Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Aplikování SAP Scriptingu k automatizování uživatelských procesů v ERP systému SAP

Využití SAP GUI Scripting API v administrativních činnostech

Bakalářská práce

Autor: Bohuslav Martin Sirůček
Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Hana Rohrová

Hradec Králové Březen 2020

Prohlášení:

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

V Hradci Králové dne 6. 3. 2020 Bohuslav Martin Sirůček

Poděkování:

Děkuji vedoucí bakalářské práce Mgr. Haně Rohrové za metodické vedení, trpělivost a ochotu, kterou mi v průběhu zpracování práce věnovala.

Anotace

Bakalářská práce se zabývá automatizací administrativních činností za pomocí SAP GUI Scripting API. Jsou zde vysvětleny možnosti programování, omezení a základní bezpečnostní rizika spojená se SAP Scriptingem. Informace v této bakalářské práci mohou využít pracovníci IT oddělení, vedoucí pracovníci administrativních oddělení, kteří ovládají základy programování a v neposlední řadě všichni, kteří mají zájem o automatizování procesů v informačním systému SAP. V teoretické části jsou popsány různé způsoby a nástroje, které se používají k automatizaci administrativních procesů ve firmách. Pro praktickou část, ve které je naprogramován ukázkový skript, bylo vybráno konkrétní administrativní oddělení s dvaceti pracovníky. Dále byly vytvořeny časové testy, ve kterých se porovnávala rychlost zpracování vybraných činností manuálně a za pomocí skriptu. V závěrečné části této bakalářské práce jsou vyhodnoceny přínosy a případná rizika týkající se automatizace různých administrativních činností ve firmě.

Annotation

Title: Application of SAP Scripting to automate user processes in SAP ERP system

The bachelor thesis deals with automation of administrative activities using SAP GUI Scripting API. There are explained programming possibilities, limitations and basic security risks associated with SAP Scripting. Information in this bachelor thesis can be used by IT staff, administrative staff, who know the basics of programming, and last but not least, everyone who is interested in automating processes in the SAP information system. The theoretical part describes various ways and tools that are used to automate administrative processes in companies. For the practical part, in which the sample skript is programmed, a specific administrative department with twenty employees in the company that deals with the processing of measured data from electricity meters was selected. Furthermore, time tests were created in which the speed of processing of selected activities was compared manually and with the help of a skript. In the final part of this thesis are evaluated the benefits and potential risks related to the automation of various administrative activities in the company.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Automatizace, robotizace jsou slova, která jsou v poslední době stále více slyšet ve všech možných odvětvích lidské činnosti. Tato bakalářská práce se věnuje softwarovým robotům, konkrétně automatizaci procesů v ERP systému SAP za pomocí SAP GUI Scripting API.

Co si vlastně představit pod pojmem softwarová automatizace, nazývaná jako RPA (Robotic Proces Automation)? Jsou to programy, které snižují nebo zcela eliminují rutinní práci administrativních pracovníků s počítačem. Je to vlastně virtuální pracovník, neboť tyto nástroje pracují s běžným uživatelským prostředím, standardními aplikacemi jako je Excel, internetový prohlížeč, Outlook a v neposlední řadě s informačním systémem, který je ve firmě používán.

Jaké procesy je vhodné automatizovat? Jsou to procesy, které jsou rutinní, mají jasná pravidla, podle kterých je možno se rozhodovat, stále se opakují. Ale samozřejmě je možné automatizovat i například tvorbu smluv, procesy spojené s personalistikou podniku, účetnictvím a dalšími činnostmi.

Softwarovou automatizaci lze rozdělit do dvou základních skupin. V první skupině je integrace automatizace přes aplikační vrstvy, která vyžaduje kompletní systémovou integraci a většinou je zaměřena na velké strategické projekty. Doba integrace takového systému se počítá v řádu měsíců až let v závislosti na složitosti procesů. Tento proces je velmi nákladný, neboť je zapotřebí zkušených softwarových architektů s dlouholetou praxí v programování, znalostí podnikových systémů a databází.

V druhé skupině je automatizace pomocí prezentační vrstvy systémů, které se ve firmě používají. Nejsou potřeba žádná zvláštní zabezpečení ani přístupy, protože se využívá standardních přístupů a práv, která má každý uživatel nastavena ke konkrétní aplikaci, kterou používá. Jedná se o software, který je nainstalován na běžném pracovním počítači a využívá uživatelské rozhraní ostatních systémů stejně jako člověk. Nemusí se tedy nijak upravovat stávající programy. Pro lepší představu to lze přirovnat k záznamu maker v Excelu, neboť většina těchto programů pracuje na stejném principu. Uživatel si v robotické aplikaci vytvoří nový projekt a pak spustí záznam činnosti například přepisování

řádků z Excelu do formuláře podnikového systému. Po dokončení činností uživatel záznam zastaví a projekt uloží. Tímto způsobem je zautomatizována jednoduchá administrativní činnost. Na trhu je dnes již několik firem, které nabízejí tento typ software, jako například firma Winautomation (https://www.winautomation.com) nebo firma UiPath (https://www.uipath.com).

V této bakalářské práci je vysvětleno, jakým způsobem lze automatizovat administrativní procesy v podnikovém systému SAP pomocí SAP GUI Scripting API. Není tedy potřeba žádný jiný software, pouze znalosti programování a znalosti procesů, které chceme automatizovat.

2 Cíl práce

Cílem této práce je ukázat možnosti, jak efektivně automatizovat rutinní administrativní činnosti v informačním systému SAP pomocí SAP GUI Scripting API a porovnat časové úspory při zpracování dat skriptem oproti manuálnímu zpracování uživatelem. Budou zde vysvětleny a popsány nejvíce používané příkazy SAP GUI Scripting API, jejich omezení a ošetření výjimek. Na vytvořeném ukázkovém skriptu bude předvedeno automatizování procesu ve vybraném administrativním oddělení. Metody zde vysvětlené se budou moci použít pro automatizování mnoha dalších administrativních procesů v SAP. Dále zde budou rozebrány bezpečnostní rizika při používání softwarových robotů v korporátním firemním prostředí a jejich možné následky.

3 Metodika zpracování

Bakalářská práce je založena na vlastních dlouholetých praktických zkušenostech s automatizováním a zjednodušováním procesů ve firemním korporátním prostředí. O problematice SAP Scriptingu není doposud v českém jazyce napsána žádná kniha, ani české zdroje na internetu nejsou k dispozici. Znalosti a doporučení jsou zde čerpány z dokumentace [1] a oficiálních stránek SAP [3].

Jedna z otázek, kterou je potřeba si na začátku položit je, proč vlastně automatizovat nějaký administrativní proces, když vše funguje? To platí hlavně pro korporátní společnosti, ve kterých je nemalé množství procesů rozčleněno do samostatných oddělení. Je také důležité se zeptat na možnosti úprav již jednou vytvořeného automatizovaného procesu a nesmíme také opomenout samotnou bezpečnost používání softwarového robota. Tyto otázky jsou stěžejní u procesů, které je třeba často měnit na základě vnitropodnikových nařízení a závazných směrnic. Aby bylo možno odpovědět na tyto otázky, byl vybrán konkrétní podnikový proces, který byl pomocí SAP GUI Scripting API automatizován. Následně byly vyhodnoceny časové úspory používáním robotizovaného procesu po dobu jednoho měsíce.

4 Podnikový informační systém SAP

Informační systém SAP je produktem stejnojmenné německé firmy se sídlem ve Walldorfu, která byla založena již v roce 1972. Zaměstnává více než 98 000 zaměstnanců ve 140 zemích. Na základě tržní kapitalizace je SAP třetím největším světovým výrobcem softwaru a má více než 437 000 zákazníků ve 180 zemích. Údaje jsou z roku 2019 a jsou převzaty z oficiálních stránek SAP [2].

Zkratka SAP znamená Systems, Applications, Products in data processing a jde o systém ERP (Enterprise Resource Planning) neboli plánování podnikových zdrojů. Samotný systém se skládá z několika funkčních modulů:

- **PP** – Production Planning (plánování výroby)
  Používá se pro plánování a řízení výrobních činností podniku, jako jsou požadavky na materiál, plánování kapacit, pohyby materiálu. Modul je plně integrován s dalšími moduly SAP.
- **MM** – Material Management (správa materiálu)
  Podporuje funkce nákupu zásob, které jsou v každodenních obchodních operacích. Modul obsahuje funkcionality pro příjem zboží, skladování materiálu, plánování založené na spotřebě materiálu.
- **FICO** – Financial & Controlling (finance a kontrola)
  Modul zaznamenává a zpracovává finanční transakce v reálném čase, aby poskytl požadované vstupy pro externí účely vykazování (finanční a daňová správa).
- **SD** – Sales & Distribution (prodej a distribuce)
  Modul určený pro logistiku, který slouží pro vyřizování prodejních objednávek, fakturační procesy, distribuci zásilek zákazníkům.
- **HR** – Human Resources (lidské zdroje)
  Modul pro kompletní řízení životního cyklu zaměstnanců, výplatu mezd, personální rozvoj. Zaznamenává veškerá data od přijetí zaměstnance až po ukončení jeho pracovního poměru v organizaci.

SAP ERP není jediným softwarovým produktem společnosti SAP. Další aplikace, které společnost SAP nabízí jsou například:

- **SAP CRM** – Customer Relationship Management (řízení vztahů se zákazníky)

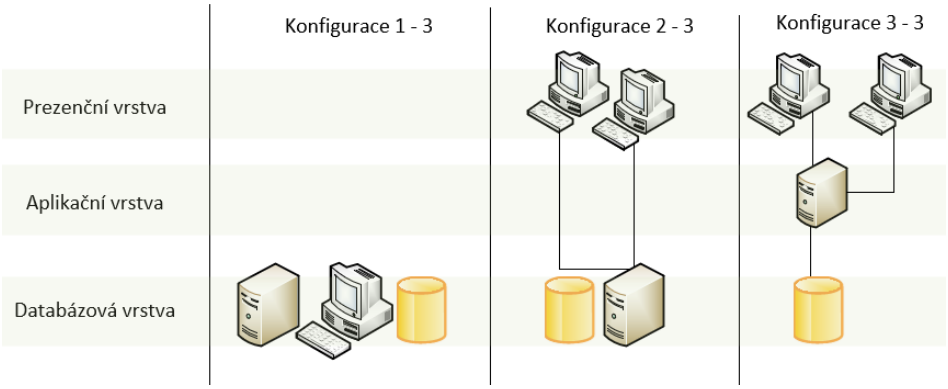
  Software, který automatizuje a integruje činnosti zaměřené na zákazníka, jako je prodej, marketing, zákaznický servis. Nabízí nástroje pro analýzu zákazníků od historie kontaktů, až po aktivitu na sociálních sítích.
- **SAP BI** - Business Intelligence

  Je aplikace, která se zaměřuje především na to, aby svým uživatelům poskytovala uživatelsky přívětivou a smysluplnou formu reprezentace dat, která by mohla být užitečná pro účely analýzy a obchodní rozhodování.

Všechny produkty, které firma SAP nabízí lze vyhledat na oficiálních stránkách [3].

4.1 Architektura systému SAP ERP

Informace v této kapitole byly čerpány z výukového webu Tuturialspoint [4] a oficiálních stránek společnosti SAP v sekci SAP BusinessObjects [5]. Systém SAP je nejčastěji strukturován do třívrstvé architektury, která umožňuje vysokou flexibilitu a škálovatelnost. V některých případech se používá struktura jedno nebo dvouvrstvá a to většinou v malých firmách, kde se nepředpokládá rozšiřování systému. V třívrstvé architektuře SAP, poskytuje databázová vrstva úroveň ukládání obchodních dat, aplikační vrstva zpracovává obchodní logiku a prezentační vrstva poskytuje rozhraní pro uživatele. Jednotlivé vrstvy znázorňuje obrázek 1.



Obrázek 1 Architektura vrstev systému SAP. Zdroj: Vlastní

Prezentační vrstva je většinou umístěna na počítačích uživatelů a poskytuje grafické rozhraní SAP (SAP GUI). SAP GUI je aplikace, kterou lze nainstalovat na jakýkoli počítač se systémem MS Windows nebo Mac OS a poskytuje rozhraní pro komunikaci mezi systémem SAP ERP a uživatelem. Tato vrstva poskytuje i API pro SAP GUI Scripting.

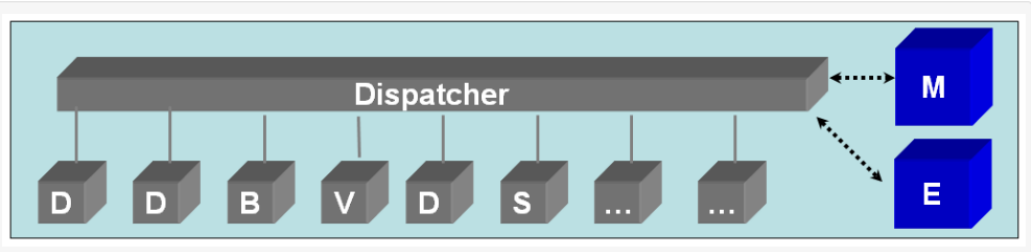
Aplikační vrstva provádí obchodní logiku, zodpovídá za zpracování klientských transakcí, spouštění sestav, koordinaci přístupu k databázi, řídí tiskové úlohy a propojení s jinými aplikacemi. Aplikační vrstva je v podstatě jádrem systému SAP ERP. Aplikační logiku je možné distribuovat mezi několik serverových počítačů v situacích, kdy zátěž překračuje výpočetní výkon jednoho serveru.

Databázová vrstva slouží pro ukládání dvou typů objektů: podnikově generovaná data a aplikační programy SAP. Obchodní data představují datové objekty vytvořené uživateli systému v rámci různých podnikových procesů. Například kmenové záznamy zákazníků nebo prodejní záznamy. Aplikační programy SAP jsou rutiny psané v programovacím jazyku ABAP, které jsou načteny do aplikačních serverů SAP z databáze za běhu systému. Je možné použít databáze od různých dodavatelů jako je například Microsoft nebo Oracle. Výkon databázové vrstvy určuje, jaká bude škálovatelnost celé instalace SAP ERP, protože obvykle každý systém SAP je nasazen v jedné instanci databáze.

Jak je uvedeno v článku [5] jádro systému SAP tvoří soubor spustitelných programů a utilit pro zpracování obchodní logiky. Veškeré procesy jádra, které jsou spuštěny nebo zastaveny, se nazývají instance SAP. Každá instance SAP obsahuje dispečera (obrázek 2) a několik pracovních procesů. Dispečer předává úkoly do jednoho z pracovních procesů. Systém SAP má několik druhů pracovních procesů, které jsou vytvořeny pro různé úkoly. Mezi těmito instancemi musí být jedna speciální instance zvaná centrální instance, která má dvě komponenty a to:

- **Message proces** (**M**)
  slouží k navázání komunikace mezi různými instancemi v systému SAP. Příkladem je přihlášení uživatele do systému SAP, kterému message proces automaticky přiřadí jednu z dostupných instancí aplikace. Dále pak budou všechny transakční požadavky týkající se tohoto uživatele přesměrovány na vybranou instanci.

- **Enqueue server (E)**
  používá se pro správu zámků v databázových tabulkách. Tyto zámky zajišťují, že aktualizace databáze jsou prováděny korektně a ve správném pořadí. Enqueue server zaručuje konzistenci kmenových dat.



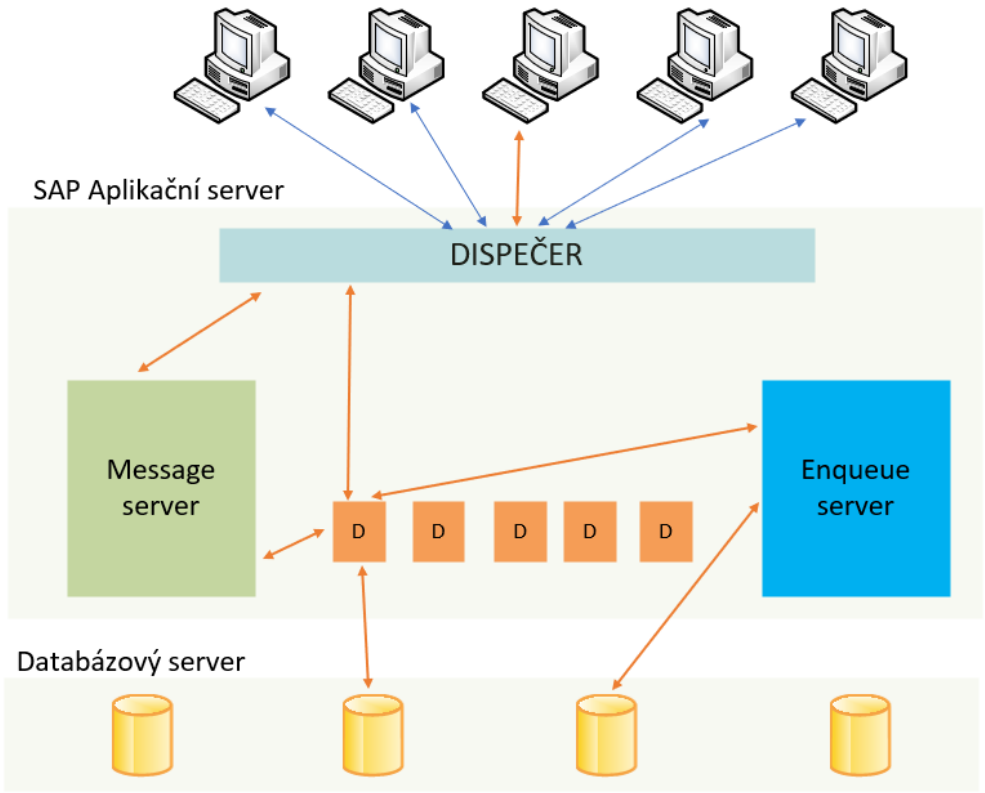
Obrázek 2 Instance SAP, znázornění dispečera s procesy M a E. Zdroj: [7]

Přehled pracovních procesů systému SAP:

- **D**- Dialog work
  Dialogové pracovní procesy jsou zodpovědné za zpracování online transakčních požadavků uživatelů.
- **B** - Batch work
  Dávkové pracovní procesy jsou zodpovědné za zpracování naplánovaných úloh na pozadí v systému SAP.
- **V** - Update work
  Aktualizační pracovní procesy jsou odpovědné za provádění aktualizací v databázi.
- **S** - Spool work
  Jsou procesy zodpovídající za tiskové služby v systému SAP.
- **G** - Gateway work
  Procesy odpovědné za umožnění komunikace mezi aplikacemi.

Pracovní procesy systému SAP (obrázek 3) fungují následujícím způsobem. Jako první přichází žádost, požadavek od uživatele z prezenční vrstvy. Tento požadavek je zpracován dispečerem centrální instance SAP, který předá požadavek procesu zpracování zpráv (M). Proces zpráv (M) rozhodne, zda se tento požadavek zpracuje v této instanci nebo bude předán do jiné instance. Pak je proces zpracován jedním

z pracovních procesů příslušného typu, a pokud je to nutné, tak systém SAP provede aktualizaci databáze prostřednictvím serveru enqueue (E). Posledním krokem je zpětná informace o výsledku žádosti, která jde v obráceném pořadí zpět k iniciátoru žádosti.



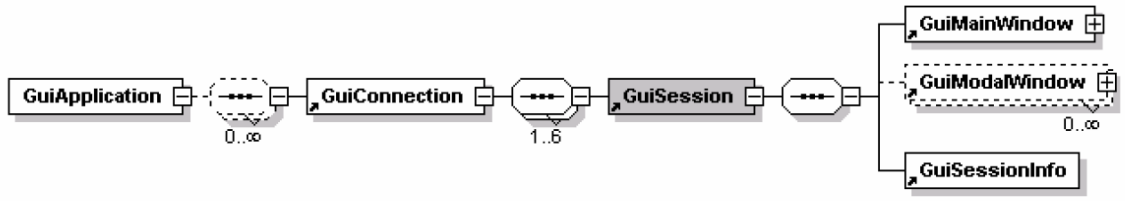
Obrázek 3 Znázornění procesu požadavku v SAP. Zdroj: Vlastní

4.2 SAP GUI Scripting API

Podpora skriptování je k dispozici pro SAP R / 3 3.1I, 4.0B, 4.5B, 4.6B, 4.6C, 4.6D a pro všechny produkty založené na novějších verzích SAP. Poslední verze je 6.0 z roku 2006. Od této doby se používá systém upgrade SAP Enhancement Packages (EhP). SAP EhP nevyžaduje klasickou aktualizaci systému, jedná se pouze o technickou instalaci balíčku, který obsahuje vylepšení systému a aktivaci nových funkcí. Nejnovějším vylepšeným balíčkem SAP pro SAP ERP 6.0 byl EhP8 SPS12 z roku 2019, informace je ze stránek SAP [6]. Ve výchozím nastavení je SAP GUI Scripting z bezpečnostních důvodů zakázáno v každém nainstalovaném systému SAP. Správce musí podporu skriptování povolit. Více je o nastavení skriptování popsáno v kapitole 4.6.1 Oprávnění k používání skriptu – logování uživatelů

SAP GUI API je automatizační rozhraní, které lze použít k provedení jakýchkoliv činností za pomocí skriptu, které by jinak provedl uživatel ručně při práci s informačním systémem SAP. Není žádný rozdíl mezi komunikací SAP GUI generovanou skriptem a SAP GUI generovanou uživatelem. Je to rozhraní, kterým lze ovládat veškeré prvky SAP, které uživatel vidí na obrazovce. Jsou to tlačítka, vstupní pole, rozevírací seznamy, checkboxy, radiobuttony, tabulky. Dále je možno získávat hodnoty z popisků, modálních oken a dalších ovládacích prvků. Tímto rozhraním lze vytvářet nástroje pro testování aplikací na straně serveru nebo testovat integraci nových aplikací na straně klienta. Skript má stejná práva pro spouštění transakcí SAP a zadávání dat jako uživatel, který je přihlášený do informačního systému. Pro ověřování dat zadaných skriptem se používají v systému stejná pravidla, jako pro data, které zadává uživatel ručně.

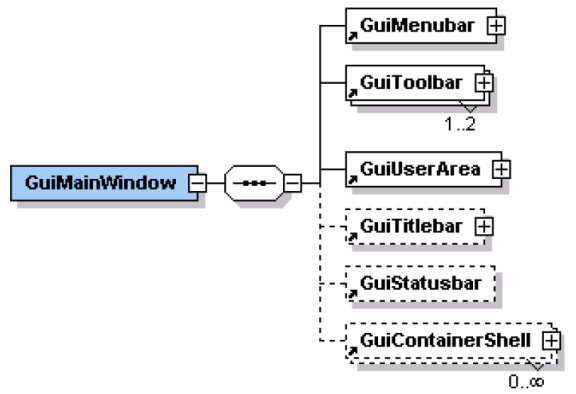
Na obrázku 4 je objektové schéma, které představuje základní objekty, ve kterých je spuštěno skriptování. Objekt GuiApplication představuje proces, ve kterém probíhá činnost systému SAP. Je to hlavní proces, ke kterému se připojují jednotliví uživatelé. GuiConnection zde představuje objekt samotného uživatele, který je přihlášen do systému SAP pomocí uživatelského jména a hesla. Každý přihlášený uživatel může vytvořit 6 samostatných relací neboli otevřených oken systému SAP. Na obrázku je to reprezentováno objektem GuiSession, který představuje konkrétní prováděnou úlohu, v SAPu nazývanou transakce.



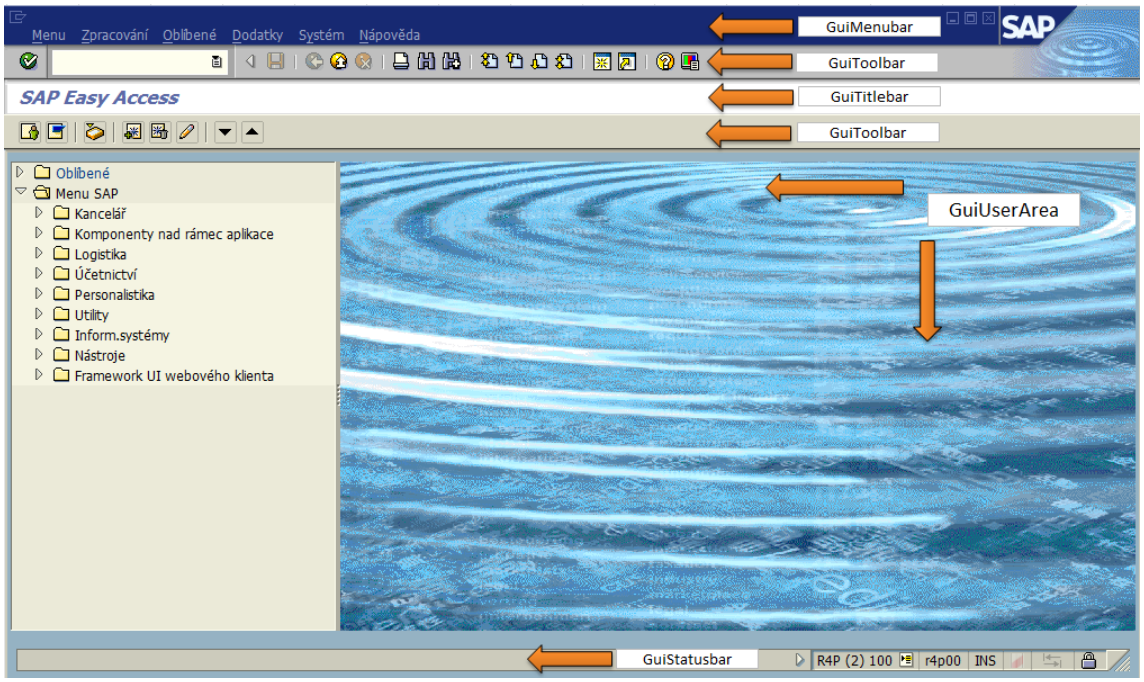
Obrázek 4 Objektové schéma základních procesů SAP. Zdroj: [1]

4.2.1 Objekty uživatelského rozhraní

Každá relace GuiSession má objekt GuiMainWindow, který je reprezentován grafickým rozraním (obrazovkou) se kterým pracuje konkrétní uživatel. Toto rozhraní je rozděleno do několika samostatných objektů, jak je znázorněno na obrázku 5. Na obrázku 6 je vidět, jaký konkrétní objekt odpovídá části obrazovky, kterou vidí uživatel.



Obrázek 5 Schéma objektů uživatelského rozhraní. Zdroj:[1]



Obrázek 6 Grafické znázornění objektů v GUI. Zdroj: Vlastní.

GuiManubar je objekt, ve kterém jsou implementovány metody pro ovládání rozevíracích nabídek aplikace. Objekt GuiToolbar implementuje metody, které slouží k ovládání položek na panelu nástrojů, na kterém jsou umístěna tlačítka, ikony, vstupní prvky. GuiTitlebar reprezentuje objekt, který slouží pro zobrazování názvu transakce (GuiSession), ve které uživatel aktuálně pracuje. Podobným objektem je GuiStatusbar, který zobrazuje různá hlášení při aktivitách uživatele. Hlavním objektem je GuiUserArea, ve které jsou implementovány metody pro práci s formulářovými prvky, jako jsou checkboxy, radibuttony, inputboxy, comboboxy, tlačítka, tabulky, scrollbary, popisky.

4.3 Administrativní procesy

Pojem administrativní procesy si lze představit jako kancelářské činnosti, které vykonávají zaměstnanci v rámci fungování organizací, institucí, firem. Příkladem takovýchto činností může být kontrola, příjem, vystavení faktur, objednávek. Přepisování dat z jednoho systému do druhého, propočty, analýzy, vytváření různých přehledů, reportů.

Proces se dá zjednodušeně charakterizovat jako sled logických činností, které vykonává uživatel tak, aby mohl dojít k předem určenému výsledku zadaného úkolu. Každý proces má na svém začátku určité vstupy (data), které vyhodnotí a zpracuje podle předepsaných postupů (metodik) tak, aby na konci celého procesního řetězce byl požadovaný výstup, který lze použít jako vstup pro další procesy. Aby bylo možno automatizovat nejenom administrativní proces, je nutno ho umět přesně popsat. Pro vizuální znázornění podnikových procesů existuje několik možností jako je například BPMN (BusinessProcess Model and Notation) nebo využití vývojových diagramů. Pro návrh programu se mě nejlépe osvědčilo, vytvořit si vývojový diagram, do kterého celý proces rozkreslíme. Vizuální pohled na proces zobrazený pomocí vývojového diagramu lépe umožní navrhnout strukturu programu. Tato bakalářská práce je zaměřena na rutinní administrativní procesy a v následující kapitole je popsáno, jak byly konkrétní procesy vybrány.

4.3.1 Výběr vhodných administrativních procesů k zautomatizování

Pro tuto bakalářskou práci bylo vybráno oddělení na základě podnikového požadavku pro optimalizaci firemních procesů. V tomto oddělení je dvacet administrativních pracovníků, kteří každý den zpracovávají data z programu Excel, informačního systému SAP a dalších podnikových systémů. Jedná se o rutinní činnosti, které jsou vzhledem k množství zpracovávaných údajů náchylné na chyby při přepisování položek z jednoho systému do druhého. Zároveň u takového množství dat nastává problém s kapacitním plánováním práce v době dovolených, zkrácených pracovních úvazků nebo onemocnění pracovníků. Většina procesů je ve firmách vázaná na termíny dodavatelů a v nemalém případě i na termíny legislativní, což je hlavně příklad korporací, které dodávají energie a podléhají

státním regulacím. Vzhledem k těmto okolnostem nelze některé procesy z důvodu nedostatečných pracovních kapacit časově posouvat, neboť by v důsledku nesplnění legislativních požadavků, mohla firma dostat finanční pokutu. To je případ i tohoto konkrétního oddělení.

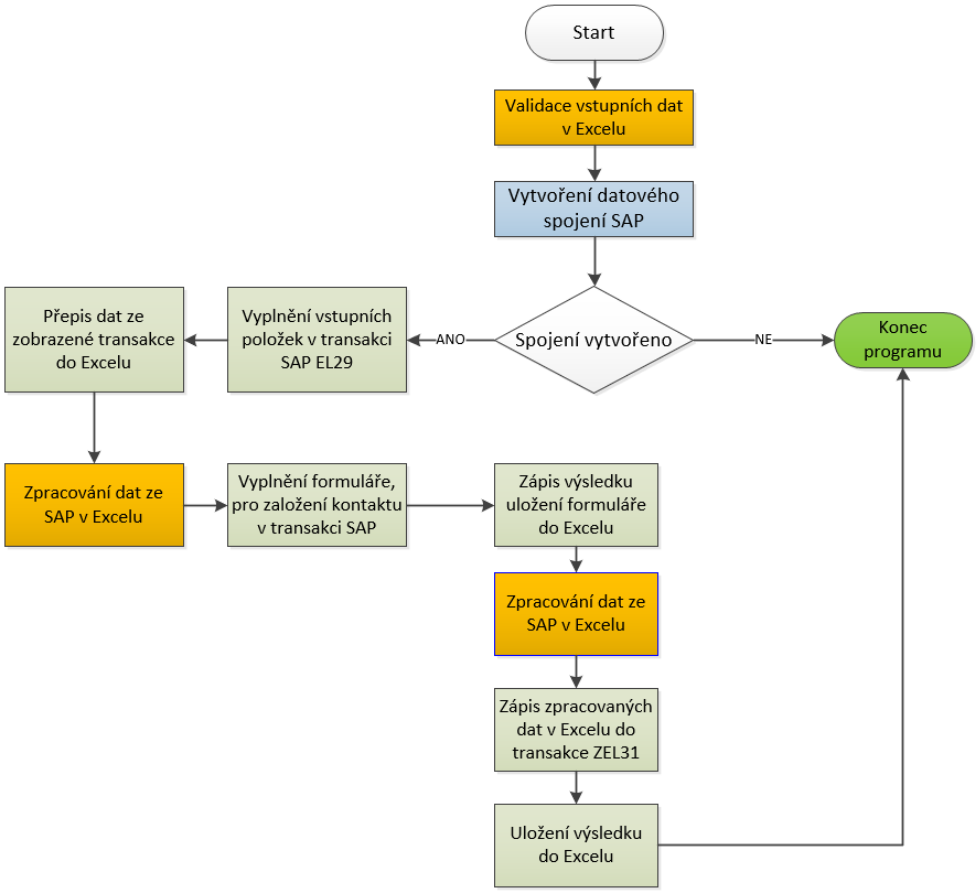
První, co je potřeba udělat, je zmapovat všechny procesy na oddělení a rozdělit si je podle pracovních priorit. To znamená vybrat činnosti, které jsou potřeba vykonat co nejdříve a jejich zpracování je podmínkou pro zahájení následujících procesů. Nemá smysl automatizovat činnosti, které se v celkovém pracovním procesu vyskytují v jednotkách a nejsou podmíněny časovým termínem zpracování, neboť to na celkovou optimalizaci nemá vliv a není to efektivní vzhledem k naprogramování a ošetření veškerých výjimek v programu. Pokud máme vybrané procesy, které by bylo vhodné automatizovat, musíme zjistit, zda je vůbec možné automatizaci provést se současnými prostředky. V našem případě se 90 % procesů vykonává s informačním systémem SAP a programem Excel. Po důkladné analýze a konzultacích s vedoucím pracovníkem oddělení, byly vybrány tři procesy, které se automatizovaly pomocí SAP GUI Scripting API. V této bakalářské práci je vysvětlena automatizace jednoho konkrétního procesu, na kterém jsou rozebrány všechny potřebné příkazy SAP Scriptingu, které byly použity pro automatizování i ostatních procesů. Kompletní skripty včetně komentářů, jsou součástí přílohy na CD.

4.3.2 Popis procesu pomocí vývojového diagramu

Autorka Jana Pšeničková v knize Algoritmizace [8] definuje vývojový diagram následovně: „*Vývojový diagram je symbolický algoritmický jazyk, který se používá pro názorné zobrazení algoritmu, a je to jedna z nejdokonalejších forem zápisu algoritmů. Používá se zejména při vývoji softwaru jako komunikační prostředek při týmové spolupráci analytiků s programátory. Využívá se k dokumentačním účelům, vývojový diagram je přehlednější než zápis programu. Vývojové diagramy se skládají z jednotlivých symbolů, jež jsou mezi sebou spojeny orientovanými čarami, které jsou označeny šipkou vyjadřující směr postupu algoritmu. Obecně vžitý postup psaní značek je odshora dolů a zleva doprava. V některých případech tento postup nemůže být dodržen (například v cyklech se čára vrací o několik kroků zpět, nebo je potřeba*

dostat se z konce stránky na začátek z levého sloupce do pravého). V těchto případech je šipka nezbytně nutná.“

Obrázek 7 představuje vývojový diagram pro proces, který byl vybrán pro tuto bakalářskou práci a který byl automatizován pomocí SAP GUI Scripting API. Jedná se o činnost, při které administrativní pracovníci dostanou přidělená data v Excelu a mají za úkol pomocí tří transakcí v SAP, tato data zpracovat. Denně se v tomto procesu zpracovává 500 řádků z Excelu. Pokud nenastane problém s daty nebo informačním systémem, tak tato činnost trvá souhrnně v průměru 10 hodin. Popis tohoto procesu, proč a za jakým účelem se to dělá není obsahem této bakalářské práce, která je zaměřena na to, jak tyto činnosti automatizovat pomocí SAP GUI Scripting API.



Obrázek 7 Vývojový diagram procesu zpracování dat. Zdroj: Vlastní

4.4 Návrh programu

Vzhledem ke skutečnosti, že budeme pracovat s aplikací Excel, je celý program vytvořen v programovacím jazyku VBA (Visual Basic for Applications). Je to nejjednodušší způsob, jak rychle a efektivně vytvořit automatizovaný proces, neboť není potřeba žádných doplňkových knihoven a plně si vystačíme s VBA, který je součástí Excelu a je podporován samotným API Scripting v SAP.

Jedna část programu se týká samotného připojení na rozhraní SAP ze sešitu Excel. K tomuto účelu slouží čtyři příkazy, které nám vytvoří datové spojení s aplikací Excel a informačním systémem SAP (ve vývojovém diagram znázorněno jako Vytvoření datového spojení SAP). Aby spojení proběhlo korektně, musí být uživatel již přihlášen do informačního systému SAP. Samozřejmě, že jde udělat i automatické přihlášení při spuštění skriptu, ale vzhledem k vnitřním předpisům podniků a zásad bezpečného vývoje, je to příliš vysoké bezpečnostní riziko, a to hlavně z důvodu, že by v samotném kódu skriptu, bylo uvedeno jméno a heslo, což je neakceptovatelné.

Část programu, který vytvoří datové spojení mezi aplikací Excel a SAP:

```
Global session
Sub spojeniSAP()
    Dim rotEntry As Object
    Dim application As Object
    Dim connection As Object

    Set rotEntry = GetObject("SAPGUI")
    Set application = rotEntry.GetScriptingEngine
    Set connection = application.Children(0)
    Set session = connection.Children(0)
End Sub
```

Syntaxe tohoto kódu je převzata z dokumentace [1] a upravena pro VBA.

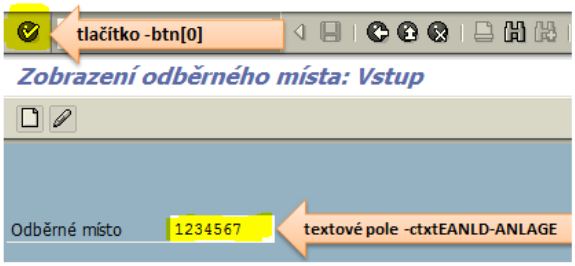
Objekt session je v programu definován jako globální objekt, aby s ním mohly pracovat samostatné funkce, do kterých program rozdělíme.

Další část programu je již o samotné logice zpracování vstupních dat z Excelu za pomocí skriptingu v informačním systému SAP. Jednotlivé části

programu odpovídají navrženému vývojovému diagramu a jsou popsány v kapitole 4.4.5 Vnitřní logika programu.

První, co je potřeba vysvětlit, je způsob, jakým se položky z Excelu posílají do SAP. Slouží k tomu příkaz, který je složen z několika částí. První část příkazu je námi vytvořený datový objekt „**session**“, který byl vytvořen v předchozí části programu. Je to spojení mezi naším programem a aplikací SAP. Na tento příkaz navazuje metoda „**findById**“, která slouží k vyhledání konkrétního prvku v transakci SAP ve které je uživatel právě přihlášen. Jako jednoduchý příklad zde uvádím příkaz, který vyplní konkrétní textové pole ve formuláři SAP (obrázek 8) hodnotou 1234567 a poté dalším příkazem stiskne tlačítko na odeslání formuláře. Celý příkaz na vyplnění formuláře vypadá následovně:

session.findById("wnd[0]/usr/ctxtEANLD-ANLAGE").Text = "1234567"



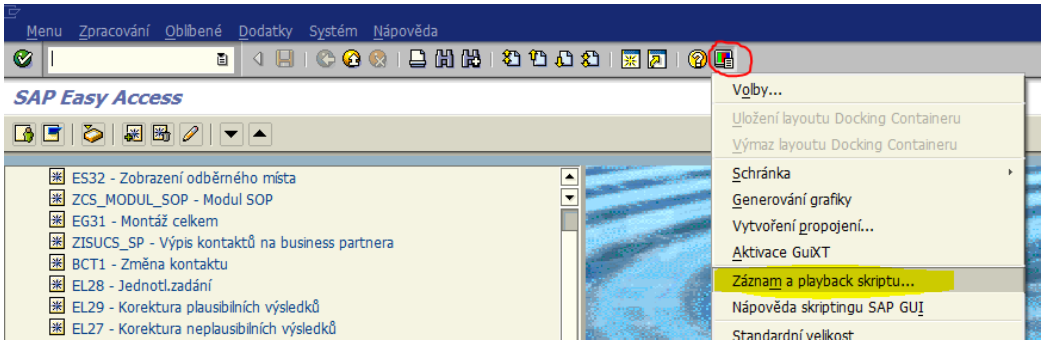
Obrázek 8 Znázornění ovládacích prvků. Zdroj: Vlastní.

Textový řetězec, který se nachází v uvozovkách v závorce příkazu, lze rozdělit do několika částí, které jsou odděleny lomítkem a představují jednotlivé objekty, se kterými příkaz pracuje. První objekt je „**wnd[0]**“, který reprezentuje konkrétní okno aplikace SAP (objekt GuiMainWindow), se kterým uživatel pracuje. V závorce je uvedeno pořadové číslo okna neboli transakce, ve které je uživatel přihlášen. Za prvním lomítkem se nachází další objekt „**usr**“ který zde představuje objekt GuiUserArea. Za druhým lomítkem je již konkrétní prvek, v našem případě „**ctxtEANLD-ANLAGE**“, který představuje vstupní textové pole. Za závorkou je již metoda „**Text = "1234567"**“, která zapíše do textového pole hodnotu z uvozovek. Pokud místo uvozovek a hodnot v nich použijeme námi definovanou proměnnou typu String, můžeme do proměnné ukládat hodnoty z Excelu.

Příkaz na odeslání formuláře k dalšímu zpracování (simulace stisknutí tlačítka) je: **session.findById("wnd[0]/tbar[0]/btn[0]").press**. Za prvním

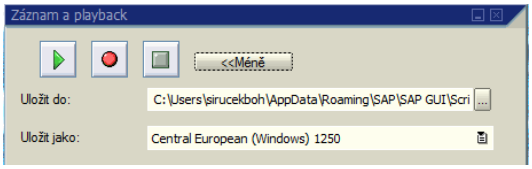
lomítkem se tentokrát nachází objekt „**tbar[0]**“, který zde představuje objekt GuiToolbar neboli nástrojovou lištu, na které se nacházejí ovládací prvky. V závorce je opět pořadové číslo. Na jedné zobrazované obrazovce mohou být dvě nástrojové lišty číslované 0 a 1. Za druhým lomítkem je již konkrétní prvek, v našem případě „**btn[0]**“, který představuje objekt tlačítka. V závorce je tentokrát uvedeno ID tlačítka. Za závorkou celého řetězce je metoda, která se provede. Zde je to příkaz „**press**“, který vyvolá událost stisknutí tlačítka s ID číslem 0 na nástrojové liště s pořadovým číslem 0 v transakci s pořadovým číslem 0. Takto jednoduše lze sestavit příkaz, kterým můžeme ovládat konkrétní prvky v GUI SAP.

Aby bylo možno poskládat celý příkaz, musíme znát ID jednotlivých prvků, které chceme ovládat, měnit nebo získat jejich hodnotu. K tomuto účelu můžeme využít integrovanou funkci SAPu, kterou je záznam a playback skriptu. Tuto funkci najdeme na panelu nástrojů, jak ukazuje obrázek 9.



Obrázek 9 Výběr záznamu a playbacku skriptu. Zdroj: Vlastní

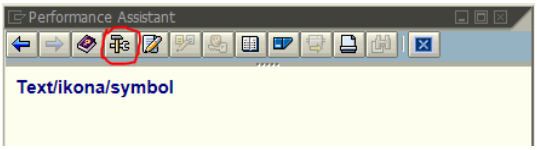
Funkci lze připodobnit k záznamu maker v Excelu. Po klepnutí na položku Záznam a playback skriptu se otevře modální okno (obrázek 10). V tomto okně si lze nastavit cílovou složku, do které se vygeneruje zaznamenaný skript. Po stisknutí červeného tlačítka se spustí metoda záznam skriptu a od této doby se budou zaznamenávat veškeré události, které vykoná uživatel v GUI SAP do doby, než záznam zastavíme. Pokud tedy chceme znát ID nějakého prvku, který se nachází na obrazovce, stačí na tento prvek klepnout ukazatelem myši a tato událost se zaznamená do vygenerovaného skriptu.



Obrázek 10 Modální okno Záznamu a playbacku skriptu. Zdroj: Vlastní.

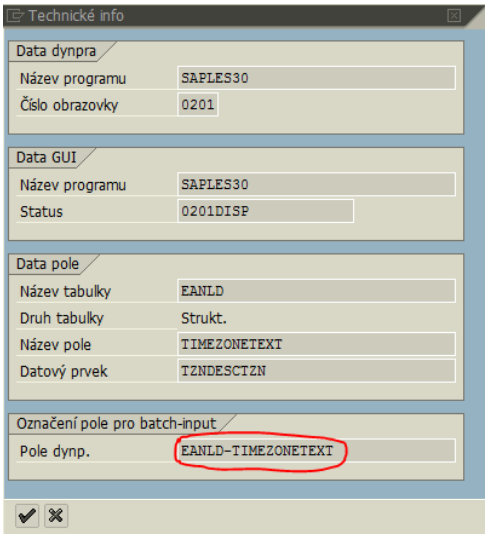
Skript se ukládá do souboru s koncovkou VBS (Visual Basic Script) a lze ho tedy editovat například v poznámkovém bloku, který je standardně integrován do prostředí Windows. Záznamem skriptu můžeme získat všechna ID ovládacích prvků, se kterými pracujeme a pomocí metod k tomu určeným zjistit i hodnoty těchto prvků. Jedinou výjimkou jsou hodnoty z tabulek. Pro jejich získání musíme procházet tabulku v programovém cyklu, který je potřeba naprogramovat, to nám funkce záznamu skriptu nezajistí.

Další možností, jak získat ID konkrétního prvku, je využití nápovědy systému a její části technické informace. Postupujeme tak, že na obrazovce označíme konkrétní prvek, u kterého chceme znát jeho systémový název a stiskneme klávesu F1. Otevře se modální okno Performance Assistant ve kterém v nástrojové liště klikneme na tlačítko Technické informace, jak ukazuje obrázek 11.

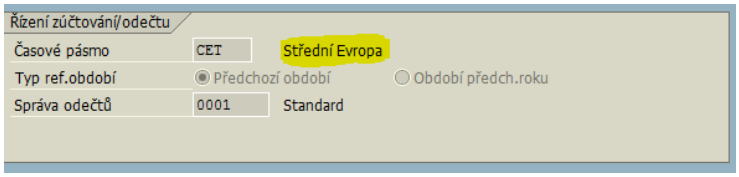


Obrázek 11 Okno Performance Assistant. Zdroj: Vlastní

Tím se zobrazí modální okno Technické informace (obrázek 12), kde je v poli (Pole dynp.), vypsán systémový název prvku, který jsme si na obrazovce označili. Tento název použijeme pro složení celého příkazu, který by byl v této konkrétní ukázce: **session.findById("wnd[0]/usr/txtEANLD-TIMEZONETEXT").Text**. Příkaz vrátí textovou hodnotu (Střední Evropa), prvku **label** jak je vidět na obrázku 13.



Obrázek 12 Technické informace vybraného prvku. Zdroj: Vlastní



Obrázek 13 Zobrazení popisku. Zdroj: Vlastní

Tento druhý způsob získání konkrétního ID prvku se hodí v případě, kdy již ve vytvořeném programu potřebujeme změnit jednu položku, třeba z důvodu testování. V takovém případě nemá smysl spouštět celé nahrávání skriptu, neboť to není efektivní. Jediné, na co musíme pamatovat je, že systémový název prvku, který získáme z technických informací, neobsahuje takzvaný prefix neboli typové určení prvku. V našem ukázkovém příkladu byl prefix txt neboli textové pole, zde konkrétně typu label. Typy a používání prefixů je vysvětleno v následující kapitole.

4.4.1 Prefixy používané u systémového názvu prvku

Prefix neboli předpona se používá u systémového názvu prvku z důvodu jasné identifikace datového typu prvku. Je to vlastně zkrácený název typu objektu prvku, díky kterému je umožněno využívat veškeré metody, které objekt nabízí. Hlavní prefixy, které se využívají, jsou vypsány v přehledu níže, ostatní jsou v dokumentaci [1]. Pokud bychom prefix nepřidali k systémovému názvu, tak skript vyvolá výjimku nenalezení objektu a skončí. Tato výjimka se objevuje i při

úpravě transakce, kdy se změní celý název systémového prvku nebo se systémový prvek zamění za jiný, pak musíme upravit název prvku i ve skriptu.

Přehled nejpoužívanějších prefixů:

- **ctxt** – GuiCTextField
  Jedná se o vstupní textové pole s rozevíracím tlačítkem se seznamem. Toto tlačítko může být viditelné pouze při kliknutí do textového pole.
- **txt** – GuiTextField
  Textové pole, které se podle nastavené vlastnosti outputFiel, která je typu boolean, chová buď jako vstupní jednořádkové textové pole, nebo jako klasický popisek label.
- **lbl** – GuiLabel
  Klasický popisek určený pouze ke čtení.
- **rad** – GuiRadioButton
  Tlačítko typu radio button.
- **tbl** – GuiTableControl
  Tabulka s možností zápisu hodnot
- **ssub** – GuiScrollContainer
  Datový kontejner, který představuje posouvatelnou obrazovku se scroll barem.
- **cmb** – GuiComboBox
  Ovládací prvek combo box s rozevíracím seznamem, ve kterém lze provést výběr položky z načteného seznamu.

4.4.2 Metody pro získání a změnu hodnot z tabulek objektu shell

Objekt shell je abstraktní třída, která nemá žádné ID a obsahuje načtená data z databáze v konkrétní transakci. Dá se říci, že ID tohoto objektu je název transakce, ve které je načten. Tato data jsou v GUI SAP zobrazena pomocí tabulky. Tabulka shell je v transakci jen jedna, lze k ní přistupovat pomocí příkazu **session.findById("wnd[0]/usr/shell/shellcont/shell")** a jeho dostupných metod. Jedna z nejvíce využívaných metod je „**getCellValue“**, která získá hodnotu

z konkrétní buňky tabulky. Metoda má dva parametry, číslo řádku, který je datového typu Long a název sloupce, který je typu String. Celý příkaz je:
session.findById("wnd[0]/usr/shell/shellcont/shell").getCellValue(1,“alfa“)
Řádky v tabulce jsou indexovány od 0. Výše uvedený příkaz vrátí hodnotu z druhého řádku ze sloupce s názvem alfa. Vrácená hodnota je typu String. Pokud potřebujeme do tabulky hodnoty zapsat, nebo je upravovat, tak použijeme metodu „**modifyCell“**, která má tři parametry a to, číslo řádku, které je datového typu Long, jméno sloupce a vlastní hodnotu. Tyto dva poslední parametry jsou datového typu String. Celý příkaz, který zapíše do druhého řádku a do sloupce s názvem alfa, textovou hodnotu z proměnné value, bude vypadat následovně:
session.findById("wnd[0]/usr/shell/shellcont/shell").modifyCell 1, "alfa", value.

Pro získání počtu řádků v tabulce použijeme metodu „**rowCount“**, která má návratovou hodnotu typu Integer. Pro zjištění počtu sloupců slouží metoda „**columnCount“**. Tyto dvě metody se využívají v programovém cyklu For, kterým procházíme jednotlivé buňky tabulky. Další využívaná metoda je „**getCellType(1,“alfa“)“**, která má dva parametry, číslo řádku a jméno sloupce. Návratová hodnota je typu String a může nabývat hodnot „Normal“, „Button“, „Checkbox“, „ValueList“. Tato metoda se používá pro kontrolu hodnot v tabulce. Pokud potřebujeme ověřit, zda se v tabulce nachází komponenta Checkbox a v jakém se nachází stavu („zaškrtnuto“, „odškrtnuto“), použijeme výše zmiňovanou metodu a následně metodu „**getCellCheckBoxChecked(1,“alfa“)“**, která má návratovou hodnotu typu Boolean. Další metody, které se tak často nevyužívají, jsou vysvětleny v dokumentaci [1].

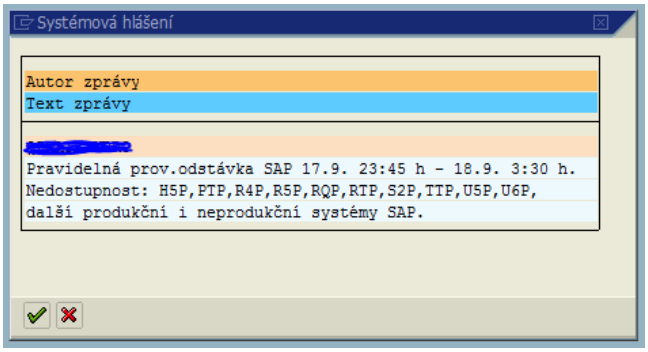
4.4.3 Ostatní příkazy používané v transakcích

Základní příkaz, který použijeme k přihlášení se do konkrétní transakce je **session.StartTransaction "jmeno_transakce"**, tento příkaz spustí námi zadanou transakci, která je tak připravena pro zadávání vstupních dat. K odhlášení se z transakce slouží příkaz **session.EndTransaction**, který ukončí konkrétní transakci a vrátí uživatele na úvodní obrazovku. Tento příkaz uzavře transakci bez jakýchkoliv oznámení, tedy bez uložení zpracovávaných dat, to musíme mít zajištěno před provedením tohoto příkazu. Dalším příkazem je

session.findById("wnd[0]").sendVKey cislo, který posílá do systému virtuální stisk klávesy. Tento příkaz se hodí pro případ, že potřebujeme nasimulovat třeba stisk klávesy „Enter“, funkčních kláves F1 – F12 a dalších kláves. Seznam číselných kódů, které lze použít, je součástí přílohy této bakalářské práce. Příkaz **session.findById("wnd[0]").maximize** slouží k maximalizování okna transakce a využívá se při testování skriptu, aby bylo vidět celé GUI, se kterým skript pracuje. Posledním zde zmíněným příkazem je **session.Children.Count**, který nám vrací číselnou hodnotu typu int. Tento příkaz je vhodné používat při každém vkládání nebo editaci dat v informačním systému, jako kontrolu vyskakovacích modálních oken, která se objevují, když se jedná o uložení kmenových dat. Tato modální okna se zobrazují i v některých transakcích jako okna potvrzovací. Pokud se po nějakém úkonu v transakci objeví modální okno, tak nám příkaz vrátí číselnou hodnotu 2 a pak musíme vyřešit, jak s tímto modálním oknem dále pracovat. Více příkazů, které nejsou pro automatizování procesů tak důležité, jako je na příklad zjišťování verze SAPu nebo souřadnic na obrazovce naleznete v dokumentaci [1].

4.4.4 Systémová hlášení SAP – nutnost ošetření ve skriptu

Systémová hlášení v SAP jsou krátké zprávy, které informují uživatele o změnách v systému, o chybách transakcí, nedostupnosti transakcí a dalších věcech. Ve stavovém řádku („GuiStatusbar“) se zobrazují zprávy, které informují uživatele o úspěšnosti provedené akce nebo o chybě transakce. Tyto zprávy lze jednoduše získat příkazem **session.findById("wnd[0]/sbar").Text**. Tento příkaz nám vrátí textový řetězec, který můžeme dále vyhodnotit a určit, jak se program bude chovat. Dalším druhem zpráv jsou zprávy přerušovací, které se zobrazují formou modálního okna a které do systému posílají administrátoři SAP. Tyto zprávy upozorňují uživatele o změnách nebo nepřístupnosti některých transakcí v systému SAP a je nutno tyto zprávy uzavřít ručně, jinak nelze v SAP pokračovat v další práci. S přerušovací zprávou je tedy potřeba počítat při vytváření skriptu. Tento problém lze vyřešit příkazem **session.TestToolMode = 1**, který zaručí, že během provádění skriptu se nebudou tyto zprávy zobrazovat a skript může být korektně proveden až do konce. Příkaz umístíme hned na začátek programu před příkaz **session.StartTransaction.**



Obrázek 14 Příklad přerušovacího systémového hlášení. Zdroj: Vlastní

4.4.5 Vnitřní logika programu

Pokud automatizovaný proces pracuje s několika transakcemi, je z hlediska budoucích úprav a testování, nutnost mít každou transakci v samostatné funkci, která bude mít námi požadovanou návratovou hodnotu. Díky tomu je možné si napsat samostatné testovací scénáře pro ověření různých výjimek, více je psáno v kapitole 4.6 Bezpečnostní rizika používání SAP GUI Scripting API.

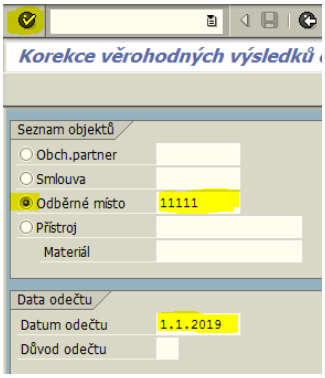
První, co musíme naprogramovat, je ověření vstupních dat, se kterými budeme pracovat. Tento krok je nezbytný, pokud chceme předejít pozdějším nepříjemnostem. Není totiž pravidlem, že data vyexportovaná z informačního systému, která je nutno dále zpracovat ručně, jsou zcela konzistentní. Chyby exportu dat se ve většině případů týkají nesprávným migrováním dat z jednoho systému do druhého, což je zcela běžné ve firmách, kde se pracuje s více informačními systémy najednou. Pokud, tato data zpracovává člověk, tak vidí, co lze zpracovat a co je nutno vyexportovat znovu. Skript musí být tedy schopen, podle nastavených pravidel, vstupní data validovat a rozhodnout, co se zpracuje automaticky a co zůstane ke zpracování administrativním pracovníkem. Jednou z nejzákladnějších úprav vstupních dat, je odstranění netisknutelných znaků, neboť při zadání takovéto položky do vstupního formuláře nějaké transakce v SAP dojde k výjimce běhu programu a dostaneme informační hlášení, že data neexistují. Jelikož zpracováváme data v Excelu, použijeme pro odstranění netisknutelných znaků funkci „**VYČISTIT**“ jejíž syntaxe je „**VYČISTIT(Text)**“ argument v závorce je jakákoliv hodnota z buňky sešitu Excel. Ve VBA použijeme syntaxi „**vysledek = WorksheetFunction.Clean(Text)**“. Následující úprava dat zajistí odstranění mezer před a za zpracovávanou položkou. K tomu slouží funkce

„**PROČISTIT**“ kterou můžeme ve VBA použít následujícím způsobem „**vysledek = WorksheetFunction.Trim(Text)**“. Více se lze o problematice čištění dat v Excelu dozvědět na oficiálních stránkách podpory Microsoftu [9].

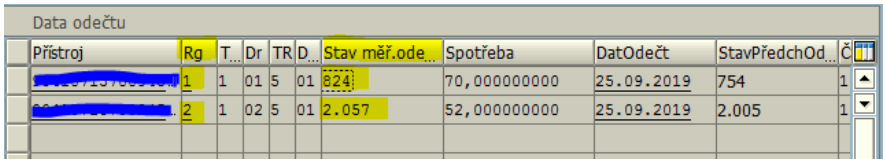
Máme-li ošetřena vstupní data, vytvoříme si samostatné funkce, které nám vyplní položkami z Excelu vstupní formuláře v transakcích SAP a vrátí požadované hodnoty. V textu této kapitoly budou uvedeny jen části skriptů, které jsou dle mého názoru důležité pro pochopení toho, jak do systému SAP data posílat a jak je ze systému dostat pro další zpracování. Kompletní kód je součástí přílohy na CD. První funkce, kterou si pojmenujeme „**GetDataEL29“**, nám na základě vstupních dat spočítá v SAPu hodnoty, které použijeme pro další transakci. Vstupní data pro tuto funkci jsou datum odečtu a číslo odběrného místa. Tyto dvě hodnoty předáme funkci jako parametry datového typu String. Kód, který spustí transakci, vyplní požadované údaje do formuláře (obrázek 15) a ten následně odešle do systému, by měl vypadat následovně:

```
1. session.StartTransaction "EL29"
2. session.findById("wnd[0]/usr/radRELX1-ANLAGE_T").Select
3. session.findById("wnd[0]/usr/ctxtREL28D-ANLAGE").Text = vstupOM
4. session.findById("wnd[0]/usr/ctxtREL28D-ADAT").Text = vstupDatumOdectu
5. session.findById("wnd[0]/tbar[0]/btn[0]").press
6. kontrolaStatusu = session.findById("wnd[0]/sbar").Text
```

První řádek skriptu spustí transakci s názvem EL29, druhý řádek provede přepnutí radiobuttonu na požadovanou položku. Další dva příkazy vyplní požadovaná textová pole. Následuje příkaz pro stisknutí tlačítka, které odešle formulář do systému. Poslední šestý příkaz zapíše do proměnné „kontrolaStatusu“ hodnotu z objektu GuiStatusbar. Pokud odeslání formuláře proběhlo v pořádku, tak tato proměnná bude prázdná a zobrazí se nám obrazovka s tabulkou (obrázek 16), která se načetla dle vstupních parametrů.



Obrázek 15 Vstupní formulář. Zdroj: Vlastní



Obrázek 16 Načtená tabulka po odeslání formuláře. Zdroj: Vlastní

Nyní potřebujeme získat dvě číselné hodnoty z načtené tabulky, které pak zapíšeme do tabulky v Excelu a využijeme dále ve skriptu. Kód pro načtení hodnot a uložení do proměnných je:

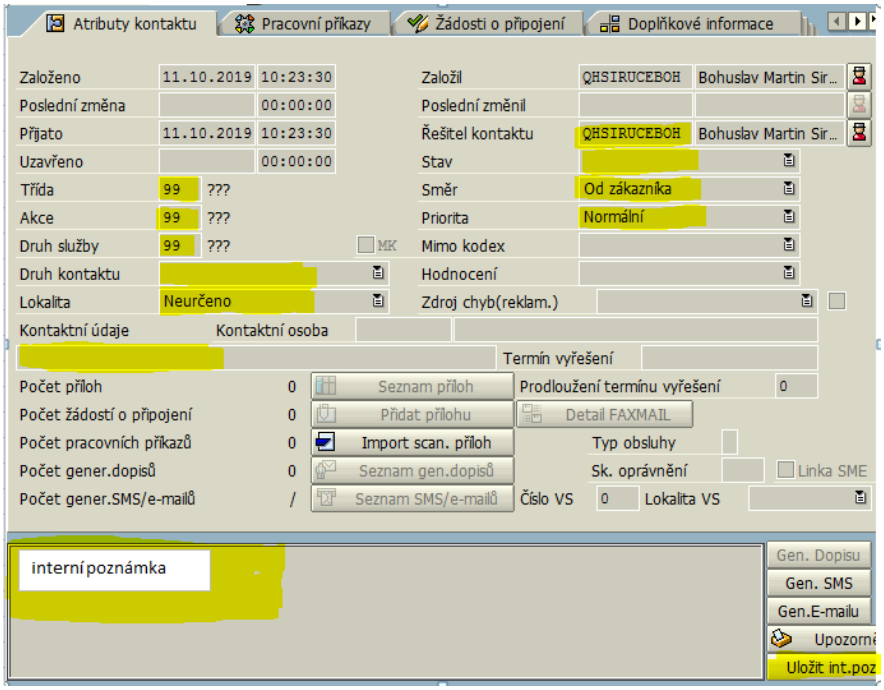
1. pocetRad=session.findById("wnd[0]/usr/tblSAPLEL01CONTROL_SINGENT").RowCount
2. For i = 0 To pocetRad – 1
3. cislo = session.findById("wnd[0]/usr/tblSAPLEL01CONTROL_SINGENT/txtREABLD-ZWSTAND[6," & i & "]").Text
4. If Left(cislo,1) = "_" OR cislo = empty Then Exit For
5. cisloReg =session.findById("wnd[0]/usr/tblSAPLEL01CONTROL_SINGENT/txtREABLD-ZWNUMMER[1," & i & "]").Text
6. If cisloReg = "1" Then registr1 = cislo
7. If cisloReg = "2" Then registr2 = cislo
8. Next i

První řádek skriptu naplní proměnnou „pocetRad“ číslem počtu viditelných řádků v tabulce. Tuto proměnnou použijeme ve For cyklu, kterým budeme procházet jednotlivé řádky a načítat z nich požadované hodnoty. Třetí řádek nám do proměnné „cislo“ zapíše hodnotu ze sloupce „Stav měř.ode.“ Na dalším řádku je kontrola, zda proměnná číslo, nemá poslední znak podtržítko nebo prázdnou hodnotu. Kontrolu na podtržítko musíme udělat, neboť v některých tabulkách je

místo prázdné hodnoty řetězec znaků z podtržítek „_____“. Na sedmém řádku si načteme do proměnné „cisloRegistru“ hodnotu ze sloupce s názvem „Rg“, tuto hodnotu budeme vyhodnocovat ve dvou následujících příkazech, kde podle výsledku naplníme proměnné „registr1“ a „registr2“, které nám námi vytvořená funkce vrátí jako návratovou hodnotu.

Následující funkce bude sloužit k založení kontaktu v transakci SAP. Tato transakce slouží k vytvoření záznamu pro fakturaci nebo pro jiná oddělení, která jsou v jiném organizačním začlenění a jsou závislá na procesech z rozdílných částí organizace. Funkci nazveme například **„ZalozKontakt“** a část jejího kódu bude vypadat následovně:

```
1. – 10.
session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/ctxtBCONTD-CCLASS").Text = trida
11.session.findById("wnd[0]/usr/cntlZCONTROL_CONTAINER/shellcont/shell").Text = interniPoznamka
12.session.findById("wnd[0]/usr/btnBT_NSAI").press
13.pocetModal = session.Children.Count
14.If pocetModal > 1 Then
15.  session.findById("wnd[1]/usr/btnBUTTON_1").press
16.End If
17.session.findById("wnd[0]/tbar[0]/btn[11]").press
18.vysledek = session.findById("wnd[0]/sbar").Text
```



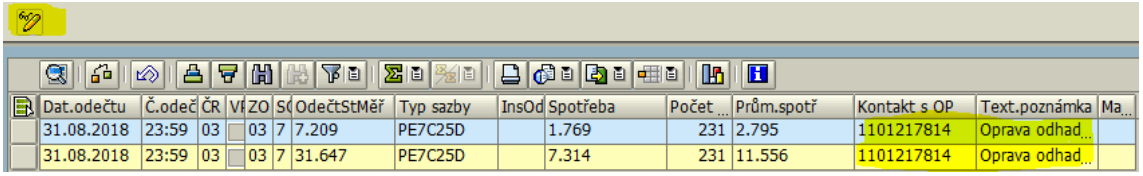
Obrázek 17 Formulář v transakci pro založení kontaktu. Zdroj: Vlastní

Prvních deset příkazů slouží k vyplnění polí formuláře (obrázek 17) hodnotami, které předáváme jako parametr funkce. V příkladu je uveden pouze jeden řádek, neboť ostatní jsou stejném, jen se mění ID formulářového prvku a proměnná, kterou zapisujeme do formuláře. V příkazu je to vyznačeno červeně. Řádek jedenáct vloží proměnnou „interniPoznamka“ do formulářového pole. Na dvanáctém řádku, je příkaz stisknutí tlačítka, který nám uloží interní poznámku do systému. Pokud bychom tento krok vynechali a uložili celý formulář, tak se interní poznámka neuloží. Po provedení tohoto příkazu následuje kontrola, zda se otevřelo modální okno, neboť při ukládání interní poznámky dojde k otevření potvrzovacího modálního okna, které nás informuje o tom, že se chystáme uložit interní poznámku do systému. Pokud tedy příkaz vrátí hodnotu větší než 1, tak příkazem na řádku 15 zajistíme stisknutí tlačítka „ANO“ na tomto modálním okně, které potvrdí naše rozhodnutí o uložení interní poznámky. Příkaz se od ostatních liší číslem v hranaté závorce u objektu GuiMainWindow **„wnd[1]“**, jednička znamená, že skript bude pracovat s další obrazovkou, v tomto případě s modálním oknem. Na sedmnáctém řádku je příkaz, který kompletně vyplněný formulář uloží do systému SAP. Poslední osmnáctý řádek skriptu zapíše do proměnné „vysledek“ text z objektu GuiStatusbar. Tato proměnná je návratová hodnota funkce. Text

z objektu GuiStatusbar je to informace o tom, zda uložení formuláře proběhlo v pořádku, či nikoli.

Poslední funkci, kterou budeme potřebovat, si nazveme **„VlozPoznakuZEL31“**. Tato funkce bude mít za úkol do systému SAP uložit textovou poznámku s daty, které nám systém spočítal v první funkci a číslo kontaktu, které nám vygenerovala transakce z předchozí funkce. Je zde opět uveden jen segment skriptu s nejdůležitějšími příkazy, celá funkce je součástí přílohy na CD. Část kódu, kterou si popíšeme, je následující:

1. session.findById("wnd[0]/tbar[1]/btn[5]").press
2. pRad = session.findById("wnd[0]/usr/cntlZISUMD_ZEL31/shellcont/shell").RowCount
3. For i = 0 To pRad - 1
4. session.findById("wnd[0]/usr/cntlZISUMD_ZEL31/shellcont/shell").modifyCell i, "ZZPOZN_ODEC", vstupPoznamka
5. session.findById("wnd[0]/usr/cntlZISUMD_ZEL31/shellcont/shell").modifyCell i, "ZZCONTACT", vstupKontakt
6. Next i
7. session.findById("wnd[0]/tbar[0]/btn[11]").press
8. vysledek = session.findById("wnd[0]/sbar").Text



Dat.odečtu	Č.odeč	ČR	VI	ZO	S(	OdečtStMěř	Typ sazby	InsOd	Spotřeba	Počet	Prům.spotř	Kontakt s OP	Text.poznámka	Ma
31.08.2018	23:59	03		03	7	7.209	PE7C25D		1.769	231	2.795	1101217814	Oprava odhad...	
31.08.2018	23:59	03		03	7	31.647	PE7C25D		7.314	231	11.556	1101217814	Oprava odhad...	

Obrázek 18 Tabulka typu shell se zapsanými buňkami. Zdroj: Vlastní

První řádek skriptu provede stisknutí tlačítka, které zajistí odemknutí načtené tabulky pro úpravy. Další řádek skriptu zapíše do proměnné „pRad“ počet řádků tabulky. Následující kód prochází jednotlivé řádky tabulky (obrázek 18) a do příslušných buněk zapisuje potřebné údaje, které byly předány jako parametry této funkce. Příkaz ze sedmého řádku provede uložení tabulky do systému a poslední příkaz nám vrátí textovou hodnotu výsledku uložení.

4.5 Efektivita automatizace oproti manuálnímu zpracování

Aby bylo možné změřit efektivitu automatizace, je nezbytné nejprve provést několik měření, která se týkají ručního zpracování dat v celém procesu. Do tohoto měření je nutno zahrnout celou řadu faktorů, jako je množství dat na vstupu, vytíženost informačního systému a v neposlední řadě i počet pracovníků, kteří zpracovávají data ručně. V knize Zlepšování podnikových procesů [10] je uvedeno: *„Měření procesů je nezbytné pro vytvoření podmínek k učení se a sledování účinnosti implementovaných procesních změn, stejně jako vytvoření nástrojů pro pozdější kontrolu a optimalizaci procesu.“* Je zcela jisté, že jakákoliv změna v nějakém procesu ovlivní i procesy, které na něho navazují.

Veškerá měření probíhala po dobu jednoho měsíce od pondělí do pátku za pomocí napsaného programu ve VBA, který u zpracovaných dat v Excelu vytvářel časové značky. Ukázka tabulky s měřenými časy manuálního zpracování dat je součástí přílohy. Podle časových značek byl vypočítán čas na zpracování jednotlivých kroků v procesu a tento čas byl následně zprůměrován. V tabulce 1 je porovnání průměrných časů při zpracování celého procesu manuálně a skriptem. Z tohoto měření byly vyloučeny případy, kdy byla některá vstupní data nekonzistentní a bylo je nutno vyexportovat ze SAP ručně. Největší rozdíl mezi ručním a automatickým zpracováním byl u formuláře v SAP, který představuje obrázek 17. V tomto formuláři je nutno vyplnit 11 položek, stisknout tři tlačítka a výsledek zapsat zpět do Excelu. Ruční zpracování tohoto formuláře zabere průměrně 47 sekund, ale zpracování skriptem trvá necelé 2 sekundy. Celý proces, který se sestává z přepsání dat z Excelu do tří transakcí a přepsání vygenerovaných dat zpět do Excelu, trvá ručním zpracováním průměrně 170 sekund. Skript tuto práci provede za 4 sekundy. Z těchto základních údajů je zřejmé, že automatické zpracování dat skriptem je skoro 42krát rychlejší než manuální zpracování. Převedeme-li tyto hodnoty na měsíční časovou hodnotu, kde budeme uvažovat 22 pracovních dní a denní zpracování 500 procesů, bude manuální zpracování dat trvat 520 hodin. Oproti tomu bude skript ta samá data zpracovávat pouhých 12 hodin. Toto jsou čisté průměrné časy, kdy systémy optimálně fungují a vstupní data jsou v pořádku.

Tabulka 1 Časové porovnání zpracování procesu manuálně a skriptem

	1 proces	Denní objem dat (500 procesů)	22 pracovních dnů (11000 procesů)
Ruční zpracování dat	170 sekund	23 hodin	520 hodin
Automatické zpracování dat	4 sekundy	0,6 hodin	12 hodin

4.6 Bezpečnostní rizika používání SAP GUI Scripting API

Stejně jako uživatel udělá chybu, kterou nelze zjistit podle nastaveného pravidla pro ověření (například při zadávání identifikačního čísla klienta, zamění uživatel jednu číslici ve vstupním formuláři a po odeslání se mu načtou data někoho jiného. Pokud si uživatel nezkontroluje další položky, jako jméno, příjmení, adresu, nastane problém a do systému se uloží chybná data), tak i skript, který není dostatečně otestován a nemá nastaveny kontrolní mechanismy, může do systému zanést chyby. Rozdíl je v tom, že skript zpracuje data mnohonásobně rychleji než při ručním zadávání uživatelem a může běžet bez dozoru. Tím může dojít k vygenerování mnoha nekorektních dat, než se na chybu přijde. Samozřejmě, že skript můžeme využít i pro zpětnou opravu těchto chybných dat, ale čím později se na chybu přijde, tím obtížněji se bude opravovat.

Abychom předešli riziku zanesení chyb do informačního systému při používání automatizovaných procesů, je nutno sestavit si testovací scénář. Jako první by se měla otestovat vstupní data, se kterými má skript pracovat. Pokud některá hodnota neodpovídá námi nastaveným pravidlům nebo dokonce chybí, je nutno zajistit, aby tento řádek nebyl skriptem zpracován a byl vytříděn pro manuální kontrolu. Touto prvotní kontrolou si zajistíme, že do automatizovaného procesu se nedostanou nekorektní data a nemusíme každou položku při automatickém zadávání do formulářů kontrolovat zvlášť, což podstatně zrychlí zpracovávání dat. Více je o tomto kroku napsáno v kapitole 4.4.5 Vnitřní logika programu.

Další samostatný test se týká kontroly dat po odeslání automaticky vyplněného formuláře v transakci SAP. To znamená, skript vyplní automaticky formulář vybranými daty a odešle ho do systému. Systém, pokud proběhne vše

korektně, zobrazí další námi požadované informace. Zda proběhlo vše v pořádku, zjistíme z objetu GuiStatusBar příkazem **session.findById("wnd[0]/sbar").Text**. Pokud byl formulář zpracován korektně, tak nám výše uvedený příkaz vrátí prázdný textový řetězec. Vrátí-li nám příkaz jakýkoliv neprázdný textový řetězec, tak opět zpracovávaný řádek ze vstupního souboru vytřídíme a vypíšeme k němu hlášení ze SAP. Důrazně doporučuji kontrolu GuiStatusBaru po každém automatickém stisknutí tlačítka nebo automatickém vybrání položky z rozevíracího seznamu. Předejde se tím mnoha nepříjemnostem, a díky vypisování systémových hlášení se dají nastavit další pravidla, která má skript vyhodnocovat. Některá systémová hlášení neznamenají chybu, ale informují o korektním provedení dané akce. To se týká situací, kdy do informačního systému zapisujeme, měníme a ukládáme data. Pak vrácený textový řetězec z GuiStatusBaru musíme porovnat a vyhodnotit dle našich nastavených pravidel ve skriptu.

4.6.1 Oprávnění k používání skriptu – logování uživatelů

Jak již bylo zmíněno v kapitole 4.2 SAP GUI Scriptin API, je používání skriptingu po instalaci SAP defaultně zakázáno. Uživatel, který nemá administrátorská práva, nemůže skriptování povolit. Administrátorská práva mají ve většině firem pouze správci systému SAP. Administrátor může na základě žádosti vypnout nebo zapnout skriptování pro konkrétní aplikační server nastavením parametru profilu **sapgui / user_scripting**. Hodnota TRUE tohoto parametru umožňuje skripting využívat. Od verze SAP 6.40 jsou k dispozici podrobnější parametry nastavení profilu skriptování. Parametr **sapgui/user_scripting_disable_recording** nastavený na hodnotu True umožní přehrávání skriptů, ale nepovolí jejich nahrávání. To je vhodné pro uživatele, kteří skriptem spouštějí předem přednastavené reporty a potřebují pouze měnit vstupní parametry skriptu, jako například datum. Dalším parametrem je **sapgui / user_scripting_set_readonly** Pokud je tento parametr nastaven na hodnotu True, tak skripty neumožní ovládat žádné prvky uživatelského rozhraní, umožňují pouze čtení jejich hodnot. Toto nastavení se hodí jen u specifických transakcí a pro pracovníky, kteří s informačním systémem nemají žádné zkušenosti.

Je ještě možnost vypnout skriptování přímo na uživatelském počítači, a to zásahem do registru Windows **HKLM \ SOFTWARE \ SAP \ SAPGUI Front \ SAP Frontend Server \ Security \ UserScripting** a nastavení klíče na hodnotu 0. Výchozí nastavení tohoto klíče má po instalaci hodnotu 1.
Aby bylo možné oprávněné spouštění skriptu, musí být uživatel nejprve do informačního systému přihlášen pomocí jména a hesla. Je hrubým porušením bezpečnostních pravidel do skriptu tyto údaje zadávat, neboť se vystavujeme nemalému riziku zneužití těchto údajů. Více je o bezpečnosti nastavení SAP v dokumentaci SAP GUI Scripting Security Guide [13].

Nedílnou součástí skriptu by měla být funkce, která provede zápis uživatele do přidělené databáze. Do databáze by se mělo zaznamenat, kdo skript spustil, název skriptu a čas ukončení skriptu. Tato databáze slouží pro bezpečnostní audit, který se běžně v korporacích provádí. Dále lze tuto databázi využít pro vyhodnocování využívání skriptů. Pokud jako další údaj pro zápis do databáze přidáme množství zpracovaných dat, a další doplňující položky, tak získáme neocenitelný přehled pro kapacitní plánování zdrojů ve firmě.

4.6.2 Všeobecná pravidla pro bezpečný vývoj programu

Bezpečnost programu je jedním z kritérií kvality aplikace stejně jako funkční požadavky. Zajištění kvality software je proces nazývaný Quality Assurance (QA) a jeho hlavní myšlenkou je, že kvalitní výsledek je zajištěn kvalitním procesem vývoje. V knize Testování pro programátory [11] je uvedeno: *„Quality Assurance je komplexní aktivita, která zajišťuje, že standardy, procesy a procedury použité během vývoje a údržby software jsou vhodné a správně dodržované. Na rozdíl od testování, které je jeho podmnožinou se QA zabývá prevencí chyb, implementací a vylepšováním firemních nařízení a jako takové je to záležitost všech zainteresovaných.“* V každé korporaci, kde probíhá návrh aplikačních řešení software, by měl být vytvořen závazný dokument, ve kterém jsou stanovena pravidla pro bezpečný vývoj, testování a provozování softwaru, který je ve vlastnictví této korporace. Dokument by měl stanovovat činnosti pro zajištění bezpečnosti vyvíjených aplikací, které jsou v souladu s bezpečnostními směrnicemi, pravidly a postupy za účelem

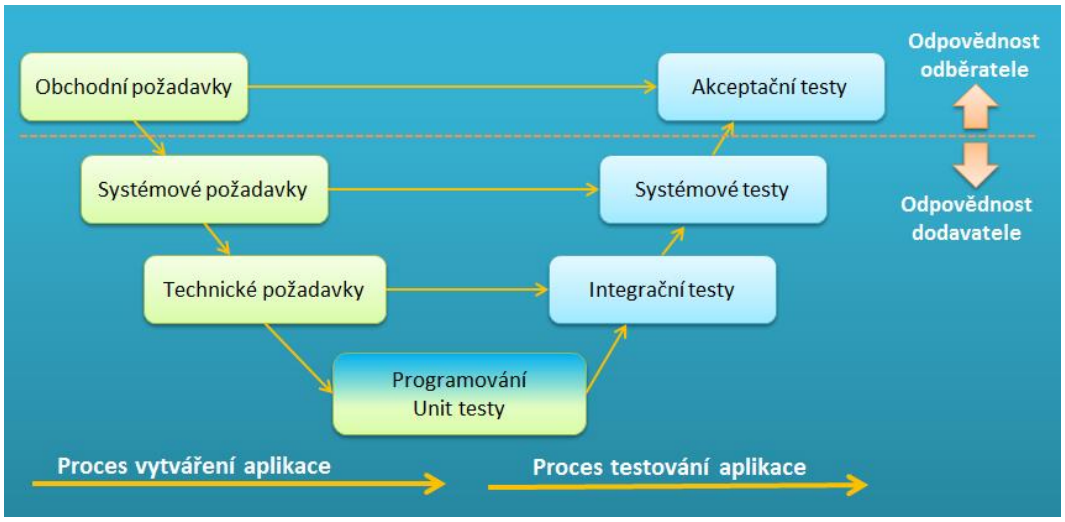
minimalizace rizik při provozu těchto aplikací. Níže jsou vypsána některá pravidla, která by měla být ve vnitropodnikových směrnicích uvedena.

- Jsou minimalizovány potenciální cesty útoku, které mohou být využity k odcizení firemních dat (odstranění nadbytečných služeb na serverech, odstranění nepotřebného kódu, který sloužil k testování).
- Součástí vývoje musí být i bezpečnostní testy a kontroly kódu, které, mají za cíl odhalit chyby dříve, než je aplikace schválena do provozu.
- Vývoj aplikace musí podléhat standardizovanému životnímu cyklu, který minimálně zahrnuje specifikaci požadavků, vývoj a testování.
- Logování aplikace musí být její nedílnou součástí a v souladu se standardem informační a kybernetické bezpečnosti korporace.
- Aktualizace, změny a opravy se musí řídit stejnými bezpečnostními pravidly, jako by šlo o nový vývoj.
- Testovací data musí být komplexní a v takovém objemu, aby mohlo být provedeno dostatečné otestování všech požadovaných funkcionalit aplikace.
- Akceptační testy musí probíhat v testovacím prostředí, které odpovídá produkčnímu prostředí.

Nejen při odlaďování programu je dobré mít napsanou metodu, která nám bude do externího souboru zapisovat logy výskytu chyb, výjimek, hodnotu určených proměnných a cokoliv dalšího, co je potřeba sledovat a co by nám usnadnilo zpětnou analýzu problému nejen při testování. Logování se většinou provádí do textového souboru TXT nebo XML dokumentu, ale je možno samozřejmě logy zapisovat i do nějaké databáze, záleží pouze na našich preferencích a způsobu dalšího zpracování logů.

Samotné testy softwaru lze rozdělit do několika úrovní (testing levels), které jsou určeny fází, ve které se testuje. Jednotlivé úrovně testování jsou znázorněny na obrázku 19, který představuje takzvaný V-Model, jenž byl definován již v druhé polovině osmdesátých let 20. století a pojednává o něm kapitola v knize Efektivní testování softwaru [12]. V-Model nám v souhrnu říká, že systém nestačí pouze

navrhnout a naprogramovat, ale že je potřeba v určitých časových pásmech zařadit testování, které ověří návrh a realizaci systému.



Obrázek 19 V-Model úrovně testování softwaru. Zdroj: Vlastní

- **Jednotkové testy (Unit testing)**

  Testují se jednotlivé metody tříd, funkce, kterým se předávají různé hodnoty a kontrolují se vrácené výsledky.
- **Integrační testy**

  Testy se někdy označují jako testy vnitřní integrace a ověřuje se v nich nejenom správnost komunikace mezi jednotlivými metodami, ale také mezi komponentami a operačním systémem, hardwarem a dalšími rozhraními systémů, které s programem spolupracují. Chyby, které se objeví při integračním testování a nebudou opraveny, se zcela jistě objeví v průběhu dalších úrovní testování. Proto, pokud odstraníme chyby v tomto kroku testování, budeme mít následující testování jednodušší.
- **Systémové testy**

  Testuje se aplikace jako funkční celek napříč celým systémem. Je to testování z pohledu zákazníka podle předem stanovených scénářů, při kterých se simulují situace, které mohou nastat v praxi. Bez systémového testování by byla bezporuchovost výsledného produktu výrazně ohrožena.

- **Akceptační testy**

  Jsou to testy na straně zákazníka, který si většinou se svými testery provede akceptační testy na svém prostředí. Pokud se vyskytnou nesrovnalosti mezi specifikací a samotnou aplikací, vrací se aplikace zpět k vývojovému týmy, který tyto nedostatky odstraní a aplikace se znovu přetestuje u zákazníka.

5 Shrnutí výsledků

Skript, který byl naprogramován pro automatizování vybraného procesu a vysvětlen v této bakalářské práci, zpracuje měsíční objem dat průměrně za 12 hodin, kdežto jeden člověk vykoná tuto práci za 520 hodin, rychlost skriptu znázorňuje tabulka 1. Z těchto výsledků je zcela jasná efektivita automatického zpracování dat. Z praktických zkušeností je také zřejmé, že pokud nemáme 100 % validních dat, která vstupují do automatizovaného procesu, tak ani automatizovaný proces nemůže veškerá data zpracovat. Proto, u takových procesů nelze plně nahradit manuální činnost člověka.

Při řešení problému automatizace administrativního procesu, je nejtěžší a zároveň nejdůležitější přesně popsat proces, který chceme automatizovat. Další, co nesmíme opomenout je dostatečné testování, které v celém vývoji zabere nejvíce času, ale bez toho to nejde, pokud chceme mít skript dostatečně robustní. Samotné naprogramování je časově nenáročné, více času zabere vytvoření dokumentace a testování. Automatizování procesů není primárně určeno k nahrazení lidí, ale k eliminování rutinních činností, které zbytečně zabírají čas pro smysluplnou práci. SAP Scripting lze úspěšně využít pro řešení mimořádných situací, jako je odstávka systému a následné nahromadění práce, které by bylo nutné řešit přesčasovými hodinami. V těchto situacích jsou skripty vhodné alespoň na částečné předzpracování dat.

6 Závěry a doporučení

Automatizování procesů ve firmách je nedílnou součástí jejich rozvoje a konkurenceschopnosti. Tato bakalářská práce měla za cíl ukázat, jakým způsobem využít SAP GUI Scripting API informačního systému SAP k automatizování rutinních administrativních činností a porovnat časy zpracování dat manuálně oproti zpracování skriptem. Na konkrétním procesu byly vysvětleny principy a možná úskalí, která se týkají využívání SAP GUI Scripting API. Bylo poukázáno na bezpečnostní rizika používání robotizovaných procesů. Dle výsledků rychlosti zpracovaných dat skriptem (tabulka 1) bylo dokázáno, že skript je mnohonásobně rychlejší než člověk. Na druhou stranu, pokud dojde k neočekávané situaci nebo nějaké výjimce ve skriptu nebo informačním systému, jsou znalosti člověka nepostradatelné a na to je nutné brát zřetel při zavádění automatizace. Nejpodstatnější věc, na kterou je potřeba ve skriptu pamatovat, je kontrola a vyhodnocování každého automatizovaného kroku, který skript vykoná. Neboť dle výsledků porovnání rychlosti zpracování dat člověkem a skriptem, je jasné, že skript, pokud není dostatečně odladěn a otestován, může během několika málo okamžiků zanést do informačního systému velké množství chyb. Je také důležité zajistit, aby ke skriptu byla vytvořena dostatečná dokumentace pro případ, kdy programátor, který skript vytvořil, z firmy odejde. Dle vlastních zkušeností vím, že dokumentace ke skriptům se ve firmách příliš nevede. Ve většině případů se jedná hlavně o menší a jednoúčelové skripty, které byly vytvořeny jedním programátorem. Každý má vlastní styl programování a používá odlišnou logiku návrhu programu a názvy proměnných. Někdy je kód programu tak nesrozumitelný, a pokud k němu není žádná dokumentace, je lepší ho naprogramovat znovu a vytvořit k němu kompletní dokumentaci včetně vývojového diagramu, aby bylo možno se v programu co nejrychleji orientovat a provést potřebné úpravy.

Skript, který byl pro tuto bakalářskou práci vytvořen, je naprogramován ve Visual Basic for Applications a celý program běží v prostření Excelu. Toto řešení je vhodné, pokud automatizujeme jednoduchý proces zpracování dat, ve kterém se nevyžaduje v některém kroku sofistikované rozhodnutí člověka, jako je například

výběr období, podle kterých se provedou další výpočty. Pokud nebudeme uvažovat o zavedení strojového učení nebo jiné metody umělé inteligence, a přesto chceme alespoň z části takovýto proces automatizovat, je vhodné navrhnout kompletní program, který bude člověku co nejvíce pomáhat, ale některá rozhodnutí nechá na něm. To je již vhodné naprogramovat jako samostatnou aplikaci například v programovacím jazyce C# pro který jsou vytvořeny speciální knihovny, které umožňují práci se SAP GUI Scripting API. Jedná se o dvě knihovny, **SAPFEWSELib,** která obsahuje metody pro práci s objekty SAP a **SapROTWr**, což je knihovna, která umožňuje přístup k Running Objesct Table a vytváří datové spojení mezi aplikací C# a SAP GUI Scripting API. Podobné knihovny jsou i pro aplikace psané v programovacím jazyku Java.

7 Seznam použité literatury

[1] SAP GUI Scripting API Documentation. *SAP Help Portal* [online]. Walldorf: SAP, 2012 [cit. 2019-10-19]. Dostupné z: https://help.sap.com/doc/saphelp_me151/15.1.3VERSIONFORSAPME/en-US/56/94cf535b804808e10000000a174cb4/frameset.htm

[2] SAP. *SAP* [online]. Walldorf: SAP, 2019 [cit. 2019-10-19]. Dostupné z: https://www.sap.com/corporate/en/company.html

[3] SAP Products. *SAP* [online]. Walldorf: SAP, 2019 [cit. 2019-10-19]. Dostupné z: https://www.sap.com/products-a-z.html

[4] SAP - Quick Guide. *Tutorials Point* [online]. Hyderabad: Tutorials Point India Limited, 2015 [cit. 2019-10-19]. Dostupné z: https://www.tutorialspoint.com/sap/sap_quick_guide.htm

[5] SAP BusinessObjects. *SAP Help Portal* [online]. Walldorf: SAP, 2019 [cit. 2019-10-19]. Dostupné z: https://help.sap.com/viewer/f1a6aa8ade38419692d0cf7186b6c585/10.0/en-US/fa99af336faf1014878bae8cb0e91070.html

[6] SAP Enhancement Packages. *SAP Help Portal* [online]. Walldorf: SAP, 2019 [cit. 2019-10-19]. Dostupné z: https://help.sap.com/viewer/6cb692c0292d43a2b8f4cfc1bae6e6f0/750%20SP14/en-US/d9a64d4bbd3d4856bbde2695e2fcfb4c.html

[7] How Does SAP Work?. *ERProof* [online]. San Francisco: ERProof, 2017 [cit. 2019-10-19]. Dostupné z: https://erproof.com/how-does-sap-work/

[8] PŠENČÍKOVÁ, Jana. *Algoritmizace*. 2. aktualizované vydání. Kralice na Hané: Computer Media, 2009. ISBN 978-80-7402-034-6.

[9] Deset nejlepších způsobů čištění dat. *Microsoft Office* [online]. Redmond: Microsoft Corporation, 2019 [cit. 2019-10-19]. Dostupné z: https://support.office.com/cs-cz/article/deset-nejlep%C5%A1%C3%ADch-zp%C5%AFsob%C5%AF-%C4%8Di%C5%A1t%C4%9Bn%C3%AD-dat-2844b620-677c-47a7-ac3e-c2e157d1db19

[10] SVOZILOVÁ, Alena. *Zlepšování podnikových procesů*. 1. vydání dotisk. Praha: Grada Publishing, 2019, 232 s. ISBN 978-80-247-3938-0

[11] HEROUT, Pavel. *Testování pro programátory*. 1. vydání. České Budějovice: Kopp, 2016, 406 s. ISBN 978-80-7232-481-1

[12] BUREŠ, Miroslav. *Efektivní testování softwaru*. První vydání. Praha: Grada Publishing, 2016, 232 s. ISBN 978-80-247-5594-6

[13] SAP GUI Security Guide. *SAP Help Portal* [online]. Walldorf: SAP, 2010 [cit. 2019-10-19]. Dostupné z: https://help.sap.com/viewer/ca5169c2f72448eeb608cd09564ccf90/760.03/en-US/10eb2d38522846e6a7f46601a0c4f927.html

8 Přílohy

Tabulka číselných kódů pro příkaz „**sendVKey“** převzato z [1]

VKey	Klávesová kombinace	VKey	Klávesová kombinace
00	Enter	31	Ctrl+F7
01	F1	32	Ctrl+F8
02	F2	33	Ctrl+F9
03	F3	34	Ctrl+F10
04	F4	35	Ctrl+F11
05	F5	36	Ctrl+F12
06	F6	37	Ctrl+Shift+F1
07	F7	38	Ctrl+Shift+F2
08	F8	39	Ctrl+Shift+F3
09	F9	40	Ctrl+Shift+F4
10	F10	41	Ctrl+Shift+F5
11	Ctrl+S	42	Ctrl+Shift+F6
12	F12	43	Ctrl+Shift+F7
13	Shift+F1	44	Ctrl+Shift+F8
14	Shift+F2	45	Ctrl+Shift+F9
15	Shift+F3	46	Ctrl+Shift+F10
16	Shift+F4	47	Ctrl+Shift+F11
17	Shift+F5	48	Ctrl+Shift+F12
18	Shift+F6	70	Ctrl+E
19	Shift+F7	71	Ctrl+F
20	Shift+F8	72	Ctrl+/
21	Shift+F9	73	Ctrl+\
22	Shift+Ctrl+0	74	Ctrl+N
23	Shift+F11	75	Ctrl+O
24	Shift+F12	76	Ctrl+X
25	Ctrl+F1	77	Ctrl+C
26	Ctrl+F2	78	Ctrl+V
27	Ctrl+F3	79	Ctrl+Z
28	Ctrl+F4	80	Ctrl+PageUp
29	Ctrl+F5	81	PageUp
30	Ctrl+F6	82	PageDown

Ukázka části tabulky s naměřenými časy ručního zpracování procesu

Vstupní data			Data vygenerovaná v SAP			Vstupní data		informace o zpracování	
datum odečtu	číslo OM	OP	stav V	stav N	číslo kontaktu	poznámka	informace k odečtu	status zpracování	čas zpracování (s)
01.09.2019	1	55	10	100	255	Byla provedena oprava stavů	Vygenerovaný kontakt č. 255	ok	140
01.09.2019	2	56	10	100	256	Byla provedena oprava stavů	Vygenerovaný kontakt č. 256	ok	144
01.09.2019	3	57	10	100	257	Byla provedena oprava stavů	Vygenerovaný kontakt č. 257	ok	170
01.09.2019	4	58	10	100	258	Byla provedena oprava stavů	Vygenerovaný kontakt č. 258	ok	159
01.09.2019	5	59	10	100	259	Byla provedena oprava stavů	Vygenerovaný kontakt č. 259	ok	184
01.09.2019	6	60	10	100	260	Byla provedena oprava stavů	Vygenerovaný kontakt č. 260	ok	176
01.09.2019	7	61	10	100	261	Byla provedena oprava stavů	Vygenerovaný kontakt č. 261	ok	176
01.09.2019	8	62	10	100	262	Byla provedena oprava stavů	Vygenerovaný kontakt č. 262	ok	189
01.09.2019	9	63	10	100	263	Byla provedena oprava stavů	Vygenerovaný kontakt č. 263	ok	179
01.09.2019	10	64	10	100	264	Byla provedena oprava stavů	Vygenerovaný kontakt č. 264	ok	176
01.09.2019	11	65	10	100	265	Byla provedena oprava stavů	Vygenerovaný kontakt č. 265	ok	184
01.09.2019	12	66	10	100	266	Byla provedena oprava stavů	Vygenerovaný kontakt č. 266	ok	180
01.09.2019	13	67	10	100	267	Byla provedena oprava stavů	Vygenerovaný kontakt č. 267	ok	169
01.09.2019	14	68	10	100	268	Byla provedena oprava stavů	Vygenerovaný kontakt č. 268	ok	159
01.09.2019	15	69	10	100	269	Byla provedena oprava stavů	Vygenerovaný kontakt č. 269	ok	167
01.09.2019	16	70	10	100	270	Byla provedena oprava stavů	Vygenerovaný kontakt č. 270	ok	187
01.09.2019	17	71	10	100	271	Byla provedena oprava stavů	Vygenerovaný kontakt č. 271	ok	149
01.09.2019	18	72	10	100	272	Byla provedena oprava stavů	Vygenerovaný kontakt č. 272	ok	181
01.09.2019	19	73	10	100	273	Byla provedena oprava stavů	Vygenerovaný kontakt č. 273	ok	166
01.09.2019	20	74	10	100	274	Byla provedena oprava stavů	Vygenerovaný kontakt č. 274	ok	150
01.09.2019	21	75	10	100	275	Byla provedena oprava stavů	Vygenerovaný kontakt č. 275	ok	163
01.09.2019	22	76	10	100	276	Byla provedena oprava stavů	Vygenerovaný kontakt č. 276	ok	183
01.09.2019	23	77	10	100	277	Byla provedena oprava stavů	Vygenerovaný kontakt č. 277	ok	163
01.09.2019	24	78	10	100	278	Byla provedena oprava stavů	Vygenerovaný kontakt č. 278	ok	176
01.09.2019	25	79	10	100	279	Byla provedena oprava stavů	Vygenerovaný kontakt č. 279	ok	189
01.09.2019	26	80	10	100	280	Byla provedena oprava stavů	Vygenerovaný kontakt č. 280	ok	173
01.09.2019	27	81	10	100	281	Byla provedena oprava stavů	Vygenerovaný kontakt č. 281	ok	156
01.09.2019	28	82	10	100	282	Byla provedena oprava stavů	Vygenerovaný kontakt č. 282	ok	177
01.09.2019	29	83	10	100	283	Byla provedena oprava stavů	Vygenerovaný kontakt č. 283	ok	145

Kompletní funkce **GetDataEL29**

```
Function GetDataEL29(vstupDatumOdectu As String, vstupOM As String)

    Dim i As Integer
    Dim pocetRad As Integer
    Dim kontrolaStatusu As String
    Dim cisloReg, registr1, registr2 As String

    pocetRad = 0
    registr1 = "-"
    registr2 = "-"
    ReDim vyslednaData(2)

        session.EndTransaction
        session.EndTransaction
        session.StartTransaction "EL29"
        session.findById("wnd[0]").maximize

        '--------------------vstupní obrazovka transakce-----------------------------
        session.findById("wnd[0]/usr/radRELX1-ANLAGE_T").Select
        session.findById("wnd[0]/usr/ctxtREL28D-ANLAGE").Text = vstupOM
        session.findById("wnd[0]/usr/ctxtREL28D-ADAT").Text = vstupDatumOdectu
        session.findById("wnd[0]/tbar[0]/btn[0]").press
       '-----------------------------------------------
       kontrolaStatusu = session.findById("wnd[0]/sbar").Text

       If Len(kontrolaStatusu) < 1 Then
            pocetRad = session.findById("wnd[0]/usr/tblSAPLEL01CONTROL_SINGENT").RowCount 'počet kontaktu
           '-------------------procházení tabulky--------------------
           For i = 0 To pocetRad - 1
                cislo = session.findById("wnd[0]/usr/tblSAPLEL01CONTROL_SINGENT/txtREABLD-ZWSTAND[6," & i & "]").Text
                cisloReg =session.findById("wnd[0]/usr/tblSAPLEL01CONTROL_SINGENT/txtREABLD-ZWNUMMER[1," & i & "]").Text
                If Left(cislo,1) = "_" OR cislo = empty Then Exit For
                If cisloReg = "1" Then registr1 = cislo
                If cisloReg = "2" Then registr2 = cislo
           Next i

        vyslednaData(0) = registr1
        vyslednaData(1) = registr2
     End If

    session.EndTransaction
    GetDataEL29 = vyslednaData

End Function
```

Kompletní funkce **ZalozKontakt**

```
Function ZalozKontakt(cisloOP  As String, trida  As String, akce  As String, kontaktniUdaje As String,  druh As String, lokalita As String _
                  , resitel As String, stav As String, smer As String, priorita As String, interniPoznamka As String, sluzba As String,)

    Dim vysledek As String
    Dim pocetModal as Integer
    Dim kontrolaStatusu As String

    vysledek = "nezaloženo"

    session.EndTransaction
    session.EndTransaction
    session.StartTransaction "BCT0"
    session.findById("wnd[0]/usr/ctxtBCONTD-PARTNER").Text = cisloOP
    session.findById("wnd[0]/tbar[0]/btn[0]").press

    kontrolaStatusu = session.findById("wnd[0]/sbar").Text

    If Len(kontrolaStatusu) < 1 Then
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/ctxtBCONTD-CCLASS").Text = trida
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/ctxtBCONTD-ACTIVITY").Text = akce
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/ctxtBCONTD-ZZDRUH_SLUZBY").Text = sluzba
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/cmbBCONTD-CTYPE").Key = druh
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/cmbBCONTD-ZZREAS").Key = lokalita
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/ctxtBCONTD-ZZWORKER").Text = resitel
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/cmbBCONTD-ZSTAV").Key = stav
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/cmbBCONTD-F_COMING").Key = smer
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/cmbBCONTD-PRIORITY").Key = priorita
        session.findById("wnd[0]/usr/tabsTSTRIP/tabpT_CA/ssubSUBSR_TSTRIP:SAPLBPT1:9900/txtBCONTD-ZZKONTAKT").Text = kontaktniUdaje
                '-----interní poznámka-------
            session.findById("wnd[0]/usr/cntlZCONTROL_CONTAINER/shellcont/shell").Text = interniPoznamka
            session.findById("wnd[0]/usr/btnBT_NSAI").press 'uložení poznámky

            pocetModal = session.Children.Count
            If pocetModal > 1 Then
                '------------potvrzení uložení interní poznámky------------
                session.findById("wnd[1]/usr/btnBUTTON_1").press
            End If

        'Uložení celého formuláře
        session.findById("wnd[0]/tbar[0]/btn[11]").press
        kontrolaStatusu = session.findById("wnd[0]/sbar").Text
        vysledek = kontrolaStatusu
    End If
        ZalozKontakt = vysledek

End Function
```

Kompletní funkce **VlozPoznakuZEL31**

```
Function VlozPoznamkuZEL31(vstupOM As String, vstupDatumOd As String, vstupDatumDo As String, vstupPoznamka As String _
                         , vstupKontakt As String)

    Dim vysledek As String
    Dim pRad As Integer
    Dim i As Integer
    Dim kontrolaStatusu As String

    vysledek = "---"
    session.EndTransaction
    session.EndTransaction
    session.StartTransaction "ZEL31"
    session.findById("wnd[0]").maximize

    '--------------------vstupní obrazovka------------------------------
    session.findById("wnd[0]/usr/ctxtSO_OM-LOW").Text = vstupOM
    session.findById("wnd[0]/usr/ctxtSO_DATUM-LOW").Text = vstupDatumOd
    session.findById("wnd[0]/usr/ctxtSO_DATUM-HIGH").Text = vstupDatumDo
    session.findById("wnd[0]/tbar[1]/btn[8]").press
    '-------------------------------------------------

    kontrolaStatusu = session.findById("wnd[0]/sbar").Text
         If Len(kontrolaStatusu) < 1 Then
             session.findById("wnd[0]/tbar[1]/btn[5]").press
             pRad = session.findById("wnd[0]/usr/cntlZISUMD_ZEL31/shellcont/shell").RowCount

            For i = 0 To pRad - 1
              session.findById("wnd[0]/usr/cntlZISUMD_ZEL31/shellcont/shell").modifyCell i, "ZZPOZN_ODEC", vstupPoznamka
              session.findById("wnd[0]/usr/cntlZISUMD_ZEL31/shellcont/shell").modifyCell i, "ZZCONTACT", vstupKontakt
            Next i
               session.findById("wnd[0]/tbar[0]/btn[11]").press
               kontrolaStatusu = session.findById("wnd[0]/sbar").Text
               vysledek = kontrolaStatusu
         End If
           VlozPoznamkuZEL31 = vysledek

End Function
```

Oskenované zadání práce

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2018/2019

Studijní program: Aplikovaná informatika
Forma studia: Kombinovaná
Obor/kombinace: Aplikovaná informatika (ai3-k)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení:	**Bohuslav Martin Sirůček**
Osobní číslo:	**I1700303**
Adresa:	Třída Edvarda Beneše 1544, Hradec Králové – Nový Hradec Králové, 50012 Hradec Králové 12, Česká republika
Téma práce:	Aplikování Sap scriptingu k automatizování uživatelských procesů v ERP systému SAP
Téma práce anglicky:	Apply SAP scripting to automate user processing in the SAP ERP system
Vedoucí práce:	Mgr. Hana Rohrová Katedra informačních technologií

Zásady pro vypracování:

Cíl práce: Cílem této práce je ukázat možnosti automatizování rutinních administrativních činností v informačním systému SAP pomocí SAP GUI scripting API.

Osnova

1. Podnikový informační systém SAP
2. SAP GUI scripting API
3. Objekty uživatelského rozhraní
4. Výběr administrativního procesu k automatizování

- Popis procesu pomocí vývojového diagramu
- Vlastní návrh programu
- Prefixy používané u systémového názvu prvku
- Metody pro získání hodnot z tabulek objektu shell
- Ostatní příkazy používané v transakcích
- Systémová hlášení SAP – nutnost ošetření ve scriptu
- Vlastní logika programu
- Efektivita automatizace oproti manuálnímu zpracování
- Bezpečnostní rizika používání Sap GUI scripting API
- Oprávnění k používání scriptu - logování uživatelů
- Všeobecná pravidla pro bezpečný vývoj programů

Seznam doporučené literatury:

dokumentace - SAP GUI Scripting API