa aplikujú pri tvorbe a používaní grafických používateľských rozhraní (GUI) zostávajú po väčšine nezmenené od ich počiatkov v 90-tych rokoch.

Cieľom tejto seminárnej práce je poukázať na alternatívne možnosti rozšírenia a urýchlenia funkcionality grafických používateľských rozhraní, hlavne zo strany samotného používateľa (čiže bez podpory zo strany vývojárov daného GUI) pomocou rozličných skriptovacích a programovacích jazykov (AutoHotkey, Python, C).

Táto seminárna práca prináša a prezentuje vlastné merania a ukážky použití z danej oblasti, zároveň spomína dôvody, prečo nie sú tieto metodiky ešte štandardizované.

Seminárna práca je členená do štyroch hlavných častí priamo súvisiacich s jej témou, ktoré postupne prechádzajú od dôvodov, prečo vôbec riešiť danú tému, cez príčiny, riešenia, ukážky a vlastné merania, až po zdôvodnenie, prečo táto téma nie je častejšie rozoberaná v spoločnosti.

O téme píšem najmä kvôli tomu, že sa už dlhšie (aspoň 2 roky) venujem vo voľnom čase rozširovaniu existujúcej funkcionality GUI, najmä s cieľom urýchliť svoju prácu.

„If there's a 'trick' to it, the UI is broken."— Douglas Anderson

1. Dôvody, prečo riešiť rýchlosť práce s GUI

Ľudia v digitálnom svete často robia predvídateľné veci, ktoré sa dajú automatizovať, alebo využívajú softvér, ktorý niekedy nenaplní ich očakávania a nemajú možnosti ako rozšíriť jeho funkcionalitu.

Realita v dnešnej dobe je taká, že „softvér často obmieňa svoju vizuálnu stránku a ponecháva koncového používateľa zmäteného, pretože sa funkcionalita, ktorú koncový používateľ väčšinu času využíva presunula pod iné menu, pod-menu alebo zmenila svoj názov“. (Zhang 2013)

Ďalším problémom sú štruktúrne rozdiely medzi aplikáciami, ktoré musia často byť všetky súčasťou každodennej práce používateľa. Jedná sa o rozdielne klávesové skratky a opätovne aj o pomenovania a zaradenia v menu. (Zhang 2013)

Situácia je o to horšia pre ľudí so zdravotným znevýhodnením, ktorí sa snažia v takomto prostredí pracovať. Slepí ľudia majú najviac problémov kvôli fundamentálnym rozdielom v spôsoboch, ako sú dizajnovo spracované a rozložené aplikácie. (Franqueiro 2006)

Z hľadiska automatizácie je problém v nedostatočnej podpore masových príkazov. Ľudia sú nútení potom využívať dodatočný softvér často so svojim vlastným GUI, ktorý dopĺňa túto funkcionalitu. (Intharah 2018)

Problémom sú aj rozdiely v klávesových skratkách s rovnakou funkcionalitou medzi aplikáciami. V dnešnej dobe mimo najznámejších zaužívaných skratiek neexistujú špecifické pravidlá ako ich implementovať. Spoločnosti majú rôzne vízie ako dosiahnuť ideálne rozloženie klávesových skratiek v GUI. (Notepad++ 2019) (JetBrains 2019)

Používatelia síce majú niekedy prístup k možnostiam zmeniť tieto skratky, ale bez schopnosti synchronizovať ich jednoduchým spôsobom medzi ostatnými aplikáciami. GUI v ktorom sa tieto možnosti nastavujú je taktiež súčasťou problému, pretože vždy sa nachádza na inom mieste a má vlastné limitácie. (Zhang 2013)

Bežný používateľ nie je kvôli týmto rozdielom schopný dosiahnuť plnú efektivitu v práci s GUI aplikáciami. (Zhang 2013)

Zaužívanou skratkou je napríklad (Ctrl+S) na uloženie súboru. Problematické je napríklad uloženie všetkých otvorených súborov naraz, kde aplikácie ako Dev-C++ alebo Notepad++

definujú svoje vlastné klávesové skratky, ktoré sa v niektorých prípadoch nedajú prestaviť používateľom, v iných vôbec táto funkcionalita neexistuje.

2. Čím je spôsobená nedostatočná rýchlosť práce s GUI a ako ju zrýchliť?

Vyššie spomenuté problémy sú spôsobené nedostatočnou, či priam neexistujúcou štandardizáciou týchto GUI elementov. Mimo základných prvkov (tlačidlá, popisy, vstupné polia,...), v akomkoľvek GUI frameworku[1] vývojári pracujú, nie je aplikovaná detailnejšia štandardizácia.

Zatiaľ, čo niektoré platformy prídu s pokynmi, ako by mal dizajn a funkcionalita aplikácií smerovať (Apple Human Interface Guidelines 2019), väčšinou tieto rozhodnutia zvyknú spadať pod individuálne spoločnosti.

Rôzne spoločnosti majú tiež rôzne, často konfliktné, vízie ako "zlepšiť" klávesové skratky alebo umiestnenie položiek v menu. (Apple Human Interface Guidelines 2019) (Microsoft Design 2019)

Prípadný dodatočný softvér na masovú automatizáciu GUI tiež komplikuje situáciu, pretože je samotný obeťou nedostatočnej štandardizácie.

Ak chceme schopnosti softvéru rozšíriť bez možnosti inštalácie rozšírení, použitia akéhokoľvek API, existencie dokumentácie, či prístupu k zdrojovému kódu je pri použití správneho jazyka pre GUI automatizáciu tento proces stále pomerne jednoduchý.

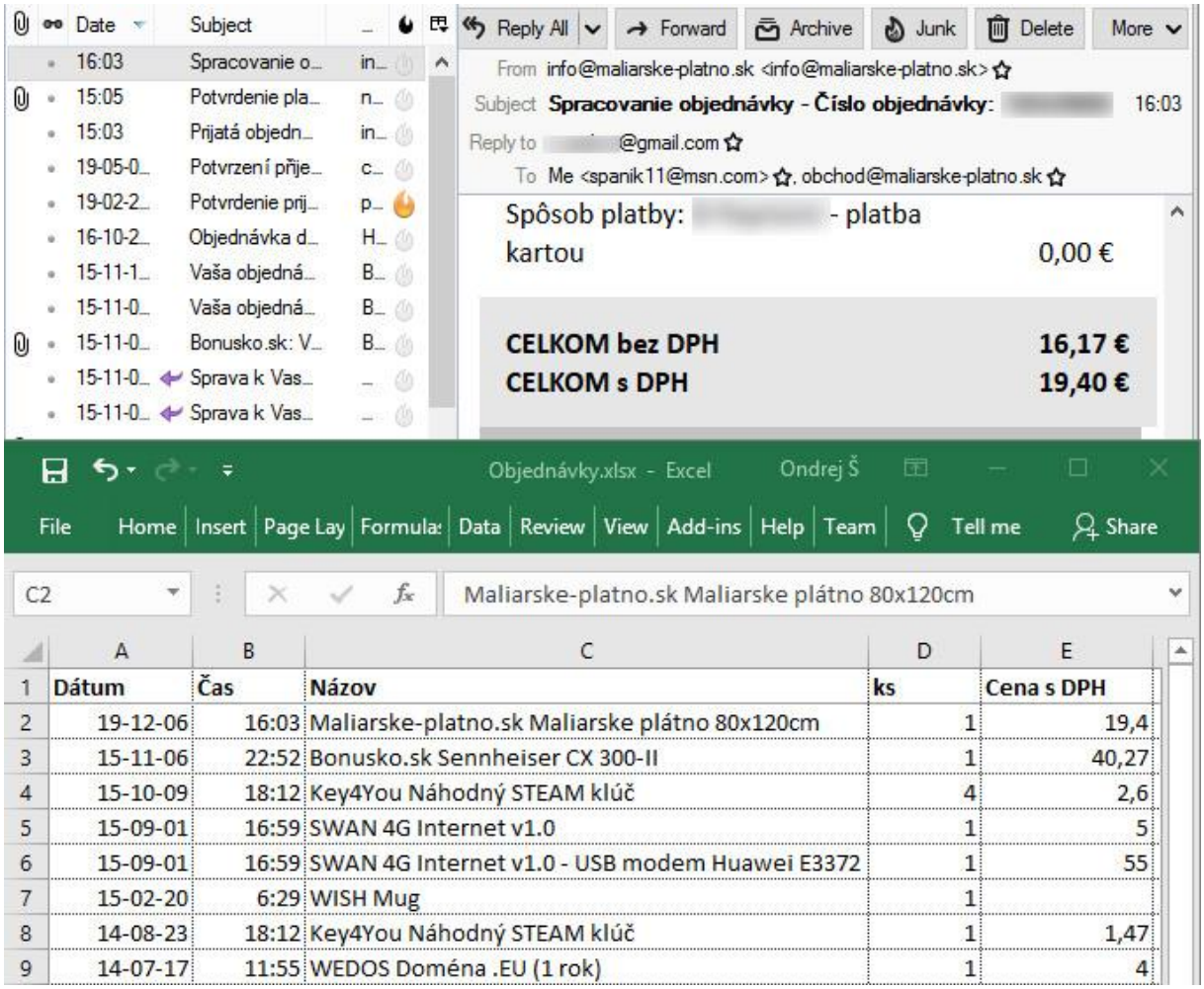
Najlepšie jazyky na dosiahnutie GUI automatizácie sú skriptovacie, nakoľko ich nie je nutné kompilovať a kvôli jednoduchosti kódu sa v nich dá napísať a upraviť funkcionalitu rýchlejšie ako v bežných programovacích jazykoch. (Loui 2008)

Skriptovacie jazyky vyššej úrovne ako napríklad Python je tiež jednoduchšie pochopiť ako klasické programovacie jazyky. (Wainer 2018)

[1] Podporná vrstva pre vývojárov softvéru

3. Príklady využitia skriptov

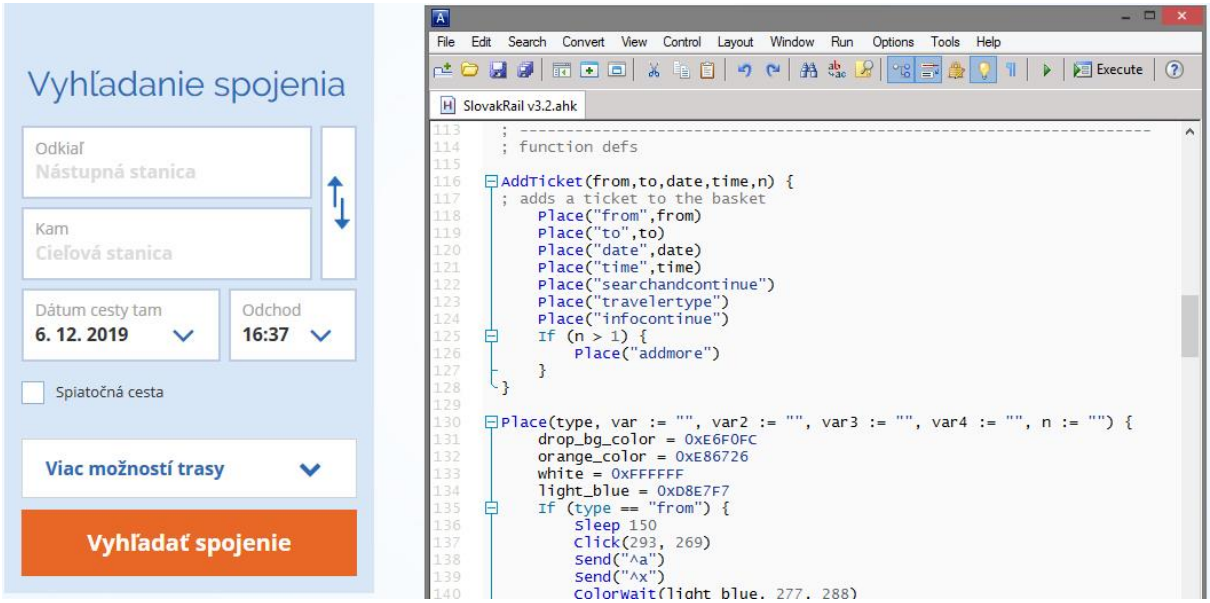
Reálnym príkladom je pracovník v kancelárií, ktorí by bežne kopíroval informácie z e-mailu do Excelu. Ak majú dané e-maily identickú štruktúru alebo iné jednoznačné prvky, dá sa napísať skript, ktorý činnosť **automatizuje**.



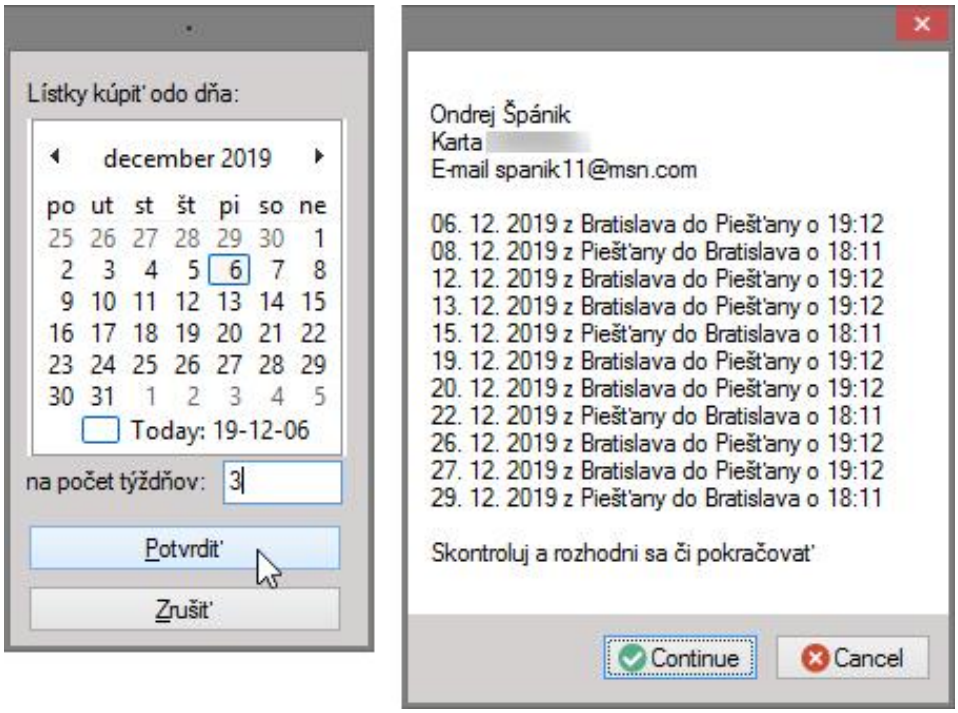
	A	B	C	D	E
1	Dátum	Čas	Názov	ks	Cena s DPH
2	19-12-06	16:03	Maliarske-platno.sk Maliarske plátno 80x120cm	1	19,4
3	15-11-06	22:52	Bonusko.sk Sennheiser CX 300-II	1	40,27
4	15-10-09	18:12	Key4You Náhodný STEAM kľúč	4	2,6
5	15-09-01	16:59	SWAN 4G Internet v1.0	1	5
6	15-09-01	16:59	SWAN 4G Internet v1.0 - USB modem Huawei E3372	1	55
7	15-02-20	6:29	WISH Mug	1	
8	14-08-23	18:12	Key4You Náhodný STEAM kľúč	1	1,47
9	14-07-17	11:55	WEDOS Doména .EU (1 rok)	1	4

Obrázok 1 - Vyťahovanie údajov z Thunderbird do Excel tabuliek

Ďalším príkladom je **opakované nakupovanie tovaru**, či cestovných lístkov v online prostredí bez možnosti jednoduchého masového nákupu, či existencie verejného API pre developerov, ktorí by chceli túto možnosť pridať.

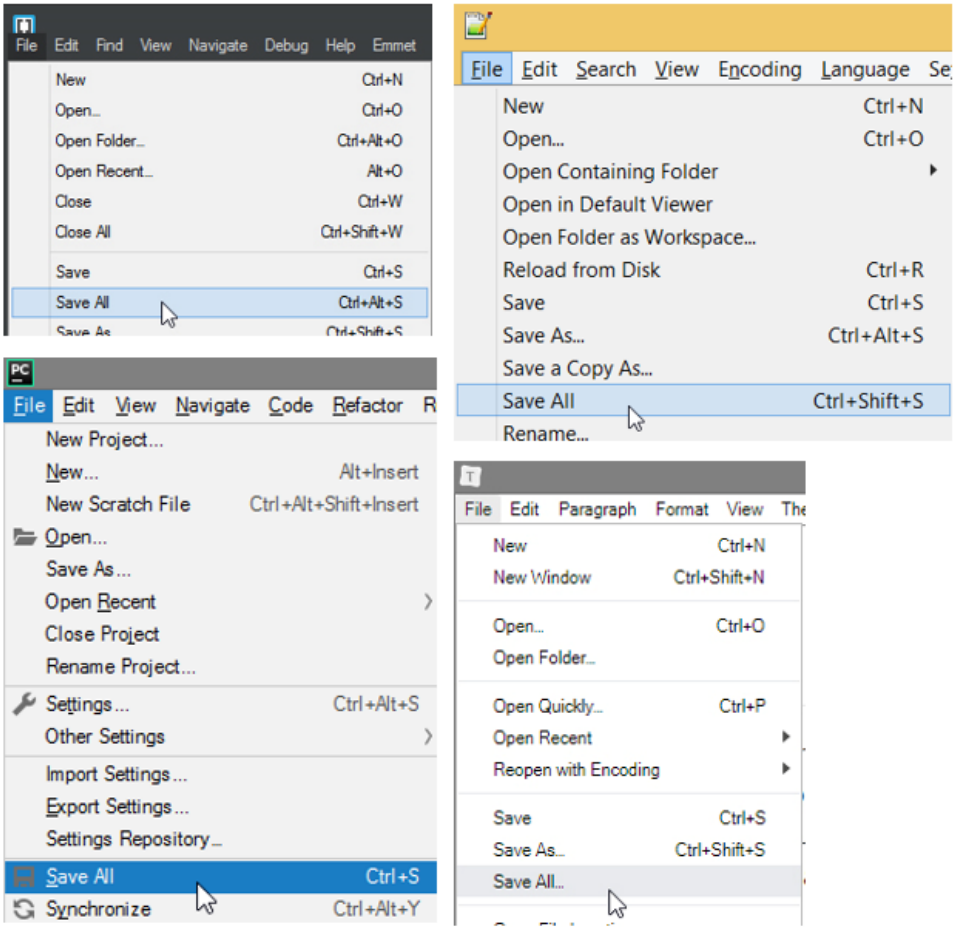


Obrázok 2 - AutoHotkey Skript na masový nákup vlakových lístkov ZSSK



Obrázok 3 - Používateľské rozhranie skriptu na masový nákup vlakových lístkov ZSSK

Častým problémom tiež býva **rozdielnosť klávesových skratiek medzi aplikáciami**, ktoré majú vykonávať identické príkazy. Z vlastnej skúsenosti viem, že tieto rozdiely spôsobujú frustráciu najmä, keď si používateľ nemôže tieto klávesové skratky jednoducho upraviť.



Obrázok 4 - Rozdielnosť v klávesových skratkách medzi aplikáciami Brackets, Notepad++, PyCharm a absencia tejto skratky v aplikácií Typora

rovnake_skratky.ahk

```
#IfWinActive, ahk_exe pycharm64.exe
^!s::^s

#IfWinActive, ahk_exe notepad++.exe
^!s::^+s

#IfWinActive, ahk_exe brackets.exe
^!s::^!s

#IfWinActive, ahk_exe typora.exe
^!s::Send,!{f} {Down 10} {Enter}
```

Obrázok 5 - Jednoduché riešenie problému rozdielnosti klávesových skratiek pomocou AutoHotkey skriptu

4. Porovnanie syntaxe a výhod jednotlivých jazykov

Rozličné skriptovacie a programovacie jazyky sú stavané a optimalizované na rozdielne operácie (viď tabuľku), preto je dôležité porovnať spôsoby ktorými jednotlivé jazyky fungujú, najlepšie v reálne aplikovateľných scenériách.

Každý zo spomenutých jazykov je procedurálny a má rozdielne charakteristiky, ktoré ho vysúvajú do popredia. Python je hlavne známy ako interpretovaný jazyk (čo znamená, že jeho kód netreba kompilovať), C je jazyk s dlhoročnou históriou a tvrdými základmi na ktorom sú stavané rôzne novodobé vetvy (C#, C++, Objective-C) a AutoHotkey nemá presnú definíciu, kombinuje totiž prvky rozličných jazykov.

Tabuľka 1 – Vhodnosť jazykov

Jazyk	C	Python	AutoHotkey
Využívaný na	Komplexné riešenia	Parsing[2], komplexné riešenia	GUI automatizácia
Výhody	Rýchlosť aplikácie	Jednoduchšia syntax oproti C, Rýchle písanie skriptov	Veľmi rýchle písanie skriptov
Náročnosť	Vyššia	Stredná	Nízka
Kompilácia	Povinná	Dobrovoľná[3]	Dobrovoľná kompilácia, Text skriptu je avšak stále čitateľný[4]
Platformy	Windows, Linux, macOS, mobilné platformy	Windows, Linux, macOS, mobilné platformy	Len Windows

4.1. Unifikácia klávesových skratiek medzi rozličnými aplikáciami

Nasledujúca ukážka skriptov unifikuje klávesové skratky pre otváranie privátneho (inkognito) okna v prehliadačoch Google Chrome (v. 78), Chromium a Mozilla Firefox (v. 70).

Jazyk **AutoHotkey** je na prácu s klávesovými skratkami priam stavaný a ako už môže z jeho názvu vyplývať, presne kvôli nej vznikol. V nasledujúcom skripte sú pomocou jednoduchej syntaxi definované akcie pre jednotlivé skratky. V tomto prípade je pre klávesu

[2] Schopnosť vyberať z dát špecifické prvky na základe zadaných kritérií
[3] Interpretované jazyky, do ktorých sa Python radí, nemusia byť kompilované
[4] Väčšina skompilovaných AutoHotkey skriptov je čitateľná v zložke RCDATA v extrahovanom .exe súbore

`F1` definované, aby namiesto jej bežnej akcie (v prehliadači Google Chrome / Chromium otvorenie stránky *Pomoc a podpora*) vlastne stlačila klávesy `^+p` (t.j. `Ctrl+Shift+P`) a `^+n` (t.j. `Ctrl+Shift+N`) podľa toho v akom prehliadači je sama stlačená.

Skript v jazyku AutoHotkey

```
; Keďže Chrome aj Chromium majú rovnakú skratku na inkognito režim,
; môžeme využiť zoskupenie pomocou GroupAdd / ahk_group
GroupAdd skupina_prehliadacov, ahk_exe chrome.exe
GroupAdd skupina_prehliadacov, ahk_exe chromium.exe

#IfWinActive, ahk_exe firefox.exe
F1::^+p

#IfWinActive, ahk_group skupina_prehliadacov
F1::^+n
```

Jazyk **Python** by mal byť v tomto prípade ideálny ako medzi-platformové riešenie alebo ako súčasť väčšieho projektu, v ktorom sa očakáva vysoká úroveň stability. Ideálne je dosiahnuť túto funkcionalitu vložením knižnice `pywinauto` na manipuláciu okien iných aplikácií v prípade operačného systému Windows a ďalšej knižnice pre priraďovanie globálnych klávesových skratiek (t.j. skratiek, ktoré fungujú medzi aplikáciami). (PyWinAuto 2018)

Problémom avšak je, že väčšina relevantných knižníc nebola dlhú dobu aktualizovaná a nie je kompatibilná s Python verziou 3. Pre Windows operačné systémy by to bola knižnica `pyhk` v Python 3 kompatibilnej verzií `pyhk3` , pre Linux s manažmentom okien cez X systém knižnicu `keybinder`. (PYHK 2017) (keybinder 2017) Keďže `pyhk3` nemá dostatočnú dokumentáciu a knižnica `keybinder` nie je kompatibilná s operačným systémom Windows, je nutné buď nájsť a použiť inú knižnicu, ktorá je zároveň aktualizovaná alebo stráviť neúmerné množstvo času aktualizovaním a nasadzovaním týchto knižníc. Ďalšia relevantná `system_hotkey`, ktorá sľubuje jednoduchú medzi-platformovú implementáciu, má veľmi zle spísanú dokumentáciu s množstvom preklepov. (PyPI system_hotkey 2016)

Vďaka knižnici `pynput` bola implementácia globálnych klávesových skratiek nakoniec možná, no proces hľadania tejto knižnice a nutnosť definovať viacero funkcií na jej sprevádzkovanie sú hlavnými nevýhodami v porovnaní s viac kompletným a k tejto téme prispôsobeným jazykom ako AutoHotkey. (pynput 2019)

Základ skriptu v jazyku Python

Z knižnice `pynput` treba importovať modul `keyboard` a následne vytvoriť načúvacie vlákno, ktorému sa definujú individuálne funkcie pre stlačenie a zdvihnutie každej klávesy používateľa.

```
from pynput import keyboard

def stlacenie(klavesa):
   pass

def zdvyhnutie(klavesa):
   pass

with keyboard.Listener(on_press=stlacenie, on_release=zdvyhnutie) as listener:
   listener.join()
```

Už v tomto bode je jasné, že to čo bolo v jazyku AutoHotkey priamočiare bude v jazyku Python komplexné, keďže už v tomto bode začína skript na zjavne jednoduchú činnosť ako nastavenie klávesovej skratky byť rozsiahly.

Nasledovne je nutné funkciám `stlacenie` a `zdvyhnutie` priradiť podmienky, ktoré budú danú klávesu porovnávať a zisťovať či nie je identická s klávesmi, ktoré používateľ zadal ako hľadané kombinácie v poli `kombinacie`. (PyPI pynput 2019) V tejto ukážke sú hľadané kombinácie `Shift+A` a `Shift+B`:

```
kombinacie = [
  {keyboard.Key.shift, keyboard.KeyCode(char='A')},
  {keyboard.Key.shift, keyboard.KeyCode(char='B')}
]
```

Ďalšími krokmi by bolo implementovať už spomínané podmienky, hľadať klávesové skratky len v oknách prehliadačov Chrome / Chromium a Firefox a vykonávať akcie iných klávesových skratiek. Aj bez dokončenia tohto skriptu je jasné, že pre bežného človeka by prácu s grafickými rozhraniami urýchlil len po vynaložení neúmerného množstva námahy a času.

V jazyku **C** je tvorba tohto druhu skriptu ešte komplexnejšia, keďže jednotlivé knižnice je ešte ťažšie nájsť a ich používanie vyžaduje ešte viac úsilia. Medzi-platformové riešenia musia byť programované separátne, keďže každá platforma je závislá na kompletne iných knižniciach. V prípade operačného systému Windows je jedným z nepriamych príkladov funkcia `RegisterHotkey` (Microsoft 2018), ktorá síce znie jednoducho, no po prečítaní v tomto prípade vynikajúco spísanej dokumentácie bude používateľ pravdepodobne kompletne zmätený, keďže k jej pochopeniu je nutné mať nielen dlhoročné skúsenosti s programovaním.

4.2. Zatvorenie všetkých okien okrem aktívneho

Skript v jazyku AutoHotkey, ktorý zobrazí zoznam kandidátov na zatvorenie

```
; Získame zoznam ID všetkých otvorených okien
WinGet open_windows, List

; Prechádzame cez celý zoznam otvorených okien
Loop %open_windows% {

   id := open_windows%A_Index%             ; ID okna je pod premennou open_windows%A_Index%
                                           ; A_Index je zabudovaná premenná Loop iterácií

   WinGetTitle wintitle, ahk_id %id%       ; Získa titul okna
   WinGet name, ProcessName, ahk_id %id%   ; Získa názov procesu okna

   ; Nasledujúce riadky pridajú do logu ID všetkých otvorených okien
   ; Okná bez názvu a okno "Program Manager" (čo je vlastne pracovná plocha) nezatvárame
   ; Pomocou zabudovanej funkcie WinActive zistíme či je okno aktívne
   log .= "`n" . id
   If WinActive("ahk_id" . id)
       log .= " <-- je aktivne okno, urcite nezatvarat"
   Else If (wintitle == "") || (wintitle == "Program Manager")
       log .= " <-- nezatvarat"
   Else
       log .= " <-- " . name . ""
}
MsgBox % log
```

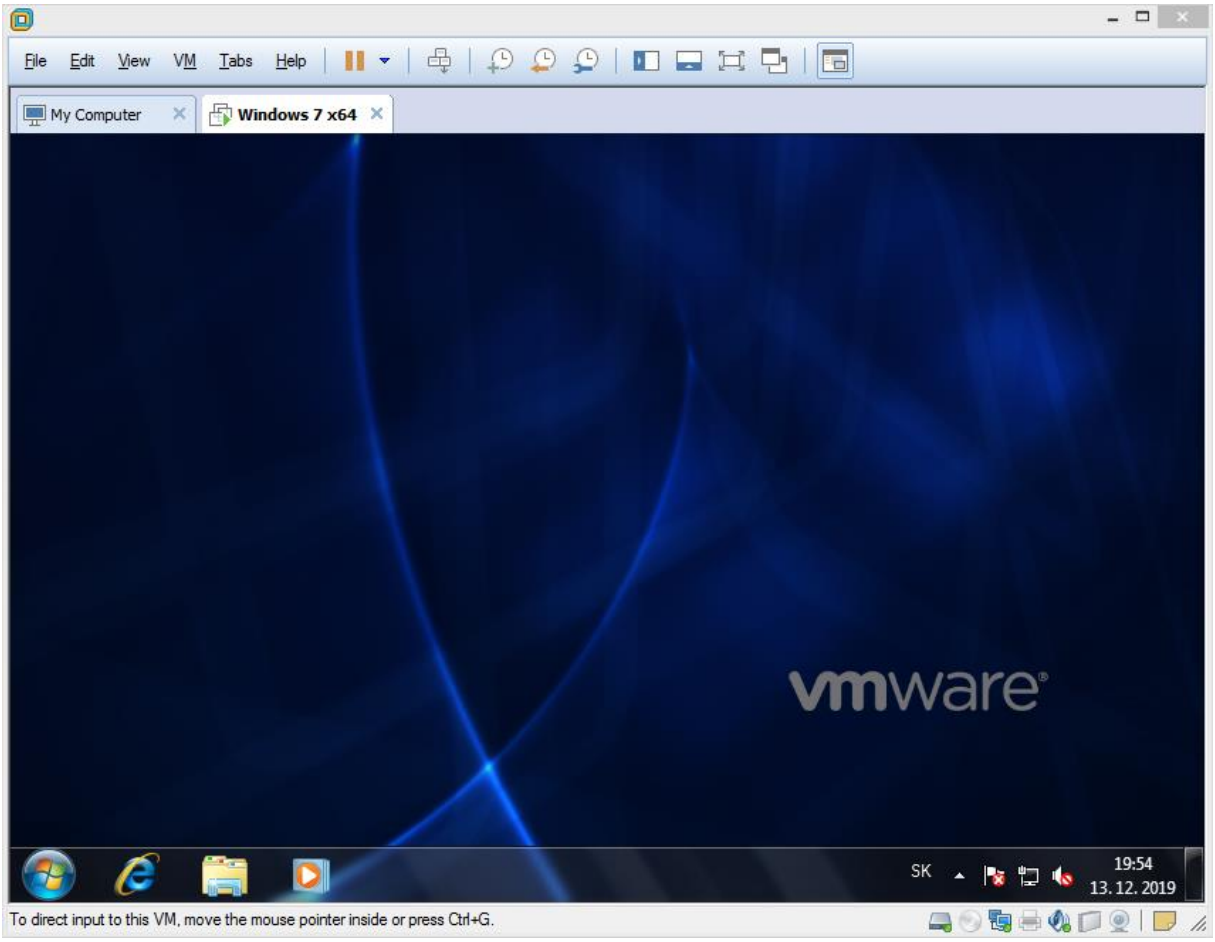
Skript v jazyku AutoHotkey, ktorý neaktívne okná zatvorí

```
; Získame zoznam ID všetkých otvorených okien
WinGet open_windows, List

; Prechádzame cez celý zoznam otvorených okien
Loop %open_windows% {

   id := open_windows%A_Index%             ; ID okna je pod premennou open_windows%A_Index%
                                           ; A_Index je zabudovaná premenná Loop iterácií

   WinGetTitle wintitle, ahk_id %id%       ; Získa titul okna
   WinGet name, ProcessName, ahk_id %id%   ; Získa názov procesu okna

   ; Okná bez názvu a okno "Program Manager" (čo je vlastne pracovná plocha) nezatvárame
   ; Pomocou zabudovanej funkcie WinActive zistíme či je okno aktívne
   ; Zabudovaná funkcia WinClose zatvorí okno
   If (wintitle != "") && (wintitle != "Program Manager") && !WinActive("ahk_id" . id)
       WinClose ahk_id %id%
}
```

Jazyk C sa v tomto prípade dá využiť pri urýchlení grafických operácií, ktoré by mohli vzniknúť v systémoch, ktoré neumožňujú zistiť stav aktívnosti okna, napríklad automatizovaný vstup z hlavného operačného systému do virtuálnych systémov cez softvér ako VMware alebo Oracle VirtualBox.

4.3. Grafické hľadanie špecifického prvku v okne

V nasledujúcom skripte bude hľadané tlačidlo Štart menu vo virtuálnom systéme Windows 7 bežiacom na VmWare Workstation, s tým, že skript bude spustený z hosťovského systému a teda kompletne oddelený od akejkoľvek činnosti vo virtuálnom systéme.



Obrázok 6 - Okno virtuálneho systému v ktorom bude hľadané tlačidlo štartu

Prvok je hľadaný na základe obrázka `orb.png` uloženého v adresári skriptu.

Obrázok 7 - orb.png

Skript v jazyku AutoHotkey

```
#SingleInstance Force
okno := "Windows 7 x64 - VMware"
hladany_obrazok := "orb.png"

; ak neexistuje okno nepokracuje
If !WinExist(okno) {
   MsgBox Okno %okno% neexistuje
   ExitApp
}

; ziska sirku a vysku okna
WinGetPos,,,okno_w,okno_h,%okno%

; aktivuje okno
WinActivate, %okno%
WinWaitActive, %okno%

; hlada v aktivnom okne poziciu hladaneho obrazka s toleranciou odtienov farby 5 z 255
ImageSearch, hladane_x,hladane_y,0,0,%okno_w%,%okno_h%,*5 %hladany_obrazok%

; vypise poziciu hladaneho obrazka
if ErrorLevel = 2
   MsgBox Nebolo mozne uskutocnit hladanie.
else if ErrorLevel = 1
   MsgBox Hladany obrazok sa nenasiel.
else
   MsgBox Hladany obrazok sa nasiel na sirke %hladane_x% a vyske %hladane_y%.
```

Skript v jazyku Python

```
import pyautogui

ho = "orb.png"                      # hladany obrazok
hs = pyautogui.locateOnScreen(ho)  # hladane suradnice

if hasattr(hs, "left") and hasattr(hs, "top"):
   print("Hladany obrazok sa nasiel na sirke " + str(hs.left) + " a vyske " + str(hs.top))
else:
   print("Hladany obrazok sa nenasiel")
```

Výstup

```
Hladany obrazok sa nasiel na sirke 20 a vyske 602
```

V tomto Python skripte je problémom úzka podpora funkcionality knižnice `pyautogui`, ktorá obsahuje funkcie najmä na hľadanie prvkov a nemá schopnosť limitovať vyhľadávanie na isté okno, okrem prebratia súradníc pre limitovanie hľadania na región na obrazovke, ktoré sa dajú nadobudnúť nejakou inou knižnicou. V jazyku AutoHotkey je táto funkcionalita priamo zabudovaná a funkcia `ImageSearch` počíta len s vyhľadávaním v aktívnom okne, pokiaľ nie je špecifikované inak pomocou funkcie `CoordMode`. (AutoHotkey 2019)

5. Medzi-platformové (cross-platform) riešenia

V dnešnej dobe neexistuje jednotná podpora GUI automatizácie medzi rôznymi platformami, nakoľko každá využíva iné pracovné prostredia. Napríklad len v desktopových distribúciách Linuxu má používateľ na výber medzi viacerými (Gnome, XFCE, KDE, Budgie, Cinnamon,...) s tým, že každé rieši grafické prvky rozdielne. (GNOME 2019) (XFCE 2019) (KDE 2019)

Táto problematika odradzuje vývoj knižníc a jazykov určených na GUI automatizáciu. Napríklad jazyk AutoHotkey prešiel viacerými iteráciami a pokus portovať tento jazyk na iné platformy cez .NET implementáciu známy ako „IronAHK“ zlyhal kvôli nedostatočnému záujmu zo strany developerov. (IronAHK 2011)

Koncept kombinácie medzi-platformovej a GUI automatizácie avšak nie je kompletne stratený. Projekty ako knižnica PyAutoGUI pre Python sú v momentálnej dobe aspoň čiastočne úspešné, avšak ich funkcionalita je vysoko limitovaná. PyAutoGUI je po vizuálnej stránke schopné pracovať s obrazovkou len ako jedným veľkým celkom, bez schopnosti čítať informácie o oknách, či vôbec detegovať individuálne okná. (PyPI PyAutoGUI 2019)

6. Merania

6.1. Výpočet efektivity grafických skriptov medzi jazykmi

Systém nebol počas priebehu meraní používaný, okrem meracích aplikácií otvorených na pozadí. Viditeľný prvok bol vždy na rovnakej pozícií (20;602). Je očakávané, že funkcie v oboch jazykoch skončia hneď po nájdení prvku. Ak prvok nebol viditeľný, na usúdenie, že sa nedá nájsť museli preskenovať aj zvyšok plochy obrazovky.

Funkcia v jazyku AutoHotkey predvolene prehľadáva len aktívne okno a aby pracovala podobným spôsobom ako už spomenuté funkcie knižnice PyAutoGui v Pythone, použitý bol príkaz `CoordMode Pixel, Screen` a funkcie `WinActivate` a `WinWaitActive` vypnuté. Keďže funkcia v Pythone skenuje len primárny monitor, v AutoHotkey stačilo použiť zabudované premenné `A_ScreenWidth` a `A_ScreenHeight` pre limitovanie na rozmery primárneho monitora. Posledná úprava v AutoHotkey skripte spočíva vo výmene funkcie `MsgBox` za `Tooltip` ktorý nečaká na potvrdenie od používateľa pred pokračovaním (resp. skončením skriptu) a nebrzdí meranie.

Na meranie boli použité vlastné meracie programy `max_trvanie.c` a `trvanie.c`. Ich zdrojový kód je súčasťou príloh.

Tabuľka 2 – Meranie rýchlosti vybraných grafických výpočtov medzi jazykmi

N = 10	AutoHotkey (ImageSearch)		Python (pyautogui.locateOnScreen)	
Stav hľadaného prvku	viditeľný	neviditeľný	viditeľný	neviditeľný
Externé trvanie spolu (s)	1,984	2,125	13,188	19,437
Min ext. trvanie (s)	0,172	0,187	1,171	1,719
Max ext. trvanie (s)	0,234	0,250	1,407	2,172
Rozdiel min a max trvania	0,062	0,063	0,236	0,453

Z meraní je jasne vidieť časový rozdiel keď funkcia hneď skončila po nájdení prvku alebo pokračovala ďalej po koniec obrazovky ak prvok nebolo možné nájsť.

Rýchlosť meraného AutoHotkey skriptu je takmer 10 násobne vyššia ako rýchlosť Python skriptu. Spôsobené to je najmä tým, že knižnica `PyAutoGui` sa spolieha na ďalšie knižnice

PyScreeze a Pillow a že zabudované AutoHotkey funkcie sú už v binárnom kóde, stačí ich len zavolať s argumentami v skripte.

Pre koncového používateľa predstavujú tieto rozdiely najlepší dôvod, prečo sa oplatí automatizovať GUI pomocou viac kompletného jazyka, než hľadať jednotlivé knižnice a nevedieť zaručiť ich rýchlosť a stabilitu.

Meraný skript v jazyku AutoHotkey

```
#SingleInstance Force
CoordMode Pixel, Screen

hladany_obrazok := A_ScriptDir . "\orb.png" ; absolutna cesta k hladanemu obrazku

; hlada na vsetkych monitoroch poziciu hladaneho obrazka
ImageSearch, hladane_x,hladane_y,0,0,A_ScreenWidth,A_ScreenHeight,*5 %hladany_obrazok%

; vypise poziciu hladaneho obrazka
if ErrorLevel = 2
   Tooltip Nebolo mozne uskutocnit hladanie.
else if ErrorLevel = 1
   Tooltip Hladany obrazok sa nenasiel.
else
   Tooltip Hladany obrazok sa nasiel na sirke %hladane_x% a vyske %hladane_y%.

ExitApp
```

Meraný skript v jazyku Python

```
import pyautogui

ho = "C:/Desktop/orb.png"            # absolutna cesta k hladanemu obrazku
hs = pyautogui.locateOnScreen(ho)    # hladane suradnice

if hasattr(hs, "left") and hasattr(hs, "top"):
   print("Hladany obrazok sa nasiel na sirke " + str(hs.left) + " a vyske " + str(hs.top))
else:
   print("Hladany obrazok sa nenasiel")
```

7. Súčasná implementácia v spoločnosti a možnosti jej rozšírenia

V dnešnej dobe je rozširovanie funkcionality či už GUI alebo iných častí softvéru rozšírené najmä medzi internetovými komunitami s niche[5] zámermi. Rôzny autori v rámci vlastného záujmu pracujú na rozšíreniach a podporných vrstvách pre rôzne skriptovacie jazyky, ktoré následne šíria pre jazyk AutoHotkey najmä cez komunitné fóra AutoHotkey. (AutoHotkey Boards 2019)

7.1. Problém podpory

Keďže sa jedná najmä o komunitné projekty, poväčšine s otvoreným zdrojom, ktoré nie sú šité na mieru a nie je venovaný čas zaručeniu ich kompatibility a stability, výsledkom skriptov je nutnosť opätovne skripty kontrolovať, upravovať a často-krát aj kompletne prepisovať kvôli nízkej kompatibilite medzi rôznymi zariadeniami, ktorá je najmä časovo spôsobená malými zmenami v používateľských rozhraniach GUI.

Tento problém poznám z vlastnej skúsenosti, keďže skript opätovného masového nakupovania tovaru (konkrétne vlakových lístkov), ktorý bol uvedený v ukážkach skriptov musel byť prepisovaný v priebehu dvoch rokov už tri krát. Najviac rozšíreným dôvodom prepisovania boli zmeny zo strany obchodníka, kvôli ktorým som skript musel vždy nečakane upravovať.

Komunitný zámer projektov, ku ktorým prispievajú používatelia s rôznymi pozadiami taktiež značí nezáujem poskytovať komerčné riešenia, alebo problém nájsť vhodný kompromis medzi komunitným vnímaním, ak by sa projekty rozhodli komerčnú cestu vyskúšať.

7.2. Potenciálne legálne problémy

Ešte ťažšie ako už spomenuté problémy je zaručiť aby skripty spĺňali kritériá autorského práva. Prvý odsek sekcie § 89 autorského práva, ktorý hovorí o tom, že používateľ „*nesmie používať počítačový program v rozpore s bežným využitím*“ (Zákon 185/2015 Z. z.) je len jedna z mnoho, ktoré by bolo nutné pri skriptoch obhájiť.

Dobrým príkladom tohto problému je pridanie masovej funkcionality. Pôvodná myšlienka autora softvéru mohla byť, že masová funkcionalita nebude priamou súčasťou softvéru

[5] Z angl. „niché“ – neoslovuje veľké množstvo ľudí

z komerčných, alebo iných dôvodov a pridanie takejto funkcionality koncovým používateľom pomocou skriptu by mohlo byť v rozpore s už spomenutým bežným využitím.

7.3. Uplatnenie v praxi

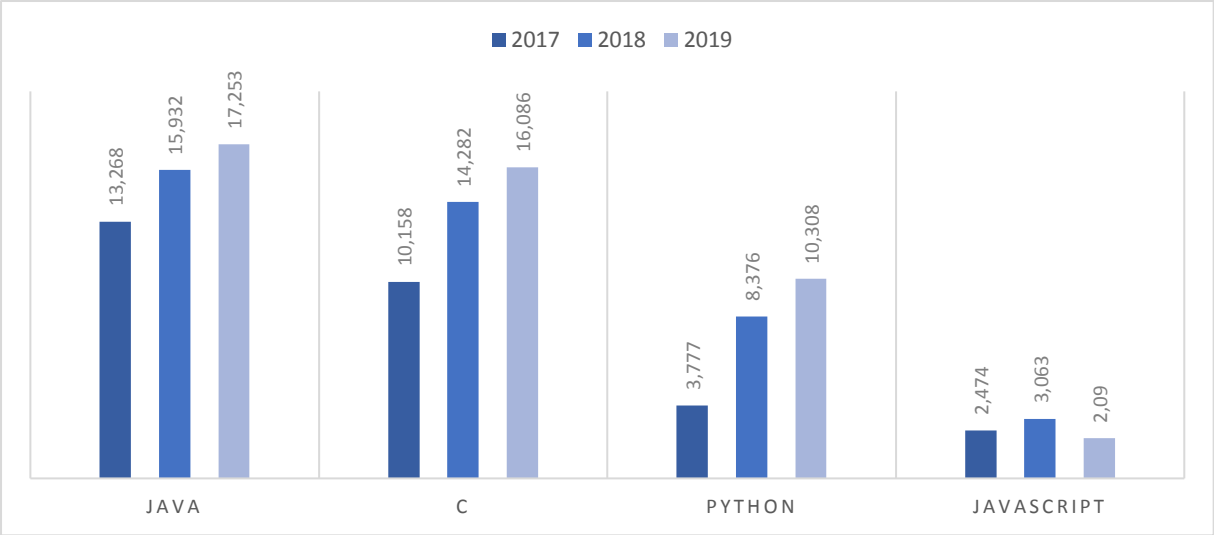
Výsledkom vyššie spomenutých problémov je nízka šanca, že by sa automatizácia vo forme zrýchlenia práce s GUI dala komerčne uplatniť. Pre jazyk AutoHotkey komerčná budúcnosť rozhodne nie je prvoradá, keďže ho zastrešuje organizácia, ktorej cieľom je „*zabrániť monetizácii*“. (AutoHotkey Foundation 2019)

Svetovo je avšak komerčne využívané množstvo skriptovacích jazykov, každý s inými cieľmi, ktoré sa často prekrývajú.

Komerčné riešenie pre GUI automatizáciu môže spočívať v limitovanejších podporných programoch automatizácie, v správe pod spoločnosťou s nadriadenou kontrolou. Príkladom tohto spôsobu je aplikácia Apple Shortcuts. (Apple Shortcuts User Guide 2019)

Tabuľka 3 - Uplatnenie rozličných skriptovacích jazykov

Skriptovací jazyk	**Zameranie**
Perl	Využitie v širokej škále odvetví ako weby, siete, automatizácia, GUI, databázy. Bol nahradený vo veľkej miere jazykom Python. (Balakrishnan 2018)
JavaScript	Tradične využívaný na webové rozhrania zo strany klienta. V dnešnej dobe už aj zo strany servera pomocou node.js. (node.js 2019)
Bash shell skripty (.sh súbory)	Základný druh skriptu v Linuxe a macOS. Je využívaný veľmi často pre urýchlenie práce s terminálom a pri inštalácií rôznych knižníc a programov. (Both 2019)
batch (.bat súbory)	Základný druh skriptu v operačnom systéme Windows. Nie je využívaný až tak často ako bash v Linuxe, avšak jeho princíp je rovnaký. Novšie a komplexnejšie od neho sú Powershell skripty. (Microsoft 2019)
VBScript	Odvodený od jazyka Visual Basic, cieľovou platformou bol web. V dnešnej dobe zastaraný kvôli jeho nahradeniu inými technológiami.
Tcl	Využívaný najmä na jednoduché GUI implementácie, rýchle navrhovanie a testovanie.
AutoIT	Jazyk veľmi podobný jazyku AutoHotkey, menej populárny. Jeho cieľom je tiež automatizovať prácu s GUI. (AutoIT n. d.)



Graf 1 - Súčasné dominujúce jazyky v priemysle (TIOBE 2019)

Podľa TIOBE Indexu je v grafe uvedená popularita súčasných dominujúcich jazykov. Problémom avšak je, že dáta na základe ktorých je daná popularita určovaná nie sú dostatočne transparentné a - ako je na stránke indexu uvedené - nezahŕňajú vyhľadávania, ktoré sú lokalizované alebo obsahujú synonymá namiesto slova „programovanie“. (TIOBE 2019) Kvôli nedostatočnej transparentnosti nie je možné určiť či je popularita jazyka JavaScript určovaná len z vyhľadávaní, v ktorých je explicitne spomenutý, alebo či sú zahrnuté aj vrstvy JavaScriptu ako „JQuery“ alebo „node.js“.

Záver

Možnosti, ktorými sa dá riešiť rýchlosť práce s GUI, sú často komplexné aj v jazykoch ako Python, a nie sú dobre podporované v medzi platformových situáciách.

Existencia jazykov ako AutoHotkey, určených pre zjednodušenie zrýchľovania GUI, vďaka svojím schopnostiam získať informácie o oknách a manipulovať s nimi napomáha rýchlemu vyriešeniu GUI automatizácie.

Ako príklady skriptov napovedajú, pri akejkoľvek digitálnej činnosti sa môže nájsť nespočetné množstvo prípadov, kedy je GUI vylepšovanie a automatizácia viac než vhodná. Každý z prípadov prináša vlastné unikátne problémy, niektoré sa preukážu časom, iné hneď.

Väčšina problémov, ktoré sa dotýkajú GUI automatizácie sú aspoň dočasne riešiteľné, aj keď je autor skriptu schopný získať len minimálne množstvo informácií o obsahu okna.

Aj napriek neexistujúcej komerčnej podpore, nízkej kompatibilite a stabilite nekomerčné využitie vidí stále benefity v bohatej komunite, na ktorej si jazyky ako AutoHotkey zakladajú svoje hodnoty.

Vlastnoručná GUI automatizácia je určite vhodná pre ľudí odhodlaných experimentovať s komplexnými otázkami a dolaďovať veci do posledného detailu.

Bibliografia

CHANG, Tsung-Hsiang. 2011. Using graphical representation of user interfaces as visual references. In: *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*. s. 27-30.

INTHARAH, Thanapong, Michael FIRMAN a Gabriel BROSTOW. 2018. RecurBot: Learn to Auto-complete GUI Tasks From Human Demonstrations. In: *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. s. LBW595.

LOUI, Ronald. 2008. In praise of scripting: Real programming pragmatism. *Computer*. IEEE, roč. 41, s. 22-26.

WAINER, Jacques a Eduardo XAVIER. 2018. A Controlled Experiment on Python vs C for an Introductory Programming Course: Students' Outcomes. *ACM Transactions on Computing Education (TOCE)*. ACM, roč. 18, s. 12.

ZHANG, Sai, Hao Lü a Michael ERNST. 2013. Automatically repairing broken workflows for evolving GUI applications. In: *Proceedings of the 2013 International Symposium on Software Testing and Analysis*. s. 45-55.

Apple Human Interface Guidelines. 2019. [online]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines/

Apple Shortcuts User Guide. 2019. [online]. Dostupné z: https://support.apple.com/guide/shortcuts/welcome/ios

AutoHotkey Boards. 2019. [online]. Dostupné z: https://www.autohotkey.com/boards/

AutoHotkey Foundation. 2019. [online]. Dostupné z: https://www.autohotkey.com/foundation/

AutoHotkey. 2019. *Dokumentácia pre klávesové skratky* [online]. Dostupné z: https://www.autohotkey.com/docs/Hotkeys.htm

AutoIT. n. d. *Informácie o jazyku* [online]. Dostupné z: https://www.autoitscript.com/site/autoit/

BALAKRISHNAN, Akshay. 2018. *What is Perl? How relevant it is and how to get started!* [online]. Dostupné z: https://blog.usejournal.com/what-is-perl-how-relevant-it-is-and-how-to-get-started-d802e7aba2cd

BOTH, David. 2019. *Introduction to automation with Bash scripts* [online]. Dostupné z: https://opensource.com/article/19/12/automation-bash-scripts

FRANQUEIRO, Kenneth a Robert SIEGFRIED. 2006. Designing a scripting language to help the blind program visually. In: *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*. s. 241-242.

GNOME. 2019. *Developer Center* [online]. Dostupné z: https://developer.gnome.org/

IronAHK. 2011. *GitHub Repozitár* [online]. Dostupné z: https://github.com/Paris/IronAHK

JetBrains. 2019. PyCharm Reference Card. Dostupné z: https://resources.jetbrains.com/storage/products/pycharm/docs/PyCharm_ReferenceCard.pdf

KDE. 2019. *Dokumentácia* [online]. Dostupné z: https://docs.kde.org/

keybinder. 2017. *GitHub Repozitár* [online]. Dostupné z: https://github.com/kupferlauncher/keybinder

Microsoft Design. 2019. [online]. Dostupné z: https://www.microsoft.com/design

Microsoft. 2019. *Batch, Powershell, Windows commands* [online]. Dostupné z: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands

Microsoft. 2018. *RegisterHotkey Dokumentácia* [online]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-registerhotkey

node.js. 2019. *About node.js* [online]. Dostupné z: https://nodejs.org/en/about/

Notepad++. 2019. Cheat Sheet. Dostupné z: http://www.cheat-sheets.org/saved-copy/Notepad++_Cheat_Sheet.pdf

PYHK. 2017. *GitHub Repozitár* [online]. Dostupné z: https://github.com/schurpf/pyhk

pynput. 2019. *Dokumentácia o práci s klávesnicou* [online]. Dostupné z: https://pynput.readthedocs.io/en/latest/keyboard.html

PyPI PyAutoGUI. 2019. *Dokumentácia* [online]. Dostupné z: https://pypi.org/project/PyAutoGUI/

PyPI pynput. 2019. *Dokumentácia* [online]. Dostupné z: https://pypi.org/project/pynput/

PyPI system_hotkey. 2016. *Dokumentácia* [online]. Dostupné z: https://pypi.org/project/system_hotkey/

PyWinAuto. 2018. *Dokumentácia* [online]. Dostupné z: https://pywinauto.readthedocs.io/en/latest/

TIOBE. 2019. *TIOBE Index* [online]. Dostupné z: https://www.tiobe.com/tiobe-index/

XFCE. 2019. *Dokumentácia* [online]. Dostupné z: https://docs.xfce.org/

Zákon 185/2015 Z. z.. . Autorský zákon § 89.

Prílohy

Vlastný merací program **max_trvanie.c**

```c
#include <stdio.h>
#include <sysinfoapi.h>

const char * program_na_odmeranie = "python.exe main.py";
//const char * program_na_odmeranie = "main.ahk";

long int nameraj() {
    long int start = GetTickCount();
    int stav = system(program_na_odmeranie);
    long int end = GetTickCount();
    return (end - start);
}
int main() {
    int i;
    int pocet_spusteni;
    int trvanie, trvanie_min, trvanie_max;
    int trvanie_spolu = 0;
    scanf("%d", &pocet_spusteni);
    for(i = 0; i < pocet_spusteni; i++) {
        if (i) {
            trvanie = nameraj();
            trvanie_spolu += trvanie;
            if (trvanie < trvanie_min)
                trvanie_min = trvanie;
            else if (trvanie > trvanie_max)
                trvanie_max = trvanie;
        }
        else {
            trvanie = nameraj();
            trvanie_spolu += trvanie;
            trvanie_min = trvanie;
            trvanie_max = trvanie;
        }
    }
    printf("\n\n\n");
    printf("Externe trvanie spolu (s): %0.3lf\n",
    ((float)trvanie_spolu)/1000);
    printf("Najkratsie ext. trvanie (s): %0.3lf\n",
    ((float)trvanie_min)/1000);
    printf("Najdlhsie ext. trvanie (s): %0.3lf\n",
    ((float)trvanie_max)/1000);
    printf("Rozdiel tychto hodnot (s): %0.3lf\n",
    ((float)trvanie_max - (float)trvanie_min)/1000);
    return 0;
}
```

Vlastný merací program **trvanie.c**

```c
#include <stdio.h>
#include <sysinfoapi.h>

int main() {
    long int start = GetTickCount();
    int stav = system("python.exe main.py");
    long int end = GetTickCount();
        float trvanie = ((float)end - (float)start)/1000;
        printf("Externe trvanie (s): %0.3lf",trvanie);
    return 0;
}
```