**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

FRANTIŠEK CABADAJ

**Vývoj automatizovaného testovacieho systému pre kontrolu komunikácie medzi SAP a externými úložiskami**

Vedúci práce: doc. Ing. Marek Kvet, PhD.

Registračné číslo: 556425

Žilina, 2020

**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA ......................................

BAKALÁRSKA/DIPLOMOVÁ/

DIZERTAČNÁ PRÁCA

ŠTUDIJNÝ ODBOR:

MENO PRIEZVISKO

**Názov práce**

Žilinská univerzita v Žiline

Fakulta ..................

Školiace pracovisko..............

Žilina, 2016

#### Úvodné pokyny pre použitie šablóny

Väčšina nadpisov šablóny je previazaná na dokument Pokyny\_pre\_vypracovanie\_ZP.docx. Sú v ňom podrobnejšie informácie o vypracovaní záverečnej práce. Linky budú správne fungovať vtedy a len vtedy keď si uložíte dokument so záverečnou prácou do toho istého priečinka, v ktorom sa nachádza uvedený dokument. Priamo z tejto šablóny nefungujú.

#### (Pri vytváraní svojho dokumentu pokyny pre použitie šablóny vymažte!)

#### [Čestné vyhlásenie](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Čestné_vyhlásenie)

Podľa národnej legislatívy nie je v záverečnej práci povinné.

#### [Poďakovanie](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Poďakovanie)

Nie je povinné, vysvetlenie a príklad poďakovania

[ABSTRAKT V ŠTÁTNOM JAZYKU](C:\\Users\\DonChameleón\\Desktop\\Pokyny_pre_vypracovanie_ZP.docx" \l "Abstrakt)

CABADAJ, František: Vývoj automatizovaného testovacieho systému pre kontrolu komunikácie medzi SAP a externými úložiskami. [Bakalárska práca] – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky;

PRIEZVISKO, Meno: *Názov témy záverečnej práce*. [Bakalárska/ Diplomová/ Dizertačná práca]. – Názov univerzity. Názov fakulty; Názov školiaceho pracoviska. – Školiteľ/Vedúci: (tituly pred menom) Meno a priezvisko (tituly za menom) – Stupeň odbornej kvalifikácie: bakalár/magister/iný. – Mesto: skratka fakulty a univerzity, rok predloženia. Počet strán (napr. 35 s.)

[Vysvetlenie a príklad.](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Príklad_abstrakt)

[**Kľúčové slová**](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Kľúčové_slová)**:**  (jednoslovné alebo viacslovné termíny, ktoré charakterizujú vecný obsah práce)

[ABSTRAKT V CUDZOM JAZYKU](C:\\Users\\DonChameleón\\Desktop\\Pokyny_pre_vypracovanie_ZP.docx" \l "Abstrakt_CJ)

[Vysvetlenie.](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Abstrakt_CJ)

**Key words:**

[Obsah](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Obsah)

Zoznam obrázkov 7

Zoznam tabuliek 8

Zoznam skratiek 9

Úvod 10

1 Súčasný stav riešenej problematiky doma a v zahraničí 12

2 Teoretické východiská 13

2.1 Životný cyklus informačného systému https://raygun.com/blog/software-development-life-cycle/ 13

2.2 Testovanie 22

2.2.1 Funkčné testovanie 22

2.2.2 Automatizované testovanie https://testguild.com/automation-testing/ 25

3 Ciele práce 42

4 Praktická časť 43

4.1 Analýza externých úložísk 43

4.2 Zber a vyhodnotenie požiadaviek 43

4.3 Návrh automatizovaného testovacieho systému 43

4.3.1 Architektúra 43

4.4 Implementácia testovacieho systému 43

4.5 Vyhodnotenie dosiahnutých výsledkov 43

5 Metodika práce a metódy skúmania 44

6 Výsledky práce a diskusia 45

6.1 Výsledky práce 45

6.2 Diskusia 45

Záver 46

Zoznam použitej literatúry 47

Zoznam príloh 48

Prílohy 49

Príloha A: Názov prílohy 50

Príloha B: Obsah DVD 51

# [Zoznam obrázkov](C:\\Users\\DonChameleón\\Desktop\\Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_obrázkov)

**Nenašli sa žiadne položky zoznamu obrázkov.**

# [Zoznam tabuliek](C:\\Users\\DonChameleón\\Desktop\\Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_obrázkov)

**Nenašli sa žiadne položky zoznamu obrázkov.**

# [Zoznam skratiek](C:\\Users\\DonChameleón\\Desktop\\Pokyny_pre_vypracovanie_ZP.docx" \l "Zoznam_skratiek)

ANSA Automated Network Simulation and Analysis

BFD Automated Network Simulation and Analysis

Cisco IOS Cisco Internetwork Operating System

# [Úvod](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Úvod)

Úvodný text sa píše v štýle odseku textu Normálny. Ten je v tejto šablóne nastavený a neodporúčame ho meniť.

Zásady spracovania záverečnej práce sú uvedené [TU](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Zásady_spracovania_ZP).

Formálna úprava záverečných prác je uvedená [TU](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Formálna_úprava_ZP)

#### Pokyny pre používanie šablóny

V šablóne sú pripravené štyri typy nadpisov:

* **Nadpis 1** je číslovaný a začína vždy na novej stránke (dá sa priradiť klávesom **F5**)
* **Nadpis 2** nadväzuje číslovaním na Nadpis 1 (**F6**)
* **Nadpis 3** nadväzuje číslovaním na predchádzajúce nadpisy (**F7**)
* **Nadpis 4** je bez čísla (**F8**)

V šablóne sú pripravené ďalšie štýly odseku textu, pri ktorých odporúčame nemeniť ich nastavené vlastnosti:

* **Normálny** je určený na písanie bežného textu (**F4**)
* **Obrázok** je zarovnaný na stred a určený pre obrázky a ich popis (**F10**)
* **Skratky** je určený na písanie zoznamu skratiek použitých v dokumente
* **Literatúra** je určený na písanie zoznamu literatúry (ešte nie je pripravený)
* **Tabuľka** je určený na vkladanie textu do tabuliek
* **Typ práce** je určený na vypísanie typu práce do hlavičky

#### Pokyny pre prácu s ilustráciami

Ilustrácie sú obrázky obsahujúce grafy, diagramy, mapy, schémy a pod. Nie je potrebné rozlišovať rozličné typy ilustrácií, stačí, ak sa všetky označia ako „Obrázok”.

#### Vloženie ilustrácie

Obrázky a tabuľky vkladáme v tejto šablóne takto:

1. Karta **Vložiť** → **Obrázok**
2. Na obrázku klikneme na pravé tlačidlo myši → **Vložiť popis**
3. Do poľa **Popis** dopíšeme názov obrázku
4. Na obrázok a jeho popis použijeme štýl **Obrázok** (**F10**)

V texte sa na vytvorené obrázky a tabuľky odkazujeme krížovými odkazmi. Krížový odkaz na obrázky a tabuľky vytvoríme takto:

1. Nastavíme sa kurzorom tam, kde sa má objaviť odkaz,
2. Na karte **Vložiť** → **Krížový odkaz**
3. V dialógovom okne pre **Krížový odkaz** v poli **Typ odkazu** vyberieme **Tabuľka** alebo **Obrázok**
4. V poli **Vložiť odkaz** vyberieme **Iba menovka a číslo**
5. Zo zoznamu popisov vyberieme tabuľku alebo obrázok a zaškrtneme **Hypertextový odkaz**
6. Klikneme na **Vložiť**

#### Aktualizácia krížových odkazov a popisov obrázkov

Pokiaľ v texte vymažete alebo pridáte ilustráciu, tak je potrebné prečíslovanie ilustrácií a krížových odkazov:

1. Vyberte celý dokument (**Ctrl + A**)
2. Stlačte kláves **F9**

Obrázky by mali byť kreslené v rovnakom štýle s popisom v jazyku práce. Podrobnejší popis pre prácu s ilustráciami nájdete [tu](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Ilustrácie).

#### Pokyny pre prácu s tabuľkami

Každá tabuľka musí mať poradové číslo a titulok, umiestnený zvyčajne nad tabuľkou, pričom titulok je zarovnaný na pravú stranu tabuľky. Tabuľky by mali byť rovnako naformátované s rovnakým vzhľadom buniek.

Podrobnejší popis pre prácu s tabuľkami nájdete [tu](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Tabuľky).

#### Zoznam použitej literatúry

Bibliografické odkazy vkladajte cez kartu **Referencie,** zoskupenie nástrojov **Citácie a bibliografia**.

# [Súčasný stav riešenej problematiky doma a v zahraničí](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Súčasný_stav_riešenia)

# Teoretické východiská

## Životný cyklus informačného systému <https://raygun.com/blog/software-development-life-cycle/>

Software is a complex product that is developed and delivered through a series of steps. That is the one thing all the various methods have in common: one way or another, software, like all products, starts as an idea. The idea then becomes a document, or perhaps a prototype, depending on the method in use.

Softvér je komplexný produkt, ktorý je vyvíjaný a dodávaný zákazníkovi pomocou série krokov. Ako iné produkty aj vývoj softvéru začína s myšlienkou. Táto myšlienka sa postupne stáva dokumentom alebo prototypom, čo záleží na forme metódy ktorá sa na vývoj používa.

Whether a document, diagram, or working software, the artifact created in one step becomes the input to the next step. Eventually, the software is delivered to the customer. The sequence of steps used by these methods is commonly referred to as the Software Development Lifecycle (SDLC.)

Či už je to dokument, diagram alebo funkčný softvér, artefakt\* vytvorený v jednom kroku sa stáva vstupom do ďalšieho. Nakoniec je softvér dodaný zákazníkovi. Postupnosti krokov, použitých v týchto metódach sa spoločne hovorí Software Development Lifecycle (SDLC).

\*artefakt – je jedným z mnohých druhov vedľajších produktov vyrobených počas vývoja softvéru ako napríklad Unified Modeling Languare (UML) diagram.

The process of software development is a never-ending cycle. The first release of a software application is rarely “finished.” There are almost always additional features and bug fixes waiting to be designed, developed, and deployed.

Proces vývoja softvéru je nikdy nekončiaci cyklus. Prvé nasadenie softvérovej aplikácie je zriedkavo považované za dokončené. Takmer vždy sa vyskytnú chyby dizajnu alebo funkcie, čakajúce na opravu.

Reports from error monitoring software about usability and bugs feed back into the process of software development, and become new feature requests and improvements to existing features. This is why the Software Development Life Cycle is the most general term for software development methods. The steps of the process and their order vary by method. Regardless of method, they typically run in cycles, starting over with each iteration.

Chyby, ktoré boli zachytené používateľmi softvéru sa vracajú späť do procesu vývoja softvéru a stávajú sa novými požiadavkami na vylepšenie existujúcich funkcií. Práve z tohto dôvodu je SDLC najvšeobecnejší pojem pre metódy vývoja softvéru. Kroky procesu ako aj ich poradie je jedinečné pre každú metódu, majú však spoločné to, že zvyčajne bežia v cykloch, začínajú odznova s každým opakovaním.

Benefits of the SDLC

“If you fail to plan, you are planning to fail” - Benjamin Franklin

It’s very difficult to carry out a complex, team effort such as software development without some kind of plan. Each software development methodology (several will be detailed below) is a plan framework for how to develop software. There is much debate about which method is best overall, which is best suited to a particular type of software, and how to measure success in software development. One thing, however, is certain: any plan is better than no plan.

Výhody SDLC.

„ Ak zlyháš pri plánovaní, plánuješ zlyhať“ – Benjamin Franklin

Je veľmi náročné vykonávať komplexnú tímovú prácu ako je vývoj softvéru bez nejakého druhu plánu. Každá metóda vývoja softvéru má svoje výhody a nevýhody. Existuje veľa diskusií o tom, ktorá z nich je najvhodnejšia pre konkrétny typ softvéru a ako merať jej úspech. Jedna vec je však istá, akýkoľvek plán je lepší ako žiaden.

Without some kind of structured plan, software development teams tend to devolve into a “herd of cats.” Developers don’t know what they’re supposed to create. Project managers have no idea how much progress is made towards completion of a project. Without a plan, the business doesn’t even have a way to decide whether the final product meets their requirements.

Bez nejakého štrukturovaného plánu, majú tímy vývoja softvéru tendenciu prejsť do stavu, kedy nevedia ako budú pokračovať. Projektoví manažéri netušia aký veľký krok sa urobil k dosiahnutiu cieľa projektu. Spoločnosť sa bez plánu nemá ako uistiť, či konečný produkt spĺňa ich požiadavky.

A formally defined method for software development in the form of the SDLC achieves a number of benefits:

A common vocabulary for each step

Defined communication channels between development teams and stakeholders

Clear roles and responsibilities among developers, designers, business analysts, and project managers

Clearly-defined inputs and outputs from one step to the next

A deterministic “definition of done” that can be used to confirm whether a step is truly complete

The Seven Phases of the SDLC

Click here to jump to the 7 Phases of the SDLC infographic.

Formálne definovaná metóda pre vývoj softvéru vo forme SDLC dokáže zaistiť viacero výhod:

* Spoločný slovník pre každý krok
* Sú definované komunikačné kanály medzi vývojovým tímom a zúčastnenými stranami
* Jasne stanovená zodpovednosť medzi vývojármi, dizajnérmi, obchodnými analytikmi a projektovými manažérmi
* Jasne definované vstupné a výstupné parametre z jedného kroku do ďalšieho
* Deterministická definícia toho čo sa vykonalo, ktorá sa môže použiť na potvrdenie, či je krok skutočne dokončený

These steps are (very) roughly the same from one methodology to another. They tend to occur in this order, though they can also be mixed together, such that several steps occur in parallel.

Tieto kroky sú zhruba rovnaké medzi jednotlivými metódami. Majú tendenciu vyskytovať sa v tomto poradí aj keď sa môžu navzájom spájať tak, že sa niekoľko krokov vykonáva paralelne.

As we’ll discuss later, Agile methods tend to “wind together” all of these steps into a tight, rapidly-repeating cycle. Waterfall methods tend to take each of these steps in turn. Outputs from one become inputs to the following step.

Agilné metodiky majú tendenciu zoskupiť tieto kroky do opakujúceho sa cyklu. Vodopádová metóda zvykne podniknúť každý z týchto krokov postupne. Výstupy z jedného sa stanú vstupmi do nasledujúceho kroku.



1. Planning

The planning phase involves aspects of project and product management. This may include:

Capacity planning

Project scheduling

Cost estimation

Provisioning

Resource allocation (both human and materials)

The outputs of the planning phase include: project plans, schedules, cost estimations, and procurement requirements. Ideally, Project Managers and Development staff collaborate with Operations and Security teams to ensure all perspectives are represented.

1. **Plánovanie**

Fáza plánovania môže zahŕňať:

* Plánovanie kapacít
* Plánovanie projektu
* Odhad nákladov
* Prideľovanie zdrojov (ľudských aj materiálnych)

Medzi výsledky tejto plánovacej fázy patria: plán projektu, rozvrh, odhad nákladov a požiadavky na obstarávanie. V najlepšom prípade projektoví manažéri a vývojári spolupracujú so zástupcami plánovania a bezpečnosti aby zabezpečili zastúpenie všetkých perspektív.

2. Requirements

The business must communicate with IT teams to convey their requirements for new development and enhancement. The requirements phase gathers these requirements from business stakeholders and Subject Matter Experts (SMEs.)

Architects, Development teams, and Product Managers work with the SMEs to document the business processes that need to be automated through software. The output of this phase in a Waterfall project is usually a document that lists these requirements. Agile methods, by contrast, may produce a backlog of tasks to be performed.

1. **Požiadavky**

Zákazník musí komunikovať s IT tímom pri stanovení požiadaviek na vývoj a vylepšenie požadovaného softvéru. Fáza požiadaviek zhromažďuje tieto požiadavky od zákazníka a Subject Matter Experts (SME).

3. Design and prototyping

Once the requirements are understood, software architects and developers can begin to design the software. The design process uses established patterns for application architecture and software development. Architects may use an architecture framework such as TOGAF to compose an application from existing components, promoting reuse and standardization.

Developers use proven Design Patterns to solve algorithmic problems in a consistent way. This phase may also include some rapid prototyping, also known as a spike, to compare solutions to find the best fit. The output of this phase includes:

Design documents that list the patterns and components selected for the project

Code produced by spikes, used as a starting point for development

1. **Dizajn**

Keď už máme jasne stanovené požiadavky, architekti a vývojári môžu začať navrhovať softvér. Proces navrhovania používa stanovené vzory na vývoj aplikačnej architektúry a softvéru. Architekti môžu používať architektúru ako The Open Group Architecture Framework (TOGAF) na zostavenie aplikácie z existujúcich komponentov, podporu opätovného použitia a štandardizácie.

Vývojári používajú overené návrhové vzory na konzistentné riešenie algoritmických problémov. Táto fáza môže zahŕňať aj niekoľko techník Rapid prototypingu, taktiež nazývaných Spike, ktoré slúžia na porovnanie riešení s cieľom nájsť to najvhodnejšie.

Výstup z tejto fázy zahŕňa:

* Návrhové dokumenty v ktorých sú uvedené vzory a komponenty vybrané pre projekt
* Kód vytvorený technikou Spike, ktorý sa používa ako východisko pre vývoj

4. Software development

This phase produces the software under development. Depending on the methodology, this phase may be conducted in time-boxed “sprints,” (Agile) or may proceed as a single block of effort (Waterfall.) Regardless of methodology, development teams should produce working software as quickly as possible. Business stakeholders should be engaged regularly, to ensure that their expectations are being met. The output of this phase is testable, functional software.

1. **Vývoj softvéru**

Táto fáza vyprodukuje vyvíjaný softvér. V závislosti od metodológie môže byť táto fáza vykonávaná v časovo obmedzených šprintoch (Agilná metodika) alebo ako TODO. Bez ohľadu na metodiku by vývojové tímy mali čo najrýchlejšie vyprodukovať funkčný softvér. Zákazník by sa mal pravidelne zapájať do vývoju, aby sa splnili stanovené požiadavky. Výstupom z tejto fázy je funkčný testovateľný softvér.

5. Testing

The testing phase of the SDLC is arguably one of the most important. It is impossible to deliver quality software without testing. There is a wide variety of testing necessary to measure quality:

Code quality

Unit testing (functional tests)

Integration testing

Performance testing

Security testing

The best way to ensure that tests are run regularly, and never skipped for expediency, is to automate them. Tests can be automated using Continuous Integration tools, like Codeship, for example. The output of the testing phase is functional software, ready for deployment to a production environment.

1. **Testovanie**

Fáza testovania je v SDLC je pravdepodobne jednou z najdôležitejších, pretože bez riadneho testovania nie je možné dodať kvalitný softvér. Existuje široké spektrum testov potrebných na určenie kvality ako napríklad:

* Unit testovanie
* Integračné testovanie
* Testovanie výkonu
* Testovanie bezpečnosti

Najlepším spôsobom, ako zaistiť, aby sa testy nezanedbávali a vykonávali pravidelne, je ich automatizácia. Výstupom testovacej fázy je funkčný softvér pripravený na nasadenie do produkčného prostredia resp. k zákazníkovi.

6. Deployment

The deployment phase is, ideally, a highly automated phase. In high-maturity enterprises, this phase is almost invisible; software is deployed the instant it is ready. Enterprises with lower maturity, or in some highly regulated industries, the process involves some manual approvals. However, even in those cases it is best for the deployment itself to be fully automated in a continuous deployment model. Application Release Automation (ARA) tools are used in medium and large-size enterprises to automate the deployment of applications to Production environments. ARA systems are usually integrated with Continuous Integration tools. The output of this phase is the release to Production of working software.

1. **Nasadenie**

Fáza nasadenia je ideálne vysoko automatizovaná. Vo vyspelých spoločnostiach je táto fáza takmer neviditeľná, softvér je nasadený hneď ako je dokončený. V menej vyspelých spoločnostiach alebo v niektorých viac regulovaných odvetviach tento proces vyžaduje určité manuálne schválenia. Avšak aj v týchto prípadoch je najlepšie aby samotné nasadenie bolo plne automatizované. Nástroje Aplication Release Automation (ARA) sa používajú v stredných a veľkých podnikoch na automatizáciu nasadzovania aplikácií do produkčných prostredí. Systémy ARA sú zvyčajne integrované s nástrojmi na kontinuálnu integráciu. Výstupom z tejto fázy je nasadenie funkčného softvéru do produkčného prostredia.

7. Operations and maintenance

The operations and maintenance phase is the “end of the beginning,” so to speak. The Software Development Life Cycle doesn’t end here. Software must be monitored constantly to ensure proper operation. Bugs and defects discovered in Production must be reported and responded to, which often feeds work back into the process. Bug fixes may not flow through the entire cycle, however, at least an abbreviated process is necessary to ensure that the fix does not introduce other problems (known as a regression.)

1. **Údržba**

Fáza údržby je akoby začiatkom konca. Životný cyklus informačného systému tu nekončí. Softvér sa musí neustále monitorovať aby sa zabezpečila správna prevádzka. Chyby a nedostatky zistené v produkcií sa musia nahlásiť a spracovať, čo často vedie k tomu, že sa vyžaduje dodatočná práce na projekte. Oprava chýb nemusí pretekať celým cyklom, je však potrebný aspoň skrátený postup, aby sa zabezpečilo, že oprava nespôsobí ďalšie problémy známe ako regresia.

## Testovanie

Ako sa už spomínalo pri fáze testovania v SDLC, testovanie je jedným z najdôležitejších krokov k vývoju funkčného softvéru. Práve z tohto dôvodu existuje široké spektrum testov. <https://www.edureka.co/blog/functional-testing-vs-non-functional-testing/>



Typy softvérového testovania sa delia do dvoch základných skupín a to funkčné (functional) a ne-funkčné (non-functional) testovanie.

### Funkčné (functional) testovanie

Functional testing is a type of software testing where the system is tested against the functional requirements or specifications like the technical details, data manipulation and processing, and other specific functionalities. You might think, functional testing just refers to testing a function (method) of your module or any class but that’s not what it does. It tests a slice of the functionality of the whole system.

Funkčné testovanie je typ softvérového testovania, pri ktorom sa systém testuje na základe funkčných požiadaviek alebo špecifikácií, ako sú technické detaily, manipulácia a spracovanie údajov a ďalšie špecifické funkcie. Možno si myslíte, že funkčné testovanie sa týka iba testovania funkcie (metódy) nejakého modulu alebo triedy, čo ale nie je celkom pravda. Testuje sa, či správne funguje určitá časť celkového systému.

Types of Functional Testing

Let’s take a look at different types of functional testing. They are namely:

Unit testing

Integration testing

System testing

Interface testing

Regression testing

User-acceptance testing

Now, let’s see each one of them in detail

Unit testing: A unit is the smallest testable part of any software. It usually has one or a few inputs and a single output. It is a level of software testing where individual units/components of the software are tested. The main purpose is to validate each unit of the software, performs as designed.

**Typy funkčného testovania:**

**Unit testovanie:** Unit (jednotka?) je najmenšia možná testovateľná časť akéhokoľvek softvéru. Zvyčajne má jeden alebo viacero vstupných parametrov a jeden výstupný parameter. Je to úroveň softvérového testovanie, kde sú testované individuálne Unity (komponenty). Hlavným cieľom je overiť, či každý komponent softvéru pracuje tak ako by mal.

Integration testing: It is a level of software testing where individual units are combined and tested as a group. The main purpose of this type of testing is to expose faults in the interaction between integrated units.

**Integračné (integration) testovanie:** Je to úroveň softvérového testovania, kde individuálne Unity (jednotky) sú kombinované do testovacích skupín. Hlavným cieľom tohto typu testovania je odhaliť chyby pri interakcií medzi integrovanými Unitami.

System testing: It is a level of software testing where the complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. It is a series of different tests whose sole purpose is to exercise the system.

**Systémové (System) testovanie:** Je úroveň softvérového testovania, kde sa testuje celkový integrovaný softvér. Cieľom tohto testu je vyhodnotiť súlad systému so stanovenými požiadavkami. Je to séria rôznych testov, ktorých jediným účelom je precvičiť celkové fungovanie systému.

Interface testing: It is a level of software testing which verifies whether the communication between two different software is done correctly.

**Interface testovanie:** Je to úroveň softvérového testovania, ktorá overuje komunikáciu jedného softvéru s druhým.

Regression testing: Regression testing verifies that a code change in the software does not have an impact on the existing functionality of the product.

**Regresné (Regression) testovanie:** Overuje či zmeny v kóde nemajú dopad na existujúcu funkcionality produktu.

User-acceptance testing: It is a level of software testing where a system is tested for acceptability. The sole purpose of this test is to evaluate the system’s compliance with its business requirements and assess whether it is acceptable for delivery. It checks whether the software can handle the required tasks in real-world scenarios.

**User-acceptance testovanie:** Je úroveň softvérového testovania kde sa testuje prijateľnosť systému. Hlavným účelom tohto testu je vyhodnotiť súlad systému s jeho obchodnými požiadavkami a posúdiť, či je prijateľný na odovzdanie zákazníkovi. Kontroluje, či softvér dokáže zvládnuť požadované úlohy v scenároch s praktickým využitím.

Now, let’s see what is Non-functional testing and see what exactly it does to test a software.

### Ne-funkčné (Non-functional) testovanie

Non-functional Testing

Non-functional testing is another type of software testing which is used to check the non-functional aspects like performance, usability, reliability, etc. of a software application which are not tested using functional testing. Non-functional testing helps in testing the readiness of a system.

Ne-funkčné testovanie kontroluje ne-funkčné aspekty ako výkonnosť, využiteľnosť, spoľahlivosť atď. softvérovej aplikácie, ktoré sa netestujú vo funkčnom testovaní. Ne-funkčné testovanie pomáha overiť pripravenosť systému.

It defines the way a system operates, rather than specific behaviors of that system. This is totally in contrast to functional testing, which tests against functional requirements that describe the functions of a system. Basically, non-functional testing is done to check and evaluate all the non-functional parameters which are not covered under functional testing. All the non-functional parameters such as speed, scalability, security, reliability, and efficiency of an application are tested under the non-functional testing.

It makes an application robust and prepares it against certain vulnerabilities.

Definuje skôr spôsob fungovania systému ako jeho konkrétne správanie. Čo je opakom toho o čo sa snaží funkčné testovanie. V zásade, ne-funkčné testovanie vykonáva a vyhodnocuje všetky ne-funkčné parametre, ktoré nie sú zahrnuté do funkčného testovania, ako je rýchlosť, škálovateľnosť, bezpečnosť a efektívnosť aplikácie. Toto testovanie robí aplikáciu robustnou a pripraví ju TODO.

**Typy ne-funkčného (non-functional) testovania:**

Documentation testing: It helps to estimate the required testing efforts and track the requirements. Software documentation includes a test plan, test cases, and requirements section. It tests the documented artifacts.

**Dokumentačné (documentation) testovanie:** Pomáha odhadnúť požadované úsilie na testovanie a sledovať požiadavky. Softvérová dokumentácia zahŕňa testovací plán, testovacie scenáre a sekciu požiadaviek. Testuje zdokumentované artefakty.

Installation testing: It is a type of quality assurance work in the software industry that converges on what customers will need to do to install and set up the new software successfully. It checks if the software application is successfully installed and is working as expected. The testing process may involve full, partial or upgrades install/uninstall processes.

**Inštalačné (installation) testovanie:** Je typ práce pri zabezpečovaní kvality v softvérovom priemysle, ktorá sa zameriava na to, čo budú zákazníci musieť urobiť, aby mohli úspešne nainštalovať a nastaviť nový softvér. Kontroluje, či je aplikácia úspešne nainštalovaná a či pracuje podľa očakávaní. Tento proces môže obsahovať úplné alebo čiastočné inštalácie. TODO

Performance testing: It is defined as a type of software testing that is used to ensure that the software applications will perform well under their expected workload. Performance testing is considered as a heart of Non-functional testing. It is further divided into a few types.

**Výkonnostné (performance) testovanie:** Je definované ako typ softvérového testovania, ktoré je používané na zabezpečenie toho, aby aplikácie pracovali dobre pri očakávanom pracovnom zaťažení. Výkonnostné testovanie sa považuje za srdce ne-funkčného testovania. Ďalej sa delí na niekoľko typov.

Load Testing is a type of performance testing conducted to evaluate the behavior of a system at an increasing workload.

Stress Testing is a type of performance testing conducted to evaluate the behavior of a system at or beyond the limits of its anticipated workload.

Endurance Testing is a type of performance testing conducted to evaluate the behavior of a system when a significant workload is given continuously.

Spike Testing is a type of performance testing conducted to evaluate the behavior of a system when the load is suddenly and substantially increased.

* Záťažové (load) testovanie - vyhodnocuje správanie systému pri postupnom zvyšujúcom sa pracovnom zaťažení.
* Stress (dôrazové) testovanie – vyhodnocuje správanie systému na hranici alebo za hranicou zvládnuteľného pracovného zaťaženia.
* Endurance (vytrvalostné) testovanie - vyhodnocuje správanie systému pri veľkej dlho trvajúcej pracovnej záťaži.
* Spike testovanie – vyhodnocuje správanie systému pri náhlom zvýšení pracovnej záťaže.

Reliability testing: It assures that the product is fault free and is reliable for its intended purpose. It is about exercising an application so that the failures are discovered before the system is deployed.

**Reliability testovanie (spoľahlivosti)** – zaručuje, že produkt bude bezchybný a spoľahlivý pre v tom na čo bol navrhnutý. Ide o testovanie aplikácie tak, aby sa chyby objavili pred zavedením systému

Security testing: It is a variant of software testing which ensures that the system and applications of the organization are free from loopholes. Security testing is about finding all possible weaknesses of the system which might result in a loss of information.

This is all about the different types of Non-functional testing. Now let us move forward and compare functional & non-functional testing on various parameters.

### Automatizované testovanie <https://testguild.com/automation-testing/>

Automatizované testovanie je technika, ktorá sa používa na zvýšenie rýchlosti overovania alebo iných opakovaných úloh v životnom cykle informačného systému. Z tohto dôvodu, aby ušetrili čas, sa veľa spoločností snaží zobrať manuálne testovacie scenáre a previesť ich na automatizované testovacie scenáre.

Automatizovaný testovací nástroj potom vykoná kroky testovacieho scenára automaticky bez ľudského pričinenia.

Also, automated tools use a programming approach to emulate a usser interacting with an application and verifying test steps using programming assertions.

 Taktiež automatizovaný testovací nástroj používa programátorský prístup aby napodobnil interakciu používateľa s aplikáciou a overil kroky testu použitím programátorských výrokov (tvrdení)??!!.

Prečo testovať automaticky? (<https://testguild.com/automation-testing/>) Môžu byť otázky?

So zvýšenou rýchlosťou, ktorou sa softvér v dnešnej dobe vyvíja, je automatické testovanie nevyhnutnou súčasťou vývoja nového softvéru. Praktiky ako priebežná integrácia, vývojárska praktika ktorá od vývojárov vyžaduje integrovať kód do zdieľaného úložiska niekoľko krát denne, vyžadujú testy ktoré bežia rýchlo a spoľahlivo. Veľa manuálneho overovania zabráni dosiahnutiu požadovanej rýchlosti vývoja softvéru.

Je takmer isté, že v dnešnom modernom vývojovom prostredí, by sme bez automatizovania neuspeli.

Hoci hlavným dôvodom prečo sa tímy snažia testovať automatizovane je aby ušetrili spoločnosti čas a peniaze, dôležité je tiež dodať vývojárom spätnú väzbu aby boli pri odovzdaní kódu čo najskôr informovaný o tom, že zmena kódu ktorú odovzdali niečo pokazila. Ďalšie dôvody pre automatizované testovanie sú:

* Overenie novších verzií softvéru
* Oslobodenie testerov od triviálnych scenárov aby sa mohli sústrediť na komplexnejšie testovacie scenáre
* Väčšie pokrytie testov
* Znovupoužiteľnosť (Reusability ) (Neviem či vhodný preklad)
* Rýchlejšie nasadenie softvéru
* Šetrí čas
* Dodáva rýchlu spätnú väzbu pre vývojárov ohľadom chýb v softvéri

Teória je taká, že automatizované testy ušetria spoločnostiam čas a peniaze. Vyzerá to však tak, že veľa ľudí neberie do úvahy čas a peniaze, ktoré treba investovať do udržania stabilného testovacieho systému.

Existujú teda nevýhody pri vytváraní automatizovaných testov? (?)

Keďže na tvorbu automatických testov sa používajú programovacie jazyky, automatizácia sa stáva samostatným projektom na vývoj. Čo sa v skutočnosti robí je, že sa vyvíja časť softvéru na testovanie inej časti softvéru, preto je potrebné investovať rovnaké úsilie do vývoju automatických testov. Dodržujú sa rovnaké procesy a osvedčené postupy ako by sa použili pri vývoji akéhokoľvek iného projektu.

Automatizované testovanie je náročné, presne ako väčšina ostatných projektov na vývoj softvéru. Taktiež prezentuje veľa rovnakých problémov výziev. Ak sa k automatizovanému testovaniu bude pristupovať ako k druhotriednemu vývoju, bude mať problémy s údržbou a spoľahlivosťou z dlhodobého hľadiska.

Proces automatizovaného testovania sa dá zhrnúť do týchto krokov:



**Príprava**

Najskôr je potrebné pripraviť sa a pochopiť ciele funkčného testovania, aké testovacie dáta sú potrebné a čo treba overiť.

**Písanie**

Zmena požiadaviek na automatizované riešenie. Určenie počiatočného a koncového stavu pre každý test. Testy by mali byť úplne nezávislé jeden od druhého. Je potrebné pridať dostatočné kontroly na uistenie sa, že sa aplikácia správa podľa zadaných špecifikácií. Každý test by mal mať konkrétny účel.

**Realizácia**

Realizácia testov by mala byť spoľahlivá. Odporúča sa spustiť každý test aspoň tri krát za sebou predtým ako sa skontroluje kód.

**Vyhodnotenie**

Overenie, či automatizované testy robia to, čo sa od nich čaká. Pomocou manuálnych testov sa overí, či sa testy správajú podľa požiadaviek.

**Komunikácia**

Každý člen tímu by mal byť informovaný o výsledkoch testov. Pochybné testy by mali byť opravené čo najskôr, inak vzniká riziko, že tím bude výsledky testov ignorovať.

**Opakovanie/Spravovanie**

Ak sa ukáže pochybný test, je potrebné ho prerobiť aby sa stal viac spoľahlivým. Najdôležitejšie je však vymazať akékoľvek testy, ktoré nie sú spoľahlivé a neboli opravené v danom časovom rámci.

Automatické testovanie je teda využívané pre uľahčenie práce vývojárov a odľahčenie záťaže na testovacie oddelenie. Existujú však prípady, kedy automatické testovanie nedokáže pokryť všetky scenáre a musia sa testovať manuálne. Preto pri vývoji nového softvéru vo firmách, ktoré nemajú testovací tím, robia vývojári manuálne testy predtým ako sa nový softvér nasadí u zákazníka.

### Manuálne testovanie

Existujú scenáre, ktoré nie je možné vykonať automaticky alebo čas strávený vývojom takéhoto scenára by presiahol jeho užitočnosť. Jedná sa väčšinou o kód Graphical user interface (GUI), kde by vývoj automatického scenáru takmer s istotou pôvodný presiahol čas strávený vývojom GUI. V takomto prípade testuje scenáre testovací tím alebo samotný vývojár, ktorý kód napísal.

V priebehu regresného testovania, zisťovania či nedávne zmeny v kóde neporušili alebo nezničili už existujúcu funkcionalitu systému, zoberie manuálny tester existujúci testovací scenár pracovného postupu a krok po kroku ho vykoná.

Je však niekoľko dôvodov prečo môže byť manuálne testovanie problematické:

* Používa veľa prostriedkov – na manuálne testovanie potrebujeme testovací tím, ktorý môže byť finančne náročný
* Je časovo náročné – celkové pokrytie testovacích scenárov manuálne zaberá veľa času
* Niekedy nemusí pokrývať celý testovaný scenár – určité scenáre vyžadujú schopnosti programátorov
* Kvôli opakujúcej sa povahe, testeri môžu pri manuálnom vykonávaní zanedbať alebo prehliadnuť niektoré dôležité kroky čo môže viesť k nekonzistentnému stavu testu.

### SAP

What is SAP? <https://www.guru99.com/what-is-sap-definition-of-sap-erp-software.html>

Je to európska medzinárodná spoločnosť založená Dietmarom Hoppom, Hans-Wernerom Hectorom, Hassom Plattnerom, Klausom Tschiraom a Clausom Wellenreutherom v roku 1972. Celosvetovo od roku 2010 má SAP viac ako 140 000 inštalácií, viac než 25 podnikových riešení pre špecifické priemyselné odvetvia a viac ako 75 000 zákazníkov v 120 krajinách.

SAP je skratka pre System aplications and Products je jednotka na ERP trhu v spracovaní dát. Je to štvrtá najväčšia softvérová spoločnosť na svete. SAP R/3 systém je balík biznis softvéru, ktorý je navrhnutý pre integráciu všetkých oblastí podnikania Spoločnosť vyvíja softvérové riešenia pre riadenie obchodných operácií a vzťahov so zákazníkmi.

Po veľkom úspechu R/3, vytvoril SAP AG viac a viac niche\* (Ako sa postaviť k slovíčkam ktoré je vhodné nepreložiť resp. spraviť nejakú legendu) softvér ako napríklad Customer Relationship Management (CRM), SRM, XI (aktuálne nazývaný aj ako Process Integration skratkou PI) čím sa znova ukázala kvalita softvéru tejto spoločnosti.

Všetky podnikové procesy sú vykonávané v jednom SAP systéme a zdieľajú spoločné informácie.

<https://searchsap.techtarget.com/definition/SAP?fbclid=IwAR22o4XxOhTOdOar3gePgFADmEt3Wb_kbEV4HqH7u1ItcwA3OhA8khKCV2I>

Odhaduje sa, že 77% peňazí vymieňaných prostredníctvom globálnych obchodných transakcií sa dostane do kontaktu so systémom SAP. Väčšinu zákazníkov tvoria malé a stredné podniky. Spoločnosť ponúka lokálne, cloudové a hybridné modely na nasadenie s tým, že možnosť cloud computing je to na čo sa chce spoločnosť v budúcnosti sústrediť.

V roku 2011, spoločnosť uviedla na trh SAP HANA, in-memory\* databázovú platformu, ktorá je na čele budúcej stratégie spoločnosti. HANA bola hlavným projektom pre SAP, ktorý uviedol, že má v úmysle nahradiť databázou HANA tradičné databázy, ktoré SAP používa pre svoje obchodné aplikácie.

**Enterprise resource planning (ERP)**

SAP SE je jedným z najväčších dodávateľov softvéru pre Enterprise Resource Planing \*(ERP) a súvisiacich podnikových aplikácií. ERP systém spoločnosti umožňuje zákazníkom viesť ich obchodné procesy vrátane účtovníctva, predaja, výroby, ľudských zdrojov a financií v integrovanom prostredí. Integrácia zabezpečuje tok informácií z jedného komponentu SAPu (skloňovanie cudzích slov môže byť?) do druhého bez potreby redundantného\* zadávania údajov a pomáha vymáhať finančné, procesné a právne kontroly. Umožňuje tiež efektívne využívanie zdrojov, vrátane pracovnej sily, strojov a výrobných kapacít.

SAP ERP systém, nazývaný SAP ERP Central Component (SAP ECC), je spoločný termín pre funkčné a technické moduly spoločnosti SAP, ktoré umožňujú podnikom riadiť obchodné procesy prostredníctvom zjednoteného systému. ECC je miestna verzia SAP, ktorá je zvyčajne implementovaná v stredných a veľkých spoločnostiach. Pre menšie spoločnosti ponúka SAP svoju Business One ERP platformu.

SAP ERP má rozdielne hlavné moduly, ktoré sú rozdelené do funkčných a technických modulov, z ktorých každý má submoduly\*.

Funkčné moduly SAP systému zahŕňajú:

* Human Capital Management ([SAP HCM](https://searchhrsoftware.techtarget.com/definition/human-capital-management-HCM))
* Production Planning ([SAP PP](https://searchsap.techtarget.com/definition/SAP-Production-Planning))
* Materials Management ([SAP MM](https://searchsap.techtarget.com/definition/SAP-Materials-Management-MM))
* Project System ([SAP PS](https://searchsap.techtarget.com/definition/SAP-Project-System-PS))
* Sales and Distribution ([SAP SD](https://searchsap.techtarget.com/definition/SAP-Sales-and-Distribution-SAP-SD))
* Plant Maintenance ([SAP PM](https://searchsap.techtarget.com/definition/SAP-Plant-Maintenance-PM))
* Finance and controlling ([SAP FICO](https://searchsap.techtarget.com/definition/SAP-FICO-SAP-Finance-and-SAP-Controlling))
* Quality Management ([SAP QM](https://searchsap.techtarget.com/definition/SAP-Quality-Management-QM))

**História spoločnosti SAP**

SAP začal v roku 1972 piatimi bývalými zamestnancami IBM v Mannheime v Nemecku. Pôvodným cieľom pre SAP bolo poskytnúť zamestnancom možnosť využiť bežnú korporátnu\* databázu na komplexný rozsah aplikácií v reálnom čase.

V roku 1973, SAP vydal R/1, systém finančného účtovníctva. R/1 bežal na IBM serveroch a diskovom operačnom systéme (DOS). Mal jednostupňovú architektúru, v ktorej boli prezentačná, aplikačná a databázová vrstva na jednej platforme.

V roku 1979, SAP vydal R/2, systém s centrálnym počítačom (mainframe), ktorý poskytoval spracovanie údajov v reálnom čase v rámci účtovníctva, výroby, zásobovacieho reťazca a ľudských zdrojov. R/2 používal dvojvrstvovú architektúru, kde prezentačná vrstva bola na jednej platforme a aplikačná a databázová vrstva boli na druhej. R/2 pomohlo podporiť rast spoločnosti SAP a predajca rozšíril svoju klientelu na približne 200 spoločností.

V roku 1992, SAP vydal R/3, ktorý reprezentoval zmenu z \*mainframe na klient-server model a z dvojvrstvovej architektúry na trojvrstvovú architektúru, v ktorej prezentačná, aplikačná a databázová vrstva boli umiestnené osobitne?? R/3 bol kritický produkt pre SAP, ktorý preniesol túto spoločnosť na svetovú scénu.



V roku 1999, SAP uviedol na trh mySAP, čo pre spoločnosť znamenalo novú stratégiu zameranú na kombináciu elektronického obchodu s aplikáciami v R/3. Jeden rok po vydaní R/3 sa spoločnosť SAP spojila so spoločnosťou Microsoft, aby novú verziu priniesla na Windows NT. Do roku 1997 zamestnávala spoločnosť SAP 13 000 ľudí.

In 2004, the company launched SAP [NetWeaver](https://searchsap.techtarget.com/definition/NetWeaver) which received a lot of industry attention as the first fully interoperable, web-based, cross-application platform that could be used to develop not only SAP applications, but others, as well. SAP SE reported that more than 1,000 customers acquired the application development platform that year. Also in 2004, the successor to R/3, the SAP ERP system (or SAP ECC, for SAP ERP Central Component) was released. Customers already using R/2 or R/3 were still supported, but new customers were required to implement SAP ERP.

V roku 2004 spoločnosť uviedla na trh SAP NetWeaver, ktorý získal veľkú pozornosť ako prvá plne interoperabilná\* webová platforma s krížovými aplikáciami, ktorá sa mohla použiť nie len na vývoj SAP aplikácií. SAP SE informovala, že v tomto roku získalo platformu pre vývoj aplikácií viac ako 1000 zákazníkov. Taktiež v tomto roku bol nasadený nasledovník R/3, SAP ERP systém, inak SAP ECC pre SAP ERP centrálny komponent. Zákazníci, ktorí už využívali R/2 alebo R/3 systém mali stále podporu spoločnosť, ale noví zákazníci boli povinný použiť SAP ERP systém.

Spoločnosť SAP tvrdí, že jej primárne zameranie na rast spočíva na interných inováciách prostredníctvom vývoja a zdokonaľovania vlastných produktov. Ako krok týmto smerom spoločnosť vytvorila SAP Labs, ktoré vyvíjajú a vylepšujú základné produkty. Tieto sa nachádzajú v klastroch\* vyspelých technológií na celom svete, napríklad v Bangalore v Indii a v Palto Alto v Kalifornii.

Since 1996, the company has also made more than 60 acquisitions. A major focus for the company in recent years has been building its cloud computing capabilities and enabling greater mobility. Acquiring companies with such technologies has helped to build those capabilities. A few acquisitions that serve as examples are:

Od roku 1996, spoločnosť taktiež uskutočnila viac ako 60 akvizícií\*. Hlavným cieľom spoločnosť v posledných rokoch bolo budovanie schopností v rámci internetového počítania (cloud computing\*), čo by umožnilo väčšiu mobilitu. Získanie spoločností, ktoré s týmito technológiami pracujú, pomohlo vybudovať tieto schopnosti. Niekoľko akvizícií, ktoré slúžia ako príklady sú:

* Qualtrics, 2018, manažment skúseností, doteraz druhá najväčšia akvizícia
* CallidusCloyd, 2018, riadenie výkonnosti obchodného tímu
* Concur Technologies, 2014, softvér na správu cestovania a výdavkov ako online služba, doteraz najväčšia akvizícia
* [Fieldglass](https://searchsap.techtarget.com/definition/Fieldglass), 2014, cloudová kontingentná práca a služby
* Hybris, 2013, e-commerce, súčasť balíka SAP [Customer Engagement and Commerce](https://searchcustomerexperience.techtarget.com/definition/SAP-Customer-Experience-Suite-CEC-Suite) suite
* [Ariba](https://searchsap.techtarget.com/definition/Ariba-Ariba-Network), 2012, cloudový B2B\* trh
* [SuccessFactors](https://searchsap.techtarget.com/definition/SAP-SuccessFactors), 2011, správa ľudského kapitálu v cloude
* [BusinessObjects](https://searchsap.techtarget.com/definition/SAP-BusinessObjects-BI), 2007, obchodné spravodajstvo

Externé úložiská

K tejto časti si nie som istý, či by sme to mali dať do teórie alebo do praxe, keďže o týchto úložiskách sa dá písať dosť veľa pričom 1. bod zadania témy je analýza externých úložísk. Tak do teórie by sme mohli dať len niečo všeobecne o externých úložiskách a do praxe analýzu a porovnanie konkrétne menovaných úložísk v téme.

V dnešnej digitálnej dobe takmer každý používa nejakú formu externého úložiska. Či sa jedná o externý pevný disk, jednotku USB flash alebo nejakú formu clouduového úložiska ako Google Drive alebo Microsoft One Drive.

Úložiská som vybral podľa zadania témy.

**Apache Hive** <https://aws.amazon.com/big-data/what-is-hive/>

Apache Hive je distribuovaný systém skladovania údajov, odolný voči poruchám, ktorý umožňuje analýzu v rozsiahlom merítku. Dátový sklad poskytuje centrálne úložisko informácií, ktoré možno ľahko analyzovať, aby sa dali robiť rozhodnutia založené na informáciách z dát. Hive umožňuje používateľom čítať, zapisovať a spravovať petabyty dát s využitím SQL.

Hive je založený na Apache Hadoop, čo je open-source rámec (framework), ktorý sa používa na efektívne ukladanie a spracovanie veľkého množstva údajov. Výsledkom je, že Hive je úzko integrovaný s Hadoopom (skloňovanie?) a je navrhnutý tak, aby dokázal rýchlo spracovať petabyty dát. Čo robí Hive unikátnym je jeho schopnosť dotazovať sa na veľké množstvá údajov pomocou Apache Tez alebo MapReduce cez rozhranie podobnému Structured Query Language (SQL).

Bol vytvorený, aby umožnil ľuďom, ktorí nie sú programátori ale sú oboznámení s SQL, pracovať s petabajtami dát pomocou rozhrania podobného SQL s názvom HiveQL. Tradičné relačné databázy sú určené na malé a stredné množiny údajov ale nespracovávajú dobre obrovské množiny údajov. Hive namiesto toho používa dávkové spracovanie, takže pracuje rýchlo vo veľmi veľkej distribuovanej databáze. Hive transformuje dotazy (query, preložiť?) HiveQL na úlohy MapReduce alebo Tez, ktoré sú spustené v rámci distribuovaného plánovania úloh Apache Hadoop, Yet Another Resource Negotiator (YARN). Dopytuje údaje uložené v distribuovanom úložisku, ako Hadoop Distributed File System (HDFS) alebo Amazon S3. Hive ukladá svoje databázové a tabuľkové metadata (metaúdaje/metadáta, preložiť? ) do metastoru, čo je databázou alebo súborom zálohovaný sklad, ktorý umožňuje ľahkú abstrakciu a zisťovanie údajov.

Výhody Hive

* Rýchlosť – je navrhnutý na rýchle spracovanie petabajtov údajov pomocou dávkového spracovania
* Dobre známe prostredie (preklad??) – poskytuje povedom, SQL-like rozhranie, ktoré je prístupné aj ľudom, ktorí nie sú programátori
* Škálovateľnosť – dá sa ľahko distribuovať a meniť podľa potrieb

FAST

Hive is designed to quickly handle petabytes of data using batch processing.

FAMILIAR

Hive provides a familiar, SQL-like interface that is accessible to non-programmers.

SCALABLE

Hive is easy to distribute and scale based on your needs.

Hive includes HCatalog, which is a table and storage management layer that reads data from the Hive metastore to facilitate seamless integration between Hive, Apache Pig, and MapReduce. By using the metastore, HCatalog allows Pig and MapReduce to use the same data structures as Hive, so that the metadata doesn’t have to be redefined for each engine. Custom applications or third party integrations can use WebHCat, which is a RESTful API for HCatalog to access and reuse Hive metadata.

**AWS Redshift** <https://www.techradar.com/news/what-is-amazon-redshift>

When it comes to [cloud computing services](https://www.techradar.com/best/best-cloud-computing-services" \t "_blank), Amazon Redshift is a powerhouse of data warehousing. Used by some of the largest companies in the world, including Ford Motor Company, Lyft, Intuit, and Pfizer plus countless more, the data warehouse is used to store [cloud databases](https://www.techradar.com/best/best-cloud-databases" \t "_blank) and the related production data. The concept of Big Data hinges on an ability to process, store, and analyze data in vast treasure troves, and that is essentially what Amazon Redshift provides.

More than almost any other product, Amazon Redshift has powered the advent of Big Data and data warehousing, allowing companies to build powerful applications and generate reports that provide all of the data they need to run a business.

The best way to understand Amazon Redshift is to start with some basic terms and what they mean, and how this adds up to a powerful, fast, and extensive data warehouse product. Once you understand the terms you can start seeing the benefits of the product.

One of the first things to understand about Amazon Redshift is that you can start small. Any company can sign up for a single node that allows you to store a database and the related data, and then to start running queries and reports on that data (and run your own custom applications). The first node you create is called the leader node. If you add more, they are called a compute node. You could define Amazon Redshift as a cluster of nodes.

Of course, it is far more complex than that -- Redshift forms the basis for a collection of cloud computing products that are part of Amazon Web Services. For the [cloud storage](https://www.techradar.com/news/the-best-cloud-storage" \t "_blank) component, there is Amazon S3 (or Amazon Simple Storage Service) which provides the object storage itself.

That said, many companies start with one node when they start a data warehousing initiative. As your data warehousing needs expand and change, you can add more nodes into a cluster. This helps you build more applications, run more queries, and perform more [analytics](https://www.techradar.com/best/best-cloud-analytics" \t "_blank). Pricing can depend on how long you want to keep those nodes active. The prices go down when you reserve nodes for longer periods of time, such as one year or three years.

Beyond that, the other important thing to know about Amazon Redshift is that most of the complexity occurs behind the scenes. This includes the [endpoint security](https://www.techradar.com/news/best-endpoint-security-software" \t "_blank), management, scaling, provisioning, and anything else related to data warehousing. There’s a web console your [IT service management](https://www.techradar.com/best/best-itsm-tools) team uses to deal with instances and to create new nodes, but you don’t have to plan or manage any of the performance characteristics, the back-ups or archives, or any of the [infrastructure management](https://www.techradar.com/best/best-infrastructure-management-service" \t "_blank) related to the database or the data, including the servers or networks.

Amazon recently announced some improvements to Redshift. One of the key changes is that the nodes you use can be optimized separately for performance or storage. Previously, a cluster was maintained for both performance and storage allocation. Amazon also improved networking speed, especially the connections between Redshift and Amazon S3. Amazon claims the Redshift now delivers 3x the performance of competing data warehouse products.

[What is IaaS? Everything you need to know](https://www.techradar.com/news/what-is-infrastructure-as-a-service)

[What is PaaS? Everything you need to know](https://www.techradar.com/news/what-is-platform-as-a-service)

[Data lakes explained](https://www.techradar.com/news/what-is-a-data-lake)

Benefits of Redshift

As with any cloud computing initiative, the reason to use Amazon Redshift has to do with flexibility. As mentioned previously, companies can choose to create a single node as a starting point, but from there they can create massive clusters containing many nodes for every reporting need they have any for any web application. To say the possibilities are endless in terms of database control is not quite true but with cloud computing, it will seem like it.

Another benefit beyond flexibility in terms of what you can do and the applications you run, there is also an advantage to how it is all managed. Your Information Technology staff do not need to manage the cloud computing infrastructure at all, nor do they need to manage the servers, networks, or storage required. Since it is all in the cloud, and it is all part of Amazon Web Services (or AWS), it is all managed remotely and updated automatically.

One last benefit to consider is that Amazon Redshift provides the framework for a company to go beyond its current limitations. This might be a new application that uses a database in the cloud (and data stored in the cloud), or it might be a new way to analyze company data. Some firms even create brand new divisions and departments based on their newfound capability to understand and process data. An example of this might be an automotive manufacturer who has the ability to analyze data in real-time and develop autonomous driving features.

In the end, the power of Amazon Redshift is only limited by the imagination of the company starting a new initiative, developing a new product, or forming a new division.

**Oracle** <https://www.oracle.com/storage/>

Why Choose Oracle Engineered Storage?

Oracle engineered storage solutions transcend traditional storage offerings, helping your applications and databases to run faster. Automated storage management simplifies operations and reliably protects and recovers mission-critical Oracle Databases with zero data-loss capabilities. Oracle engineered storage solutions are architected, tested, and optimized with the complete Oracle technology stack so you don’t have to do the integration yourself.

Integration

Utilize unique capabilities.

Only Oracle can fully integrate storage resources with Oracle Database, yielding unmatched efficiency, recoverability to any point in time, and management automation.

Performance

Accelerate critical workloads.

High-throughput architectures with built-in Oracle Database optimizations deliver extreme performance and security for dynamic, multiapplication workloads.

Protection

Recover any data.

Only Oracle’s Zero Data Loss Recovery Appliance was developed as an Oracle Database extension that continuously protects and validates data, reports on recoverability status, and automates recovery from outages to any point in time.

Value

Reduce TCO.

Oracle engineered storage reduces database storage capacity requirements by up to 50 percent, enables 4:1 storage consolidation, and automates up to 90 percent of routine setup and tuning tasks.

**MSSQL Server** <https://searchsqlserver.techtarget.com/definition/SQL-Server>

Microsoft SQL Server is a relational database management system (RDBMS) that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments. Microsoft SQL Server is one of the three market-leading database technologies, along with Oracle Database and IBM's [DB2](https://searchdatacenter.techtarget.com/definition/DB2).

Inside SQL Server's architecture—How SQL Server works

Like other RDBMS technologies, SQL Server is primarily built around a row-based table structure that connects related data elements in different tables to one another, avoiding the need to redundantly store data in multiple places within a database. The relational model also provides referential integrity and other integrity constraints to maintain data accuracy. Those checks are part of a broader adherence to the principles of atomicity, consistency, isolation and durability, collectively known as the [ACID properties](https://searchsqlserver.techtarget.com/definition/ACID), and are designed to guarantee that database transactions are processed reliably.

The core component of Microsoft SQL Server is the SQL Server Database Engine, which controls data storage, processing and security. It includes a relational engine that processes commands and queries and a storage engine that manages database files, tables, pages, indexes, data buffers and transactions. Stored procedures, triggers, views and other database objects are also created and executed by the Database Engine.

Sitting beneath the Database Engine is the SQL Server Operating System, or SQLOS. SQLOS handles lower-level functions, such as memory and I/O management, job scheduling and locking of data to avoid conflicting updates. A network interface layer sits above the Database Engine and uses Microsoft's Tabular Data Stream protocol to facilitate request and response interactions with database servers. And at the user level, SQL Server DBAs and developers write T-SQL statements to build and modify database structures, manipulate data, implement security protections and back up databases, among other tasks.QL Server services, tools and editions

Microsoft also bundles a variety of data management, business intelligence ([BI](https://searchbusinessanalytics.techtarget.com/definition/business-intelligence-BI)) and analytics tools with SQL Server. In addition to the R Services and now [Machine Learning](https://searchenterpriseai.techtarget.com/definition/machine-learning-ML) Services technology that first appeared in SQL Server 2016, the data analysis offerings include SQL Server Analysis Services, an analytical engine that processes data for use in BI and data visualization applications, and SQL Server Reporting Services, which supports the creation and delivery of BI reports.

On the data management side, Microsoft SQL Server includes SQL Server Integration Services, SQL Server Data Quality Services and SQL Server Master Data Services. Also bundled with the DBMS are two sets of tools for DBAs and developers: SQL Server Data Tools, for use in developing databases, and [SQL Server Management Studio](https://searchsqlserver.techtarget.com/definition/Microsoft-SQL-Server-Management-Studio-SSMS), for use in deploying, monitoring and managing databases.

SQL Server editions:

Microsoft offers SQL Server in four primary editions that provide different levels of the bundled services. Two are available free of charge: a full-featured Developer edition for use in database development and testing, and an Express edition that can be used to run small databases with up to 10 GB of disk storage capacity. For larger applications, Microsoft sells an Enterprise edition that includes all of SQL Server's features, as well as a Standard one with a partial feature set and limits on the number of processor cores and memory sizes that users can configure in their database servers.

Other editions include a Standard version, Web version and a Business Intelligence version. The Business Intelligence version provides Self Service and Corporate Business Intelligence tools with the Standard Edition capabilities.

However, when SQL Server 2016 Service Pack 1 (SP1) was released in late 2016, Microsoft made some of the features previously limited to the Enterprise edition available as part of the Standard and Express ones. That included In-Memory OLTP, PolyBase, columnstore indexes, and partitioning, data compression and change data capture capabilities for [data warehouses](https://searchdatamanagement.techtarget.com/definition/data-warehouse), as well as several security features. In addition, the company implemented a consistent programming model across the different editions with SQL Server 2016 SP1, making it easier to scale up applications from one edition to another.

# Ciele práce

Cieľom bakalárskej práce je vytvoriť funkčný systém automatizovaných testov, pomocou ktorého sa bude kontrolovať komunikácia systému SAP s vybranými externými úložiskami dát (Apache Hive, AWS Redshift, Oracle , MSSQL)

Obsah práce možno zhrnúť nasledovne:

1. Analýza vybraných úložísk a identifikácia vhodných scenárov na automatizáciu testovania,

2. Zber a vyhodnotenie požiadaviek na vytváraný systém automatického testovania,

3. Návrh a implementácia softvérového riešenia v jazyku ABAP tak, aby sa jednotlivé testy vykonávali vo vopred určenom čase a výsledky testov aby sa uchovávali na vopred definovanom mieste a v presne určenej forme,

4. Návrh a implementácia testov, tvorba testovacích scenárov,

5. Vyhodnotenie dosiahnutých výsledkov,

6. Dokumentácia riešenia vo forme UML.

# [Praktická](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Ciele_práce) časť

## Analýza externých úložísk

## Zber a vyhodnotenie požiadaviek

## Návrh automatizovaného testovacieho systému

### Architektúra

## Implementácia testovacieho systému

## Vyhodnotenie dosiahnutých výsledkov

# [Metodika práce a metódy skúmania](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Metodika_práce)

# [Výsledky práce a diskusia](C:\\Users\\DonChameleón\\Desktop\\Pokyny_pre_vypracovanie_ZP.docx" \l "Výsledky_práce)

## Výsledky práce

Podrobný popis postupov podľa metodiky riešenia.

## Diskusia

Výsledky, ktoré boli dosiahnuté riešením.

# [Záver](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Záver)

# [Zoznam použitej literatúry](file:///C:\Users\DonChameleón\Desktop\Pokyny_pre_vypracovanie_ZP.docx#Zoznam_použitej_literatúry)

# [Zoznam príloh](C:\\Users\\DonChameleón\\Desktop\\Pokyny_pre_vypracovanie_ZP.docx" \l "Prílohy)

**Príloha A** Názov

**Príloha B** Názov

# 

# Prílohy

## Príloha A: Názov prílohy

Každá ďalšia príloha začína na novej strane.

## Príloha B: Obsah DVD

Priložené DVD obsahuje:

* Práca v elektronickej podobe (formát PDF)