Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Analýza a návrh informačního systému v UML pro společnost GHC Genetics

Bc. Sakun Aleksandr

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Aleksandr Sakun

Systémové inženýrství a informatika
Informatika

Název práce

Analýza a návrh informačního systému v UML pro společnost GHC Genetics

Název anglicky

Analysis and design of information system in UML for company GHC Genetics

---

Cíle práce

Cílem diplomové práce je analyzovat a navrhnout funkční model informačního systému pomocí UML. Tento model umožňuje efektivně spravovat a řídit chod firmy, pomoci sběru, uchování, přenosu a zpracování dat.
Dílčí cíle diplomové práce jsou:
- provést rešerši v oblasti návrhu informačních systémů
- sběr a analýza požadavků
- navrhnout objektový model, stavový model a model interakcí

Metodika

Metodika diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Pomoci získaných poznatků bude vytvořen informační systém s použitím UML jazyka. Na základě teoretické a praktické bude formulován závěr diplomové práce.

Doporučený rozsah práce

60 – 80 stran

Klíčová slova

Softwarové inženýrství, UML, informační systém, Wireframe, Analýza požadavků, objektový model, stavový model a model interakcí

Doporučené zdroje informací

BRUCKNER Tomáš. Tvorba informačních systémů, [2012]. ISBN 978-80-247-4153-6.
BUCHALCEVOVÁ Alena, STANOVSKÁ Iva. Příklady modelů analýzy a návrhu aplikace v UML, [2013]. ISBN 978-80-245-1922-7.
VRANA Ivan. Projektování informačních systémů s UML, [2008]. ISBN 978-80-213-1817-5.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Mgr. Vladimír Očenášek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 30. 03. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci " Analýza a návrh informačního systému v UML pro společnost GHC Genetics" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2023 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu diplomové práce Ing. Mgr. Vladimíru Očenáškovi, Ph.D., za cenné rady a čas, který mi věnoval v průběhu psaní této práce.

Analýza a návrh informačního systému v UML pro společnost GHC Genetics

Abstrakt

Tato diplomová práce se zaměřuje na analýzu a návrh informačního systému v jazyce UML pro společnost, která poskytuje genetické testovaní, covid testovaní a očkovaní. Cílem je vytvořit efektivní systém, který usnadni proces poskytování služeb a zároveň sníží administrativní náročnost.

Teoretická část práce se věnuje studiu odborných informačních zdrojů, komponent informačního systému, analýze uživatelských požadavků a metodik tvorby informačních systémů, včetně popisu standardů jazyka UML.

Praktická část práce obsahuje popis navrženého systému a analýzu uživatelských požadavků. Následuje návrh systému, který zahrnuje model tříd, stavový model a model interakcí. V závěru práce jsou navrženy wireframe a diagram nasazení pro vizualizaci systému.

Klíčová slova: softwarové inženýrství, UML, informační systém, návrh informačního systému, modelování informačního systému, wireframe, analýza požadavků, objektový model, stavový model a model interakcí

Analysis and design of information system in UML for company GHC Genetics

Abstract

This thesis focuses on the analysis and design of an information system in the UML language for a company that provides genetic testing, COVID testing, and vaccination. The goal is to create an efficient system that facilitates the process of service provision while reducing administrative complexity.

The theoretical part of the thesis focuses on studying professional information sources, information system components, user requirements analysis, and methodologies for creating information systems, including a description of UML language standards.

The practical part of the thesis includes a description of the proposed system and an analysis of user requirements. This is followed by a system design that includes a class model, state model, and interaction model. In conclusion, wireframes and a deployment diagram are proposed for system visualization.

Keywords: software engineering, UML, information system, information system design, information system modelling, wireframe, requirements analysis, object model, state model and interaction model

Obsah

1 Úvod

Informační systémy a informační a komunikační technologie (IS/ICT) se stále více dotýkají oblastí lidské činnosti, které se velmi rychle a dynamicky mění. Jsou neoddělitelnou součástí současného světa. A ve firmách, které provozuje podnikatelskou činnost, zejména nezbytné zavedení informačního systému pro zlepšení interakce mezi zákazníkem a společností. Taková společnost dosahuje mnohem efektivněji stanovených cílů, snižuje náklady, čas na zpracování dat a stejně zvyšuje produktivitu práce

Tato diplomová práce se zabývá analýzou a návrhem informačního systému, který je určený pro společnost „GHC Genetics“. V práci jsou použity i profesní zkušenosti autora, které získal během působení na administrativní pozici.

Hlavním cílem je minimalizovat administrativní zátěž a efektivně podporovat firemní procesy, zejména komunikaci s klienty, což je klíčovým faktorem. Systém pomůže zkrátit dobu práce s administrativou, objednáváním zboží a komunikací s laboratoří.

Hlavním úkolem teto práce je vytvořit pracovní informační systém pomocí grafického modelovacího jazyka UML (Unified Modeling Language), který pomůže rozdělit zátěž mezi všechny zaměstnance a konkretizovat definovaný cíl firmy, stejně snížit náklady.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem diplomové práce je analyzovat a navrhnout funkční model informačního systému pomocí UML. Tento model umožňuje efektivně spravovat a řídit chod firmy, pomoci sběru, uchování, přenosu a zpracování dat.
Dílčí cíle diplomové práce jsou:

- provést rešerši v oblasti návrhu informačních systémů
- sběr a analýza požadavků
- navrhnout objektový model, stavový model a model interakcí

2.2 Metodika

Metodika diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Pomoci získaných poznatků bude vytvořen informační systém s použitím UML jazyka. Na základě teoretické a praktické bude formulován závěr diplomové práce.

3 Teoretická východiska

3.1 Informační systém

Pojem informační systém (zkratka IS) je velmi podobný pojmu byznys systém. Komponenty IS se obvykle shodují s komponentami byznys systému, často je zde však důležitější informace o komponentě (o člověku, strojí, materiálu atd.). Informační systém a byznys systém se tedy mohou shodovat svými komponentami, liší se ale svým účelem. Účelem IS je zajištění správných informace na správném miste ve správný čas. Místem, kam mají byt informace dodaný, jsou obvykle lide, kteří jsou součásti byznys systému (uživatelé IS), a kritériem správnosti je vhodnost podpory byznys systému v plněni jeho účelu (v případě podniku obvykle zájmena v dosahovaní cílů). (Bruckner, 2012)

Pro plnění účelů informačních systému jsou důležité informační a komunikační technologii (ICT). Proto často používáme pro informační systém podporovaný informačními a komunikačními technologiemi zkratku IS/ICT. Informační a komunikační technologie (ICT) jsou hardwarové a softwarové prostředky pro sběr, přenos, ukládaní, zpracovaní a distribuce informaci a pro vzájemnou komunikaci lidi a technologických komponent IS (Voříšek, 2008)

Informační systém zahrnuje jak automatizované, tak neautomatizované činnosti. Automatizované činnosti podporuje software, tedy programové vybavení. V anglicky psané odborné literatuře je pojem software (či zkratka SW) používán často a přenáší se i do české odborné literatury. V kontextu tvorby softwaru se používá také termín programový systém. Programový systém je softwarový produkt, který je tvořen množinou programových jednotek (modulů, objektů, komponent, služeb) a jejich vzájemných vazeb (Buchalcevová, 2005). Pojmem aplikační software (zkrácené aplikace) rozumíme takový software, který je určen k užití přímo uživatelem. V oblasti podnikových informačních systémů je tedy aplikační software takový software, který používají uživatelé informačního systému při řešení svých informačních potřeb v byznysu.

Tvorba informačního systému obvykle zahrnuje tvorbu aplikačního softwaru nebo alespoň jeho parametrizaci a nasazení. To je důležitá, ale pouze dílčí problematika tvorby informačních systémů. Neméně důležitou částí je zajištění, aby software byl vhodně použitelný v byznysu. Proto se na tvorbu programových systémů díváme nutně v kontextu byznysu.

3.1.1 Klasifikace IS dle úrovní řízení

Operativní úroveň (TPS)

Pro potřeby vrcholového řízení na strategické řídicí úrovni. Zde potřebujeme spíše informace, které charakterizují celkové fungování podniku, jako podklad pro strategické řízení.
Data, se kterými pracuje systém typu EIS, jsou většinou pořizována v systémech TPS a MIS. Data pro EIS se ovšem vyznačují vysokou agregací a jsou strukturovaná. Oproti TPS a MIS, které většinou pracují s okamžitým stavem, pracuje EIS s daty v širším časovém horizontu. Pro EIS je typické použití prostředků, které označujeme pojmem „Business Intelligence“ (BI).Jedná se vytváření centrálních datových skladů (které slučují data z různých zdrojů a systémů) nebo analytické nástroje pro analýzu vzájemných závislostí, jako je OLAP či Data Mining (dolování dat).EIS často vystupují v roli prezentační vrstvy pro prostředky BI.
Typické funkce systémů EIS jsou:

- plánování v dlouhodobém horizontu
- ekonomická analýza celkového hospodaření firmy
- hodnocení podnikatelských záměrů
- příprava inovačních akcí
- formulace strategických projektů metodami projektového řízení
- podpora specifikace marketingové strategie firmy
- manažerské výkaznictví
- rozbor situace na trhu apod

(Ing. Roman Danel, Ph.D., Informační systémy [přednáška]. Ostrava: VŠB-TUO)

Taktická úroveň (MIS)

Systémy typu MIS se zabývají řízením podniku na taktické úrovni řízení.
Do této oblasti spadají ekonomická, organizační a obchodní hlediska a oblast kontroly.
Základní oblasti MIS systémů:

- Obchodně logistické procesy
- Finančně účetní procesy
- Průřezové aplikace celopodnikového charakteru (správa, legislativa, řízení lidských zdrojů, marketing, jakost...)

Charakteristika činností prováděných v systémech MIS:

- Evidence procesů
- Zpracování ekonomických analýz
- Převažují evidenční a analytické práce

Systémy TPS a MIS vypovídají o aktuálním stavu podnikových procesů.
(Ing. Roman Danel, Ph.D., Informační systémy [přednáška]. Praha: VŠB-TUO)

Strategická úroveň (EIS)

Cílem informačního systému typu TPS (Transaction Processing System) je podpora hlavních činností podniku na operativní úrovni (provozní úroveň řízení, sledování transakcí –tj. jednotlivých výrobních operací).Je to blok aplikací zaměřený na hlavní činnosti podniku (Core Business). Je nejspecifičtější podle zaměření podniku a jeho konkrétní řešení nejvíce závisí na konkrétní činnosti podniku.

TPS systémy se tedy liší podle odpovědí na následující otázky:

- Jaký je charakter podniku? (výrobní, obchodní, dopravní...?)
- Je-li výrobní, Jaký je charakter výroby? (kusový, kontinuální...?)

U výrobních podniků jsou TPS založeny na tzv. CIM concepcion (Computer Integrated Manufacturing). Principem CIM je integrace výrobních procesů:

- Výrobkových
- Zakázkových

(Ing. Roman Danel, Ph.D., Informační systémy [přednáška]. Praha: VŠB-TUO)

3.2 Podnikové informační systémy

Automatizace podnikání je proces implementace a provozu souboru pracovních prostředků (programů a zařízení), které zajišťují minimalizaci rutiny, optimalizaci pracovních a výrobních zdrojů s cílem zvýšit produktivitu a efektivitu všech obchodních procesů.

Podnikový informační systém, někdy též EIS z anglického Enterprise Information System, podporuje a zefektivňuje funkci či workflow podnikových procesů a je nedílnou součástí projektového managementu. Podnikové informační systémy v sobě schraňují veškerá podniková data, delegují je mezi další informační systémy a usnadňují správu veškerých informací, které podniky a organizace potřebují ke svému fungování.
(Rascasone.com, 2021)

EIS je tedy speciální typ informačního systému (IS), který je prioritně určen pro větší podniky skládající se z množství oddělení nebo pracovišť. Právě díky tomuto IS spolu jednotlivá oddělení mohou efektivně spolupracovat a vždy využívat všechna potřebná data nebo informace.
(Rascasone.com, 2021)

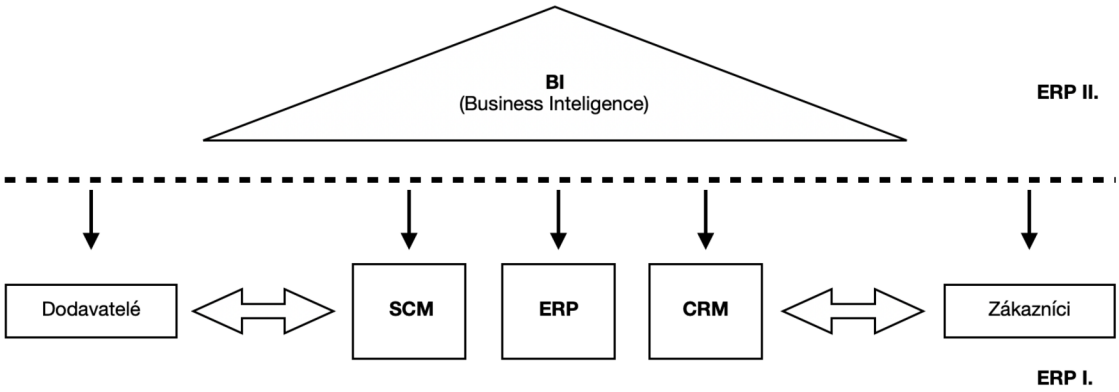
S růstem firmy se tak podnikový informační systém stává její stále potřebnější a nedílnou součástí. Současně nebrání dalšímu rozvoji ku příkladu v rámci expanze na mezinárodní trh. EIS vám zkrátka umožní zpracovat mnohem větší množství dat než klasické tabulky nebo tužka a papír.
(Rascasone.com, 2021)

S tím se pojí zejména vyšší kvalita služeb a přehled nad všemi sklady, zaměstnanci i účetnictvím. Ke každému tomuto sektoru můžete využívat speciální informační systém (skladový, docházkový nebo fakturační) a ten propojit s podnikovým informačním systémem. Díky této integraci budete mít všechna data vždy pohromadě.
(Rascasone.com, 2021)

3.2.1 členění podnikových informačních systém

Informační systém podniku se obecně člení do dvou větví, na ekonomickou a ERP. Na ty navazují i další informační systémy – každý zaměřený na nějakou specifickou oblast. Patří sem zmíněné docházkové a fakturační systémy, ale i HRM nebo CRM a mnoho, mnoho dalších.
(Rascasone.com, 2021)

ERP systémy přitom utváří jakýsi základ podnikové infrastruktury. V mnoha případech přitom bývají pokládány přímo za podnikový informační systém jako takový. ERP jsou ovšem již zaměřeny přímo na plánování lidských zdrojů a organizaci lidské práce.
(Rascasone.com, 2021)



Obrázek 1: Základní schémata podnikového software (Zdroj: Vlastní zpracování)

ERP (Enterprise Resource Planning)

ERP je plánování podnikových zdrojů. Ale jaký smysl má tento pojem? V nejobecnější podobě ERP lze definovat jako souhrn všech základních obchodních procesů potřebných pro řízení společnosti: finance, řízení lidských zdrojů, výroba, dodavatelský řetězec, služby, nákupy a další. Na nejzákladnější úrovni ERP pomáhá efektivně řídit všechny tyto procesy v integrovaném systému. Často se nazývá systém vedení záznamů v podniku.

CRM (Customer Relationship Management)

Systém řízení vztahů se zákazníky – aplikační software určený k automatizaci spolupráce se zákazníkem s cílem zvýšit úroveň prodeje, zefektivnit marketing a zlepšit služby zákazníkům uložením informací o zákaznících a historii vztahů s nimi, založením a zlepšením podnikových procesů a následnou analýzou výsledků.

SCM (Supply Chain Management)

Systém řízení dodavatelského řetězce – software určený k automatizaci a řízení všech fází zásobování podniku a ke kontrole veškerého zásobování podniku. SCM systémy pokrývá celý komoditní cyklus: nákup surovin, výroba, distribuce zboží.

BI (Business intelligence)

Business Intelligence je software, který pomáhá manažerovi analyzovat informace o své společnosti a jejím okolí. Technologie BI umožňují analyzovat velké množství informací, zaostřit pozornost uživatelů pouze na klíčových faktorech účinnosti, simulovat výsledek různých možností akcí, sledovat výsledky rozhodování.

3.3 Životný cyklus informačního systému

Analýza (system analysis)

Modelování budoucího systému na konceptuální úrovni

Návrh (system design)

Modelování budoucího systému na technologické úrovni; převedení specifikací na hierarchii stále detailnějších schémat, která definují požadovaná data a rozkládají procesy, jež se mají s daty provádět, až na úroveň, kdy mohou být vyjádřeny v podobě instrukcí pro počítačový program

Vývoj systému (system development)

Psaní a testování počítačového softwaru (programování), vývoj vstupních a výstupních formulářů.
(Ing. Petra Pavlíčková, Ph.D., Řízení IT projektů [přednáška]. Praha: ČZU, 2018)

Implementace systému

Uvedení systému (hardwaru i softwaru) do provozu – instalace fyzického systému a aktivity s tím související, jako je školení operátorů a uživatelů.
(Ing. Petra Pavlíčková, Ph.D., Řízení IT projektů [přednáška]. Praha: ČZU, 2018)

Správa systému

Další vývoj funkcí a struktury systému, jež vyplývají ze změněných požadavků a technologií, zkušeností s užíváním systému a dolaďování jeho výkonu.
(Ing. Petra Pavlíčková, Ph.D., Řízení IT projektů [přednáška]. Praha: ČZU, 2018)

Údržba systému

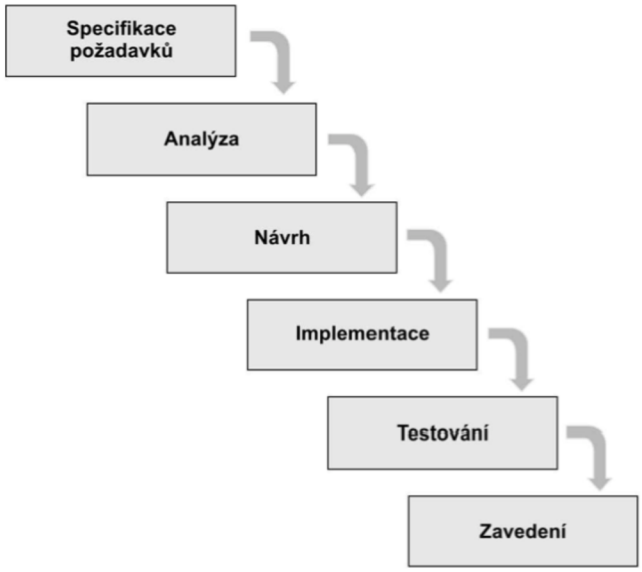
Úprava systému při jeho provozování podle nově vzniklých požadavků uživatele, resp. pro odstranění závad.
(Ing. Petra Pavlíčková, Ph.D., Řízení IT projektů [přednáška]. Praha: ČZU, 2018)

3.3.1 Modely životního cyklu

Vodopádový model

V době svého vzniku byl Vodopádový model velkým průlomem, protože umožnil systematický postup vývoje tím, že rozdělil celý proces do jednotlivých fází. Ačkoli je dnes často kritizován, ale stále se v praxi ještě hodně používá. Je úspěšný v situacích, kde jsou požadavky již přesně definovány a během vývoje se příliš nemění. Poskytuje přehled o

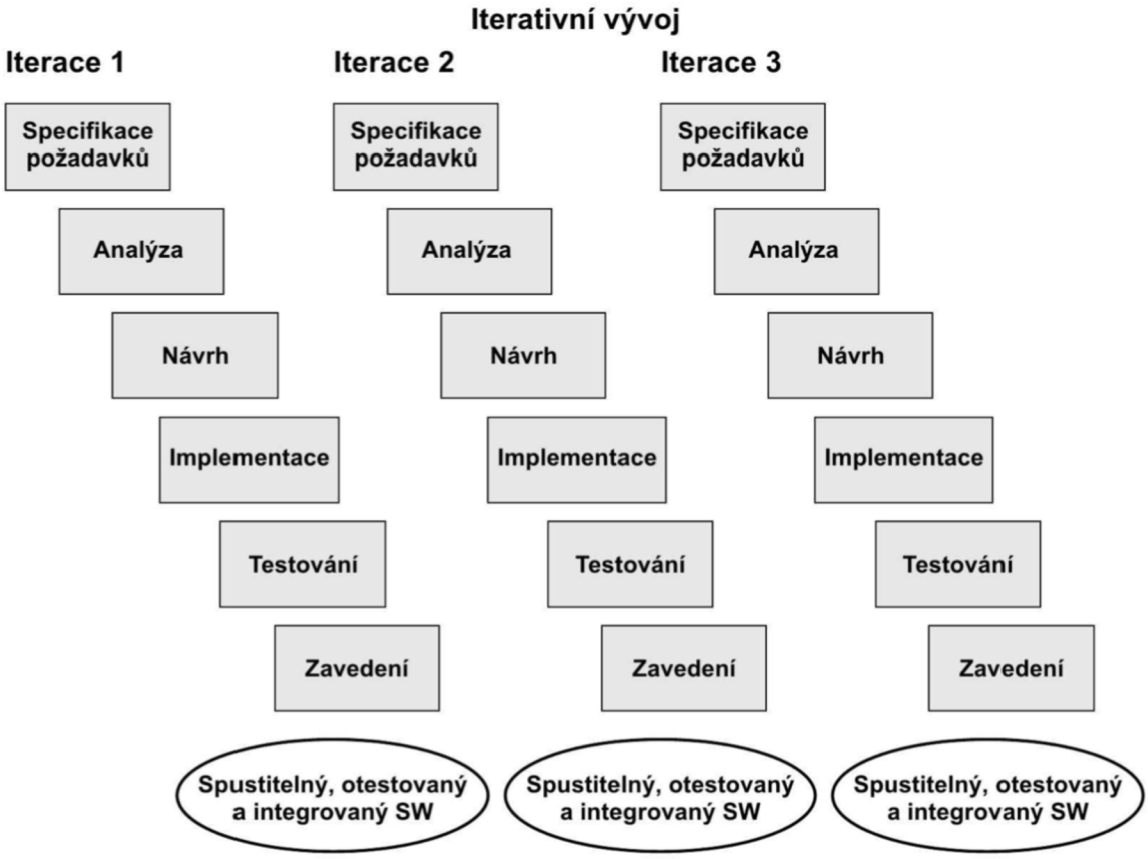
celkovém rozsahu řešení. Nicméně, pokud není možné všechny požadavky na projekt specifikovat na začátku, nebo jsou nutné časté změny, model přináší problémy. Další nevýhodou je, že zákazník má v průběhu projektu omezenou kontrolu, jelikož je do vývoje zapojen především na začátku a konci procesu. Hlavním problémem je také pozdní integrace programového systému, která se provádí až po naprogramování všech modulů. Během integrace se často objevují problémy, což vede k přeprogramování a celkovému zpoždění projektu. (Bruckner, 2012)



Obrázek 2: Vodopádový model (Zdroj: Bruckner, 2012)

Model pro iterativní vývoj

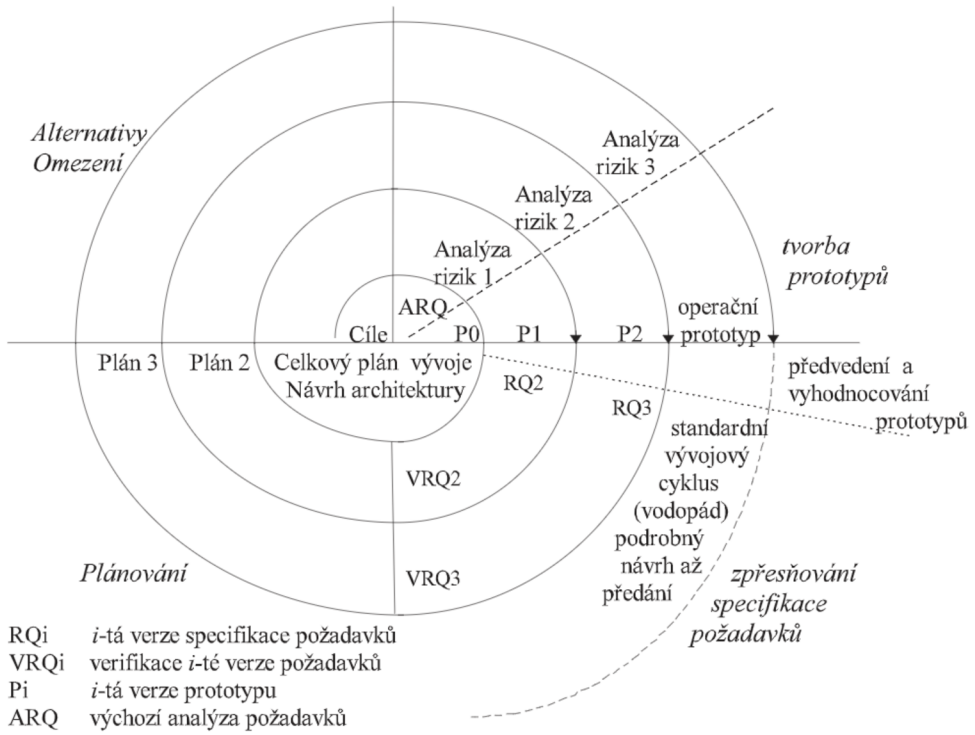
Nevýhody vodopádového vývoje odstraňuje iterativní vývoj, který je dnes moderním způsobem vývoje. Pojem ite-race poprvé v souvislosti s modelem vývoje použil Barry Boehm, když definoval spirálový model (Boehm, 1988). Iterativní vývoj staví na skutečnosti, že člověk řeší lépe menší problémy, a proto rozkládá celý projekt na řadu dílčích projektů – iterací. Každá iterace přitom obsahuje všechny fáze vývoje – plánování, analýzu, návrh, integraci, testování a zavedení, tedy takový malý vodopád, jak ukazuje obrázek dole. Výsledkem každé iterace musí být fungující část systému, která je otestována a integrována s výsledky předchozích iterací. Iterace je tedy kompletní cyklus vývoje, který končí dodávkou (interní či externí) spustitelného produktu (Larman, 2004).



Obrázek 3: Iterativní vývoj (Zdroj: Bruckner, 2012)

Spirálový model

Spirálový model (B.W.Boehmem, 1986-88) zahrnuje vývoj ve formě posloupnosti planů, ale na začátku projektu nejsou definovány všechny požadavky. Požadavky jsou upřesněny v důsledku vývoje planu. Ten model je kombinací analýzy rizik a prototypového přístupu



Obrázek 4: Spirálový model vývoje softwaru (Zdroj: Král, 1998)

1. Výhody

- umožňuje rychlejší zobrazení uživatelům systému funkční produkt, čímž se aktivuje proces upřesnění a doplnění požadavků
- umožňuje změnu požadavků při vývoji informačního systému, což je typické pro většinu vývoje
- poskytuje větší flexibilitu při řízení projektu
- umožňuje získat spolehlivější a udržitelnější systém
- umožňuje zlepšit proces vývoje-analýza prováděná v každé iteraci umožňuje posoudit, co by mělo být změněno ve vývoje, a zlepšit ji v další iteraci;
- snižuje riziko zákazníka. Zákazník může s minimálními finančními ztrátami dokončit vývoj neperspektivního projektu.

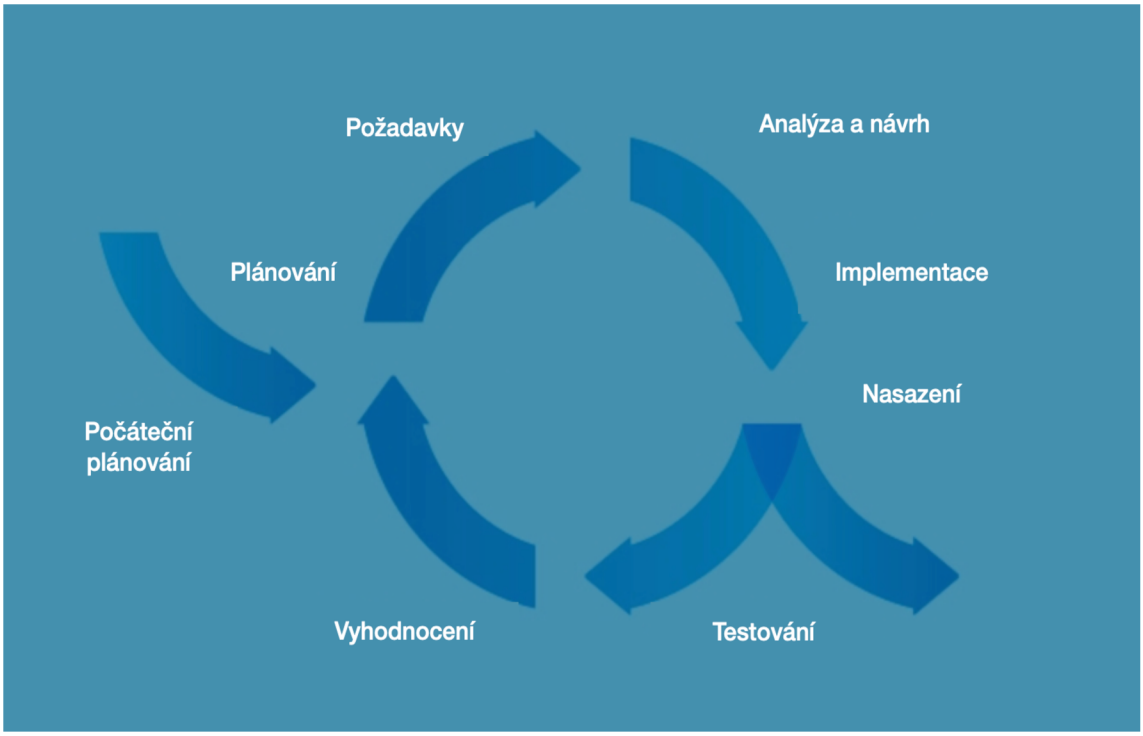
2. Nevýhody

- zvyšuje nejistotu developera v perspektivách rozvoje projektu.

- operace dočasného a zdrojového plánování celého projektu jsou obtížné. K vyřešení tohoto problému je nutné zavést dočasná omezení pro každou fázi životního cyklu. Přechod se provádí podle plánu, i když ne všechny plánované práce jsou hotové.

Inkrementální model

Je příklad iterativního přístupu k vývoji softwaru IS, který zahrnuje rozdělení životního cyklu projektu na sekvenci iterací, z nichž každá představuje "mini-projekt", který zahrnuje všechny fáze životního cyklu.
V tomto případě se na každé iteraci získává funkční verze softwarového systému, který má funkčnost všech předchozích. Výsledkem iterace je konečný produkt, který zajišťuje realizaci všech požadavků.
Z hlediska struktury životního cyklu se model nazývá iterativní (mluvíme o procesu). Z hlediska vývoje produktu – inkrementální (to znamená zvýšení funkčnosti produktu). Inkrement – přírůstek, zvýšení (např. v jazyce programování – zvýšení hodnoty o 1)



Obrázek: 5 Inkrementální model (Zdroj: Vlastní zpracování)

1. **Výhody**

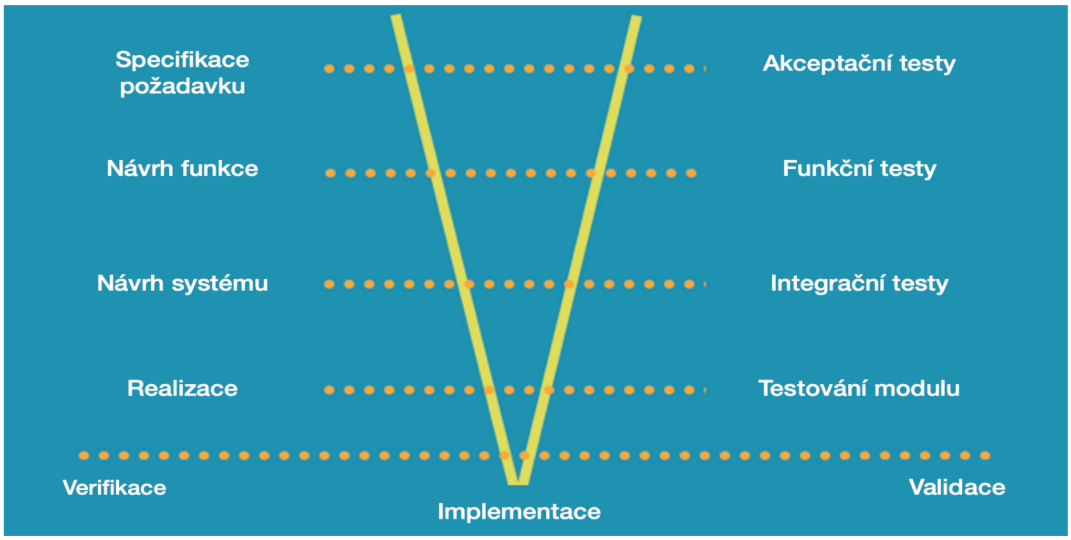
- pracovní aplikace se projevuje v rané fázi životního cyklu produktu
- pružnost
- pomocí iterace zjednoduší testování a úpravy
- jednodušší řízení rizik
- každá iterace je ovladatelný kontrolní bod projektu

2. **Nevýhody**

- Každá fáze iterace je nepohyblivá
- Problémy týkající se architektury systému

V-Model

Podobně jako u vodopádového modelu, i V-Model používá krokovou strukturu. Tento model se obzvláště hodí pro systémy, u kterých je důležitá jejich neustálá funkčnost. Vyniká především svým zaměřením na pečlivou kontrolu a testování produktu již v raných fázích návrhu. Testování se provádí současně s odpovídající fází vývoje, například jednotkové testy se píší během kódování. (Iquest.cz, 2017)



Obrázek 6: V-Model (Zdroj: Vlastní zpracování)

Kdy použít V-Model?

- Potřeba důkladného testování výrobku. Pomocí V-Modelu probíhá Validace a Verifikace

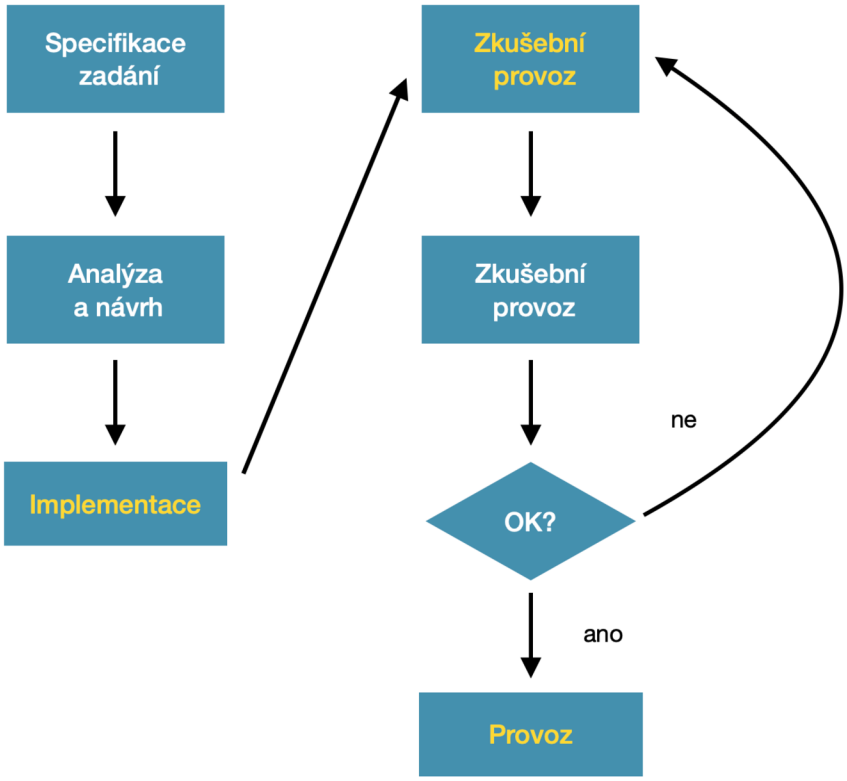
- Pokud ve vašem projektu máte stanovené a pevné požadavky

- Nejsou žádné problémy s dostupností inženýrů a testerů s potřebnou kvalifikací.

(Iquest.cz, 2017)

Prototypový model

Tento přístup k vývoji IS je založen na plánování a řízeném inkrementálním přístupu. Prototyp se skládá z částečné implementace produktu, což může být logická nebo fyzická forma, která zahrnuje všechna vnější rozhraní systému. Prototyp je prezentován uživatelům, kteří ho testují a poskytují zpětnou vazbu. Na základě této zpětné vazby se specifikace požadavků na systém upřesňuje a prototyp je dále upravován a doplňován, až dosáhne podoby výsledného systému. Na rozdíl od přístupu "výzkumník", tento přístup vychází z pečlivé analýzy a návrhu systému, který obsahuje plánované stupně vývoje prototypu a pravidla pro jeho řízení. (Mgr. Martinů Jiří, doc. Ing. Čermák Petr, Ph.D., Metodiky vývoje software [přednáška]. Olomouc: MVŠO, 2018.)



Obrázek 7: Prototypový model (Zdroj: Vlastní zpracování)

3.4 Metodiky vývoje informačních systémů

Proces vývoje softwaru se skládá z podobných fází, ale způsob, jakým jsou tyto fáze propojeny a jak fungují, je určen metodikou vývoje informačního systému (IS). Metodika má následující úkoly:

- pomáhá realizovat a vytvořit užitečný produkt
- poskytuje jasnou představu o krocích práce a snížit pravděpodobnost chyb během vývoje díky rozdělení úkolů.

V současnosti můžeme sledovat dva hlavních metodických přístupů – rigorózní metodiky a agilní metodiky. (Buchalcevová, 2005)

3.4.1 Rigorózní metodiky

Rigorózní metodiky jsou přesvědčeny, že procesy při vývoji IS lze popsat, plánovat, řídit a měřit. Cílem těchto metodik je podrobně a přesně definovat procesy, činnosti a produkty, což způsobuje, že bývají velmi objemné. Tyto metodiky jsou obvykle založeny na sériovém vývoji, jako je například vodopádový model, ale existují i rigorózní metodiky založené na iterativním a inkrementálním vývoji.
Příkladem rigorózních metodik jsou OPEN a Rational Unified Process (RUP). (Buchalcevová, 2005)

OPEN (Object-oriented Process, Environment and Notation)

Metodika OPEN je veřejně přístupná metodika, která podporuje celý životní cyklus vývoje a zaměřuje se zejména na vývoj objektově orientovaných a komponentových aplikací. Vznikla jako výsledek projektu COMMA (Common Object Methodology Metamodel Architecture), jehož cílem bylo vytvořit metamodely různých metodik a metod objektově orientované analýzy a návrhu. (Buchalcevová, 2005)

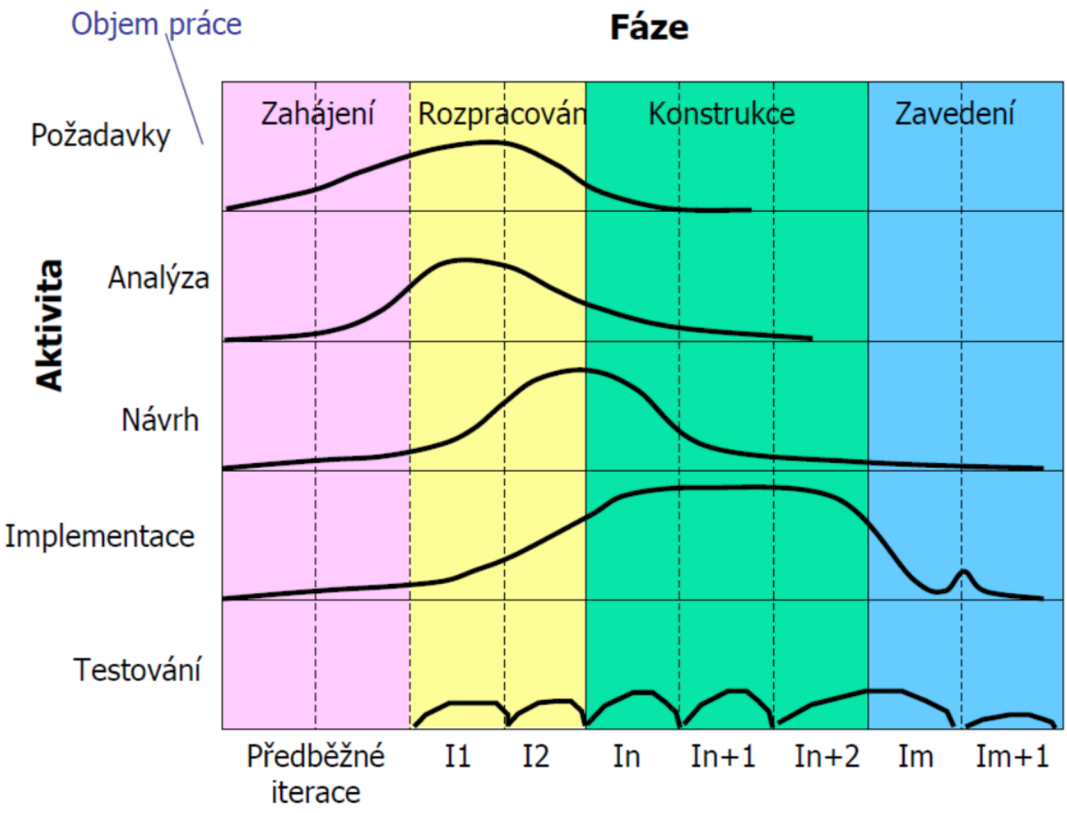
UP (Unified Process)

Metodika UP (Unified Process) je založena na průmyslovém standardu SEP (Software Engineering Process) a USDP (Unified Software Development Process), což je unifikovaný proces vývoje softwaru. UP je zkrácené označení a jde o generický proces, který je založen na jazyku UML (Unified Modeling Language). UP je otevřenou metodikou, kterou je možné přizpůsobit cílům a podmínkám daného vývoje, včetně používaných norm, šablon a nástrojů. Unifikovaný proces (UP) znamená:

- řízení požadavky a případy užití
- řízení rizikem
- základ na architektuře
- iterativní a přírůstkový proces vývoje SW produktu

UP je rozdělen do jednotlivých iterací, z nichž každá prochází pěti základními pracovními procesy:

- stanovení požadavků
- analýza
- návrh
- implementace
- testování
- iterace v UP



Obrázek 8: Etapy (fáze) metodiky UP (Zdroj: Martinů, Čermák Petr, [přednáška]. Olomouc: 2018.)

RUP (Rational Unified Process)

Metodika RUP (Rational Unified Process) je variantou metodiky UP (Unified Process), přičemž obsahuje všechny normy, které jsou obsaženy v UP.
RUP disponuje bohatým uživatelským rozhraním a rozšiřuje UP v mnoha důležitých aspektech. I když obě metodiky mají mnoho společného, významné rozdíly lze najít spíše v konkrétních detailech a úplnosti metodik než v samotném významu. (Mgr. Martinů Jiří, doc. Ing. Čermák Petr, Ph.D., Metodiky vývoje software [přednáška]. Olomouc: MVŠO, 2018.)

3.4.2 Agilní metodiky

Agilní přístupy k vývoji softwaru byly vytvořeny jako reakce na složitost a náročnost klasických metodik vývoje, které se někdy ukázaly jako neúměrné potřebám určitých typů informačních systémů. Tyto přístupy jsou nezbytné především pro projekty s nejasným nebo se často měnícím zadáním. Rigorózní metody mají své nedostatky, jako je vodopádový model, přílišný dohled vedoucích projektu a těžkopádná reakce na změny. Agilní přístupy se snaží zjednodušit a urychlit proces vývoje, vracejí se ke starším praktikám a usilují o větší pružnost a flexibilitu. V této souvislosti je RUP považován za běžnou verzi UP s rozšířením v mnoha detailech a úplnosti metodiky, včetně bohatého uživatelského prostředí. (Mgr. Martinů Jiří, doc. Ing. Čermák Petr, Ph.D., Metodiky vývoje software [přednáška]. Olomouc: MVŠO, 2018.)

Agilní metodiky se řídí třemi hlavními principy:

- **přírůstkový (iterativní) vývoj s velmi krátkými iteracemi** – tento přístup spočívá v tom, že nejdůležitější funkce SW jsou nejprve vytvořeny a poté předány zákazníkovi k ověření. Na základě zpětné vazby od zákazníka jsou další funkce postupně přidávány.
- **důraz na komunikaci mezi zákazníkem a vývojářem** – zástupci zákazníka jsou členy vývojového týmu a úzce spolupracují s vývojáři na návrhu systému. Krátké iterace umožňují zákazníkům sdělit své zkušenosti a zpětnou vazbu.
- **přísné automatizované testování** – pro každý software je vytvořena komplexní sada testů, které jsou provedeny před každou změnou SW a poté opět při testování

změněného SW. (Mgr. Martinů Jiří, doc. Ing. Čermák Petr, Ph.D., Metodiky vývoje software [přednáška]. Olomouc: MVŠO, 2018.)

Scrum

Scrum je metodika, která se skládá z principů, jež umožňují postup vývoje informačního systému. Podstatou této metodiky je vývojový cyklus nazývaný Sprint, který představuje iterativní proces. V průběhu Sprintu týmy sledují pokrok vývoje a upravují své plány na denních 15minutových schůzkách, které se konají ve stoje, takzvaných stand-up meetingech. V každém Sprintu jsou výsledky funkčních částí softwaru prezentovány na konci iterace. Sprinty jsou pevně stanoveny v čase a mají trvání od 2 do 4 týdnů. Tyto iterace jsou zodpovědné za vytváření funkčního rozšíření softwaru.
(Mgr. Martinů Jiří, doc. Ing. Čermák Petr, Ph.D., Metodiky vývoje software [přednáška]. Olomouc: MVŠO, 2018.)

Metodika Scrum se zakládá na důležitém poznání dvou faktorů:

- pohledy a požadavky zákazníků se mohou neustále měnit
- objevují se neočekávané situace, na které nelze předem plně připravit.

(Mgr. Martinů Jiří, doc. Ing. Čermák Petr, Ph.D., Metodiky vývoje software [přednáška]. Olomouc: MVŠO, 2018.)

XP (Extreme Programming)

Jedna z agilních metodik vývoje IS. Název metodiky pochází z myšlenky aplikovat užitečné tradiční metody a postupy vývoje softwaru a zvýšit jejich na nový "extrémní" úroveň. Takže například provádění *code review*, která spočívá v ověření kódu jednoho programátora jiným. Hlavní výhodou této metody je párové programování, kde kód, určený do produkčního prostředí, vždy společně vytváří dva programátoři. Jeden z nich je tzv. řidič a druhý navigátor. První z nich píše kód, druhý neustále vše sleduje, přidává připomínky, kontroluje program a přemýšlí o možných souvislostech. Oba spolu neustále komunikují, aby byla okamžitě zapojena zpětná vazba.
(Bonsai-development.cz, 2018)

3.5 Analýza požadavků na informační systém

Analýza požadavků je podmínkou úspěšného dokončení projektu vývoje. Požadavek musí být měřitelný, testovatelný, proveditelný, měřitelný, musí se vztahovat ke konkrétnímu byznys požadavku a příležitosti, stejně musí být dostatečné detailně definovaný pro účely návrhu systému.

Analýza požadavků zahrnuje tři typy činností:

- Sběr požadavků – komunikace se zákazníky a uživateli pro určení požadavků
- Analýza požadavků – identifikace vztahu požadavků
- Zaznamenání požadavků – specifikace procesů a dokumentování požadavků v různých formách

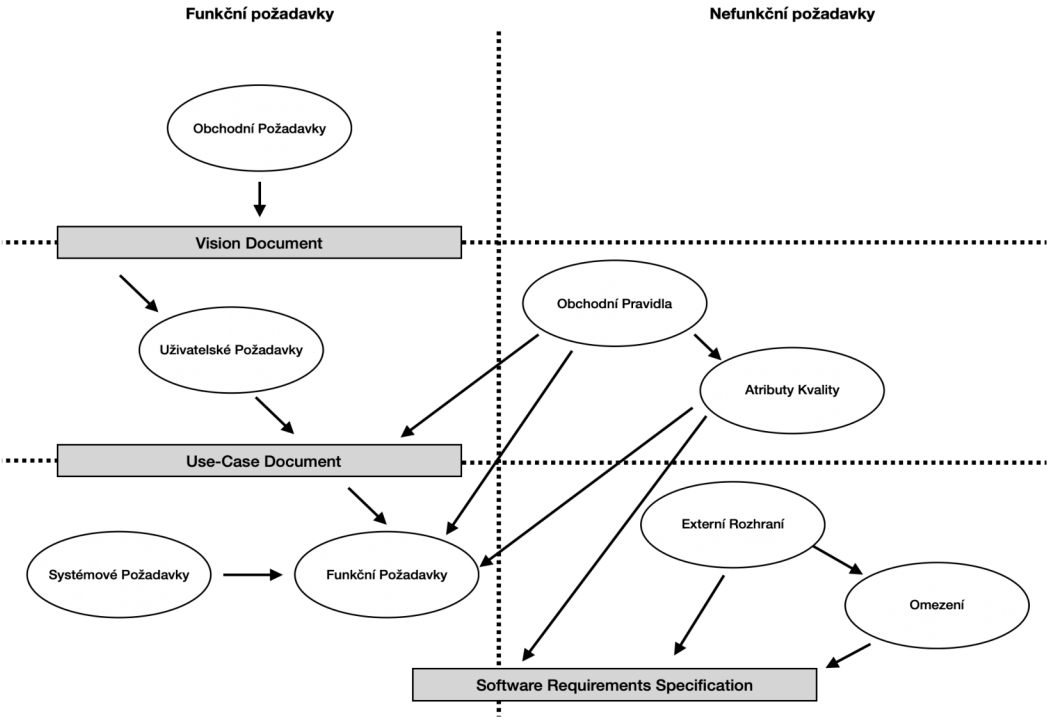
Cíle – jdeme po jednotlivých požadavků a chceme zkoumat detailně, snažíme zachytat omezeni:

- Vymezit hranice systému
- Umožnit přesnější odhad pracnosti
- Vyjasnit si zadaní se zákazníkem
- Zachycení omezení

(Ing. Petra Pavlíčková, Ph.D., Řízení IT projektů [přednáška]. Praha: ČZU, 2018)

3.5.1 Typy požadavků

Základní kategorizace požadavků je funkční a nefunkční (obecné) požadavky



Obrázek 9: Druhy požadavků (Zdroj: Vlastní zpracování)

Funkční požadavky – co by měl systém dělat.

K funkčním požadavkům patří:

- **Obchodní požadavky** – co by měl systém dělat z hlediska podnikání
- **Uživatelské požadavky** – popisují cíle / úkoly uživatelů systému, které musí uživatelé dosáhnout/splnit pomocí vytvořeného softwarového systému. Tyto požadavky jsou často prezentovány pomocí použití (Use Case)
- **Funkční požadavky** – definují funkčnost (chování) softwarového systému

Do skupiny funkčních požadavků patří i **systémové požadavky**. Tyto vlastnosti mohou popisovat požadavky jak na hardware (typ a frekvence procesoru, množství paměti RAM, objem pevného disku), tak na softwarové prostředí (operační systém, přítomnost nainstalovaných systémových komponent a služeb atd.). Obvykle jsou tyto požadavky vypracovány výrobcem nebo autorem softwaru.

Druhou skupinou požadavků jsou **nefunkční požadavky**. Jinými slovy, jak bude systém fungovat

- **Obchodní pravidla** – určují, proč by měl systém fungovat přesně tak, jak je požadováno . Mohou to být odkazy na právní předpisy atd.

- **Externí rozhraní** – nejsou to jen uživatelské rozhraní, ale stejně protokoly o interakci s jinými systémy. Často jsou například webové stránky propojeny s CRM systémy.

- **Atributy kvality** – se týkají otázek průhlednosti, integrity, udržitelnosti systému atd.

- usability (použitelnost)
- performance (výkon)
- maintainability (údržba)
- reliability (spolehlivost)
- interfaces (rozhraní)
- scalability (rozšiřitelnost)
- user and software interface (uživatelské a softwarové rozhraní)

- **Omezení** – soubor požadavků, které zužují výběr možných řešení pro jejich realizaci.

3.5.2 Metody analýzy požadavků

Interview s uživatelem

Nutná příprava otázek předem a promyšlen konkrétní scénář. A hlavní výhoda je v tom, že pomoci neverbální informace lze posoudit, co pro uživatele je nejvíc důležité. (Ing. Roman Danel, Ph.D., Analýza a projektování systémů [přednáška]. Ostrava: VŠB-TUO)

Dotazník (Questionnaire)

Dotazník je jedním z nejlepších řešení pro shromažďování požadovaných dat a názorů uživatelů. Výhodou je možnost oslovit velké množství respondentů a odpovědi statisticky vyhodnotit.

Druhy otázek:

- OPEN-END – respondent odpovídá svými slovy
- CLOSE-END – odpověď je omezena (multiple choice, hodnocení 1 až 5 atd.)

Close-end otázky obvykle převažují, protože se lepe statistiky vyhodnocují
(Ing. Roman Danel, Ph.D., Analýza a projektování systémů [přednáška]. Ostrava: VŠB-TUO)

Pozorování (přítomnost na pracoviště)

Hlavní výhodou je v přítomnosti analytika na pracovišti, analytik sleduje a vyhodnocuje činnost uživatelů při práci. (Ing. Roman Danel, Ph.D., Analýza a projektování systémů [přednáška]. Ostrava: VŠB-TUO)

Studium dokumentace a písemných zdrojů

Studium psané dokumentace je docela užitečný zdroj informací, ale hlavní problém je v tom, že dokumentace není aktualizována a udržována, z tohoto důvodu mohou být získané informace nesprávné. (Ing. Roman Danel, Ph.D., Analýza a projektování systémů [přednáška]. Ostrava: VŠB-TUO)

JAD (Joint Application Design)

Je proces, který se používá k navrhování a vývoji počítačového systému / řešení. Sbírá požadavky vedle sebe podle obchodních potřeb při vývoji nových informačních systémů pro společnost, což znamená, že JAD zapojuje klienta nebo koncové uživatele do procesu navrhování a vývoje. Zahrnuje také přístupy ke zlepšení kvality specifikace a účasti uživatelů prostřednictvím postupných společných workshopů s názvem *Jad sessions*. Vzhledem k tomu, že je klient zapojen do celého procesu vývoje, vede to k rychlejšímu vývoji a větší spokojenosti klienta. (Ing. Roman Danel, Ph.D., Analýza a projektování systémů [přednáška]. Ostrava: VŠB-TUO)

3.6 Přístupy k analýze informačních systémů

Oblast analýzy a návrhu IS můžeme rozdělit na dvě různé metodologie – objektivně orientovaný přístup a strukturovaný přístup.

3.6.1 Objektový přístup k analýze a návrhu IS

Objektově orientovaný přístup se zaměřuje na mapování struktury a chování informačních systémů v podobě malých modulů, které kombinují data i procesy. Hlavním cílem objektově orientovaného přístupu (OOP) je zvýšit kvalitu a výkon systémové analýzy.
Ve fázi analýzy se objektově orientované modely používají k zaplnění mezery mezi problémem a řešením. Funguje to ideálně v situaci, kdy jsou systémy neustále projektovány a přizpůsobeny . Probíhá identifikace objektů v problémové oblasti pomocí klasifikace z hlediska dat a chování. (Chlapek, 2005)
Výhody OOP:

- Usnadňuje změny v systému při nízkých nákladech
- Podporuje opětové použití komponent
- Zjednoduší problém integrace komponent pro konfiguraci velkého systému
- Usnadňuje navrhování distribuovaných systémů

(Chlapek, 2005)

Základní prvky objektového přístupu:

- **Objekty** – objekt je něco, co existuje v problémové doméně a lze jej identifikovat podle dat (atributu) nebo chování
- **Atributy** – popisují informace o objektu
- **Chování** – určuje, co může objekt dělat. Definuje operaci prováděnou na objektech

- **Třída** – kategorie (skupina) objektů, které mají podobné vlastnosti a stejné nebo podobné chování

- **Metody** – metody určují chování třídy. Nejsou ničím jiným než akcí, kterou může objekt provést

- **Zpráva** – zpráva je volání funkce nebo procedury z jednoho objektu na druhý. Informace zasílané objektům za účelem spuštění metod. Zpráva je v podstatě volání funkce nebo procedury z jednoho objektu na druhý

(Chlapek, 2005)

Zapouzdření

Je proces skrývání informací. Sjednocení procesu a dat do jednoho celku. Data objektu jsou skryta před zbytkem systému a jsou k dispozici pouze prostřednictvím služeb třídy. Umožňuje vylepšení nebo úpravu metod používaných objekty bez ovlivnění jiných částí systému (Chlapek, 2005).

Abstrakce

Jedná se o proces převzetí nebo výběru potřebných metod a atributů pro určení objektu. Zaměřuje se na základní vlastnosti objektu vzhledem k pohledu uživatele (Chlapek, 2005).

Vztahy

Všechny třídy v systému spolu souvisejí. Objekty neexistují izolovaně, existují ve vztahu k jiným objektům. Existují tři typy objektových vztahů: Agregace, Asociace, Generalizace (Chlapek, 2005).

Dědičnost

Dědičnost je skvělá funkce, která umožňuje vytvářet podtřídy ze stávající třídy děděním atributů a operací stávajících tříd (Chlapek, 2005).

Polymorfismus

Polymorfismus je schopnost přijímat různé formy. To platí jak pro objekty, tak i pro operace. Skutečný typ polymorfního objektu je skrytý v super třídě nebo rodičovské třídě.
V polymorfní operaci může být operace provedena různými způsoby pro různé třídy objektů. To nám umožňuje manipulovat s objekty různých tříd tím, že známe pouze jejich společné vlastnosti (Chlapek, 2005).

3.6.2 Strukturovaný přístup k analýze a návrhu IS

Strukturovaný přístup je klasickou metodou analýzy a návrhu informačních systémů, který byl navržen v 70. letech (Ken Orr, Tom DeMarco…). Tom DeMarco položil základy tohoto přístupu v roce 1979 v práci nazvané „Strukturovaná analýza a specifikace systémů“. Jeho základními doporučeními jsou:

- rozdělit systém na subsystémy
- použití grafických modelů pro vizualizaci
- tvorba logického modelu systému před implementací.

(Chlapek, 2005)

Strukturovaný přístup se vyznačuje použitím techniky dekompozice, která umožňuje rozdělit projekt na menší, přesně definované aktivity a určit pořadí a interakce mezi nimi. (Ráček, 2006).

Strukturovaný přístup modeluje systém pomocí:

ERD (Entity Relationship Diagram)

ER diagramy slouží k modelování struktury informačního systému a databáze v grafické podobě. Cílem ERD je zmapovat data ukládaná do databáze a jejich vzájemné vztahy.
Výstupem ERD je popis logické struktury databáze: **entita, atribut, relace, kardinalita vztahu, parcialita vztahu** (Ráček, 2006).

DFD (Data Flow Diagram)

Data Flow Diagram (DFD, Diagram datových toků) je jeden z nástrojů pro modelování funkcí informačních systémů. DFD je součástí strukturované analýzy a návrhu systémů. Pomocí DFD lze modelovat celé organizace.
Prvky DFD diagramu: proces, tok, sklad, terminátor (Řepa, 1999).

STD (State Transition Diagram)

Změna aspektů aspektů systému během času. STD diagram musí mít pouze jeden počáteční bod, ale může mít jeden nebo více koncových bodů.
Prvky STD diagramu: stavy, události, přechody (Řepa, 1999).

Data dictionary (Datový slovník)

Datový slovník dat obsahuje metadata (data o datech), tj. data databáze. On je velmi důležitý, protože obsahuje informace o tom, co je v databázi, kdo má přístup, kde je databáze fyzicky uložena. Uživatelé databáze obvykle nekomunikují s datovým slovníkem, je zpracováván pouze správci databáze (Řepa, 1999).

Structure chart

Slouží pro znázornění hierarchie programových modulů systému, ve tvaru grafu (stromu). Kořenem je hlavní programový modul, uzly znázorňují dílčí volané moduly (Řepa, 1999).

Flow chart (vývojový diagram)

Alternativní vyjádření algoritmu, vhodnější pro strukturované programování je strukturogram. Pro modelování systému má Flow Chart nejmenší důležitost, ale vhodný pro popisy funkcí.

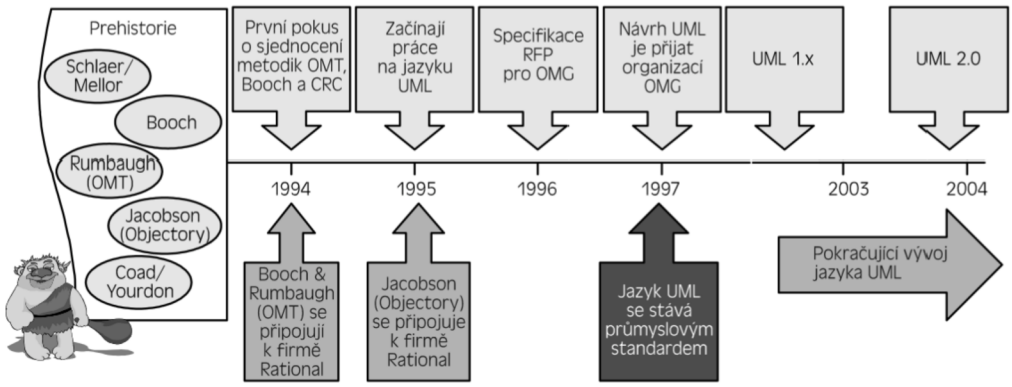
3.7 UML (Unified Modeling Language)

„Jazyk UML (Unified Modeling Language, unifikovaný modelovací jazyk) je univerzální jazyk pro vizuální modelování systémů. Přestože je nejčastěji spojován s modelováním objektově orientovaných softwarových systémů, má mnohem širší využití, což vyplývá z jeho zabudovaných rozšiřovacích mechanismů.“ (Arlow, 2007)

Jazyk UML sjednocuje nejlepší praktiky modelovacích technik a softwarového inženýrství a je vhodný pro použití všemi CASE (computer-aided software engineering) nástroji. UML se používá nejen při vývoji softwaru, ale také při zobrazování toků výrobních procesů a jiných oblastech. UML diagramy popisují jak celý systém, tak jednotlivé objekty v něm, jeho hranice, strukturu a chování. UML není programovacím jazykem, ale pomocí UML diagramů lze generovat kód v různých jazycích a existuje na to mnoho specializovaných nástrojů. Jednotný modelovací jazyk UML je úzce spojen s objektově orientovanou analýzou a návrhem. (Arlow, 2007).

3.7.1 Historie UML

Před rokem 1994 byl svět objektově orientovaných metod v chaosu. Existovalo několik soutěžících jazyků pro vizuální modelování, každá metoda přinesla něco nového, včetně nových notací a kvalit. Tyto metody měly své stoupence i protivníky, ale nejvíce trhu si mezi sebe rozdělily metody OMT (Object Modeling Technique, Rumbaugh) a Booch (Booch). Objectory (Jacobson) měla nejlepší postavení mezi metodikami. Tyto metody měly pouze syntaxi pro vizuální modelování a různé aforismy a poučky. (Arlow, 2007).



Obrázek 10: Rozvoj objektově orientovaných metod (Zdroj: Arlow, 2007)

Metodika Fusion, kterou vytvořil Coleman v roce 1994, se pokusila o sjednocení tehdejšího trhu s metodikami pro vizuální modelování, avšak autoři metod Booch, Jacobson a Rumbaugh nebyli do jejího vývoje zapojeni. Fusion se nakonec neujala a propadla v zapomnění, zatímco autoři metod Booch a Rumbaugh spojili síly v firmě Rational Corporation, kde pracovali na vytvoření jazyka UML. Tento fakt znepokojoval mnoho lidí, protože Rational najednou ovládla většinu trhu s metodikami. V roce 1996 OMG (Object Management Group) navrhla specifikaci RFP pro vizuální modelování a jako standard byl navržen jazyk UML. V roce 1997 OMG jazyk UML přijalo a stal se prvním průmyslovým standardem objektově orientovaného jazyka pro vizuální modelování. Od té doby byl UML široce přijat veřejností. (Arlow, 2007)

3.7.2 Základní charakteristiky a modelování v rámci UML

Existují tři základní typy diagramů v jazyce UML:

- Funkční – tyto diagramy zahrnují schémata případů použití, které popisují funkčnost systému z pohledu uživatele. Tyto diagramy ukazují, jak uživatelé budou interagovat se systémem a jakým způsobem systém odpovídá na jejich požadavky.
- Objektové – tyto diagramy popisují strukturu systému pomocí objektů, atributů, vztahů a operací. Tyto diagramy umožňují vizualizovat, jak jsou objekty ve vzájemném vztahu a jak spolu komunikují.
- Dynamické – tyto diagramy se dělí na několik kategorií, jako jsou diagramy interakce, stavu a aktivity. Tyto diagramy popisují vnitřní fungování systému v čase a ukazují, jak se systém chová v různých situacích a jaké jsou jeho reakce na různé události. (Arlow, 2007).

Strukturální diagramy (např. diagramy tříd) popisují statickou strukturu systému a vztahy mezi jeho částmi, zatímco dynamické diagramy (např. diagramy stavů) popisují chování systému v čase a interakce mezi jeho částmi. (Arlow, 2007).

Struktura jazyka UML

Nejlepší způsob, jak pochopit funkci UML jako vizuálního jazyka, je prostřednictvím jeho struktury. Booch (1999) popisuje strukturu jazyka UML jako skládající se ze tří částí:

- Stavební bloky – jsou základní prvky, z nichž se skládají modely, diagramy a relace
- Společné mechanismy – jsou obecné metody používané k dosažení specifických cílů v UML.
- Architektura – představuje pohled na architekturu navrhovaného systému v rámci UML.

(Arlow, 2007)

Stavební bloky jazyka UML

Podle knihy The Unified Modeling Language User Guide (Booch, 1999) je jazyk UML sestaven ze tří stavebních bloků:

- z předmětů (things) - samotné prvky modelu,
- vztahů (relationships) - souvislost mezi dvěma a více předmětů
- diagramů (diagrams) - pohledy na modely UML

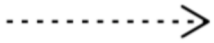
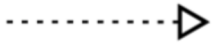
(Arlow, 2007)

1. Předměty

Strukturní abstrakce, *chování*, *seskupení* a *poznámky* jsou čtyři hlavní kategorie prvků v modelu UML. Strukturní abstrakce zahrnují podstatná jména, jako jsou třídy, rozhraní, spolupráce a případy užití. Chování se zaměřuje na slovesa modelu UML, jako jsou interakce a stavy. Seskupení umožňuje sémanticky souvisejícím prvkům být sjednoceny do soudržných jednotek. Poznámky jsou komentáře, které lze připojit k prvku modelu za účelem popisu nebo vysvětlení.(Arlow, 2007)

2. Vztahy (relace)

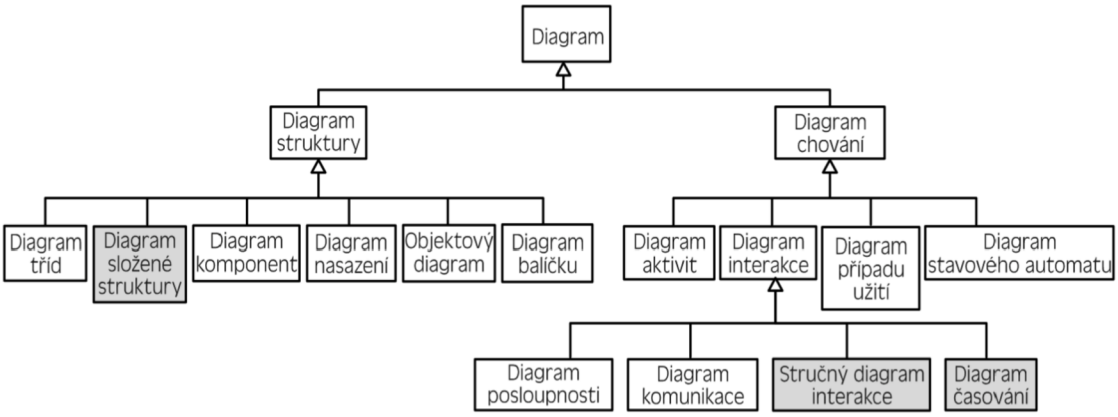
Relace v modelu UML popisují vztahy mezi různými prvky, jako jsou strukturní abstrakce a seskupování.

Typ relace	Syntaxe UML zdroj cíl	Stručný popis
Závislost (Dependency)		Změna v určitém předmětu ovlivňuje význam závislého předmětu.
Asociace (Association)		Popis množiny spojení mezi objekty.
Agregace (Aggregation)		Cílový prvek je součástí zdrojového prvku
Kompozice (Composition)		Silnější forma agregace (má více omezení)
Ochranná nádoba (Containment)		Zdrojový prvek obsahuje cílový prvek
Zobecnění (Generalization)		Jeden prvek je specializací jiného prvku a lze jej nahradit obecnějším (univerzálnějším) prvkem.
Realizace (Realization)		Asociace mezi klasifikátory, kde jeden klasifikátor určuje dohodu, jejíž uskutečnění zaručuje druhý klasifikátor

Obrázek 11: Typy relace (vztahů) (Zdroj: Arlow, 2007)

3. Diagramy

Diagramy UML jsou reprezentací modelu a poskytují pohled na něj. Na rozdíl od modelu mohou být z diagramu odstraněny relace a předměty, ale ty stále zůstávají v modelu. To je hlavní rozdíl mezi diagramem a modelem, jak uvádí Arlow (2007). Existuje celkem třináct různých typů diagramů UML, a mezi ně patří například diagramy tříd, aktivit, komponent, případů užití nebo sekvenční diagramy. Novějším typem diagramu, který byl zaveden v jazyce UML, je diagram se šedým pozadím, který zobrazuje interakce mezi objekty a událostmi.



Obrázek 12: Typy diagramů (Zdroj: Arlow, 2007)

UML se skládá z množství objektů, které jsou navzájem propojeny různými způsoby, aby se vytvořily diagramy reprezentující statické (strukturální) a dynamické (chování) aspekty systému. Tento přístup umožňuje vytvoření přesného a srozumitelného modelu, který lze použít k návrhu a implementaci systému (Arlow, 2007).

Strukturální diagramy UML

- **Diagramy tříd** – podle Arlowa (2007) jsou diagramy tříd nejčastěji používanou variantou diagramů UML a představují základ objektově orientovaného řešení. Tyto diagramy zobrazují třídy v rámci systému spolu s jejich atributy, operacemi a vztahy mezi třídami. Používají se zejména při modelování větších systémů.

- **Diagram složené struktury** – tyto diagramy jsou využívány pro znázornění vnitřní struktury třídy (Arlow, 2007).

- **Diagramy komponent** – popisují vztahy mezi komponentami softwarového systému a slouží především k práci s rozsáhlými strukturami, kde se vyskytují mnohé komponenty. V takových strukturách se mezi komponentami vytvářejí interakce, které se realizují pomocí rozhraní (Arlow, 2007).

- **Diagram nasazení** – popisuje, jak je systém implementován na hardwarové a softwarové úrovni (Arlow, 2007)

- **Objektový diagram** – zobrazují, jak systém interaguje s uživateli a dalšími systémy, což je ilustrováno pomocí příkladů z reálného světa. Tyto diagramy také ukazují, jak bude systém fungovat v určitém časovém okamžiku. (Arlow, 2007)

- **Diagram balíčku** – balíčky v diagramu UML mohou reprezentovat různé úrovně systému a umožnit tak opětovné vytvoření jeho architektury (Arlow, 2007)

Diagramy chování

Diagram aktivit – znázorňují průběh pracovních nebo provozních procesů v rámci různých komponent nebo částí systému. Tyto diagramy slouží jako alternativa k diagramům stavů a umožňují podrobnější popis toku činnosti v systému (Arlow, 2007).

Diagram interakce – tento typ diagramu je podobný sekvenčnímu diagramu, nicméně jeho zaměření je na komunikaci mezi objekty pomocí zpráv (Arlow, 2007)

.

Diagram posloupnosti – zobrazují postupnou interakci mezi objekty a popisují, jak objekty spolu komunikují v rámci daného scénáře (Arlow, 2007)

Diagram komunikace – tyto diagramy jsou podobné sekvenčním diagramům, avšak klíčovým rozdílem je jejich zaměření na zprávy, které jsou předávány mezi objekty. Sekvenční diagramy mohou být použity k přenosu stejných informací pomocí odlišné sady objektů. (Arlow, 2007)

Stručný diagram interakce – Arlow (2007) uvádí, že existuje sedm různých typů interakčních diagramů, které se používají v objektově orientovaném modelování, a tento konkrétní diagram popisuje jejich pořadí použití.

Diagramy časové osy – podobné sekvenčním diagramům, neboť ukazují chování objektů v daném časovém rozmezí. Základní diagramy zobrazují jediný objekt, zatímco složitější

diagramy ukazují interakci několika objektů v průběhu určitého časového období (Arlow, 2007).

Diagram případu užiti – tyto diagramy znázorňují konkrétní funkce systému a ukazují, jak jsou spojeny s interními nebo externími ovladači (Arlow, 2007).

Diagramy stavů – podobně jako diagramy aktivit zobrazují chování objektů, avšak zaměřují se na to, jak se objekty chovají v různých stavech (Arlow, 2007).

Mechanika jazyka UML

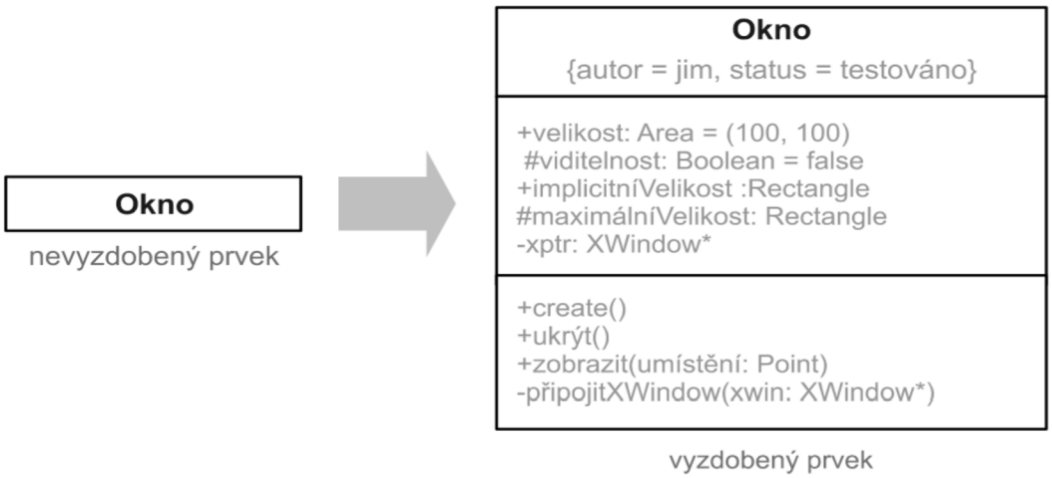
UML jazyk má čtyři základní mechanismy, které jsou konzistentně používány v celém jazyce. Tyto mechanismy popisují čtyři různé způsoby modelování objektů (Arlow, 2007).

1. Specifikace

UML modely jsou charakterizovány minimálně dvěma aspekty – grafickým a textovým. Grafický aspekt umožňuje vizualizaci modelu prostřednictvím diagramů a symbolů, zatímco textový aspekt se skládá ze specifikací jednotlivých prvků modelu. Tyto specifikace popisují sémantiku prvků modelu pomocí textových popisů. Soubor těchto specifikací tvoří *jádro* modelu a poskytuje sémantický základ pro udržení modelu pohromadě a jeho porozumění (Arlow, 2007).

2. Ornamenty

UML má výhodu v tom, že každý prvek modelu je reprezentován jednoduchým symbolem, ke kterému lze přidat řadu ornamentů, aby byly zobrazeny další informace v rámci diagramu. Tento přistup umožňuje snadné zvýrazněni a zdůraznění důležitých detailu pro zlepšení čitelnosti a srozumitelnosti diagramu (Arlow, 2007).



Obrázek 13: Rozšíření modelu pomocí ornamentu (Zdroj: Arlow, 2007)

3. Podskupiny

Dělení podskupin v jazyce UML zahrnuje dvě různá vidění světa:

- Skupina rozhraní a implementace – tato část diagramů se soustředí na vztah mezi klasifikátorem, který reprezentuje abstraktní koncept určitého objektu, a jeho instancemi, což jsou konkrétní výskyty tohoto objektu v systému (Arlow, 2007).

- Skupina klasifikátorů a instancí – tato skupina se soustředí na schopnosti objektu (rozhraní) a způsob, jakým tyto schopnosti jsou implementovány (implementace) v rámci klasifikátorů a instancí (Arlow, 2007).

4. Mechanismy rozšiřitelnosti

UML byl navržen tak, aby byl otevřený pro kontrolované rozšíření, což znamená, že umožňuje přidání dalších prvků nebo funkcionalit.

Mechanismy rozšíření v UML jsou:

- Omezení – mechanismem omezení UML, který rozšiřuje sémantiku stavebního bloku a umožňuje přidávat nová pravidla nebo upravovat stávající. Je v modelu vyznačen jako textový řetězec uzavřený do složených závorek „{}“ (Arlow, 2007).

- Stereotyp – mechanismus stereotypů v jazyce UML umožňuje rozšíření slovní zásoby jazyka a vytváření nových prvků modelu na základě již existujících prvků. Stereotyp je variací stávajícího prvku s tím, že si uchovává jeho tvar, ale liší se v podstatě. Stereotyp je zobrazen jako název, uzavřený v uvozovkách, umístěný nad názvem existujícího prvku. (Arlow, 2007).

- Označené hodnoty – tato vlastnost rozšiřuje možnosti specifikace prvků v jazyce UML, umožňuje přidávat nové informace do popisu prvku. Označené hodnoty, které jsou zobrazeny jako řetězec v závorkách pod názvem prvku. (Arlow, 2007).

Architektura UML

Při tvorbě softwarových systémů je důležité mít na zřeteli různé perspektivy a pohledy na systém. Koncoví uživatelé, analytici, vývojáři, systémoví integrátoři, testeři a projektoví manažeři mají každý své vlastní zájmy a pohledy na systém v různých fázích jeho životního cyklu. Pro správné řízení těchto pohledů a podporu iterativního a přírůstkového vývoje systému je nejdůležitějším artefaktem jeho architektura. (Arlow, 2007).

- Pohled případu užití (Use Case view) - pokrývá případy použití, které popisují chování systému pozorované koncovými uživateli, analytiky a testery. Tento druh nespecifikuje skutečnou organizaci softwarového systému, ale hybné síly, na kterých závisí tvorba architektury systému. V UML jsou statické aspekty tohoto typu přenášeny pomocí diagramů případů užití (Use-Case Diagram) a dynamické aspekty pomocí diagramů interakce (Sequential Diagram a Collaboration Diagram), stavů (Statechart Diagram) a akcí (Activity Diagram) (Arlow, 2007).

- Logický pohled (Design view) - zabývá se třídami a objekty, které popisují návrhový problém a jeho řešení. Tento pohled se zaměřuje zejména na funkční požadavky a služby, které systém musí poskytovat koncovým uživatelům. Statické aspekty tohoto pohledu se obvykle reprezentují pomocí diagramů tříd a objektů, zatímco dynamické aspekty lze zobrazit pomocí diagramů interakce, stavů a akcí (Arlow, 2007).

- Pohled procesů (Process view) zahrnuje mechanismy paralelismu a synchronizace v softwarovém systému, jako jsou spustitelná vlákna a procesy. Cílem tohoto pohledu je popsat výkon a propustnost systému. Statické a dynamické aspekty jsou v UML zobrazovány pomocí stejných diagramů jako v logickém pohledu, ale se zaměřením na aktivní třídy, které reprezentují spustitelná vlákna a procesy (Arlow, 2007).

- Pohled implementace (Implementation view) - popisuje fyzickou organizaci softwarového systému, která zahrnuje komponenty a soubory potřebné pro vytvoření a nasazení kódu systému. Tento pohled se zaměřuje na správu konfigurace systému, která sestává z nezávislých komponent a souborů, které lze kombinovat různými způsoby. Statické aspekty tohoto pohledu jsou reprezentovány pomocí diagramů komponent a dynamické aspekty pomocí diagramů interakce, stavů a akcí v jazyce UML (Arlow, 2007).

- Pohled nasazení (Deployment view) - tento pohled se zaměřuje na soubor fyzických výpočetních uzlů, které utvářejí hardwarovou architekturu softwarového systému. Hlavním úkolem tohoto pohledu je popsat distribuci, dodávku a instalaci jednotlivých komponent hardwaru systému. V UML jsou statické aspekty tohoto pohledu reprezentovány diagramy nasazení (Deployment Diagram), zatímco dynamické aspekty jsou reprezentovány diagramy interakce, stavů a akcí (Arlow, 2007).

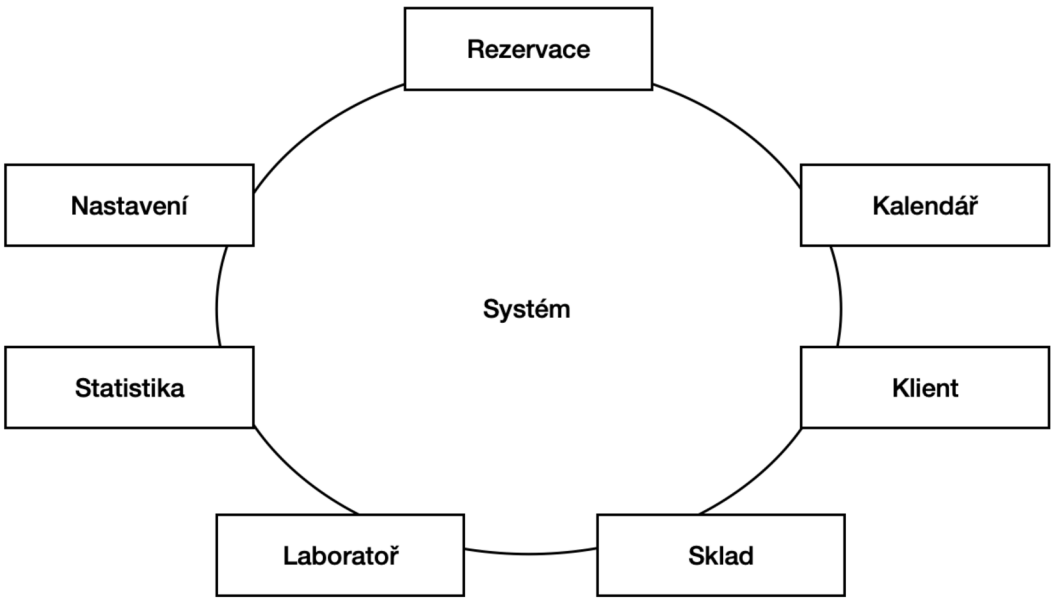
4 Vlastní práce

Praktická část diplomové práce se zaměřuje na analýzu a návrh informačního systému, který bude sloužit k podpoře prostřednictví služeb v oblasti medicíny a genetických testů. Autor dospěl k takovému závěru po osobní zkušenosti práce a studia vnitřní struktury společnosti. Hlavním cílem je ulehčit a optimalizovat práci společnosti pomocí informačního systému.

Dalším krokem bude analýza požadavků, které byly získány od potenciálních uživatelů. Tyto požadavky jsou hlavním základem pro vytvoření modelů – tříd, interakcí a stavů. A pro vizualizaci systému bude použit – wireframe

4.1 Popis systému

Předpokládaný informační systém efektivně podporuje celý proces práce s klienty, databází, laboratoří a skladem, automatizuje práci společnosti. Stejně umožňuje sledovat celkovou statistiku prodejů a provedených testů.



Obrázek 14: Wireframe – Domovská stránka

Na obrázku... je znázorněna schéma předpokládaného systému. Jak je znázorněno na schématu, jsou zde všechny potřebné funkce, které pomohou maximalizovat výnosy, automatizovat procesy, kontrolovat zboží a laboratoře společnosti. Systém poskytuje přístup

ke všem rezervacím a informacím o nich. Také jsou zde údaje o zákaznících a všechny jejich poslední testy. V záložce "sklad" bude probíhat kontrola dostupných produktů. Stejně má funkci pro zpětnou vazbu s laboratoří a různé zdravotní záznamy, vzorky a biologické materiály. A samozřejmě celková statistika provedených testů a příjmů.

4.2 Analýza požadavků

V této kapitole je popsána analýza požadavků pro systém, která je základem celé informační systémy. Data byla získána s pomocí osobní zkušenosti práce a pozorování, stejně jako na základě rozhovoru s kolegy, kteří jsou potenciálními uživateli. Požadavky byly rozděleny do dvou kategorií – funkční a nefunkční.

4.2.1 Funkční požadavky

přihlášení do systému

Každému uživateli bude k dispozici funkce autorizace do systému.
Základní funkce – autorizace do systému, odhlašování ze systému, změna e-mailu a hesla.

obsahuje kalendář

V kalendáři bude vidět rozvrh. Také se zobrazí datum s rezervací a typy testů, které lidé vybrali.
Základní funkce – připojení k dalším modulům, sledování rezervovaných termínů

správa klientů

Tato funkce bude obsahovat databázi všech klientů a stejně prohlížení údajů o všech testech, na kterých byl klient.
Základní funkce – vyhledání klientů, změna údajů klienta, přístup k jejich údajům a testům.

kontrola rezervace

V tomto modulu budou viditelné rezervace na dnešní období a možnost přidat nebo odstranit rezervace.
Základní funkce – přidání rezervace, odstranění rezervace, prohlížení rezervací.

sběr statistiky

V modulu statistik bude k dispozici všechna informace o počtu provedených testů, jejich kategoriích a zisku.
Základní funkce – třídění podle určitých kategorií

kontrola zboží

V tomto modulu k dispozici aktuální dostupnost zboží a možnost objednat chybějící zboží ze skladu.
Základní funkce – aktuální dostupnosti zboží, propojení se skladem, objednání chybějících zboží.

spojení s laboratoří

V tomto modulu je hlavní funkcí přímý kontakt s laboratoří. Systém by měl být schopen řídit laboratorní informace, včetně sledování vzorků, řízení výsledků testů a tvorby reportů.
Základní funkce – kontakt s laboratoří, sledování vzorků, čas analýzy a aktuální stav.

správa uživatelů

v tomto modulu bude možnost změnit přístupová práva pro každého uživatele – od běžných administrativních pracovníků až po majitele firmy.

4.2.2 Nefunkční požadavky

Systém bude bezpečný

Aby byl systém zabezpečen, bude použit protokol HTTPS s využitím TLS pro autentifikaci.

Systém bude schopen zpracovávat velké množství dat s rychlým a spolehlivým výkonem

Systém bude přístupný z webového serveru pomocí webového prohlížeče a bude podporovat prohlížeče Chrome, Firefox, Opera, Edge a Safari s jednotnou implementací chování a vzhledu, aby byl nezávislý na operačním systému uživatele.

Systém bude škálovatelný, aby uspokojil budoucí růst počtu zákazníků a objemu dat

Horizontální škálování (scaling out): při tomto postupu se zvyšuje počet fyzických uzlů systému, na kterých jsou umístěny jeho komponenty, což umožňuje zpracovávat více dotazů a zvyšovat propustnost systému.
Vertikální škálování (scaling up): při tomto postupu se zvyšuje výkon fyzických uzlů systému, například přidáním procesorů, operační paměti nebo diskového prostoru. Tento přístup umožňuje zpracovávat složitější úlohy a zvyšovat výkon systému.
Využití cloudových technologií: cloudové služby umožňují horizontální i vertikální škálování systému pomocí automatického škálování zdrojů na základě zátěže na systém.

Systém bude schopen zvládat více uživatelů a současný přístup k datům

Systém bude podporovat souběžný přístup k databázi pro více uživatelů. K tomu lze použít architekturu klient-server, kde klienti odesílají požadavky na server a server zpracovává požadavky a vrací odpovědi. Každý klient má své jedinečné ID, aby server mohl sledovat jejich aktivity a poskytovat přístup k datům pouze příslušným uživatelům.

Systém bude mít robustní mechanismus zálohování a obnovení

S ohledem na předpokládané velké množství dat v systému, je důležité rozhodnout o nejefektivnější frekvenci zálohování, zohledňující zároveň náklady na různé úrovně zálohování. Po provedení analýzy bylo rozhodnuto, že pro zálohování systému budou každodenně využívána nová data. Dále bude určen interval pro zálohování celého systému.

Systém bude dodržovat obecné nařízení o ochraně osobních údajů (GDPR).

To zahrnuje ochranu osobních údajů, jako jsou jméno, pohlaví, věk, e-mailová adresa, datum narození a adresa, a které slouží k identifikaci fyzických osob. Toto nařízení se bude vztahovat jak na klienty, tak na uživatele systému.

Systém bude rozšiřitelný

Systém by měl být připraven na velké změny a přidání nových testů a nových funkcí.

4.3 Návrh systému

Na základě teoretické části a analýzy požadavků byly vytvořeny UML diagrama tříd, model stavu a model interakce. Při tvorbě modelů byl použit jazyk UML a online webová aplikace drawio. V závěrečné části práce jsou znázorněny wireframe systému a dostupné způsoby realizace.

4.3.1 Model tříd

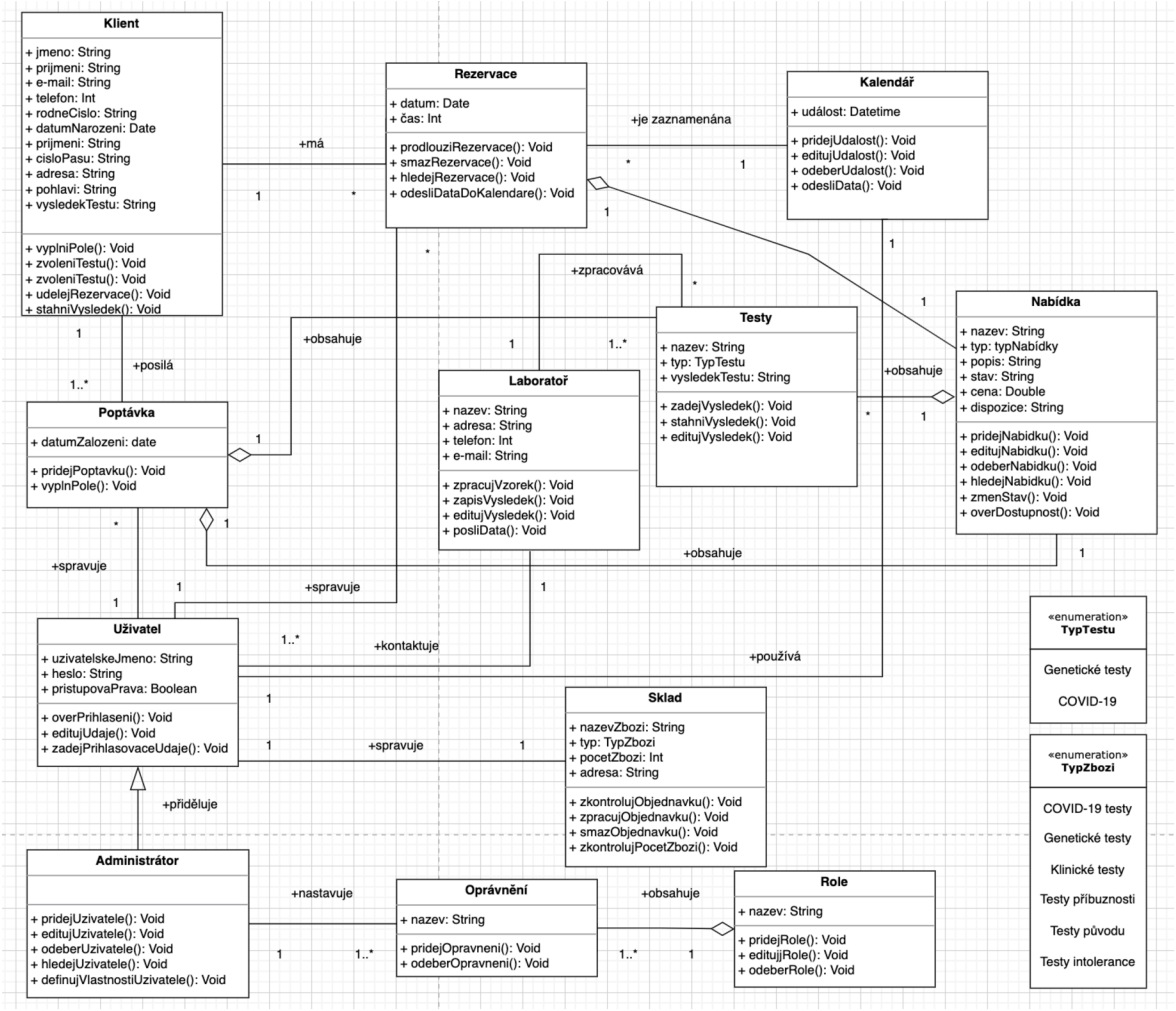
V praktické části práce byl vytvořen UML diagram tříd, který modeluje různé třídy a jejich vztahy v genetické společnosti. Nejprve byly analyzovány teoretické požadavky a identifikovany klíčové třídy, které by měly být zastoupeny v diagramu. Tyto třídy zahrnovaly entity jako Klient, Rezervace, Sklad, Laboratoř, Uživatel, Administrátor, Poptávka, Nabídka, Role, Oprávnění, Kalendář a Testy.

Pomocí UML notace byl vytvořen diagram tříd a online nástroje drawio. V tomto diagramu byly definovany atributy a metody pro každou třídu a také reprezentovány vztahy mezi nimi. Například třída Rezervace měla vztah s třídou Kalendář, což znamenalo, že rezervace je spojena s konkrétním datem a časem.

Diagram tříd poskytl jasnou vizuální reprezentaci struktury systému a pomohl nám identifikovat potenciální problémy nebo nejasnosti v požadavcích. Stejně sloužil jako

užitečný nástroj během implementační fáze, aby vývojáři mohli odkazovat na diagram pro pochopení vztahů mezi třídami.

Byly identifikovány atributy pro každou třídu a označeny je uvnitř obdélníku. Atributy představují vlastnosti nebo charakteristiky třídy a stejně byly identifikovány metody každé třídy. Metody představují akce nebo funkce, které třída může provádět. Dalším krokem určeny vztahy mezi třídami. V tom diagramu byly použity vztahy generalizace, agregaci a asociaci. Pevná šipka směřující z jedné třídy na druhou označuje vztah generalizace, zatímco diamantový tvar na konci šipky označuje agregační vztah. Jednoduchá šipka označuje vztah asociace a posledním krokem byly přidány notace multiplicity k označení počtu instance jedné třídy, které mohou být spojeny s jednou instancí druhé třídy.



Obrázek 15: Diagram tříd (Zdroj: Vlastní zpracování)

Datový slovník

Třída Uživatel

Třída *Uživatel* zahrnuje jednotlivé osoby, které mají zaregistrované účty v systému. Způsob, jakým mohou uživatelé v systému pracovat, je ovlivněn třídou *Oprávnění*.

Třída Administrátor

Třída *Administrátor* je podtřídou třídy *Uživatel* a dědí její atributy a operace, což naznačuje generalizační spojení mezi těmito třídami. Hlavní funkcí třídy *Administrátor* je však správa všech uživatelů systému.

Třída Oprávnění

Třída *Oprávnění* řídí používání systémových funkcí různými uživateli. *Administrátor* má možnost nastavit oprávnění pro každou jednotlivou systémovou funkci podle potřeb provozu.

Třída Role

Tato třída obsahuje základní opravy pro specifické uživatele. Vztah mezi třídami Oprávnění a Role je typu agregace, což znamená, že Role se skládá z oprávnění, které jsou jeho součástí.

Třída Klient

Tato třída slouží jako modul pro všechny zákazníky a uchovává jejich základní osobní údaje. Správu klientů zajišťuje třída *Uživatel*

Třída Poptávka

představuje žádost zákazníka o konkrétní test. Mezi třídami *Poptávka* a *Testy* existuje vztah agregace, stejně jako mezi *Poptávkou* a *Nabídkou*. To znamená, že jedna třída je součástí druhé. Správu *Poptávek* zajišťuje třída *Uživatel*.

Třída Nabídka

Tato třída představuje konkrétní nabízené služby (testy). Nabídku spravuje třída *Uživatel*

Třída Sklad

Tato třída je modulem pro správu skladu, který ukládá všechny hlavní zboží. Za správu skladu odpovídá *Uživatel*, k tomu však musí mít uživatel speciální přístupová práva.

Třída Laboratoř

Tato třída se používá k modelování a reprezentaci laboratoře v genetické společnosti. Laboratorní třídu lze použít ke sledování a zpracovávání genetických a covid testů.

Třída Testy

Tato třída se používá k ukládání informací o konkrétním testu, který se nabízí společnost. Kromě toho lze testovací třídu použít k přidružení testu k jiným třídám, jako je laboratoř, poptávka a nabídka v závislosti na konkrétním případě použití.

Třída Rezervace

Tato třída se používá k reprezentaci rezervace provedené klientem na konkrétní čas a datum pro použití zdrojů, jako je služby nabízené genetickou společností. Obvykle obsahuje informace, jako jsou kontaktní údaje klienta, datum a čas rezervace, rezervované zdroje nebo služby a stav rezervace (např. potvrzené, čekající, zrušené). Třída může také zahrnovat metody správy a aktualizace rezervace, jako je potvrzení nebo zrušení rezervace nebo přidání nebo odebrání zdrojů nebo služeb. Za správu rezervace odpovídá *Uživatel*.

Třída Kalendář

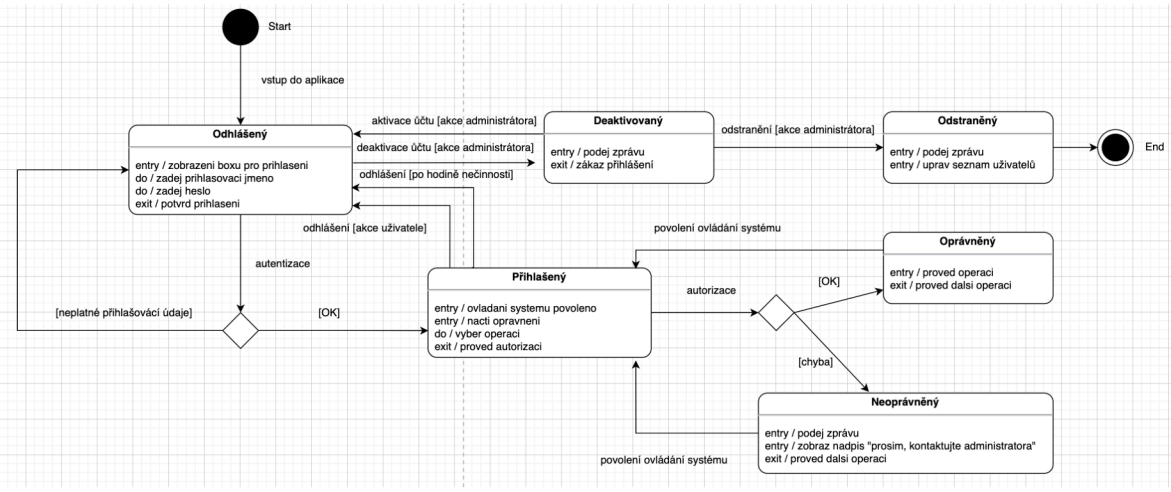
Tato třída se používá ke správě a plánování rezervací v genetické společnosti. Třídu kalendáře lze použít k zobrazení a správě nadcházejících rezervací, stejně jako k vytváření nových rezervací a k aktualizaci nebo zrušení stávajících rezervací. Třída *Uživatel* používá a upravuje kalendář. Všechny rezervace se automaticky zapisují do kalendáře.

4.3.2 Stavovy model

Stavový diagram slouží k popisu chování systému nebo jeho části a k reprezentaci různých stavů, které systém nebo objekt může nabývat. Tyto stavy se navzájem propojují a mezi nimi dochází k přechodům, během nichž se provádějí určité akce. Stavové diagramy se běžně používají při vývoji softwaru k modelování chování systému a k navrhování a implementaci logiky systému. Konkrétně pro vývoj softwaru byly vybrány a rozebrány třídy *Uživatel* a *Laboratoř*.

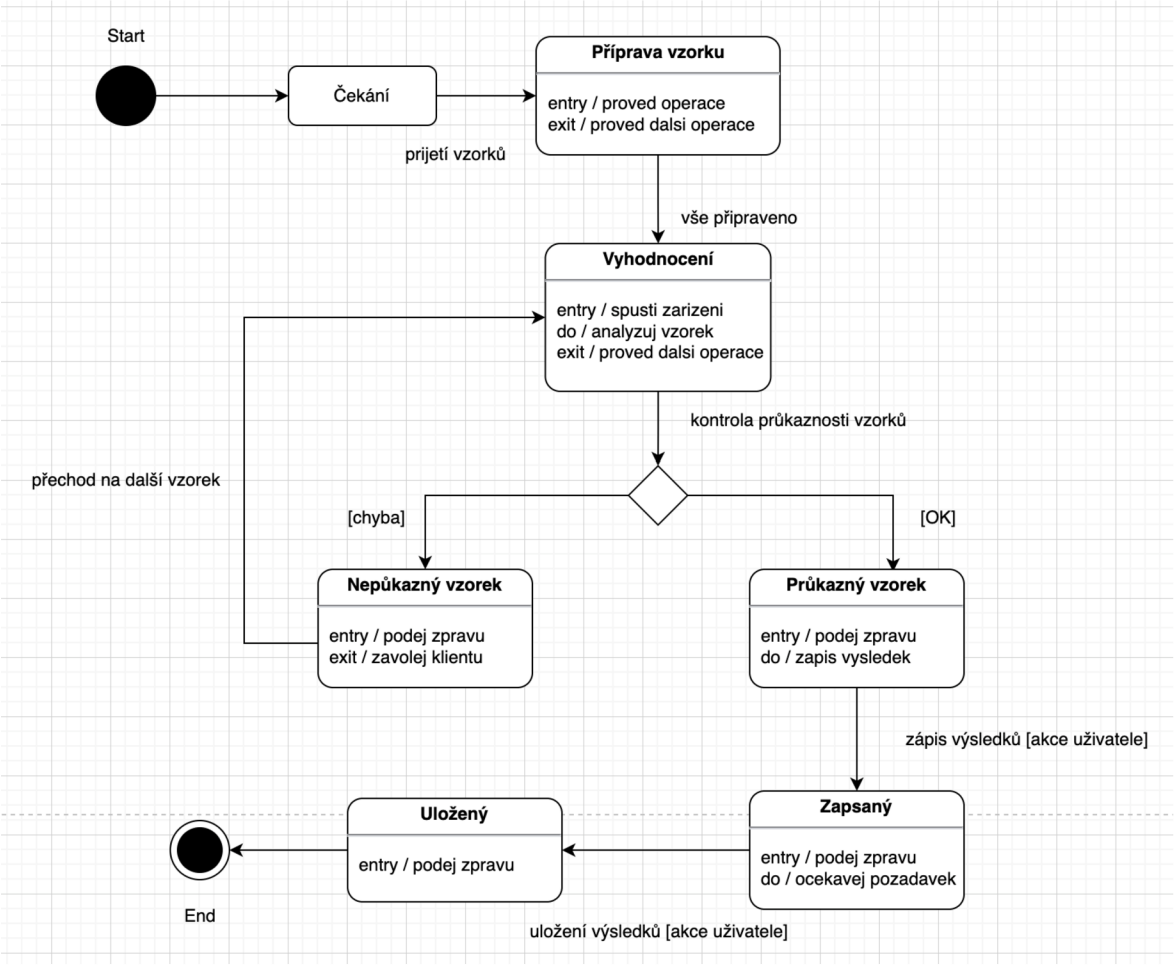
Stavový diagram pro třídu Uživatel

Stavový diagram třídy Uživatel na obrázku č.16 zobrazuje fáze, kterými tato třída prochází. Stavový diagram reprezentuje různé stavy, ve kterých může být uživatel zaregistrován v systému. Třída Uživatel má šest různých stavů v systému: *Přihlášený*, Odhlášený, *Oprávněný*, *Neoprávněný*, *Odstraněný* a *Deaktivovaný*.



Obrázek 16: Stavový diagram pro třídu Uživatel

Při otevření aplikace je uživatel automaticky Odhlášený a musí se přihlásit správnými přihlašovacími údaji a projít autentizací, aby se dostal do stavu Přihlášení a měl přístup k operacím a kontrole systému. Pokud uživatel zadá nesprávné přihlašovací údaje, zůstane v stavu Odhlášený. Po úspěšné autorizaci se uživatel stává Oprávněným a může provádět požadované akce v systému, po dokončení, kterých se vrátí do stavu Přihlášený. V případě neúspěšné autorizace se uživatel stane Neoprávněným a musí informovat administrátora o neúspěšné autorizaci. Pokud uživatel neaktivní déle než hodinu, může se odhlásit manuálně nebo být automaticky odhlášen. Administrátor může deaktivovat účet uživatele, čímž se uživatel dostane do stavu Deaktivovaný a nebude se moci přihlásit do systému. Uživatel může být buď odstraněn administrátorem ze stavu Deaktivovaný, nebo se může sám odhlásit a přejít do stavu Odhlášený, pokud je administrátorem opět aktivován. Třída Uživatel je v konečném stavu, když je odstraněna.



Obrázek 17: Stavový diagram pro třídu Laboratoř

Fáze, kterými prochází třída, jsou znázorněny ve stavovém diagramu pro třídu Laboratoř na obrázku č.17 . Existuje sedm různých stavů systému pro třídu Laboratoř: *Čekání*, *Příprava vzorku*, *Vyhodnocení*, *Neprůkazný vzorek*, Průkazný vzorek, *Zapsaný* a *Uložený*.
Laboratoř začíná ve stavu Čekání, jakmile jsou vzorky doručeny a přijaty. Následuje Příprava vzorků pro analýzu. V tomto kroku dochází k rozdělení vzorků podle priority a přiřazení čísel. Když je vše připraveno, následuje přechod k fázi Vyhodnocení – zařízení jsou spuštěna a probíhá analýza vzorků. Po dokončení tohoto kroku dochází k ověření kvality odebraných vzorků, pokud vše proběhlo úspěšně, systém pošle zprávu o přechodu na další fázi, ale pokud je nalezena nějaká chyba, kontaktují se s klientem a požádají ho, aby přišel odevzdat test znovu a systém přechází na analýzu dalšího testu. Po úspěšném provedení analýzy systém Zapisuje výsledky a posledním krokem je Uložení výsledků a odeslání na telefonní číslo a e-mail.

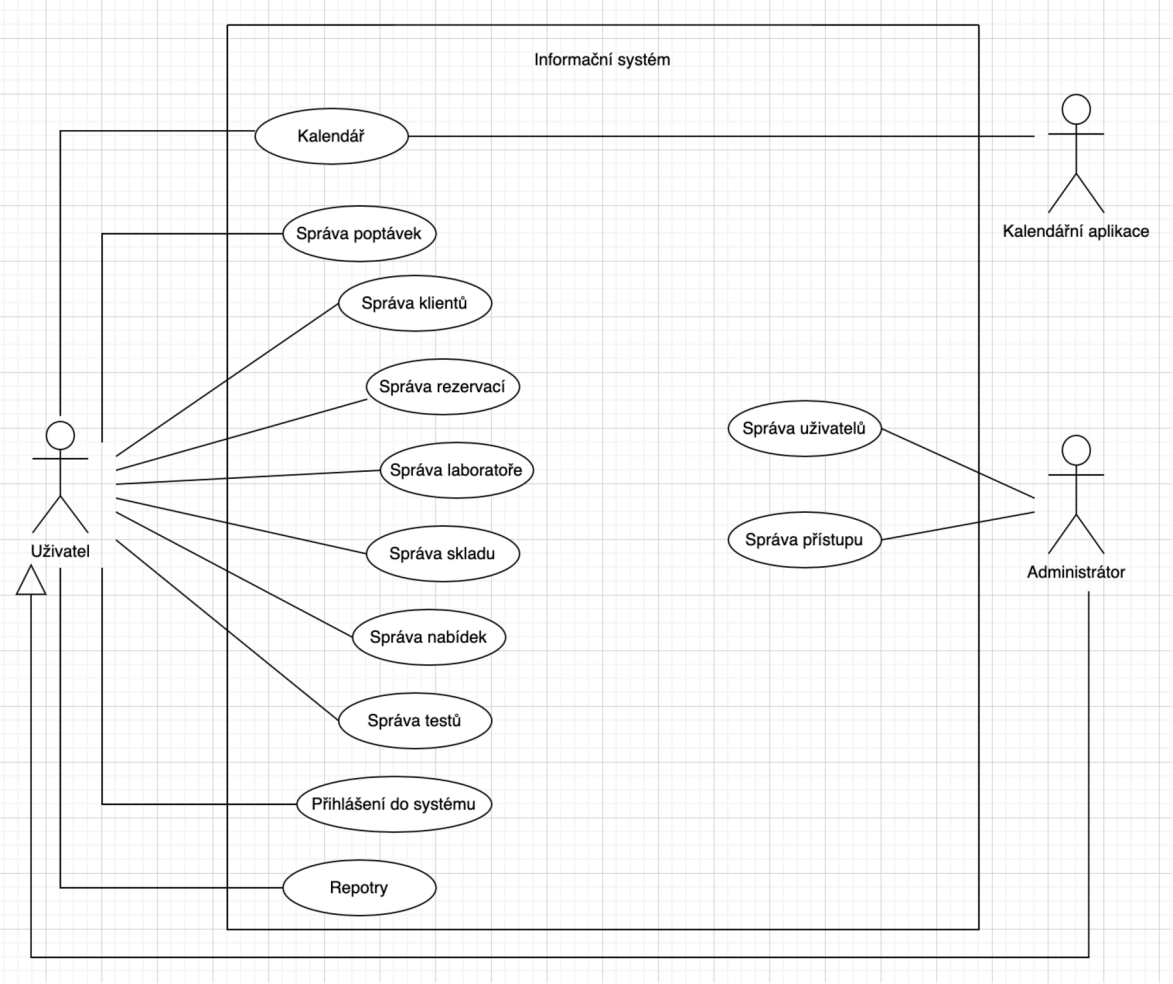
4.3.3 Model interakci

Model interakcí zobrazuje spolupráci objektů v systému a skládá se ze tří diagramů: Use Case (Diagram případů užití), Sekvenční a Aktivit diagram. Tento typ diagramu je často využíván v Unifikovaném modelovacím jazyku (UML) a slouží k ilustrování interakcí mezi objekty nebo komponentami v systému. Jeho hlavním účelem je vizualizace chování objektů a jejich interakcí v reálném čase.

Use Case (Diagram případů užití)

Diagram případů užití je další diagramový typ v rámci UML, který se používá k ilustraci interakcí mezi aktéry a systémem. V tomto diagramu aktéři představují vnější entity, které interagují se systémem, a případy užití popisují různé způsoby, jakými mohou tyto entity se

systémem spolupracovat. Použití těchto diagramů umožňuje modelovat funkční požadavky systému.



Obrázek 18: Use Case pro celkový pohled na systém

Aktéři na diagramu zahrnují Uživatele a Administrátora. Uživatel je reprezentací fyzické osoby, která využívá funkce systému v souladu se svými oprávněními a má tam svůj uživatelský účet. . Administrátor má na starost všechny uživatele v systému a zajišťuje, aby vše fungovalo správně. Diagram ukazuje, že existuje vztah generalizace mezi uživatelem a administrátorem. V důsledku toho Administrátor zahrnuje své vlastní případy užití a zároveň dědí všechny případy užití od uživatele. Kalendářní aplikace je dalším identifikovaným představitelem, se kterým systém komunikuje. Kalendářní aplikace je soukromý kalendář uživatelů, do kterých se zadává sdílená informace z kalendáře systému.

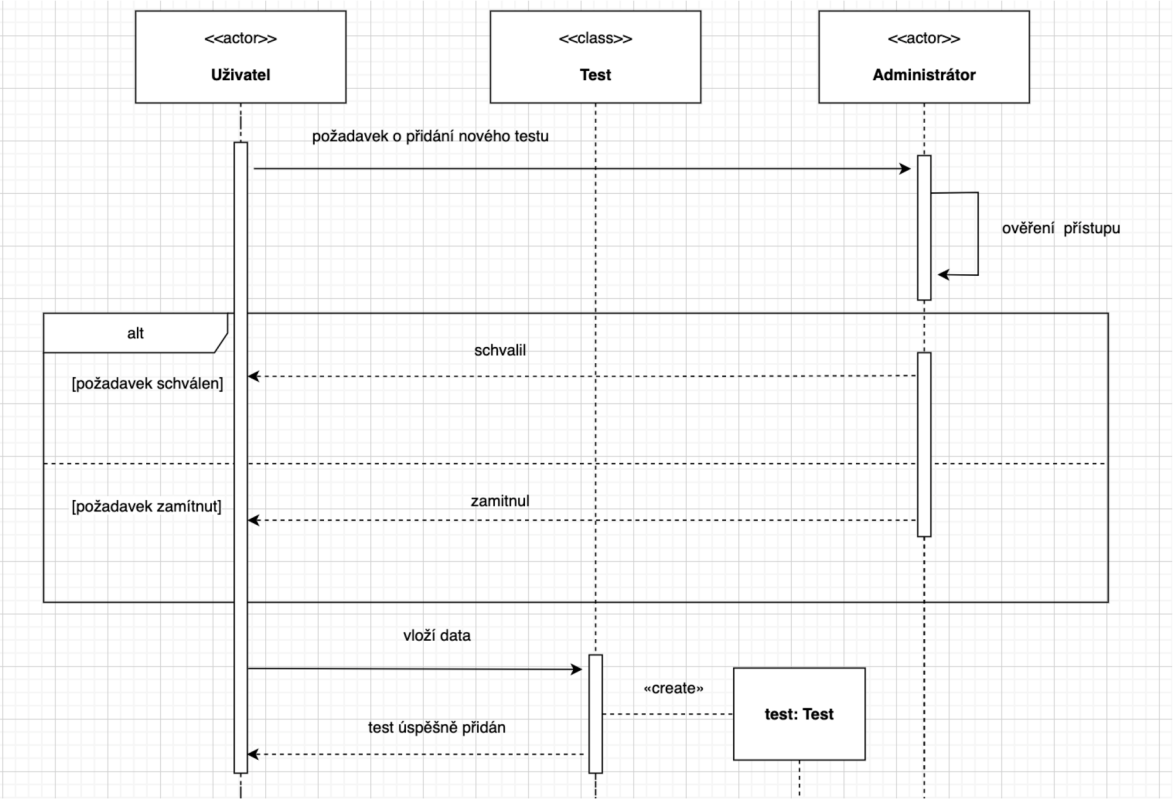
Sekvenční diagram

Sekvenční diagram je typ diagramu interakce, který ilustruje interakce mezi objekty nebo komponentami v systému v čase. Výměna zprav v čase v rámci množiny objektů. Vhodný pro popis posloupnosti chování z pohledu uživatele. Sekvenční diagramy pomáhají modelovat průběh událostí v systému a mohou se použít k identifikaci potenciálních problémů při návrhu systému.

Scénář případu užití *Přidat test* popisuje vložení nového genetického či covid testu do systému a jeho uložení.

Případ užití	Přidat test
ID	1
Stručný popis	Uživatel vkládá nový test do systému
Hlavní aktéři	Uživatel, Administrátor
Vstupní podmínky	Uživatel a Administrátor už se přihlásili
Hlavní scénář	1. Uživatel pošle požadavek o přidání nového testu 2. Systém ověří platnost požadavku a pak odešle zprávu Administrátoru ke schválení. 3. Administrátor obdrží žádost a zkontroluje přistup uživatele 4. KDYŽ Administrátor schválí požadavek 4.1.Uživatel má přistup a muže pokračovat dal 5. JINAK Administrátor zamítnul požadavek 5.1. Uživatel nemá přistup 6. Uživatel vloží data 7. Systém vytvořil test 8. Systém pošle potvrzovací zprávu
Výstupní podmínky	Test je vložen do systému
Alternativní scénáře	Chyba ověření přístupu uživatele Chyba v ověření zdrojů dat

Tabulka 1: Scénář případu užití – Přidat test



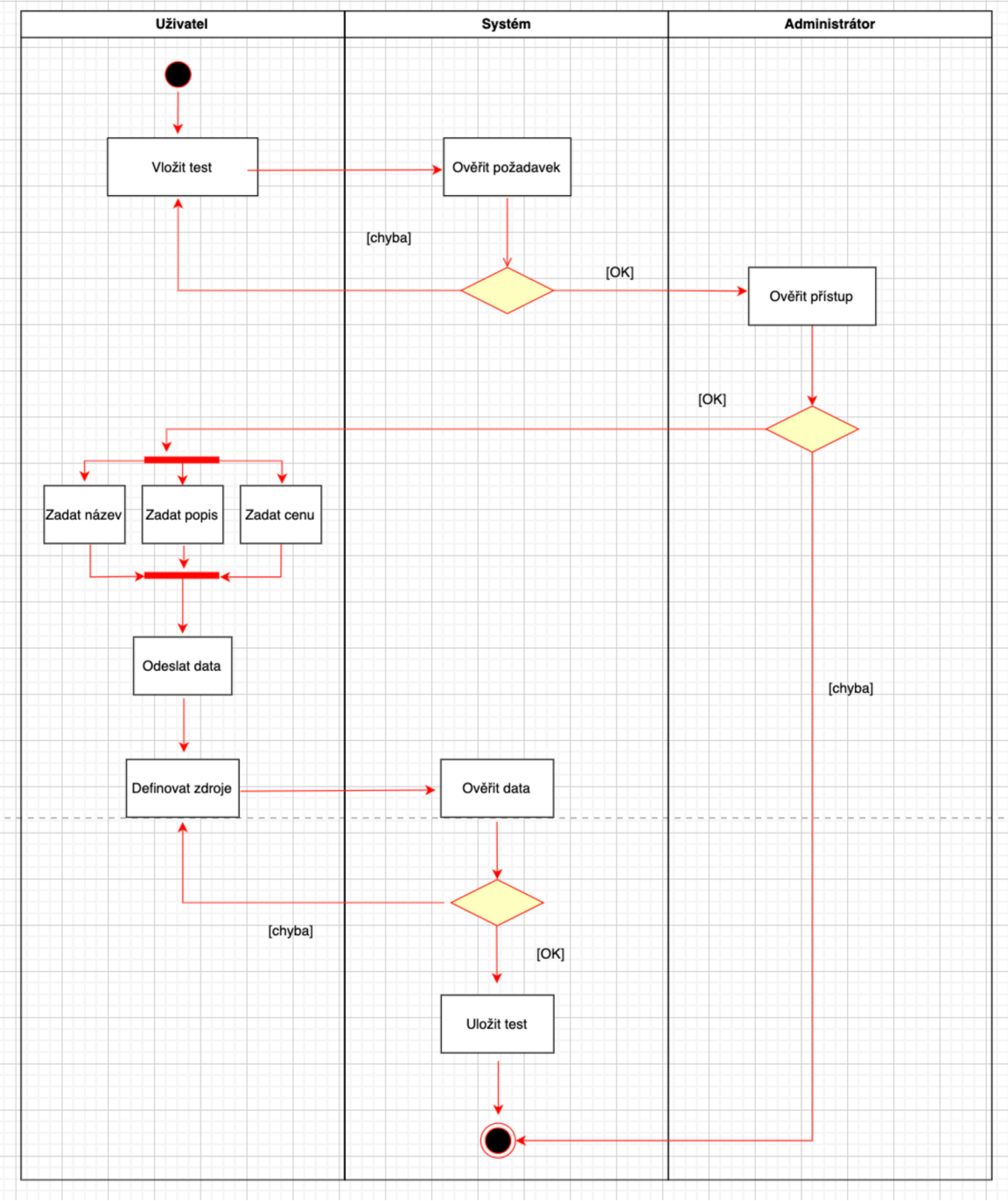
Obrázek 19: Sekvenční diagram – Přidat test

Tento sekvenční diagram popisuje proces přidání nového testu do systému. Hlavní aktéři jsou Uživatel a Administrátor a třída Test. Dochází k interakci se systémem, pokud je požadavek platný dalším krokem Administrátor zkontroluje oprávnění uživatele. Při schválení přístupu Uživatel zadá všechny potřebné informace pro přidání nového testu. Nový test je vytvořen se stereotypem «create» a na konci Uživatel obdrží zprávu o úspěšném přidání testu.

Aktivit diagram

Diagramy aktivit jsou diagramy používané v Unified Modeling Language (UML) k modelování toku činností a akcí v systému. Používají se k ilustraci toku kontroly v rámci procesu a k prokázání vzájemného vztahu činností. Diagramy aktivit se skládají z řady aktivit, reprezentovaných jako pole, které jsou spojeny šipkami pro zobrazení pořadí činností. Každé pole aktivity obsahuje popis aktivity a šipky představují tok kontroly z jedné aktivity na druhou. Diagramy aktivit mohou také zahrnovat plavecké dráhy, což jsou vodorovné čáry používané k rozdělení diagramu do různých částí na základě rolí nebo

odpovědností subjektů zapojených do činností. To usnadňuje pochopení toku kontroly a odpovědností v rámci systému. Kromě modelování toku činností mohou diagramy aktivit zahrnovat také rozhodovací body, kde tok kontroly závisí na výsledku rozhodnutí, a kontrolní struktury, jako jsou smyčky a větve, které umožňují modelování opakujících se nebo podmíněných činností.



Obrázek 20: Diagram aktivit – Vložení testu do systému

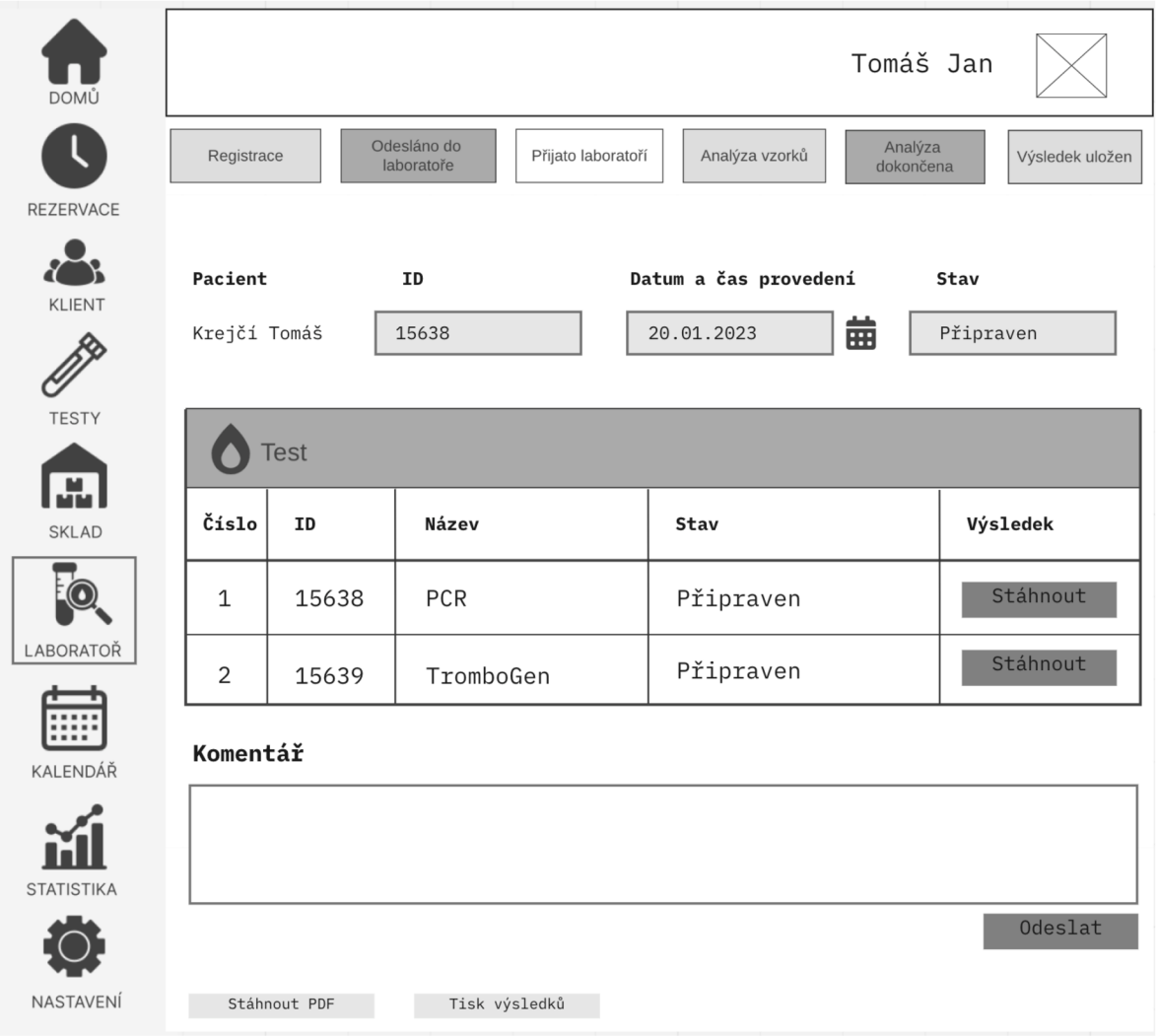
Tento obrázek znázorňuje diagram aktivit, na rozdíl od sekvenčního diagramu je zde zobrazen detailnější postup vložení nového testu

4.3.4 Wireframes

Wireframe je vizualizace návrhu uživatelského rozhraní pro webovou stránku nebo software aplikaci. Wireframy se používají ke komunikaci celkové funkcionality a struktury webové stránky nebo aplikace a poskytují základ pro další práci s designem a vývojem. Jsou klíčovým nástrojem v procesu designu a vývoje, protože umožňují designérům a vývojářům experimentovat s různými rozvrženími, navigacemi a dalšími aspekty uživatelského rozhraní.

Obrázek 21: Wireframe – Domovská stránka

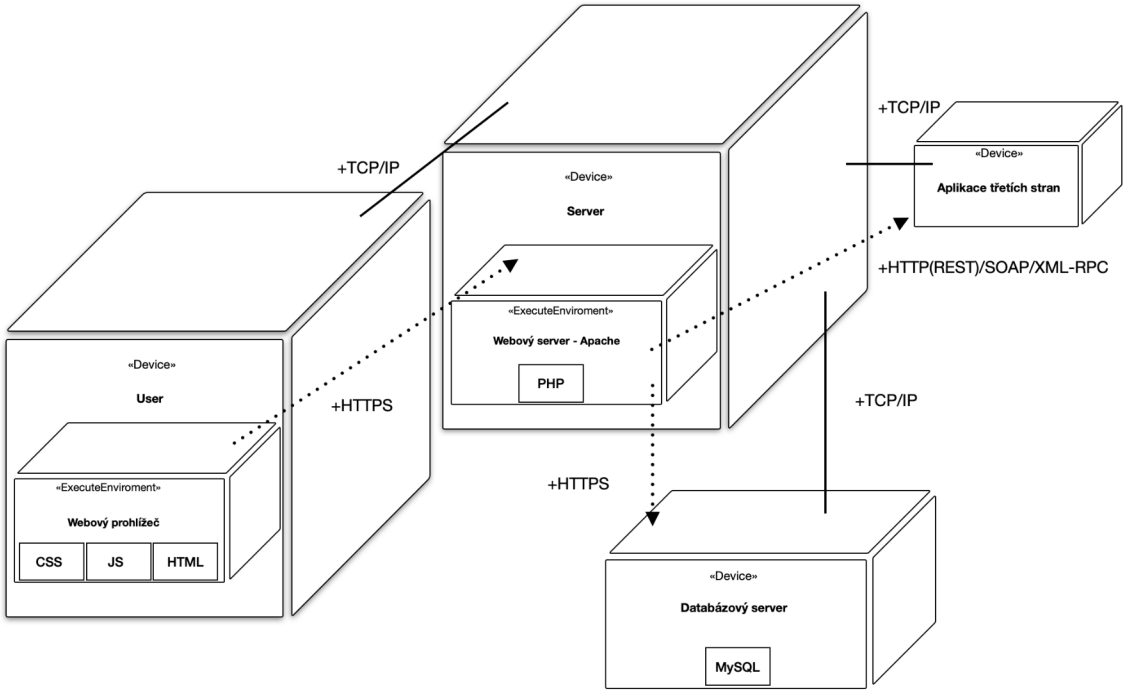
Na tomto obrázku je zobrazen wireframe domovské stránky, tak bude pravděpodobně vypadat rozhraní pro zaměstnance firmy. Stejně jsou zobrazeny hlavní sekce – Rezervace, Klient, Testy, Sklad, Laboratoř, Kalendář, Statistika, Nastavení. Na domácí stránce je zobrazen kalendář s aktuálním datem, počtem testů za dnešní den a aktuálními rezervacemi.



Obrázek 22: Wireframe – Správa laboratoře

Na tomto obrázku je zobrazen wireframe detail laboratoře. Na wireframu lze vidět, jak prostřednictvím systému kontaktovat laboratoř, najít člověka podle ID a zjistit, na jakých testech byl, stejně kdykoliv můžeme stáhnout nebo vytisknout výsledky. A můžeme zanechat komentář laboratoře o určité osobě.

4.3.5 Realizace



Obrázek 23: Nasazení IS

V diagramu je zobrazen třívrstvý design systému, který autor doporučuje pro integraci. Vrstva prezentace je zodpovědná za renderování grafického rozhraní pro uživatele v webovém prohlížeči, a využívá technologie HTML, CSS a JavaScript. K interakci mezi prezenční vrstvou a webovým serverem se používá protokol HTTPS. Pro implementaci systému autor navrhuje použít PHP a běhové prostředí webového serveru Apache. Databáze MySQL, která se nachází na jiném serveru, komunikuje s webovým serverem. Pro komunikaci s aplikacemi třetích stran jsou k dispozici webové služby, které používají rozhraní REST API, protokoly SOAP a XML-RPC.

5 Výsledky a diskuse

V této sekci budou prezentovány názory autora ohledně celkové kvality provedené práce a také budou uvedeny hlavní výzvy, se kterými se autor setkal. Tato část bude založena na subjektivním posouzení a hodnocení autora.

5.1 Zhodnocení návrhu

Hodnocení kvality navrženého systému nelze provést standardními metodami, jako jsou normy kvality ISO a jiné hodnotící metody, protože nebylo možné zohlednit všechny aspekty systému, jako je výkon, latence atd. Navržený systém je součástí diplomové práce, avšak nebyl implementován. Proto nebude provedeno hodnocení standardizovanou metodikou. Návrh systému byl založen na funkčních a nefunkčních požadavcích a byl vytvořen tak, aby každý dílčí funkční požadavek byl realizován případem užití, čímž je systém plně funkční. Nefunkční požadavky jsou obtížnější k ověření, protože většinu z nich lze posoudit až po implementaci systému. Použitelnost systému může být částečně posuzována na základě navržených wireframes, které přehledně zobrazují menu systému a umožňují si představit finální verzi systému.

V rámci návrhu systému byly vytvořeny tři modely: model třídy, stavový model a model interakcí. Tyto UML diagramy byly vybrány tak, aby reprezentovaly nejdůležitější součásti systému. Diagramy jsou zjednodušené, což umožňuje konečný návrh systému být zcela univerzální a ponechat prostor pro detailnější rozpracování a volnost pro implementace. Konečný návrh tak může sloužit jako pevný základ pro vývoj systémů podobného typu. Autor považuje těsné spojení mezi laboratoří a odběrovým místem za největší přínos navrhovaného systému.

5.2 Diskuse

V rámci praktické části byl vyřešen problém automatizace chodu firmy, zvýšení příjmů prostřednictvím zkrácení času na zpracování dat, nahrávání testů, komunikaci s laboratoří a skladem. No a samozřejmě hlavní výhodou je – tento systém je ideálním řešením pro naši společnost, která se zabývá genetickým testováním, testováním na Covid a očkováním, protože již existující informační systémy jsou zaměřeny na trochu jiné funkce

Hlavním problémem pro autora bylo, že málo informací o tom, jak pracuje naši laboratoř, takže při vývoji modelů pro laboratoř jsem vycházel z dat, která jsem získal při práci ve firmě a rozhovory s pracovníky laboratoře. Problémy, se kterými jsem se setkal při psaní praktické části se týkaly především tvorby UML diagramů. Jedním z největších problémů – je sběr dat pro analýzu a návrh systému. Dalším problémem bylo přizpůsobení UML diagramů konkrétním požadavkům projektu a jednoduchost tříd v hlavním Use case (případu užití), což mi nedovolilo provést dekompozici případu užití. Autor snažil se však překonat tyto problémy pomocí důkladného výzkumu a konzultací s potenciálními uživateli.

6 Závěr

Výsledkem této diplomové práce je analýza a návrh informačního systému v jazyce UML pro společnost, která poskytuje genetické testování, covid testování a očkování. Cílem této práce bylo vytvořit systém, který bude efektivně podporovat proces zprostředkování služeb v oblasti genetického testování.

Teoretická část diplomové práce se věnovala studiu odborných informačních zdrojů, z nichž byly získány relevantní teoretické východiska pro tvorbu informačního systému. Tato část pojednávala o různých typech informačních systémů, životním cyklu informačních systémů, analýze uživatelských požadavků a přístupech a metodikách k tvorbě informačních systémů. Důraz byl kladen na popis standardů jazyka UML, které byly využity při návrhu informačního systému. Všechny získané poznatky z teoretické části byly důkladně aplikovány v praktické části diplomové práce a pomohly splnit hlavní cíle práce.

V rámci praktické části projektu byla provedena analýza a návrh informačního systému, částečně založená na autorově odborných zkušenostech. Nejprve byla poskytnuta stručná charakteristika navrhovaného systému s důrazem na jeho hlavní cíl – efektivní podporu obchodních aktivit a snížení administrativní náročnosti.

Hlavním přínosem tohoto systému je úplná automatizace práce laboratoře a odběrových míst. Pracovník se vždy může podívat, kolik testů je odesláno, v jaké fázi je probíhá analýza, jak dlouho ještě čekat na výsledky a nejvhodnější je, že při neprůkaznosti vzorků se automaticky odešle SMS zákazníkovi a zároveň pro naprostou jistotu pracovník laboratoře zavolá zákazníkovi.

V analytické části byla provedena analýza požadavků, která byla jedním z dílčích cílů diplomové práce. V této části práce byly požadavky stanoveny pomocí potenciálních uživatelů systému. Požadavky byly rovněž rozděleny na funkční a nefunkční. S využitím všech těchto dat byl vytvořen systém pomocí modelovacího jazyka UML.

V průběhu návrhu systému byly vytvořeny a nakresleny základní modely pomocí modelovacího jazyka UML – model tříd, stavový model a model interakcí. Při navrhování systému byl prvním modelem diagram tříd, který popisoval hlavní třídy a jejich vztahy, byl také napsán datový slovník, který podrobně popsal funkčnost každé třídy a její základní funkce. Dalším modelem byl stavový model tento model je zachycen stavovými diagramy a ukazuje nám chování tříd v systému. Posledním byl model interakcí, který ukázal nám spolupráce objektů v systému.

Na závěr praktické části diplomové práce byly vytvořeny modely wireframes, které slouží k lepšímu pochopení přibližného vzhledu informačního systému. Podle autora by měl být tento systém implementován jako webová aplikace s třívrstvou architekturou, což je zobrazeno v diagramu nasazení. Přestože by bylo možné přejít přímo k fázi implementace, pokud by byly jednotlivé části modelu podrobněji rozpracovány a doplněny algoritmy základních funkcí, autor považuje hlavní cíl práce za splněný. Navržený model systému tedy poskytuje kvalitní základ pro další detailní zpracování návrhu a následnou implementaci.

7 Seznam použitých zdrojů

BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. Praha, 2012. Management v informační společnosti. ISBN 978-80-247-4153-6.

BUCHALCEVOVÁ, Alena a Iva STANOVSKÁ. Příklady modelů analýzy a návrhu aplikace v UML. Praha, 2013. ISBN 978-80-245-1922-7.

ARLOW, Jim a Ila NEUSTADT. UML 2 a unifikovaný proces vývoje aplikací. Praha, 2007. ISBN: 978-80-251-1503-9.

LARMAN, Craig. Agile And Iterative Development: A Manager's Guide. Boston, 2004. ISBN: 978-01-311-1155-4

PAVLÍČKOVÁ, Petra. Řízení IT projektů [přednáška]. Praha: ČZU, 2018.

DANEL, Roman. Informační systémy [přednáška]. Ostrava: VŠB-TUO, 2011.

DANEL, Roman. Analýza a projektování systémů [přednáška]. Ostrava: VŠB-TUO, 2011.

KOĎOUSKOVÁ, Barbora. Rascasone.com: Podnikové informační systémy (EIS) a jejich funkce [online]. [cit. 2021-04-13]. Dostupné z: https://www.rascasone.com/cs/blog/podnikove-informacni-systemi-eis

Iquest.cz: Metody vývoje aplikací. Waterfall, V-model, Inkrementální model [online]. [cit. 2017-06-16]. Dostupné z: https://blog.iquest.cz/2017/07/metody-vyvoje-aplikaci-waterfall-v.html

KRÁL, Jaroslav. Informační systémy: Specifikace, realizace, provoz. Veletiny: Science, 1998. ISBN: 80-86083-00-4

MARTINŮ, Jiří. Metodiky vývoje software [přednáška]. Olomouc: MVŠO, 2018.

BUCHALCEVOVÁ, Alena. Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky. Praha: Grada, 2005. Management v informační společnosti. ISBN 80-247-1075-7.

ŠOCHOVÁ, Z. A E. KUNCE. Agilní metody řízení projektů. 1. vyd. Brno: Computer Press, 2014. 176 s. ISBN 978-80-251-4194-6.

LARMAN, C. Agile and Iterative Development: A Manager's Guide. Boston: Addison-Wesley, 2004. 342 s. ISBN 978-01-311-1155-8.

KADLEC, V. Agilní programování: metodiky efektivního vývoje softwaru. 1. vyd. Brno: Computer Press, 2004. 278 s. ISBN 978-80-251-0342-0.

PEKAŘ, Lukáš. Bonsai-development.cz: Párové programování [online]. [cit. 2018-05-13]. Dostupné z: https://bonsai-development.cz/clanek/parove-programovani

CHLAPEK, Dušan. Vývoj informačních systémů (pracovní sešit ke cvičením) / Dušan Chlapek, Václav Řepa, Iva Stanovská. 1. vyd. Praha : Oeconomica, 2005. 161 s. brož. ISBN:80-245-0977-6

RÁČEK, Jaroslav. Strukturovaná analýza systémů. Brno: Masarykova univerzita, 2006. ISBN
80-210-4190-0.

ŘEPA, Václav. Analýza a návrh informačních systémů. Praha: Ekopress, 1999. ISBN 80-86119-13-0. S. 403.

BASL, Josef a Roman BLAŽÍČEK. Podnikové informační systémy: podnik v informační společnosti. 3., aktualiz. a dopl. vyd. Praha, 2012. Management v informační společnosti. ISBN: 978-80-247-4307-3.

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam použitých zkratek

IS – informační systém

ICT – Informační a komunikační technologie

UML – Unified Modeling Language

SW – sowtware

BI – Business Intelligence

TPS – Transaction Processing System

MIS – Management Information Systems

EIS – Executive Information System

OLAP – Online Analytical Processing

CIM – Computer Integrated Manufacturing

ERP – Enterprise Resource Planning

HRM – Human Resources Management

CRM – Customer Relationship Management

SCM – Supply Chain Management

RUP – Rational Unified Process

OPEN – Object-oriented Process, Environment and Notation

COMMA – Common Object Methodology Metamodel Architecture

UP – Unified Process

SEP – Software Engineering Process

USDP – Unified Software Development Process

XP – Extreme Programming

JAD – Joint Application Design

OOP – objektově orientovaný přístup

ERD – Entity Relationship Diagram

DFD – Data Flow Diagram

STD – State Transition Diagram

CASE – computer-aided software engineering

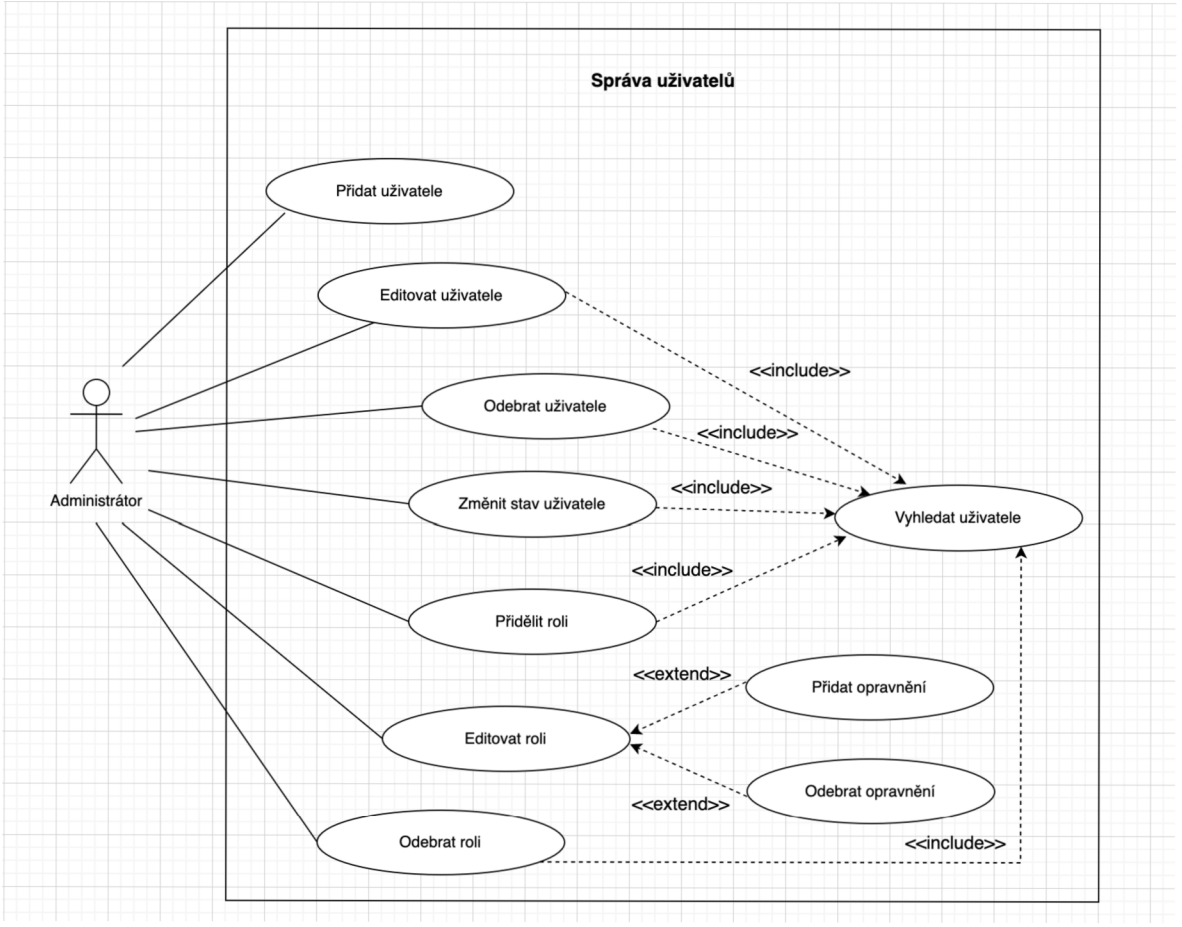
OMT – Object Modeling Technique

OMG – Object Management Group

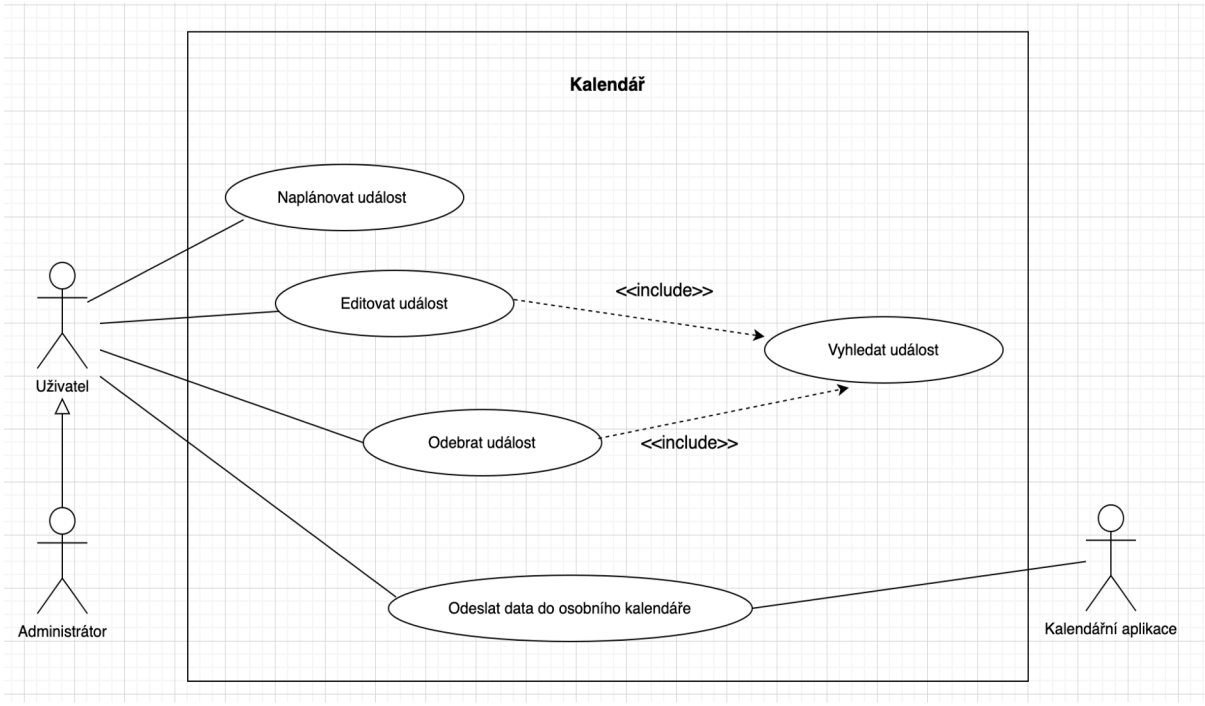
RFP – Request For Proposal

CORBA – Common Object Request Broker Architecture

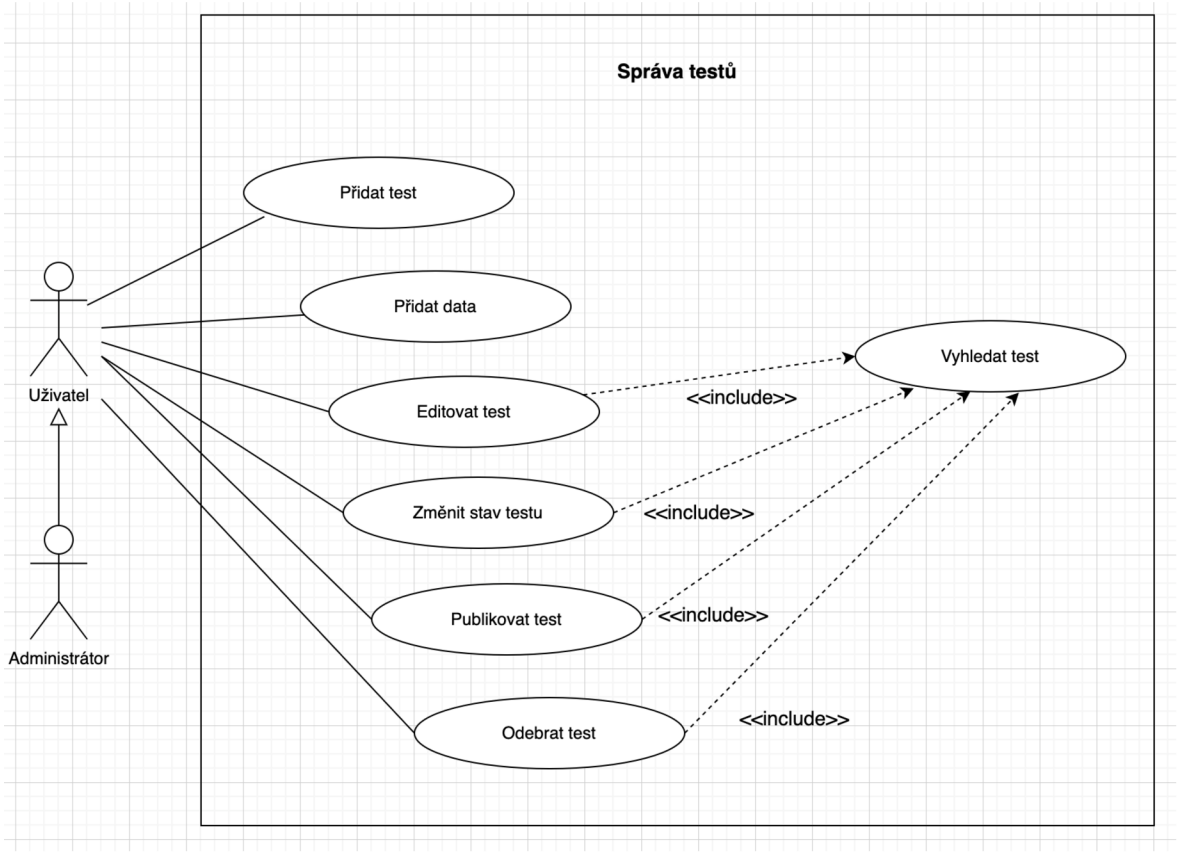
Přílohy



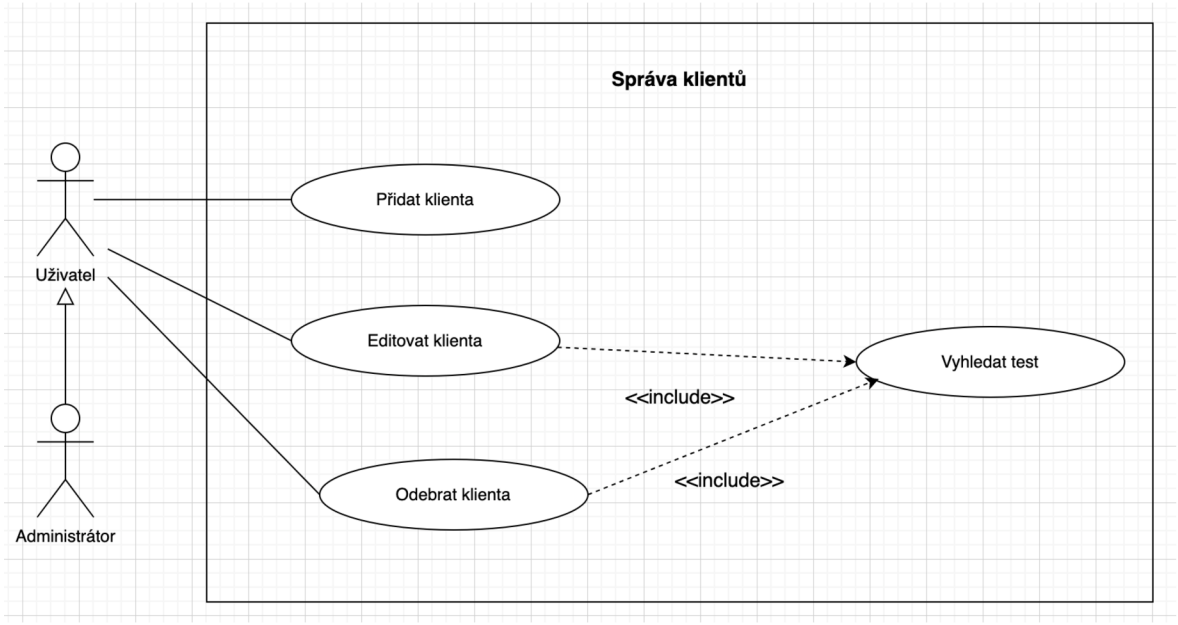
Use Case (případy užití) Správa uživatelů



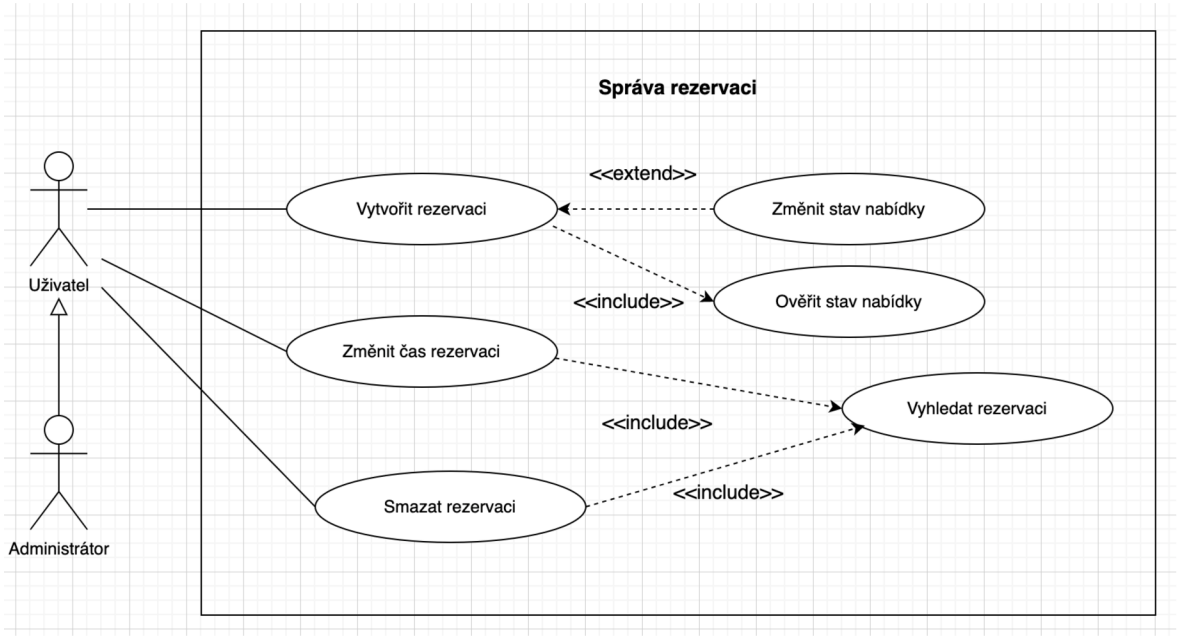
Use Case (případy užití) Kalendář



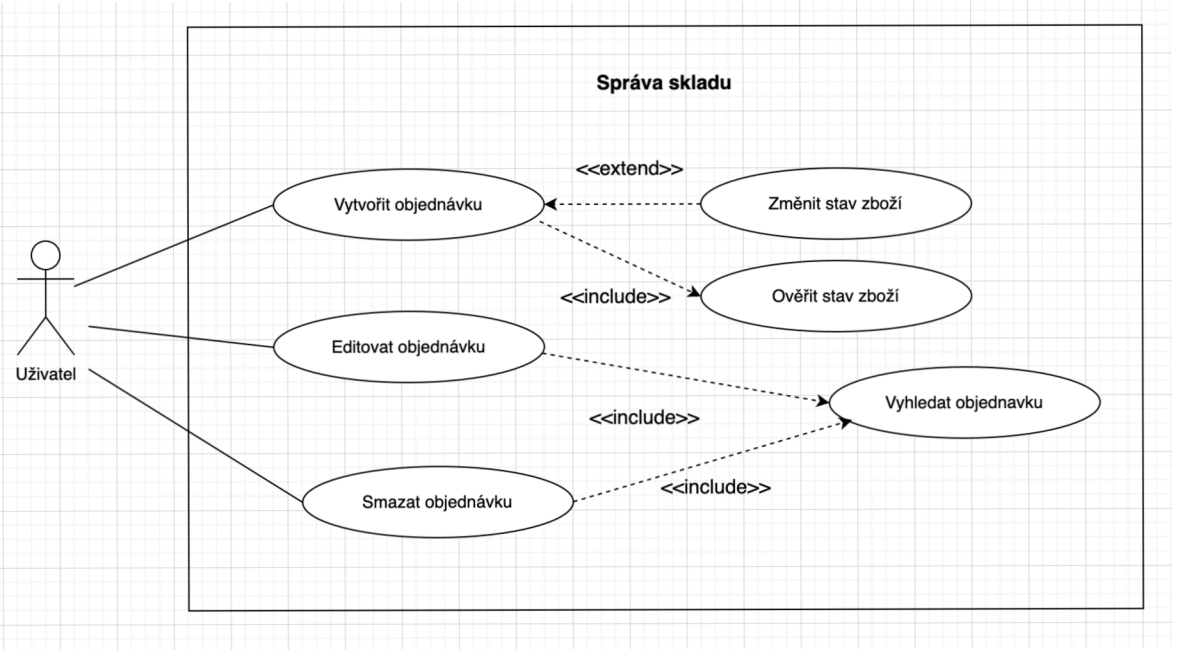
Use Case (případy užití) Správa testů



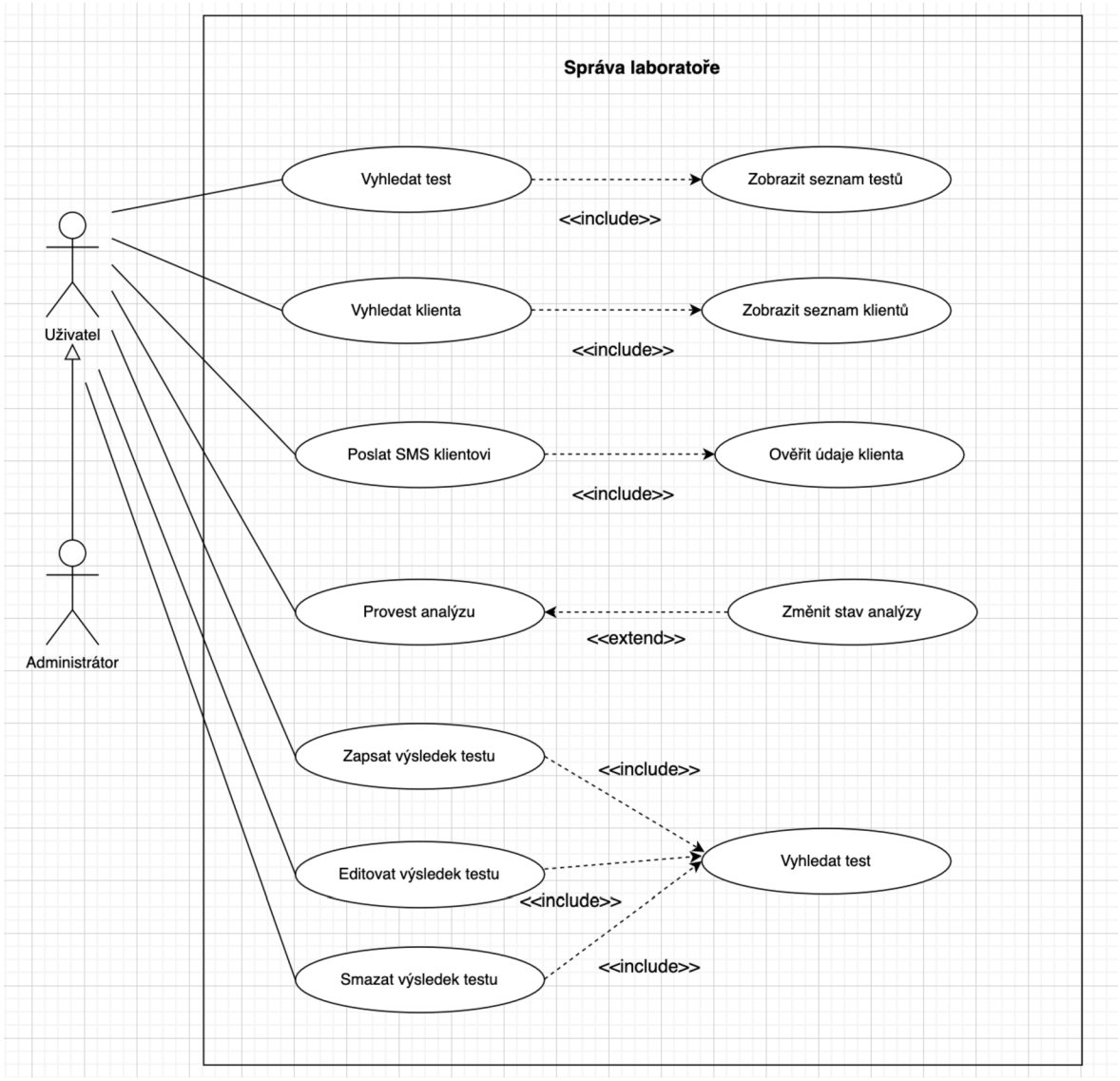
Use Case (případy užití) Správa klientů



Use Case (případy užití) Správa rezervaci



Use Case (případy užití) Správa skladu



Use Case (případy užití) Správa laboratoře