Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-XXXX-XXXXX

Filip Mojto

POUŽITIE JEDNODUCHEJ AUTOMATIZÁCIE ZA ÚČELOM ZJEDNODUŠENIA KAŽDODENNÝCH RUTINNÝCH ČINNOSTÍ

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Miesto vypracovania:	Ústav počítačového inžinierstva a aplikovanej informatiky
Vedúci práce:	Mgr. Andrej Uhrín
Ďalší vedúci:	Ing. Katarína Jelemenská PhD.

Máj 2024

Slovenská technická univerzita v Bratislave
Ústav počítačového inžinierstva a aplikovanej informatiky

Fakulta informatiky a informačných technológií
2023/2024

ZADANIE BAKALÁRSKEJ PRÁCE

Autor práce:	Filip Mojto
Študijný program:	informatika
Študijný odbor:	informatika
Evidenčné číslo:	FIIT-16768-116253
ID študenta:	116253
Vedúci práce:	Mgr. Andrej Uhrin
Vedúci pracoviska:	Ing. Katarína Jelemenská, PhD.
Názov práce:	**Použitie jednoduchej automatizácie za účelom zefektívnenia každodenných rutinných činností**
Jazyk, v ktorom sa práca vypracuje:	slovenský jazyk
Špecifikácia zadania:	Automatizácia predstavuje jednu z nádejí aktuálnej doby ponúkajúcich možnosti zvýšiť kvalitu a množstvo dodanej práce. Napriek tomu jej plný potenciál sa nedarí dosiahnuť väčšine technologických firiem. Identifikácia spôsobov a implementácia jednoduchej automatizácie na úrovni jednotlivca ponúka možnosť ako pochopiť problémy a ponúknuť riešenia k využitiu potenciálu automatizácie. Analyzujte činnosti vhodné na automatizáciu a ich potenciál - dopad automatizácie na prácu jednotlivca. Pre väčší prínos práce zvoľte činnosti dobre známe širokému spektru ľudí - napr. automatizácia procesu objednávania jedla podľa kalendára (počet osôb, čas, adresa, preferencie na typ kuchyne a pod.). Na základe analýzy vyberte vhodný proces na automatizáciu. Navrhnite spôsob automatizácie vybraného procesu a návrh implementujte vo forme robota (softvéru), ktorý bude bežať na základe dohodnutých podmienok (triggers) automaticky. Otestujte a zhodnoťte prínos automatizácie k efektivite práce pracovníka, dopad na kvalitu a kvantitu. Identifikujte prípadné problémy, spojené s automatizáciou.

Rozsah práce:	40
Termín odovzdania práce:	21.05.2024
Dátum schválenia zadania práce:	
Zadanie práce schválil:	

Čestne prehlasujem, že som túto prácu vypracoval samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, 15.5.2023

Filip Mojto

Tu moze byt podakovanie

Anotácia

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program:	Informatika
Autor:	Filip Mojto
Bakalárska práca:	Použitie jednoduchej automatizácie za účelom zjednodušenia každodenných rutinných činností
Vedúci bakalárskej práce:	Mgr. Andrej Uhrín
Ďalší vedúci:	Ing. Katarína Jelemenská PhD.

Máj 2024

Tento dokument sa venoval komplexnej analýze automatizácie v hardvérovej i softvérovej podobe. Hlavným cieľom bolo získať jasný prehľad o stave automatizácie v súčasnosti, potom sa sústrediť na zvolený prípad. Po dôkladnej analýze práca navrhovala a implementovala jednoduchú automatizáciu sústreďujúcu sa na vybraný prípad v oblasti automatizácie. Prvá kapitola úvodom prehľadne zasväcuje čitateľa do skúmanej problematiky automatizácie, neskôr sa venuje významným témam a ich vplyvom na automatizáciu ako napríklad umelá inteligencia. Kapitola sa snažila najskôr poskytnúť všeobecný prehľad o problematike, neskôr sa viac konkretizovala na významné pod-oblasti ako roboty, automatizácie realizované počítačom a hardvérom a na záver skúmala stav automatizácie založenej na softvéri. Druhá kapitola sa viac orientovala na nami zvolený prípad - virtuálny obchod, kde hlavným cieľom bolo získať prehľad o stave automatizácie vo virtuálnom obchode, analyzovať rôzne príklady a zistiť tak možné nedostatky, ktoré by bolo možné zautomatizovať. Posledná kapitola realizovala softvérovú automatizáciu vo forme programu, ktorá sa špecializovala na zvolený prípad a ponúkla tak zaujímavé riešenie v skúmanej oblasti.

Annotation

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree course:	Informatics
Author:	Filip Mojto
Bachelor's Thesis:	The use of simple automation for the purpose of streamlining everyday routine tasks
Supervisor:	Mgr. Andrej Uhrín
Additional advisor:	Ing. Katarína Jelemenská

May 2024

This document focused on a comprehensive analysis of automation in both hardware and software forms. The main objective was to gain a clear overview of the current state of automation and then concentrate on a chosen case. After a thorough analysis, the work proposed and implemented a simple automation focusing on the selected case in the field of automation. The first chapter begins with a clear introduction, providing the reader with an overview of the studied issue of automation. It later delves into significant topics and their impact on automation, such as artificial intelligence. The chapter initially aimed to offer a general overview of the problem, later becoming more specific in exploring important sub-areas like robots, computer and hardware-based automation, and finally examining the state of software-based automation. The second chapter was more oriented towards the chosen case – a virtual store. The primary goal was to obtain an overview of the state of automation in virtual stores, analyze various examples, and identify potential shortcomings that could be automated. The final chapter implemented software automation in the form of a program, specializing in the chosen case and providing an interesting solution in the studied area.

Obsah

1 Úvod

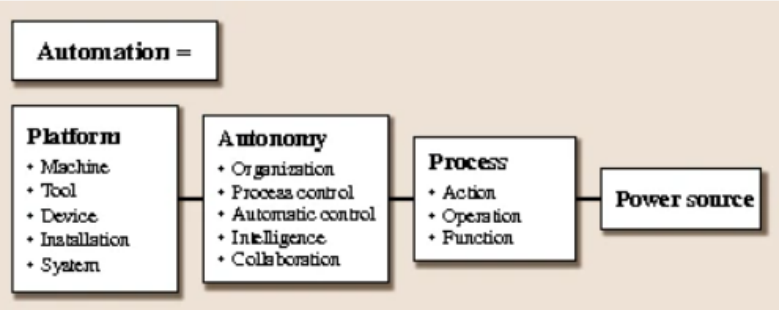
2 Automatizácia

Začnime stručným úvodom do problematiky. Nakoľko sa v našej práci sa primárne venujeme procesu automatizácie, je vhodné najskôr zo všeobecnejšieho pohľadu ozrejmiť, čo vlastne pojem automatizácia znamená. **Automatizácia** pochádza z anglického výrazu 'automation' (niekedy aj 'automatization') a znamená zjednodušenie a zefektívnenie vykonávania manuálnych úloh alebo činností, a to aj za účelom zvýšenia presnosti. [1]. Automatizácia už dávno prekročila svoje korene spočívajúce vo výrobe a rozšírilia ich na mnohé ďalšie uplatnenia, či už v zdravotníctve, bezpečnosti, doprave, poľnohospodárstve, v budovaní alebo v energetike a v mnohých ďalších oblastiach. Vývoj v oblasti automatizácie kladie dôraz na efektívnosť, produktivitu, výslednú kvalitu a spoľahlivosť spoliehajúc na systémy, ktoré operujú autonómne, často v štrukturovaných prostrediach a na dlhšiu dobu a na explicitné usporiadanie takýchto prostredí. Toto tvrdenie dáva do popredia akýsi dôraz, ktorý automatizácia kladie buď viac na dokonalejšiu štruktúru, a teda na spoľahlivosť, alebo preferuje menej štruktúrované prostredia, kde má zase väčšiu prispôsobivosť. [2] Ešte uvádzame jeden, tentokrát strúčnejší výrok, ktorý tvrdí, že automatizácia je mocná, má obrovský a obvivuhodný vplyv na civilizáciu, na ľudstvo, avšak môže prinášať i riziká.[3]

Ďalším dôležitým pojmom v oblasti automatizácie je **robot** alebo **automat** (z ang. Automaton). Robot je autonómne zariadenie, ktoré je závislé na nejakom zdroje energie a môže vykonávať komplexné série akcií bez nutnosti ľudského zásahu ako odpoved na programy alebo rôzne externé stimuly. Robot definuje autonómne zariadenie alebo nástroj, nezahŕňa však také platformy ako infraštruktúra či rôzne inštalácie alebo ďalšie automatické systémy ako ako automatizačný softvér.[3]

Stručný formalizmus

V tejto časti stručne načrtneme, akú má komplexný proces automatizácie štruktúru. Tú možno vidieť na obrázku 1. Presne sa tu vymedzujú možné vlastnosti typické pre akúkoľvek automatizáciu. Ide o akúsi konfiguráciu, ktorá charakterizuje akéhokoľvek robota alebo inú platformu.



Obr. 1: Základná štruktúra automatizácie[3]

Každá automatizácia musí fungovať na nejakej **platforme** alebo na **systéme platforiem**. Či už je to zariadenie, inštalácia alebo systém. **Autonómia** predstavuje samostatnú, zautomatizovanú oblasť, ktorú automatizácia zahŕňa a charakterizuje jej komplexnú funkcionalitu. Napríklad pre UI by bola autonómia inteligencia. Čo daná automatizácia vykonáva, to popisuje jej **proces**, napríklad môže vykonať nejakú akciu, či už poslať mail alebo plánovať rutinné zálohovanie dát. Automatizácie vykonávajú vo veľa prípadoch aj rôzne operácie, napríklad v oblasti výroby, ktoré boli kedysi vykonávané ľudskou manuálnou prácou. Alebo môže ísť o vykonávanie funkcií, napríklad softvér vykonáva pomocou algoritmov rôzne kalkulácie, atď.[3]:

2.1 Umelá inteligencia

Ako vlastne súvisí umelá inteligencia s automatizáciou ? Najskôr stručný opis umelej inteligencie.

Umelá inteligencia (často pod skratkou AI) je schopnosť systému vnímať očakávané ale aj neočakávané, nové udalosti, rozhodovať aké akcie je nutné vykonať za týchto podmienok či okolností a náležite akcie plánovať. Hlavné oblasti umelej inteligencie sú systémy založené na zbere vedomostí, počítačové senzorické systémy, systémy na spracovanie jazyka a strojové učenie.[3]

Umelá inteligencia je vlastne ľudská inteligencia implementovaná na stroje či zariadenia, a to hlavne skrz počítače a komunikáciu. Ide opäť o automatizáciu, takže AI dokáže fungovať sammostatne, bez ľudskej manipulácie počas jej fungovania a môže kombinovať inteligenciu viacerých ľudí a navyše aj zlepšiť vlastné schopnosti automatickým učením a adaptáciou. Dokáže byť automaticky distribuovaná, duplikovaná, zdieľaná, zdedená a v prípade nutnosti aj obmedzená či úplne odstránená. S príchodom týchto vlastností došlo prirodzene k ohromnému progresu v oblasti technológií.[3]

Teraz uvedieme rozdielny pohľad tento jav, možno trocha negatívny. Častou konfrontovanou nevýhodou je zníženie pracovných miest a tým aj zvýšenie nezamestnanosti. Nahradí automatizácia v kombinácii s umelou inteligenciou bežných pracovníkov ? Jeden zdroj uvádza, že kým niektoré odhady v tomto pohľade sú menej pesimistické, určité kľúčové štúdie uviedli, že veľké množstvo pracovníkov príjde kvôli rýchlo sa rozvíjajúcej automatizácii v blízkej budúcnosti o prácu. Pohľad na tento trend je kontroverzný. Kým niektoré hypotézy hovoria o zlepšení práce, napríklad zvýšením autonómie, kreativite a slobode pracovníka, iné hovoria o zvýšení neistoty zachovania si pracovného pomeru.[4]

While some estimates are less striking, several key studies have estimated that large numbers of workers will find themselves out of a job or needing to make a major transition in the near future due to automation.

2.2 Príklady automatizácie

Teraz sa môžme ísť pozrieť na konkrétne prípady automatizácie v praxi. Existuje množstvo príkladov z rôznych oblastí, či už z oblasti robotiky alebo v oblasti softvéru.

Regulátor parnej turbíny

Procesom je operácia parnej turbíny, ktorá automaticky riadi kompresor alebo generátor. Regulátor funguje na platforme zariadenia integrovaného systémom s kontrolérom programovateľnej logiky (PLC). Automanómia zahŕňa najmä automatickú aktiváciu či deaktiváciu a kontrolu rýchlosti turbíny, či programovateľné parametre regulátor pomocou obrazovky a mnohé ďalšie.[3]

Spracovávanie digitálnych fotografií

Platformou je v tomto prípade softvérový systém. Process sú rôzne akcie editovania alebo vylepšovania fotografií, pridávania rôznych grafických prvkov, odstranovanie fľakov a dalšie procesy typické pre grafický editor. Autonómia predstavuje plne automatické funkcie hneď ako ich používateľ aktivuje. Softvér môže tieto akcie vykonávať semiautomaticky alebo aj ako zautomatizovanú sériu akcií.[3]

Robotické striekanie náterov vozidiel

Túto automatizáciu spravidla vykonávajú roboti, teda zariadenia a stroje. Okrem iného musia byť zahrnuté aj rôzne sensory, dopravný pás, sprejovacie vybavenie a integrácia s programovacím

(a) Automatické striekanie náterov vozidiel

(b) Automatické editovania fotografií

Obr. 2: Ilustrované prípady automatizácii v praxi[3]

softvérom určenom na plánovanie akcií. Procesom je teda sprejovanie spolu s automatickým riadením pohybu auta, otváraním dverí a ďalšie. Autonómiu tvorí flexibilita pohybov, vyhýbanie sa kolíziám, koordinácia pásu, pohyb samotného robota, sprejovacie a striekacie operácie a ďalšie.

Teraz uvedieme ešte ďalšie prípady automatizácie, tentokrát formou tabuľky. Tá obsahuje zamerania rôznych štúdií a článkov v oblasti automatizácie. Pre

Tabuľka 1: Ďalšie prípady v oblasti automatizácie[5]

Autonómia	Prípad štúdie	Rok	SD	UE	UIA
Automatizácia činností	Automatizácia spotrebičov	2018	áno	nie	nie
Automatizácia činností	Manažment IT žiadostí	2018	áno	nie	nie
Manažement činností	Manažement zamestnaneckých činností	2018	áno	áno	áno
Architektúra Chatbot-a	Vládne služby	2019	nie	nie	nie
Automatizácia framework-ov	Analýza biznis dokumentu	2019	áno	nie	nie
Manažement činností	Plánovanie úloh pre zamestnancov	2020	áno	áno	áno

SD je skratka anglický výraz *Service Deployed* a znamená to, či daná štúdia poskytla servic pre automatizáciu manuálnej práce zamestnancami. UE znamená *Users Evaluation* a hovorí o tom, či bol výsledný produkt automatizácie hodnotený používateľmi. UIA je skratkou pre *User Impact Analysis* a označuje, či bola vykonávaná nejaká analýza na dopad na užívateľov.

2.3 Automatizácie založené na počítačoch

Teraz, ked sme pochopili pojmu automatizáciu ako takej, môžeme sa teraz sústrediť viac na softvér. Najskôr si priblížime automatizačné systémy založené na počítačoch.

Osobné počítače sa v súčasnosti čoraz častejšie používajú na implementáciu časovo náročných úloh v (distribuovaných) automatizačných systémoch[6]. Tento trend vychadza z viacerých faktorov, napríklad:

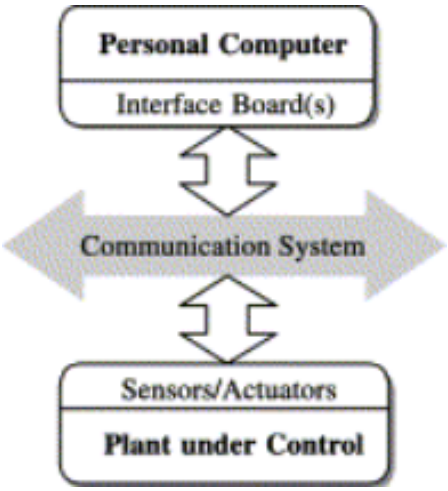
- Ohromný nárast počítačového výkonu
- Dostupnosť RTOS

- Vznik programovacích jazykov vhodných na kontrolné úlohy
- Narastajúce množstvo zariadení s rozhraním Fieldbus

Prvý faktor zaručuje, že v rovnakom stroji bude možnosť monitorovania či kontroly a súčasne aj možnosť vykonávanie úloh v reálnom čase. Druhý faktor umožňuje efektívne plánovanie úloh s tým, že kritické dostanú vyššie priority. Rozhranie Fieldbus umožňuje realizáciu distribuovaných architektúr s použitím hardvérových alebo softvérových produktov od rôznych výrobcov.[6]

Hardvérová štruktúra

Na obrázku 5 možno vidieť najvšeobecnejší typ automatizačného systému založeného na PC. Možno vidieť, že osobný počítač, na ktorom prebiehajú automatizačné úlohy, komunikuje so strojovým zariadením skrz nejaký komunikačný systém.



Obr. 3: Hardvérová štruktúra[6]

Tento typ rozhrania používaného počítačom je evidentne závislý na type komunikačného systému. V posledných rokoch bolo pripojenie ku stroju implementované väčšinou pomocou spojení implementovaných od jedného bodu k druhému, pričom každý koniec bol určený pre dané rozhranie. Takže sa vúčšinou jednalo o vstupné alebo vystupné dosky schopné spracovať digitálne alebo analógové signály. Avšak nedávano, najmä adopciou už spomínaných rozhraní Fieldbus, sa prispelo k vytvoreniu distribuovaných konfigurácií, v ktorých všetky komponenty automizačného systému boli priamo pripojené na takéto rozhranie. Zaujímavosťou je, že toto ďalej viedlo k adopcii novej generácie inteligentných senzorov a aktuátorov, ktoré boli schopné komunikovať v sieti.[6]

Softvérová štruktúra

Softvérová architektúra počítača musí zabezpečiť korektné vykonávanie automatizačných úloh, pre ktoré sú typické časové kritické požiadavky. Z tohto dôvodu sa bežne aplikuje operačný systém typu RTOS, ako možno vidieť na obrázku 4.

2.4 Typy automatizačného softvéru

2.4.1 Inteligentné spracovanie dokumentov - IDP

Dokumenty sú zdrojom informácií. Presnejšie predstavujú médium pre prenos informácií. Faktom je, že veľa informácií je získavaných z technických správ, vládnych správ, novín, kníh, žurnálov atď. Získavanie informácií z takýchto dokumentov pomocou informačného systému môže byť veľmi náročné a zdĺhavé a môže vážne obmedziť využitie informačných systémov.[7]

Teraz si môžme stručne definovať proces spracovania dokumentov. Vysvetlíme si to pomocou tzv. **základného modelu pre spracovanie dokumentov** (ang. Basic Model for Document Processing). Pre tento model bolo navrhnutých niekoľko základných konceptov, napríklad to, že konkrétny dokument dokument má nasledovné štruktúry:

- **Geometrická štruktúra**
- **Logická štruktúra**

Inteligentné spracovanie dokumentov má spravidla 2 fázy - **analýza dokumentu** a **dokumentové porozumenie**. Analýza dokumentu predstavuje extrakciu gemetrickú štruktúru z dokumentu a mapovanie geometrickej štruktúru na logickú štruktúru je definované ako dokumentové porozumenie. Hneď, ako sa podarí získať logickú štruktúru, jej význam možno dekódovať pomocou umelej inteligencie alebo iných techník[7].

Kľúčovým konceptom v automatickom spracovávaní dokumentov je štruktúra. Štruktúra dokumentu je výsledok rozdeľovania obsahu dokumentu na menšie a menšie časti, ktoré sa nazývajú **objekty**. Objekt, ktorý nemožno rozdeliť na menšie objekty sa nazýva **základný objekt**, ostatné objekty možno nazývať **zložené objekty**[7].

Dokumentová analýza predstavuje rozdelenie dokumentovej štruktúry na niekoľko blokov, ktoré reprezentujú súvislý sled komponentov dokumentu ako napríklad riadky textu, nadpisy, grafické inštancie ako napríklad obrázky a ďalšie. Pri vykonávaní tohto procesu môže, ale nemusí systém zahŕňať znalosť daného formátu. Výsledná analýza štruktúry môže byť reprezentovaná napríklad **geometrickým stromom**, načo sa bežne využívajú rôzne hierarchické či niehierarchické metódy. Pri hierarchckých metódach očakávame medzi objektami vzťahy typu rodič-dieťa a zostupujeme buď od rodičov na deti alebo naopak[7]. Hierachické metódy ponúkajú dva prístupy:

- Prístup zhora nadol (ang. **Top-down approach**)
- Prístup zdola nahor (ang. **Bottom-up approach**)

Prvý z prístupov je veľmi rýchly a efektívny najmä v situáciach, keď je formát dokumentu jasne a pevne daný. Druhý z prístupov je pomalý a výpočtovo náročnejší, avšak je vhodný v prípadoch, keď je nutné vyvinúť nejaký algoritmus vhodný na rôzne formáty a typy dokumentov či súborov. Najlepší výsledok získame rozumným kombinovaním obidvoch prístupov[7].

Teraz uvedieme niekoľko príkladov z praxe, kde možno IDP efektívne využiť, aby sa tak maximalizovala automatizácia manuálnej práce a minimalizoval výskyt rôznych výpočtových či iných chýb. Významným príkladom je napríklad spracovávania **fakturačných dokumentov**, kedy sa extrahujú rôzne citlivé dáta. Ďalším prípadom sú dokumenty týkajúce sa **poisťovníctva**, kedy sa zefektívni urýchlenie procesu nárokovania alebo komunikácia klienta. Ďalej je to napríklad spracúvanie **zmluvných dokumentov**, kedy IDP môže byť nápomocné tým, že extrahuje kľúčové údajové body zo zmlúv a pošle ich rôznym nástrojom na ďalšiu analýzu[1].

2.4.2 Automatizácia robotických procesov - RPA

V posledných rokoch bola automatizácia robotických procesov (RPA) veľmi populárna a získala si pozornosť firiem najmä v oblasti rôznych automatizačných iniciatív. RPA je prístup automatizácie v rámci širokej škály rôznych technológií slúžiacich na automatizáciu procesov. Každá z technológií zvyčajne slúži iným procesom a s rozdielnym cieľom.[8]

V situáciach, kedy ľudská manuálna práca alebo konštrukcia a integrácia biznis procesových systémov sú príliš nákladné alebo nedostatočné procesy, RPA prináša rozsiahlu biznisovo-procesovú automatizáciu. Príklad automatizácie za pomoci RPA zahŕňajú zautomatizovanie individuálnych aktivít či úloh.[8]

Ako zistiť, či je RPA vhodným prístupom pre automatizáciu konkrétneho procesu ? Treba zahrnúť viacero aspektov, napríklad organizačné schopnosti, financie alebo požadovaný čas. V prípade RPA je proces obzvlášť vhodným v situáciach, kedy obsahuje štandardnú, na pravidlách založenú štruktúru, a teda nevyžaduje schopnosť rozhodovať a učiť sa na základe rozmanitých situácií. Taktiež je vhodné, aby išlo o aktivitu, ktorú bežne manuálne vykonávajú ľudia. V neposlednom rade by malo ísť o systém s viac-systémovým prístupom. Takéto procesy zahŕňajú zvyčajne opakované hlásenie (zahŕňajúc určitú formu dátovej analýzy), generovanie masových e-mailov, archivácia, prípadne konverzia dátového formátu a grafiky.[8]

RPA teda prináša mnohé výhody, či už vo výkone, efektivite, škálovateľnosti, bezpečnosti či spoľahlivosti a pritom nevyžaduje náročnú implementáciu, ktorú naviac možno realizovať len s nízkymi nákladmi v porovnaní s bežnou automatizáciou procesov. Softvéroví roboti sa však nepodieľajú na vylepšovaní vykonávaných procesov. Ak teda robot vykonáva preddefinované správanie na základe chybných procesov, taktiež vykonajú chyby či nedostatky, čo môže mať za následky napríklad extra náklady a nadbytočné plytvanie zdrojmi. Preto je nutné procesy riadne optimalizovať a vylepšiť, než začne samotná automatizácia, ale ich aj neskôr udržiavať.[8]

Keďže RPA automatizuje opakované a manuálne úlohy, ľudskí pracovníci sú od nich oslobodení, aby tak mohli vykonávať úlohy vyžadujúce kreatívne myslenie, intelektuálny úsudok či sociálne skúsenosti. Avšak, automatizácia pomocou RPA nezaručuje úplné oddelenie ľudí a robotov, ale hľadá efektívnu mieru interakcie medzi nimi. I keď ľudia nemusia vykonávať manuálne úlohy vďaka RPA, stále je tu potreba spracúvať výnimočné scenáre, ktoré vyžadujú poznávanie, intuíciu a rozhodovanie na základe konkrétnej situácie.[8] Nižšie uvádzame tabuľku tried funkcionality softvérových robotov podla základného rozdelenia.

Tabuľka 2: Základne oblasti funkcií softvérových robotov[8]

Typ	**Trieda**	**Príklady**
Dátovo založené	Prenos dát	Aktualizácia/kódovanie dát
	Spracovanie dát	kódovanie/konvertovanie súborov
	Dátová analýza	Spracovanie reči na text
Integračne založené	Obsluha aplikácií	Zmena hodnoty v tabuľke
	Obsluha cloud-u	Vkladanie informácií na sociálne médiá
	Imitácia vstupných zariadení	Kliknutie, maximalizovanie...
Procesovo založené	Spúšťanie udalostí	Detekcia dátových zmien
	Obsluha kontrolného toku	Slučky, vetvenie, užívateľská interakcia

Dáta uvedené v tabuľke boli zistené analýzou 3 dostupných nástrojov (*UiPath*, *WorkFusion* a *Kryonsystems*), z ktorých sa každé orientovali na svoju špecifickú oblasť tak, ako je uvedené v stĺpci typ v tabuľke. Dátovo založené funkcie sa špecializujú na prenos, spracovanie a analýzu dát. Integračne založené funkcie umožňujú robotom pristupovať a riadiť rôzne aplikácie a služby. Po-

sledný typ reprezentuje spracúvanie rôznych udalostí (ang. event) a poskytuje viaceré operátory kontrolného toku[8].

2.4.3 Automatizácia biznis procesov

2.4.4 Forma umelej inteligencie

3 Virtuálny obchod

V tejto kapitole sa budeme venovať konkrétne oblasti automatizácie - virtuálnemu obchodu, ktorý sa čoraz viac stáva populárnejším ako bežné nakupovanie. Aj prípad automatizácie, ktorý implementujeme v našej práci spadá pod túto oblasť. Jestvuje veľké množstvo príkladov, ktoré sú vo virtuálnom obchodovaní nejakým spôsobom automatizované, tie ukážeme v časti *Príklady automatizácie*. Cieľom tejto práce bolo teda predstaviť ďalší príklad do tejto kolekcie. Teraz sa môžme ísť priblížiť elektronický obchod, jeho výhody či nevýhody pre každodenný život aj to, prečo by mali firmy prejsť na elektronickú formu obchodovania.

Automatizácia v oblasti elektronického obchodu je využívanie technológií a softvéru na zefektívnenie a zautomatizovanie rôznych činností a procesov v online predaji. Zahŕňa napríklad automatický manažment tovaru, spracovávanie objednávok a platieb, podporu pre zákazníkov i samotné obchodovanie a má za následok zvýšenú efektivitu, presnosť a škálovateľnosť pre elektronické obchody.[9]

V súčasnosti, elektronický obchod sa stáva technologickým riešením viacerých problémov, ktoré vznikajú počas procesov predaja alebo biznis manažmentu jednotlivých firiem.[10] Viesť virtuálny obchod dáva firmám množstvo výhod, ako napríklad:

1. Prístup z akejkoľvek geografickej oblasti
2. Neustála dostupnosť
3. Nárast počtu zákazníkov
4. Zvýšenie konkurencieschopnosti
5. Získavanie informácií o zákazníkoch
6. Zníženie nákladov a časov obsluhy

S vývojom internetu a ďalších technológií, efektivita online transakcií sa bude neustále zlepšovať. Preto obchodné spoločnosti v súčasnosti implementujú systém elektronický obchodu, aby zlepšili a zautomatizovali manažement obchodovania. Takéto spoločnosti teda často naskočia na nový obchodný smer, čím vyriešia množstvo problémov. Nato je ale nutný systém, ktorý poskytuje online katalóg produktov, ktoré obchod alebo spoločnosť predáva, má dostupnú nejakú spoľahlivú databázu na ukladenie informácií ohľadom zákazníckého nákupu.[10]

3.1 Príklady automatizácie

Áko sme spomináli, prípadov, kde automatizácia môže pomôcť urýchliť a zefektívniť proces elektronické obchodovania, je veľa. V tejto časti si uvádzame a poposujeme iba malú časť.

3.2 Shopify

3.3 Dostupné nástroje

3.3.1 Zapier

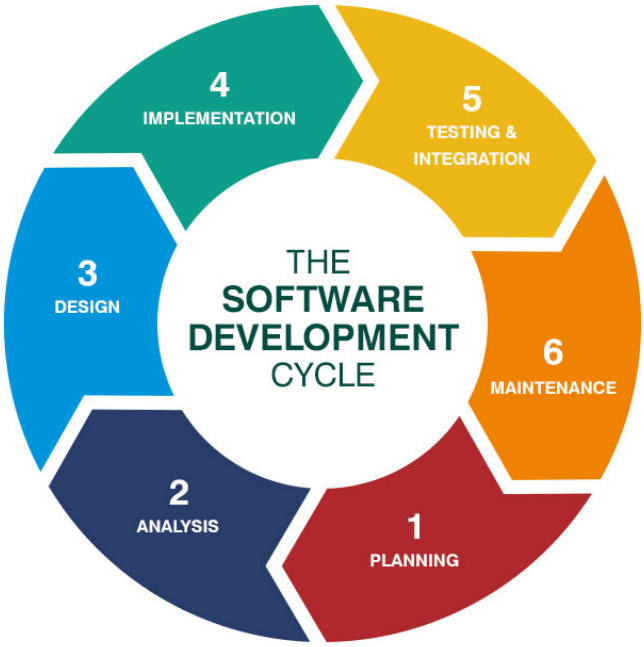
3.3.2 UIPath Studio a StudioX

3.3.3 WinAutomation

4 Opis riešenia

Po dôkladnej analýze súčasného stavu automatizácie, jej nástrojov a techník, a to aj v samotnom softvéri, sa v tejto časti presunieme na náš **implementačný návrh** pre túto problematiku. Návrh nám do veľkej miery umožní dôkladne premyslieť, pochopiť a rozpracovať stavebné piliere plánovanej aplikácie našej automatizácie. Rovnako nám pomôže riadne si určiť ciele našej aplikácie a aj jej dopad. Táto práca sa bude venovať návrhu softvéru, konkrétne aplikácie, ktorá bude realizovať jednoduchú automatizáciu na zefektívnenie manuálneho vyhľadávania produktov v papierových či online letákoch.

Nakoľko v našej práci realizujeme **softvérovú automatizáciu**, dodržiavame v tejto kapitole známe a odporúčané etapy aplikované pri vývoji a implementácii softvéru. Pre lepšie pochopenie tohto systému možno vidieť obrázok nižšie, ktorý poskytuje všeobecnú postupnosť vo forme tzv. **softvérového vývojárskeho cyklu**. Cyklus je vyjadrený pomocou niekoľkých kľúčových fáz a je vysoko odporúčané vykonávať jednotlivé fázy v uvedenom poradí.



Obr. 4: Základné etapy vývoja softvéru[11]

Pripomíname, že obrázok je len jeden z mnohých príkladov spôsobov vývoja softvéru a viaceré zdroje uvádzajú rôzne názvy i rôzne počty fáz. My sme sa rozhodli pre nasledovnú postupnosť, čomu budú zodpovedať aj nasledujúce podkapitoly:

1. **Špecifikácia cieľov a požiadaviek**
2. **Návrh**
3. **Implementácia**
4. **Overenie**

V špecifikácii sa súčasne venujeme hĺbkovej analýze problému formou určenia konkrétnych cieľov a požiadaviek, ktoré od našej aplikácie očakávame. Častou poslednou fázou býva údžba

softvéru, tú sme sa však pre dodržanie obmedzenia obsahu tejto práce rozhodli neuviesť. Ostatné fázy zodpovedajú uvedeným fázam na obrázku.

Pre prehľadné naplnenie účelov jednotlivých fáz používame v nasledujúcich podkapitolách softvérové modelovanie formou rôznych **diagramov**. Na modelovanie používame modelovací jazyk *UML*, ktorý je v súčasnosti veľmi populárny a ako prostredie pre modelovanie sme sa rozhodli pre webovú stránku *Visual Paradigm Online*.

Aby sme lepšie pochopili význam jazyka UML a prečo sme si ho vybrali pre náš návrh, uvádzame nasledovnú definíciu. UML (ang. Unified Modeling Language) je jazyk obsahujúci veľké množstvo nástrojov, ktoré enormným spôsobom zjednodušujú systémovú analýzu, a je dôležitým prostriedkom zlepšujúcim celkovú dizajnovú kvalitu návrhu. Používanie UML v analýze a dizajne umožňuje naplnenie systémových požiadaviek za pomoci objektovo prientovaného návrhu a modelov relačných databáz.[12]

4.1 Špecifikácia cieľov a požiadaviek

Na začiatku každého procesu vývoja softvéru je vhodné jasne a prehľadne špecifikovať ciele a požiadavky na softvér - neskôr budeme sledovať ich postupné napĺňanie. Je to nutné, aby sme jasne špecifikovali a vymedzili, čo možno od našej aplikácie očakávať, čo všetko je nutné implementovať, či aký bude mať naša aplikácia dosah na bežné fungovanie užívateľa. Na všetky tieto účely slúžia rôzne typy požiadaviek, z ktorých máme viaceré uvedené v nasledovných podkapitolách.

Vyslovenie hypotézy

Pred samotným procesom špecifikácie už konkrétnych cieľov či požiadaviek sme sa na začiatok rozhodli vysloviť akúsi stručnú hypotézu, ktorá v niekoľkých riadkoch zhrnie základné piliere našej automatizácie. Takýmto spôsobom v stručnosti zosumarizujeme všetko podstatné a budeme z tohto výroku pri neskoršom detailnejšom návrhu vychádzať, ale aj ho napĺňať a zdokonalovať. Teraz uvádzame samotnú hypotézu:

„Cieľom je vytvoriť plnohodnotnú desktopovú aplikáciu schopnú na základe poskytnutých dát o nákupe automaticky vyhľadať a stiahnuť dané položky z viacerých online letákov a na základe rôznych kritérií odporučiť najvhodnejší obchod pre nákup. Aplikácia tak bude šetriť ľudom peniaze a čas, ktoré by často minuli pri bežnom nevedomom nakupovaní.“

Pri skúmaní vyslovenej hypotézy sa naskytuje množstvo otázok. Tak napríklad - *“Aký operačný systém bude aplikácia podporovať ?”* Po dlhšej úvahe sme dospeli k nasledovnému záveru. Na základe času a prostriedkov, ktoré máme, sme sa zatiaľ rozhodli len pre podporu operačného systému Windows. To ale neznamená, že to úplne vylučujeme, zatiaľ je ale cieľom zapojiť hlavne tento systém, nakoľko je v súčasnej dobe stále najpopulárnejší. Okrem iného, máme s ním najviac skúseností.

Ďalšia, nemenej dôležitá otázka znie: *“Aký programovací jazyk alebo iný nástroj v oblasti automatizácie softvéru budeme v našom návrhu využívať ?”* S touto otázkou sme sa zaoberali asi najdlhšie. Aj na základe analýzy nástrojov v predchádzajúcej kapitole sme mali možnosť rozhodnúť sa buď pre väčšie množstvo programovania, kedy by sme sa rozhodli pre realizovanie signifikantnej časti aplikácie v nejakom konkrétnom **programovacom jazyku**, alebo si zvoliť nejaký voľne dostupný **nástroj na automatizáciu**, v ktorom by programovanie nebolo až tak nutné. Po dôkladnom zvážení daných možností sme sa nakoniec rozhodli pre prvú alternatívu a

programovací jazyk **Python**, a to najmä z väčších skúseností s programovaním i so samotným jazykom.

Je nutné zaoberať sa aj otázkou, *aké obchody bude aplikácia podporovať ?* Odpoveď je taká, že očakávame, že systém ponúkne čo najväčšiu rôznorodosť a bude vychádzať z dostatočného množstva obchodov, a to na základe letákov dostupných online. Teda aplikácia bude schopná stiahnuť z webovej stránky aktuálne dáta predstavujúce reálne produkty ponúkané konkrétnym obchodným reťazcom v danom okamihu. V práci budeme implementovať skripty pre sťahovanie produktov zo stránky *www.wolt.com*, ktorá zahŕňa viaceré reťazce fungujúce na Slovensku, ale i v zahraničí a dokonca sú tieto reťazce rozdelené aj do konkrétnych predajní.

Ostáva ešte množstvo otázok otvorených, tie sa však pokúsime zodpovedať už v samotnom návrhu. Teraz sa môžme pozrieť už na konkrétne príklady softvérových požiadaviek. Pre lepšie pochopenie, čo to vlastne softvérové požiadavky sú, uvádzame stručnú charakteristiku. Softvérové požiadavky tvoria podstatnú časť akéhokoľvek softvérového projektu. Identifikujú a objasňujú rôzne vlastnosti a súčasti, správanie a výkon, ktorý klienti alebo užívatelia očakávajú.[13]

Softvérové požiadavky sa bežne delia nasledovne:

1. **Biznisové**
2. **Používateľské**
3. **Softvérové**
   (a) **Funkčné**
   (b) **Nie-funkčné**

My sa budeme zaoberať najmä biznisovými a softvérovými požiadavkami. Používateľské požiadavky nešpecifikujeme osobite formou podkapitoly, ale iba nimi dopĺňame ostatné požiadavky.

4.1.1 Biznis požiadavky

Biznisové požiadavky v podstate predstavujú merateľné ciele či ukazovatele projektu. Prehľadne sa pomocou nich konkretizujú dôvody, prečo sa vlastne projekt realizuje.[13] V tejto časti uvádzame niekoľko biznis požiadaviek pre náš projekt. Predstavujú našu osobnú motiváciu pre realizáciu tohto projektu.

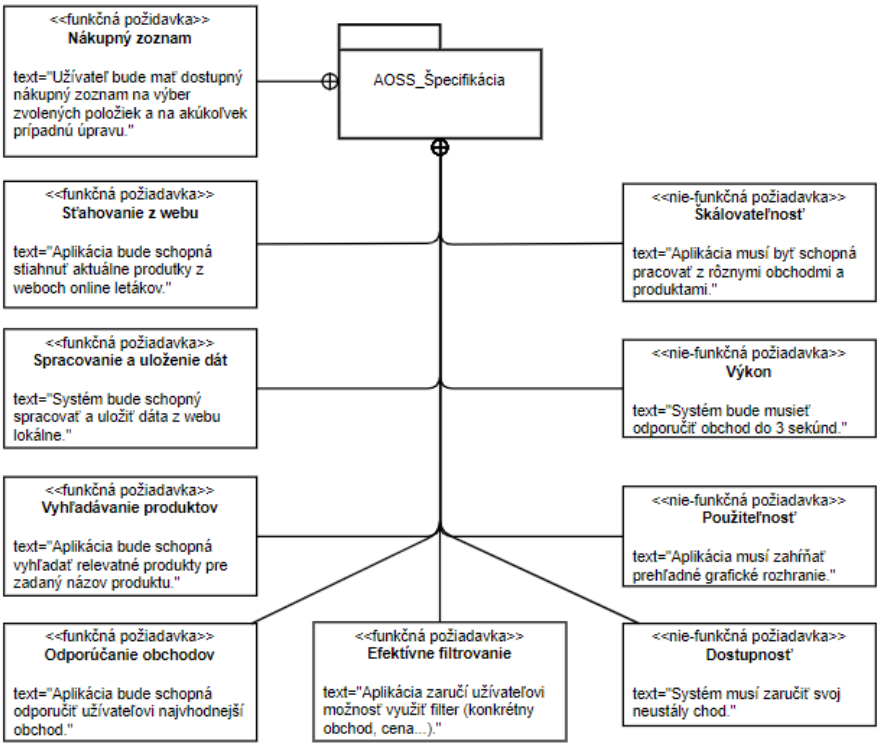
1. Aplikácia bude masívne šetriť zákazníkom čas.
2. Aplikácia dokáže zákazníkom šetriť peniaze.
3. Aplikácia zvýši návštevnosť jednotlivých obchodov.
4. Systém zvýši celkovú prehľadnosť užívateľov o jednotlivých obchodoch a ich produktoch.

Z požiadaviek vidno, že pozitívny vplyv aplikácie na každodenné fungovanie užívateľa bude badateľný, ale takisto bude softvér prinášať pozitívny efekt aj zapojeným obchodným reťazcom.

4.1.2 Softvérové požiadavky

Softvérové požiadavky sa ďalej delia na **funkčné** a **nie-funkčné.** Funkčné požiadavky jasne definujú, čo musí softvér vykonávať, aké sú jeho vlastnosti a funkcie. Nie-funkčné požiadavky sa sústreďujú skôr na všeobecné vlastnosti systému. Taktiež sú známe ako vlastnosti kvality.[14]

Na obrázku 3 možno vidieť niekoľko príkladov pre funkčné i nie-funkčné požiadavky charakterizujúce náš systém. Pre prehľadnosť sme sa rozhodli nezahŕňať užívateľské požiadavky osobite - na obrázku 3 sú uvedené spolu s funkčnými požiadavkami.



Obr. 5: Softvérové požiadavky

Z daného diagramu požiadaviek získavame lepší prehľad už aj o konkrétnych **aktéroch** či **komponentoch** nášho systému. Ak to chceme zhrnúť, systém bude musieť fungovať v nasledovnej postupnosti:

1. Sťahovať údaje z webu
2. Spracovať a uložiť dáta
3. Efektívne vyhľadávať produkty
4. Inteligentne odporúčať obchody

4.2 Návrh riešenia

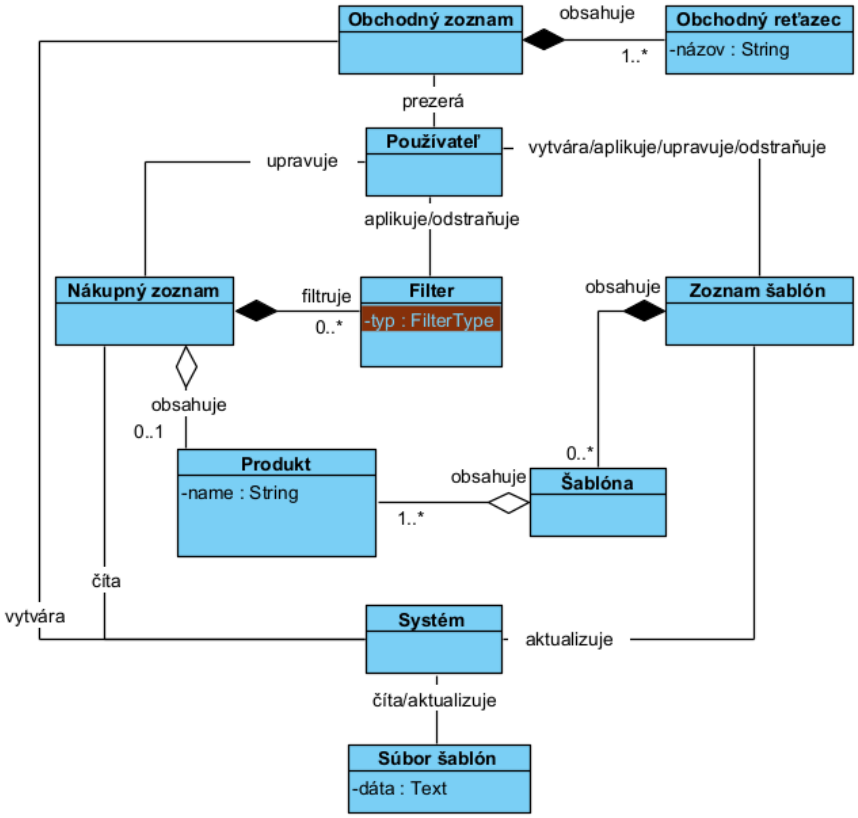
V minulej časti sme si úspešne stanovili naše osobné biznis ciele, funkčné aj nefunkčné požiadavky na systém. Teraz sa môžme ísť venovať návrhu našej implementácii, v ktorom sa budeme pokúšať o ich postupné napĺňanie. Návrh budeme postupne rozvíjať podľa odporúčaných krokov a ilustrovať za pomoci odporúčaných diagramov.

Túto kapitolu sme sa rozhodli rozdeliť do 2 menších kapitol. Prvá z nich 4.2.1 sa venuje identifikácii prípadov použitia, návrhu základnej štruktúry a architektúry v implementácii. Ďalšie z nich sa už venujú podstatným prípadom použitia, ktoré sme si identifikovali a približujú ich prúd vykonávania.

4.2.1 Prípady použitia, štruktúra a architektúra návrhu

Diagram tried

V tejto fáze je vhodné identifikovať jednotlivých aktérov systému, teda aké entity vystupujú a ako vzájomne spolupracujú v našej aplikácii. Na naplnenie týchto účelov sa v praxi bežne využívajú **diagramy tried**. Diagramy tried zachytávajú aspekty štruktúry softvéru, naznačujú stavebné bloky systému a aj to, ako spolu súvisia.[15] V klasickom diagrame tried budú vystupovať **triedy** a tie budú pospájané rôznymi **vzťahmi** (šípkami) podľa toho, ako spolu súvisia.



Obr. 6: Diagram tried

Na obrázku 6 možno vidieť načrtnutú štruktúru pre našu implementáciu. Je na ňom vidieť identifikovaných niekoľko dôležitých tried i vzťahov medzi nimi. Pre prehľadnosť sme každý vzťah opísali slovne a dané slovo umiestnili bližšie k triede, ktorá má daný vzťah k druhej triede. Tak napríklad, trieda *Používateľ* **upravuje** objekt triedy *Nákupný zoznam*. Pre prípady vzťahov *one-to-many*, teda agregovaných alebo compozičných asociácií sme uviedli aj kardinalitu. Z tejto štruktúry sa zatiaľ črtajú dvaja aktéri, a to:

1. **Užívateľ** (User)
2. **Systém** (System)

Začnime opisom funkcionality pre bežného užívateľa aplikácie. Možno vidieť, že trieda s názvom *User* môže vykonávať celkovo štyri funkcie. Prvá z nich umožňuje užívateľovi upravovať svoj nákupný zoznam podľa potreby. To zahŕňa pridávanie konkrétnych objektov pre triedu *Product*, kde užívateľ uvedie názov produktu a pridá ho do svojho zoznamu. Užívateľ takisto môže zo zoznamu produkt odstrániť. Rovnako tak môže aplikovať filter, či už na filtrovanie nákupov podľa názvu obchodu alebo ceny. Užívateľ takisto dokáže upravovať svoj vlastný zoznam šablón. Šablóny sme si už popísali a používateľ ich teda môže vytvárať, prezerať či odtraňovať. Vždy má dostupný aktualizovaný zoznam šablón. Na záver má užívateľ takisto možnosť prezerať si zoznam reťazcov odporúčaných aplikáciou.

Teraz sa presuňme na funkcie samotného systému. Pri štarte aplikácie systém číta externý súbor obsahujúci dáta týkajúce sa šablón pre konkrétneho užívateľa. Z týchto dát vytvorí zoznam, ktorý naplní špecifickými objektmi triedy *Šablóna* a ktorý je počas behu programu plne sprístupnený užívateľovi. Elementy tohto zoznamu sú počas behu programu pravidelne aktualizované, aby v prípade pádu alebo inej nečakanej udalosti dochádzalo k čo najmenším stratám na dátach. Systém počas programu v prípade potvrdenia užívateľom číta objekt triedy *Nákupný zoznam*, z ktorého následne stiahne požadované dáta z viacerých webových stránok predstavujúcich online letáky. Tieto dáta následne vyhodnotí a poskytne užívateľovi objekt triedy *Obchodný zoznam*, ktorý predstavuje jeden alebo viac odporúčaných reťazcov.

Na úspešné vyhodnotenie užívateľských požiadaviek musí systém stiahnúť dáta z viacerých webových stránok, porovnať ich a odporučiť najlepšie prípady. Bolo by teda vhodné pridať do diagramu aj nejakú triedu predstavujúcu takúto stránku, to sme však pre celkovú prehľadnosť diagramu neuvádzali.

Diagram prípadov použitia

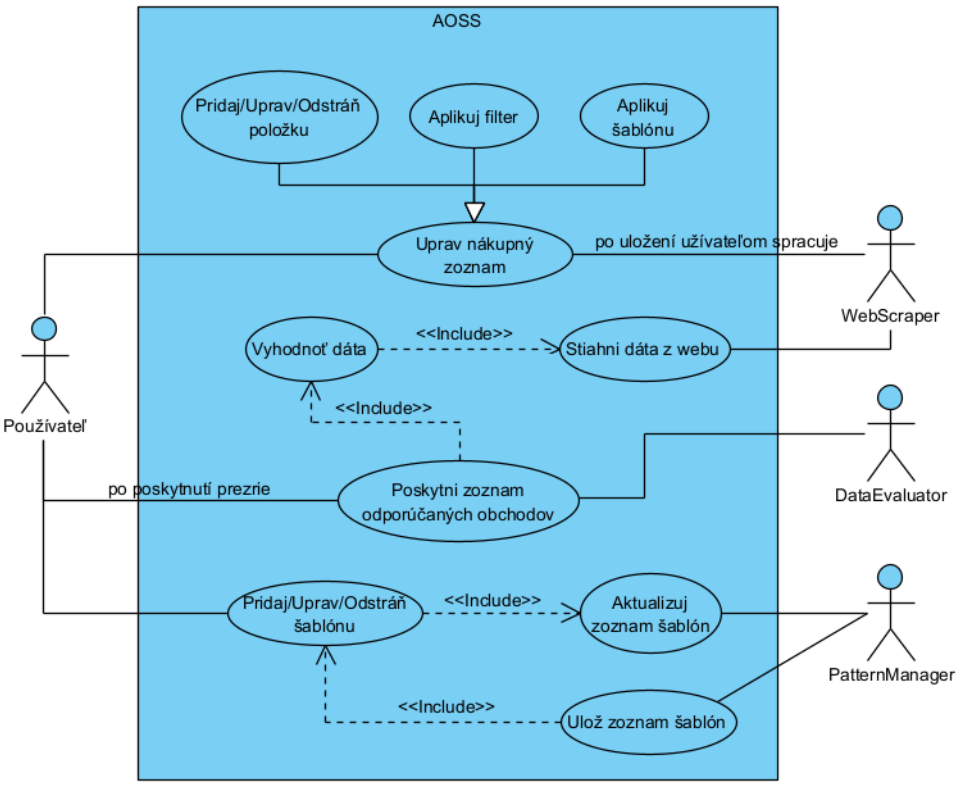
Počas fázy analýzy požiadaviek, diagramy prípadov použitia pomáhajú identifikovať aktérov a pomocou prípadov použitia definovať správanie sa systému. Okrem aktérov a prípadov použitia UML definuje menšie množstvo možných vzťahov na nadviazanie štruktúry medzi aktérmi a prípadmi použitia:

1. **Association**
2. **Extend**
3. **Include**
4. **Generalization**

Prvý vzťah predstavuje najbežnejší vzťah medzi prípadmi použitia a aktérmi a naznačuje asociáciu. Často to znamená, že aktér vykonáva daný prípad. Ďalší zo vzťahov hovorí, že jeden prípad použitia môže byť rozšírený o iný za splnenia určitej podmienky. Tretí znamená, že jeden prípad použitia v určitom momente zahŕňa a vykonáva iný prípad použitia. Posledný vzťah znamená, že jeden prípad generealizuje, teda špecifikuje, rodičovský prípad. Podobne, ako fungujú princípy OOP, detský prípad dedí všetky vlastnosti a asocácie rodiča.[16]

Obrázok 7 zobrazuje náš vlastný diagram prípadov použitia. Prvé, čo je dobré si všimnúť, je modrý štvoruholník predstavujúci oblasť pôsobnosti systému. Obsahuje názov aplikácie a možné prípady použitia, ktoré jednotliví aktéri vykonávajú.[16]

V diagrame tried v kapitole 4.2.1 sme už aktérov v systéme ako tak načrtli, boli to triedy *Používateľ* a *Systém*. Aktér **Používateľ** predstavuje bežného užívateľa našej aplikácie. Užívateľia systémov sú v diagramoch prípadov použitia bežne umiestňovaní na ľavú stranu. Systém predstavuje už spomínaná oblasť pôsobenia a každý prípad použitia by mal byť asociovaný s nejakým



Obr. 7: Diagram prípadov použitia

konkrétnym aktérom, preto sme identifikovali celkovo troch aktérov, ktorí predstavujú **komponenty** systému.

Ako vieme, diagram prípadov použitia takisto abstraktne definuje funkcionalitu systému. Z obrázka vidno, že bežnému užívateľovi systém umožňuje upravovať nákupný zoznam, a to či už aktualizáciou položiek predstavujúce produkty v obchode alebo aplikovaním nejakého typu filtra na zoznam. Takisto v prípade opakovaného nákupu môže využiť aj nejakú z predefinovaných šablón. V prípade šablón systém takisto umožňuje užívateľovi aktualizovať svoj vlastný zoznam šablón. Posledný prípad použitia predstavuje funkciu systému, ktorá užívateľovi prehľadne zobrazí odporúčané obchodné reťazce.

Čo sa týka samotných funkcií systému a jeho komponentov, tak už vieme, že systém musí vedieť na základe nákupného zoznamu, ktorý si užívateľ špecifikoval, stiahnuť potrebné dáta z podporovaných online letákov. Táto funkcionalita náleží komponentu *WebScraper*. Komponent *DataEvaluator* predstavuje akýsi algoritmus, ktorý stiahnuté dáta z webu vyhodnotí a poskytne mieru uspokojenia pre jednotlivé obchodné reťazce. V diagrame si možno všimnúť, že nato, aby bol užívateľov poskytnutý zoznam odporúčaných obchodov, je potrebné najskôr vyhodnotiť dáta. Nato je zase potrebné stiahnuť dáta z webu. Všetky tieto typy asociácie predstavujú vzťah *<<include>>*. Posledný komponent *PatternManager* má na starosti pravidelnú aktualizáciu užívateľmi definovaných šablón počas behu programu. V diagrame si možno opäť všimnúť, že nato, aby užívateľ mohol pracovať s zoznam šablón sa vyžaduje, aby bol vykonaný prípad použitia

Načítaj zoznam šablón. Takisto, aby mohol byť vykonaný prípad ulož zoznam šablón je potrebná nejaká modifikácia zo strany používateľa.

4.2.2 Sťahovanie dát z online letákov

V diagrame prípadov použitia sme si definovali dva významné prípady použitia *Stiahni dáta z webu* a *Poskytni zoznam odporúčaných obchodov*. Oba totižto predstavujú vcelku veľkú implementačnú výzvu, a preto pre ne poskytneme podrobnejší návrh v nasledovných dvoch podkapitolách. Začnime prvým z nich - ako už názov napovedá, bude úlohou aktéra systému *WebScraper* poslať žiadosť na online web stránku, ktorá mu ako odpoveď vráti svoje dáta. V našom prípade pôjde o online webový leták nejakého obchodného reťazca, odkiaľ budeme sťahovať dáta relevatné pre jednotlivé produkty. Toto je nutné z toho dôvodu, aby naša aplikácia bola schopná porovnávať čo najaktuálnejšie výsledky jednotlivých reťazcov.

Teraz sa môžme presunúť na trochu detailnejší návrh vyššie opísaných požiadaviek pomocou **diagramov sekvencií**. Pre správne pochopenie voľby práve tohto typu diagramu uvádzame stručnú definíciu - sekvenčné diagramy reprezentujú interakcie vykonávajúce sa sekvenčne medzi jednotlivými komponentami v softvérovej architektúre a rovnako aj ich aktívny čas.[17] Tento diagram je v tomto prípade veľmi vhodný, pretože nám vo väčších detailoch priblíží interakciu jednotlivých komponentov systému a aj ich čas.

Na obrázku 8 možno vidieť náš vlastný diagram sekvencií opisujúci prípad použitia *Stiahni dáta z webu*. Celkovo sme si v ňom definovali 4 entity, jedna predstavuje **externého používateľa** aplikácie a ostatné sú **objekty** alebo **komponenty systému**. Keď si priblížime dané komponenty, tak *DataManager* a *WebScraper* budú tvoriť náš vlastný systém, resp. aplikáciu. *Online Web Leták* predstavuje externú web stránku, teda komponent systému konkrétneho reťazca.

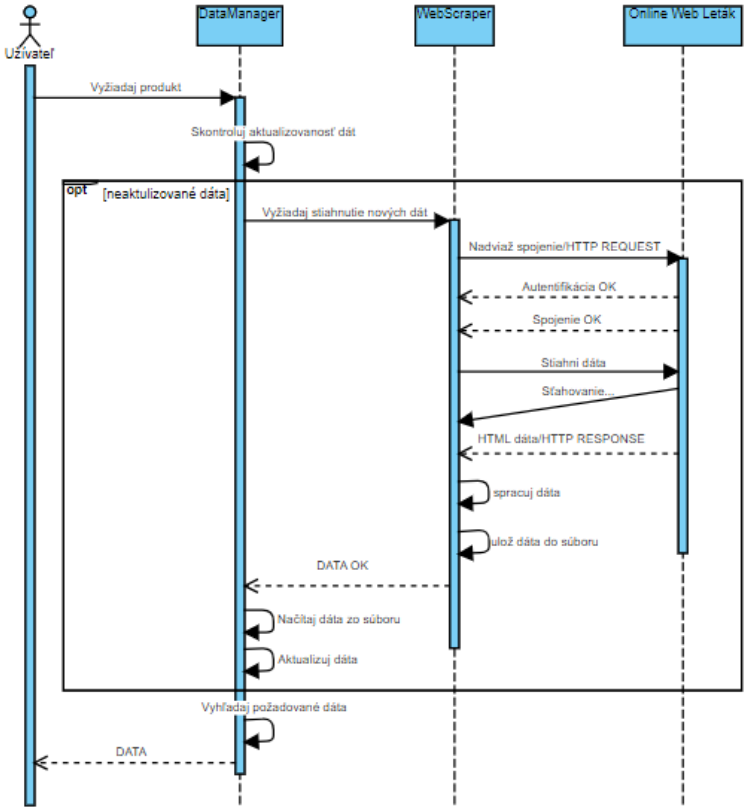
Samotné **interakcie** sú reprezentované šípkami. Z obrázku vidno, že užívateľ si vyžiada dáta o produkte, teda si vytvorí nákupný zoznam a pošle ho aplikácií na vyhodnotenie. Komponent *DataManager* obdrží dané požiadavky, no najskôr skontroluje, či nie je nutné aktualizovať dáta uložené lokálne. Tu nastáva možnosť, že dáta nie sú aktuálne a je najskôr potrebné stiahnúť nové. V takom prípade vyšle žiadosť komponentu *WebScraper* obsahujúcu názov žiadaného produktu. *WebScraper* najskôr musí cez internet nadviazať spojenie s online letákom pomocou **protokolu HTTP**. Prvým krokom je HTTP žiadosť, ktorá bude zahŕňať:

1. **Unikátny zdrojový lokátor - URL**
2. **Autorizačný kľúč**

Formát samotného URL bude samozrejme závislý na formáte URL webovej stránky online letáku a bude zahŕňať samotný názov produktu, pre ktorý sa na stránke zobrazia dostupné výsledky. Rovnako bude obsahovať aj číslo webovej stránky, ak bude informácií o produktoch príliš veľa, aby boli len na jednej stránke. Autorizačný kľúč slúži na autorizáciu žiadosti nášho systému o dáta spočívajúcu v potvrdení, že nejde o nejakého robota, ktorý chce vykonávať pravidelné sťahovanie dát, nakoľko v súčasnosti má veľké množstvo webových stránok zabudované rôzne obranné mechanizmy proti takýmto pokusom.

Ak je autorizácia úspešná a spojenie nadviazané, pošle webová stránka **HTTP odpoveď**, ktorá bude obsahovať dáta v jazyku *HTML*. Takéto dáta sú zvyčajne veľmi robustné a bude úlohou komponentu *WebScraper* ich spracovať a extrahovať z nich to najpodstatnejšie, a to názvy produktov a ich ceny. Proces extrakcie bude opäť závislý na štruktúry konkrétnej stránky.

Po úspešnom stiahnutí a spracovaní dát z webu má *WebScraper* ešte za úlohu dáta niekde uložiť. Tu sa naskytá veľa možností, napríklad uložiť dáta lokálne do súboru, prípadne do databázy. Pre jednoduchosť našej práce sme sa rozhodli pre ukladanie do súboru vo formáte **csv**. To znamená, že v jednotlivých riadkoch v súbore budú atribúty (názov, cena...) oddelené čiarkou.



Obr. 8: Sekvenčný diagram pre PP *Stiahni dáta z webu*

To, čo komponent *DataManager* spraví s danými dátami, závisí od konkrétneho prípadu. Ak užívateľ požaduje iba vyhľadanie nejakého lokálneho produktu, tak iba vyhľadá spomedzi lokálnych dát tie relevatné pre požadovaný produkt. Ak však užívateľ požaduje automatické odporučenie konkrétneho reťazca, tak komponent musí pomocou algoritmu vyhľadať produkty vo všetkých reťazcoch a z nich na základe kritéria nájsť tie najvhodnejšie.

Len pripomíname, že v diagrame nie je pre prehľadnosť zahrnuté **grafické rozhranie**, cez ktoré komunikuje užívateľ so systémom a jeho komponentami.

4.2.3 Efektívne vyhľadávanie produktov a odporúčanie reťazcov

V minulej časti sme podrobne navrhli, ako budeme schopní stiahnuť reálne a aktuálne dáta dostupné z online letákov jednotlivých reťazcov. Takisto sme opísali ako ďalej dané dáta spracujeme a uložíme. Teraz sa pôjdeme pozrieť na ďalšie dva dôležité prvky našej aplikácie, a to je efektívne vyhľadávanie relevantných produktov a inteligentné odporúčanie reťazcov na základe definovaných metrík. Na naplnenie týchto cieľov bude nutná realizácia určitých algoritmov, preto v nasledujúcich podkapitolách uvádzame detailnejší opis ich jednotlivých krokov pomocou **diagramov aktivít**.

Diagram aktivití sú v UML grafické notácie opisujúce správanie inštancií tried počas ich životnosti. Často slúžia na modelovanie toku systému v modelovaní softvérových či webových

aplikácií, prípadne v biznis modelovaní.[18]

Konkrétne v našom prípade pôjde o to, že užívateľ si buď vyhľadá produkt alebo vytvorí svoj vlastný nákupný zoznam, na ktorý bude očakávať automatizované spracovanie, vyhodnotenie a odporučenie obchodného najvýhodnejšieho obchodného reťazca. Je teda nutné implementovať nejaký vysoko efektívny a optimalizovaný algoritmus na vyhľadávanie spomedzi desaťtisícov možných produktov a hľadanie najväčšej zhody s hľadanou položkou.

Na hľadanie zhody medzi textovými reťazcami existujú viaceré algoritmy, my sme sa rozhodli pre **Levenshtein-ovu vzialenosť**.

Levenshtein-ova vzdialenosť je metrika mejarúca počet operácií (vloženie, odstránenie alebo nahradenie znaku) požadovaných na konvertovanie reťazca znakov A na reťazec B. Používa sa v rôznych výpočtových doménach, zahŕňajúc textovú či hudobnú analýzu, prípadne v informatike.[19]

PP04 - Nájdi zhodu medzi produktami

Teraz sa môžme ísť detailnejšie pozrieť na spomínaný prípad použitia identifikovania zhody medzi hľadaným produktom a dostupnými produktami pre jednotlivé reťazce. Z diagramu prípadov použitia uvedeného v kapitole 4.2.1 už vieme, že ho vykonáva entita *DataManager*. Tento prípad použitia a jeho kroky sme opísali pomocou diagramu aktivít na obrázku 9 nižšie.

Diagram používa tzv. "plávacie línie"(ang. *swimlanes*) pre identifikáciu účinkujúcich v a nimi vykonávaných aktivít v diagrame. Celkovo sa na tejto interakcií podielajú dve entity:

1. **Užívateľ**
2. **DataManager**

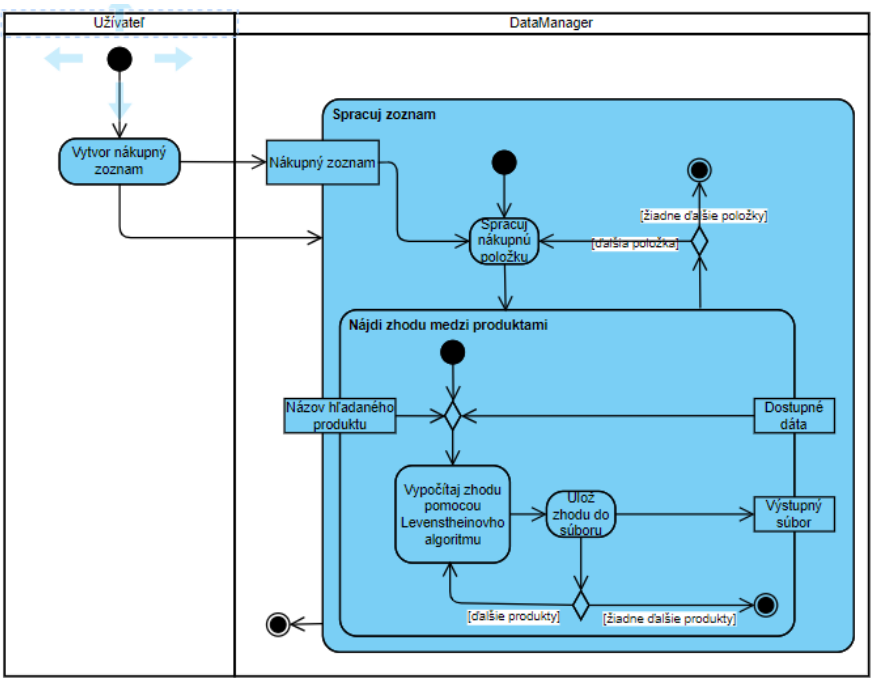
Ako sme už spomínali, proces začne užívateľ, ktorý si napríklad vytvorí nákupný zoznam a potvrdí svoj výber. Následne je nutné požiadavku spracovať, a to tak, že entita *DataManager* načíta prvú položku. Začne sa vykonávať dôležitá aktivita *Najdi zhodu medzi produktami*, ktorá dostane na vstup spracovávanú položku, plus dostupné dáta, ktoré predstavujú uložené produkty v súboroch. Pre každý dostupný produkt poskytovaný nejakým reťazcom je nutné identifikovať zhodu pomocou Levenshtein-ovej vzdialenosti a tú uložiť do **výstupného súboru**. Toto sa opakuje, dokým nie sú všetky produkty vybavené.

Z diagramu vidno, že proces končí na strane enitity *DataManager*. Tá spracovala užívateľovu požiadavku o vyhľadanie položky (prípadne celého zoznamu) a dané zistenia uložila do súboru. Tieto dáta môže ďalej spracovať entita *ShopSearcher*, ktorému sa budeme venovať v nasledujúcej kapitole.

PP05 - Odporuč najvhodnejší reťazec

Entita *DataManager* úspešne identifikovala zhody pre všetky produkty v zozname so všetkými dostupnými produktami. Na základe týchto dát teraz entita *ShopSearcher* vyhodnotí dané zhody a na základe kritérií užívateľa odporučí ten najvhodnejší reťazec (prípadne reťazce). Detailnejšie si to opíšeme pomocou diagramu uvedenom na obrázku 10. Tu máme 3 aktérov, a to používateľa a entity *DataManager* a *ShopSearcher*. Keď sa pozrieme na prvé dve aktivity, vidíme, že ide o tie, ktoré sú detailnejšie opísané v predchádzajúcom diagrame. Toto naznačuje významné logické nadväzovanie diagramov a interakcie medzi entitami.

Teraz sa poďme bližšie pozrieť na prípad použitia *Odporuč najvhodnejší reťazec*. Ten na vstup dostane relevantné produkty spracované *DataManager* entitou a bude postupne pre každý reťazec počítať jeho *mieru celkovej úspešnosti*. Pre každý reťazec sa najskôr extrahujú relevantné produkty ponúkané daným reťazcom a to sa ešte rozdelí na zhody na základe konkrétneho produktu. Entita *DataManager* teda bude musieť identifikovať obchod aj konkrétnu skupinu



Obr. 9: Diagram aktivít pre PP04 *Nájdi zhodu medzi produktami*

relevantných produktov pre špecifický produkt z nákupného zoznamu pomocou **unikátnych identifikátorov**.

Ak sa stane, že bude viacero relevantných produktov pre hľadaný produkt, bude užívateľ vyzvaný, aby určil, ktoré sú preňho potrebné. Cieľom nášho projektu je ale aplikácia slúžiaca ako automatizácia, a preto bude našou snahou, aby nevznikali viaceré zhody na jeden produkt.

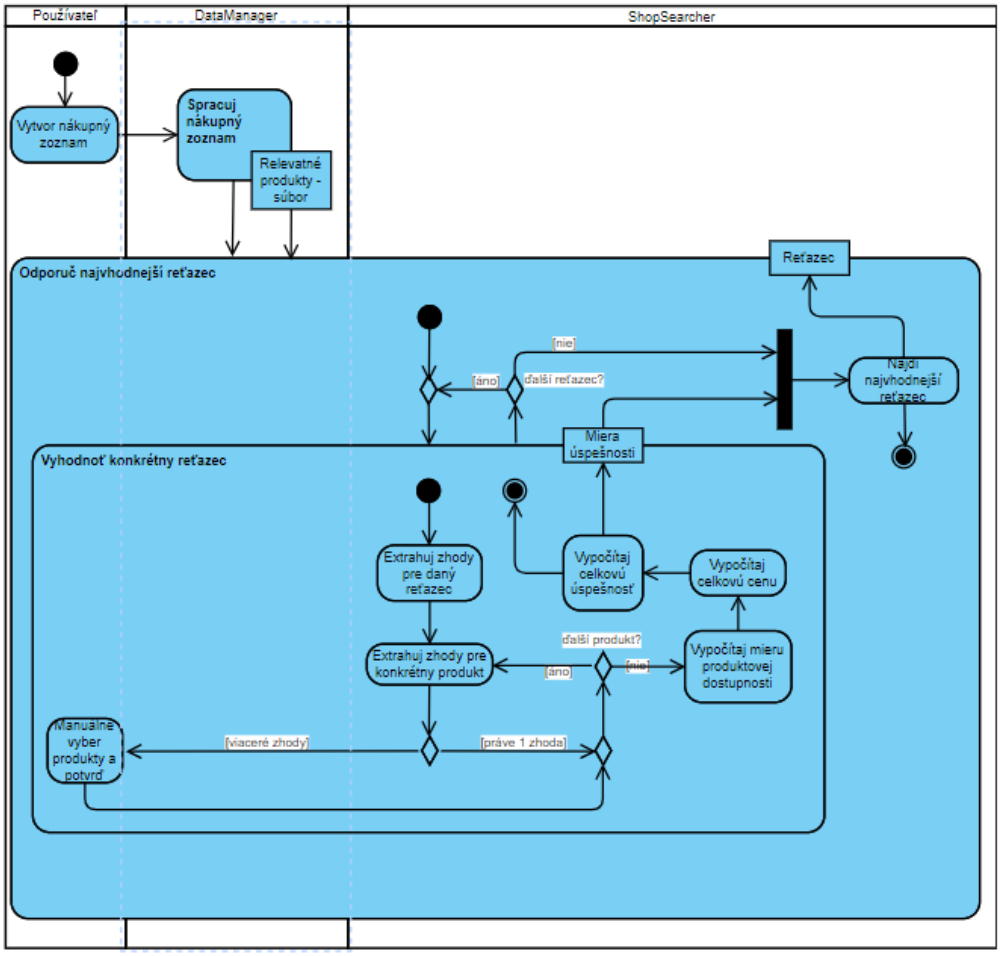
Keď budú všetky produkty pre daný reťazec úspešne spracované, je nutné vypočítať dve metriky:

1. Miera produktovej dostupnosti
2. Celková cena

Produktová dostupnosť predstavuje percento uspokojených produktov z nákupného zoznamu. Celková cena bude predstavovať súčet cien všetkých produktov v zozname vytvorenom entitou *ShopSearcher*. Užívateľ má možnosť pri vytváraní zoznamu definovať vlastné kritéria, teda napríklad či chce viac mieru produktovej dostupnosti alebo ceny. Entita *ShopSearcher* teda pre každý reťazec vypočíta jeho **celkovú úspešnosť** na základe produktovej dostupnosti a jej váhy a ceny a jej váhy.

4.3 Implementácia

4.4 Overenie



Obr. 10: Diagram aktivít pre PP05 *Odporuč najvhodnejší reťazec*

5 Zhodnotenie výsledkov

6 Technická dokumentácia

Literatúra

[1] A. Hero, "What is automation software? types & examples." Webová stránka, https://automationhero.ai/blog/what-is-automation-software-types-examples/, Máj 2023.

[2] K. Goldberg, "What is automation?," *IEEE Transactions on Automation Science and Engineering*, vol. 9, January 2012. Date of Publication: 13 December 2011, s. 1-2.

[3] S. Y. Nof, "Automation: What it means to us around the world," in *Springer Handbook of Automation* (S. Y. Nof, ed.), Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 13-52.

[4] L. Nazareno and D. S. Schiff, "The impact of automation and artificial intelligence on worker well-being," *Technology in Society*, vol. 67, p. 101679, 2021.

[5] R. S. Bavaresco, L. C. Nesi, J. L. Victória Barbosa, R. S. Antunes, R. da Rosa Righi, C. A. da Costa, M. Vanzin, D. Dornelles, S. C. Junior, C. Gatti, M. Ferreira, E. Silva, and C. Moreira, "Machine learning-based automation of accounting services: An exploratory case study," *International Journal of Accounting Information Systems*, vol. 49, p. 100618, 2023.

[6] S. Vitturi, "Pc-based automation systems: an example of application for the real-time control of blowing machines," *Computer Standards & Interfaces*, vol. 26, no. 2, pp. 145–155, 2004.

[7] Y. Y. Tang, S.-W. Lee, and C. Y. Suen, "Automatic document processing: A survey," *Pattern Recognition*, vol. 29, no. 12, pp. 1931–1952, 1996.

[8] P. Hofmann, C. Samp, and N. Urbach, "Robotic process automation," *Electronic Markets*, vol. 30, pp. 99–106, 03 2020.

[9] S. G. Contributor, "What is ecommerce automation? how it works plus benefits." Webová stránka, https://www.shopify.com/enterprise/what-is-ecommerce-automation, September 2021.

[10] A. Tupia-Astoray and L. Andrade-Arenas, "Implementation of an e-commerce system for the automation and improvement of commercial management at a business level," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 1, 2021.

[11] T. Pham, "6 stages for software development procedure you need to know." Webová stránka, https://saigontechnology.com/blog/6-stages-for-software-development-procedure-you-need-to-know, Október 2019.

[12] L. Cavique, M. Cavique, A. Mendes, and M. Cavique, "Improving information system design: Using uml and axiomatic design," *Computers in Industry*, vol. 135, p. 103569, 2022.

[13] S. J. Bigelow, "What are the types of requirements in software engineering?." Webová stránka, https://www.techtarget.com/searchsoftwarequality/answer/What-are-requirements-types, December 2020.

[14] altexsoft, "Functional and nonfunctional requirements: Specification and types." Webová stránka, https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/, Júl 2021.

[15] G. Bergström, F. Hujainah, T. Ho-Quang, R. Jolak, S. A. Rukmono, A. Nurwidyantoro, and M. R. Chaudron, "Evaluating the layout quality of uml class diagrams using machine learning," *Journal of Systems and Software*, vol. 192, p. 111413, 2022.

[16] T. von der Maßen and H. Lichter, "Modeling variability by uml use case diagrams," in *Proceedings of the International Workshop on Requirements Engineering for product lines*, pp. 19–25, 2002.

[17] V. Karampure, Wang, "Uml sequence diagram to axiomatic design matrix conversion: a method for concept improvement for software in integrated systems," *Procedia CIRP*, vol. 100, pp. 457–462, 2021. 31st CIRP Design Conference 2021 (CIRP Design 2021).

[18] M. Abbas, R. Rioboo, C.-B. Ben-Yelles, and C. F. Snook, "Formal modeling and verification of uml activity diagrams (uad) with focalize," *Journal of Systems Architecture*, vol. 114, p. 101911, 2021.

[19] D. Castells-Rufas, "Gpu acceleration of levenshtein distance computation between long strings," *Parallel Computing*, vol. 116, p. 103019, 2023.