**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**

# Diplomová práce

## Využití programovacího jazyku Python k automatizaci kancelářské činnosti

**Bc. Ladislav Typlt**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Ladislav Typlt

Systémové inženýrství a informatika
Informatika

Název práce

**Využití programovacího jazyku Python k automatizaci kancelářské činnosti**

Název anglicky

**Usage of programming language Python for automating office tasks**

---

**Cíle práce**

Diplomová práce je tematicky zaměřena na problematiku využití programovacího jazyku Python k automatizaci kancelářské činnosti. Hlavním cílem diplomové práce je porovnat knihovny pro účely automatizace kancelářské činnosti. Postranním cílem diplomové práce je představit možnosti využití programovacího jazyku Python. Dílčí cíle práce jsou následující:
- Vytvořit kritickou literární rešerši k problematice programovacího jazyku Python.
- Analyzovat možnosti využití programovacího jazyku Python pro účely automatizace.
- Provést komparaci knihoven pro automatizaci.
- Navrhnout použití v podnikové praxi.
- Syntetizovat dosažené výsledky a formulovat závěry práce.

**Metodika**

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Komparace vybraných knihoven pro automatizaci je provedena prostřednictvím nalezení vhodného kritéria a jeho kvantifikací, s využitím řady mezinárodních norem SQuaRe. Vlastní řešení je realizováno formou návrhu a implementace balíku vlastních aplikací vyvinutých v programovacím jazyku Python. Na základě syntézy teoretických poznatků a výsledků vlastního řešení budou formulovány závěry diplomové práce.

**Doporučený rozsah práce**

60-80 stran

**Klíčová slova**

Programovací knihovny, python, automatizace, mezinárodní normy

---

**Doporučené zdroje informací**


BROŽOVÁ, H. – HOUŠKA, M. – ŠUBRT, T. – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. PROVOZNĚ EKONOMICKÁ FAKULTA, – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. KATEDRA OPERAČNÍ A SYSTÉMOVÉ ANALÝZY. *Modely pro vícekriteriální rozhodování.* Praha: Credit, 2009. ISBN 978-80-213-1019-3.

ČSN ISO/IEC 25000 (369006) Systémové a softwarové inženýrství – Požadavky a hodnocení kvality systémů a softwaru (SQuaRE)

KAZIL, Jacqueline a Katharine JARMUL. Data wrangling with Python. ISBN 1491948817.

LUTZ Mark, Programming Python, 4th Edition. O'Reilly Media, Inc. 2010. ISBN: 9781449398712.

Python documentation. Python 3.6.2 documentation [online]. [cit. 2017-09-10]. Dostupné z: https://docs.python.org/3/

ŠUBRT, T. *Ekonomicko-matematické metody.* Plzeň: Vydavatelství a nakladatelství Aleš Čeněk, s.r.o., 2015. ISBN 978-80-7380-563-0.


---

**Předběžný termín obhajoby**

2021/22 ZS – PEF

**Vedoucí práce**

Ing. Jan Tyrychtr, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 19. 11. 2020

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 19. 11. 2020

**Ing. Martin Pelikán, Ph.D.**

Děkan

V Praze dne 30. 03. 2022

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci "Využití programovacího jazyku Python k automatizaci kancelářské činnosti" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. 3. 2022 _____

**Poděkování**

Rád(a) bych touto cestou poděkoval(a) vedoucímu práce panu doc. Ing. Janu Tyrychtrovi, Ph.D. za vedení práce a trpělivost při procesu tvorby.

# Využití programovacího jazyku Python k automatizaci kancelářské činnosti

**Abstrakt**

Práce se zabývá tématem automatizace kancelářské činnosti. Nejprve jsou v rámci literární rešerše přiblíženy oblasti programovacích metod a algoritmizace, objektová metodologie UML a vícekriteriální rozhodování, následně programovací jazyk Python, a oblasti automatizace, kterých se práce týká, kvalita SW, prvky jejího měření a aktuální stav ISO/IEC norem. Další část tvoří hodnocení jakosti a porovnání knihoven metodou AHP, výstupem je sada skriptů k použití pro podnikovou praxi.

**Klíčová slova:** automatizace, programovací knihovny, python, mezinárodní normy, vícekriteriální analýza

# Usage of python for office work automation

**Abstract**

The thesis deals with the topic of office automation. Firstly, the literature search introduces the areas of programming methods and algorithmization, object-oriented methodology UML and multi-criteria decision making, then the Python programming language, and the areas of automation covered by the work, the quality of software, elements of its measurement and the current status of ISO/IEC standards. The next part consists of quality assessment and comparison of libraries using the AHP method, resulting in a set of scripts for use in corporate practice.

**Keywords**: automation, programming libraries, python, international standards, multi-criteria analysis

# Obsah

# 1 Úvod

Zatížení administrativou různých forem, a rostoucí byrokratizace, představují interdisciplinární a celospolečenský fenomén, u kterého i vzhledem k vývoji tuzemského a unijního právního prostředí, nelze očekávat snížení, a který nemá jednoduché řešení. Jistou míru úlevy může poskytnout oblast informačních technologií, konkrétně vedle zřejmé digitalizace, i oblast související s tématem této práce, a to RPA (*Robotic Process Automatization*). Samozřejmě to především v případě vyšší úrovně harmonizace, formou předem definovaných procesů šetřících náklady na vývoj nebo údržbu SW (*software*), nebo například customizací ERP (*Enterprise resource planning*) systémů. Problematikou této práce je související téma, a to obecně možnosti využití jazyka Python pro automatizaci kancelářské činnosti. Tento jazyk má poměrně jednoduchou syntaxi, někdy přirovnávanou k pseudokódu, z toho důvodu je tezí práce, že základní znalosti v tomto jazyce umožní (i vzhledem k širokému portfoliu open-source knihoven) zefektivnit činnost hospodářsko-ekonomického pracovníka, a jiných pracovníků, průřezově napříč obory.

Pro účely diplomové práce, a pro možnost aplikace metody AHP (*Analytic Hierarchy Process*), je kancelářská činnost zúžena, a to podle obsahu nejpoužívanějších kancelářských balíků, a dále zjednodušeně rozdělena na interakci s 5 kategoriemi (oblastmi) kancelářského SW. Ty jsou pro tyto účely následující:

- Tabulkové procesory (*spreadsheet*)
- Textové procesory
- Portable Document Format (PDF)
- Nástroje na tvorbu prezentací
- Klient ke správě elektronické pošty

V rámci těchto oblastí jsou hodnoceny možnosti automatizace následujících činností:

- tvorba dokumentu (souboru) hodnoceného formátu
- extrakce dat z dokumentu (souboru) hodnoceného formátu
- Manipulace s obsahem dokumentu (souboru) hodnoceného formátu
- Manipulace s dokumentem (souborem) nativně v hodnoceného formátu
- Manipulace s metadaty (vlastnosti souboru) hodnoceného formátu

A to s tím, že se předpokládá, že v případě umožnění těchto činností, umožní hodnocená knihovna (případně v kombinaci s dalšími jazykovými knihovnami jazyka Python)

užitečnější a praktičtější úkony, jako je například přejmenování souboru v souborovém systému podle obsahu dokumentu, hromadné vytváření souborů, konverze do dalších formátů, integraci s dalšími systémy, a tak podobně. Jako kritéria hodnocená metodou AHP (s částečným využitím norem řady SQuaRE – specificky indikátorů ISO/IEC 25010) byly vybrány následující:

- Funkční úplnost
- Funkční správnost
- Funkční vhodnost
- Dokumentace
- Velikost knihovny

Jako vybrané a následně pomocí AHP metody vzájemně porovnávané knihovny, bylo z portfolia jazykových knihoven jazyka Python zvoleno následující:

- openpyxl
- Python-docx
- PyPDF2
- python-pptx
- pypff/libpff

Práce vedle stěžejní praktické části s ukázky snippletů přibližuje další možnosti automatizace, které realizovány nebyly. Stručně je popsán „*State of Art*“ a přiblíženy specifika relevantních a přidružených oblastí. Následuje obsah jednotlivých částí.

V metodice je nejdříve představena algoritmizace a programovací metody, dále uveden stručný popis použité objektové metodologie UML (*Unified Modeling Language*), a následně popis použité metody pro výpočet vah kritérií, podle kterých jsou varianty – jednotlivé knihovny, porovnávány – metody AHP.

V teoretické části je nejprve přiblížen programovací jazyk Python použitý pro vývoj jednotlivých automatizačních aplikací (tzv. *snippletů*) a některá jeho specifika, jako je virtuální prostředí a nástroj PIP, dále je zde uvedena rešerše k problematice automatizace, kancelářské činnosti obecně (ze které je vycházeno při kategorizaci jednotlivých hodnocených úkonů), kancelářských balíků a relevantních souborových formátů. V závěru podkapitoly jsou uvedeny některé současné trendy, možnosti a nástroje spjaté s problematikou automatizace. V další podkapitole je stručný popis použitých knihoven, postup k výběru hodnocených knihoven, a obecně popsané některé další jazykové knihovny

jazyka Python. Další a poslední část teoretické části se zabývá hodnocením kvality softwaru a některými relevantními normami, a to především řady ISO 25000 SQuaRE.

Praktická část je zaměřena nejprve na hodnocení jakosti jednotlivých knihoven, kde je nejprve uveden model jakosti, a následně ukázka možnosti implementace jednotlivých porovnávaných knihoven, včetně demonstrativního diagramu případu užití u jednoho úkonu kategorie (oblasti) softwaru - knihovny, která z hodnocení kritérií u jednotlivých variant a následné syntézy preferencí vyšla jako nejlepší. Většina kritérií, respektive jejich vah, byla stanovena značně subjektivně, například u funkční úplnosti na základě autorem definovaných činností (úkolů a cílů), které by knihovna měla splňovat, u funkční správnosti a funkční vhodnosti na základě slovního popisu ohodnocení, stejně tak pro danou oblast může být některá z nezahrnutých knihoven vhodnější, z těchto důvodů je hodnocení spíše ilustrativní.

V závěrečné části je uvedeno vyhodnocení a popis možnosti využití v podnikové praxi, jehož implementace je uvedena v příloze.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Diplomová práce je tematicky zaměřena na problematiku využití programovacího jazyku Python k automatizaci kancelářské činnosti.

Hlavním cílem diplomové práce je porovnat knihovny pro účely automatizace kancelářské činnosti.

Postranním cílem diplomové práce je představit možnosti využití programovacího jazyku Python.

Dílčí cíle práce jsou následující:

- Vytvořit kritickou literární rešerši k problematice programovacího jazyku Python.
- Analyzovat možnosti využití programovacího jazyku Python pro účely automatizace.
- Provést komparaci knihoven pro automatizaci.
- Navrhnout použití v podnikové praxi.
- Syntetizovat dosažené výsledky a formulovat závěry práce.

## 2.2 Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Komparace vybraných knihoven pro automatizaci je provedena prostřednictvím nalezení vhodného kritéria a jeho kvantifikací, s využitím řady mezinárodních norem SQuaRe (*System and Software Quality Requirements and Evaluation*, Požadavky na kvalitu systému a softwaru a jejich hodnocení). Vlastní řešení je realizováno formou návrhu a implementace balíku vlastních aplikací vyvinutých v programovacím jazyku Python. Na základě syntézy teoretických poznatků a výsledků vlastního řešení budou formulovány závěry diplomové práce.

V následujících podkapitolách metodiky je nejdříve představena algoritmizace a programovací metody, následně objektová metodologie UML, nakonec použitá metoda AHP a obecně oblast vícekriteriálního rozhodování.

Následná realizace praktické části bude provedena prostřednictvím editoru Visual Studio Code, UML diagramy budou vytvořeny za pomocí nástrojů Drawio a Plantuml a

tabulky včetně realizace AHP metody za pomocí tabulkového procesoru MS Excel. V závěru části budou popsány cíle, kterých autor dosáhl, včetně námětů rozvoje do budoucna.

### 2.2.1 Algoritmizace a programovací metody

Pojem algoritmus sám o sobě má více definicí. Například podle příspěvku (Jandová, 2010) je **algoritmus** přesně definovaná konečná řada správně určených výroků, také nazývaných jako instrukce nebo příkazy, které poskytují řešení problému nebo specifické kategorie více problémů pro jakoukoliv přijatelnou sadu vstupních hodnot, pokud nějaké jsou. Jinými slovy algoritmus popisuje návod na řešení problému krok za krokem s tím, že na svém konci dosáhne řešení a neběží věčně. Určité kroky musí následovat v určitém pořadí, aby algoritmus mohl dosáhnout cíle (Jandová, 2010).

Algoritmus musí být konečný a jednoznačný. U každého algoritmu musí být jasně definován vstup a výstup. Formálně jsou podle (Blahuta, 2017) definovány tyto vlastnosti, které každý algoritmus musí splňovat:

- konečnost: Algoritmus neskončí v nekonečném cyklu
- resultativnost: Po konečném počtu kroků vrátí výstup (může být i chyba)
- správnost: Výstup musí být správný
- determinovanost: V každém kroku musí být zcela jasný způsob pokračování algoritmu (vč. větvení)
- hromadnost (univerzálnost): Algoritmus musí být popsán obecně, nikoli pro konkrétní případy
- opakovatelnost: Při stejném vstupu stejný algoritmus musí dát stejný výsledek (Blahuta, 2017).

Vedle toho můžeme podle (Blahuta, 2017) algoritmy rozdělit do skupin jako jsou rekurzivní algoritmy, pravděpodobnostní, paralelní, genetické algoritmy a další, možnostmi zápisu algoritmů je slovní vyjádření, matematické vyjádření, vývojový diagram, rozhodovací tabulka, objektová tabulka a počítačový program (Blahuta, 2017).

Například podle (Virius, 1995) se v algoritmech se setkáváme se třemi základními **konstrukcemi**, které označujeme jako posloupnost (**sekvenci**) příkazů, s cyklem (**iterací**) a s podmíněnou operaci (**selekcí**, výběr). Posloupnost (sekvence) je dále podle autora tvořena jedním nebo několika kroky, které se provedou právě jednou v daném pořadí. Cyklus (iterace) představuje podle autora část algoritmu, která se opakuje, dokud je splněna podmínka opakování a vždy se skládá z podmínky opakování a z těla cyklu, tedy z

operací, které se opakují. Podmíněná operace (selekce) představuje větvení algoritmu a je tvořena podmínkou a jednou, dvěma nebo více výběrovými složkami. Při analýze algoritmů nás dále podle autora zajímá nejen správnost (zda je dostán správný výsledek) a v případě numerických algoritmů přesnost, ale podle autora také doba, kterou budeme k provedení algoritmu potřebovat, a množství operační paměti, které bude potřebovat program, realizující algoritmus. Při hodnocení časové náročnosti můžeme vycházet z celkového počtu elementárních kroků, tedy např. instrukcí strojového kódu, které musíme provést, v závislosti na rozsahu vstupních příp. výstupních dat. Dále jsou podle autora tématem spjatým s algoritmy **datové struktury**. Jako základní datové struktury označuje (Virius, 1995) proměnnou, pole, záznam, a objekt, s tím, že se s nimi (až na objekt) podle autora setkáme ve většině funkcionálně orientovaných programovacích jazyků. **Proměnná** podle autora představuje vlastně pojmenované místo v paměti počítače a vytvoří se na základě deklarace, ve které se sdělí její jméno (obvykle identifikátor) a typ, kde specifikací typu je určována množina hodnot, které do dané proměnné budeme smět ukládat, a množina operací, které s danou proměnnou budeme moci provádět a nepřímo tím obvykle také určujeme velikost proměnné, tj. množství paměti, které bude proměnná zabírat (Virius, 1995).

Co se týče jazyka Python, podle (Summerfield, 2019) Python nezná proměnné jako takové, nabízí však tzv. odkazy na objekty a ohledně neměnitelných objektů, jako jsou int (*integer*, celé číslo) a str (*string*, řetězec), pak neexistuje žádný rozeznatelný rozdíl mezi proměnnou a odkazem na objekt. U proměnlivých objektů dál podle autora již sice rozdíl existuje, ale v praxi však nemá téměř žádný význam a z tohoto důvodu oba termíny, proměnná a odkaz na objekt, budou (mohou) znamenat totéž (Summerfield, 2019).

Obecně neformálním popisem algoritmu na vyšší úrovni používaným například pro edukační a publikační účely je pak **pseudokód**. Podle (Sawa, 2021) je cílem pseudokódu sdělit čtenáři informace, které jsou důležité z hlediska popisovaného algoritmu a často se používá běžná matematická notace, někdy i přirozená řeč, pokud je popis nějaké operace jednodušší v přirozené řeči než v podobě kódu. Často jsou podle autora v pseudokódu ignorovány detaily, které nejsou pro daný algoritmus důležité a příkladem takových detailů je třeba ošetření chybových stavů (např. nedostatku paměti) nebo použité datové typy (Sawa, 2021). Dále se podle autora v pseudokódu typicky třeba pracuje s proměnnými, do nichž je možno přiřadit jako hodnotu libovolně velké celé číslo a neřeší se, že ve skutečném programu napsaném v nějakém programovacím jazyce pak je pro tyto

proměnné použit nějaký konkrétní datový typ kde jsou hodnoty, kterých může proměnná nabývat, omezeny na nějaký konečný interval celých čísel. Autor dále uvádí, že oproti popisu v přirozené řeči má pseudokód výhodu, v tom, že je v něm mnohem zřetelněji vidět celková struktura algoritmu, je tam jasně vidět, jak jsou do sebe zanořeny cykly a jak se větví podmínky (Sawa, 2021).

Vzhledem k tomu, že je algoritmus v zásadě pouze teoretickým řešením postupu, je důležité přiblížit praktickou část, kterou je v SW inženýrství implementace algoritmu v podobě jeho zápisu do programovacího jazyka. Postup, kdy lze při tvorbě programu (viz dále) pro počítač prostřednictvím algoritmu řešit nějaký problém, se nazývá **algoritmizace** a lze ji rozdělit do několika částí (Blahuta, 2017):

- formulace problému
- analýza úlohy
- vytvoření algoritmu
- sestavení programu
- odladění programu (Blahuta, 2017).

Je-li algoritmus zapsán ve formě, které rozumí počítač, pak tedy mluvíme o **programu**. Program vzniká v programovacím jazyce (forma zápisu algoritmu pro počítač) a tuto činnost nazýváme **programování** (Blahuta, 2017).

Existuje mnoho programovacích jazyků[1], a více kritérií jak je rozdělit. Jedním ze způsobů seskupení, podle toho co programovací jazyky dělají, jsou **programovací paradigmata** – podle (Skoupil, 2007) jsou to vyzkoušené a ověřené programovací styly, vedoucí k tvorbě kvalitního softwaru, s tím, že jazyky mohou patřit do více než jednoho paradigmatu. Přehled základních podle (Skoupil, 2007) je uveden níže:

- Naivní paradigma (Klasický jazyk Basic)
- Procedurální paradigma (Klasické jazyky jako Fortran, C, Modula2 nebo Pascal)
- Objektově orientované paradigma (historické jazyky jako Simula a Smalltalk, používané C++, Java, C#)
- Funkcionální paradigma (Lisp, Haskell, Miranda)
- Logické paradigma (Prolog) (Skoupil, 2007).

[1] pro představu v žebříčku popularity programovacích jazyků Tiobe index je jich hodnoceno 274, s tím, že se jedná pouze o jazyky Turing-kompletní. Index se vypočítá z počtu výsledků vyhledávání dotazů a pro výpočet se používá 25 vyhledávačů a dvacet nejpopulárnějších je pravidelně zveřejňováno.

Pro účely práce jsou podstatné dvě skupiny paradigmat, obecně vnímané jako hlavní, a to imperativní a deklarativní, kde podle uvedeného členění je funkcionální programování formou **deklarativního** programování, a obecně většina běžných jazyků, včetně objektově orientovaných, byla navržena především k podpoře procedurálního (**imperativního**) programování.

V případě programování v jazyce Python se podle autora této práce, a to na základě rešerše k problematice, obecně dá říct, že záleží především na preferencích a znalostech vývojáře, které z programátorských metod (paradigmat) zvolí. Ilustrativně podle článku na webu newrelic.com od (Mueller, 2018) umožňuje totiž Python osvojení čtyř stylů (paradigmat, přístupů) programování v jazyce Python a to imperativní, funkcionální, objektově orientovaný a procedurální, s tím, že někteří lidé kombinují imperativní a funkcionální, jiní je považují za odlišné, a autor (Mueller, 2018) na webu uvádí i výhody a nevýhody jednotlivých přístupů a k tomu uvádí tyto příklady:

- **Funkcionální**: Každý příkaz je považován za matematickou rovnici a vyhýbáme se jakýmkoli formám stavu nebo proměnlivých dat. Hlavní uvedená výhoda je pro paralelní zpracování (*parrallel processing*), jelikož nejsou stavy, a mnoha vývojáři je preferováno pro rekurzi a lambda calkulus, ale implementace funkcionálního programování v jazyce Python se odchyluje od standardu (tj. není úplná podpora paradigmatu) a pro "čistší" implementaci jsou vhodnější jazyky jako např. Haskell
- **Imperativní**: Výpočet jako přímá změna stavu programu, užitečné pro manipulaci s datovými strukturami, jednoduchý kód (plná podpora paradigma)
- **Objektově orientovaný**: Vychází z datových polí (*data fields*), se kterými se zachází jako s objekty a manipuluje se s nimi pouze prostřednictvím předepsaných metod. Není plně podporováno jelikož nejsou implementovány funkce jako skrývání dat (zapouzdření, *encapsulation*)
- **Procedurální**: K úlohám se přistupuje jako k postupným iteracím, kdy jsou běžné úlohy umístěny do funkcí, které jsou volány podle potřeby. Tento styl kódování upřednostňuje iteraci, sekvencování, výběr a modularizaci, a podle autora je implementace v rámci jazyka vynikající (Mueller, 2018).

Vedle paradigmat je z hlediska práce důležité představit dělení programovacích jazyků podle úrovně abstrakce (blízkosti k HW) zjednodušeně na nižší a vyšší úrovně, kde **vyšší programovací jazyky** (vysokoúrovňové nebo problémově orientované jazyky) lze dále podle (Pilgrim, 2010) rozdělit do skupin kompilovaných a interpretovaných jazyků,

využívajících interpreta a mezi které patří programovací jazyk Python. Podle serveru (stackoverflow) se u kompilovaných jazyků napsaný kód jednou zkompiluje tj. přeloží do instrukcí procesoru (binárního kódu) a ten je možné spustit[2], u interpretovaných existují různá řešení ale obecně se kód překládá do tzv. byte kódu (*Bytecode*), a obecnou vlastností interpretovaných programů je, že jsou pomalejší než ty kompilované. Jazyky dále obecně mohou mít například dynamické nebo statické typování, kde se u statického přiřazuje datový typ proměnné při deklaraci a u dynamického je datový typ sdružen s hodnotou, a jazyky s dynamickým typováním jsou obecně pomalejší nebo silné a slabé typování, kde silně typovaný provádí kontrolu typů a opačně (stackoverflow). Skriptovací jazyky jsou podle (stackoverflow) obecně jazyky určené k rozšíření nebo propojení existujících aplikací a komponent.

Dalšími důležitými pojmy v oblasti jsou syntaxe a sémantika. Podle (Skoupil, 2007) popisuje **syntaxe** programovacího jazyka formální strukturu programu, definuje klíčová slova, identifikátory, čísla a další programové entity a určuje způsob, jak je lze kombinovat, vedle toho **sémantika** programovacího jazyka určuje logický význam jednotlivých výrazů jazyka (Skoupil, 2007).

### 2.2.2 Objektová metodologie UML

Jazyk UML a jeho struktura

Nástroje CASE (*Computer Aided Software Engineering*, počítačem podporované softwarové inženýrství) patří podle (Kanisová, 2006) mezi specializovaný software na podporu analýzy a návrhu systémů a v současné době všechny objektově orientované CASE nástroje vycházejí z jazyka UML (Kanisová, 2006). Existuje více různých definic jazyka UML. Význam akronymu je podle (Franěk, 2014) UML = *unified modeling language* (tj. unifikovaný modelovací jazyk), kde:

- Unifikovaný: unifikuje Booch, OMT a Objectory modelovací jazyky
- Modelovací: UML je jazyk pro specifikaci, visualizaci, konstrukci a dokumentaci artefaktů SW systémů (Franěk, 2014).

Dále podle (Franěk, 2014) je UML využitelný i pro business modelování a pro modelování ne-SW systémů a v UML lze podle autora modelovat jakýkoliv typ aplikace běžící na jakémkoliv typu a kombinaci HW (hardware), OS (operační systém),

[2] např. ve Windows soubory s příponou exe

programovacím jazyku a sítě. (Franěk, 2014). Podle (Petrlíková, 2005) je všeobecně přijímaným standardem jazyka pro vizuální modelování, který byl původně vytvořen společností Rational a dnes jej spravuje Object Management (Petrlíková, 2005).

Podle (Microsoft, 2019) se jazyk poprvé objevil na scéně v 90. letech, když tři softwaroví inženýři, Grady Booch, Ivar Jacobson a James Rumbaugh, chtěli přijít s méně chaotickým popisem stále složitějšího vývoje softwaru, a zároveň oddělit metodiku od samotného procesu a dnes je UML stále standardním zápisem pro developery, projektové manažery, majitele podniků, podnikatele v oblasti technologií a profesionály v celé řadě odvětví (Microsoft, 2019).

Podle (Kanisová, 2006) je **modelovací jazyk UML** souhrnem především grafických notací k vyjádření analytických a návrhových modelů, s tím, že UML je jazyk, který umožňuje modelovat jednoduché i složité aplikace pomocí stejné formální syntaxe, a proto poskytuje možnost výsledky sdílet s ostatními návrháři, s tím, že vybrané modely jsou pochopitelné i pro zadavatele aplikace a umožní kvalitní vyjasnění požadavků a uživatelů na vytvářený systém (2006, Kanisová).

Podle (Franěk, 2014) jsou další vlastnosti UML například otevřený standard, podpora celého vývojového cyklu, podpora různě aplikačních oblastí, založení na zkušenostech potřebách komunity uživatelů a podpora celé řady nástrojů (Franěk, 2014).

Podle (Tišnovský, 2005) na serveru root.cz je celý jazyk UML založený na třech elementech, které ale nejsou z uživatelského hlediska reprezentovány v textové podobě, ale grafickými značkami v plošném (tj. dvourozměrném) grafu a tyto tři základní elementy jazyka UML se dle své funkce nazývají:

- Předměty
- Relace
- Diagramy (Tišnovský, 2005).

Dále podle (Tišnovský, 2005) jsou předměty elementy zpracovávaného modelu, jež jsou následně členěny do několika navzájem rozdílných podkategorií:

- Prvním typem podkategorií jsou podle autora takzvané **strukturní abstrakce** UML modelu, například programové třídy, aplikační či objektové rozhraní, případy užití, komponenty či uzly.
- Dále podle autora patří mezi předměty i takzvaná **chování**, která v UML diagramu prezentují interakce, tj. vzájemné komunikace mezi jednotlivými objekty. Pomocí chování lze také modelovat stavový stroj, u nějž se stavy specifikují pomocí

přechodů, událostí a aktivit. Chování se v UML diagramu dále podle autora většinou vyznačuje pomocí různým způsobem konstruovaných a různě zakončených šipek či propojovacích čar.

- Dále mezi předměty patří takzvané **seskupení**, které podle potřeby modelu graficky seskupuje části diagramu na nižší úrovni a většinou se podle autora jedná o takzvané balíčky, jež mají tvar stylizované kancelářské složky s popisem umístěným v levé horní části obdélníku (zobrazení seskupení se však podle autora může v různých prostředích odlišovat).
- Posledním a současně velmi jednoduchým typem předmětů jsou **poznámky**, které blíže specifikují vlastnosti a chování dalších elementů UML diagramu (Tišnovský, 2005).

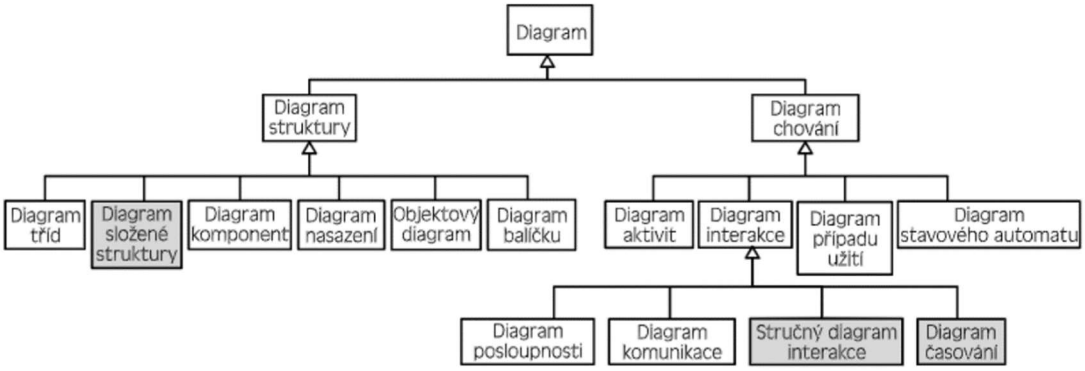
Co se týče **relací**, podle (Tišnovský, 2005) jsou vzhledem k tomu, že je zapotřebí vztahy mezi různými předměty navzájem propojovat, v jazyku UML specifikovány i relace, tj. vztahy mezi různými předměty. V UML jsou podle autora rozeznávány následující podtypy relací:

- **Asociace** – pomocí asociací se modeluje obecná souvislost předmětů, která je však v diagramu UML podle autora přesným způsobem definovaná. Speciální variantou asociace jsou takzvané **kompozice** a **agregace**, které jsou podle autora často používány v objektově orientovaných jazycích a návrzích databází.
- **Závislost** – podle autora se použije, pokud změna v jednom předmětu způsobí změnu v předmětu jiném, nebo mu známým způsobem poskytne požadovanou informaci.
- **Generalizace** – pomocí generalizace se podle autora modeluje stav, kdy je jeden předmět specializací jiného předmětu a tato relace je velmi často používána v objektově orientovaných jazycích, a implementuje se většinou pomocí dědičnosti (inheritance).
- **Realizace** – jedná se o druh vztahu, ve kterém jeden předmět představuje dohodu, za jejíž splnění je odpovědný jiný předmět. V objektově orientovaných jazycích se realizace vytváří pomocí rozhraní (interface) – samozřejmě za předpokladu, že daný OOP jazyk rozhraní podporuje (Tišnovský, 2005).

Posledním základním elementem dále podle (Tišnovský, 2005) jsou **Diagramy**, které zachycují různé aspekty modelovaného systému, jenž nemusí být obecně vyjádřen pouze jedním UML diagramem, protože je možné, například pomocí balíčků (*packages*),

sdružovat více diagramů do jednoho hierarchicky organizovaného celku. Různých typů diagramů dále podle autora existuje velké množství, například diagram případů použití, diagram tříd, diagram objektů, diagram komponent, diagram nasazení, diagram aktivit, stavový diagram, diagram spolupráce a sekvenční diagram a každý z výše vyjmenovaných typů diagramů obsahuje poněkud odlišný repertoár dostupných grafických značek (předmětů a relací) (Tišnovský, 2005).

Podle webu (docs.microsoft.com) rozlišujeme dva hlavní typy diagramů – statický (diagram struktury) a dynamický (diagram chování), od kterých jsou odvozeny další diagramy – rozdělení je pro ilustraci uvedeno na obrázku 1 níže, čerpáno z (Arlow, 2003).




Obrázek 1 Členění diagramů v UML, podle zdroj: (Arlow, 2003)

**Diagram tříd** je podle (Booch, 2005) nejčastěji používaným digramem při modelování objektově orientovaných systémů a představuje statický pohled na strukturu modelovaného systému a znázorňuje množinu tříd, rozhraní a jejich vzájemnou spolupráci a vztahy (Booch, 2005). Podle (Franěk, 2014) citujícího další autory představuje „statický pohled na modelovaný systém“ a jeho úkolem je znázornit typy objektů v systému a jejich vztahy, návrh tříd, jejich odpovědností a následné vytvoření tohoto diagramu je jedním z prvních a základních kroků analýzy navrhovaného programového systému a díky tomu, že diagram tříd zachycuje pravidla modelovaného systému, je podle autora nejdůležitějším podkladem jak pro forward engineering, tak pro reverse engineering (Franěk, 2014).

Use Case Diagram

**Diagram případů užití** (Use Case Diagram) je obecně jeden z klíčových diagramů pro modelování dynamických vlastností systému, subsystému nebo tříd. Prvním, kdo podle (Franěk, 2014) zviditelnil případy užití, byl Ivar Jacobson v roce 1992 a okamžitě po vydání jeho knihy byly případy užití přijaty komunitou objektově orientovaných vývojářů jako základní element při plánování, vývoji a realizaci projektu (Franěk, 2014). Diagram

obecně zobrazuje chování systému tak, jak ho vidí uživatel a jeho účelem je popsat funkcionalitu systému – co se od něj očekává.

Vedle toho Use case scénář popisuje podle webu (w3computing.com) posloupnost kroků popisujících interakci mezi uživatelem a systémem a základními atributy use case scénáře jsou název, který dostatečně reprezentuje cíl use case a hlavní scénář, kterým je číslovaný seznam kroků a krokem je myšlena jedna konkrétní interakce mezi účastníkem a systémem a vedle toho může také obsahovat seznam rozšíření – ke scénářům případů užití neexistuje standardizovaný formát, takže si každá organizace určuje co zahrnout, ale často jsou dokumentovány pomocí šablony dokumentu případu užití předem určené organizací, což usnadňuje čtení případů užití a poskytuje standardizované informace pro každý případ užití (*use case*) v modelu (w3computing.com).

### 2.2.3 Modely vícekriteriálního rozhodování

Podle (Získal, 2001) zkoumá procesy rozhodování vědní disciplína **Teorie rozhodování** (*Decision Theory*), jejímž zakladatelem je H. Simon a teorie rozhodování se rozvíjí od r. 1960 v rámci psychologických věd s přímými aplikacemi v ekonomii, managementu, marketingu a financích a v sociálních a politických vědách. Pro řešení praktických problémů je, dále podle autora, vhodné ztotožnit pojem rozhodování s pojmem řešení problému, a všechny rozhodovací procesy klasifikovat podle kvality informace, jež je při rozhodování k dispozici, do tří kategorií, a to na rozhodovací procesy:

- dobře strukturované,
- nestrukturované,
- semistrukturované (Získal,2001).

Dále podle (Získal, 2001) představuje **Rozhodování** dynamický vědomý proces výběru jedné z možných alternativ, kterou lze dosáhnout požadovaného cíle a **rozhodovací proces** je dále podle autora činnost řídících pracovníků, při niž usilují dojít k optimálním závěrům a správným rozhodnutím a úspěch řídících pracovníků při rozhodování závisí v podstatě na schopnosti vykonávat dvě funkce: analyzovat problém a učinit rozhodnutí. Informace jsou podle autora komoditou, která se musí vědomě a systematicky zpracovat. Rozhodování (řešení problému) dále podle (Získal, 2001) probíhá v čase a v průběhu rozhodování se rozlišují tři fáze rozhodovacího procesu:

- Identifikace problému (*Intelligence*). Dokonalé vymezení problému. Exaktní formulace cílů, analýza omezujících a podpůrných prostředků. Analýza vlastní organizace a blízkého i vzdálenějšího okolí z hlediska uvažovaných záměrů.
- Analýza a řešení problému (*Design*). Analýza identifikovaného problému, tvorba různých alternativ řešení.
- Výběr řešení (*Choice*). Výběr nejvhodnějšího řešení pro daný problém a organizaci v daném (Získal, 2001).

Podle (Šubrt, 2019) znamená rozhodování zvolení jediného rozhodnutí z několika možných alternativ rozhodnutí a rozhodovací proces je tedy postup řešení rozhodovacích problémů, ve kterých je nutno vybrat jedno rozhodnutí z více možných variant řešení a cílem je dále podle autora vybrat tu alternativu, která je z určitého hlediska nejvýhodnější a efekt plynoucí z realizace jednotlivých alternativ je ovlivňován budoucí situací, která však není rozhodovatelem ovlivnitelná. Rozhodovací proces je dále podle autora multidisciplinární problém, protože řešení je závislé na jeho věcné stránce (oblasti řešeného problému) a procedurální stránce (metody a postupy řešení). Dále podle autora jsou pro tvorbu vhodného matematického modelu důležité prvky rozhodovacího procesu, kterými jsou objekt a subjekt rozhodování, a alternativy rozhodnutí (z čeho se vybírá) (Šubrt, 2019).

**Modely vícekriteriálního rozhodování** podle (Šubrt, 2019) zobrazují rozhodovací problémy, v nichž se důsledky rozhodnutí posuzují podle více kritérií a modely vícekriteriálního hodnocení variant jsou pak zadány pomocí konečného seznamu variant a jejich ohodnocení podle jednotlivých kritérií. Teorie a model vícekriteriální analýzy variant se dále podle autora zabývá problémy, jak vybrat jednu nebo více variant z množiny přípustných variant a doporučit je k realizaci, s tím, že by rozhodovatel měl při výběru variant postupovat maximálně objektivně, k čemuž mu slouží aparát různých postupů a metod analýzy variant (Šubrt, 2019). Modely vícekriteriálního rozhodování dále podle autora zobrazují takové rozhodovací problémy, v nichž se rozhodnutí posuzují ne podle pouze jednoho, ale podle více kritérií a zohlednění několika kritérií při rozhodování přináší do řešení problému konflikty, které plynou z jejich protichůdnosti. Cílem rozhodovacího procesu je najít takovou variantu, která bude podle všech možných kritérií hodnocena co nejvyšším počtem bodů (optimální nebo kompromisní varianta). **Varianty** jsou podle autora konkrétní rozhodovací možnosti, předmět vlastního rozhodování, jsou realizovatelné a nejsou logickým nesmyslem (Šubrt, 2019).

**Kritérium** je dále podle (Šubrt, 2019) hledisko hodnocení variant, může být kvalitativní nebo kvantitativní, a volba jednotlivých kritérií je podle autora velmi důležitá. Kritéria dále musí být nezávislá a měla by pokrývat všechna hlediska výběru, přitom jich dále podle autora nesmí být zbytečně velký počet - aby problém nebyl nepřehledný.

Dále podle (Šubrt, 2019) kritéria dělíme podle různých hledisek, autor například uvádí, že podle povahy kritéria rozlišujeme kritéria maximalizační a minimalizační, nebo podle kvantifikovatelnosti kritéria rozlišujeme kvantitativní a kvalitativní. Podle autora je také pro řešení problému velmi důležité, zda a jak je kritérium preferováno před jiným. Preference kritéria vyjadřuje důležitost tohoto kritéria v porovnání s ostatními, a může být vyjadřována různým způsobem, například mohou být stanoveny aspirační úrovně kritérií, pořadí kritérii, váhy kritérií nebo způsob kompenzace kriteriálních hodnot, a preference nemusí být známy vůbec (Šubrt, 2019).

**Váha kritéria** je podle (Šubrt, 2019) obecně hodnota z intervalu <0;1>, která vyjadřuje relativní důležitost tohoto kritéria v porovnání s kritérií ostatními a součet vah všech kritérií je roven jedné.

Co se týče variant, definuje (Šubrt, 2019) varianty se speciálními vlastnostmi – dominující, vzájemně nedominované, ideální, bazální a kompromisní varianty. Ohledně klasifikace úloh vícekriteriální analýzy variant je klasifikace podle autora možná především podle dvou základních hledisek a to podle cíle řešení úlohy (výběr jedné či několika variant, úplného uspořádání, rozdělení množiny variant na efektivní a neefektivní) a podle informace, s jakou úloha pracuje – žádná, nominální, ordinální nebo kardinální informace, s tím, že kardinální informace představuje tu, která má kvantitativní i kvalitativní charakter a vyjadřuje o kolik, či jak moc, je jedno hodnocení lepší než druhé (Šubrt, 2019).

Saatyho metoda

**Saatyho metoda** lze podle (Šubrt, 2019) použít k určování vah kritérií, když je hodnotí rozhodovatel nebo pouze jeden expert, a jde o **metodu kvantitativního párového porovnávání kritérií**. Pro ohodnocení párových kritérií se používá devítibodová stupnice a je možné používat i mezistupně (hodnoty 2, 4, 6, 8):

- 1 - stejně významná, rovnocenná kritéria i a j
- 3 - slabě preferované kritérium i před j
- 5 - silně preferované kritérium i před j
- 7 - velmi silně preferované kritérium i před j

- 9 - absolutně preferované kritérium i před j (Šubrt, 2019)

Postup je dále podle autora takový, že expert porovná každou dvojici kritérií a velikosti preferencí i-tého kritéria vzhledem k j-tému kritériu zapíše do Saatyho matice **S = ($s_{ij}$)**, která je uvedena na následujících obrázku:

$$S = \begin{pmatrix} 1 & s_{12} & \cdots & s_{1n} \\ 1/s_{12} & 1 & \cdots & s_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 1/s_{1k} & 1/s_{12} & \cdots & 1 \end{pmatrix}$$

Obrázek 2 Saatyho matice podle (Šubrt, 2019)

Saatyho matice je čtvercová řádu nxn, reciproká a na diagonále je vždy hodnota 1 (protože každé kritérium je samo sobě rovnocenné) a dále podle autora, jelikož nebývají prvky této matice většinou dokonale konzistentní, měří se míra konzistence – a to například indexem konzistence (C.I., *consistency index*), s tím, že matice je považována za dostatečně konzistentní, jestliže je index konzistence menší než 0,1 (Šubrt, 2019).

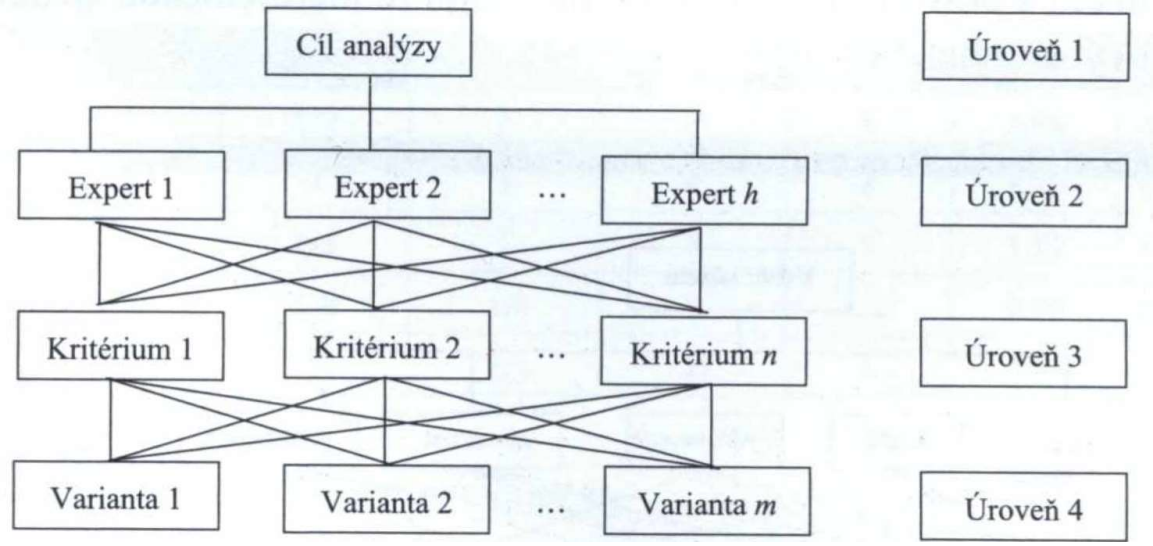
AHP metoda

Metoda **AHP** (*Analytic Hierarchy Process*, Analytický Hierarchický Proces) byla podle (Šubrt, 2019) navržena prof. Saatym v roce 1980, poskytuje rámec pro přípravu účinných rozhodnutí ve složitých rozhodovacích situacích a pomáhá zjednodušit a zrychlit přirozený proces rozhodování. AHP je dále podle autora metodou rozkladu složité nestrukturované situace na jednodušší komponenty; vytváří tedy **hierarchický systém problému** a tento hierarchický systém je podle autora zobecněním — rozšířením možností vícekriteriálního rozhodovacího systému. Dále podle autora se na každé úrovni hierarchické struktury použije Saatyho metoda kvantitativního párového porovnání a pomocí subjektivních hodnocení párového porovnání pak tato metoda přiřazuje jednotlivým komponentám kvantitativní charakteristiky vyjadřující jejich důležitost, a syntézou těchto hodnocení se pak stanoví komponenta s nejvyšší prioritou, na níž se rozhodovatel zaměří s cílem získat řešení rozhodovacího problému (Šubrt, 2019).

Základní prvky a kroky metody AHP dále podle (Šubrt, 2019) jsou:

- Konstrukce hierarchie problému
- Párové porovnávání prvků v jednotlivých hierarchických úrovních.
- Syntéza získaných preferencí a volba nejvýhodnější alternativy (Šubrt, 2019).

Pod pojmem **hierarchická struktura** se podle autora pak rozumí struktura obsahující několik úrovní, přičemž každá z nich obsahuje několik prvků a uspořádání jednotlivých úrovní hierarchické struktury odpovídá uspořádání od obecného ke konkrétnímu, s tím, že čím obecnější jsou prvky ve vztahu k danému rozhodovacímu problému, tím zaujímají v hierarchii vyšší úroveň a naopak (Šubrt, 2019). Příklad struktury typické úlohy vícekriteriální analýzy variant tříúrovňové hierarchie je uveden na obrázku níže.




Obrázek 3 Hierarchická struktura typické úlohy vícekriteriální analýzy variant podle (Šubrt, 2019)

Rozhodnutí je tedy obecně výběr jedné nebo více variant z množiny všech přípustných variant. V případě této práce je to výběr z množiny knihoven, která je nejvhodnější k automatizaci procesu kancelářské činnosti. Důležitým pojmem z hlediska použité metody AHP je míra konzistence, která se například měří indexem konzistence a který byl v rámci této diplomové práce ověřován za pomocí maticových funkcí a nástroje Hledání řešení (*Goal Seek*) tabulkového procesoru Microsoft Excel, za pomocí nalezení největšího kořene polynomu.

# 3 Teoretická východiska

## 3.1 Programovací jazyk Python a východiska automatizace

### 3.1.1 Představení jazyka

Python je podle (Gaura, 2021) interpretovaný procedurální, objektově orientovaný a funkcionální programovací jazyk, který v roce 1990 navrhl Guido van Rossum, a je vyvíjen jako open source projekt, který zdarma nabízí instalační balíky pro většinu běžných platforem (Unix, Windows, Mac OS) a ve většině distribucí systému GNU/Linux je Python součástí základní instalace. Podle autora se jedná o dynamický interpretovaný jazyk a také jej zařazujeme mezi skriptovací jazyky. Python byl podle autora navržen tak, aby umožňoval tvorbu rozsáhlých, plnohodnotných aplikací (včetně grafického uživatelského rozhraní) a jedná se o hybridní jazyk (nebo také víceparadigmatický) (Gaura, 2021). Jeho současnou verzí je podle (python.org) verze 3.10.2 z roku 2022. Například podle (Majer, 2021) mezi výhody Pythonu patří:

- Jednoduchá a přehledná syntaxe
- Dynamický styl zápisu
- Automatické přidělování paměti
- Aktivní komunita
- Velké množství knihoven

Vedle toho jsou jako nevýhody opět podle (Majer, 2021) uvedeny následující:

- Jelikož je Python jazykem interpretovaným, může být oproti jiným jazykům pomalejší a tudíž méně vhodný pro větší a složitější aplikace právě z důvodu rychlosti
- Python je vyšším programovacím jazykem, takže není vhodný k psaní programů na úrovní hardwaru
- U některých speciálních úkolů může být automatické přidělování paměti nevýhodou (Majer, 2021)

Dokument, který podle webu (peps.python.org) obsahuje pokyny a osvědčené postupy pro psaní kódu v jazyce Python se nazývá **PEP** (*Index of Python Enhancement Proposals*, Rejstřík návrhů na vylepšení jazyka Python). Byl vytvořen v roce 2001 autorem jazyka a poslední modifikace je z ledna 2022 (python.org).

### 3.1.2 Virtuální prostředí a instalace modulů

Podle webu (docs.microsoft.com) je prostředí Pythonu kontext, ve kterém se spouští kód Pythonu, zahrnuje globální, virtuální a conda prostředí a skládá z interpretu, knihovny (obvykle standardní knihovny Pythonu) a sady nainstalovaných balíčků. Tyto součásti společně určují platné jazykové konstrukce a syntaxi, funkce operačního systému, ke kterým je možné přistupovat, a balíčky, které je podle (docs.microsoft.com) možné používat. Ohledně **globálního prostředí**, si dále podle (docs.microsoft.com) každá instalace Pythonu udržuje vlastní globální prostředí a každé prostředí se skládá z konkrétního interpretu jazyka Python, jeho standardní knihovny, sady předinstalovaných balíčků a dalších balíčků, které jsou nainstalovány, když je toto prostředí aktivováno. Instalací balíčku do globálního prostředí se tento balíček zpřístupní všem projektům, které toto prostředí používají a pokud je prostředí umístěno v chráněné oblasti souborového systému vyžaduje instalace balíčků oprávnění správce (docs.microsoft.com).

Ačkoli je podle (docs.microsoft.com) práce v globálním prostředí snadný způsob, jak začít, časem se prostředí zahltí mnoha různými balíčky, nainstalovanými různé projekty, a tato nepřehlednost ztěžuje testování aplikace proti konkrétní sadě balíčků (*packages*) se známými verzemi, s tím, že ke konfliktům může dojít také v případě, že dva projekty vyžadují nekompatibilní balíčky, nebo různé verze stejného balíčku. Toto je podle webu důvod, proč vývojáři pro projekt často vytvářejí **virtuální prostředí** (docs.microsoft.com).

Virtuální prostředí je dále podle (docs.microsoft.com) podsložka v projektu, která obsahuje kopii určitého interpretu a v případě aktivace se všechny instalované balíčky nainstalují pouze do podsložky tohoto prostředí. Když je pak v tomto prostředí spuštěn program Python, je jasné, že běží pouze v rámci těchto konkrétních balíčků (docs.microsoft.com).

Vedle výše uvedených je prostředí conda takové prostředí, které je vytvořeno pomocí nástroje conda, nebo pomocí integrované správy conda. Vzhledem k tomu, že prostředí conda nejsou uložena s projektem, fungují podobně jako globální prostředí, tedy například instalací nového balíčku do prostředí conda se tento balíček zpřístupní všem projektům, které toto prostředí používají (docs.microsoft.com).

Mezi moduly, balíčky a knihovnami je podle (learnpython.com) následující vztah:

- Moduly (*modules*) jsou v podstatě soubory souvisejícího kódu uložené v souboru s příponou .py, ve kterých je možné definovat funkce, třídy nebo proměnné, a je do nich možné zahrnout spustitelný kód.
- Balíčky (*packages*) jazyka Python jsou v podstatě soubory (složky) s kolekcí modulů, které umožňují hierarchickou strukturu jmenného prostoru modulu (*hierarchical structure of the module namespace*). Moduly tedy za účelem organizace, můžeme do balíčků (a podbalíčků) organizovat. Aby byly složky považovány za balíček (nebo podbalíček), musí obsahovat soubor s názvem `**_init__.py**`, což je soubor obvykle obsahující inicializační kód příslušného balíčku.
- Vedle toho je knihovna souhrnný termín (*umbrella term*) označující opakovaně použitelný kus kódu, knihovna jazyka Python obvykle obsahuje kolekci souvisejících modulů a balíčků (learnpython.com).

V praxi se termíny často používají zaměnitelně, ale obecně se předpokládá, že zatímco balíček je kolekce modulů, knihovna je kolekce balíčků. Knihoven jsou k dispozici tisíce. Podobně jako knihovny jsou aplikační rámce (*frameworks*) jazyka Python souborem modulů a balíčků, které pomáhají urychlit proces vývoje, avšak obvykle jsou složitější než knihovny, neboť zatímco knihovny obsahují balíčky, které provádějí konkrétní operace, frameworky obsahují základní tok a architekturu aplikace (learnpython.com).

Posledním důležitým bodem v rámci této podkapitoly je nástroj **PIP**[3], což je podle webu (pypi.org) instalátor balíčků (*packages*) pro Python, pomocí kteréhož je možné instalovat knihovny (a moduly a balíčky) z **Python Package Index** (pypi.org).

### 3.1.3 Rešerše k problematice automatizace a kategorizace úkonů

V této podkapitole jsou v rámci přehledu k probírané tématice ve stručnosti definovány pojmy, s kterými se dále v práci operuje, a to automatizace a pracovní činnost. Dále jsou představena specifika jednotlivých relevantních souborových formátů  a v závěru je zde uveden stručný popis souvisejících trendů.

Automatická činnost je obecně posloupnost úkonů provedena automaticky podle předem zadaného programu bez dalšího zásahu člověka. Výchozím zdrojem je především

[3] rekurzivní zkratka, která může znamenat buď „*Pip installs Packages*“ („Pip instaluje balíčky"), nebo „*Pip installs Python*“ ("Pip instaluje Python")

studie „*Automation with Intelligence*“ a dále pak studie „*Automatizace práce v ČR*” z roku 2018 obě od společnosti Deloitte a volně dostupné na internetu.

Podle studie společnosti (deloitte.com*)* je vzhledem ke struktuře české zaměstnanosti a predikci technologických možností odhadnut potenciál pro automatizaci odpovídající 51 % pracovních míst. Robotická a inteligentní automatizace je dále podle Deloitte implementace automatizací napodobujících lidskou činnost, interakce a rozhodnutí s využitím robotických a kognitivních technologií, jež zlepšují produktivitu pracovní síly a efektivitu procesů, RPA a *Automated regulatory compliance*[4] (deloitte.com*).* Více k problematice automatizace a současných trendů v podkapitole 3.1.3.6.

Pracovní činnost

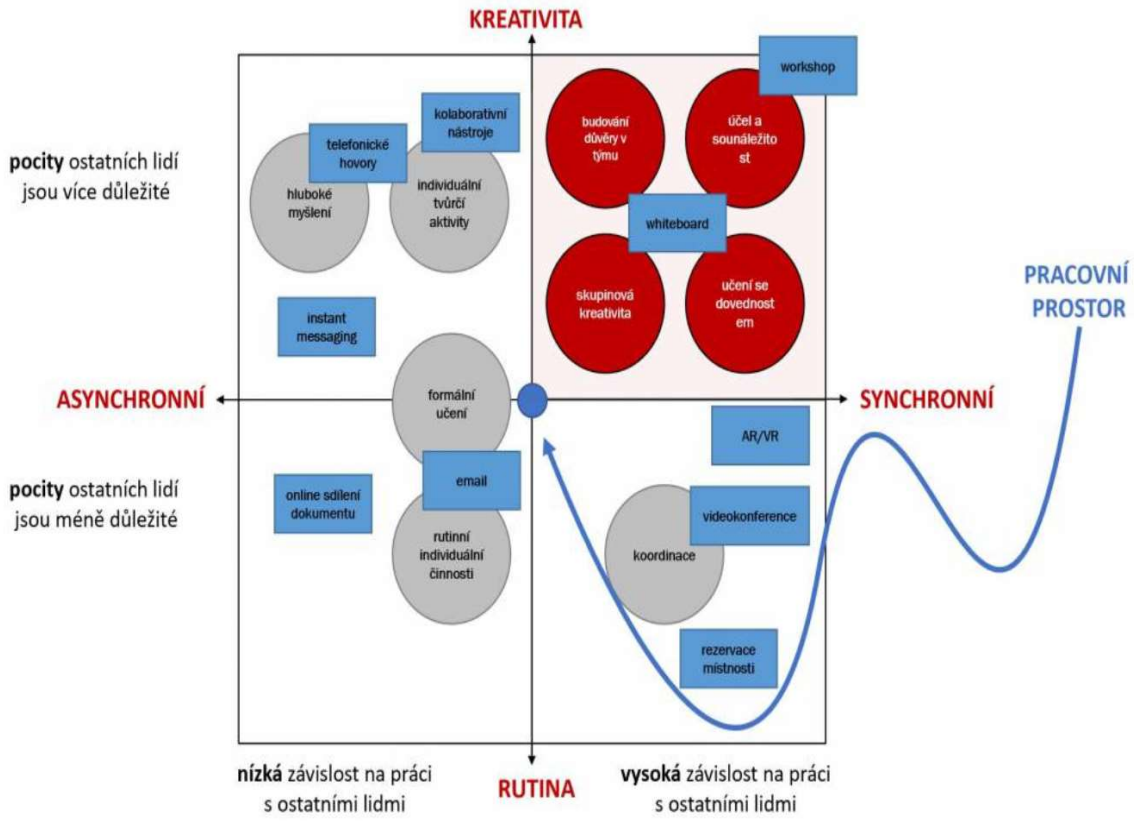
Na základě citované studie[5] (deloitte.com*)* dělíme typy práce podle dvou hledisek. Jednak se může jednat o práci manuální nebo práci znalostní, z dalšího hlediska může být práce buď rutinní nebo nerutinní. Rutinní práce je taková, kterou lze rozdělit na dílčí úkony a ty následně jednoznačně kodifikovat například v rámci počítačového programu (*Deloitte, 2018).* Oproti tomu stojí podle studie úkony nerutinní, které vyžadují řešení problémů, intuici, kreativitu, přesvědčovací schopnosti a situační přizpůsobivost. Rutinní práce je podle studie obvykle prováděna ve strukturovaném prostředí, nerutinní práce naopak tam, kde jsou prostředí a situace méně předvídatelné a vzájemnou kombinací těchto dvou dimenzí vznikají čtyři kategorie – uvedené na obrázku níže, z citované studie (deloitte.com).

[4] Automatizované dodržování předpisů, jinde např. *Compliance Automation*, „Automatizace zajišťování shody“

[5] citující studii Skills, Tasks and Technologies:Implications for Employment and Earnings, jejímž autor je D. Acemoglu


Obrázek 4: Kategorizace pracovních úkonů, zdroj: (Deloitte,2018)

Na následujícím obrázku je ilustrativně uvedena Matice pracovních činností podle článku *„Jak ovlivňují "pracovní činnosti" návrat do kancelářských prostor?“* na stránce (avmedia.cz).




Obrázek 5 Matice pracovních činností podle webu (avmedia.cz)

Kancelářská činnost

Pro účely této práce představuje kancelářská činnost obecně pracovní úkony zahrnující kancelářský software (práci s kancelářským balíkem, viz dále), sběr a zpracování dat, korekturu textu a dat, měření a kontrolu kvality. Obecně se jedná například o podstatnou část náplně pracovní činnosti „technicko-administrativně-hospodářských“ pracovníků. Například podle Katalogu prací[6], ve veřejných službách a správě se jedná především o *Díl 1.01 - Práce ve správě organizace*, pro představu jsou v Katalogu blíže popsány i jednotlivé úkony. Vedle toho se s kancelářskou činností pro účely práce částečně mohou překrývat sekce N a M Klasifikace ekonomických činností (CZ-NACE), a obecně činnosti a procesy řazené pod profesi (procesem) Office management. Pro účely splnění stanovených cílů je dále v rámci této diplomové práce kancelářská činnost zredukována na interakci s pěti kategoriemi (většinou) kancelářského SW, ty jsou následující:

- Tabulkové procesory (*spreadsheet*)
- Textové procesory
- Portable Document Format (PDF)
- Nástroje na tvorbu prezentací
- Klient ke správě elektronické pošty

Dále byly pro účely kvantifikace k praktické části vybrány demonstrativní úkony, které jsou obvykle součástí této činnosti, nebo alespoň částí jako krok v procesu, a které mohou být společné všem výše uvedeným kategoriím. Vzhledem k tomu, že se jedná o úkony[7], hodnocené a měřené v rámci praktické části, jsou níže ve stručnosti definovány. Bylo vybráno následující:

- Tvorba dokumentu požadovaného formátu [1]
- Extrakce dat z požadovaného formátu [2]
- Manipulace s obsahem požadovaného formátu [3]
- Manipulace se souborem [4]
- Manipulace s metadaty (vlastnosti souboru) [5]

V rámci kontextu práce je hodnocena možnost vytvořit dokument (soubor) požadovaného formátu s předem definovaným obsahem [**1**], s tím, že autor vychází

[6] podle Nařízení vlády č. 222/2010 Sb. Nařízení vlády o katalogu prací ve veřejných službách a správě účinné od 1.10.2010 (Aktuální znění 01.01.2022 (verze 5)

[7] případně úkoly a zadání (*specified tasks and user objectives*) z hlediska funkční úplnosti podle normy ISO/IEC 25010

z předpokladu, že v případě kdy toto knihovna umožňuje, je možnost i hromadného vytváření souborů v souborovém systému, konverze do jiného souborového formátu, a tak podobně. Extrakcí dat pak čtení tohoto obsahu [**2**]. Manipulací s obsahem se obecně rozumí editace, tj. činnosti nad rámec prohlížení a čtení přímo nativně v souboru hodnocené přípony [**3**], v závislosti na formátu se může jednat například o otočení stránky souboru, změna fontu, ale podrobnější informace o možnostech budou uvedeny u jednotlivých formátů a knihoven dále. Manipulací se souborem se v kontextu hodnocení práce rozumí především slučování a rozdělování [**4**] v rámci souborového systému na lokálním uložišti. Metadata obecně jsou "data, která poskytují informace o jiných datech" a existuje mnoho různých typů metadat. V rámci kontextu této práce se jedná o identifikační prvky připojené k souboru - vlastnosti dokumentu, a zahrnují údaje jako například název, jméno autora nebo předmět [**5**].

Souborový systém podle obecné definice obsahuje a organizuje uložené informace jako soubory a adresáře, s tím, že soubory jsou rozlišeny jmény, adresáře slouží k organizaci souborů do logických celků a vlastním obsahem souborů jsou data. Formát souboru (typ souboru) v obecné definici označuje standard, na základě něhož jsou elektronická data uložena do počítačového souboru.

Podle (Lavrinčík, 2018) je **souborový systém** (*file system*) v informačních a komunikačních technologiích označení pro způsob organizace dat ve formě souborů (a většinou i adresářů) tak, aby k nim bylo možné snadno přistupovat a souborové systémy jsou uloženy na vhodném typu elektronické paměti, která je umístěna přímo v počítači nebo může být zpřístupněna pomocí počítačové sítě. Souborový systém operačního systému dále podle autora zajišťuje ukládání a čtení dat paměťového média tak, aby s nimi uživatelé mohli pracovat ve formě souborů a adresářů a základní ideou souborového systému je potom zpřístupnění a ukládání dat pomocí hierarchicky organizovaného systému adresářů a souborů. Operační systém Windows používá u každého souborového systému pro označení jeho kořene následující písmeno abecedy (Lavrinčík, 2018).

Dále podle (Lavrinčík, 2018) mohou mít různé souborové systémy různé vlastnosti a omezení, podle stejného autora například:

- maximální velikost paměťového média, kterou je daný systém schopen pokrýt,
- délka souboru,
- délka jména souboru a přípony souboru,
- počet zanořených podadresářů,

- podporovaná znaková sada (Lavrinčík, 2018).

Co se **metadat** týče, např. podle (Celbová, 2003) se jedná o strukturovaná data, která nesou informace o primárních datech, pojem metadat je používán především v souvislosti s elektronickými zdroji a vztahuje se k datům v nejširším smyslu slova (datové soubory, textové informace, obrazové informace, hudba aj.). Funkce metadat je popisná, selekční a archivační (Celbová, 2003). Standardem pro metadatový popis digitálních objektů často vyjadřovaných prostřednictvím XML (*Extensible Markup Language*) je Dublin Core, známý také jako Dublin Core Metadata Element Set (**DCMES**). Sada metadatových prvků Dublin Core[8] obsahuje 15 základních prvků pro popis zdrojů a toto jádro Dublin Core bylo formálně standardizováno jako ISO 15836, ANSI/NISO Z39.85 a IETF RFC 5013 a je součástí širšího souboru DCMI Metadata Terms (*dublincore.org*). Ukázka XML s Dublin Core metadaty je uvedena na následujícím obrázku.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties
    xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <dc:title>Core Document Properties Exploration</dc:title>
  <dc:subject>PowerPoint core document properties</dc:subject>
  <dc:creator>Steve Canny</dc:creator>
  <cp:keywords>powerpoint; open xml; dublin core; microsoft office</cp:keywords>
  <dc:description>
    One thing I'd like to discover is just how line wrapping is handled
    in the comments. This paragraph is all on a single
    line._x000d__x000d_This is a second paragraph separated from the
    first by two line feeds.
  </dc:description>
  <cp:lastModifiedBy>Steve Canny</cp:lastModifiedBy>
  <cp:revision>2</cp:revision>
  <dcterms:created xsi:type="dcterms:W3CDTF">2013-04-06T06:03:36Z</dcterms:created>
  <dcterms:modified xsi:type="dcterms:W3CDTF">2013-06-15T06:09:18Z</dcterms:modified>
  <cp:category>analysis</cp:category>
</cp:coreProperties>
```

Obrázek 6 Ukázka XML s Dublin Core metadaty pro soubor aplikace MS Word (zdroj: python-docx.readthedocs.io)

Obecně jako metadata jsou někdy označované vlastnosti dokumentu (*Document properties*), které popisují obsah souboru, jsou uchovávána uvnitř souboru a jsou zcela nezávislá na souborovém systému – a jejich úprava a čtení je relevantní z hlediska hodnocení a realizace v praktické části. Vedle toho obecně existují například atributy

[8] relevantní pro účely práce například z toho důvodu, že v rámci porovnávaných knihoven velkou část tvoří formáty OOXML, kde je Dublicore ve formě standardizovaných metadat

souborů (*File Attributes*), které jsou uchovávány souborovým systémem a při přenosu souboru na jinou platformu se ztratí.

Kancelářský balík

Kancelářský balík[9] je obecně integrovaný program používaný v kancelářské praxi, a lze je obecně podle funkčnosti rozdělit na desktopové, online a mobilní. De facto standardem je kancelářský balík Microsoft Office zahrnující různé aplikace v závislosti na edici. K pochopení vývoje je třeba přiblížit cestu k tomuto stavu.

Podle (Gandal, 2013) byly nejdůležitějšími produkty kancelářského softwaru v 90. letech tabulkové procesory, textové procesory a kancelářské balíky/sady (*office suites*), které kombinovaly tabulkový procesor a textový procesor s dalšími funkcemi a programy s přidanou hodnotou (*value-added features and programs*). Trh s kancelářským softwarem (*office productivity software*) dále podle autora zažil během 90. let dramatické strukturální změny, a v letech 1991-1998 enormní růst (tj. za období ke kterému již máme k dispozici konzistentní údaje) a kromě toho podle autora zaznamenal trh posun od softwarových produktů pro operační systém DOS k produktům pro Windows, a k posunu vedoucí pozice na trhu produktů od společností Lotus (na trhu tabulkových procesorů) a WordPerfect (na trhu textových procesorů) ke společnosti Microsoft, a konečně, došlo ke změně strategie společnosti Microsoft, která přešla od prodeje samostatných produktů k prodeji **kancelářských balíků** (Gandal, 2013).

Vzhledem k tomu, že je pro praktickou část relevantní kancelářský balík poskytovaný společností Microsoft, bodově alespoň přehled vývoje MS Office podle webu (1plus1tech.com, 2018):

- MS Office oficiálně uveden 19. listopadu 1990.
- 1992 - MS Office 3.0 obsahoval Word 2.0 a PowerPoint a Excel ve verzi 4.0. Čísla verzí nebyla jednotná; Microsoft je správně uspořádal až po vydání Office 95.
- 1995 - Office 95 Office 7.5) Název změněn tak, aby odpovídal číslům verzí všech aplikačních programů uvnitř sady.
- 1999 - Office 2000 Microsoft v této verzi nabídl lepší zabezpečení a mnoho aktualizací předchozích aplikací.
- 2001 - Office XP - téměř všechny funkce dostupné uživatelům pracujícím v omezeném režimu byly k dispozici v systému XP.

[9] *office suite, office software, office productivity software…*

- 2003 - Office 2003 - bezpečnostní funkce, spoustu funkcí a pěkné panely nástrojů a ikony. Kromě hvězdného vzhledu byly bohaté funkce přehledně zařazeny do různých karet nabídek.
- Office 2007 - Rozhraní Ribbon (seznam nástrojů pro formátování a vytváření dokumentů.)
- Office 2010 - Webové aplikace Office
- Office 2013 - dotykový režim, Office 2013 i Office 365 využívají cloud computing. (1plus1tech.com, 2018)

V reakci na trend cloudových řešení je dnes poskytovaný produkt této formy označovaný jako Microsoft 365 (do dubna 2020 Office 365) a jedná se o soubor cloudových služeb na bázi předplatného a v podstatě na všech platformách, takže se funkční rozdělení do určité míry stírá.

Vedle kancelářských balíků Microsoftu, u kterého lze obecně konstatovat, že má stále dominantní postavení jak u nás tak ve světě, jsou dalšími relevantními kancelářskými balíky obecně například **Libreoffice** (který v podstatě nahradil dříve oblíbený OpenOffice, s ukončeným vývojem) v menší míře pak **iWork** jako kancelářský balík aplikací vytvořený společností Apple Inc pro operační systémy macOS a iOS (též multiplatformně dostupný prostřednictvím iCloud) a především produkt společnosti Google **Google Workspace** (dříve známý jako G Suite), zahrnující Dokumenty, Tabulky a Prezentace (*Google Docs, Sheets* a *Slides*), které jsou však v desktopové verzi nabízeny především online.

Podle webu (greengeeks.com) měl k roku 2020 v amerických podnicích 47,63% podíl na trhu Microsoft Office 365. Mimo podnikovou sféru ve Spojených státech možná překvapivě ovšem v oblíbenosti vede Google Workspace, s podílem na trhu 59.41% (greengeeks.com, 2022) – uváděným důvodem je, že to je právě služba založena na cloudu.

Podrobněji k sadám kancelářských nástrojů, jejich souborových formátů včetně elementárního srovnání kancelářských balíků MS Office a LibreOffice a jejich formátů se tato práce bude věnovat v závěru další části 3.1.3.1.

#### 3.1.3.1 Relevantní souborové formáty

Pro snazší orientaci je uveden demonstrativní přehled vybraných typů (kategorií) software, z hlediska klasifikačního schématu, jenž je použito v praktické části práce. Toto rozdělení je účelové pro možnost porovnání jednotlivých úkonů, a je uvedeno v tabulce 1.

| Název skupiny typu SW | Popis/Funkce | Relevantní aplikace | Relevantní přípony |
|---|---|---|---|
| **Tabulkové procesory (*spreadsheet*)** | Zpracování informace na virtuálním listu | Microsoft Excel | XLS XSLX CSV ODS |
| **Textové procesory** | Vytváření formátovaného textu | Microsoft Word | DOC DOCX ODT |
| **Portable Document Format (PDF)** | Přenositelný formát souborů | Adobe Acrobat | PDF (PDF-A…) |
| **Nástroje na tvorbu prezentací** | Tvorba prezentací | Microsoft PowerPoint | PPT PPTX ODP |
| **Klient ke správě elektronické pošty** | Správa e-mailů | Microsoft Outloook | MSG PST OST EML |

Tabulka 1 Kategorizace SW k praktické části práce, zdroj: vlastní

Jak bylo řečeno, zohledňuje kategorizace aspekty softwaru z funkčního hlediska, pro použití v rámci komparace knihoven pro účely automatizace kancelářské činnosti, tudíž je přehled značně zúžený a subjektivní podle výběru a zkušeností autora. Například kancelářský balík MS Office obsahuje vedle uvedených aplikací (tj. součástí procesu kancelářské činnosti může být) i MS Access[10] nebo MS OneNote[11]. Případně mohou být podle autora této diplomové práce z pohledu jiného uživatele součástí kancelářské činnosti různé oborové aplikace, například statistický, knihovnický nebo právní software, grafické editory, systémy CAD (*computer-aided design*, počítačem podporovaný návrh) a GIS (*Geographic information system*, Geografický informační systém) a v rámci těchto další specializované systémy.

Formát souboru

Obecně se jedná o pojem odvozený z pojmu „datový formát“, kdy formát souboru obecně určuje způsob uložení dat pro určitou aplikaci, respektive standard, na základě kterého jsou elektronická data uložena do počítačového souboru s názvem a obvykle příponou souboru[12]. Podle webu (kosek.cz) označuje formát souboru způsob logického uspořádání dat v souboru a místo popisu dat se používají jména formátů jako PDF

[10] databázový software Microsoftu s příponou „*.accdb*“
[11] poznámkový software s příponou „*.One*“
[12] Přípona souboru v systému Windows určuje akci která je se souborem provedena.

(*Portable Document Format,* Přenosný formát dokumentů), CSV (*Comma-separated values*, hodnoty oddělené čárkami) a tak podobně, a formáty lze rozdělit na textové a binární, kde textový obsahuje textová data (zobrazitelné znaky, konce řádků a případný konec souboru), a binární má podle autora alespoň část informací, která je vyjádřena jinak než čitelnou posloupností zobrazitelných znaků. Soubor obsahující vlastní text a formátovací značky je **dokument**. Dále podle autora je souborový formát, jehož specifikace je volně dostupná **otevřený**, uzavřené jsou pak utajovány (kosek.cz). Další možná definice pro **otevřený formát** je podle zákona (106/1999 Sb.), kdy je to *formát datového souboru, který není závislý na konkrétním technickém a programovém vybavení a je zpřístupněn veřejnosti bez jakéhokoli omezení, které by znemožňovalo využití informací obsažených v datovém souboru*".

S problematikou otevřených standardů souvisí standardizační organizace, neboť jsou otevřené standardy definovány pomocí specifikací (standardů/norem), vzniklých na neutrální půdě mezinárodních nebo průmyslových standardizačních organizací. Mezi nejznámější standardizační organizace podle (kosek.cz) patří:

- ISO/IEC
- W3C
- OASIS
- ECMA (kosek.cz).

Pro účely práce jsou pak relevantní především standardy ISO/IEC a OASIS.

ISO (Mezinárodní organizace pro normalizaci) a IEC (Mezinárodní elektrotechnická komise) podle (iso.org) tvoří specializovaný systém pro celosvětovou normalizaci, a národní orgány, které jsou členy ISO nebo IEC, se podílejí na tvorbě mezinárodních norem prostřednictvím technických výborů zřízených příslušnou organizací pro jednotlivé oblasti technické činnosti (iso.org), vedle toho je OASIS[13] neziskový normalizační orgán zabývající se přijímáním otevřených standardů v různých oblastech (oasis-open.org).

Obecně pro kancelářskou činnost jsou z hlediska formátů vedle PDF relevantní především uvedené přípony MS Office, ale další knihovny jazyka Python většinou umožňují ukládat soubory ve formátu i dalších kancelářských balíků a vzájemnou konverzi, z toho důvodu jsou některá specifika jednotlivých formátů uvedena.

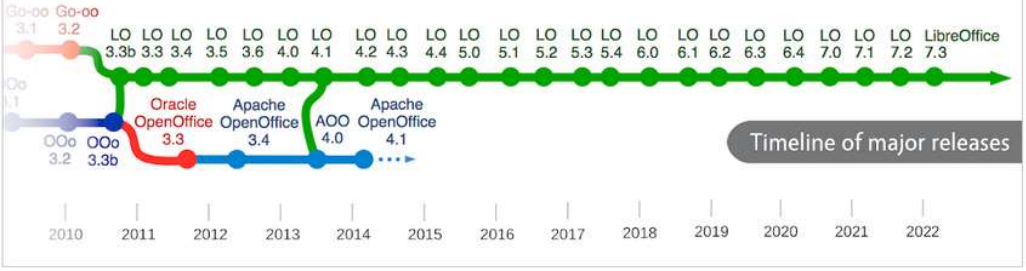
[13] *Organization for the Advancement of Structured Information Standards*, Organizace pro rozvoj standardů strukturovaných informací

Co se týče souborových formátů bude tato práce v následujících částech vycházet především z webové stránky „*The Sustainability of Digital Formats Web site*“ (Udržitelnost digitálních formátů) kde jsou podle webu (loc.gov) uvedeny popisy digitálních formátů jako "fdds" (*format description documents*, dokumenty popisu formátu). Tyto popisy byly původně vytvořeny pod záštitou programu Kongresové knihovny National Digital Information Infrastructure and Preservation Program (NDIIPP, česky Národní program pro digitální informační infrastrukturu a uchovávání informací). Dále podle tohoto webu (loc.gov) byly první popisy vypracovány jako statické soubory HTML v roce 2003 a v následujících letech průběžně aktualizovány a doplňovány, s tím, že koncem roku 2007 začal proces tvorby přecházet do režimu XML (*Extensible Markup Language*). Do roku 2012 byly pak podle tohoto webu všechny stávající popisy převedeny do XML a nově popisy již vytvářeny v tomto formátu, s tím, že v roce 2020 databáze obsahovala přes 470 záznamů. Dále podle webu je produkce popisů nyní řízena v rámci oddělení knihovní služby Kongresové knihovny (*Library Services unit of the Library of Congress*) (loc.gov).

OpenDocument

OpenDocument je podle popisu na webu (loc.gov) definován standardem ISO/IEC 26300 a je založen na OpenOffice.org XML, kde OpenOffice.org XML (OOo) je otevřený formát souboru, který podle uvedené historie v letech 2000-2002 vyvíjela společnost Sun Microsystems, a to jako otevřený komunitní projekt (*open community effort*). Vývoj OpenOffice.org byl ukončen, a podle webu projektu (libreoffice.org) bylo po odmítnutí žádosti o darování značky OpenOffice.org projektu a po prozkoumání databází ochranných známek, průzkumu v sociálních médií a po ověření použití pro adresy URL v různých zemích vybrán název "LibreOffice" (libreoffice.org). Srovnání z webu projektu včetně odkazu je uvedeno na obrázku 3.

| | LibreOffice The Document Foundation | Apache OpenOffice™ |
|---|---|---|
| **Latest major release** | **7.3 – February 2022** | 4.1 – April 2014 |
| **Minor releases since 2014** | **98** | 11 |
| **Microsoft OOXML export** | **Yes (.docx, .xlsx, .pptx)** | No |
| **Security: signing of OOXML + PDF files** | **Yes** | No |
| **Cloud/online version** | **Yes – LibreOffice Online, and products based on it** | No |
| **Broad enterprise contribution** | **Yes – Collabora, CIB, allotropia, Red Hat, SIL…** | No |
| **Source code updates in 2021** | **13,015** | 270 |




Obrázek 7 Srovnání LibreOffice a OpenOffice, zdroj: (*libreoffice.org)*

Formát OpenDocument je podle (oasis-open.org) od května 2006 standardizován Mezinárodní organizací pro normalizaci jako standard ISO/IEC 26300 a rozšířený formát OpenDocument 1.2 je standardem od července 2015 jako standard ISO/IEC 26300-1/2015 až ISO/IEC 26300-3/2015, v současnosti aktuální verze je podle webu 1.3 (oasis-open.org).

Sada kancelářských nástrojů LibreOffice obecně představuje FOSS[14] kancelářský balík projektu nadace *The Document Foundation* šířený jako svobodný software pod licencí Apache, a jako jeden z přínosů OpenOffice se obecně považuje právě zavedení mezinárodního standardu ODF[15]. LibreOffice Technology je podle (cs.libreoffice.org)

[14] *Free and Open Source Software*, Svobodný a otevřený software
[15] *Open Document Format*, Otevřený formát dokumentů

výsledkem mnohaleté činnosti v oblasti FOSS, kterou koordinuje *Engineering Steering Committee* (Řídící výbor pro inženýrství), a cílem které bylo podle webu projektu vytvořit jednotnou softwarovou platformu (desktop, mobil i cloud), umožnit transparentní sdílení obsahu a nezávislost na jednotlivých komerčních dodavatelích a strategiích vendor lock-in[16].

OOXML (Office Open XML, ISO/IEC 29500, ECMA 376)

Podle webu (loc.gov) je Office Open XML (OOXML) specifikací pro souborový formát na ukládání dokumentů kancelářských balíků jako textových dokumentů, tabulek či prezentací a formát navržený společností Microsoft byl poprvé použit v **Microsoft Office 2007**, kdy se produkty této sady zavedly nové formáty souborů pod standardem označeným ECMA-376. Tato skupinu formátů dále podle tohoto webu byla navržena tak, aby odpovídala funkčnosti proprietárních binárních formátů, používaných jako výchozí formáty v aplikacích Microsoft Office (Word, Excel a PowerPoint) do verze Office 2003, a aby byla plně kompatibilní se stávajícím korpusem dokumentů (loc.gov).

Formát Office Open XML je v podstatě souborem ZIP obsahujícím XML a další potřebné soubory, a soubory jsou menší, než původní binární - vytvářené předchozími verzemi Microsoft Office. Podle webu investigace.cz je ve státní správě odhadem asi 95 procent úřadů závislých na licencích od Microsoftu[17], a to od operačních systémů Windows, přes serverová řešení a lze předpokládat obdobnou situaci i u kancelářského SW. Vedle toho např. v kategorii tabulkových procesů má Microsoft Excel dlouhodobě dominantní postavení na trhu[18], z toho důvodu je hlavní pozornost v práci věnována této aplikaci a jejímu rozhraní, s předpokladem, že tabulkové procesory ostatních kancelářských balíků mají UI obdobné. V praktické části je proto pozornost věnována především aplikacím kancelářského balíku MS Office, kde je formát Open XML (.docx/.xlsx/.pptx) výchozí formát ve všech podporovaných verzích Microsoft Office.

V dalších podkapitolách jsou ve stručnosti představeny jednotlivé oblasti členěné z funkčního hlediska typu kancelářského SW, v rámci kterých byly jednotlivé knihovny porovnávány. Jak bylo řečeno, vzhledem k velkému množství možných porovnávaných knihoven (a povaze metody AHP) bylo téma zúženo na 5 oblastí a 5 úkonů,

[16] vendor lock-in, proprietární uzamčení závislost na službách/produktech konkrétního dodavatele
[17] https://www.investigace.cz/je-ceska-statni-sprava-rukojmim-microsoftu/
[18] A to od verze 5 z roku 1993 podle všeobecně přejímané informace, na wikipedii a dalších zdrojích

vyhodnocených jako relevantních a autor se zaměřil především na kancelářský balík MS Office.

#### 3.1.3.2 Tabulkové procesory (spreadsheet)

Obecně se jedná o software zpracovávající informace zapsané na virtuálním listu a standardní součást kancelářského balíku s multioborovým uplatněním, obecně například při plánování, evidenci nebo zpracování dat, kdy jsou data organizovány do tabulek, které tvoří sloupky a řádky. Software tohoto typu umožňuje například i čtení souborů ve formátu CSV.

Relevantní aplikace a přípony

- Calc - tabulkový procesor LibreOffice (ods)
- Excel - tabulkový procesor Office (xls,xlsx)
- Numbers - tabulkový procesor iWork (numbers)
- Gsheet - tabulkový proces Tabulky Google (gsheet)

Stručný popis UI

Specifikem je obecně názvosloví jednotlivých objektů, respektive objektové struktury této aplikace, klíčovými pojmy jsou sešit, list, buňka. Práce uživatele je prováděna na pracovní ploše, pomocí jednotlivých karet s nástroji, pracovní plochu aplikace tvoří množina buněk, ty jsou organizovány do sloupců a řádků. Mezi specifické úkony pro kategorii, včetně dalších možných úkonů pro typ kancelářského SW, které nakonec hodnoceny, patří například:

- Práce s grafy (obecně vizualizace)
- Práce s komentáři
- Práce se styly
- Další vlastnosti pracovního listu
- Podmíněné formátování
- Kontingenční tabulky

Co se týče možností automatizace, pokročilejší znalost funkcí umožňuje efektivnější práci, automatizaci výpočtů obecně umožňuje implementace aplikace VBA (*Visual Basic for Applications*) a podle webu (pyxll.com) jde vše, co lze řešit skrz VBA[19] také provést v

[19] respektive *Excel Object Model*

jazyce Python. Vedle toho je obrovská podpora knihoven specificky pro tuto oblast. Níže je uveden příklad možných alternativních nástrojů pro účely automatizace:

- Makra a UDF (*User Defined Function*)
- Power Query a Power Pivot
- Ovládací prvky ActiveX a COM[20] (Component Object Model)

Vzhledem k tomu, že jsou soubory xlsx souborem ZIP s XML (*SpreadsheetML*), je v rámci knihoven Python dále možná například manipulace prostřednictvím knihovny xml.

#### 3.1.3.3 Textové procesory

Obecně software k vytváření formátovaného textu, obvykle s přístupem k editaci metodou WYSIWYG („*What you see is what you get*“, „co vidíš, to dostaneš“), standardní součást kancelářského balíku.

Relevantní aplikace a přípony

- Writer - textový procesor LibreOffice (odt)
- Word - textový procesor Office (doc, docx)
- Pages - textový procesor iWork (pages)
- Gsheet - textový proces Dokumenty Google (docx)

Stručný popis UI

Práce uživatele je prováděna na pracovní ploše, pomocí jednotlivých karet s nástroji.
Mezi specifické úkony pro kategorii, včetně dalších možných úkonů pro typ kancelářského SW, které nakonec hodnoceny, patří například:

- Přidání odstavce
- Přidání nadpisu
- Přidání zlomu stránky (*pagebreak*)
- Přidání tabulky
- Přidání obrázku
- Použití stylu odstavce
- Přidání čísel stránek.

Co se týče možností automatizace, vzhledem k příbuznému formátu OOXML a přítomnosti VBA, v zásadě je situace obdobná jako u tabulkových procesorů. Stejně tak

[20] Jedná se o technologii programových komponent od firmy Microsoft

podobně jako u souboru přípony XLSX je DOCX soubor je ZIP s XML (*WordProcessingML*) soubory uvnitř.

#### 3.1.3.4 Portable Document Format

Stručný popis UI

Obecně je souborový formát PDF jedním z nejpoužívanějších formátů na světě, určen pro ukládání dokumentů - nezávisle na SW i HW na kterém byly pořízeny a jedná se o kombinaci textu a vektorové a bitmapové grafiky. Podle (adobe.com) je PDF formát používaný k prezentaci a bezpečnému zasílání dokumentů, který je nezávislý na softwaru, hardwaru i operačním systému. Podle abstraktu u nejnovějšího standardu *ISO 32000-2:2020 Document management — Portable document format — Part 2: PDF 2.0* specifikujícího digitální formu, je formát PDF určen pro reprezentaci elektronických dokumentů, která uživatelům umožňuje výměnu a prohlížení elektronických dokumentů nezávisle na prostředí, ve kterém byly vytvořeny, nebo na prostředí, ve kterém jsou prohlíženy nebo tištěny (iso.org). Podle webu (loc.gov) je PDF formát vyvinutý společností Adobe Systems Incorporated, který je společností Adobe popisován jako *general document representation language*[21] a představuje formátované, stránkově orientované dokumenty, které mohou být strukturované nebo jednoduché, mohou obsahovat text, obrázky, grafiku a další multimediální obsah, jako například video a zvuk. Podporovány jsou anotace, metadata, hypertextové odkazy a záložky. PDF byl proprietární formát Adobe, do 1. července 2008, kdy byl vydán jako otevřený standard a publikován Mezinárodní organizací pro normalizaci jako ISO 32000-1:2008 (loc.gov).

Mezi specifické úkony pro kategorii, včetně dalších možných úkonů pro typ kancelářského SW, které nakonec hodnoceny nebyly, patří například:

- extrakce formátu specifických informací o dokumentu
- extrakce tabulek, obrázků a multimediálního obsahu
- rozdělení a sloučení dokumentů po stránkách
- sloučení více stránek do jedné stránky
- Ořezávání a otáčení stránek
- šifrování a dešifrování souborů PDF

[21] obecný jazyk pro reprezentaci dokumentů

Co se týče možností automatizace, podle (adobe.com) zahrnuje *PDF Services API* přístup k operacím PDF *Extract API* a *Document Generation API* (adobe.com). Vedle toho existuje řada nástrojů umožňující generování a konverzi, za všechny například *pandoc* a výstup ve formátu PDF umožňuje řada aplikací.

#### 3.1.3.5 Další produkty MS Office

**Nástroje na tvorbu prezentací (MS Powerpoint)**

Obecně software, který slouží k zobrazení informací ve formě prezentace.

Relevantní aplikace a přípony

- Keynote - prezentační nástroj iWork (key)
- Impress - prezentační nástroj LibreOffice (odp)
- Google Slides - prezentační nástroj Prezentace Google (gslides)
- Powerpoint - prezentační nástroj Office (ppt, pptx)

Stručný popis UI

Práce uživatele je prováděna na pracovní ploše, pomocí jednotlivých karet s nástroji.
Mezi specifické úkony pro kategorii - další možné zvažované úkony specifické pro oblast, které nakonec nebyly hodnoceny, patří například práce s obrázky a grafy, jejich otáčení a ořezávání, úpravy odsazení v textu odstavce a tak podobně.
Co se týče možností automatizace, vzhledem k formátu OOXML (*PresentationML*) srovnatelné jako u předchozích souborů.

**Elektronická pošta (Outlook)**

Klient ke správě elektronické pošty, tudíž se nejedná přímo o aplikaci, která je součástí kancelářského balíku, ale bylo zařazeno do práce - vzhledem k tomu že se jedná o jeden z hlavních komunikačních nástrojů. Relevantními příponami mohou být msg, pst a eml, příp. ještě mbox.

Práce s formáty elektronické pošty z hlediska metadat je podle rešerše důležitá především pro oblast digitální forenziky a eDiscovery[22]. Všechny poštovní soubory podle (2016, legalnews.com) bez ohledu na formát obsahují metadata související se zprávou, a to například:

[22] eDiscovery je podle (2022, Deloitte) proces identifikace, sbírání a správy elektronicky ukládaných informací, obzvláště v případech, kdy je potřeba přísný přístup k zajištění jejich úplnosti a správnosti a tento proces umožňuje uchovávat data z různých zdrojů (mobilních telefonů, osobních počítačů, firemních serverů, fyzických kopií) na jedné platformě. (2022, Deloitte)

- Od, CC (*carbon copy*, kopie), BCC (*blind carbon copy*, skrytá kopie)
- Předmět, Datum/Čas odeslání,
- Přílohy, a další (legalnews.com)

Aplikace Microsoft Outlook používá k ukládání e-mailů (a schůzek nebo úkolů) proprietární datový formát PST (*Personal Storage Table*).

Relevantní aplikace a přípony

Stručný přehled formátů je uveden v následujícím výčtu, níže je uvedena základní charakteristika.

- EML
- MSG
- PST, OST

EML je podle (2022, loc.gov) přípona souboru pro e-mailovou zprávu uloženou do souboru ve standardním formátu MIME[23] RFC 822 pomocí aplikace Microsoft Outlook a v dalších e-mailových programech. Vzhledem k tomu, že jsou podle tohoto webu soubory EML vytvořeny v souladu s průmyslovým standardem RFC 5322, lze je používat s většinou e-mailových klientů, serverů a aplikací (2022, loc.gov). Tyto soubory dále podle webu obvykle ukládají každou zprávu jako jeden soubor (na rozdíl od MBOX, který spojuje všechny zprávy ze složky do jednoho souboru) a přílohy mohou být buď zahrnuty jako obsah MIME ve zprávě, nebo vypsány jako samostatný soubor, na který se v souboru EML odkazuje (2022, loc.gov).

Soubor osobních složek nebo PST je podle (loc.gov) otevřený proprietární formát datového souboru, který se používá k ukládání místních kopií zpráv, událostí kalendáře a dalších položek v rámci softwaru Microsoft včetně aplikace Microsoft Office Outlook. Soubory PST se používají k ukládání archivovaných položek a k udržování jejich offline dostupnosti, a sdílí PFF (*Personal Folders File format*) strukturu jako OST (*Offline Storage Table*) a PAB (*Personal Address Book*). Další formáty nebyly v práci adresovány.

Co se týče možností automatizace, stejně jako u PDF nástrojů spolupráce s elektronickou poštou existuje řada. Co se týče specifikace formátu PST, ty byly podle webu (github.com) zveřejněny až v roce 2010, což je jedna z příčin, proč je k dispozici relativně nízké množství nástrojů pro manipulaci s ním, včetně knihoven jazyka Python, jak se uvádí v odkazech dokumentace ke knihovně pypff/libpff (github.com).

[23] Multipurpose Internet Mail Extensions („víceúčelová rozšíření internetové pošty“)

#### 3.1.3.6 Soudobé trendy a nástroje RPA

V této podkapitole jsou ve stručnosti představeny soudobé trendy týkající se tématu automatizace, a dále pak v současnosti aktuální „*buzzword*“ termíny spojené s procesem digitalizace, RPA a tak podobně. Účelem je poskytnout alespoň základní vhled do problematiky a specifik této oblasti IT.

Průmysl 4.0.

Termín odkazuje ke konceptu Čtvrtá průmyslová revoluce, je spojen se souslovím "*smart factory*" a jedná se mj. o označení trendu digitalizace a s ním souvisejících změn. Termín pochází z roku 2011 z projektu v rámci strategie německé vlády v oblasti high-tech a za původce je označován německý profesor Wolfgang Wahlster. Je spojován s rozvojem a šířením technologií jako jsou umělá inteligence, robotika, nanotechnologie, internet věcí, blockchain, autonomní vozy a další a řada států, včetně ČR, má koncepci, jak podpořit Průmysl 4.0 a jak čelit důsledkům, které přinese (deloitte.com).

Robotizace standardizovaných kancelářských procesů (RPA)

Obecně se u RPA[24] jedná o implementaci automatizací napodobujících lidskou činnost, interakce a rozhodnutí s využitím robotických a kognitivních technologií, jež zlepšují produktivitu pracovní síly a efektivitu procesů a obecně asi s největší perspektivou v oblasti RPA jsou open-source nástroje Selenium a Robot framework:

- Robot framework je podle webu (quora.com) je obecný framework pro automatizaci testů, nezávislý na operačním systému a aplikaci a se snadno použitelnou syntaxi tabulkových testovacích dat a používaným přístupu k testování založený na klíčových slovech (*keyword-driven testing approach*), s tím, že testovací knihovny mohou rozšiřovat své možnosti testů prováděných buď v jazyce Python, nebo v jazyce Java, a uživatelé mohou vytvářet nová klíčová slova vyšší úrovně z již existujících klíčových slov pomocí stejné syntaxe, která se používá k vytváření testovacích případů (quora.com).
- Selenium je podle (selenium.dev) zastřešující projekt pro řadu nástrojů a knihoven umožňujících a podporujících automatizaci webových prohlížečů a podle webu poskytuje rozšíření pro emulaci interakce uživatele s prohlížeči, distribuční server pro škálování přidělení prohlížečů a infrastrukturu pro implementaci specifikace W3C WebDriver, která umožňuje psát zaměnitelný kód pro všechny hlavní webové

[24] *Robotic process automation*, Robotická automatizace procesů

prohlížeče. Jádrem Selenia je WebDriver, rozhraní pro zápis sad instrukcí, spustitelné v mnoha prohlížečích (selenium.dev).

Robotic Desktop Automation

Podle webu (quora.com) se jedná o podmnožinu RPA označovanou taky jako automatizace s účastí nebo RPA s účastí (*attended*), a obecně se jedná o bota nebo virtuálního asistenta - v počítači zaměstnance nebo koncového uživatele, s tím, že serverový robot RPA pracuje bez zásahu člověka, a proto se mu někdy také říká "neobsluhovaný robot", naproti tomu RDA bot existuje proto, aby pomáhal zefektivnit podnikové procesy, kde je nutný lidský zásah.

Business Process Management

Podle webu (automationanywhere.com) je BPM (*Business Process Management*, Správa obchodních procesů[25]) technologie a praxe optimalizace podnikových procesů a pracovních postupů tak, aby byly efektivnější a přizpůsobivější. Za tímto účelem BPM obvykle zahrnuje automatizaci podnikových procesů (BPA). Podle webu (redhat.com) je BPA použití softwaru k automatizaci opakovatelných, vícekrokových obchodních transakcí. Na rozdíl od jiných typů automatizace bývají řešení BPA komplexní, napojená na více podnikových systémů informačních technologií (IT) a přizpůsobená konkrétním potřebám organizace a je běžné, že organizace uplatňují BPA jako součást strategie digitální transformace, aby zefektivnily své pracovní postupy a fungovaly efektivněji.

Inteligentní automatizace

Inteligentní automatizace [26] je podle webu (*lepsi-reseni.cz*) kombinací technologií robotické automatizace procesů (RPA) a umělé inteligence, jenž společně umožňují rychlou komplexní automatizaci podnikových procesů a umožňují urychlení digitální transformace. Přidruženým termínem je inteligentní automatizace procesů (IPA, *Intelligent process Automation*) která kromě RPA a algoritmů strojového učení obsahuje také software pro řízení procesů, zpracování a tvorbu přirozeného jazyka a kognitivní agenty nebo boty a přináší lepší efektivitu o 20-35%, zkrácení doby zpracování o 50-60% a několikanásobnou návratnost investic, avšak toto řešení je stále v začátcích a zatím neexistují žádné případy použití úplného komplexního automatizovaného procesu (*lepsi-reseni.cz*).

[25] nověji označované jako DPA (*Digital Process Automation*, digitální automatizace procesů)
[26] IA, *Intelligent* automation

Hyperautomatizace

Hyperautomatizace je podle webu (*cfoworld.cz*) automatizace všeho, co lze v organizaci automatizovat, a jejím záměrem je zefektivnit procesy v organizaci pomocí inteligentní automatizace procesů (*IPA*), která zahrnuje umělou inteligenci, RPA a další technologie, aby mohly probíhat bez zásahu člověka. Často se využívá další technologie - například optické rozpoznávání znaků (*OCR*), inteligentní zpracování dokumentů (*IDP*) a zpracování přirozeného jazyka (*NLP*) - k zajištění vyšší kvality automatizace s využitím dat z různých zdrojů. Digitální dvojčata nebo organizace digitálních dvojčat (*DTO*, *Digital twin of an organization*) se často používají k modelování. s cílem zlepšit provoz a vyhodnotit dopad automatizace, a to vizualizací vztahů mezi funkcemi, procesy a klíčovými výkonnostními ukazateli (*cfoworld.cz*).

## 3.2 Knihovny jazyka Python a stanovení použitých kritérií pro hodnocení

Tato podkapitola obsahuje stručný popis použitých modulů, knihoven a balíčků v rámci kategorií SW. Dále jsou popsány normy, z kterých se částečně vychází při stanovení kritérií.

### 3.2.1 Možnosti pythonu pro automatizaci a popis knihoven

Pro přehled jsou v tabulce níže uvedeny úkony, na jejichž porovnávání – respektive schopnosti, možnosti a výkon jednotlivých knihoven v porovnávaných oblastech se práce zaměřuje.

| *Číslo* | Popis úkonu/úkolu/zadání |
|---|---|
| *1* | Tvorba dokumentu a hromadné vytváření (*Bulk/batch creation*) |
| *2* | Extrakce obsahu (*Data/Content extraction*) |
| *3* | Manipulace s obsahem nativně (*Data update/modify*) |
| *4* | Sloučení a/nebo rozpojení souboru (*File merge and/or split*) |
| *5* | Čtení a úprava metadat a vlastností souboru (*Metadata fields and Custom Properties*) |
| *6* | Vyplnění formulářů a práce s templates |
| *7* | Práce s templates |
| *8* | Integrace s dalšími knihovnami |
| *9* | Podpora všech možných formátů dat |
| *10* | Další možné zvažované úkony specifické pro oblast |

Tabulka 2 Hodnocené a měřené úkony v praktické části práce, zdroj: vlastní

**V tabulce v příloze A jsou uvedeny vybrané oblasti a zvažované knihovny jazyka Python, tučně vyznačené jsou ty, které budou porovnávané.**

**Použité a porovnávané knihovny**

- Knihoven jazyka Python pro spolupráci s formáty tabulkových procesorů je velké množství, pro MS Excel a manipulaci s formátem XLSX byla vybrána knihovna **openpyxl**. Knihovnu, podle dokumentace k ní, vyvinuli Eric Gazoni a Charlie Clark. Knihovna umožňuje číst, zapisovat a dalšími způsoby manipulovat se soubory aplikace Excel v jazyce Python. Na stránkách knihovny je doporučení, že ohledně bezpečnosti knihovna ve výchozím nastavení nechrání před injektáží XML, k ochraně se doporučuje instalace knihovny defusedxml (openpyxl-doc).
- Nabídka knihoven jazyka Python pro manipulaci s dalšími aplikacemi MS Office[27] je relativně malá, co se týče DOCX je k dispozici jedině **Python-docx**.
- **PyPDF2** je modul jazyka Python pro extrakci specifických informací z dokumentů, slučování souborů PDF, oddělování stránek PDF, přidávání vodoznaků do souborů, šifrování a dešifrování souborů PDF a tak podobně (PyPDF2).
- **Python-pptx** je podle (python-pptx.readthedocs.io) knihovna jazyka Python pro vytváření a aktualizaci souborů PowerPoint (.pptx). Je závislá na balíčcích lxml a Pillow, a knihovně Python Imaging Library (PIL). Funkce pro tvorbu grafů závisí na nástroji XlsxWriter (python-pptx.readthedocs.io).
- Jako nástroj pro manipulaci s formáty elektronické pošty byla vybrána knihovna **libpff (pypff)**, jenž slouží jako podpůrný nástroj pro analýzu a zpracování zdrojů klienta el. pošty MS Outlook formátu PFF a OFF (github.com).

**Používané built-in moduly**

- Email je součástí standardní knihovny Pythonu a je součástí instalace Pythonu.
- Operating system (os) je součástí standardní knihovny Pythonu a provádí ad-hoc příkazy na základním operačním systému.
- Modul shutil je součástí standardní knihovny Pythonu a nabízí řadu operací se soubory a kolekcemi souborů.
- Modul glob slouží k načtení souborů/názvů, které odpovídají zadanému vzoru.

**Další zvažované knihovny**

- Knihovna Mail-parser je wrapper pro standardní knihovnu email a umí podle (pypi.org) analyzovat e-mail ve formátu Outlook (.msg) a poskytuje snadný

[27] a při nepoužití Win32COM, xml a dalších knihoven

způsob, jak přejít z nezpracovaného (tzv. *raw*) formátu na objekt Pythonu, který je možné použít v kódu. Jedná se o modul SpamScope (pypi.org).

- Knihovna xlrd je knihovna pro čtení dat a formátování informací ze souborů Excel v historickém formátu XLS.
- Balíček pandas podle (github.com) poskytuje rychlé, flexibilní a expresivní datové struktury, navržené tak, aby práce s "relačními" nebo "označenými" daty byla snadná a intuitivní, klíčový pro něj je operace s tzv. DataFrames (github.com).
- Knihovna Reportlab je vhodná pro generování PDF a grafiky, včetně grafů (reportlab.com).
- Balíček extract_msg automatizuje extrakci klíčových údajů e-mailu (formátu MSG) a extrakci jeho příloh.

### 3.2.2 Hodnocení kvality softwaru

Pro účely práce je nutné definovat pojmy s nimiž se v práci operuje.

V rámci této části je čerpáno z prezentace "Mezinárodní normalizace kvality softwaru" dostupné online od (Vaníček). Podle autora je kvalita podle základní normy ISO 9000 (Systémy managementu kvality, základní principy a slovník) definována jako stupeň splnění požadavků souborem inherentních charakteristik (znaků), a je třeba ji hodnotit u procesu i u produktu, s tím, že proces je definován jako soubor vzájemně souvisejících a vzájemně působících činností, který přeměňuje vstupy na výstupy a produkt je výsledek procesu. Požadavek je definován (ISO 9000) jako potřeba nebo očekávání, které je stanoveno speciálně, nebo se obecně předpokládá, nebo je závazné (vyplývá například z právních předpisů). Produkty mohou být podle autora těchto čtyř kategorií:

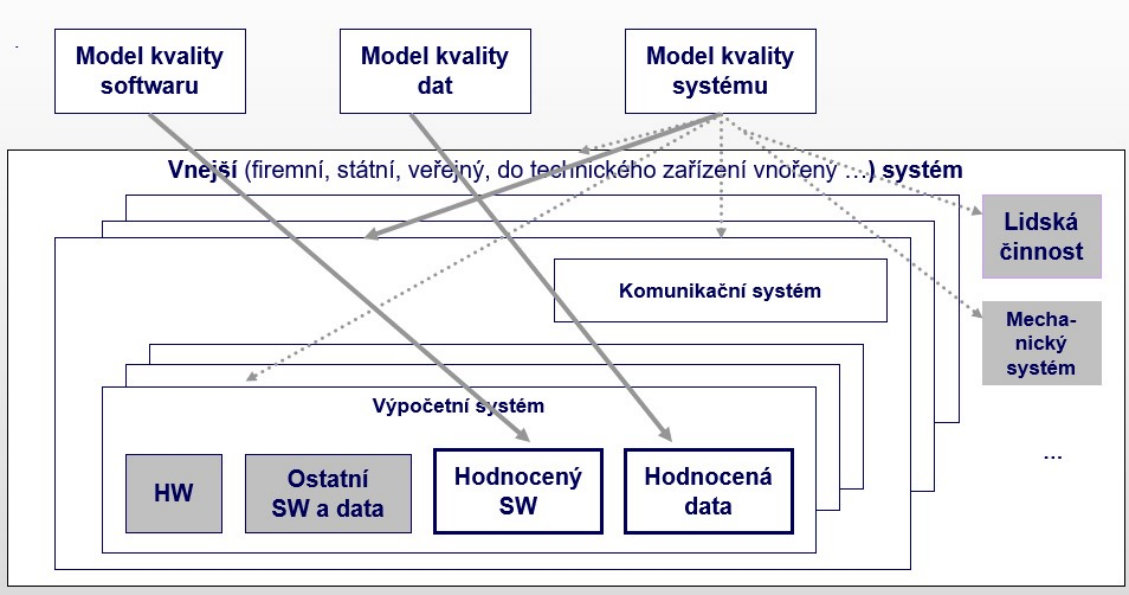
- Služby (například přeprava)
- Software (nejen počítačové programy, ale třeba slovník či „*know how*“)
- Hardware (například mechanická část nějakého stroje, třeba motoru)
- Zpracované materiály (například olej do motoru)
- Konkrétní produkt může být kombinací těchto kategorií (Vaníček)

Informatický produkt bývá podle (Vaníček) téměř vždy kombinací softwaru a služeb, případně i hardwaru. Pohled na kvalitu procesu je důležitý zejména pro vývojáře a výrobce a dodavatele, vedle toho pohled na kvalitu produktu je důležitý zejména pro opatřovatele, zákazníka a uživatele. Požadavky se výrazně odlišují podle typu produktů a existují normy

podle oborů, s tím, že pro IT existují normy ISO/IEC. Dále podle autora je, vzhledem k globálnímu trhu v informatice, potřeba pohledy na kvalitu sjednotit, a tudíž je nezbytná normalizace na co nejširší úrovni. Kvalitu je třeba hodnotit objektivně, k tomu je potřeba měření jako popisu výseku reálného světa čísly. Autor dále uvádí specifika problematiky, tři pohledy na kvalitu – vnější, vnitřní a kvalitu užití, a dále modely kvality SW, dat a systému a vztah charakteristik, podcharakteristik a atributů, měřících funkcí a prvků pro měření kvality (Vaníček). V následující podkapitole bude uvedeno podrobněji, včetně aktuálních a v rámci práce použitých norem ISO/IEC.

#### 3.2.2.1 Kvalita softwaru

V rámci této podkapitoly je čerpáno především z příspěvku "*Kvalita informačních systémů z pohledu mezinárodní normalizace*" od (Vaníček, 2007). Vztah hodnocení SW, jednotlivých modelů kvalit a výpočetních a dalších systémů podle autora je pro ilustrativní účely do problematiky uveden na obrázku níže.




Obrázek 8 Vztah hodnocení SW a kvalit, a dalších komponent, zdroj: (Vaníček, 2007)

Co se týče pojmů kvalita a jakost, užívají se v totožném významu, jelikož se podle (Vaníček, 2007) oba pojmy vyskytují i v problematice kvality SW zaměnitelně. Podle autora je **kvalita** definována jako stupeň uspokojení daných nebo stanovených požadavků zainteresovaných stran souborem vnitřních vlastností posuzované entity, s tím, že požadavky musí vycházet z reálných potřeb zúčastněných stran.

Podle autora se mezinárodní normalizace snaží standardizovat jak pohledy na kvalitu **procesu**, v kterém nějaký produkt vzniká a v kterém se používá, tak i pohledy na

kvality **produktu** jako takového, s tím, že první tento pohled (dynamický) je spíše pohledem vývojáře a výrobce a našel svůj odraz v řadě norem řady ISO 9000, a druhý (statický) pohled na kvalitu produktu je typicky pohledem zákazníka, či opatřovatele produktu, a v případě informatických produktů je typicky pohledem systémového integrátora.

Dále podle autora nevychází statický pohled na kvalitu produktu z informací o tom, za jakých okolností produkt vznikl, ale pouze z vlastností produktu jako takového, přičemž posuzovaným produktem může být jak hotový výrobek, tak poskytovaná služba, tak i jejich kombinace, a podle autora je u informatických produktů právě typické, že produkt tvoří systém jako hotový artefakt spolu s poskytovanou službou (Vaníček, 2007).

Dále uváděný problém normalizace v oblasti kvality produktů jako takových, je podle autora skutečnost, že normy pro jednotlivé typy produktů se od sebe musí podstatně lišit a nelze nalézt tolik společných požadavků na kvalitu produktů z různých oblastí, jak tomu je u zásad, jak kvalitní produkty vytvářet, což dále podle autora znemožňuje záměr, jak vytvořit obecné normy analogické normám populární řady ISO 9000 (Vaníček, 2007).

Řada norem ISO/IEC s vyhrazeným rozpětím číslování 25000 – 25099 byla podle (Vaníček, 2007) navržena s cílem odstranit vady stávajících norem pro jakost informatických projektů, podle autora vzniklých víceméně náhodně v posledním desetiletí minulého století, a projekt vývoje těchto norem dostal název SQuaRE, vzniklý jako zkratka z anglického názvu „Software Quality Requirements and Evaluation“. Struktura řady je na následujícím obrázku.

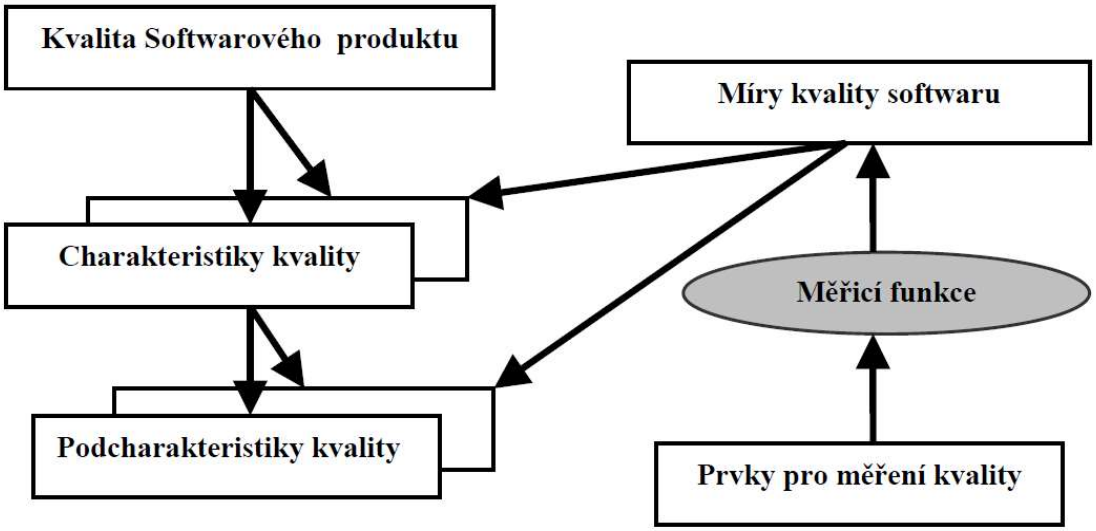

Obrázek 9 Rodina standardů ISO SQuaRE – struktura řady podle zdroj: (2019, Torchiano)

Podle (Papík, 2013) obsahuje střední oddíl struktury na obrázku dokumenty označené 2500n - zastřešující celou soustavu, a jednotlivé strany čtverce pak představují věcně provázané skupiny norem věnovaných následujícím aspektům problému hodnocení kvality:

- 2501n – Model jakosti.
- 2502n – Míry jakosti.
- 2503n – Požadavky na jakost.
- 2504n – Postupy při hodnocení jakosti.
- 25050 - 25099 - Rezervována pro příp. další doplňující dokumenty (Papík, 2013).

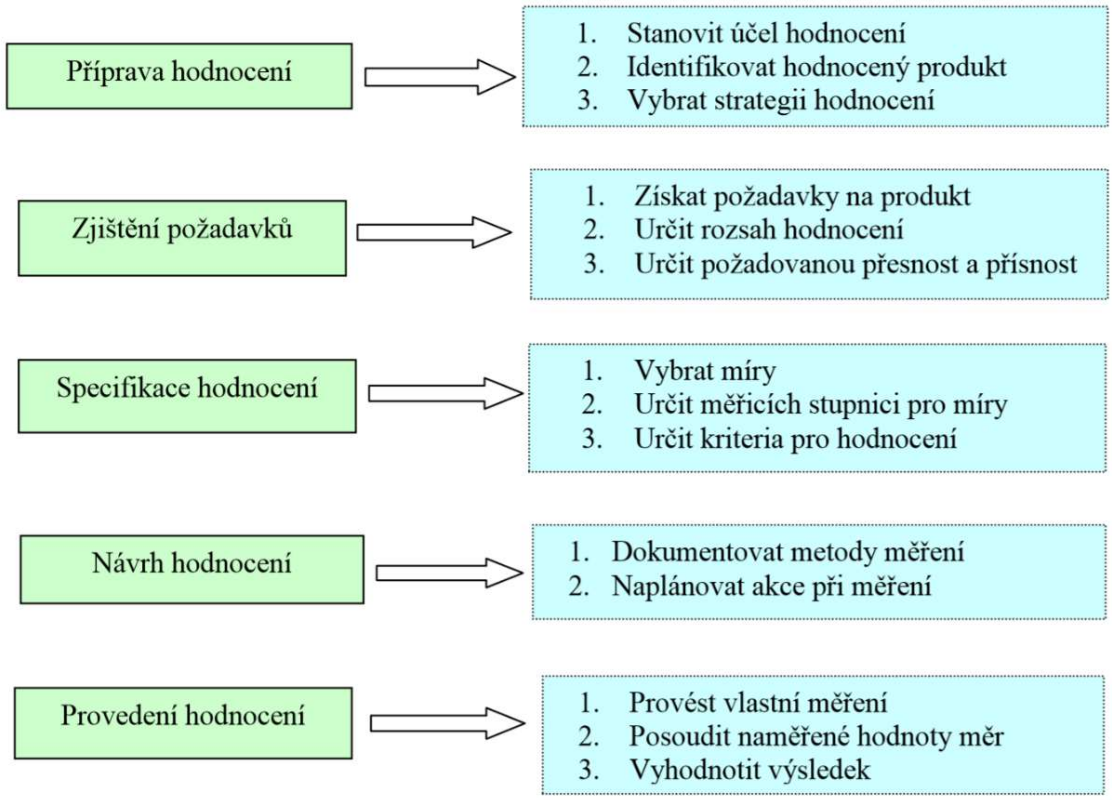
Norma *ISO/IEC 25000:2005 Průvodce SQuaRE* představuje zastřešující dokument, který popisuje filozofii celé soustavy norem a uvádí vztah řady 250nn k předchozím řadám norem pro jakost softwarového produktu (především k *ISO/IEC 9126* a *ISO/IEC 14598)* a základní užívanou terminologii. Norma ISO/IEC *25001:2007 Plánování a řízení* uvádí, jak identifikovat a analyzovat v organizaci požadavky na kvalitu informačních produktů a jak plánovat a řídit specifikaci těchto požadavků a hodnocení kvality jednotlivých produktů.

Zásadním rozdílem od staré řady je podle autora zavedení prvků pro měření, kdy vlastní míry kvality mají být získávány jako funkce, závislé na tzv. prvcích pro měření, které samy o sobě kvalitu neměří, ale lze je získat přímo z hodnoceného softwaru nebo jeho funkce. Například podle autora z rozsahu softwaru změřeného přímo třeba počtem řádků zdrojového kódu (např. metrika LOC), nebo počtu poruch za dané časové období, lze získat míru (atribut), který vypovídá o bezporuchovosti. Schéma je uvedené na následujícím obrázku.



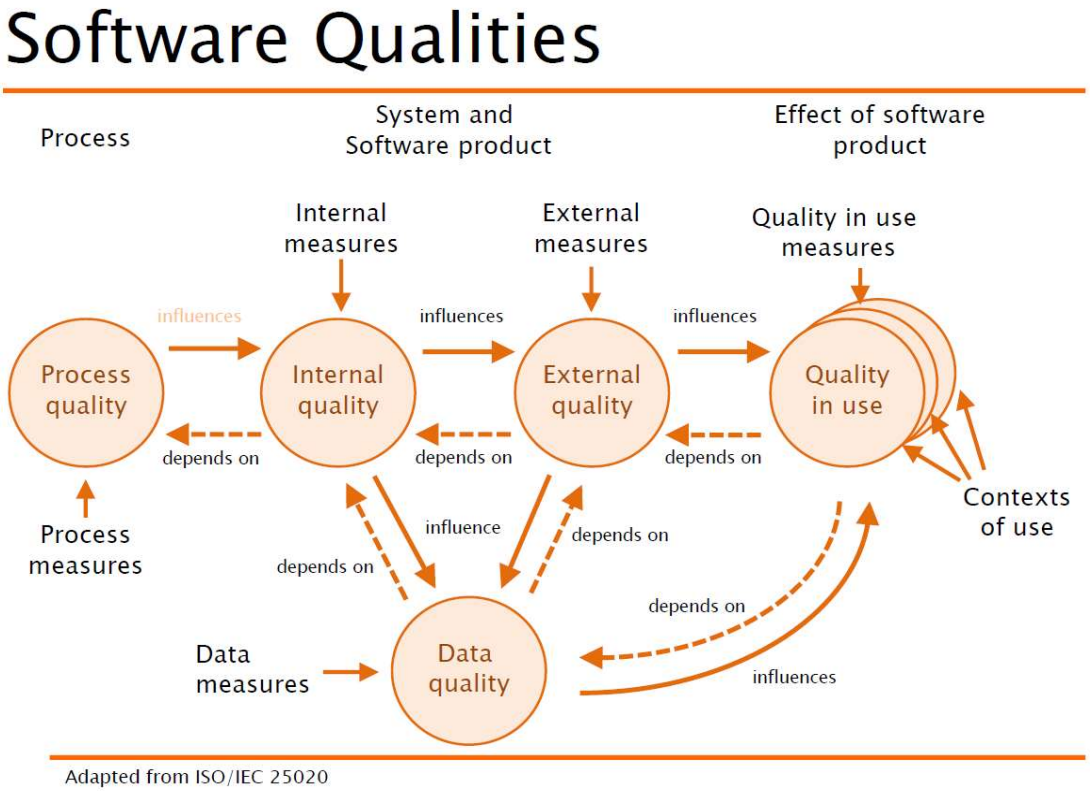

Obrázek 10 Schéma hodnocení kvality softwarového produktu zdroj: (Vaníček, 2007)

Norma dále podle (Papík, 2013) hovoří o „třech kvalitách softwaru“, vnější (*external*), vnitřní (*internal*) a o kvalitě užití (*in use*). Vnější kvalita spočívá v plnění požadavků vyplývajících z potřeb zúčastněných stran při funkci softwaru a lze ji podle autora hodnotit poté, co je software dokončen - nejdříve v etapě testování. Vnitřní kvalita je dále podle autora souhrn interních vlastností softwaru a jeho komponent a polotovarů vznikajících v různých etapách návrhu a tvorby softwaru, o kterých se domníváme, že schopnost plnit požadavky zúčastněných stran ovlivní (Papík, 2013). Etapy při hodnocení jakosti jsou uvedeny na následujícím obrázku.




Obrázek 11 Etapy při hodnocení jakosti podle (2006, Vaníček)

Jednotlivé kvality a jejich vztahy jsou uvedeny na následujícím obrázku.



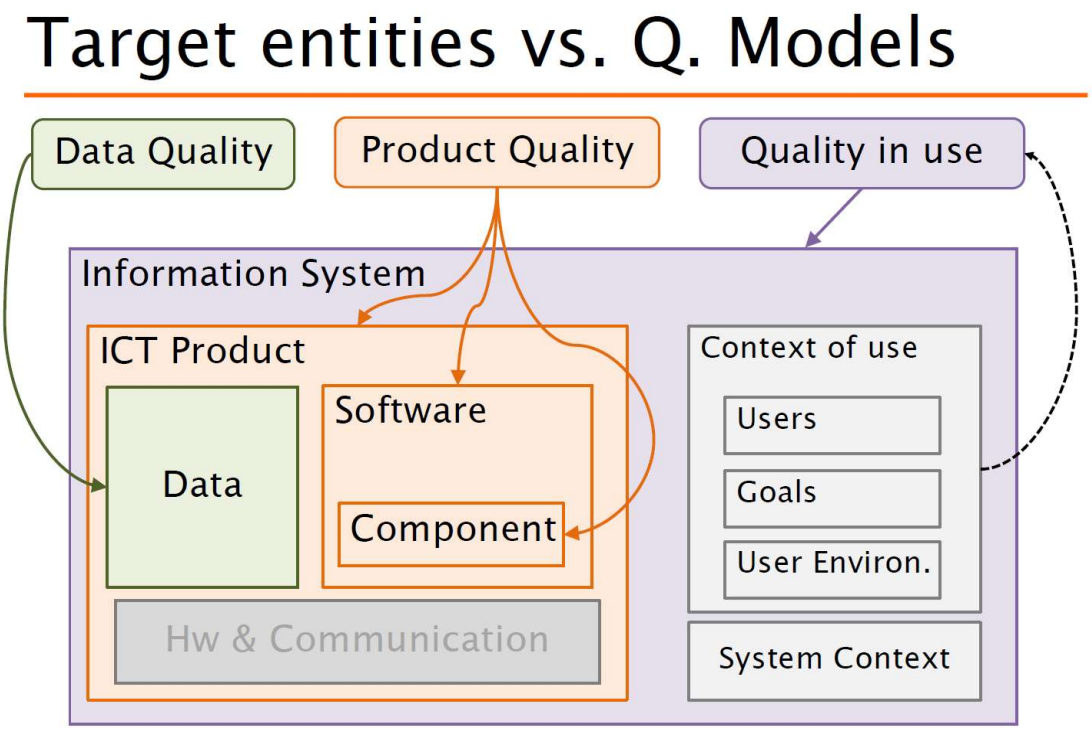

Obrázek 12 Složky SW kvality a jejich vztahy podle (2019, Torchiano)

### 3.2.2.2 ISO/IEC 250xx (SQuaRE)

Řada norem ISO/IEC 250xx je podle (Vaníček, 2007) výsledkem mezinárodního vědeckého normalizačního projektu SQuaRE, který vytváří mezinárodní normalizační organizace ISO spolu s mezinárodní elektromechanickou komisí IEC, a to v gesci společného výboru *ISO/IEC JVC 1 Informační technika*, speciálně jeho podvýboru *SC7 – Systémové a softwarové inženýrství*.

Podle (2006, Vaníček) se jedná o ucelený systém mezinárodních norem pro hodnocení jakosti (= kvality) softwarových produktů, především z hlediska potřeb jejich uživatelů, který nahradil zastaralé standardy roztříštěné do řad ISO/IEC 9126, 12119 a 14598, a rozšířil jejich stávající působnost i na jiné důležité oblasti, jakými je například jakost datové základny (2006, Vaníček). Vztah relevantních cílových entit[28] a modelu kvality je uveden na následujícím obrázku.

[28] Prvek (nebo „část“), která má být měřena.




Obrázek 13 Vztah cílových entit a modelu kvality podle (2019, Torchiano)

Modely kvality

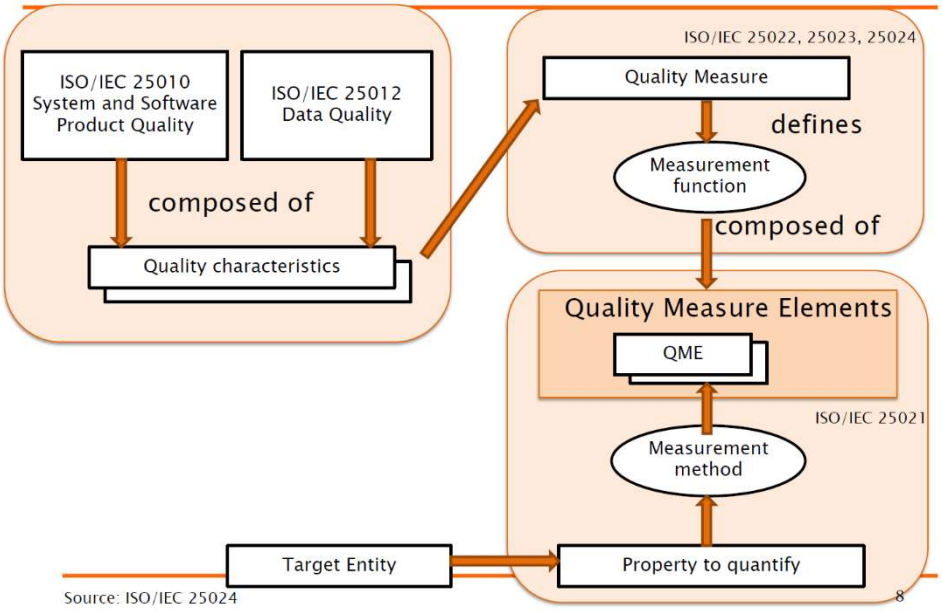
Obecné normy a standardy managementu kvality jsou příliš všeobecné, specifika v oblasti vývoje softwaru nepostihují a týkají se především procesu vývoje, organizačních opatření apod. Cíl, který má projekt SQuaRE potažmo řada standardů SO/IEC 25000, je hodnotit informatické (softwarové) produkty především z hlediska jejich uživatelů, a to je velmi významný pohled. Na obrázku je uveden vztah mezi jednotlivými složkami.

Podle (2013, Papík) se měření dílčích charakteristik a podcharakteristik kvality, provádí přes skupiny, které se nazývají QM (*Quality measures*) a ty se pak dále skládají z jednotlivých elementů - atributů (QME) pro měření kvality, na obrázku je uveden příklad výpočtu QME.

| ID | Name | Description | Measurement Function & QMEs |
|---|---|---|---|
| PRU-G-1 | (Mean) CPU utilization | How much CPU time is used to perform a given set of tasks? | X = B / A<br>A = Maximum CPU time allowed to perform a given set of tasks<br>B = Amount of CPU time actually used to perform the tasks |
| NOTE | Result value varies from 0 to infinite. Usually, the smaller is better. | | |
| PRU-G-2 | (Mean) Memory utilizatio | How much memory space is used to perform a given set of tasks? | X = B / A<br>A = Maximum memory space allowed to perform a given set of tasks<br>B = Amount of memory spaces actually used to perform the tasks |
| NOTE | Result value varies from 0 to infinite. Usually, the smaller is better. | | |

Obrázek 14 Příklad měření QME, zdroj: (2013, Papík)

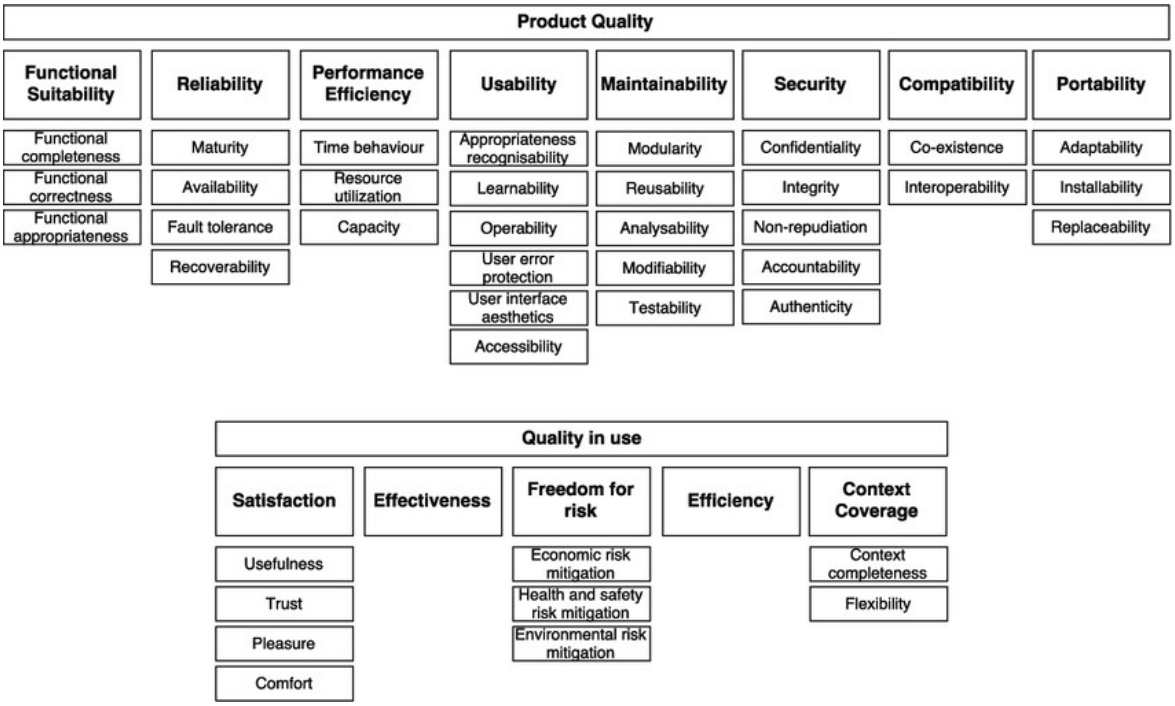
Vztahy mezi důležitými standardy a jednotlivými prvky je uveden na následujícím obrázku.




Obrázek 15 Vztahy mezi standardy a jednotlivými prvky podle zdroj: (2019, Torchiano)

### 3.2.2.3 ISO/IEC 25010

Mezinárodní norma ISO/IEC 25010 podle (iso.org ) definuje charakteristiky modelu kvality pro používání (skládající se z pěti charakteristik, z nichž některé se dále dělí) a model kvality produktu (skládající se z osmi charakteristik), jejich výčet je uveden na následujícím obrázku.




Obrázek 16 Model kvality produktu a kvality užití podle 25010 – přehled charakteristik a podcharakteristik zdroj: (iso.org)

Model kvality produktu je možno použít pro informační systémy i pro softwarové produkty, tudíž i v rámci hodnocení programových knihoven Python, ale je možné, že jsou k v rámci systému SQuaRE některé standardy vhodnější než ty použité – případně že budou doplněny. Podle (2006, Vaníček) se nové normy doplněné do systému SQuaRE objevily v souvislosti se skutečností, že specifické typy softwaru vyžadují specifické způsoby hodnocení kvality, a (například) takovým typem je podle autora konfekční software prodávaný „pultově“ nebo šířený jako „*open source*“. Pro takovéto produkty se užívá název „*Off-The-Shelf*“ software (2006, Vaníček).

# 4 Vlastní práce

Kapitola obsahuje nejprve přípravu hodnocení, následně samotné hodnocení v podobě kvantifikovaného porovnání vybraných knihoven v jednotlivých oblastech automatizace kancelářské činnosti a výstupy z tohoto hodnocení.

## 4.1 Příprava hodnocení

V této podkapitole je uvedena příprava hodnocení, tedy jednotlivé kroky, které samotnému hodnocení předchází. Patří mezi ně stanovení účelu hodnocení jakosti, identifikace hodnoceného produktu, volba strategie hodnocení a návrh modelu jakosti pro konkrétní hodnocené knihovny, tj. hodnocených charakteristik a podcharakteristik, a výběr jejich přidružených stupnic a požadavků.

**Účelem hodnocení** jakosti v rámci diplomové práce je hodnocení jakosti produktu z pohledu běžného uživatele, byť se základní znalostí skriptování v Pythonu. Zjednodušeně se v rámci kvantifikovatelnosti jedná o nalezení odpovědi na otázku, kterou z hodnocených knihoven zařadit do portfolia technických schopností pro účely usnadnění práce. Hodnocení proběhne postupně dle zvolených podcharakteristik.

**Identifikace hodnoceného produktu** již proběhla v rámci teoretické části práce, včetně podrobné definice jednotlivých pojmů, avšak pro přehlednost se jedná o knihovny jazyka Python používané v rámci automatizace kancelářské činnosti.

**Strategie hodnocení** je zvolena s ohledem na zvolený účel hodnocení. Proběhne výčet relevantních knihoven pro hodnocení, rozdělení předmětných knihoven do pěti kategorií z funkčního anebo logického hlediska, hodnocení proběhne postupně podle zvolených podcharakteristik.

Modifikovaný model kvality tvoří zvolené charakteristiky a je popsán v další podkapitole – a obsažen v příloze B. Postup hodnocení a měření dat bude uveden u každé z vybraných podcharakteristik. Výsledky měření, společně s komentářem a odůvodněním, budou uvedeny v další podkapitole a bude z nich vyvozen dílčí závěr jednotlivých kategorií s tím, že po vyhodnocení všech podcharakteristik bude vyvozeno závěrečné hodnocení v rámci splnění předem stanovených požadavků.

### 4.1.1 Model jakosti

Specifika modelu spočívají za prvé v aplikaci na open-source knihovny jazyka Python, za druhé v aplikaci na oblast knihoven pro automatizaci kancelářské činnosti – zúžené na vybrané oblasti.

První a nejpodstatnější charakteristika je **Funkční Přiměřenost**, kde budou hodnoceny všechny podcharakteristiky podle normy ISO 25010, funkční úplnost, správnost a vhodnost. Tudíž bude zjišťován obsah funkcí – splňují-li vybrané úkony, dále jestli jsou výsledky uspokojivé a s potřebnou mírou přesnosti, a jestli knihovna poskytuje usnadnění splnění stanovených úkolů a cílů v poměru k použití jiné knihovny, jiného nástroje, než jsou knihovny jazyka Python a bez použití procesu automatizace. Charakteristika obsahuje tři podcharakteristiky – hodnocené v rámci práce jako kritéria pro výběr mezi variantami knihoven:

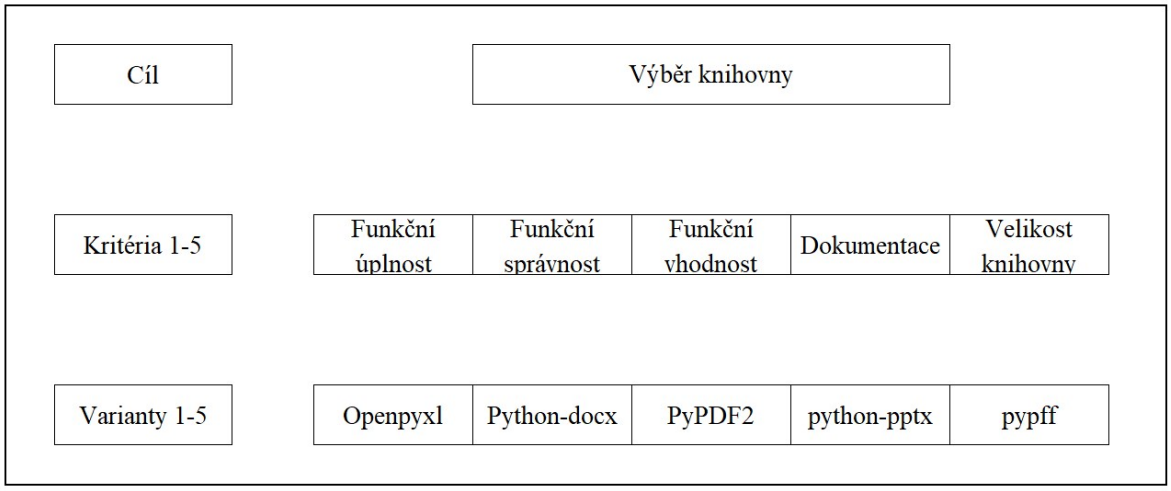
- Funkční úplnost (K1)
    - Funkčnost systému splňuje všechny zadané úkoly a cíle uživatele.
- Správnost funkcí (K2)
    - Systém vrací správné výsledky s určitou mírou přesnosti.
- Vhodnost funkcí (K3)
    - Funkce usnadňují splnění stanovených úkolů a cílů.

Další charakteristiky nevychází z norem řady ISO 25000 SQuaRE a jsou specifické pro oblast práce, **Dokumentace** (a spolehlivost knihovny) (K4) a **velikost knihovny** (K5). Dokumentace částečně zahrnuje hodnocení Sourcerank podle žebříčku hodnocení na (libraries.io). Došlo k vybrání takových podcharakteristik, aby byly knihovny posouzeny z různých náležitostí zaměřených na pohled běžných uživatelů, vynechány byly ty, které nejsou relevantní z hlediska této skupiny aktérů. Hodnocení bylo omezeno vzhledem k rozsahu práce.

**V příloze B je uvedena navržená modifikace modelu jakosti pro následné využití v rámci vícekriteriálního porovnávání.**

**V tabulce v příloze C jsou uvedeny některé další (pod)charakteristiky, zvažované jako možná kritéria.**

Níže je na obrázku uvedena hierarchická struktura pro výběr knihovny jazyka Python využitím metody AHP.




Obrázek 17 Ilustrativní tříúrovňová hierarchie - jeden cíl, pět kritérií a pět variant, zdroj: vlastní

### 4.1.2 Stupnice měření

V rámci této podkapitoly je uvedena stupnice pro stanovení vah kritérií - zhodnocení jednotlivých kritérií, a to devítibodá (příp. pětibodá) stupnice pro ohodnocení párových porovnání kritérií. Data budou vyhodnocována dle uvedené stupnice podle kardinálních informací, váhy odvozeny jako geometrický průměr řádků matice, pode vzájemné důležitosti kritérií.

Stupnice pro ohodnocení párových porovnání kritérií je následující:

- 1 - stejně významná, rovnocenná kritéria i a j
- 3 - slabě preferované kritérium i před j
- 5 - silně preferované kritérium i před j
- 7 - velmi silně preferované kritérium i před j
- 9 - absolutně preferované kritérium i před j

Hodnocené kritéria jsou následující:

- K1 Funkční úplnost
- K2 Funkční správnost
- K3 Funkční vhodnost
- K4 Dokumentace
- K5 Velikost knihovny

Hodnocené knihovny (varianty) jsou následující:

- Knihovna 1 openpyxl
- Knihovna 2 Python-docx
- Knihovna 3 PyPDF2
- Knihovna 4 python-pptx
- Knihovna 5 pypff

Vztahy mezi kritérii (druhá úroveň hierarchie) pro stanovení vah jsou následující:

**Párové porovnávání kritérií**

- Funkční úplnost je silně preferována před funkční správností.
- Funkční úplnost je silně preferována před funkční vhodností.
- Funkční úplnost je velmi silně preferována před dokumentací.
- Funkční úplnost je absolutně preferována před velikostí knihovny.
- Funkční správnost je slabě preferována před funkční vhodností.
- Funkční správnost je silně preferována před dokumentací.
- Funkční správnost je velmi silně preferována velikostí knihovny.
- Funkční vhodnost je silně preferována před dokumentací.
- Funkční vhodnost je velmi silně preferována velikostí knihovny.
- Dokumentace je silně preferována před velikostí knihovny.

Na obrázku jsou tyto uvedené vztahy stanovené jako váhy v Saatyho matici.

| 0 | **Váhy stanovené Saatyho metodou** | | Kritéria | |
|---|---|---|---|---|

| | **K1** | **K2** | **K3** | **K4** | **K5** | **$b_j$** | **$v_j$** |
|---|---|---|---|---|---|---|---|
| **K1** | **1** | 5 | 5 | 7 | 9 | 4.359695 | 0.533548 |
| **K2** | 0.2 | **1** | 3 | 5 | 7 | 1.838416 | 0.224989 |
| **K3** | 0.2 | 0.3333333 | **1** | 7 | 9 | 1.332447 | 0.163067 |
| **K4** | 0.142857 | 0.2 | 0.111111 | **1** | 5 | 0.436648 | 0.053438 |
| **K5** | 0.111111 | 0.1428571 | 0.111111 | 0.2 | **1** | 0.203934 | 0.024958 |
| | | | | | **Σ $b_i$** | 8.17114 | 1 |
| **Váha kriteria** | | | | | | | |

| | | |
|---|---|---|
| | **l_max** | **5.782616** |
| C.I. | **I_s** | 0.005234 |

Obrázek 18 Váhy stanovené Saatyho metodou pro jednotlivá kritéria, zdroj: vlastní

Výstupný vektor vah bude využit při stanovování preferencemi mezi variantami.

**Míra konzistence je menší než 0.1, tudíž lze matici považovat za dostatečně konzistentní.**

## 4.2 Hodnocení jakosti IS

V této kapitole popis jednotlivých použitých kritérií včetně uvedení názvu i v angličtině, popisu, účelu a váhy.
**Modifikované prvky měření kvality pro použití jsou uvedeny v tabulce v příloze D.**

### 4.2.1 Funkční úplnost

**Název kritéria**: Funkční úplnost (Functional Completeness)
**Popis:** Knihovny splňují zadání a cíle.
**Účel:** V rámci práce se jedná se o nejdůležitější kritérium, to je zohledněno jeho váhou. Obecně se týká spokojenosti klienta (uživatele) vzhledem ke splnění jeho očekávání. V rámci práce bude použito v tom smyslu, kolik z uvažovaných vybraných úkonů/úkolů (celkem 10, 5 hodnocených v rámci tohoto kritéria) knihovna umožňuje splnit. Na úspěšnosti kritéria je závislá většina ostatních kritérií. Klíčová je odpověď na otázku, zda knihovna umožňuje plnění všech stanovených úkolů a cílů. Patří mezi indikátory standardu ISO/IEC 25010.

Pro měření této podcharakteristiky bylo zvoleno deset úkonů, respektive 5 hodnocených, z kterých jsou některé dále zpracovávány v praktické části ve formě snippletů, ty jsou v tabulce vyznačeny šedě. K ostatním úkonům je přihlédnuto v rámci slovního hodnocení dalších kritérií, z toho důvodu jsou uvedeny. V následující tabulce jsou zaznamenány naměřené hodnoty z jednotlivých měřených kategorií.

| Číslo | Popis \| Varianta | V1 | V2 | V3 | V4 | V5 |
|---|---|---|---|---|---|---|
| FÚ | Název úkolu/úkonu/zadání | openpyxl | Python-docx | PyPDF2 | python-pptx | pypff |
| 1 | Tvorba dokumentu a hromadné vytváření (*File creation and bulk creation*) | ANO | ANO | NE | ANO | NE |
| 2 | Extrakce obsahu (*Data/Content extraction*) | ANO | ANO | ANO | ANO | ANO |
| 3 | Manipulace s obsahem nativně (*Data update/modify*) | ANO | ANO | NE | NE | NE |
| 4 | Sloučení a/nebo rozpojení souboru (*File merge and/or split*) | ANO | NE | ANO | NE | NE |
| 5 | Čtení a úprava metadat a vlastností souboru (*Metadata fields and/or Custom Properties*) | ANO | ANO | ANO | ANO | ANO |
| 6 | Vyplnění formulářů a práce se šablonami (*templates*) | ANO | ANO | ANO | NE | NE |
| 7 | Další práce s templates | ANO | ANO | ANO | NE | NE |
| 8 | Integrace s dalšími knihovnami | ANO | NE | ANO | NE | ANO |
| 9 | Podpora více možných formátů | ANO | NE | NE | NE | NE |
| 10 | Další možné zvažované úkony specifické pro oblast | ANO | NE | ANO | ANO | ANO |
| | Počet splněných úkolů (úkonů) | 10 | 6 | 7 | 4 | 4 |
| | Úspěšnost | 100% | 60% | 70% | 40% | 40% |
| | **Počet splněných hodnocených úkolů (úkonů)** | **5** | **4** | **3** | **3** | **2** |
| | Pořadí | 1 | 3 | 2 | 4 | 4 |
| | Varianta | V1 | V2 | V3 | V4 | V5 |
| | **Pořadí z hlediska hodnocených úkolů** | **1** | **2** | **3** | **3** | **4** |

Tabulka 3 Přehled výsledků hodnocení u kritéria Funkční úplnost, zdroj: vlastní

**Párové porovnávání kritérií**

- Knihovna 1 je slabě preferována před knihovnou 2
- Knihovna 1 je silně preferována před knihovnou 3
- Knihovna 1 je silně preferována před knihovnou 4
- Knihovna 1 je velmi silně preferována před knihovnou 5
- Knihovna 2 je slabě preferována před knihovnou 3
- Knihovna 2 je slabě preferována před knihovnou 4
- Knihovna 2 je silně preferována před knihovnou 5
- Knihovna 3 je rovnocenná s knihovnou 4

- Knihovna 3 je slabě preferována před knihovnou 5
- Knihovna 4 je slabě preferována před knihovnou 5

| 1 | **Saatyho matice pro kritérium** | Funkční úplnost |
|---|---|---|

| 0.533547932 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $\mathbf{b_{ij}}$ | $\mathbf{v_{ij}}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 3 | 5 | 5 | 7 | 3.499708 | 0.501067 |
| Library 2 | 0.333333 | **1** | 3 | 3 | 5 | 1.718772 | 0.246083 |
| Library 3 | 0.2 | 0.3333333 | **1** | 1 | 3 | 0.72478 | 0.10377 |
| Library 4 | 0.2 | 0.3333333 | 1 | **1** | 3 | 0.72478 | 0.10377 |
| Library 5 | 0.142857 | 0.2 | 0.333333 | 0.333333 | **1** | 0.316474 | 0.045311 |
| | | | | | $\mathbf{\Sigma\ b_i}$ | 6.984514 | 1 |

Tabulka 4 Saatyho matice pro kritérium Funkční Úplnost (*Functional Completeness*), zdroj: vlastní

Na obrázku výše je Saatyho matice pro Funkční Úplnost. Knihovna Python-docx sice teoreticky funkcionalitu spojení/rozpojení souboru podporuje, ale vzhledem k tomu, že je neúplná dokumentace, nedá se jednoduše zrealizovat a je pohlíženo jako nepodporované, to stejné platí pro Pypdf2 a nativní úpravu obsahu.

**Index konzistence vychází 0.031722792 tj. menší než 0.1. Matice je konzistentní.**

Kritérium má největší váhu a tudíž je nejdůležitější a pro výsledek v zásadě rozhodující. Jedná se o kritérium maximalizační. Stanovené úkony jsou subjektivní, porovnání má spíše demonstrativní účel a proto byly v rámci přehledu výsledků hodnocení u kritéria ponechány i další možné úkony, které nejsou hodnoceny – pro představu o možnostech praktičtějšího uchopení tématu. Vedle toho jsou možnosti hodnocených knihoven posuzována v rámci slovního hodnocení u dalších kritérií, funkční vhodnosti a správnosti.

### 4.2.2 Funkční správnost

**Název kritéria**: Funkční správnost (*Functional Correctness*)
**Popis:** Míra, se kterou knihovna vrací správné výsledky požadavků s požadovanou mírou přesnosti.
**Účel:** Jedná se o jedno z důležitých kritérií, to je zohledněno jeho váhou. obecně se týká míry přesnosti výsledků systému. V rámci práce bude použito v tom smyslu, že v rámci plnění vybraných úkonů musí uživatel provést pouze kroky nezbytné k dokončení úkolu, s vyloučením jakýchkoli dodatečných zbytečných kroků. Klíčová je odpověď na otázku, zda

knihovna poskytuje správné výsledky s potřebnou mírou přesnosti. Patří mezi indikátory standardu ISO/IEC 25010.

**Slovní ohodnocení**

Ohodnocení párových porovnání kritérií stanoveno na základě úvahy autora, a to z těchto závěrů:

- Knihovny spolupracující s formáty OOXML (openpyxl, Python-docx a python-pptx) jsou co se týče přesnosti výsledků **rovnocenné**, a v rámci srovnávání výrazně lepší než ostatní hodnocené (1)
- Knihovna PyPDF2 v rámci spolupráce s PDF v závislosti na standardech PDF (roli hraje strojová čitelnost) poskytuje **průměrně** uspokojivé výsledky (2)
- Knihovna pypff v rámci interakce se soubory e-mailových klientů umožňuje spolupráci pouze s formáty PST a OST, další požadované a možné jako MSG, EML příp. MBOX neumožňuje, v rámci srovnání poskytuje výrazně **nejhorší** výsledky (3).

**Párové porovnávání kritérií**

- Knihovna 1 je rovnocenná s knihovnou 2
- Knihovna 1 je silně preferována před knihovnou 3
- Knihovna 1 je rovnocenná s knihovnou 4
- Knihovna 1 je absolutně preferována před knihovnou 5
- Knihovna 2 je silně preferována před knihovnou 3
- Knihovna 2 je rovnocenná s knihovnou 4
- Knihovna 2 je absolutně preferována před knihovnou 5
- Knihovna 3 je silně preferována před knihovnou 5
- Knihovna 4 je silně preferována před knihovnou 3
- Knihovna 4 je silně preferována před knihovnou 5

| 2 | **Saatyho matice pro kritérium** | Funkční správnost |
|---|---|---|

| 0.2249889 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $\mathbf{b_{ij}}$ | $\mathbf{v_{ij}}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 1 | 5 | 1 | 9 | 2.1411274 | 0.308994 |
| Library 2 | 1 | **1** | 5 | 1 | 9 | 2.1411274 | 0.308994 |
| Library 3 | 0.2 | 0.2 | **1** | 0.2 | 5 | 0.5253056 | 0.075809 |
| Library 4 | 1 | 1 | 5 | **1** | 5 | 1.9036539 | 0.274724 |
| Library 5 | 0.111111 | 0.111111 | 0.2 | 0.2 | **1** | 0.2181298 | 0.031479 |
| | | | | | **Σ $\mathbf{b_i}$** | 6.929344 | 1 |

Tabulka 5 Saatyho matice pro kritérium Funkční Správnost (Functional Correctness), zdroj: vlastní

**Index konzistence vychází 0.056621001 tj. menší než 0.1. Matice je konzistentní.**

V rámci vyšší míry objektivity by bylo vhodnější porovnávat například jen knihovny pro OOXML, pak by byla výpovědní hodnota vyšší. Oproti hodnocené funkční úplnosti je váha kritéria nízká, tak zkreslení tolik výsledek neovlivní.

### 4.2.3 Funkční vhodnost

**Název kritéria**: Funkční správnost (*Functional Appropriateness*)
**Popis:** Míra, do které funkce softwaru usnadňují plnění stanovených úkolů a cílů.
**Účel:** Jedná se o jedno z důležitých kritérií, obecně se týká spokojenosti klienta vzhledem ke splnění jeho očekávání. V rámci práce bude použito ve smyslu, do jaké míry přináší knihovna užitek v podobě usnadnění splnění stanovených úkonů/úkolů a cílů v porovnání s plněním těchto úkolů bez integrace knihovny do pracovního toku/ průběhu práce (*workflow*) tj. bez automatizace procesu, případně v porovnání s dalšími nástroji domácími k oblasti. Klíčová je tedy odpověď na otázku, zda knihovna usnadňuje plnění stanovených úkolů a cílů. Patří mezi indikátory standardu ISO/IEC 25010.

**Slovní ohodnocení**

Ohodnocení párových porovnání kritérií stanoveno obdobně jako u předchozích kritérií na základě úvahy autora, a to z těchto důvodů:

- Knihovny spolupracující s formáty OOXML (openpyxl, Python-docx a python-pptx) jsou co se týče funkční vhodnosti v porovnání s ostatními **rovnocenné** (1)
- Knihovna PyPDF2 v rámci spolupráce s PDF v závislosti na standardech PDF (roli hraje strojová čitelnost) poskytuje **průměrně** uspokojivé výsledky (2)
- Knihovna pypff v rámci interakce se soubory e-mailových klientů je v rámci porovnávání nejméně vhodná, poskytuje výrazně **nejhorší** výsledky (3).

**Párové porovnávání kritérií**

- Knihovna 1 je rovnocenná s knihovnou 2
- Knihovna 1 je silně preferována před knihovnou 3
- Knihovna 1 je rovnocenná s knihovnou 4
- Knihovna 1 je absolutně preferována před knihovnou 5
- Knihovna 2 je silně preferována před knihovnou 3
- Knihovna 2 je rovnocenná s knihovnou 4
- Knihovna 2 je absolutně preferována před knihovnou 5
- Knihovna 3 je silně preferována před knihovnou 5
- Knihovna 4 je silně preferována před knihovnou 3
- Knihovna 4 je silně preferována před knihovnou 5

| 3 | **Saatyho matice pro kritérium** | Funkční vhodnost |
|---|---|---|

| 0.163067415 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $b_{ij}$ | $v_{ij}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 1 | 5 | 1 | 9 | 2.141127 | 0.308994 |
| Library 2 | 1 | **1** | 5 | 1 | 9 | 2.141127 | 0.308994 |
| Library 3 | 0.2 | 0.2 | **1** | 0.2 | 5 | 0.525306 | 0.075809 |
| Library 4 | 1 | 1 | 5 | **1** | 5 | 1.903654 | 0.274724 |
| Library 5 | 0.111111 | 0.1111111 | 0.2 | 0.2 | **1** | 0.21813 | 0.031479 |
| | | | | | **Σ $b_i$** | 6.929344 | 1 |

Tabulka 6 Saatyho matice pro kritérium Funkční Vhodnost (F. Appropriateness), zdroj: vlastní

**Index konzistence vychází 0.056621001 tj. menší než 0.1. Matice je konzistentní.**

Obdobně jako u předchozího kritéria by vyšší vypovídající hodnotu mělo porovnávání omezit v rámci jednotlivých oblastí.

### 4.2.4 Dokumentace

**Název kritéria**: Dokumentace (*Dokumentation*)
**Popis:** Míra, do které je dokumentace knihovny srozumitelná, úplná a aktuální a knihovna podle SourceRank skóre spolehlivá.
**Účel:** Jedná se o jedno z méně důležitých kritérií, v rámci práce bude použito ve smyslu ohodnocení atributů dokumentace a dále skóre knihovny na SourceRank

Způsob hodnocení je v případě hodnocení dokumentace založen na ohodnocení ve škále 1 – 6 podle subjektivního ohodnocení autora atributů srozumitelnosti, úplnosti, aktuálnosti a funkčního pokrytí relevantních úkonů, a spolehlivosti knihovny podle jejího

skóre SourceRank, jenž bylo částečně převzato z Libraries.io (libraries.io), kde jsou knihovny hodnoceny za účelem podpory kvalitních knihoven. Bylo zvažováno využití i metrik hodnocených knihovnou radon-repository-scorer a nástrojem jazyka Python radon, které poskytuje velké množství relevantních a podnětných sledovaných atributů, avšak vzhledem k rozsahu, specifikům oblasti práce a využití hodnocení za pomocí vybraných kritérií ISO SQuaRE nakonec nebylo do hodnocení zahrnuto. Pro ilustraci a snahu o relativní úplnost a její zachování, jsou metriky nástroje Radon podle (radon.readthedocs.io) níže uvedeny:

- Core contributors - základní přispěvatelé: počet přispěvatelů, jejichž celkový počet revizí tvoří 80 % nebo více z celkového počtu příspěvků.
- Continuous integration (CI) - nepřetržitá integrace (CI): úložiště má důkaz o existenci služby CI, dáno přítomností konfiguračního souboru vyžadovaného touto službou (např. a.travis.yml pro TravisCI).
- Comments ratio - poměr komentářů: poměr mezi komentáři a řádky kódu.
- Commit frequency - četnost odevzdání: průměrný počet odevzdání za měsíc.
- Commit frequency - frekvence vydání: průměrný počet událostí vydání za měsíc.
- Lines of Code - řádky kódu: počet spustitelných řádků kódu.
- Ratio of IaC scripts - poměr skriptů IaC: poměr mezi soubory Infrastructure-as-Code (IaC) a celkovými soubory.
- McCabe's complexity - McCabeova složitost, tj. cyklomatická složitost
- Halstead metrics - Halsteadovy metriky
- Maintainability Index - Index udržovatelnosti
- raw metrics - hrubé metriky (podle dokumentace na Github zahrnují SLOC, řádky komentářů, prázdné řádky atd.) (radon.readthedocs.io).

**V tabulce v příloze E je uvedeno ohodnocení jednotlivých atributů, jejímž výstupem je pořadí knihoven pro použití v rámci následného stanovení pořadí variant.**

Popis Ohodnocení atributů podle SourceRank je k dispozici vždy na stránce příslušné knihovně na libraries.io, podle (libraries.io) např. základní informace znamená přítomnost popisu, linku na homepage a keywords, nedávné vydání je ohodnoceno 1 pokud byla knihovna aktualizována v posledních šesti měsících, hodnota závislých balíčků je reflektována za pomocí logaritmické stupnice krát dva a tak podobně, v případě hodnocení

dokumentace jsou atributy vnímány rovnocenně, knihovna ohodnocena 1 nejhorší možné, 6 nejlepší (libraries.io). Výsledné pořadí je použito v rámci hodnocení metodou AHP. Na obrázku níže jsou uvedena párová ohodnocení pro toto kritérium.

**Párové porovnávání kritérií**

- Knihovna 1 je slabě preferována před knihovnou 2
- Knihovna 1 je silně preferována před knihovnou 3
- Knihovna 1 je velmi silně preferována před knihovnou 4
- Knihovna 1 je absolutně preferována před knihovnou 5
- Knihovna 2 je slabě preferována před knihovnou 3
- Knihovna 2 je silně preferována před knihovnou 4
- Knihovna 2 je velmi silně preferována před knihovnou 5
- Knihovna 3 je silně preferována před knihovnou 5
- Knihovna 3 je slabě preferována před knihovnou 4
- Knihovna 4 je slabě preferována před knihovnou 5

| 4 | **Saatyho matice pro kritérium** | Dokumentace |
|---|---|---|

| 0.0534379 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $b_{ij}$ | $v_{ij}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 3 | 5 | 7 | 9 | 3.9362834 | 0.510039 |
| Library 2 | 0.333333 | **1** | 3 | 5 | 7 | 2.036168 | 0.263834 |
| Library 3 | 0.2 | 0.333333 | **1** | 3 | 5 | 1 | 0.129574 |
| Library 4 | 0.142857 | 0.2 | 0.333333 | **1** | 3 | 0.4911186 | 0.063636 |
| Library 5 | 0.111111 | 0.142857 | 0.2 | 0.333333333 | **1** | 0.2540467 | 0.032918 |
| | | | | | **Σ $b_i$** | 7.7176168 | 1 |

Tabulka 7 Saatyho matice pro kritérium Dokumentace (*Dokumentation*), zdroj: vlastní

**Index konzistence vychází 0.059368813 tj. menší než 0.1. Matice je konzistentní.**

Kritérium je ohodnoceno nízce, proto nemá moc vypovídající hodnotu, v praxi může být vhodnější stanovit kritérium jako důležitější.

### 4.2.5 Velikost knihovny

**Název kritéria**: Velikost (*Size*)

**Popis:** Míra velikosti knihovny.

**Účel:** Jedná se o jedno z méně důležitých kritérií, v rámci práce bude použito ve smyslu popisu, velikost knihovny v rámci virtuálního prostředí.

**V tabulce v příloze F je uvedena velikost a pořadí jednotlivých knihoven z hlediska tohoto kritéria.**

**Párové porovnávání kritérií**

- Knihovna 1 je slabě preferována před knihovnou 4
- Knihovna 2 je silně preferována před knihovnou 1
- Knihovna 2 je velmi silně preferována před knihovnou 4
- Knihovna 2 je slabě preferována před knihovnou 5
- Knihovna 3 je velmi silně preferována před knihovnou 1
- Knihovna 3 je slabě preferována před knihovnou 2
- Knihovna 3 je absolutně preferována před knihovnou 4
- Knihovna 3 je silně preferována před knihovnou 5
- Knihovna 5 je slabě preferována před knihovnou 1
- Knihovna 5 je silně preferována před knihovnou 4

| 5 | **Saatyho matice pro kritérium** | Velikost knihovny |
|---|---|---|

| 0.024957824 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $\mathbf{b_{ij}}$ | $\mathbf{v_{ij}}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 0.2 | 0.2 | 3 | 0.333333 | 0.525306 | 0.070082 |
| Library 2 | 5 | **1** | 0.333333 | 7 | 3 | 2.036168 | 0.271647 |
| Library 3 | 5 | 3 | **1** | 9 | 5 | 3.68011 | 0.490967 |
| Library 4 | 0.333333 | 0.1428571 | 0.111111 | **1** | 0.2 | 0.254047 | 0.033893 |
| Library 5 | 3 | 0.3333333 | 0.2 | 5 | **1** | 1 | 0.133411 |
| | | | | | $\mathbf{\Sigma\ b_i}$ | 7.49563 | 1 |

Tabulka 8 Saatyho matice pro kritérium Velikost (Size), zdroj: vlastní

**Index konzistence vychází 0.067399366 tj. menší než 0.1. Matice je konzistentní.**

Kritérium je ohodnoceno nízce, protože nemá moc vypovídající hodnotu - v praxi by totiž bylo vhodnější hodnotit např. velikost jednotlivých importovaných objektů, v případě když pro úkon není třeba importovat celou knihovnu, nebo závislostí knihoven, je-li potřeba nějaké instalovat společně s knihovnou, a tak podobně.

## 4.3 Zhodnocení a doporučení

Níže je uvedeno výsledné pořadí variant podle AHP.

| 6 | **Pořadí variant podle metody AHP** | | |
|---|---|---|---|

| 1 | Funkční úplnost | Funkční správnost | Funkční vhodnost | Dokumentace | Velikost knihovny | | $b_j$ | $v_j$ | **Pořadí** |
|---|---|---|---|---|---|---|---|---|---|
| **Váha** | 0.533548 | 0.224989 | 0.163067 | 0.05343788 | 0.024958 | **1** | G.M. | Synt. pref. | **x** |
| openpyxl | **0.501067** | **0.308994** | **0.070082** | 0.308994236 | 0.510039 | 1.6991757 | 0.279641 | 0.330686 | 1 |
| Python-docx | **0.246083** | **0.308994** | **0.271647** | 0.308994236 | 0.263834 | 1.3995529 | 0.278781 | 0.32967 | 2 |
| PyPDF2 | **0.10377** | **0.075809** | **0.490967** | 0.075808844 | 0.129574 | 0.8759283 | 0.130562 | 0.154394 | 3 |
| python- | **0.10377** | **0.274724** | **0.033893** | 0.274723542 | 0.063636 | 0.7507453 | 0.111054 | 0.131326 | 4 |
| pypff | **0.045311** | **0.031479** | **0.133411** | 0.031479141 | 0.032918 | 0.2745979 | 0.045601 | 0.053925 | 5 |
| | | | | | | 5 | 0.845639 | 1 | |

Obrázek 19 Pořadí variant

**V tabulce v příloze G je uvedeno pořadí variant podle metody AHP. velikost a pořadí jednotlivých knihoven z hlediska tohoto kritéria.**

**V tabulce v příloze H je uvedeno pro kontrolu párové ohodnocení jednotlivých variant.**

**V tabulce v příloze CH je pro kontrolu uveden Přehled výsledků hodnocení – všechny výstupy z hodnocení**

Vzhledem k tomu, že v rámci hodnocení vyšla jako nejvhodnější knihovna Openpyxl, bude objektová metodologie aplikována na ní, a to na první hodnocený úkon Vytvoření souboru. U dalších realizovaných oblastí by byla objektová notace obdobná.

V přílohách G-CH jsou uvedeny výstupy výpočtů AHP metody, pro kontrolu jednotlivé párové ohodnocení variant a celkové pořadí variant.

- Největší důležitost má podle autora kritérium funkční úplnosti, to je reflektováno jeho váhou.
- Intenzita preferencí u kritérií funkční vhodnosti a funkční správnosti stanovena ze slovního popisu.
- Co se týče ohodnocení v kritériu funkční správnosti byla zohledněna nestandardizace jednotlivých formátů.
- Co se týče ohodnocení v kritériu funkční vhodnosti byla zohledněna existence dalších knihoven jazyka Python, vhodnější k použití k vybranému účelu a možnosti automatizace pro nestandardizace jednotlivých formátů.

- Jak v případě kritéria funkční vhodnosti a správnosti byla použita metrika čas, s tím, že její stanovení bylo určeno na základě úvahy o ušetření vykonávání daných úkonů, vzhledem k nesourodosti porovnávaných kategorií kancelářského SW pak ohodnocení proběhlo slovně.

**V následujících podkapitolách jsou ilustrativně uvedeny příklady implementace jednotlivých činností ve (většinou) hodnocených knihovnách, a to ve formě snippletů s funkcemi.**

### 4.3.1 Tabulkové procesory - MS Office - MS Excel (Openpyxl)

**V příloze I je uveden Use Case diagram pro spolupráci s XLSX – vytvoření souboru**

Rozšiřující případy užití *Vytvoř hromadně více souborů* a *Vytvoř soubor podle template (šablony)* nebyly v rámci snippletu implementovány.

**V příloze J je uveden demonstrativní Scénář případu užití (*Use Case Scenario*) pro činnost (*task*) *Vytvořit soubor v XLSX*.**

V příloze K jsou uvedeny snipplety pro všechny hodnocené činnosti v rámci spolupráce s knihovnou openpyxl.

Tvorba dokumentu

**Snipplet pro tvorbu XLSX souboru je uveden v příloze K.**

- Knihovna Openpyxl nepodporuje formát XLS, ale v rámci nabídky z portfolia knihoven jazyka Python možnosti jsou.
- S příponou ODS knihovna Openpyxl vůbec neinteraguje. Nejblíže k Openpyxl pro „.ods“ je v rámci knihoven Python knihovna pyexcel-ods.
- Pro formát CSV je vhodná built-in knihovna jazyka Python pojmenovaná csv.

Extrakce dat

**Snipplet pro extrakci dat z XLSX souboru je uveden v příloze K.**

- Na obrázku v příloze K je uveden příklad využití knihovny openpyxl pro extrakci obsahu ze souboru ve formátu XLSX.
- Platí co v předchozím případě, knihovna nepodporuje formát XLS, ale v rámci nabídky knihoven jazyka možnosti jsou, pro formát CSV je built-in knihovna csv, s příponou ODS knihovna vůbec neinteraguje, vhodné je použití knihovny pyexcel-ods.

Manipulace s obsahem

**Snipplet pro manipulaci s obsahem XLSX souboru je uveden v příloze K.**

- Na obrázku v příloze K je uveden příklad využití knihovny openpyxl pro manipulaci obsahu souboru XLSX, u ostatních formátů platí to stejné co v předchozím případě.

Manipulace se souborem

**Snipplet pro manipulaci se souborem XLSX je uveden v příloze K.**

- Na obrázku v příloze K je uveden příklad využití knihovny openpyxl pro manipulaci obsahu souboru XLSX, u ostatních formátů platí to stejné co v předchozích případech.
- Na obrázku v příloze K je uvedena manipulace se souborem se souborem XLSX – rozdělení souboru - převod souboru formátu XLSX s více listy na více souborů knihovnou Openpyxl,

Metadata

**Snipplet pro úprava metadat souboru knihovnou Openpyxl souboru XLSX je uveden v příloze K.**

- Na obrázku v příloze K je uveden příklad využití knihovny openpyxl pro manipulaci s metadaty souboru XLSX, konkrétně změna vlastností název a autor souboru, u ostatních formátů platí to stejné co v předchozích případech.
- V rámci modulu *openpyxl.packaging.core* "Core" v základních vlastnostech dokumentu odkazuje na Dublin Core, metadatový standard, který definuje základní sadu prvků pro popis zdrojů podle ISO 15836-2:2019.

### 4.3.2 Textové procesory MS Office - MS Word (python-docx)

Tvorba dokumentu a manipulace s obsahem

**Snipplet pro tvorbu DOCX souboru a manipulaci s obsahem je uveden v příloze L.**

- Knihovna nepodporuje formát DOC, ale v rámci nabídky knihoven jazyka Python možnosti jsou – například čtení knihovnou textract, nebo převod binárního DOC na otevřený DOCX knihovnou win32com.
- S příponou ODT knihovna vůbec neinteraguje. Nejblíže k python-docx pro ODT je odfpy, knihovna pro čtení a zápis souborů OpenDocument v. 1.2. .
- Knihovna v kombinaci s dalšími umožňuje i extrakci informací o fontu a tak podobně, ale v rámci práce realizováno nebylo.

Obsah vytvořeného souboru je na obrázku níže.

Vložený text do souboru formátu DOCX aplikace MS WORD - do odstavce (paragraph) **MS WORD** Příklad změny barvy fontu

Obrázek 20 Obsah vytvořeného souboru NazevSouboru.docx se změnou obsahu, zdroj: vlastní

Extrakce dat

**Snipplet využití knihovny python-docx pro extrakci obsahu ze souboru ve formátu DOCX je uveden v příloze L.**

- Další formáty obdobné, co v předchozím případě.

Manipulace s obsahem

- Samostatně manipulace s obsahem již vytvořeného dokumentu je obdobná jako u knihovny Openpyxl, vzhledem k rozsahu nebyla uvedena, ale příklad výše tyto možnosti dokazuje.

Manipulace se souborem

- Snipplet pro manipulaci se souborem – spojení souborů DOCX formátu knihovnou Python-docx není uveden, a to vzhledem k neúplné dokumentaci, ačkoliv tuto funkcionalitu (činnost) knihovna teoreticky umožňuje.
- Vedle toho existují z hlediska portfolia knihoven jazyka Python jednodušší možnosti – například win32auto nebo built-in knihovna zip, a podobně.

Metadata

**Snipplet využití knihovny python-docx pro úpravu metadat souboru ve formátu DOCX je uveden v příloze L.**

### 4.3.3 Portable Document Format – Adobe Acrobat (PyPDF2)

Tvorba dokumentu

**Snipplet využití knihovny Reportlab pro vytvoření souboru ve formátu PDF je uveden v příloze M.**

- Hodnocená knihovna PyPDF2 tuto funkcionalitu nepodporuje.
- Ilustrativně je v příloze M alespoň uvedeno, jak je možné soubor PDF vytvořit v rámci jazykových knihoven jazyka Python, a to pomocí knihovny Reportlab.

Extrakce dat

**Snipplet využití knihovny** PyPDF2 **pro načtení obsahu souboru ve formátu PDF je uveden v příloze M.**

- Snipplet pro přímou extrakci dat z PDF formátu knihovnou PyPDF2 není uveden, ale je podporováno. Existují další, specializované knihovny, a z hlediska hodnocení byly brány v potaz různé typy PDF, kdy především u těch odpovídajícím strojově čitelným standardům a jsou možné jednodušší možnosti extrahování obsahu, dále je možné využít knihovny v rámci oblasti OCR.

Manipulace s obsahem

- Snipplet pro manipulaci obsahu souboru PDF formátu knihovnou PyPDF2 není uveden, ale je podporováno. Stejně jako u předchozího bodu platí odlišnosti PDF vzhledem ke standardům a strojové čitelnosti.

Manipulace se souborem

Snipplet pro manipulaci se souborem – rozdělení souboru/spojení souboru PDF formátu knihovnou PyPDF2 není uveden, ale je podporováno a popsáno v dokumentaci.

Metadata

**Snipplet využití knihovny** PyPDF2 **pro čtení metadat souboru ve formátu PDF je uveden v příloze M.**

### 4.3.4 Tvorba prezentací MS Office - MS Powerpoint (python-pptx)

Tvorba dokumentu

**Snipplet využití knihovny python-pptx pro tvorbu souboru ve formátu PPTX je uveden v příloze N.**

- Obsah vytvořeného souboru je na obrázku níže.
- Knihovna nepodporuje formát PPT, ale v rámci nabídky knihoven jazyka možnosti jsou.
- S příponou .odp knihovna vůbec neinteraguje. Nejblíže k python-pptx pro odp je odpslides.


Obrázek 21 Obsah vytvořeného souboru NazevSouboru.pptx, zdroj: vlastní

Extrakce dat

**Snipplet využití knihovny python-pro extrakci dat ze souboru ve formátu PPTX je uveden v příloze N.**

- Snipplet pro extrakci dat z PPTX formátu knihovnou python-pptx není uveden, ale je podporováno, vzhledem k tomu že se jedná o soubory typu OOXML, obdobné jako v případě formátů XLSX a DOCX.

Manipulace s obsahem

- Snipplet pro manipulaci obsahu souboru PPTX formátu knihovnou python-pptx není uveden, ale je podporováno, obdobné jako v případě formátů XLSX a DOCX.

Manipulace se souborem

- Snipplet pro manipulaci se souborem – rozdělení souboru/spojení souboru PPTX není uveden, ale je podporováno, obdobné jako v případě formátů XLSX a DOCX.

Metadata

- Snipplet pro úpravu metadat souboru PPTX knihovnou python-pptx není uveden, ale je podporováno.

### 4.3.5 Klient ke správě elektronické pošty – MS Outlook

Tvorba dokumentu

**Snipplet využití knihovny email pro tvorbu souboru ve formátu MSG je uveden v příloze O.**

- Hodnocená knihovna tuto funkcionalitu nepodporuje. Ilustrativně je na obrázku v **příloze O** uvedeno, jak soubor MSG vytvořit v rámci jazykových knihoven jazyka Python pomocí built-in knihovny email.
- Z hlediska formátů PST/OFF nemá smysl realizovat a není podporováno. V úvahu teoreticky přichází například tvorba formou konverze více souborů MSG do PST, to umožňuje knihovna (modul) win32com, ale v rámci práce realizováno nebylo.

Extrakce dat a metadat

**Snipplet využití knihovny extract_msg pro extrakci dat a metadat z formátu MSG je uveden v příloze O.**

- Extrakci dat z formátu MSG knihovna nepodporuje. Čtení nebo úprava metadat souboru MSG knihovnou pypff není podporováno. Vhodná k použití je např. knihovna extract_msg. Ilustrativní příklad použití v případě extrakce obsahu a čtení metadat – odesílatel a subjekt e-mailu je uveden na obrázku níže
- Snipplet pro extrakci dat z formátů PST a OST knihovnou pypff není uveden, ale je podporováno. Knihovna Pypff podle dostupných informací umožňuje vyjmout text zprávy a přílohy, ale dokumentace je nedostatečná.
- Knihovna Pypff sice hypoteticky umožňuje číst z formátů PST a OST metadata jako čas vytvoření, čas doručení, odesílatele, počet příloh a tak dále, ale dokumentace je nedostatečná a tudíž v rámci práce není uvedeno.

Manipulace s obsahem

- Z hlediska formátu MSG nemá smysl realizovat, hodnocená knihovna pypff nepřichází v úvahu (formát nepodporuje), možnosti využití nějakých dalších jazykových knihoven jazyka Python z portfolia teoreticky jsou.
- Z hlediska formátů PST/OFF je popsaným způsobem u knihovny pypff podporována možnost např. extrahovat všechny zprávy jako MSG, ale v rámci práce kvůli rozsahu nebylo uvedeno.

Manipulace se souborem

- Z hlediska formátu MSG podle autora nemá smysl realizovat.

- Z hlediska formátu PST je popsaným způsobem v dokumentaci ke knihovně podporována možnost např. extrahovat všechny zprávy jako MSG, ale v rámci práce není uvedeno.

### 4.3.6 Syntéza dílčích hodnocení variant

Podle výsledků porovnávání podle zvolených kritérií je pro zvolené úkony podle metody AHP nejvhodnější knihovna Openpyxl.

- V rámci hodnocení AHP rozhodovací roli hrálo především kritérium Funkční úplnosti, ohodnocené největší váhou, neboť je podle autora z hlediska práce nejpodstatnější. Co se týče ohodnocení v kritériu funkční správnosti je zohledněna nestandardizace jednotlivých formátů, ohledně ohodnocení v kritériu funkční vhodnosti je zohledněna existence dalších knihoven jazyka Python, vhodnější k použití k specifickému účelu pro nestandardizace jednotlivých formátů, kritérium dokumentace určeno vlastním popsaným ohodnocením v kombinaci s atributy Sourcerank z webu libraries.io, jak je uvedeno v příloze E, kritérium Velikost bylo ohodnoceno jako nejméně důležité.
- Snipplety byly vytvořeny co nejjednodušší, pro dokázání teze práce, že je pomocí jazyka Python umožněno zjednodušit práci s minimem vstupních znalostí prostředí jazyka. Možností jak splnit stanovené cíle je i v rámci jednotlivých porovnávaných knihoven mnoho.
- Vedle toho určité komplikace z hlediska manipulace se souborovými formáty způsobuje původní binární formát aplikací kancelářského balíku Microsoft Office (tj. předchůdce formátu XLSX/DOCX/PPTX) a ačkoliv z pohledu uživatele zůstaly soubory zpětně kompatibilní s tradičními binárními formáty, hodnocené knihovny jazyka Python práci s předchozím formátem neumožňují.

# 5 Výsledky a diskuse

Výstupem práce je zdrojový kód který se dá použít pro ilustrativní splnění některých úkonů spadajících pod kancelářskou činnost. Vhodnost použití hodnocených knihoven závisí na mnoha okolnostech, některé nerealizované přístupy i z hlediska knihoven jazyka Python mohou být jednodušší.

Mezi další oblasti na které v práci nezbyl prostor, ale z hlediska kancelářské činnosti mohou být významné, patří například knihovny pro verifikaci digitálních podpisů jako je endesive či pyHanko, vedle toho pro interakci s podnikovým SW může umožnit ovládat myš a klávesnici a automatizovat interakci s jinými aplikacemi knihovna PyAutoGUI s jednoduchým API rozhraním, a dalšími relevantními oblastmi obecně můžou být i QR kódy, OCR, práce se specifickým oborovým SW (zpracování hlasu, networking, grafika, geodetika, projektování...) a automatizace činností na internetu jako je vyplňování formulářů prostřednictvím knihovny Selenium.

Vedle Pythonu umožňuje podobnou míru flexibility např.  Javascript, Perl nebo R, s tím, že k většině ze zmíněných SW jazyků nebo nástrojů existuje nějaká možnost propojení s jazykem Python. Další vynechanou a v mnoha ohledech pro oblast automatizace důležitou oblastí mohou být externí nástroje jako je pandoc, handbrake nebo tenserflow, pro Windows pak Autohotkey, Powershell, Windows Task Scheduler, pluginy do kancelářského SW a celkově automatizační platformy jako je Automatization Everywhere, nebo například v rámci prostředí Android Tasker, RESTask a tak podobně.

Ze závěrů pro autora vyplývajících z práce je důležitost používání otevřených formátů, definovanými mezinárodními standardy a komplikace ohledně uzavřených formátů. Dále je podstatný důraz na mezinárodní spolupráci ke standardizaci a důsledky z hlediska transparentnosti a možnostech interoperability, kterou to přináší - i z hlediska trendu digitalizace a výhod napojení na FOSS komunitu, vedle toho také důležitost dokumentace a její úplnosti a srozumitelnosti.

## 5.1 Použití v podnikové praxi

Pro ilustraci možného použití nejlépe vyhodnocené knihovny je v **Příloze P** uveden kód použitelný pro podnikovou praxi. Postup je popsán v poznámkách ke kódu, ale ilustrativně včetně možného použití pro případnou modifikaci, může být tento postup následující:

- Uživatel potřebuje vytvořit více souborů ve formátu XLSX s obsahem podle předepsaného vzoru (*Sablona.xlsx*) v novém adresáři (*NazevNoveSlozky*), soubory pojmenované podle jednotlivých řádků v jiném souboru formátu XLSX (*Polozky.xlsx*)
- Uživatel aktivuje virtuální prostředí s nainstalovanou openpyxl knihovnou a vyvolá funkci *VytvoritSoubory ()*.

Z důvodu velké podpory knihoven Python v podstatě nahradil VBA jako skriptovací jazyk v kanceláři a vedle volně dostupných open source modulů existují i placené verze některých knihoven umožňující snadnější integrace s produkty MS Office. Vedle toho jsou k dispozici i Add-iny umožňující používat například Microsoft Excel v kombinaci s Pythonem jako „user-friendly“ front-end pro práci s makry a UDF. Vedle Microsoftu je Python úzce spjat i se společností Google a umožňuje spolupráci s jejími produkty, i společně s  pro účely práce relevantními okruhy kancelářského SW, a využívá se i v oblasti CDN (*Content Delivery Network*). Velkým přínosem můžou být také možnosti datové analýzy a vizualizace dat. V práci jsou také přiblíženy trendy pro případné praktické použití v podnikové praxi relevantní.

Ke zvýšení výkonu jazyka Pythonu může dojít například implementací knihoven ctypes. Jako součást podnikové praxe může být například implementace do procesů podniku, například ve formě institutu průmyslových práv, zlepšovacího návrhu, vedoucího k provoznímu zdokonalení chodu zaměstnavatelova podniku, které bude spočívat v úspoře dosavadních nákladů, se všemi pozitivními důsledky jako je například zlepšení konkurenceschopnosti nebo ušetření nákladů.

# 6 Závěr

Předmětem diplomové práce bylo porovnání knihoven pro automatizaci jazyka Python. Vzhledem k obsáhlosti tématu byly vybrány modelové okruhy s příkladem praktického využití.

Za účelem kvantitativní komparace byly knihovny pro účely práce zjednodušeně kategorizovány do pěti oblastí, na které byla následně práce v praktické části zaměřena. S přihlédnutím k vybraným metrikám norem řady ISO/IEC 2500n byly oblasti následně z hlediska užitku kvantifikovány, a to s využitím vícekriteriálního rozhodování, konkrétně použitím Saatyho metody.

Z hlediska hodnocení knihoven v rámci oblastí kancelářské činnosti se jeví nejsložitější oblast práce s formáty PDF a elektronické pošty, a to z důvodu uzavřených formátů. Pro práci z hlediska porovnávání by bylo vhodnější téma ještě více zúžit, například na porovnávání jen v rámci OOXML. Dále by pro práci bylo na základě rešerše pravděpodobně vhodnější buď využití QME podle ISO/IEC 25021:2012, nebo využití norem řady SQuaRE z úhlu *Quality in Use*.

Mezi oblasti, které jsou nedílnou součástí automatizace, ale nebyly použity, patří například RPA, využití umělé inteligence, práce se souborovými formáty XML a JSON nebo třeba *DataFrames* knihovny Pandas. Dále obecně záležitosti týkající se webu (web scraping, automatizované vyplňování formulářů, a tak podobně) a především integrace kancelářských SW respektive formátů kancelářských SW uvedených v práci jako je Word, Excel a PDF s dalšími informačními systémy skrz API rozhraní, jenž bezesporu představuje pravděpodobně nejužitečnější a nejčastější využití automatizace pro urychlení administrativní činnosti. Na druhou stranu již existuje řada nástrojů, které repetitivní činnosti může ulehčit. Vývoj v oblasti automatizace a robotizace je v posledních letech překotný a oblast je hlavním předmětem zájmu v průmyslu 4.0 a přidružených trendech posledních let.

Testovací hypotéza zněla: programovací jazyk Python umožní zefektivnění kancelářské (administrativní, popř. technicko-hospodářské) činnosti bez ohledu na obor. Hypotéza byla potvrzena, a jako nejvhodnější knihovny pro daný okruh činností byly vybrány knihovny pro práci s aplikacemi (formáty aplikací) MS Excel, MS Word a MS Powerpoint, a dále se soubory elektronické pošty a formátem PDF, v rámci kterých byla za pomocí metody AHP vyhodnocena jako nejvhodnější knihovna pro práci s tabulkovými procesory knihovna Openpyxl.

Přes to, že je zřejmý rozdíl mezi možnostmi SW, výpočetní výkonu a znalostmi běžného uživatele, základy skriptování v jazyce s relativně jednoduchou syntaxí (přirovnávanou k pseudokódu) mohou umožnit zefektivnit činnost (nejen) hospodářsko-ekonomického pracovníka napříč obory, stejně tak jako možnost importovat „předpřipravenou“ knihovnu pro určitou oblast nebo úkon, nebo si vyhledat v rámci open-source komunity potřebná řešení.

# 7 Seznam použitých zdrojů

6 Best Office Suites to Use in 2022.[online].[cit. 2022-03-23]. Dostupné z: <https://www.greengeeks.com/blog/best-office-suites/>

Automatizace práce v ČR | Deloitte Česká republika. In: Deloitte Czech Republic [online].[cit. 30.03.2022]. Dostupné z: <https://www2.deloitte.com/cz/cs/pages/strategy-operations/articles/automatizace-prace-v-cr.html>

Archívní formát PDF/A | EARCHIVACE.[online].[cit. 2022-03-23]. Dostupné z: <http://www.earchivace.cz/legislativa-a-normy/typy-dokumentu-k-archivaci/>

ARLOW, Jim a Ila NEUSTADT. UML a unifikovaný proces vývoje aplikací. Brno: Computer Press, a. s., 2003, 408 s. ISBN 80-7226-947-X.

BLAHUTA, Jiří. ALGORITMIZACE. Moravská vysoká škola Olomouc, o.p.s..[online]. [cit. 2021-02-18]. Dostupné z: <https://www.mvso.cz/files/algoritmizace-studijni-text.pdf>

BOOCH, Grady, Ivar JACOBSON a James RUMBAUGH. The unified modeling language user guide. 2nd ed. Upper Saddle River: Addison-Wesley Professional, 2005, 494 s. ISBN 03-212-6797-4.

Business Process Management | BPM Automation. In: Automation Anywhere [online].[cit. 30.03.2022]. Dostupné z: <https://www.automationanywhere.com/rpa/business-process-management>

CELBOVÁ, Ludmila. metadata. In: KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV) Praha : Národní knihovna ČR, 2003-.[online].[cit. 2022-03-23]. Dostupné z: <https://aleph.nkp.cz/F/?func=direct&doc_number=000000543&local_base=KTD>.

compilation - Difference between compiled and interpreted languages? - Stack Overflow. [online].[cit. 2022-03-22]. Dostupné

z: <https://stackoverflow.com/questions/38491212/difference-between-compiled-and-interpreted-languages>

Component Object Model (COM) - Win32 apps | Microsoft Docs.[online].[cit. 2022-03-22]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/com/component-object-model--com--portal

Co je PDF? Portable Document Format | Adobe Acrobat DC.[online].[cit. 2022-03-22]. Dostupné z: <https://www.adobe.com/cz/acrobat/about-adobe-pdf.html>

Core Document Properties — python-docx 0.8.11 documentation.[online].[cit. 2022-03-22]. Dostupné z: <https://python-docx.readthedocs.io/en/latest/dev/analysis/features/coreprops.html#schema-excerpt>

Developing Use Case Diagrams & Use Case Scenarios. [online].[cit. 2022-03-22]. Dostupné z: <https://www.w3computing.com/systemsanalysis/use-case-diagrams-scenarios/>

Difference Between Python Modules, Packages, Libraries, and Frameworks. In: LearnPython.com [online].[cit. 30.03.2022]. Dostupné z: <https://learnpython.com/blog/python-modules-packages-libraries-frameworks/>

DUBLINCORE: DublinCore Metadata Innovation.[online].[cit. 2022-03-22]. Dostupné z: <https://www.dublincore.org/resources/glossary/dublin_core/>

eDiscovery | Deloitte Česká republika. [online]. [cit. 2022-03-22]. Dostupné z: <https://www2.deloitte.com/cz/cs/pages/deloitte-analytics/solutions/ediscovery.html>

eDiscovery Update Email file types and how to handle them > Washtenaw County Legal News. [online].[cit. 2022-02-06]. Dostupné z: <http://legalnews.com/washtenaw/1427534>

Email (Electronic Mail Format). [online]. [cit. 2022-03-22]. Dostupné z: <https://www.loc.gov/preservation/digital/formats/fdd/fdd000388.shtml>

email: Examples — Python 3.10.4 documentation.[online].[cit. 30.03.2022]. Dostupné z: <https://docs.python.org/3/library/email.examples.html>

Engineering SC | The Document Foundation - The House of LibreOffice and Document Liberation Project.[online].[cit. 2022-03-22]. Dostupné z: <https://www.documentfoundation.org/governance/engineering-steering-committee/>

ESI Processing Simplified: What Every Legal Team Should Know.[online].[cit. 2022-03-22]. Dostupné z: <https://blog.specialcounsel.com/ediscovery/esi-processing-simplified-what-every-legal-team-should-know/>

FRANĚK, Zdeněk, OBJEKTOVÉ METODY. MODELOVÁNÍ.VÝKLAD JAZYKA UML, JEHO MOŽNOSTI A. PRAKTICKÉ UKÁZKY.[online].[cit. 2022-03-22]. Dostupné z: <is.slu.cz/el/opf/zima2021/INMNKOMM/2968337/OOM_vyklad_jazyka_UML__jeho_moznosti_a_prakticke_ukazky-2014.pdf>

GANDAL, Neil & Markovich, Sarit & Riordan, Michael. (2013). Ain't it "Suite"? Bundling in the PC Office Software Market. Strategic Management Journal. 39. 10.1002/smj.2797.[online].[cit. 2022-03-22]. Dostupné z: <www.columbia.edu/~mhr21/papers/Suite.pdf>

GAURA, Jan. Studijní opora k předmětu Skriptovací jazyky. Ostrava: Vysoká škola báňská – Technická univerzita Ostrava.[online].[cit. 2022-03-22]. Dostupné z: mrl.cs.vsb.cz/people/gaura/skj/skripta.pdf

GitHub - libyal/libpff: Library and tools to access the Personal Folder File (PFF) and the Offline Folder File (OFF) format. [online].[cit. 30.03.2022]. Dostupné z: <https://github.com/libyal/libpff>

How MS Office Applications Have Changed Over the Years?.[online].[cit. 2022-03-22]. Dostupné z: <https://www.1plus1tech.com/from-ms-office-1-0-to-microsoft-office-365/>

Is RDA (robotic desktop automation) better than RPA (robotic process automation)? In: Quora [online].[cit. 30.03.2022]. Dostupné z: <https://www.quora.com/Is-RDA-robotic-desktop-automation-better-than-RPA-robotic-process-automation>

ISO/IEC 25010:2011. In: ISO [online].[cit. 30.03.2022]. Dostupné z: <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/57/35733.html>

ISO 15836-1:2017(en), Information and documentation — The Dublin Core metadata element set — Part 1: Core elements .[online].[cit. 2022-03-22]. Dostupné z: <https://www.iso.org/obp/ui/fr/#iso:std:iso:15836:-1:ed-1:v1:en>

ISO - ISO 32000-2:2020 - Document management — Portable document format — Part 2: PDF 2.0..[online].[cit. 30.03.2022]. Dostupné z: <https://www.iso.org/standard/75839.html>

ISO/IEC JTC 1 - Information technology. In: ISO [online].[cit. 30.03.2022]. Dostupné z: <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/committee/04/50/45020.html>

JAK ovlivňují „pracovní činnosti" návrat do kancelářských prostor? | AV MEDIA.cz..[online].[cit. 30.03.2022]. Dostupné z: <https://www.avmedia.cz/novinky/firmy/29_4924-jak-ovlivnuji-pracovni-cinnosti-navrat-do-kancelarskych-prostor_firmy>

java - Compiled vs. Interpreted Languages - Stack Overflow. [online].[cit. 2022-03-22]. Dostupné z: <https://stackoverflow.com/questions/3265357/compiled-vs-interpreted-languages?rq=1>

JABLONSKÝ, Josef. Operační výzkum : kvantitativní modely pro ekonomické rozhodování [Jablonský, 2002]. 1. vyd. Praha: Professional Publishing, 2002. 323 s. ISBN 80-86419-42-8.

JANDOVÁ, Radoslava. Algoritmus a jeho vlastnosti [online].[cit. 2021-02-18]. Dostupné z: <http://www.vrstevnice.com/akce/grandaction/vskola/2semestr/ted/yd14ted_clanek_jandora1.pdf>

KANISOVÁ, Hana a Miroslav MÜLLER. UML srozumitelně. Brno: Computer Press, 2004. ISBN 80-251-0231-9.

KOSEK, Jiří. TOC - Interoperabilita datových souborů a formátů.[online].[cit. 2022-03-22]. Dostupné z: <https://www.kosek.cz/xml/2010interop/toc.html>

LAVRINČÍK, Jan. Operační systémy. Olomouc. Moravská vysoká škola Olomouc, o.p.s, 2018.[online].[cit. 2022-03-22]. Dostupné z: :<https://www.mvso.cz/files/operacni-systemy.pdf>

Libraries - The Open Source Discovery Service. In: Libraries.io [online].[cit. 30.03.2022]. Dostupné z: https://libraries.io

LibreOffice Technology | LibreOffice - Free Office Suite - Based on OpenOffice - Compatible with Microsoft.[online].[cit. 30.03.2022]. Dostupné z: <https://www.libreoffice.org/discover/libreoffice-technology/>

LibreOffice Technology | LibreOffice - Svobodný kancelářský balík - Smysluplný projekt - Skvělí lidé.[online].[cit. 30.03.2022]. Dostupné z: <https://cs.libreoffice.org/discover/libreoffice-technology/>

libyal/libpff, 2022.[online].[cit. 30.03.2022]. Dostupné z: <https://github.com/libyal/libpff>

TORCHIANO, Marco. Data Quality Version 1.0.0 Marco Torchiano, 2019 Data Management and Visualization TORCHIANO, Marco. 2019. Data Quality Version 1.0.0.[online].[cit. 2022-03-22]. Dostupné z: <softeng.polito.it/torchiano/DMV/DV04-DataQuality.pdf

Metadata. In: Databáze Národní knihovny ČR, Praha: NK ČR.[online].[cit. 2022-03-22]. Dostupné z: <http://aleph.nkp.cz/F/?func=direct&doc_number=000000543&local_base=KTD>

MAJER, Martin. Řešené úlohy z oblasti zpracování informací: algoritmizace úloh. Zlín: Univerzita Tomáše Bati ve Zlíně, 2021, 91 s. [online].[cit. 2022-03-22]. Dostupné také z: <http://hdl.handle.net/10563/46140>. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, Ústav informatiky a umělé inteligence. Vedoucí práce Perůtka, Karel.

MANTUANO, Fedele. mail-parser: Wrapper for email standard library.[online].[cit. 30.03.2022]. Dostupné z: <https://github.com/SpamScope/mail-parser>

Metadata. In: Databáze Národní knihovny ČR.[online].[cit. 2022-03-22]. Dostupné z: <http://aleph.nkp.cz/F/?func=direct&doc_number=000000543&local_base=KTD>

MICROSOFT: How to create and manage Python environments in Visual Studio.[online].[cit. 2022-03-22]. Dostupné z: <https://docs.microsoft.com/en-us/visualstudio/python/managing-python-environments-in-visual-studio>

MUELLER, John Paul. Embracing the Four Python Programming Styles | New Relic.[online].[cit. 24.03.2022]. Dostupné z: <https://newrelic.com/blog/nerd-life/python-programming-styles>

NIAKANLAHIJI, Amirreza. xlrd2: Library for developers to extract data from Microsoft Excel legacy spreadsheet files (xls).[online].[cit. 30.03.2022]. Dostupné z: <https://pypi.org/project/xlrd2/>

OpenDocument Format (ODF) Family, OASIS and ISO/IEC 26300.[online].[cit. 30.03.2022]. Dostupné z: <https://www.loc.gov/preservation/digital/formats/fdd/fdd000247.shtml>

OpenDocument V1.3 OASIS Standard published - OASIS Open.[online].[cit. 2022-03-22]. Dostupné z: <https://www.oasis-open.org/2021/06/16/opendocument-v1-3-oasis-standard-published/>

openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files — openpyxl 3.0.9 documentation.[online].[cit. 30.03.2022]. Dostupné z: <https://openpyxl.readthedocs.io/en/stable/>

pandas: powerful Python data analysis toolkit [online].[cit. 30.03.2022]. Dostupné z: <https://github.com/pandas-dev/pandas>

PAPÍK, Martin. Hodnocení jakosti software. In: [online]. [cit. 2015-01-18]. Dostupné z: <http://www.kardos.cz/konf/2013/papik_2013.docx>

PEP 8 – Style Guide for Python Code | peps.python.org.[online].[cit. 2022-03-22]. Dostupné z: <https://peps.python.org/pep-0008/>

PETRLÍKOVÁ, K.: Objektově orientované metody návrhu systémů. [Diplomová práce]. Brno 2005. 78 s. Masarykova univerzita. Fakulta informatiky. Vedoucí práce: RNDr. Jaroslav Ráček, Ph.D.

PyPDF2 on Pypi. In: Libraries.io [online].[cit. 30.03.2022]. Dostupné z: <https://libraries.io/pypi/PyPDF2>

PYTHON.ORG.[online].[cit. 2022-03-22]. Dostupné z: <https://www.python.org/downloads/release/python-3102/>

python-pptx — python-pptx 0.6.21 documentation.[online].[cit. 30.03.2022]. Dostupné z: <https://python-pptx.readthedocs.io/en/latest/>;

PILGRIM, Mark. Ponořme se do Python(u) 3 = Dive into Python 3 / Mark Pilgrim. 2010. ISBN 9788090424821.

pip · PyPI. [online].[cit. 2022-03-30]. Dostupné z: <https://pypi.org/project/pip/>

Rešerše: AI/BI/chatbot (2Q 2020) | | Největší katalog ICT řešení [online].[cit. 30.03.2022]. Dostupné z: <https://lepsi-reseni.cz/reserse-ai-bi-chatbot-2q-2020/>

ReportLab Toolkit - ReportLab.com.[online].[cit. 30.03.2022]. Dostupné z: <https://www.reportlab.com/software/opensource/rl-toolkit/>

SAWA, Zdeněk. Ostrava. Technická universita Ostrava 2018.[online].[cit. 2022-03-22] Dostupné z: <www.cs.vsb.cz/kot/download/uti2013/uti-01-cz.pdf>

SKOUPIL, David. ÚVOD DO PARADIGMAT PROGRAMOVÁNÍ. Olomouc: Katedra informatiky, 2018.[online].[cit. 2022-03-22]. Dostupné z: :<https://www.mvso.cz/files/operacni-systemy.pdf>

Soubory vyhovující standardům PDF/X, PDF/A a PDF/E (Adobe Acrobat Pro).[online].[cit. 2022-03-22] Dostupné z: <https://helpx.adobe.com/cz/acrobat/using/pdf-x-pdf-a-pdf.html>

Srovnání kancelářských balíků MS Office, LibreOffice, Google.[online].[cit. 2022-03-22] Dostupné z: <https://www.itnetwork.cz/ms-office/srovnani-kancelarskych-baliku-ms-office-libreoffice-google>

Standardy pro metadata — Národní digitální knihovna.[online].[cit. 2022-03-22] Dostupné z: <https://old.ndk.cz/standardy-digitalizace/metadata>

SUMMERFIELD, Mark. Python 3 : výukový kurz. Computer Press, 2010.[online].[cit. 2022-03-22] Dostupné z: <http://infozdroje.czu.cz/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat02728a&AN=czu.000060551&lang=cs&site=eds-live>

Sustainability of Digital Formats: Planning for Library of Congress Collections.[online].[cit. 2022-03-22] Dostupné z: <https://www.loc.gov/preservation/digital/formats/index.html>

SYNKOVÁ, Veronika. Vybrané aplikace metadatového formátu Dublin Core. Ikaros [online]. 2008, ročník 12, číslo 5.urn:nbn:cz:ik-12779. ISSN 1212-5075.[online].[cit. 2022-03-22] Dostupné z: <http://ikaros.cz/node/12779>

ŠUBRT, Tomáš, 2019. Ekonomicko-matematické metody. 3. upravené a rozšířené vydání. Plzeň: Vydavatelství a nakladatelství Aleš Čeněk. ISBN 978-80-7380-762-7.

TIŠŇOVSKÝ, Pavel. Nástroje pro tvorbu UML diagramů. Root.cz, 2005.[online].[cit. 2022-03-24]. Dostupné z: <https://www.root.cz/clanky/nastroje-pro-tvorbu-uml-diagramu/#k01>

The Selenium Browser Automation Project. In: Selenium [online].[cit. 30.03.2022]. Dostupné z: <https://www.selenium.dev/documentation/>

UML Use Case Diagrams: Guidelines - Visual Studio 2015 | Microsoft Docs. [online].[cit. 2022-03-22]. Dostupné z: <https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2015/modeling/uml-use-case-diagrams-guidelines?view=vs-2015>

VANÍČEK, Jiří. Jak postupovat při hodnocení jakosti softwarových produktů.Česká zemědělská univerzita v Praze, Provozně ekonomická fakulta, Katedra informačního inženýrství.[online].[cit. 2022-02-06]. Dostupné z: <ww.kardos.cz/konf/2006/kvalita%20vanicek.doc>

VANÍČEK, Jiří. Jak postupovat při hodnocení jakosti softwarových produktů [online][cit. 30.03.2022]. Dostupné z: www.kardos.cz/konf/2006/kvalita%20vanicek.doc

VANÍČEK, Jiří. Mezinárodní normalizace kvality softwaru - ppt stáhnout.[online].[cit. 30.03.2022]. Dostupné z: <https://slideplayer.cz/slide/3749831/>

VIOLINO, Bob. Hyperautomatizace jako strategický trend letošního roku - CFOworld.cz. In: CFOworld [online].[cit. 30.03.2022]. Dostupné z: <https://www.cfoworld.cz/clanky/hyperautomatizace-jako-strategicky-trend/>

VIRIUS, Miroslav. Základy algoritmizace. Praha : ČVUT, 1995. 179 s. ISBN 80-01-01346-4.[online].[cit. 2022-03-22]. Dostupné z: www.jaderny-prvak.8u.cz/2Fwp-content/2Fuploads/2F2013/2F02/2FZ/25C3/25A1klady-algoritmizace-skripta-21.pdf

WALKER, Destiny Peterson & Matthew. extract-msg: Extracts emails and attachments saved in Microsoft Outlook's .msg files [online].[cit. 30.03.2022]. Dostupné z: <https://github.com/TeamMsgExtractor/msg-extractor>

Welcome to Radon's documentation! — Radon 4.1.0 documentation. In: [cit. 30.03.2022]. Dostupné z: https://radon.readthedocs.io/en/latest/

What is the demand for the Robot Framework automation tool? - Quora. [online].[cit. 30.03.2022] Dostupné z: <https://www.quora.com/What-is-the-demand-for-the-Robot-Framework-automation-tool>

ZÍSKAL, Jan, 2001. Ekonomicko matematické metody: studijní texty pro distanční studium : určeno pro posluchače PEF. Vyd. 2. Praha: Česká zemědělská univerzita. ISBN 978-80-213-0761-2.

# 8 Seznam obrázků, tabulek, grafů a zkratek

## 8.1 Seznam obrázků



## 8.2 Seznam tabulek

## 8.3 Seznam použitých zkratek

| | |
|---|---|
| AHP | Analytic hierarchy process |
| CASE | Computer Aided Software Engineering |
| CAD | Computer aided design |
| CDN | Content delivery network |
| CI | Consistency index (Míra konzistence) |
| CSV | Comma-separated values |
| COM | Component Object Model |
| DOC | Přípona Word (binární, zastaralá) |
| DOCX | Přípona Word |
| DTO | Digital twin of an organization |
| EML | Electronic mail, přípona souboru „eml“ |
| ERP | Enterprise resource planning |
| FOSS | Free and open-source software |
| IDP | Intelligent document processing |
| IPA | Intelligent process automation |
| MS | Microsoft |
| MSG | Přípona souboru aplikace Microsoft Outlook „msg“ |
| ODF | Přípona otevřeného souborového formátu „.odt“ |
| ODT | OpenDocument Format Standard ODF od OASIS |
| OFF | Offline Folder File aplikace MS Outlook |
| OST | Offline Storage Files aplikace MS Outlook |
| OOo | OpenOffice.org, ukončený kancelářský balík |
| OOXML | Office Open XML |
| PDF | Portable Document Format |
| PPTX | Přípona „.pptx“ aplikace Microsoft Powerpoint |
| PST | Personal Folders File Microsoft Outlook 2003 |
| SQuaRE | Software Quality Requirements and Evaluation |
| RDA | Robotic Desktop Automation |
| RPA | Robotic Process Automation |
| UDF | User-Defined Function, uživatelsky definovaná funkce |
| UML | Unified Modeling Language |

| | |
|---|---|
| QM | Quality Measurement, měření kvality |
| QME | Quality Measure Elements, prvky měření kvality |
| XLS | Přípona „.xls“ aplikace Microsoft Excel (binární, zastaralá) |
| XLSX | Přípona „.xlsx“ aplikace Microsoft Excel, specifikace OOXML |
| XML | Extensible Markup Language, značkovací jazyk konsorcia W3CRPA |
| VBA | Visual Basic for Applications |

# Přílohy

Příloha A Kód pro práci s tabulkovými procesory (spreadsheet)

Příloha B Navržená modifikace modelu jakosti pro hodnocení knihoven pro automatizaci

Příloha C Zvažovaná kritéria relevantní k hodnocení jakosti knihoven Python

Příloha D Pro účely práce modifikované prvky měření kvality (*Quality measure elements*)

Příloha E Vlastní ohodnocení pro AHP v rámci kritéria Dokumentace

Příloha F Velikost a pořadí jednotlivých knihoven v rámci kritéria Velikost

Příloha G Pořadí variant podle metody AHP

Příloha H Párová ohodnocení jednotlivých variant.

Příloha CH Přehled výsledků hodnocení

Příloha I Use Case diagram pro spolupráci s XLSX – vytvoření souboru

Příloha J Demonstrativní Scénář případu užití (*Use Case Scenario*) pro činnost (*task*) *Vytvořit soubor v XLSX*.

Příloha K Všechny hodnocené činnosti v rámci spolupráce s knihovnou openpyxl

Příloha L Některé hodnocené činnosti v rámci spolupráce s knihovnou python-docx

Příloha M Některé hodnocené činnosti v rámci spolupráce s knihovnou PyPDF2

Příloha N Některé hodnocené činnosti v rámci spolupráce s knihovnou python-pptx

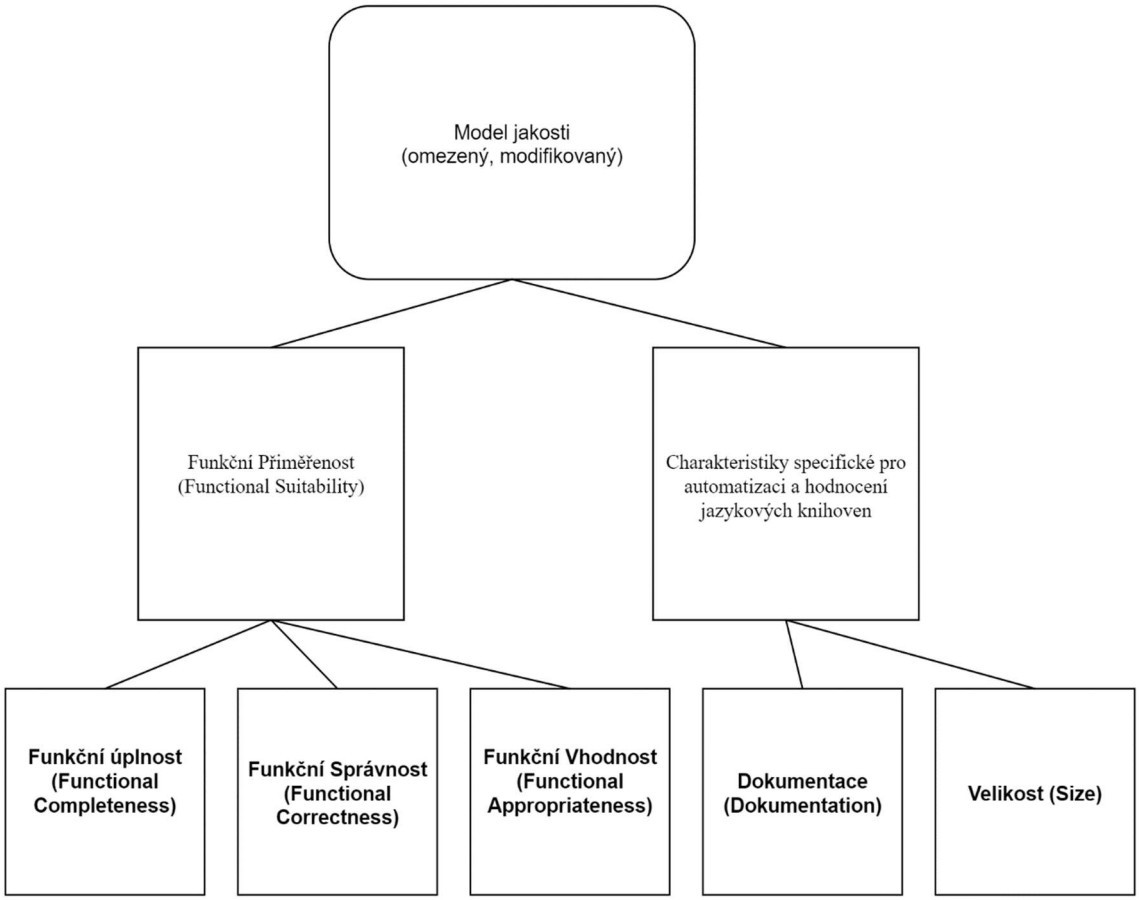
Příloha O Některé hodnocené činnosti v rámci spolupráce se soubory klientů ke správě elektronické pošty

Příloha P  Příklad použití knihovny Openpyxl v podnikové praxi

## Příloha A Vybrané oblasti a zvažované knihovny jazyka Python

| Výběr knihovny | | | Kategorie | Pracuje s formátem |
|---|---|---|---|---|
| 1 | Úkony s XLSX | xlrd | Microsoft Excel | xls |
| 2 | | XlsxWriter | Microsoft Excel | xlsx |
| 3 | | Pandas | Microsoft Excel | xlsx |
| 4 | | **Openpyxl** | Microsoft Excel | xlsx |
| 5 | | formulas | Microsoft Excel | xlsx |
| 6 | Úkony s PDF | **PyPDF2** | Portable Document Format | pdf |
| 7 | | Textract | Portable Document Format | pdf |
| 8 | | PdfPlumber | Portable Document Format | pdf |
| 9 | | pikepdf | Portable Document Format | pdf |
| 10 | | PyMuPDF | Portable Document Format | pdf |
| 11 | Úkony s ostatními vybranými produkty MS Office | **Pypff/libpff** | Microsoft Outlook | pff/off |
| 12 | | independetsoft.msg | Microsoft Outlook | msg |
| 13 | | **python-pptx** | Microsoft PowerPoint | pptx |
| 14 | | pywin32 | - | - |
| 15 | | **Python-docx** | Microsoft Word | docx |

# Příloha B Navržená modifikace modelu jakosti pro hodnocení knihoven pro automatizaci



## Příloha C Zvažovaná kritéria relevantní k hodnocení jakosti knihoven Python

| Číslo | Relevantní kritéria k automatizaci | Stručný popis kritéria |
|---|---|---|
| **1** | Funkční úplnost (větší lepší) | Funkčnost systému splňuje všechny zadané úkoly a cíle uživatele |
| **2** | Funkční přiměřenost (větší lepší) | Systém vrací správné výsledky s určitou mírou přesnosti |
| **3** | Funkční správnost (větší lepší) | Funkce usnadňují splnění stanovených úkolů a cílů. |
| **4** | Časové chování (menší lepší) | Splnění požadavků na dobu odezvy a zpracování |
| **5** | Využití zdrojů (menší lepší) | Potřebné množství a typy prostředků aby se splnil všechny požadavky |
| **6** | Poslední aktualizace knihovny a její stáří (novější lepší) | Aktuálnost knihovny |
| **7** | Stáří knihovny (novější lepší) | Velikost potřebných nově instalovaných balíků |
| **8** | Velikost knihovny a závislostí (menší lepší) | Velikost potřebných nově instalovaných balíků |
| **9** | Pythonicness/Přívětivost/user-friendliness/Radon-repository-scorer (větší lepší) | Míra uspokojující interakce s knihovnou |
| **10** | Rozpoznatelnost vhodnosti | Uživatelé mohou rozpoznat, zda je produkt nebo systém vhodný pro jejich potřeby |

## Příloha D Pro účely práce modifikované prvky měření kvality (*Quality measure elements*)

| Číslo kritéria (měř. prvku) | Charakteristika kvality (Q. characteristic) | Subch. kvality (použité jako kritéria pro výběr varianty) (Quality sub-char.) | Název míry kvality/ (Quality measure name) | Upravené prvky měření kvality (Q. measure elements) | Popis prvku modifikovaného pro použití v rámci práce |
|---|---|---|---|---|---|
| 1 | Funkční přiměřenost (Functional Suitability) | **Funkční úplnost (Functional Completeness)** | Funkční pokrytí (functional coverage) | Procentuální vyjádření úspěšnosti pokrytí požadovaných úkolů/úkonů/zadání (Number of failures) | Umožňuje knihovna splnit všechny vybrané úkoly a cíle uživatele? (Úspěšnost v procentech vybraných úkolů) |
| 2 | | **Funkční Správnost (Functional Correctness)** | Správnost funkcí (Functional correctness) | Míra přesnosti výsledků (Number of failures) | Poskytuje knihovna uspokojivé výsledky s potřebnou mírou přesnosti? |
| 3 | | **Funkční Vhodnost (Functional Appropriateness)** | Funkční vhodnost cíle využití (functional appropriateness of usage objective) | Míra usnadnění splnění stanovených úkolů a cílů (Time) | Poskytuje knihovna usnadnění splnění stanovených úkolů a cílů v poměru k použití jiné knihovny/jiného nástroje/bez nástroje? |
| | | | Funkční vhodnost systému (functional appropriateness of system) | | |
| 4 | Specifické souhrnné charakteristiky pro oblast práce | **Dokumentace a důvěryhodnost knihovny** | Atributy dokumentace a spolehlivosti knihovny podle SourceRank | Vlastní míra (Vlastní souhrnná metrika) | Jsou ke knihovně všechny relevantní informace potřebné k plnění úkolů a cílů, a je knihovna důvěryhodná? |
| 5 | | **Velikost knihovny (Size of library)** | Velikost knihovny (Size of library) | Velikost knihovny (Program Size, kB) | Jaká je velikost knihovny? |

## Příloha E Vlastní ohodnocení pro AHP v rámci kritéria Dokumentace

| Atributy | openpyxl | python-docx | PyPDF2 | python-pptx | libpff-python (pypff) |
|---|---|---|---|---|---|
| Jsou přítomny základní informace? | 1 | 1 | 1 | 1 | 1 |
| Je k dispozici zdrojové úložiště? | 0 | 1 | 1 | 1 | 1 |
| Přítomnost Readme? | 0 | 1 | 1 | 1 | 1 |
| Přítomnost licence? | 1 | 1 | 1 | 1 | 1 |
| Má více verzí? | 1 | 0 | 1 | 1 | 1 |
| Sleduje SemVer? | 0 | 1 | 0 | 0 | 0 |
| Nedávné vydání? | 1 | 1 | 1 | 1 | 1 |
| Není zcela nové? | 1 | 0 | 1 | 1 | 1 |
| 1.0.0 nebo vyšší? | 1 | 4 | 1 | 0 | 0 |
| Závislé balíčky | 6 | 3 | 6 | 4 | 0 |
| Přispěvatelé | 0 | 1 | 1 | 1 | 0 |
| **Mezisoučet** | **12** | **14** | **15** | **12** | **7** |
| Další hodnocené atributy | openpyxl | python-docx | PyPDF2 | python-pptx | libpff-python (pypff) |
| Atribut srozumitelnosti dokumentace | 5 | 5 | 2 | 3 | 3 |
| Atribut úplnosti dokumentace | 5 | 3 | 4 | 2 | 3 |
| Atribut aktuálnosti dokumentace | 3 | 2 | 3 | 2 | 3 |
| Atribut funkčního pokrytí dokumentace | 5 | 3 | 2 | 3 | 4 |
| **Počet získaných bodů celkem** | 30 | 27 | 26 | 22 | 20 |
| **Výsledné pořadí** | **1** | **2** | **3** | **4** | **5** |

**Příloha F Velikost a pořadí jednotlivých knihoven v rámci kritéria Velikost**

| | Název knihovny | Size (KB) | Pořadí (size) |
|---|---|---|---|
| **Library 1** | openpyxl | 1574.687 | **4** |
| **Library 2** | Python-docx | 1053.478 | **2** |
| **Library 3** | PyPDF2 | 392.788 | **1** |
| **Library 4** | python-pptx | 1878.339 | **5** |
| **Library 5** | pypff/ libpff | 1236.992 | **3** |

**Příloha G Pořadí variant podle metody AHP**

| Pořadí variant podle metody AHP | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Funkční úplnost | Funkční správnost | Funkční vhodnost | Dokumentace | Velikost knihovny | **Syntéza preferencí** | **Pořadí** |
| Váha | 0.533547932 | 0.224988949 | 0.163067415 | 0.05343788 | 0.024957824 | 1 | x |
| **openpyxl** | 0.501066879 | 0.308994236 | 0.070081576 | 0.308994236 | 0.510038725 | 0.330685953 | **1** |
| **Python-docx** | 0.246083269 | 0.308994236 | 0.27164735 | 0.308994236 | 0.263833779 | 0.329669521 | **2** |
| **PyPDF2** | 0.103769526 | 0.075808844 | 0.490967357 | 0.075808844 | 0.129573679 | 0.154394154 | **3** |
| **python-pptx** | 0.103769526 | 0.274723542 | 0.033892648 | 0.274723542 | 0.063636045 | 0.131325514 | **4** |
| **pypff** | 0.045310799 | 0.031479141 | 0.133411069 | 0.031479141 | 0.032917772 | 0.053924857 | **5** |

# Příloha H Párová ohodnocení jednotlivých variant

| **Kritérium 1** | Popis | Pár | Vztahy mezi kritérii |
|---|---|---|---|
| **Kritérium 1** | Funkční úplnost | I_s (CI) | 0.031722792 |
| | Knihovna 1 je slabě preferována před knihovnou 2 | K1xK2 | 3 |
| | Knihovna 1 je silně preferována před knihovnou 3 | K1xK3 | 5 |
| | Knihovna 1 je silně preferována před knihovnou 4 | K1xK4 | 5 |
| | Knihovna 1 je velmi silně preferována před knihovnou 5 | K1xK5 | 7 |
| | Knihovna 2 je slabě preferována před knihovnou 3 | K2xK3 | 3 |
| | Knihovna 2 je slabě preferována před knihovnou 4 | K2xK4 | 3 |
| | Knihovna 2 je silně preferována před knihovnou 5 | K2xK5 | 5 |
| | Knihovna 3 je rovnocenná s knihovnou 4 | K3xK4 | 1 |
| | Knihovna 3 je slabě preferována před knihovnou 5 | K3xK5 | 3 |
| | Knihovna 4 je slabě preferována před knihovnou 5 | K4xK5 | 3 |
| **Kritéria 2 - 3** | Funkční správnost + Funkční vhodnost | **Pár** | **Vztahy mezi kritérii** |
| | | I_s (CI) | 0.056621001 |
| | Knihovna 1 je rovnocenná s knihovnou 2 | K1xK2 | 1 |
| | Knihovna 1 je silně preferována před knihovnou 3 | K1xK3 | 5 |
| | Knihovna 1 je rovnocenná s knihovnou 4 | K1xK4 | 1 |
| | Knihovna 1 je absolutně preferována před knihovnou 5 | K1xK5 | 9 |
| | Knihovna 2 je silně preferována před knihovnou 3 | K2xK3 | 5 |
| | Knihovna 2 je rovnocenná s knihovnou 4 | K2xK4 | 1 |
| | Knihovna 2 je absolutně preferována před knihovnou 5 | K2xK5 | 9 |
| | Knihovna 3 je silně preferována před knihovnou 5 | K3xK5 | 5 |
| | Knihovna 4 je silně preferována před knihovnou 3 | K4xK3 | 5 |
| | Knihovna 4 je silně preferována před knihovnou 5 | K4xK5 | 5 |
| **Kritérium 4** | Dokumentace | **Pár** | **Vztahy mezi kritérii** |
| | | I_s (CI) | 0.059368813 |
| | Knihovna 1 je slabě preferována před knihovnou 2 | K1xK2 | 3 |
| | Knihovna 1 je silně preferována před knihovnou 3 | K1xK3 | 5 |
| | Knihovna 1 je velmi silně preferována před knihovnou 4 | K1xK4 | 7 |
| | Knihovna 1 je absolutně preferována před knihovnou 5 | K1xK5 | 9 |
| | Knihovna 2 je slabě preferována před knihovnou 3 | K2xK3 | 3 |
| | Knihovna 2 je silně preferována před knihovnou 4 | K2xK4 | 5 |
| | Knihovna 2 je velmi silně preferována před knihovnou 5 | K2xK5 | 7 |
| | Knihovna 3 je silně preferována před knihovnou 5 | K3xK5 | 5 |
| | Knihovna 3 je slabě preferována před knihovnou 4 | K3xK4 | 3 |
| | Knihovna 4 je slabě preferována před knihovnou 5 | K4xK5 | 3 |
| **Kritérium 5** | Velikost knihovny | **Pár** | **Vztahy mezi kritérii** |
| | | I_s (CI) | 0.067399366 |
| | Knihovna 1 je slabě preferována před knihovnou 4 | K1xK4 | 3 |
| | Knihovna 2 je silně preferována před knihovnou 1 | K2xK1 | 5 |
| | Knihovna 2 je velmi silně preferována před knihovnou 4 | K2xK4 | 7 |
| | Knihovna 2 je slabě preferována před knihovnou 5 | K2xK5 | 3 |
| | Knihovna 3 je velmi silně preferována před knihovnou 1 | K3xK1 | 5 |
| | Knihovna 3 je slabě preferována před knihovnou 2 | K3xK2 | 3 |
| | Knihovna 3 je absolutně preferována před knihovnou 4 | K3xK4 | 9 |
| | Knihovna 3 je silně preferována před knihovnou 5 | K3xK5 | 5 |
| | Knihovna 5 je slabě preferována před knihovnou 1 | K5xK1 | 3 |
| | Knihovna 5 je silně preferována před knihovnou 4 | K5xK4 | 5 |

## Příloha CH Přehled výsledků hodnocení

Váhy stanovené Saatyho metodou pro jednotlivá kritéria

| 0 | **Váhy stanovené Saatyho metodou** | | Kritéria | |
|---|---|---|---|---|

| | **K1** | **K2** | **K3** | **K4** | **K5** | $b_j$ | $v_j$ |
|---|---|---|---|---|---|---|---|
| **K1** | **1** | 5 | 5 | 7 | 9 | 4.359695 | 0.533548 |
| **K2** | 0.2 | **1** | 3 | 5 | 7 | 1.838416 | 0.224989 |
| **K3** | 0.2 | 0.3333333 | **1** | 7 | 9 | 1.332447 | 0.163067 |
| **K4** | 0.142857 | 0.2 | 0.111111 | **1** | 5 | 0.436648 | 0.053438 |
| **K5** | 0.111111 | 0.1428571 | 0.111111 | 0.2 | **1** | 0.203934 | 0.024958 |
| | | | | | **Σ $b_i$** | 8.17114 | 1 |
| **Váha kriteria** | | | | | | | |

| | | |
|---|---|---|
| | **l_max** | **5.782616** |
| C.I. | **I_s** | 0.005234 |

Saatyho matice pro kritérium Funkční Úplnost (Functional Completeness)

| 1 | **Saatyho matice pro kritérium** | Funkční úplnost |
|---|---|---|

| 0.533547932 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $b_{ij}$ | $v_{ij}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 3 | 5 | 5 | 7 | 3.499708 | 0.501067 |
| Library 2 | 0.333333 | **1** | 3 | 3 | 5 | 1.718772 | 0.246083 |
| Library 3 | 0.2 | 0.3333333 | **1** | 1 | 3 | 0.72478 | 0.10377 |
| Library 4 | 0.2 | 0.3333333 | 1 | **1** | 3 | 0.72478 | 0.10377 |
| Library 5 | 0.142857 | 0.2 | 0.333333 | 0.333333 | **1** | 0.316474 | 0.045311 |
| | | | | | **Σ $b_i$** | 6.984514 | 1 |

Saatyho matice pro kritérium Funkční Správnost (Functional Correctness)

| 2 | **Saatyho matice pro kritérium** | Funkční správnost |
|---|---|---|

| 0.2249889 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $b_{ij}$ | $v_{ij}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 1 | 5 | 1 | 9 | 2.1411274 | 0.308994 |
| Library 2 | 1 | **1** | 5 | 1 | 9 | 2.1411274 | 0.308994 |
| Library 3 | 0.2 | 0.2 | **1** | 0.2 | 5 | 0.5253056 | 0.075809 |
| Library 4 | 1 | 1 | 5 | **1** | 5 | 1.9036539 | 0.274724 |
| Library 5 | 0.111111 | 0.111111 | 0.2 | 0.2 | **1** | 0.2181298 | 0.031479 |
| | | | | | **Σ $b_i$** | 6.929344 | 1 |

Saatyho matice pro kritérium Funkční Vhodnost (F. Appropriateness)

| 3 | **Saatyho matice pro kritérium** | Funkční vhodnost |
|---|---|---|

| 0.163067415 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $\mathbf{b_{ij}}$ | $\mathbf{v_{ij}}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 1 | 5 | 1 | 9 | 2.141127 | 0.308994 |
| Library 2 | 1 | **1** | 5 | 1 | 9 | 2.141127 | 0.308994 |
| Library 3 | 0.2 | 0.2 | **1** | 0.2 | 5 | 0.525306 | 0.075809 |
| Library 4 | 1 | 1 | 5 | **1** | 5 | 1.903654 | 0.274724 |
| Library 5 | 0.111111 | 0.1111111 | 0.2 | 0.2 | **1** | 0.21813 | 0.031479 |
| | | | | | **Σ $\mathbf{b_i}$** | 6.929344 | 1 |

Saatyho matice pro kritérium Dokumentace (Dokumentation)

| 4 | **Saatyho matice pro kritérium** | Dokumentace |
|---|---|---|

| 0.0534379 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $\mathbf{b_{ij}}$ | $\mathbf{v_{ij}}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 3 | 5 | 7 | 9 | 3.9362834 | 0.510039 |
| Library 2 | 0.333333 | **1** | 3 | 5 | 7 | 2.036168 | 0.263834 |
| Library 3 | 0.2 | 0.333333 | **1** | 3 | 5 | 1 | 0.129574 |
| Library 4 | 0.142857 | 0.2 | 0.333333 | **1** | 3 | 0.4911186 | 0.063636 |
| Library 5 | 0.111111 | 0.142857 | 0.2 | 0.333333333 | **1** | 0.2540467 | 0.032918 |
| | | | | | **Σ $\mathbf{b_i}$** | 7.7176168 | 1 |

Saatyho matice pro kritérium Velikost (Size)

| 5 | **Saatyho matice pro kritérium** | Velikost knihovny |
|---|---|---|

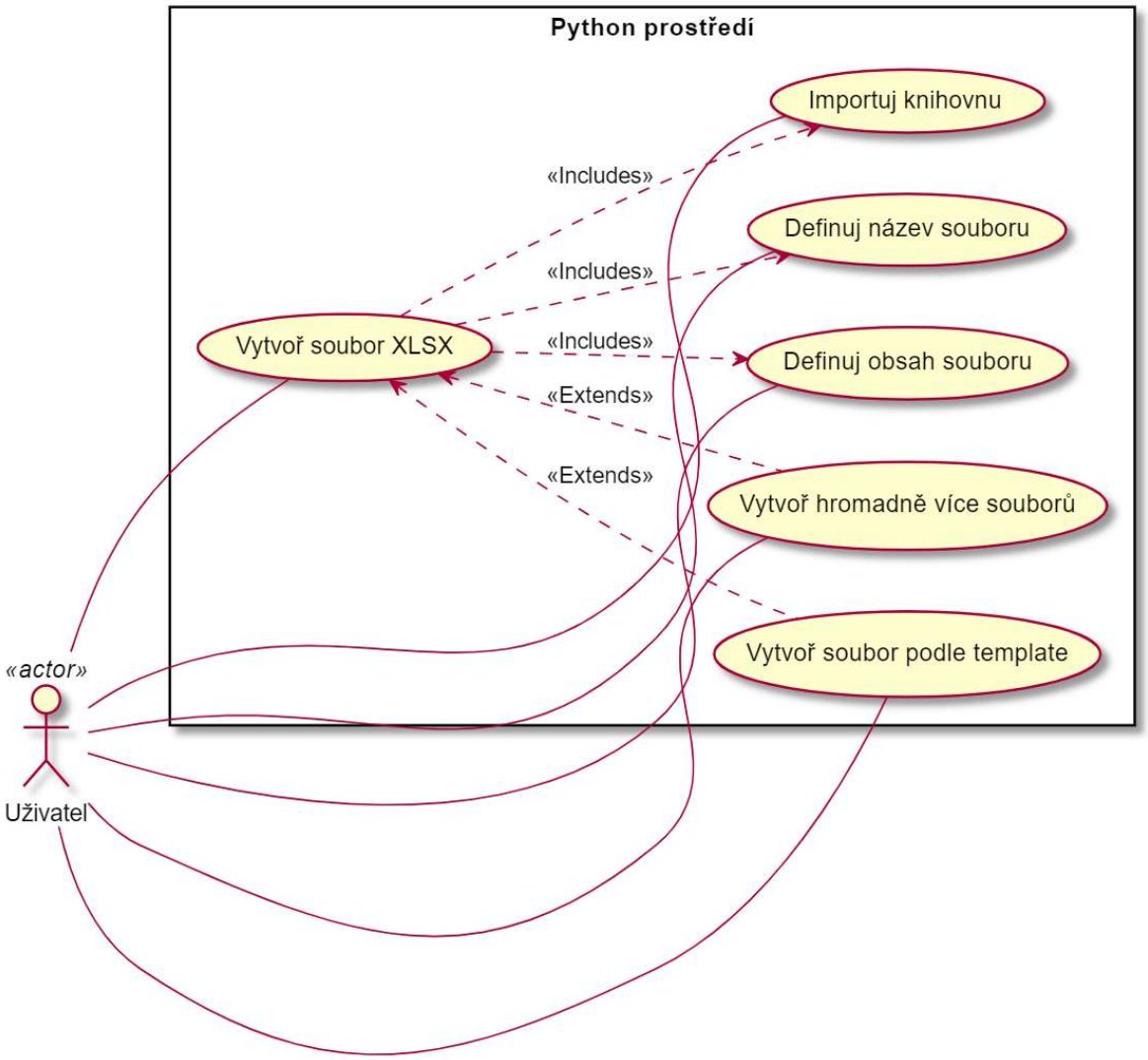
| 0.024957824 | Library 1 | Library 2 | Library 3 | Library 4 | Library 5 | $\mathbf{b_{ij}}$ | $\mathbf{v_{ij}}$ |
|---|---|---|---|---|---|---|---|
| Library 1 | **1** | 0.2 | 0.2 | 3 | 0.333333 | 0.525306 | 0.070082 |
| Library 2 | 5 | **1** | 0.333333 | 7 | 3 | 2.036168 | 0.271647 |
| Library 3 | 5 | 3 | **1** | 9 | 5 | 3.68011 | 0.490967 |
| Library 4 | 0.333333 | 0.1428571 | 0.111111 | **1** | 0.2 | 0.254047 | 0.033893 |
| Library 5 | 3 | 0.3333333 | 0.2 | 5 | **1** | 1 | 0.133411 |
| | | | | | **Σ $\mathbf{b_i}$** | 7.49563 | 1 |

Pořadí variant

| 6 | **Pořadí variant podle metody AHP** | | |
|---|---|---|---|

| 1 | Funkční úplnost | Funkční správnost | Funkční vhodnost | Dokumentace | Velikost knihovny | | $\mathbf{b_j}$ | $\mathbf{v_j}$ | **Pořadí** |
|---|---|---|---|---|---|---|---|---|---|
| **Váha** | 0.533548 | 0.224989 | 0.163067 | 0.05343788 | 0.024958 | **1** | G.M. | Synt. pref. | **x** |
| openpyxl | **0.501067** | **0.308994** | **0.070082** | 0.308994236 | 0.510039 | 1.6991757 | 0.279641 | 0.330686 | 1 |
| Python-docx | **0.246083** | **0.308994** | **0.271647** | 0.308994236 | 0.263834 | 1.3995529 | 0.278781 | 0.32967 | 2 |
| PyPDF2 | **0.10377** | **0.075809** | **0.490967** | 0.075808844 | 0.129574 | 0.8759283 | 0.130562 | 0.154394 | 3 |
| python- | **0.10377** | **0.274724** | **0.033893** | 0.274723542 | 0.063636 | 0.7507453 | 0.111054 | 0.131326 | 4 |
| pypff | **0.045311** | **0.031479** | **0.133411** | 0.031479141 | 0.032918 | 0.2745979 | 0.045601 | 0.053925 | 5 |
| | | | | | | 5 | 0.845639 | 1 | |

**Příloha I Use Case diagram pro spolupráci s XLSX – vytvoření souboru**



**Příloha J Demonstrativní Scénář případu užití (*Use Case Scenario*) pro činnost (*task*) *Vytvořit soubor v XLSX*.**

| *Případ užití* | Vytvořit soubor v XLSX |
|---|---|
| *ID* | 1 |
| *Stručný popis* | Vytvořit soubor daného formátu v požadovaném souborovém formátu s obsahem bez formátování. |
| *Hlavní aktéři* | Uživatel |
| *Vstupní podmínky* | Uživatel má aktivované virtuální prostředí s nainstalovanou knihovnou. Uživatel importoval knihovnu. Uživatel má v objektu uložený textový řetězec. |
| *Hlavní scénář* | 1. Aktivace virtuálního prostředí<br>2. Import knihovny<br>3. Iniciace vkládaného obsahu<br>4. Iniciace pracovního sešitu do objektu ps kontejnerem Workbook()<br>5. Kontejner Workbook() v objektu ps<br>6. Definice počátečního sloupce a řádku<br>7. Zápis do buněk pro cyklus v rozsahu 0 až délka vkládaných dat<br>8. Pojmenování souboru<br>9. Uložení kontejneru do souboru |
| *Výstupní podmínky* | Soubor s požadovaným formátem s požadovaným textem je vytvořen. |
| *Alternativní scénáře* | Chyba vytvoření. |
| | |

## Příloha K Všechny hodnocené činnosti v rámci spolupráce s knihovnou openpyxl

Tvorba dokumentu

```
#! Python 3
# -*- coding: utf-8 -*-

## Vytvorit novy soubor [1]
## MS Excel

from openpyxl import Workbook

nazev_souboru = "NazevSouboru"
polozky = ['A','B','C','D']
sloupec_start = 1  # začátek dat ve sloupci A
radek_start = 3  # začátek dat pod záhlavím řádku 2
jmeno_listu = "List" # pojmenování listu

def VytvoritSoubor ():

  """Zjednodušený příklad funkce pro vytvoření souboru ve formátu XLSX
  knihovnou Openpyxl"""

  ps = Workbook() # není nutné vytvořit soubor v souborovém systému
  ps['Sheet'].title = jmeno_listu ## Prejmenuje vychozi nazev souboru
  pl1 = ps[jmeno_listu] # Označeni pracovniho list souboru s kterým se pracuje
  for i in range(0,len(polozky)): # V rozsahu poctu polozek
      pl1.cell(radek_start+i, sloupec_start).value = polozky[i]
  ps.save(nazev_souboru + '.xlsx')

VytvoritSoubor()
```

Extrakce dat

```
#! Python 3
# -*- coding: utf-8 -*-

## Extrakce dat [2]
## MS Excel

from openpyxl import load_workbook

nazev_souboru = f"NazevSouboru.xlsx"

def ExtrahovatData ():

    """Zjednodušený příklad funkce pro extrahování obsahu ze souboru ve formátu
    XLSX knihovnou Openpyxl"""

    ps = load_workbook(nazev_souboru, data_only=True) # Načtení celého sešitu.
    # Vypíše názvy všech listů v souboru.
    for jmeno_listu in ps.sheetnames:
        print(f"V sešitu je list s názvem {jmeno_listu}")
    # Načtení jednoho pracovního listu - podle specifikovaného jména listu
    ws = ps['List']
    vsechny_radky = list(ws.rows) # uloží obsah všech řádků do listu

    print(f"V sešitu je obsah do {len(vsechny_radky)} řádku") # Vypíše délku
    obsahu řádků
    for row in vsechny_radky[0:6]:
        obsah_radku = row[0].value
        print( f"\n {obsah_radku}" )

ExtrahovatData ()
```

Manipulace s obsahem

```
#! Python 3
# -*- coding: utf-8 -*-

## Manipulace s obsahem [3]
## MS Excel

from openpyxl import load_workbook
from openpyxl.styles import Font

nazev_puvodniho_souboru = f"NazevSouboru.xlsx"
nazev_modifikovaneho_souboru = "NazevModifikovanehoSouboru"
barva_font_bunky = "FF0000" # v případě modifikace Font(name='Arial', size=14)
souradnice_modifikovane_bunky="A3"


def ZmenitObsah ():

  """Zjednodušený příklad funkce pro modifikaci obsahu souboru ve formátu XLSX
  knihovnou Openpyxl"""

  wb = load_workbook(nazev_puvodniho_souboru)
  ws = wb.active
  modifikace_bunky = ws[souradnice_modifikovane_bunky] # přiřadí k buňce
  font_zmena = Font(color=barva_font_bunky) # přiřadí k fontu nebo barvě
  modifikace_bunky.font = font_zmena
  wb.save(nazev_modifikovaneho_souboru + '.xlsx')

ZmenitObsah()
```

Manipulace se souborem

```
#! Python 3
# -*- coding: utf-8 -*-

## Manipulace se souborem [4]
## MS Excel

from openpyxl import load_workbook, Workbook

nazev_souboru = f"NazevSouboru.xlsx"

def UpravSoubor ():

    """Zjednodušený příklad funkce pro manipulaci se souborem formátu XLSX v
    souborovém systému, rozumí se spojit nebo rozdělit - ukázka převodu souboru
    s více pracovními listy na více souborů"""

    ps = load_workbook(nazev_souboru, data_only=True) # Načtení celého sešitu.
    for sheet in ps.worksheets:
        novy_ps = Workbook()
        ps = novy_ps.active
        for row_data in sheet.iter_rows():
            for row_cell in row_data:
                ps[row_cell.coordinate].value = row_cell.value

        novy_ps.save('{0}.xlsx'.format(sheet.title))

UpravSoubor ()
```

Metadata

```
#! Python 3
# -*- coding: utf-8 -*-

## Práce s metadaty  [5]
## MS Excel

from openpyxl import load_workbook

nazev_souboru = f"NazevSouboru.xlsx"
metadata_nazev_souboru = "DP-nazev-souboru"
metadata_autor_souboru = "DP-autor-souboru"

def UpravMetadata ():

    """Zjednodušený příklad funkce pro manipulaci s metadaty souboru formátu
    XLSX - ukázka změny vlastností název souboru a autor souboru"""

    ps = load_workbook(nazev_souboru, data_only=True)
    obj = ps.properties   # Získání původních vlastností
    # print (obj)   # Pro zobrazení původních vlastností
    ps.properties.title = metadata_nazev_souboru  # Nastavi jmeno (title)
    ps.properties.creator = metadata_autor_souboru  # autora souboru (author)
    new_obj = ps.properties   # Nastavi nove vlastnosti
    # print (new_obj)   # Zobrazi nove vlastnosti
    ps.save(nazev_souboru)

UpravMetadata()
```

## Příloha L Některé hodnocené činnosti v rámci spolupráce s knihovnou python-docx

Tvorba dokumentu a manipulace s obsahem

```
#! Python 3
# -*- coding: utf-8 -*-

## Vytvorit novy soubor [1] a příklad použití manipulace s obsahem [3]
## MS Word

from datetime import datetime
from docx import Document
from docx.shared import Pt, RGBColor

nazev_souboru = "NazevSouboru"
obsah_pridaneho_odstavce = "Vložený text do souboru formátu DOCX aplikace MS
WORD - do odstavce (paragraph)"

def VytvoritSoubor ():

  """Zjednodušený příklad funkce pro vytvoření souboru ve formátu DOCX
  knihovnou python-docx a manipulace s obsahem v rámci paragrafu tohoto
  dokumentu"""

  dokument = Document()
  p1 = dokument.add_paragraph(obsah_pridaneho_odstavce)
  p1.add_run(' MS WORD ').bold = True # Přidat další text - font tučně
  eg = p1.add_run('Příklad změny barvy fontu')
  eg.font.size = Pt(18) # Změna velikosti písma
  eg.font.color.rgb = RGBColor(150,125,10) # Změna barvy písma
  dokument.save(nazev_souboru + '.docx')

VytvoritSoubor ()
```

Extrakce dat

```
#! Python 3
# -*- coding: utf-8 -*-

## Extrakce dat [2]
# ## MS Word

import docx

nazev_souboru = f"NazevSouboru.docx"

def ExtrahovatData ():

  """Zjednodušený příklad funkce pro vytvoření souboru ve formátu DOCX
  knihovnou python-docx"""

  dokument = Document(nazev_souboru)
  ObsahText = []
  for docpara in dokument.paragraphs:
    ObsahText.append(docpara.text)
  print(docpara.text)

ExtrahovatData ()
```

Metadata

```
#! Python 3
# -*- coding: utf-8 -*-

## Práce s metadaty [5]
## MS Word

from docx import Document

nazev_souboru = f"NazevSouboru.docx"
metadata_autor_souboru = "DP-autor-souboru"

def UpravMetadata ():

  """Zjednodušený příklad funkce pro úpravu metadat souboru ve formátu DOCX -
  ukázka změny vlastností autor souboru"""

  doc = Document(nazev_souboru)
  doc.core_properties.author = (metadata_autor_souboru)
  doc.save(nazev_souboru)

UpravMetadata()
```

## Příloha M Některé hodnocené činnosti v rámci spolupráce s knihovnou PyPDF2

Tvorba dokumentu (knihovna Reportlab)

```
#! Python 3
# -*- coding: utf-8 -*-

## Vytvorit novy soubor [1]
## PDF

from reportlab.lib.pagesizes import A4
from reportlab.lib.units import cm
from reportlab.pdfgen.canvas import Canvas

nazev_souboru = "NazevSouboru"

def VytvoritSoubor ():

    """Zjednodušený příklad funkce pro vytvoření souboru v PDF formátu
    knihovnou Reportlab"""

    canvas = Canvas(nazev_souboru + '.pdf', pagesize=A4)
    canvas.setFont("Times-Roman", 20)
    canvas.drawString(10 * cm, 5 * cm, "Obsah souboru - Diplomová práce")
    canvas.save()

VytvoritSoubor()
```

Načtení obsahu souboru a metadat ve formátu PDF (PyPDF2)

```
#! Python 3
# -*- coding: utf-8 -*-

## Práce s metadaty [5]
## PDF

from PyPDF2 import PdfFileReader

nazev_souboru = f"NazevSouboru.pdf"


def metadata_souboru(path):

    """Zjednodušený příklad pro čtení metadat souboru knihovnou PyPDF2 """

    with open(path, 'rb') as f:
        pdf = PdfFileReader(f)
        info = pdf.getDocumentInfo()
        number_of_pages = pdf.getNumPages()

    print(info)
    author = info.author
    creator = info.creator
    producer = info.producer
    subject = info.subject
    title = info.title

if __name__ == '__main__':
    path = nazev_souboru
    metadata_souboru(path)
```

Manipulace s obsahem

```
#! Python 3
# -*- coding: utf-8 -*-

## Manipulace s obsahem [3]
## PDF

from PyPDF2 import PdfFileWriter

nazev_souboru = f"NazevSouboru.pdf"


def UpravSoubor ():

      """Zjednodušený příklad funkce pro modifikaci obsahu souboru ve formátu
      PDF - přidání prázdných stran"""

      file = PdfFileWriter()
      file.addBlankPage(width= 100,height= 200)
      file.addBlankPage(width=60,height=50)
      output = open(nazev_souboru, 'wb')
      file.write(output)

UpravSoubor ()
```

**Příloha N Některé hodnocené činnosti v rámci spolupráce s knihovnou python-pptx**

Tvorba dokumentu

```
from pptx import Presentation

## Vytvorit novy soubor [1]
# ## MS Powerpoint

nazev_souboru = "NazevSouboru"

def VytvoritSoubor ():

  """Zjednodušený příklad funkce pro vytvoření souboru ve formátu PPTX
  knihovnou python-pptx"""

  prs = Presentation()
  title_slide_layout = prs.slide_layouts[0]
  slide = prs.slides.add_slide(title_slide_layout)
  title = slide.shapes.title
  subtitle = slide.placeholders[1]

  title.text = "Text - nadpis"
  subtitle.text = "Diplomová práce"
  prs.save(nazev_souboru + '.pptx')

VytvoritSoubor ()
```

✓ 0.1s

Extrakce dat

```
#! Python 3
# -*- coding: utf-8 -*-

## Extrakce dat [2]
# ## MS Powerpoint


from pptx import Presentation
import glob

def ExtrahovatData ():

    """Zjednodušený příklad funkce pro extrahování obsahu ze souboru formátu
    PPTX knihovnou python-pptx"""

    for eachfile in glob.glob("*.pptx"):
        prs = Presentation(eachfile)
        print(eachfile)
        print("----------------------")
        for slide in prs.slides:
            for shape in slide.shapes:
                if hasattr(shape, "text"):
                    print(shape.text)

ExtrahovatData ()
```

**Příloha O Některé hodnocené činnosti v rámci spolupráce se soubory klientů ke správě elektronické pošty**

Tvorba dokumentu (formát MSG, knihovna email)

```
import email
from email.message import EmailMessage
from email.parser import BytesParser

## Vytvorit novy soubor [1]
# Klient el. pošty - formát msg

msg = EmailMessage()
msg['Subject'] = "Subject"
msg['From'] = ("Odesílatel")
msg['To'] = ("Příjemce"),
msg.set_content(
"""\
obsah zprávy
""")

# Vytvořit místní kopii
with open('nazev_souboru.msg', 'wb') as f:
    f.write(bytes(msg))

# Načtení místní kopie
with open('nazev_souboru.msg', 'rb') as fp:
    loaded_msg = BytesParser(policy=policy.default).parse(fp)
```

Extrakce dat a metadat (MSG, knihovna extract_msg)

```
#! Python 3
# -*- coding: utf-8 -*-

## Extrakce dat [2] Práce s metadaty  [5]
# ## MS Outlook (msg)

import extract_msg

nazev_souboru = f"NazevSouboru.msg"

with extract_msg.Message(nazev_souboru) as msg:
    msg_sender = msg.sender
    msg_to = msg.to
    print(msg_sender, msg_to)
```

## Příloha P Příklad použití knihovny Openpyxl v podnikové praxi

```
#! Python 3
# -*- coding: utf-8 -*-

## Pouziti v podnikove praxi
## MS Excel

import openpyxl
import os
import shutil

# Soubor obsahujici data nazvu souboru
nazvy_souboru = openpyxl.load_workbook('Polozky.xlsx') # Pridat nazev souboru
nazvy_souboru_list = nazvy_souboru['List1'] # Pridat nazev listu
# Uchopit soubor Sablona
Sablona = 'Sablona.xlsx'
# Nazev nove slozky
slozka = 'NazevNoveSlozky'

# Parametry: start bunka, end bunka, list z ktereho kopirovat.
def RozsahKopirovani(startCol, startRow, endCol, endRow, sheet):
    RozsahVybrany = []
    # Prochazeni vybranymi radky
    for i in range(startRow,endRow + 1,1):
        # Pripoji radek do vybraneho seznamu
        RadekVybrany = []
        for j in range(startCol,endCol+1,1):
            RadekVybrany.append(sheet.cell(row = i, column = j).value)
        # Prida seznam vybranych radku a vnori do RozsahVybrany
        RozsahVybrany.append(RadekVybrany)
    return RozsahVybrany

def VytvoritSoubory():
    # Vytvori slozku pro soubory
    aktualni_adresar = os.getcwd()
    slozna_nov_cesta = os.path.join(aktualni_adresar,slozka)
    try:
        NovaSlozka = os.makedirs(slozna_nov_cesta)
    except:
        print("Slozka jiz existuje")
        return
    # Ziskani dat pro vytvoreni nazvu souboru
    VybranyRozsah = RozsahKopirovani(1,2,2,11,nazvy_souboru_list)
    # Projde ve smycce kazdy radek
    for i in VybranyRozsah:
        nazev_souboru = i[0]+" "+i[1]+" automatizace.xlsx"
        cela_cesta_souboru = os.path.join(slozka,nazev_souboru)
        shutil.copy(Sablona, cela_cesta_souboru)
    # Spojeni cesty s novym nazvem souboru
# VytvoritSoubory()
```