DIPLOMOVÁ PRÁCE

Analýza a implementace informačního systému na MÚVS ČVUT

Analysis and information system implementation at MIAS CTU

STUDIJNÍ PROGRAM

Řízení rozvojových projektů

STUDIJNÍ OBOR

Projektové řízení inovací v podniku

VEDOUCÍ PRÁCE

doc. Ing. Lenka ŠVECOVÁ, Ph.D.

TYROL

MARTIN

2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Tyrol Jméno: Martin Osobní číslo: 397764

Fakulta/ústav: Masarykův ústav vyšších studií (MÚVS)

Zadávající katedra/ústav: Oddělení manažerských studií

Studijní program: (N3949) Řízení rozvojových projektů

Studijní obor: (6208T183) Projektové řízení inovací v podniku

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Analýza a implementace informačního systému na MÚVS ČVUT

Název diplomové práce anglicky:

Analysis and information system implementation at MIAS CTU

Pokyny pro vypracování:

CÍL: Cílem DP je zmapování procesů pro hodnocení mezd zaměstnanců a vytvoření informačního systému pro podporu těchto procesů.
PŘÍNOS: Přínosem práce je zefektivnění procesů prostřednictvím informačního systému.
OSNOVA: 1. Business analýza; 2. Modelování procesů; 3. Hodnocení výkonnosti zaměstnanců; 4. Zabezpečení informačních systémů; 5. Návrh a implementace informačního systému; 6. Testování; 7. Závěr

Seznam doporučené literatury:

1) UML 2 a unifikovaný proces vývoje aplikací: objektově orientova-ná analýza a návrh prakticky, ARLOW Jim, 2007
2) Tvorba informačních systémů: principy, metodiky, architektury, BRUCKNER Tomáš, 2012
3) Play for Java: Covers Play 2, LEROUX Nicolas, Sieste de KAPER, 2014
4) Řízení lidských zdrojů: nejnovější trendy a postupy, ARMSTRONG, Michael Stuart, 2007

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Lenka ŠVECOVÁ, Ph.D., MÚVS ČVUT v Praze, oddělení manažerských studií

Jméno a pracoviště konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: 5. 1. 2017 Termín odevzdání diplomové práce: 5. 5. 2017

Platnost zadání diplomové práce: 31. 8. 2018

Podpis vedoucí(ho) práce | Podpis vedoucí(ho) ústavu/katedry | Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

19. 4. 2017
Datum převzetí zadání

Podpis studenta(ky)

TYROL, Martin. *Analýza a implementace informačního systému na MÚVS ČVUT*. Praha: ČVUT 2017. Diplomová práce. České vysoké učení technické v Praze, Masarykův ústav vyšších studií.

MASARYKŮV ÚSTAV
VYŠŠÍCH STUDIÍ
ČVUT V PRAZE

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně. Dále prohlašuji, že jsem všechny použité zdroje správně a úplně citoval a uvádím je v přiloženém seznamu použité literatury.
Nemám závažný důvod proti zpřístupňování této závěrečné práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Praze dne: 19. 05. 2017 Podpis:

Poděkování

Rád bych poděkoval doc. Ing. Lence Švecové, Ph.D. za vedení diplomové práce, za věcné připomínky, cenné rady, poskytnuté informace a vstřícnost při konzultacích. Dále děkuji rodičům za podporu, zázemí a důvěru při studiu.

Abstrakt

Tato diplomová práce si klade za cíl analyzovat a implementovat informační systém pro vnitřní potřeby Masarykova ústavu vyšších studií na ČVUT v Praze. Popisuje oblasti štíhlého podniku a podrobněji se zaměřuje na štíhlou administrativu. Pro business analýzu využívá práce UML diagramů, které popisují, co bude systém umět. Konkrétně analyzuje oblasti potřebné pro hodnocení zaměstnanců, které obsahují jejich aktivity ve sledovaném období. Práce popisuje metody pro řízení vývoje a implementaci software, a také metody pro jeho testování. Na základě analyzovaných požadavků je vytvořen návrh nového systému, na který navazuje implementace. Výsledný inovační software je vytvořený v Play Frameworku jako webová aplikace, ke které může přistupovat každý zaměstnanec. Na základě porovnání časové úspory před a po implementaci, je viditelná úspora ve prospěch implementace.

Klíčová slova

UML, analýza, software, framework, štíhlá administrativa

Abstract

This diploma thesis aims at analyzing and implementing the information system for internal needs of the Masaryk Institute of Advanced Studies at CTU in Prague. It describes the areas of a lean organisation, and focuses in more detail on lean administration. For business analysis, thesis use UML diagrams that describe what the system will be able to do. In particular, it analyzes the areas needed for the evaluation of the employees, which contain their activities in the period under review. This thesis describes methods for software development and implementation management, as well as methods for its testing. On the basis of the analyzed requirements, a new system design is developed, followed by the implementation. The resulting innovative software is created in Play Framework as a web application that can be accessed by every employee. Based on the comparison of time savings before and after implementation, there is visible savings in favor of implementation.

Key words

UML, analysis, software, framework, lean administrative

Obsah

Úvod

Pro udržování aktuálních a relevantních dat v dnešní době slouží informační systémy. A je jedno, jestli se jedná o výrobní podnik nebo administrativu. Informační systémy poskytují vzájemně propojené informace a procesy, které pracují s těmito výstupy. Tyto informace je tak možné zpracovávat mnohem rychleji a efektivněji. Informační systém umožňuje zlepšit služby zákazníkům, kterými jsou v tomto případě zaměstnanci Masarykova ústavu vyšších studií ČVUT v Praze. Ti vykazují činnost, ať už v podobě výuky, vedení kvalifikačních prací, práce na projektech nebo vědecko-výzkumnou činnost. Pro vedení tohoto ústavu je nesmírně důležité udržovat informace pohromadě a aktuální a na základě nich hodnotit výkon pracovníků, s cílem zachovat spravedlnost v systému odměňování.

Cíle práce

Cílem práce je analyzovat procesy vedoucí k stanovení osobního ohodnocení a následně navržení informačního systému, který bude realizován pomocí webové aplikace, založené na platformě Play Framework. Tento framework je založený na jazyku Java a Scala a jeho výhodou je snadné použití a vhodnost pro informační systémy.
Přínosem této práce bude analyzovaný a navržený informační systém pro správu aktivit zaměstnanců na Masarykově ústavu vyšších studií na ČVUT v Praze. Inovací systému bude nejen odstranění plýtvání nákladů, času a nadbytečné práce, ale i centrální místo pro správu dat.
Diplomová práce je rozdělena na pět části. První tři jsou teoretické, obsahují definice a důležité informace ke štíhlé administrativě, business analýze a vývoji software. Praktická část se zabývá samotnou analýzou a implementací. Zde byl proveden sběr a analýza požadavků. Na ně navazuje návrh budoucího stavu a tvorba návrhů obrazovek (wireframů). Poslední částí je zhodnocení přínosů pro společnost ve formě úspory času.

TEORETICKÁ ČÁST

1 Štíhlá organizace

Podstatou štíhlé organizace, označované také jako *Lean Organization*, je zvyšování výkonnosti podniku, který se zaměřuje na zvýšení své konkurenceschopnosti, pomocí co nejlepšího využití zaměstnanců, pracovních prostor a dostupných prostředků, bez zbytečného plýtvání. Jak uvádí Košturiak ve své knize (Košturiak, a další, 2006 str. 17), tyto činnosti jsou potřebné dělat správně, hned napoprvé a rychleji a lépe, než ostatní. Tato myšlenka vznikla ve výrobní sféře. Štíhlý podnik reprezentuje obrázek 1.

Obrázek 1 Štíhlý podnik *Zdroj: upraveno dle (Košturiak, a další, 2006 str. 20)*

Co nám přinese štíhlý podnik? Není to samoúčelné redukování nákladů. Z pohledu pracovníků je přínosem lépe organizovaná práce, pracovní prostředí a také prémie. Z pohledu majitelů a akcionářů je přínosem budování konkurenční výhody a růst společnosti. Z pohledu zákazníka je to přidaná hodnota, flexibilita, vysoká kvalita a nízká cena.

Zeštíhlení nám umožní snížit skladové zásoby na minimum, snížit prostoje ve výrobě, zlepšit organizaci práce, snížit zmetkovitost výrobků a zvýšit kvalitu.

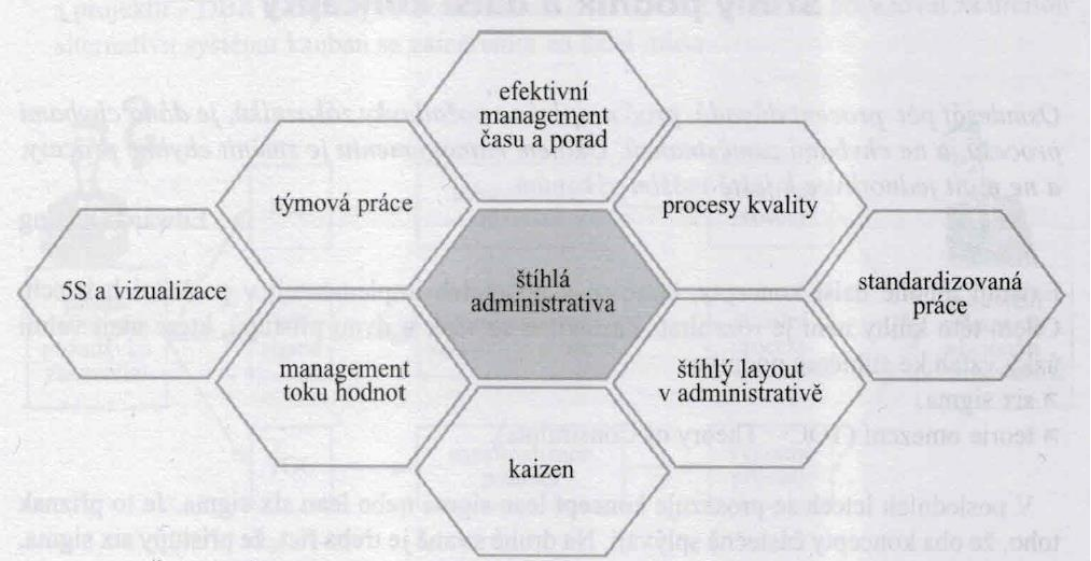
Mezi prvky štíhlé organizace patří štíhlá výroba, štíhlá logistika, štíhlý vývoj a štíhlá administrativa. Pro potřeby této práce se budu dále zabývat pouze štíhlou administrativou.

1.1 Štíhlá administrativa

Základem jsou štíhlé procesy v administrativě. Zefektivnění procesů pomůže zpřehlednit fungování a v ideálním případě i ušetřit náklady. Prvky štíhlé výroby nelze jednoduše převzít a aplikovat v administrativě. Štíhlost procesů vychází z konceptu Six

Sigma a teorie omezení (*Theory of Constraint*). Je potřeba nástroje Six Sigma, Lean managementu a další techniky upravit požadavkům služeb (Dohnal, 2016). Popis těchto pojmů popisuje v rámci této první kapitoly níže.

Obrázek 2 Štíhlá administrativa *Zdroj: (Košturiak, a další, 2006 str. 35)*



Štíhlá administrativa (Obrázek 2) začíná zavedením systému standardů a pořádkem. V administrativě se produktem rozumí informace. Informace není vidět, není měřitelná, těžko se definuje a má různou hodnotu pro každého účastníka. Pro její zviditelnění se používají formuláře, e-maily, atd.

V administrativě je produktem nejen informace, ale i její vznik, přenos a zpracování, které ovlivňují procesy. Ty jsou měřitelné a sestávají z řady činností.

1.1.1 Druhy plýtvání – muda

Plýtvání, označované z japonského slova *muda*, je nejen ve výrobě, ale i v administrativě. Označuje všechny druhy plýtvání a ztráty, které zvyšují náklady výrobku nebo služby bez zvýšení jejich hodnoty. Jak uvádí Košturiak ve své knize (Košturiak, a další, 2006 str. 34), hlavní formy plýtvání v administrativě jsou:

Nadprodukce – V administrativě se nadprodukce může týkat sběru informací, které nikdo nevyužije, nepotřebná hlášení, zbytečný tisk velkého množství papírů.

Nadbytečná práce – Nadbytečné rozesílání dokumentů, informací a jejich zpracování. Nejasná zadání, kdy je potřebné se doptávat telefonicky nebo elektronicky. Zbytečná evidence dokumentů a hlášení.

Zbytečný pohyb – Neustálý pohyb mezi počítačem, tiskárnou, apod.

Zásoby – E-maily čekající na vyřízení, přeplněné přihrádky na dokumenty způsobují prodlužování doby procesu.

Čekání – Neustálé čekání na informace, dokumenty a apod. Čekání je plýtvání, protože nepřidává žádnou hodnotu a dochází tím k oddalování schvalování.

Opravování – Časté chyby, nepochopení a omyly v dokumentech zdržují procesy, protože je potřeba je následně opravit.

Doprava – Velké množství schvalovacích míst, kterými musí dokumenty projít, samotné přecházení mezi vzdálenými budovami, apod.

Nevyužité schopnosti pracovníků, jejich myšlenek – Musí docházet k zapojení všech zaměstnanců na všech úrovních organizace. Je nutné je zapojit k přinášení myšlenek na zlepšení a zapojovat je do jejich zavádění.

1.1.2 Plýtvání v softwarovém vývoji

I při vývoji software dochází k plýtvání, mimo výše uvedeného, například:

- Plýtváním jsou nejasné požadavky a zadání. Dochází k mnoha nedorozuměním.
- Zbytečný kód, kdy se používají zbytečné složité knihovny a příliš obecná architektura. Při návrhu databáze je vhodné schéma vyvíjet postupně, jak je potřeba.
- Nízká kvalita a nedostatečné testování.

1.2 Zavádění štíhlé administrativy

Zavádění štíhlé administrativy má dvě fáze. První je analýza procesů v organizaci. Následuje optimalizace procesů za účelem odstranění plýtvání. Pro zavedení je nutné získat podporu a zapojení zaměstnanců, dokáží identifikovat místa plýtvání, o kterých nemá vedení organizace představu.

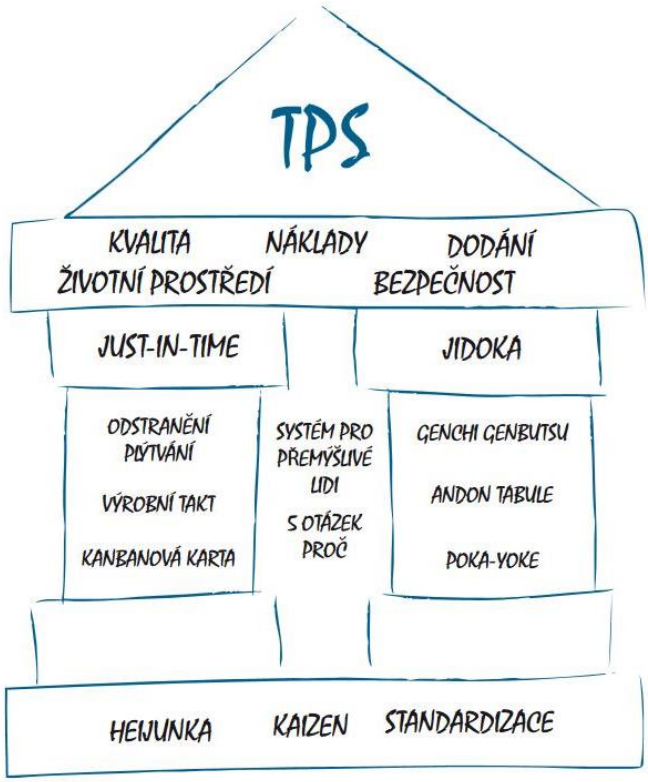
1.3 Přínos štíhlé administrativy

Zavedení štíhlé administrativy umožní zlepšení schopnosti plánovat, zvýší se kvalita výkonů, transparence a tím se zvýší motivace zaměstnanců. Zároveň dojde ke zlepšení spokojenosti zákazníků. Nelze opomenout ani úsporu nákladů.

1.4 TPS

Toyota Production System (TPS) znamená soubor vzájemně provázaných principů a metod, které směřují k neustálému zdokonalování procesů a eliminaci nežádoucího plýtvání zdroji. TPS je filozofií komplexního řízení výroby. Hlavními zakladateli TPS byli Taiichi Ohno, Shigeo Shingo a Eiji Toyoda. Systém byl vyvinut po druhé světové válce v automobilovém průmyslu. Toyota nabízela různé modely a jejich konfigurace. Chtěla uspokojit zákazníky, proto začala zavádět pružnou výrobu tak, aby uměla reagovat na požadavky zákazníků, lépe využívala skladovací prostory a výrobní zařízení. Zároveň získala větší produktivitu (Toyota Material Handling CZ s.r.o., 2010 stránky 5-7).

Obrázek 3 Výrobní systém Toyota TPS *Zdroj: (Toyota Material Handling CZ s.r.o., 2010 str. 5)*



Obrázek 3 zobrazuje podstatu TPS. Podoba domu není zcela náhodná. Podobně jako dům, i firma je pevná a stabilní pouze tehdy, když má pevné základy, nosné pilíře a střechu. Tu tvoří co nejnižší náklady, rychlost dodání a kvalita výsledného produktu nebo služby. Pilíře tvoří *Just-in-time* a *Jidoka*, které popisuji dále. Středem celého systému jsou lidé, kteří naplňují jednotlivé části systému. Základy celého TPS jsou standardizace, neustálé zlepšování a systém *heijunka* pro vyrovnávání výkyvů. Princip je popsán níže.

1.5 Just-in-time

Hlavním principem výroby *Just-in-time* je princip tahu, který je řízen reálnou potřebou zákazníka. Minimalizujeme tak výrobu „na sklad". Systém tahu využívá metody *Kanban* (blíže popisuje část 1.5.1), která řídí proces, aby nevznikala úzká místa. Pokud je však najdeme, je vhodné aplikovat *Kaizen* (viz 1.9) pro zlepšení (Keřkovský, a další, 2012 stránky 71-72).

1.5.1 Kanban

Výrobu obecně můžeme rozdělit na tlakový a tahový princip. Princip tlaku využívají podniky, které vyrábějí podle stanoveného množství. Odbyt jim automaticky zajišťuje nedostatek zboží na trhu. Nemusejí se tak příliš starat o skutečné potřeby zákazníka, zboží mu dotlačí (Šimon, a další).

Na druhé straně existuje princip tahu, který se zajímá o potřeby zákazníka. Nabízí mu variabilitu daného výrobku. Zákazník si může volit barvu, příslušenství a podobně. To samozřejmě přináší tlak na výrobce a jeho výrobu, aby dodal zákazníkovi výsledný produkt v požadované kvalitě v co nejkratším čase.

Slovo „Kanban" pochází z japonských slov *kan* (karta) a *ban* (signál). Metoda Kanban se původně používala pro řízení počtu lidí v chrámu. Kdo šel dovnitř, dostal lístek a při odchodu jej zase odevzdal. Tím se zamezilo přetlaku lidí v chrámu, protože počet lístků byl omezený a bez něj dovnitř nikdo nesměl (Šochová, a další, 2014 str. 105).

Princip metody zavedla společnost Toyota pro účinné řízení toku ve výrobě. Systém proto využívá principu tahu, kde se dodávka dostává na místo až je potřeba, tedy „just in time". Označuje se tak systém pro plánování výroby za účelem rychlé reakce pro uspokojení poptávky zákazníků. Autorem je Taiichi Ohno (Toyota Material Handling CZ s.r.o., 2010 str. 7).

Na každou etapu výroby může být pouze tolik úloh, kolik je povolených karet. Tím se redukuje rozpracovaná výroba. Po implementaci Kanbanu lze určit, jak dlouho se který proces zpracovává a jak dlouho bude muset zákazník čekat. Každá Kanban karta obsahuje informaci o množství a datu, kdy má být daná věc vyrobena.

Pravidla Kanbanu jsou podle (Vochozka, a další, 2012 str. 432) následující:

- Dokud nepřijde Kanban karta, pracoviště zůstává nečinné.
- Každý přepravní kontejner má svou Kanban kartu.
- Kontejnery pro stejnou činnosti, mají stejnou velikost.

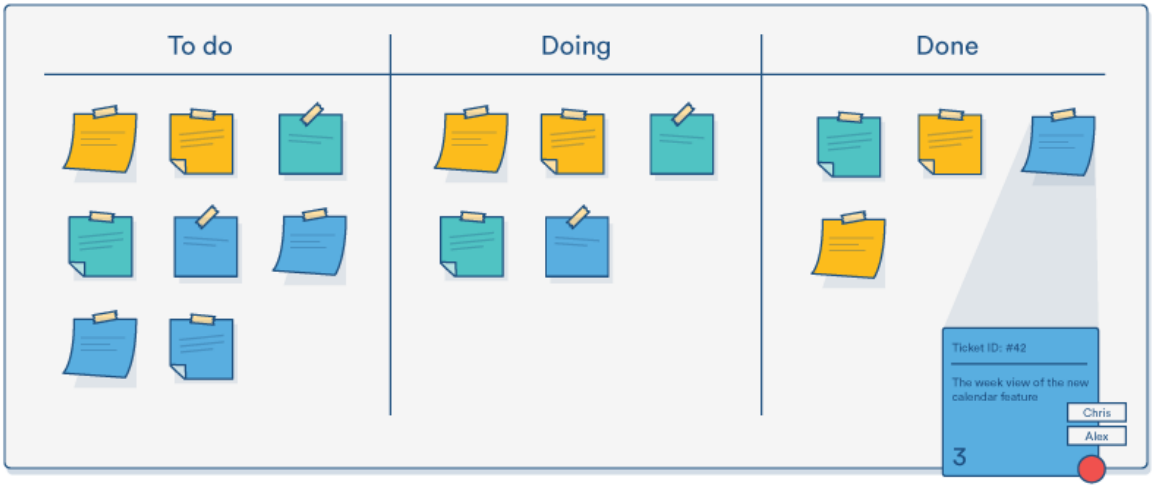
V dnešní době se místo samotných karet často používají počítačové programy, případně různé čipy, které umožňují sledování, například RFID.

1.5.2 Využití Kanban v softwarovém vývoji

Vývoj software probíhá podobně jako výroba v dílně. Je zde také problém rozpracovaného výrobku. Vývojový tým si převezme požadavek ze zásobníku, přesune se štítek na tabuli a tím pádem lze na požadavku pracovat. Po dokončení si jej může převzít testovací tým. Vedoucí pracovník vidí, že se uvolnilo místo pro vývoj dalšího požadavku. Omezíme tedy rozpracování více částí software, které by pak najednou bylo moc rozsáhlé otestovat (Kotačka, 2014).

Zároveň však *Kanban* není metodika, ale technika, systém. Lze ji použít jak na vodopádový model nebo agilní metodiku *Scrum* (viz Příklady agilních metodik). Tato technika neříká, jak vyvíjet software. Pro menší projekt nemusí být Scrum vhodný, například u projektů v nižších řádech stovek hodin je vhodné využít pouze Kanban. Proti Scrumu také není zákazník přímo součástí týmu. Ukázku kanbanové tabule zobrazuje obrázek 4.

Obrázek 4 Kanban *Zdroj: (Radigan, 2017)*



Kanban je definován pomocí pěti základních procesů, které definuje (Kotačka, 2014) takto:

- Vizualizuj pracovní postup (workflow) – Celý proces spočívá v rozdělení tabule na sloupce podle stavů našeho workflow. Jednotlivé úkoly (tasky) jsou prezentovány lístečky, které umístíme do odpovídajícího sloupce na tabuli. Případně můžeme využít softwarové nástroje pro vizualizaci.

- Limituj rozdělanou práci – Tento bod určuje zaměstnanci, který dokončil svoji práci, aby nejdříve pomohl ostatním kolegům s jejich úkoly, až poté začal pracovat na nových, ještě neotevřených úkolech.

- Měř a spravuj proces – Procesy, které jsou zvoleny pro průběh prací, musíme měřit.

- Udělej procesní politiky explicitní – Všichni, kteří pracují na procesech, musejí být seznámeni s procesem a musejí dodržovat pravidla a zásady. Každý musí vědět, co znamená „Hotovo" v daném stavu procesu.

- Použij modely pro rozpoznání příležitostí ke zlepšení – Využití analýzy úzkých míst ke zlepšení celkového fungování procesu.

1.6 Jidoka

Princip *Jidoka* lze popsat jako „automatizaci s lidským dotykem". V každém kroku výroby se kontroluje kvalita a každý člen je zodpovědný za provedení kontroly před odesláním na následující pracoviště. Při zjištění chyby se musí závada odstranit, i kdyby se měla výroba zastavit. Podpůrné procesy pro odhalení a okamžité řešení abnormalit zajišťují *andon* tabule, standardizace a *Poka-Yoke* (Vochozka, a další, 2012 str. 428).

1.7 Poka-Yoke

Poka-Yoke (v japonštině znamená „*poka*" neúmyslnou chybu a „*yokeru*" znamená předejít) je technika spočívající v nalezení a eliminaci chyb dříve, než způsobí problémy. V oblasti počítačových programů jsou to často okna, která „vyskočí a zobrazí hlášku", zda opravdu chcete odejít bez uložení. Případně software nabízí možnost automatického ukládání (Svozilová, 2011 str. 164).

Technika by tedy měla být včasná, čili pomoci v momentě, kdy nastane chyba. Zároveň by měla být přesná ve smyslu jednoduché identifikace, co způsobilo chybu. Údržba zachytávání poruch by měla být jednoduchá, u složitých řešení hrozí riziko vytvoření chyby

v Poka-Yoke. A v poslední řadě by řešení nápravy chyby mělo být natolik jednoduché, aby to neobtěžovalo uživatele.

1.8 Heijunka

I když zavedeme výrobu *Just-in-time*, nikdy nebudou schopni dodavatelé a ani výrobní linky reagovat bezproblémově na požadavky. *Heijunka* (zprůměrování) znamená vyvážit vyráběné množství, druhy výrobků a požadovaný čas (Vochozka, a další, 2012 str. 429).

1.9 Filozofie Kaizen

Kaizen vyjadřuje určitý filozofický směr, který úzce souvisí s pojmem štíhlý podnik a organizace. Výraz *Kaizen* je složený ze dvou slov, *„kai"* – změna a *„zen"* – lepší, tedy změna k lepšímu. Jedná se o neustálé zlepšování dnešní situace, aby zítra bylo lépe a nedocházelo k opakovaným chybám. Žádný proces nelze nikdy prohlásit za zcela dokonalý a vždy je prostor pro zlepšování. Podstata Kaizenu pochází z Japonska, kdy byl poprvé realizován ve firmách po 2. světové válce (Bauer, 2016 str. 13).

Kaizen lze rozdělit na tři navazující kroky:

- Definování současné situace
- Stanovení cílového stavu
- Činnosti vedoucí k dosažení požadovaného cíle

1.10 Další metody

Pro zjišťování plýtvání, odstranění plýtvání a zlepšování procesů slouží vybrané metody uváděné v této podkapitole.

1.10.1 Teorie omezení

Teorie omezení se soustředí na výkon procesů při využití maximálního průtoku přes úzká místa. Každý systém v sobě zahrnuje minimálně jedno úzké místo (omezení). Posílením tohoto nejslabšího článku dojde ke zvýšení pevnosti celého systému. Zároveň se však objeví nový nejslabší článek, který vyžaduje posílení. Jedná se tak o proces neustálého vylepšování. Tuto teorii původně definoval E. Goldratt v roce 1984 (Veber, 2006 str. 124).

1.10.2 Lean Six Sigma

Lean Six Sigma je metoda, která slouží ke zvyšování hodnoty společnosti prostřednictvím neustálému zdokonalování firemních procesů. Jedná se o spojení přístupu Lean, který optimalizuje procesy prostřednictvím odstraňování plýtvání a metodiky Six Sigma. Ta slouží ke snížení variability výstupů v procesech prostřednictvím odstraňování odchylek a závad (Svozilová, 2011 str. 62).

1.10.3 Metoda 5S

Metoda 5S přináší zlepšení pracovního prostředí a odstranění zbytečných aktivit, jejichž důsledkem dojde k ulehčení a zjednodušení práce a zvýší se přehlednost v materiálovém a informačním toku (Bauer, 2016 str. 90).

- Seiri (Roztřídit) – Roztřídění věcí podle toho, zda jsou zcela nepotřebné, využívané týdně či měsíčně anebo se jedná o věci každodenní potřeby. Rozdělení můžeme provést například barevnými nálepkami.

- Seiton (Srovnat) – Rozmístění potřeb a pomůcek používaných všemi pracovníky v kanceláři na jedno místo. Vyznačení minimální a maximální úrovně zásob. Tím se zredukuje čas k hledání věcí a zbytečná manipulace.

- Seiso (Vyčistit) – Odstranění zdrojů nečistot na pracovišti. Vznikne hygienické a bezpečné pracovní prostředí.

- Seikutsu (Standardizovat) – Stanovení standardů, které musejí být na pracovišti dodržovány. Dochází ke kontrolám na konci pracovní doby, eliminaci příčin a sledování odchylek.
- Shitsuke (Setrvat) – Dodržování stanovených postupů a standardů.

2 Vybrané nástroje štíhlé administrativy

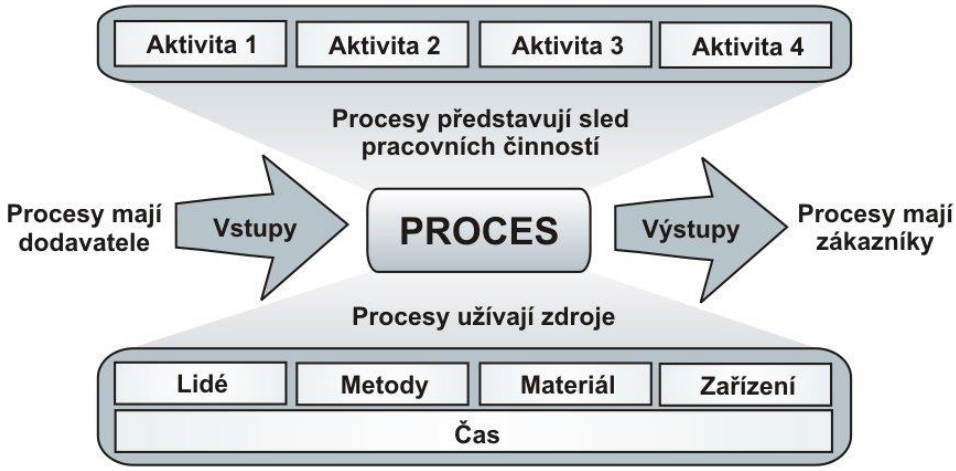
2.1 Procesní řízení

Procesní řízení je kontinuální činnosti managementu vedoucí k zavedení, provozu, rozvoji a neustálému zlepšování procesní organizace. Hlavní myšlenkou je rychlá reakce na změnové požadavky od zákazníka.

- Proces je souhrnem činností, transformujících souhrn vstupů do souhrnu výstupů pro jiné lidi nebo procesy, používajíce k tomu lidi a nástroje (Řepa, 2007 str. 15).

- Projekt je časově ohraničený sled jedinečných událostí a aktivit vedoucích ke splnění specifického cíle (Svozilová, 2011 str. 21).

Obrázek 5 Pohled na průběh procesu *Zdroj: (Hejna, 2007 str. 15)*



2.1.1 Rozdělení procesů

Procesy můžeme dělit z několika různých hledisek, nejčastěji se dělí podle důležitosti a účelu procesu z hlediska přidané hodnoty pro zákazníka.

- **Hlavní proces**

  Hlavní procesy vytvářejí přidanou hodnotu v podobě výrobku nebo služby pro zákazníka. Generují tržby a probíhají napříč celou organizací. Hlavní procesy je možné dále dělit na podprocesy, kde v sobě zahrnují menší část hlavního procesu.

- **Řídící proces**

  Tyto procesy zajišťují fungování organizace a její rozvoj. Řídí činnosti.

- **Podpůrné procesy**

  Podpůrné procesy zajišťují chod organizace. Jedná se například o komunikaci se zákazníkem, sledování kvality výrobků, případně měření.

2.1.2 Další dělení procesů

Procesy dále můžeme dělit na měkké a tvrdé. U měkkých procesů může být pořadí procesů měněno, kdežto u tvrdých procesů je přesně dáno jejich pořadí a nelze je měnit. Dalším rozdělením jsou procesy sériové a paralelní. U sériových probíhají procesy postupně za sebou, u paralelních je možné současné provádění. Existují i další rozdělení, ale pro potřeby této diplomové práce nejsou potřebné a nebudu je uvádět.

2.1.3 Modelování podnikových procesů

Při aplikaci procesního řízení je vhodné procesy a jejich průběh popsat, znázornit a namodelovat.

2.1.4 Základní oblasti procesního řízení

Při procesním řízení je potřeba dodržovat základní oblasti:

- Znalost procesů – Je důležité mít jasně definované procesy, ale také vědět jaké vstupy a výstupy jednotlivé procesy mají. Zároveň musejí být zřejmé zdroje, které jsou potřebné. Každý pracovník si musí uvědomovat svou roli v organizaci i smysl své práce.

- Měřitelnost procesů – Každý proces musí být měřitelný. Chceme-li procesy zlepšovat a optimalizovat, musíme je monitorovat a měřit.

- Zpětná vazba – Bez zpětné vazby od zákazníků často nedojde k vylepšení. Zákazník může být externí nebo interní. Interní zákazníci jsou ti, kteří jako vstup svých procesů používají jiné firemní procesy, mají tedy znalosti, co je potřeba zlepšovat.

2.2 Identifikace plýtvání

Identifikace plýtvání následuje v cyklu DMAIC ve fázi Analyzování (Analyse). Identifikace a eliminace plýtvání jsou základem štíhlé administrativy. Postupuje se ve třech fázích: vizuální, procesní, produktová, které popisuji níže a vycházel jsem z (Bejčková, 2015).

Visual Office Kaizen

Za první pilíř štíhlé administrativy je považován Visual Office Kaizen, pomocí kterého se snažíme identifikovat druhy plýtvání pouhým pohledem. Zkoumáme čistotu pracoviště, jeho bezpečnost, uspořádání položek a jejich použitelnost. Používáme zde především metodu 5S a tvorbu standardů na pracovišti.

Process Office Kaizen

Jedná se o druhý pilíř štíhlé administrativy, kde se zaměřujeme na nalezení a eliminaci plýtvání u administrativních procesů. Jedná se o procesy, které přímo souvisejí s přidanou hodnotou, případně přímo nesouvisejí, ale zásadně přidanou hodnotu podporují. Používáme zde mapování toku hodnot a procesní analýzy.

Object Office Kaizen

Třetí, a zároveň poslední pilíř štíhlé administrativy, se zaměřuje na samotnou optimalizaci produktu, který poskytujeme zákazníkovi (a to buď internímu, nebo externímu). Jedná se reporty, faktury, prezentace, apod.

2.3 Metody k identifikaci plýtvání

2.3.1 Interview s pracovníky

Dotazování probíhá mezi tazatelem a respondentem podle předem připravených otázek. Cílem je získat odpovědi od respondenta.

2.3.2 Procesní analýza

Graficky znázorněný sled aktivit ve výrobním i nevýrobním procesu pomocí symbolů.

2.4 Management toku hodnot

Management toku hodnot („Value Stream Management") je základní metodou při zeštíhlování. Síla metody spočívá v její jednoduchosti a rychlosti, kdy pomocí obrázku lze získat ucelený pohled na plýtvání v podniku. Pro zachycení se používá technika mapování toku hodnot. Vybere se tok hodnot, které se zlepší a vytvoří se mapa. Následně se odstraní procesy, které nepřidávají hodnotu, vytvoří se mapa budoucího stavu a tyto změny se implementují (Košturiak, a další, 2006 str. 43).

Při eliminaci plýtvání v procesech se postupuje následovně (Košturiak, a další, 2006 str. 45):

1. Vybereme tok hodnot, které chceme zlepšit
2. Zachytíme současný stav pomocí mapy a odstraníme činnosti, které nepřidávají hodnotu
3. Zachytíme mapu budoucího stavu
4. Závěrem implementujeme budoucí stav a vyhodnotíme výsledky

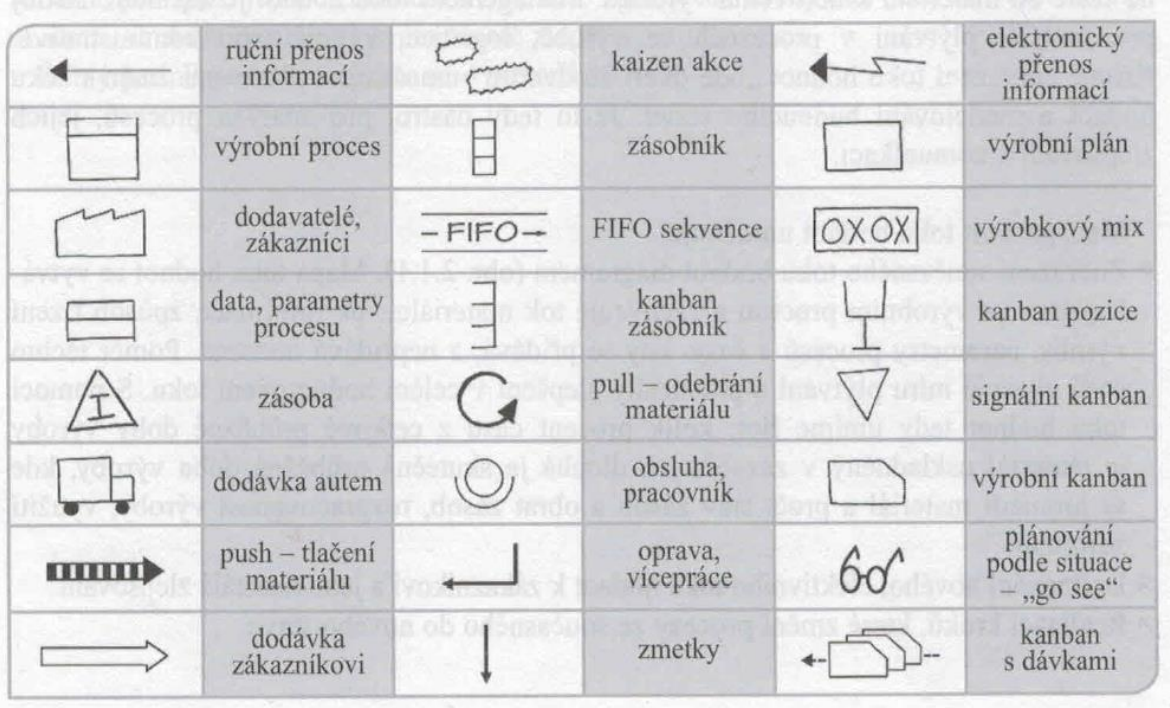
2.5 Mapování toku hodnot

Technika mapování toku hodnot (označovaná jako „Value Stream Mapping") pochází z japonské firmy Toyota, která ji využívala k analýze výrobních procesů a lokalizaci a následnému řešení úzkých míst. V administrativě se jedná o informační toky a služby. Materiálové toky proudí zleva doprava a informační naopak zprava doleva.

„Jedná se o grafickou techniku, která pomocí standardizovaných ikon popisuje souvislosti a vazby v materiálových i informačních tocích v konkrétním hodnotovém toku daného výrobku nebo rodiny výrobků" (Mašín, 2003 str. 10).

Mapa toku hodnot zachycuje tok informací a parametry procesů. Základní používané ikony znázorňuje obrázek 6. Výstupem je mapa současného stavu, která vizualizuje pozorovaný hodnotový tok, umíme tedy říci, kde dochází k problémům. Poté je výstupem nová mapa budoucího stavu (Košturiak, a další, 2006 str. 43).

Obrázek 6 Ikony mapování hodnotového toku *Zdroj: (Košturiak, a další, 2006 str. 44)*



Mapování toku hodnot můžeme využít při:

- Mapování průběhu administrativních a vývojových procesů
- Návrhu nových procesů
- Mapování průběhu operace

Přínosem je eliminace plýtvání v procesech, zjednodušení systému řízení a lepší pochopení průběhu procesů a souvislostí mezi nimi.

2.6 Metody k odstranění plýtvání

Pro odstranění plýtvání slouží Demingův cyklus, případně metoda DMAIC. Obě metody byly využity při testování implementovaného systému.

2.6.1 Demingův cyklus

Demingův cyklus je základem všech štíhlých způsobů řízení, často se označuje jako PDCA cyklus (Veber, 2006 str. 126). Jedná se o metodu postupného zlepšování výrobků, služeb, procesů pomocí čtyř základních činností, které jsem popsal pod obrázkem 7.

Obrázek 7 PDCA cyklus *Zdroj: (Chování.eu, 2017)*

- Plan – Naplánuj práci, která bude přinášet maximální hodnotu
- Do – Vyrob produkt podle zadání
- Check – Změř a zkontroluj výsledek
- Act – Vyhodnoť výsledky a urči, jestli změnit směr

2.6.2 Metoda DMAIC v projektovém řízení

Cyklus zlepšování, který je součástí metody Six Sigma. Pět jeho fází popisují anglické výrazy Define, Measure, Analyze, Improve, Control. Jedná se o systematický, cyklický proces nepřetržitého vyhledávání úzkých míst (tzv. *bottleneck*) a jejich odstraňování (Schwalbe, 2016 str. 315).

Obrázek 8 znázorňuje, jak probíhá DMAIC cyklus. Jednotlivé fáze jsou podrobněji popisovány pod obrázkem.

Obrázek 8 DMAIC cyklus *Zdroj: (ManagementMania, 2016)*

Jednotlivé fáze cyklu podle (Schwalbe, 2016 str. 315):

- D – Definovat (Define) – V této první fázi se definují cíle, získávají informace a popisuje se stav, kterého chceme dosáhnout. Zároveň se určují lidé, kteří to budou mít na starosti. Určuje se rozsah procesu, jeho vstupy a výstupy. Určujeme, čeho chceme dosáhnout, ale ne jak.

- M – Měřit (Measure) – Dosažení cílů je možné pouze na základě měření. Jak jinak totiž zjistit, že jsme cíle dosáhli. V této fázi určíme ukazatele a sesbíráme informace o současné situace.

- A – Analyzovat (Analyze) – Informace, které jsme zjistili, je nutné podrobně analyzovat. Musíme analyzovat příčiny problémů, nedostatků, atd. Cílem je zjistit a určit klíčové problémy, které mají vlivy na výskyt vad.

- I – Zlepšovat (Improve) – Odstraněním příčin problémů získáme zlepšení. Nastaví se nové parametry procesu a jeho optimalizace. Vše se děje za účelem zlepšení nákladů a zvýšení spokojenosti zákazníka. Cílem této fáze je vytvořit, vyzkoušet a následně implementovat řešení, která odstraní hlavní příčiny vzniku vad.

- C – Řídit (Control) – Pokud se nám podaří úspěšně odstranit vady a dojde ke zlepšení, je potřeba veškeré změny standardizovat. Je také nutné důsledně sledovat, zda jsou změny uplatňovány a jsou součástí každodenních činností. Cílem tedy je zabezpečení trvalého udržení zlepšeného stavu.

3 Specifické přístupy při vývoji SW

Etalonem business analýzy je standard BABOK (Business Analysis Body of Knowledge), kterou publikuje IIBA (International Institute of Business Analysis). BABOK je souhrn znalostí z oblasti business analýzy. Říká, co by měl analytik pro svou práci znát a umět. Není to metodika, přesto je to užitečný souhrn i pro někoho, kdo už v oblasti analýzy má praxi. Doporučuje, jak plánovat a řídit analýzu. Jak sbírat požadavky a řídit jejich životní cyklus. To znamená jednak jejich prioritizaci, ale také jejich aktuálnost. Dále obsahuje analýzu strategie, která popisuje současný stav, definuje stav budoucí a také rizika. Po sběru požadavků se provede analýza, zpravidla ve formě diagramů a také ověření, zda požadavky naplňují potřeby firmy (IIBA, 2015).

3.1 Business analýza a návrh

Prvky analýzy, které popisuje tato kapitola, jsou vhodné pro popis procesů, mapování aktivit a sběr požadavků. Pro mapování procesů a aktivit slouží CASE nástroje, kterých existuje více. Součástí business analýzy je popis současného (as-is) stavu obchodní procesů, návrh budoucího (to-be) stavu a sběr požadavků. Navazuje samotný návrh řešení.

3.2 UML notace

Jazyk UML je zkratka pro unifikovaný modelovací jazyk (z angl.. Unified Modeling Language). Je to jednotný jazyk grafický pro tvorbu diagramů, který vzniknul za účelem zpracování softwarových návrhů tak, aby mu rozuměli jak programátoři, tak i další zainteresované osoby. Jazyk podporuje objektově orientovaný přístup, kdy zachycuje relace a chování. Nejedná se o metodiku, jak navrhovat a znázorňovat systémy, nejedná se o projektové řízení ani o programovací jazyk, jedná se „pouze" o prostředky pro tyto činnosti (ARLOW, a další, 2007 str. 28).

Při použití UML si musíme ujasnit, **CO** se má udělat (komunikace mezi analytikem a uživateli), dále **JAK** (namodelování procesů, tříd, apod.) se to má udělat a následně je to potřeba **REALIZOVAT** (samotné vytvoření zdrojového kódu).

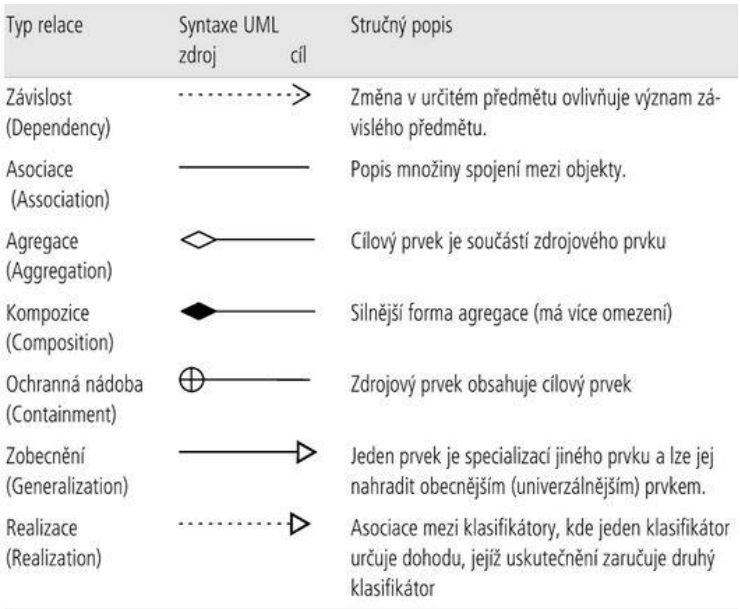
Standard UML definuje skupina OMG (Object Management Group) a je založená na pohledu 4+1 (viz Podkapitola: 4+1 pohled).

3.2.1 Základní stavební bloky UML

Mezi stavební bloky patří základní prvky modelů, vztahy mezi nimi a diagramy (ARLOW, a další, 2007 str. 35).

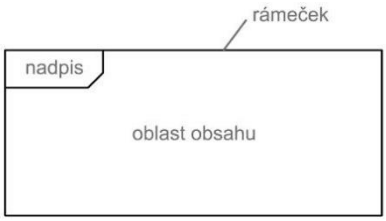
- Předměty – jsou to prvky modelů, které se dále dělí na:
  - Strukturní abstrakce – jedná se o podstatná jména modelu UML, jako jsou třída, rozhraní, případ užití, spolupráce, aktivní třída, komponenta a uzel
  - Chování – jedná se o slovesa modelu UML, jako například interakce, stav
  - Seskupení – jsou to balíčky k seskupování významově souvisejících prvků modelu do soudržných jednotek
  - Poznámky – jedná se o anotace, které se připojí k modelu s úmyslem zachytit informaci sestavenou jen k tomuto účelu
- Relace – umožňují zachytit významový vztah mezi dvěma a více předměty. Jejich typy a popis zobrazuje obrázek 9.

Obrázek 9 Typy relací v UML *Zdroj: (ARLOW, a další, 2007 str. 36)*



- Diagramy – Vizualizují pohledy na modely, co a jak bude systém dělat. Diagram není model. Diagramy se dělí na statické, popisující strukturu systému a dynamické, popisující chování systému. Syntaxi diagramu zobrazuje následující obrázek 10.

Obrázek 10 Diagram v UML *Zdroj: (ARLOW, a další, 2007 str. 38)*



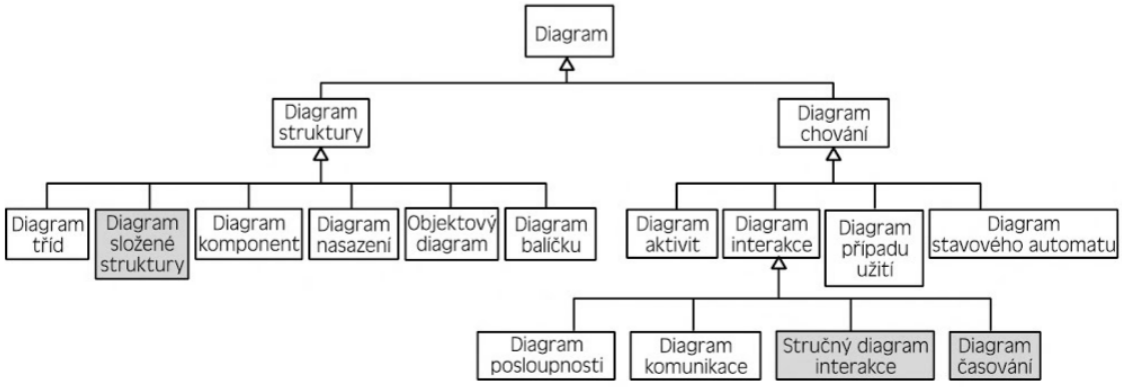
Násobnost vazeb

Multiplicitu neboli násobnost můžeme uvádět u vazeb. Zápis reprezentuje číslo označující konkrétní hodnotu „1“, hvězdička (*) nebo případně symbol n označující libovolný počet (i číslo 0) a interval značící se „1..*“. Pro objasnění uvedu příklady. U vazby typu „1:1“: Každý člověk má pouze jedno rodné číslo. U vazby typu „m:n“: Zákazník si do objednávky může přidat libovolný počet položek a objednávek může vytvořit více.

Diagramy

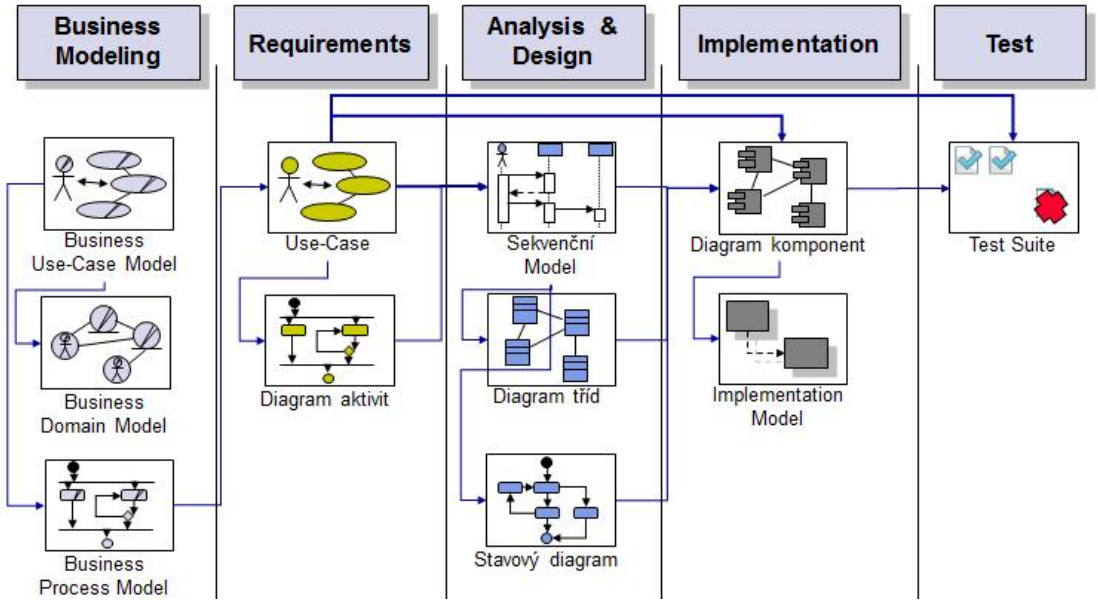
Diagramů je dohromady 14 a dělí se na dvě hlavní větve. První jsou strukturní diagramy, které popisují předměty a jejich chování. Druhá část popisující chování předmětů, se označuje jako diagramy chování. Rozdělení diagramů znázorňuje obrázek 11.

Obrázek 11 UML diagramy *Zdroj: (ARLOW, a další, 2007 str. 37)*



Každý z diagramů se používá v jiné fázi vývoje, jak znázorňuje obrázek 12. Pro účely této práce budu v praktické části používat pouze některé vybrané diagramy.

Obrázek 12 Dělení UML *Zdroj: (Žoltá, 2016)*



3.2.2 Obecné mechanismy UML

Jazyk UML obsahuje čtyři mechanismy, které jsou používány v celém jazyku konzistentně (ARLOW, a další, 2007 str. 49). Prvním mechanismem jsou **specifikace**. Jsou textovým popisem sémantiky jednotlivých prvků. Následují **ornamenty**. Každý element může mít buď jednoduchý tvar, nebo k němu můžeme přidat dodatečné informace (ornamenty). Měly by se používat, pouze pokud zvyšují srozumitelnost a čitelnost modelu. Třetím mechanismem jsou **podskupiny**.

Poslední jsou **mechanismy rozšiřitelnosti**:

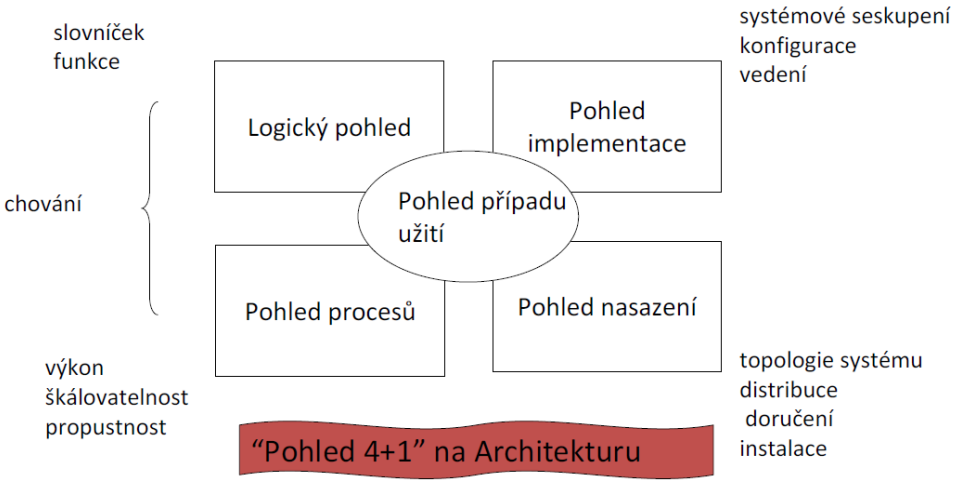
- Omezení – text ve složených závorkách {}, např. {každý uživatel má jedinečný email}. Podmínka v těchto závorkách musí být vždy splněna. Jako standardní rozšíření UML definuje jazyk OCL (Object Constraint Language).
- Stereotypy – s jejich pomocí lze vytvořit nový element. Zápis je ve dvojitých ostrých závorkách, např. „ <<nový>>“. Případně může mít stereotyp přiřazen symbol (např. notebook).
- Označené hodnoty – umožňují přidávat nové vlastností k elementům modelu, např. {autor=Martin}.

3.2.3 4+1 pohled

Pohled 4+1 je architekturní styl k organizování aplikační struktury pro splnění potřeby zadavatelů. Základem jsou případy užití, na které se lze podívat logickým, procesním, implementačním pohledem a pohledem nasazení. K jednotlivým pohledům lze poté přiřadit jednoduše příslušné UML diagramy (Rábová, 2016).

Následující obrázek 13 znázorňuje tento pohled. Začínáme pohledem případu užití, resp. Use-case diagramy, které popisují, CO za aktivity se bude vykonávat a kdo je bude vykonávat. Následující pohledy popisují, JAK se mají aktivity vykonávat.

Obrázek 13 Pohled 4+1 na architekturu *Zdroj: (Šimerda, 2005)*



Jednotlivé pohledy podle (ARLOW, a další, 2007 str. 49):

- Pohled případu užití – odhaluje základní požadavky na architekturu. Obsahuje diagram případu užití. Pohled případu užití zajímá koncové uživatele.
- Logický pohled – logická struktura toho, co by měl systém vykonávat, identifikuje hlavní balíčky, třídy a vazby mezi nimi. Jedná se o diagramy tříd, objektů, balíčků a stavové. Logický pohled zajímá především analytiky a designery.
- Pohled procesů – pohled na chování systému, jeho propustnost, zotavení z chyb, rozšiřitelnost systému a škálovatelnost. Obsahuje diagramy sekvenční, komunikace, aktivit, interakce a času. Procesní pohled zajímá systémové integrátory.
- Pohled implementace – zaměřený na rozdělení systému do menších samostatně realizovatelných komponent. Obsahuje diagram komponent. Implementační pohled zajímá především programátory.

- Pohled nasazení – fyzické rozložení komponent, nasazení, instalace a ladění výkonu. Obsahuje diagram nasazení. Pohled nasazení zajímá všechny tvůrce systému.

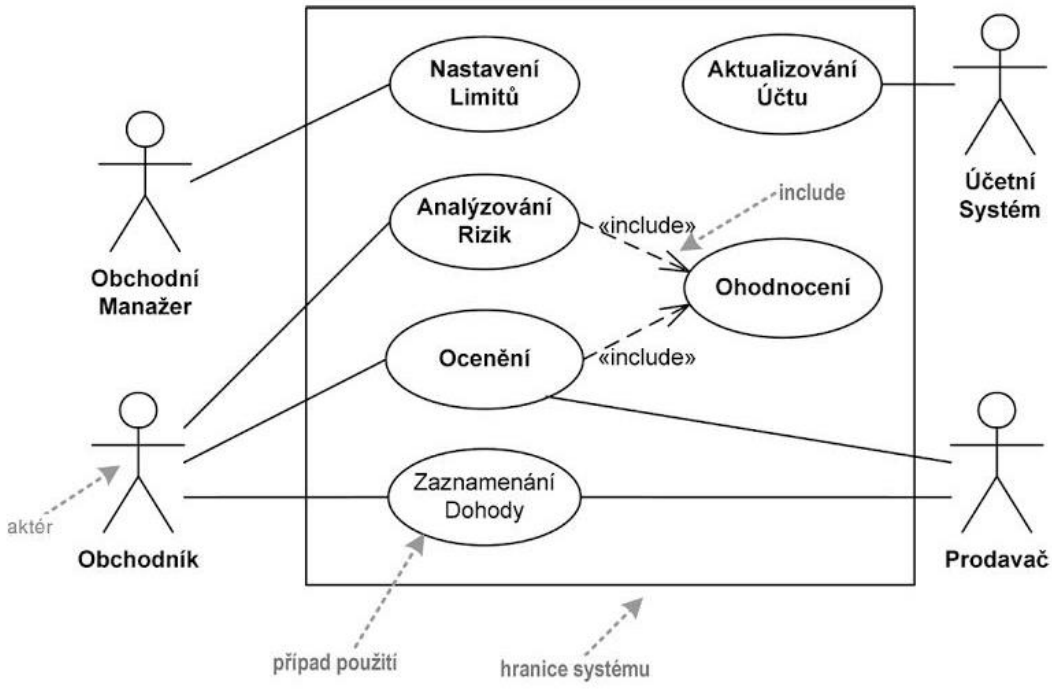
3.2.4 Diagram případů užití

Při návrhu software se sbírají požadavky od zadavatele a vzniká potřeba si je vizualizovat. Pro tyto účely slouží diagram případů užití označovaný také jako Use case diagram. Zobrazuje interakci mezi uživatelem a systémem. Prvky diagramu jsou aktéři, případy užití a relace. Případ užití se značí oválem a uvnitř je popis dané činnosti (ARLOW, a další, 2007 str. 96).

Účastník (actor) představuje v systému spouštěcí mechanismus pro danou funkcionalitu systému. Zobrazují se vně systému. Může jimi být cokoliv, co spouští systém. Tedy i například čas, kdy se v pravidelných intervalech provádí zálohování, apod. Účastníka znázorňujeme symbolem postavy a pod ním je uvedeno jeho jméno.

Relací je myšlený vztah mezi účastníkem a případem užití. Diagram čteme zleva doprava. „*Rozlišujeme tyto druhy: relace přístupu systému znázorněná plnou čárou, include (pokud jeden případ zahrnuje jiný případ), extend (pokud nějaký případ rozšiřuje chování jiného) a generalizace (vyjadřuje dědičnost mezi objekty)*" (Rejnková, 2009). Podobu diagramu zobrazuje obrázek 14.

Obrázek 14 Příklad diagramu případu užití *Zdroj: (Fowler, 2009 str. 106)*



3.2.5 Diagram tříd

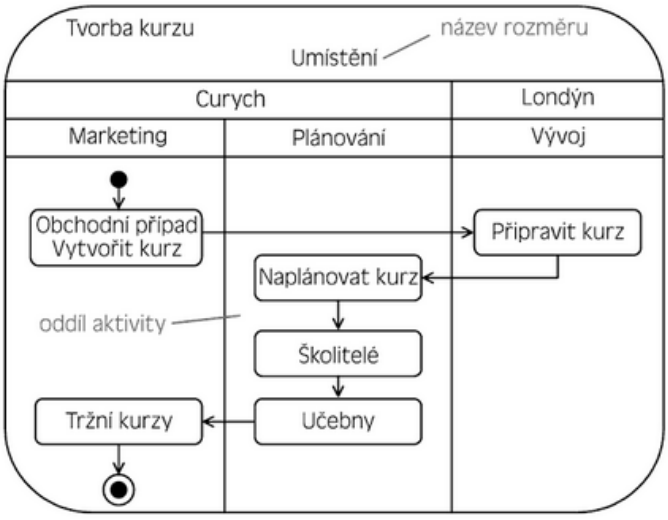
Diagram tříd poskytuje statický náhled na systém. Znázorňuje jednotlivé třídy, operace u objektů a jejich vazby. Když jej programátor přepíše do kódu, kód musí fungovat.

Třídu reprezentuje obdélník, kde je dále definovaný název, stereotypy, atributy, metody a výjimky. Před každým atributem je uveden modifikátor přístupu, tedy, jak je atribut viditelný z jiných částí systému. Rozlišujeme čtyři modifikátory. Mínus značí privátní atribut, plus veřejný, znak hash (#) protected a tilda (~) atribut viditelný z balíčku, jak uvádí (ARLOW, a další, 2007 str. 153).

3.2.6 Diagram aktivit

(ARLOW, a další, 2007 str. 286) popisuje diagram aktivit vhodný pro zachycení toků činností. Je užitečný při analýze obchodních procesů a případu užití, kde nabízí zjednodušený pohled na dění v průběhu procesů. Zachycuje konkrétní aktivity a jejich posloupnost. Posloupnost může být sekvenční, případně paralelní. Diagram začíná zahajujícím uzlem, končí ukončujícím. Aktivity se zachycují do plaveckých drah, tzv. Swimlanes, které obsahují vzájemně interagující aktéry, viz Obrázek 15.

Obrázek 15 Diagram aktivit *Zdroj: (ARLOW, a další, 2007 str. 292)*



3.2.7 Sběr požadavků

Účelem je popsat, jaké by mělo být chování systému, jeho vlastnosti a omezení na něj kladená (Plechatý, 2013). Při sběru požadavků často dochází k problémům jako je nejasná představa, neochota komunikovat, měnící se prostředí nebo například různé „jazyky" komunikace. Sběr může probíhat na schůzkách, kdy je ale důležitá volba osob, které se budou účastnit. Jiný pohled bude mít každodenní uživatel a vedení společnosti. Dále je možné využít dotazníků, apod. Požadavky se dělí na dvě skupiny, funkční a nefunkční.

Zjednodušeně se dá říci, že zatímco funkční požadavky definují CO má systém dělat, nefunkční požadavky popisují JAK má systém fungovat (ARLOW, a další, 2007 str. 80).

Funkční požadavky

Funkční požadavky definují, co v systému má být implementováno, aby to uspokojilo potřeby zákazníka.

Příkladem může být: „Systém bude ověřovat validitu uživatele pomocí hesla".

Nefunkční požadavky

Tyto požadavky definují omezení na systém. Musí být ověřitelné. Nefunkční požadavky se přímo netýkají konkrétních poskytovaných služeb zákazníkům.

Příkladem může být následující požadavek: „ Systém bude používat písmo Technika v exportovaných dokumentech".

3.3 Business Process Model and Notation

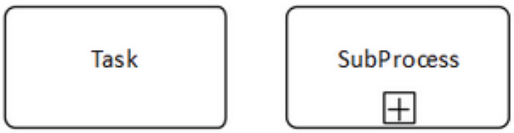
Business Process Modeling Notation (BPMN) je grafická notace pro modelování procesů. Oproti objektově orientovanému UML je tedy vhodnější pro jejich modelování. Vyplňuje mezeru mezi analýzou procesů a jejich následnou implementací. Notaci vytvořila iniciativa BPMI (Business Process Management Initiative) za účelem čitelnosti pro všechny účastníky životního cyklu procesu (jedná se například o business analytiky, vývojáře a další) (Briol, 2013 str. 22).

Základními prvky diagramů této notace jsou aktivity, propojovací objekty, brány a události.

Aktivity

Aktivita je podle (Šedivá, 2009 str. 301) jakákoliv činnost, změnu stavu, pod-proces nebo úlohu, ke kterým dochází v modelovaném procesu. Může být atomická nebo v sobě může obsahovat samostatný proces, tzv. pod-proces, který dole uprostřed přidává plus, viz Obrázek 16.

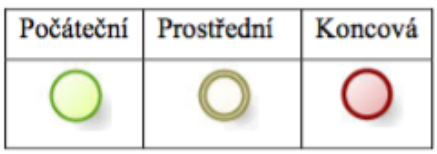
Obrázek 16 Znázornění aktivit v BPMN *Zdroj: (Stachecki)*



Události

Událost je podle (Šedivá, 2009 str. 301) v BPMN určena k zachycení stavu, ve kterém se proces nachází, a znázorňujeme ji pomocí kolečka. Standard rozlišuje grafický symbol, podle toho, jestli událost nastává na začátku procesu (počáteční), při běhu procesu (prostřední), či na konci procesu (koncová). Základní události znázorňuje obrázek 17.

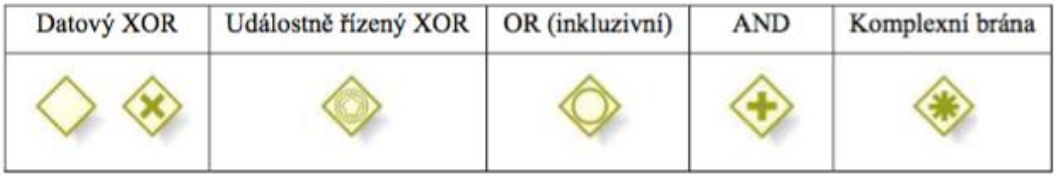
Obrázek 17 Typy událostí *Zdroj: (MÁČEL, 2009 str. 19)*



Brány

Brány uvádí (MÁČEL, 2009 str. 20) jako objekty pro řízení větvení a slučování toků nebo procesů. Reprezentuje je čtverec, případně kosočtverec stojící na špici. Brány se dělí podle přístupu k podmínkám, viz Obrázek 18.

Obrázek 18: Typy bran *Zdroj: (MÁČEL, 2009 str. 20)*



Datový XOR - rozděluje proces na dvě a více cest přičemž vykonána bude pouze jedna na základě podmínky. Standard umožňuje definovat cestu, která se provede, pokud žádná jiná cesta nesplňuje podmínku brány.

Událostně řízený XOR - rozděluje cestu stejně jako datový XOR s tím rozdílem, že každá cesta čeká na naplnění události jako je přijetí zprávy nebo vypršení časového limitu. Proces pokračuje pouze jen jednou z cestou, ve které dojde k naplnění události.

OR - umožňuje pokračovat jednou nebo více větvemi zároveň.

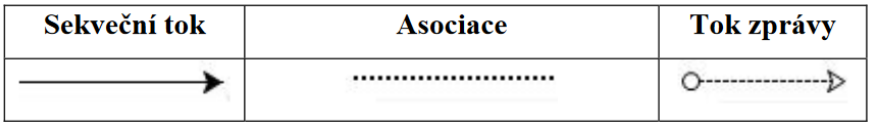
AND – vyjadřuje paralelní pokračování v procesu všemi přípustnými cestami.

Komplexní brána - se používá pro složité větvení popsané v připojeném výrazu, kombinuje chování více typů bran.

Spojovací objekty

Spojovací objekty rozděluje (MÁČEL, 2009 str. 21) na sekvenční tok, asociaci a tok zpráv. Sekvenční tok definuje pořadí provádění činností v procesu. Musí existovat alespoň jedna cesta od počáteční do konečné události. Asociace znázorňuje vzájemné spojení dat a artefaktů s ostatními grafickými elementy. Tok zprávy znázorňuje komunikaci mezi účastníky procesu. Typy spojovacích objektů zobrazuje obrázek 19.

Obrázek 19 Typy spojovacích objektů *Zdroj: (MÁČEL, 2009 str. 21)*



Plavecké dráhy

Plavecké dráhy (Swimlanes) umožňují kategorizovat, opticky organizovat činnosti a sledovat jejich vzájemnou interakci. Dráhy jsou součástí bazénu, který definuje hranice procesu a reprezentující účastníky. Jednotlivé bazény můžeme propojit pomocí toku zpráv (Šedivá, 2009 str. 303).

Dráhy rozdělují bazén na činnosti, které provádějí jednotliví aktéři. Propojení mezi drahami dochází pomocí sekvenčních toků.

Artefakty

Artefakty umožňují rozšířit procesy o další elementy, které pomáhají zvyšovat informační hodnotu modelu. Za artefakty lze považovat datové objekty, skupiny a anotace. Datové objekty znázorňují data využívaná při vykonávání dané aktivity, připojují se pomocí asociační vazby. Skupiny umožňují seskupování prvků, nezasahují ale do samotného toku diagramu. Anotace zvyšují přehlednost a srozumitelnost modelu (Vašíček, 2008).

3.3.1 CASE nástroje

CASE je zkratka pro Počítačem podporované systémové/softwarové inženýrství (z angl. Computer Aided Systems/Software Engineering). Jak uvádí (Gála, a další, 2015 str. 31), jedná se o nástroje pro analýzu a návrh software. Návrh probíhá pomocí modelování diagramů a následné generování kódů z těchto modelů. Dále je možné modely sdílet, vytvářet z nich dokumentaci. CASE nástroje se používají, protože je jednodušší pochopit obrázek, než složitě napsaný text. Snižují chybovost, zvyšují produktivitu práce a usnadňují údržbu.

Příklady nástrojů mohou být MS Visio (Microsoft), Enterprise Architect (Sparx System), Oracle Designer (Oracle).

3.4 Vývoj SW

Aby se dal vývoj software řídit, existují metodiky tradiční a agilní, které se liší přístupem, a jejich popis uvádím v následujících podkapitolách. Zároveň popisuji metodiku MMDIS, která je vyvíjena na Vysoké škole ekonomické v Praze a popisuje 11 principů. Dále popisuji metodiky, které slouží k celkovému řízení IT.

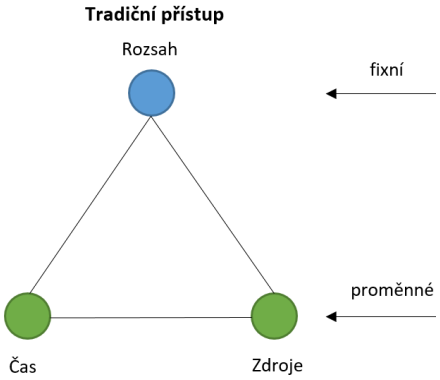
3.4.1 Metodiky vývoje software

Metodiky se dělí na dvě velké skupiny: tradiční a agilní metodiky. U každého projektu uvažujeme tři klíčové konstanty: čas, náklady (zdroje) a rozsah (funkcionalita).

Tradiční metodiky

Tradiční metodiky jsou historicky největší skupina metodik. Charakterizuje je vodopádový model, kde probíhá posloupnost jednotlivých činností během celého vývojového cyklu (Bruckner, a další, 2012 str. 102).

Obrázek 20 Tradiční přístup *Zdroj: vlastní zpracování*

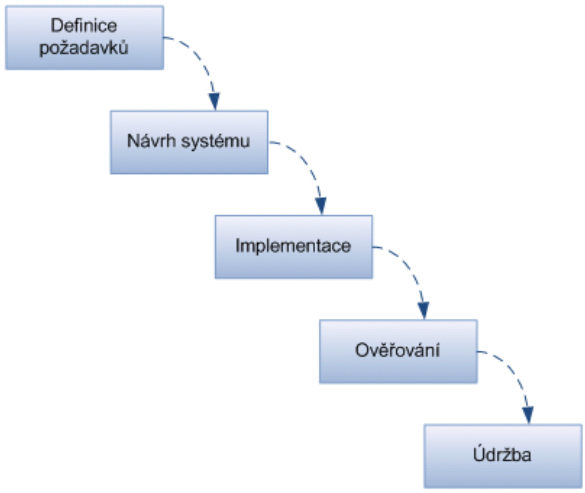


U tohoto typu metodiky je neměnnou veličinou rozsah projektu, čas a náklady jsou variabilní, jak je zobrazuje obrázek 20. Výhodou jsou dobré podmínky pro řízení projektu, procesy jsou opakovatelné a předpokládají definování všech požadavků předem. Nevýhodou je tedy neměnnost požadavků v průběhu projektu, metodika se neumí příliš dobře vypořádat s náhlými změnami.

Vodopádový model

Nejstarší model tradiční metodiky, který dostal název podle postupného provádění fází, kdy znázorněný model na obrázku 21 připomíná vodopád.

Obrázek 21 Vodopádový model *Zdroj: (Pavlík, 2012)*

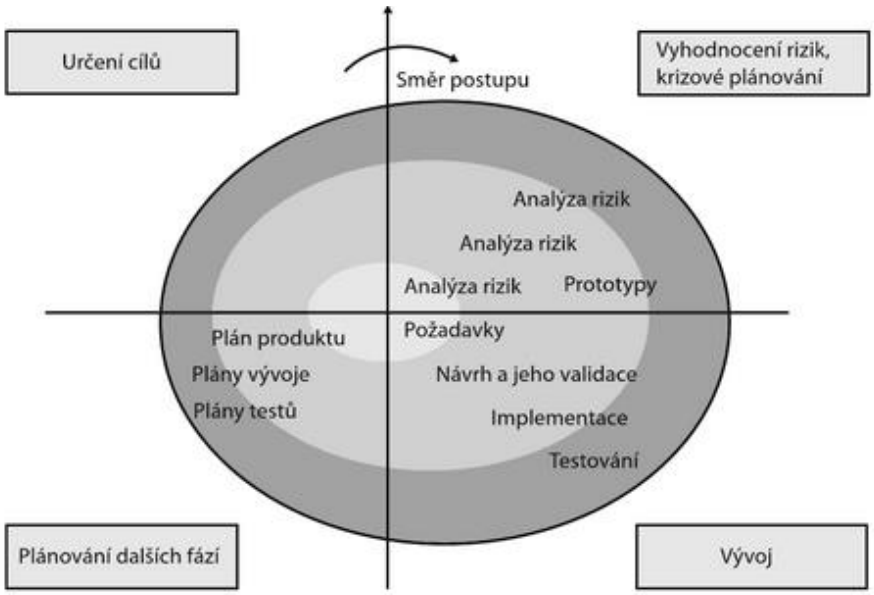


Každá fáze je nejprve schválená, teprve poté se přejde k další fázi cyklu. Není možné se vracet do předchozích fází. To značí nevýhodu, kdy není možné reagovat na změnové požadavky klientů v průběhu realizace projektu (Bruckner, a další, 2012 str. 108).

Spirálový model

Spirálový model (Obrázek 22) je založen na iterativním přístupu, zavádí opakovanou analýzu všech rizik a je tak vhodný i pro větší projekty (Page, a další, 2017 str. 64).

Obrázek 22 Spirálový model *Zdroj: (Page, a další, 2017 str. 64)*



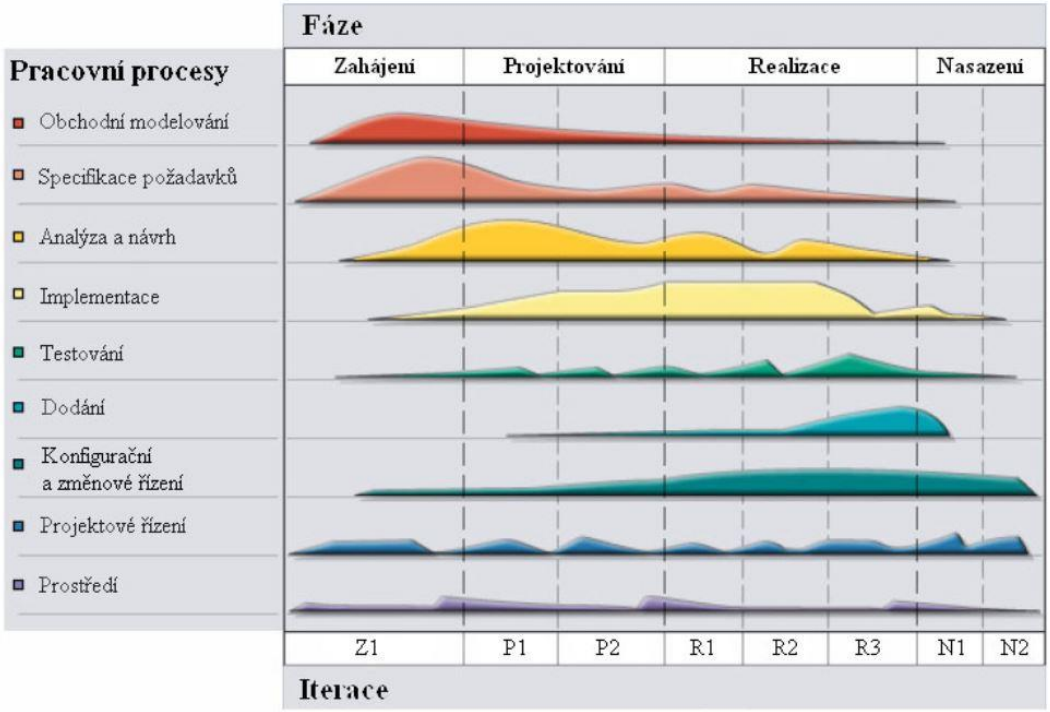
Model má 4 kvadranty:

1. Určení cílů - Stanovení cílů a stanovení omezujících podmínek projektu
2. Analýza vnějších a vnitřních rizik
3. Realizace -V této fázi cyklu se provádí výroba aktuálního stavu projektu.
4. Plánování, zhodnocení realizovaného cyklu a přidělení zdrojů pro další cyklus

Rational Unified Process

Rational Unified Process (RUP) je objektově orientovaný iterativní přístup k životnímu cyklu software. Využívá jazyka UML pro modelování procesů (Bruckner, a další, 2012 str. 117). Metodika RUP obsahuje čtyři základní fáze (viz Obrázek 23), každá obsahuje několik dalších iterací. Před začátkem každé iterace musí být splněna kritéria předchozí iterace.

Obrázek 23 Rational Unified Process *Zdroj: (Julinek, 2008)*

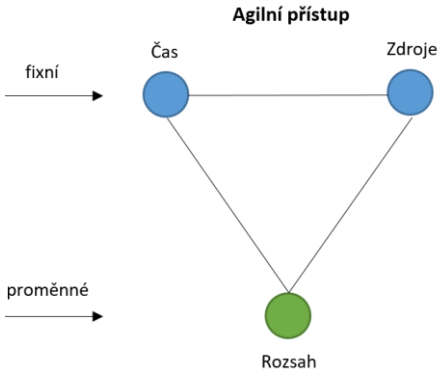


Fáze zahájení definuje rozsah projektu a účel. Následuje projektovací fáze, kde je potřeba analyzovat požadavky zákazníka a celého projektu. V realizační fázi probíhá samotná tvorba zdrojových kódů, proto je i časově nejvíce náročná. Předávací fáze obsahuje předání zákazníkovi, případně do dalšího cyklu.

Agilní metodiky

Agilní metodiky kladou velký důraz na zákazníka. Nejedná se o proces, ale spíše o přístup k vývoji software.

Obrázek 24 Agilní metodika *Zdroj: vlastní zpracování*



Jak zobrazuje obrázek 24, za neměnné považují zdroje a čas a jako variabilní složka je rozsah. Na počátku se stanoví nejdelší možný čas a náklady. Název metodiky vzešel z anglického slova *agile,* to znamená svižný, případně hbitý. Metodika podle (Page, a další, 2017 str. 65) tedy pružně reaguje na změnu. To je ohromná výhoda, protože zákazník často až za běhu projektu upřesňuje své požadavky.

Na začátku projektu se samozřejmě definuje, co má být výstupem, ale pouze v hrubých obrysech. Až během projektu se díky úzké spolupráci se zákazníkem projekt postupně upřesňuje.

Příklady agilních metodik

Při popisu následujících příkladů jsem vycházel z (Šochová, a další, 2014 stránky 27-124):

Adaptive Software Development – Adaptivní vývoj softwaru nechápe odchylku od plánu jako chybu, ale jako příležitost k učení. Poskytuje iterativní fáze spekulace – spolupráce – učení.

Test Driven Development – Filozofií je, že testovací kód musí být připraven před začátkem psaní kódu. Výhodou je předvídatelný software, kde je prozkoumané chování. Nevýhodou naopak pevné řízení projektu a pro programátory nepohodlí psát testy na počátku.

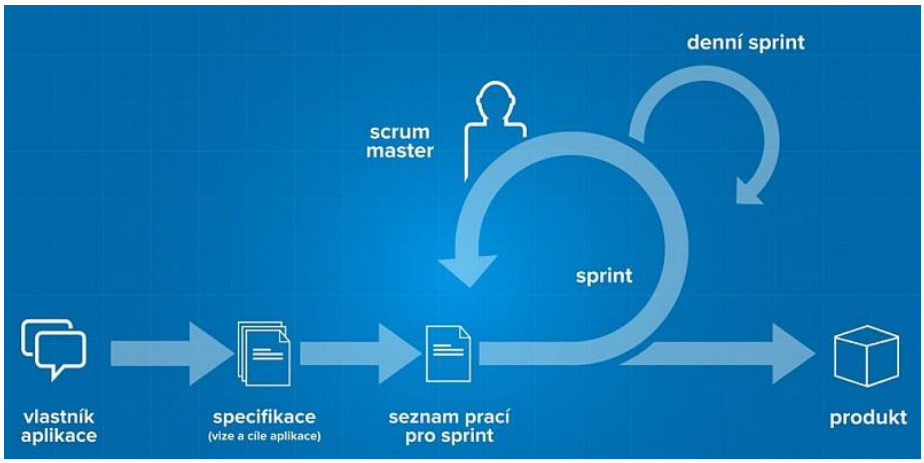
Feature Driven Development – Vlastnostmi řízený vývoj, založený na krátkých iteracích. Vývoj je založen na vytvoření globálního modelu systému. Z něj by mělo být zřejmé, celkové směřování vývoje. Cílem je naznačit co bude systém obsahovat a jak

bude s okolím komunikovat. Vývoj probíhá v pěti fázích. První tři jsou sekvenční a poslední dvě iterativní. Iterace zpravidla trvají 2 týdny. Metodika je vhodná pro menší projekty. Vlastnost je jakákoli malá, užitečná funkčnost z pohledu zákazníka. Je měřitelná, srozumitelná a realizovatelná (Danel, 2011).

Extrémní programování – Myšlenkou této metodiky je dodávání softwaru v krátkých vývojových cyklech. Je to spolehlivá metoda, která nabízí nejvyšší možnou kvalitu a maximální rychlost při učení se. Ale není spolehlivá pro odhad odevzdání kompletní dodávky. Extrémní se nazývá, protože dotahuje běžné principy do extrému. Například, pokud se vyplácí testování, budeme my i zákazník nepřetržitě testovat.

SCRUM – Název pochází z rugbyové terminologie (Mlýn hráčů se společně snaží dostat míč na pozici). Cílem je dodat funkční část zákazníkovi co nejdříve, dostat od něj zpětnou vazbu, zpracovat jeho požadavky a takto stále dokola. Vývoj prochází intervaly, tzv. „sprinty" (nejdéle by měl trvat měsíc). Po odsouhlasení náplně sprintu, je zahájen jeho běh a obsah by se neměl po dobu jeho trvání měnit. Každý den jsou krátké Scrum Meetings, na kterých se shrnují dosavadní výsledky a identifikují nové úkoly. SCRUM tedy nemá přesně naplánovaný vývoj, ten se určuje denními úkoly (Jonák, 2016). Průběh cyklu je zobrazen na obrázku 25.

Obrázek 25 Metodika SCRUM *Zdroj: (Malkusová, 2016)*



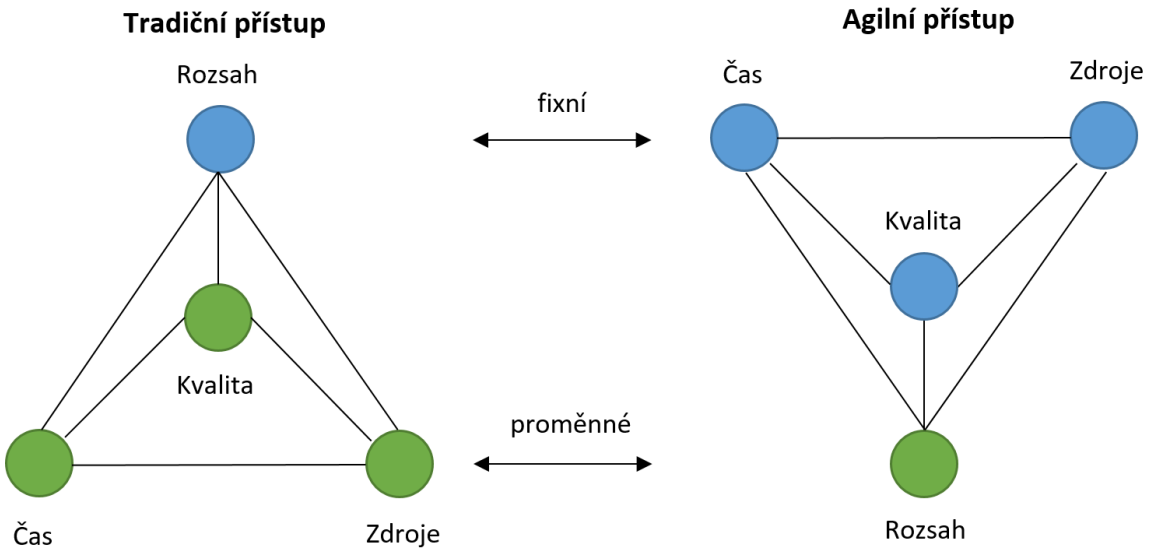
V metodice SCRUM existují 3 druhy rolí: Vývojový tým, vlastník aplikace a Scrum master. Ten je prostředník mezi zbývajícími dvěma rolemi. Poskytuje vývojovému týmu veškeré prostředky, které potřebuje a dohlíží na dodržování pravidel metodiky. SCRUM tým má obvykle velikost přibližně 7 lidí (Malkusová, 2016).

Porovnání rozdílů v metodikách

Jak bylo uvedeno výše, tradiční přístup má jako proměnné konstanty čas a zdroje. Tedy dodáváme předem dohodnutou věc, ale čas a rozpočet se můžou měnit. Když pohled rozšíříme o kvalitu dodání, vidíme u tradičního přístupu, že čím více se blížíme k termínu dodání, trpí kvalita a dochází k prodloužení času a navýšení zdrojů

U agilního přístupu je přístup ke kvalitě odlišný. Za daný čas a dané zdroje chceme co nejlepší kvalitu, a proto je proměnnou částí rozsah. To znázorňuje i obrázek 26.

Obrázek 26 Porovnání rozdílů v metodikách *Zdroj: (Vlastní zpracování, 2017)*



V tabulce 1 jsem porovnal oba přístupy k vývoji software. Odlišností je mnohem více, vybral jsem sedm, které z mého pohledu nejvíce odrážejí rozdíly.

Tabulka 1 Porovnání tradičního a agilního přístupu *Zdroj: vlastní zpracování podle (Buchalcevová, 2005 str. 51)*

	Tradiční	Agilní
Přístup ke změnám	Snaha minimalizovat změny	Jsou vítány a umožňují zákazníkovi přehodnotit své požadavky s ohledem na nové skutečnosti
Zásah zákazníka	V předem stanovených milnících, především v začátku a na konci fáze	Průběžně, je řídícím subjektem celého projektu

Dokumentace	Rozsáhlá	Není podstatná dokumentace, ale pochopení
Orientováno na	Procesy a dokumentaci. Lidé jsou nahraditelní	Lidé jako klíčový faktor
Kvalita	Zaměřeno na kvalitu procesů, ne přínosu pro zákazníka	Zaměřeno na přínos pro zákazníka a vysokou kvalitu produktu
Čas k objevení problémů	Dlouhý	Krátký
Způsob vývoje	Iterativní vývoj s dlouhými iteracemi	Inkrementální (přírůstkový) vývoj s krátkými iteracemi
Předpoklady	Požadavky je možné definovat předem	Předem je možné definovat jen hrubé požadavky
Použití	Velké, standardní projekty	Menší projekty, kde je cílem co nejdříve dostat funkční část na trh

3.4.2 Řízení procesů provozu a rozvoje ICT

Pro nastavení a řízení procesů existují různé metody a rámce. Patří mezi ně MMDIS, ITIL a COBIT, které postupně představím. V rámci této diplomové z důvodu účasti pouze zpracovatele práce, nebyla žádná tato metodika použita, je však vhodné je zmínit. Jsou to metodiky používané na projektech, kde je více členů a pomáhají tyto projekty řídit.

Metodika MMDIS

MMDIS je zkratka Multidimensional Management and Development of Information Systems a nepřekládá se. Jedná se o multidimenzionální metodiku v přístupu k vývoji informačních systémů. Není to striktní metodika, jedná se spíše o průvodce průběhu projektu a napovídá, co a proč dělat v které fázi. Od roku 1990 ji vyvíjí Katedra informačních technologií na Vysoké škole ekonomické v Praze (Bruckner, a další, 2012 str. 120).

11 základních principů metodiky MMDIS

Při popisu základních principů metodiky MMDIS, které vznikly generalizací osvědčených přístupů, popisuje (Bruckner, a další, 2012 stránky 126-137).

Multidimenzionalita

Na každý složitý problém je nutné nahlížet z různých pohledů (dimenzí). Při řešení pomocí MMDIS metodiky se doporučuje nahlížet na informační systém pomocí pohledů. Uživatelský pohled odpovídá na otázku, komu je informační systém určen a jaké ICT služby bude nabízet jednotlivým skupinám uživatelů. Druhý pohled je řešitelský, který odpovídá na otázku, jak systém vytvoříme, jak ho budeme provozovat a jaké ICT služby budeme dodávat. Dále můžeme nahlížet pomocí úrovně abstrakce nebo dle obsahové dimenze.

Integrace

Každý složitý systém má mnoho částí (komponent/modulů) a obsahuje mnoho vazeb mezi nimi. Vývoj a provoz systému je spojen s řízením těchto vazeb.

Vrstevnost

Při řešení složitého problému jej dekomponujeme do různých úrovní abstrakce, kdy každá nižší vrstva řeší problém na větší úrovni podrobnosti.

Flexibilita

Vyvíjí se nejen okolí systému, ale i požadavky na chování systému, a proto musí být systém schopen se těmto změnám přizpůsobit, pokud možno snadno a rychle. Řešením je parametrizace.

Otevřenost

Větší a rozsáhlejší změny je často nutné řešit novými komponentami, proto musí být systém schopný odebírat staré komponenty a nahrazovat je novými. Nutností je navrhnout architekturu tak, aby systém měl co nejmenší vazby mezi svými komponentami. Často je vhodné využívat zejména standardizované komponenty právě pro již zmíněné co nejmenší vazby.

Standardizace

Mnohá řešení jsou zavázaná standardy (normy, směrnice, zákony). Často je ale možné si navrhnout vlastní standardy, případně použít všeobecně doporučované. Zlevní se tím celé řešení, protože můžeme využít znovupoužitelnosti komponent.

Kooperace

Cílem celého vývoje je kvalita, škálovatelnost a rychlá reakce. Toho dosáhneme, pokud najdeme klíčové kompetence, na kterých postavíme svůj podnikatelský záměr. Na ostatní záležitosti je většinou levnější získat je od obchodních partnerů.

Procesní pojetí

Pro analýzu sociálně-ekonomických systémů je procesní popis lepší, než funkčně organizační. Jasně zachycuje způsob reakce systému na významné události.

Učení a růst (postupné zlepšování procesů)

Cílem principu je systematické zlepšování procesů a řízení firmy. Učením získáváme znalosti a lepší postupy, které jsou klíčem ke konkurenceschopnosti.

Lokalizace zdrojů a rozhodnutí

Zdroje a rozhodnutí lze řešit centralizovaně (převažuje u ICT zdrojů) nebo decentralizovaně. Jednotlivé varianty se liší podle nákladů, rychlosti reakce na události a rizika.

Měřitelnost (metriky)

Americký zakladatel moderního managementu Peter Drucker prohlásil: „*Když něco nemůžete změřit, nemůžete to ani řídit.*" S jeho výrokem lze jen souhlasit.

Identifikujeme, co lze měřit, stanovíme vhodné metriky a způsob jejich získávání a určíme, jaké hodnoty jsou optimální. Změříme a analyzujeme výsledky. Pokud jsou výsledky mimo přípustný interval, budeme reagovat.

3.4.3 Další nástroje a metodiky k řízení IT

IT Governance

IT Governance znamená řízení IT pomocí struktury vztahů a procesů tak, aby IT v maximální míře umožňovalo a podporovalo dosažení podnikatelských cílů. Přidanou hodnotou je redukce a řízení rizik nad procesy v IT. Hlavním cílem je návratnost investic (Čermák, 2010).

Obrázek 27 IT Governance *Zdroj: (Čermák, 2010)*

Strategický soulad se zaměřuje na soulad mezi IT a obchodem. Vzájemná komunikace a kooperace je potřebná pro společné řešení.

Dodávání hodnoty je proces, který hlídá jednotlivé činnosti tak, aby byla zajištěna hodnota projektu, která byla definovaná. Činnosti, které pomáhají zvednout hodnotu projektu, jsou podporovány, ostatní jsou odstraněny. IT by mělo přispívat ke snižování nákladů a zvyšování výnosů. Lze měřit například pomocí ROI.

Řízení rizik se musí provádět na všech úrovních organizace. Úkolem procesu je identifikovat rizika a vhodně na ně reagovat.

Řízení zdrojů znamená optimalizaci IT infrastruktury a znalostí s jejich efektivní využití.

Měření výkonnosti často využívá metody, pomocí které můžeme sledovat, jak IT přispívá k plnění podnikových cílů v různých oblastech. Opět zde platí pravidlo „co nejsme schopni měřit, nelze řídit".

ITIL

ITIL (Information Technology Infrastructure Library), jak popisuje (Bruckner, a další, 2012 str. 219), je sada knižních publikací, které popisují způsob řízení ICT služeb a ICT infrastruktury. Pro řízení IT služeb používají procesně orientovaný přístup. Každá aktivita v procesu musí přinášet přidanou hodnotu pro jejího uživatele. V rámci ITIL je klíčová orientace na kvalitu a spolehlivost IT služeb.

Hlavní myšlenkou implementace ITIL je snaha změnit IT oddělení na kvalitního poskytovatele služeb zákazníkům (i interním). Vyžaduje to změnu fungování oddělení a myšlení lidí. Neřeší ale konkrétní podobu organizační struktury ani obsah a podobu pracovních postupů. Říká pouze, které procesy implementovat, ale neříká jak.

Koncept vzniknul koncem 80. let v Anglii jako vládní zakázka pro vytvoření metodiky, jak řídit IT služby. ITIL ale nakonec není metodika, výsledkem jsou best practices (nejlepší zkušenosti) v daném oboru. Kritikou je nekonkrétnost, co přesně se ve kterých situacích má dělat. Aktuálně je ITIL ve verzi 3.

COBIT

COBIT (Control Objectives for Information and Related Technology), jak uvádí (Bruckner, a další, 2012 str. 219) je standard vyvinut pro správné postupy řízení, kontroly a auditu informačních technologií. Jedná se o metodiku určenou především výkonnému managementu. Implementuje myšlenky IT Governance.

Aktuální verzí je COBIT 5 z roku 2012. První verzi vydala organizace ISACA v roce 1996. Oproti ITIL je COBIT volně ke stažení. Dává do souvislosti IT zdroje, IT procesy a informační kritéria.

3.4.4 Metody řízení ICT projektů

Při realizaci změn, které mají daný začátek a konec a jsou unikátní, je vhodné využívat metodiky. Říká se, že není potřeba stále dokola vymýšlet kolo, když už ho někdo vymyslel (Obrázek 28 je toho důkazem). Proto jsou zde metodiky. Jsou to popsané kroky, jak řídit dané činnosti a jak zdárně dosáhnout cíle.

Obrázek 28 Příklad využití metodik *Zdroj: (Janiš, 2015)*

PMBOK

PMBOK (Project Management Body of Knowledge) je mezinárodně uznávaný standard řízení projektů, který vydává PMI (Project Management Institute). Mezi ostatními standardy a metodikami patří mezi nejstarší a nejobecnější. Aktuálně je nejnovější pátá verze, která definuje 47 procesů užitých v pěti procesních skupinách. Začíná přípravou projektu, pokračuje přes plánování projektu, provádění projektu, monitorování a kontrolu projektu po aktivity potřebné pro ukončení projektu nebo fáze. Dále definuje deset znalostních oblastí (MBI, 2013). Cílová skupina je široká, PMBOK je určen všem se zájmem o projektové řízení.

PRINCE2

Metodika PRINCE2 (Projects In Controlled Environments) je, jak uvádí (Máchal, a další, 2015 str. 84), produktově orientovaná metodika pro řízení IT projektů hojně využívaná v mezinárodním měřítku. Jedná se o metodiku pod správou britské vládní agentury OGC (Office of Government Commerce) podřízené ministru financí. Metodika nepokrývá oblast řízení lidí, plánovací techniky, techniky řízení rizik, apod. Navržení PRINCE2 je takové, aby byl aplikovatelný na jakýkoli typ projektu (i když byl původně určen jen pro IT).

3.4.5 Řízení architektury informačního systému

Enterprise Architektura

Enterprise architektura (EA) pracuje s mapami a referenčními modely (vrstvami). Popis vrstev a jejich použití v EA popisuje TOGAF.

TOGAF

TOGAF (The Open Group Architecture Framework) je nejpopulárnější metodika pro Enterprise Architekturu. Obsahuje sadu metod, doporučení pro EA (Hrušková, 2017). Pro implementaci TOGAF není potřeba žádný speciální software, přestože existují modelovací nástroje pro EA.

TOGAF klade důraz na propojení IT a business cílů. Je však velmi flexibilní, snaží se přizpůsobit do individuálního firemního prostředí a navázat se na již existující projektové a manažerské struktury.

3.5 Specifika při vývoji SW

3.5.1 Pojmy

Implementace

Implementací informačního systému se rozumí jeho zavedení do praxe.

Relační databáze

Databáze je kolekce vzájemně souvisejících souborů dat uložených společně. Relační databáze je potom databáze, která je založena na databázových tabulkách. V nich jsou tyto tabulky vzájemně provázané pomocí unikátních identifikátorů označovaných jako primární klíče a jsou určeny relace.

3.5.2 Architektura

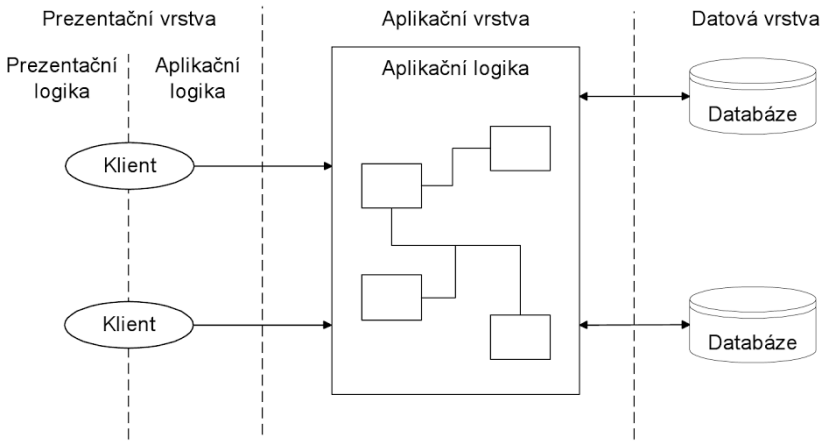
Architektura informačních systémů dává budování systému určitý směr, je to klíčový prvek řízení IS. Musí být názorná, srozumitelná a jednoduchá. Zároveň respektuje podnikové cíle a celkovou strategii podniku.

Třívrstvá architektura

Třívrstvá architektura, podle (Bollinger, a další, 2003 str. 231), je jeden z typů architektury informačních systémů. Jak napovídá název, rozděluje se na tři vrstvy. Jednotlivé vrstvy je možné vyvíjet, udržovat, případně měnit nezávisle na sobě, za předpokladu zpětně kompatibilního rozhraní mezi jednotlivými vrstvami.

- Prezentační – Jedná se o vrstvu, kterou vidí přímo uživatel, zajišťuje mu prezentaci údajů a umožňuje jeho interakci se systémem.
- Aplikační – Vrstva zajišťující zprostředkovávání operací mezi daty a prezentační vrstvou.
- Datová – Vrstva zajišťující práci s daty.

Obrázek 29 Třívrstvý model architektury *Zdroj: (Bollinger, a další, 2003 str. 232)*

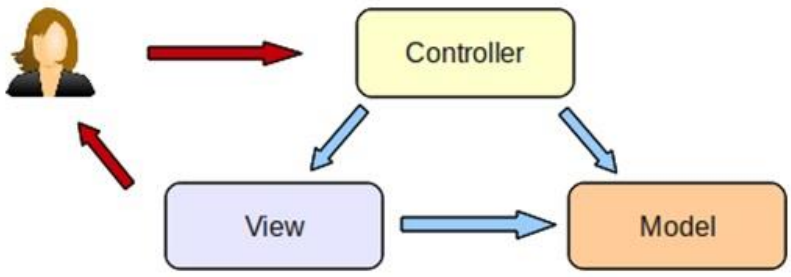


Výhodou třívrstvé architektury zobrazené na obrázku 29, je již zmíněná nezávislá správa jednotlivých vrstev, dále oddělení znalostí pro jejich správu, ale také oddělení kompetencí. To má výhodu v případě více dodavatelů, kdy každou vrstvu vytváří jiný dodavatel. Také může upravovat dle potřeby zabezpečení jednotlivých vrstev nezávisle na sobě.

Architektura MVC

MVC reprezentuje zkratku pro tři pilíře – Model, View, Controller. Architektonický vzor MVC, jak jej popisuje (Čada, 2009), odděluje doménový model aplikace od uživatelského rozhraní. Často bývá synonymem pro třívrstvou architekturu, ale není to pravda. Jak napovídá obrázek 29 a obrázek 30, MVC tvoří pomyslný trojúhelník, ale třívrstvá architektura má lineární podobu.

Obrázek 30 Model MVC *Zdroj: (Božek, 2012)*



Model představuje část, kde se provádí business logika – výpočty, vyhledávání, ukládání do databáze. Neřeší ale formu ukládaných dat (zda se jedná o relační databázi, xml soubor, apod). Model by měl být na zbytku aplikace natolik nezávislý, pokud jej přesuneme na jiný server, měli bychom jen s malým úsilím být schopni zachovat funkčnost a bezpečnost celé aplikace.

View představuje uživatelské rozhraní, pomocí kterého uživatel interaguje se systémem.

Controller se stará o předání informací z View do Modelu, který se postará o zpracování a následně aktualizuje View s novými informacemi.

REST

REST (Representational State Transfer) je architektonický styl zaměřený na webové aplikace. Neřeší ale vnitřní strukturu aplikace, pouze definuje přístup k rozhraní pro komunikaci s vnějším okolím. Jak uvádí (Sommerville, 2017), existují čtyři základní operace CRUD (Create – operace POST, Retrieve – operace GET, Update – operace PUT a Delete – operace DELETE):

POST – Odesílá data na server ke zpracování. Často se jedná o vytváření nového záznamu na základě získaného stavu.

GET – Operace, která získává data ze serveru a nesmí je měnit.

PUT – Operace nahrazující existující záznam, případně vytvářející nový, pokud původní neexistuje.

DELETE - Smaže záznam ze serveru.

3.5.3 Zabezpečení informačních systémů

Zabezpečení informačních systémů není jednoduchá záležitost. Základní pojmy a hrozny popisuje (Lapoš, a další, 2006). Existují hrozby informačních systémů, které můžou být úmyslné, neúmyslné, přírodní, případně technické:

- Úmyslné – přístup neoprávněných osob, které mohou poškodit zařízení a ovlivnit tím systém
- Neúmyslné – neúmyslný zásah uživatele do informačního systému
- Přírodní – Požár, blesk, apod.
- Technické – Výpadek elektrického napětí, nesprávné chování informačního systému, ztráta dat

Dalším rizikem je samotné zpřístupnění systému. Je nutné myslet na autorizaci uživatelů, zobrazit jim pouze příslušné akce, a také si ověřit, že mají do systému přístup. Ověření se obvykle provádí zadáním uživatelského jména a hesla. Ta lze v databázi ukládat mnoha způsoby.

Pojmy, které se v souvislosti se zabezpečením používají:

- Identifikace – každý uživatel je identifikován
- Autentizace – uživatel prokáže totožnost např. heslem
- Autorizace – uživateli jsou umožněny úkony podléhající jeho roli

Pro další pojmy v této podkapitole jsem využil třetí kapitoly (Programujeme vlastní sociální síť, 2016).

Zabezpečení pomocí session

Pokud máme webový informační systém, kde každému návštěvníkovi chceme zobrazit pouze přihlašovací formulář a pouze autentizovaným uživatelům vše ostatní, je vhodné použít session. To je bezestavová informace, kde každému uživateli je vygenerováno unikátní číslo, podle kterého jej můžeme na daném webu identifikovat.

Ukládání hesel

Mezi metody ukládání můžeme volit prostý text nebo hashovací funkce (Boswell, 2012).

Prostý text

V této formě jsou hesla uložena tak, jak je napíšeme. Neobsahují žádné zabezpečení. Kdokoliv se dostane k databázi, kde jsou daná hesla uložená, je může zneužít.

Hash

Ukládání hesla v podobě hashe – otisku je vhodná varianta. Hash je výsledkem matematické operace. Výhodou je, že je stejný pro stejná vstupní data. Zároveň má konstantní délku. Mezi další výhody patří téměř nemožnost najít identický hash pro různá vstupní data. Drobná změna ve vstupních datech vyvolá velkou změnu na výstupu. Z hashe je prakticky nemožné rekonstruovat původní text.

Ověřování hesel probíhá tak, že z hesla zadaného uživatelem, se vytvoří hash, který se porovná s údajem v databázi.

Hashovací funkce – MD5, SHA-1, SHA-2

Jsou to algoritmy, které jsou určeny pro rychlý výpočet hashe v u velkých vstupních dat. Řádově to jsou desítky sekund u hesel o 6-8 znacích.

Salt

Málokterý uživatel je schopný si zapamatovat delší heslo, nejlépe pro každý server unikátní. Zároveň však platí, že čím je heslo delší, tím je složitější jej rozlousknout. Proto se sahá k tzv. metodě solení (salt). Ke kratším heslům přidáváme na serveru dlouhý řetězec znaků. Hash se tedy nepočítá z menšího počtu znaků, ale z několika desítek znaků (Malý, 2011).

Salt ale není všemocný. Pokud se útočník dostane do databáze, pravděpodobně získá i řetěz používaný pro solení.

Bcrypt

Bcrypt používá kromě hesla a salt řetězce parametr „cost", kterým je možné ovlivnit náročnost výpočtu hashe. Tím lze ovlivnit rychlost zpracování (Jak na webové stránky, 2016). Bcrypt je hashovací funkce, která je mnohonásobně pomalejší, než funkce předchozí. Přesto to nevadí. Proč? Uživatel nepozná, jestli se jeho heslo hashovalo 100 ms nebo méně. U větších souborů už by to ale mohl být problém. Další výhodou tohoto pomalého přístupu je, že odrazuje hackery při použití útoku hrubou silou. Jejich vynaložené úsilí a výpočetní výkon je mnohonásobně větší. Principem fungování Bcrypt je několikanásobné interní vykonání hashovací funkce. Bcrypt používá například Twitter, Drobox, Mall.cz a Slevomat.

Zálohování dat

Data můžeme zálohovat kompletní, tedy vždy zálohujeme všechno. Nevýhodou je náročnost na datový prostor. Další variantou je inkrementální záloha, kde zálohujeme vždy jen přírůstek od poslední zálohy. K obnovení tedy potřebujeme celý řetězec inkrementálních záloh. Poslední variantou je diferenční zálohování. To také nezabírá velký datový prostor, zaznamenává totiž změny od poslední kompletní zálohy.
Přestože existují hashovací funkce, neznamená to, že nemusíme používat silná hesla. Naopak! Výpočetní výkon počítačů se neustále vylepšuje, občas se objeví zpráva o zjištěné mezeře v zabezpečení, apod.

3.5.4 UX Design

UX, neboli user experience, neboli uživatelský prožitek zastřešuje všechny aspekty interakce koncového uživatele se službou, či produktem. Design není snadné definovat, pokud se ale budeme držet kontextu UX a návrhu webových aplikací, lze využít definice Kathryn Best: *„Design popisuje proces tvoření produktu a také výsledný produkt, tohoto procesu"* (Best, 2006 str. 6).

Spojení UX a designu, vznikne nový pojem User Experience Design (UXD). Jedná se o komplexní systém, který chápe uživatele, prostředí, ve kterém se pohybuje a služby, které využívá nebo chce využívat. Cílem je spokojený zákazník. Jak ukazuje obrázek 31Obrázek 31, UXD je komplexní proces, který se týká nejen grafické podoby systému, ale přístupnosti, ergonomie a dalších oblastí. Zahrnuje také HCI (Human Computer Interaction). Jedná se oblast, která se zabývá studiem lidí a jejich prací s počítačem, jak je uzpůsobena vzájemná komunikace.

Obrázek 31 User Experience *Zdroj: (Gube, 2010)*

Zásady návrhu:

- Analýza požadavků a získání zpětné vazby od koncového uživatele
- Návrh UX
- Realizace prototypu a testování s uživateli

3.5.5 Wireframes

Wireframes jsou prototypy uživatelského rozhraní. Představují první náhled výsledného produktu. Usnadňují tak komunikaci se zákazníkem a lidmi v týmu. Každý si ujasní představu, o čem je řeč. Lze využít tužku a papír pro rychlý náčrt nebo sofistikované programy jako je Axure nebo Balsamiq.

3.5.6 Technologie pro tvorbu webové aplikace

Co je to Framework a proč jej využít

Zdrojové kódy většiny aplikací řeší podobné problémy, proto se často vytvoří sady funkcí nebo tříd, které problémy řeší a lze je využít i na jiných projektech. Tyto sady kódů se nazývají frameworky (Mordánky, 2011). Výhodou použití je zefektivnění vývojářské práce. Nemusíme „znovu objevit kolo", ale naopak využijeme vyzkoušené vlastnosti, které ušetří čas.

Při výběru vhodného frameworku musíme vzít v úvahu jednak programovací jazyk, který známe a chceme využívat. Dále je to aktuálnost, výhodou je velká základna uživatelů a tím pádem i diskuzní fóra s mnoha řešenými problémy. Zároveň frameworky jsou různě náročné na výkon, kdy načítají do paměti části, které nikdy nevyužijí, a tím dojde ke zpomalení.

Kam s daty

Každý informační systém pracuje s daty a ta je potřeba někam ukládat. Obvykle je to databázový server, na kterém je umístěna databáze. Vybírat můžeme podle relačních databází, kde se může jednat o MySQL, Oracle Database nebo například PostgreSQL. Nebo můžeme využít NoSQL databáze, kde jsou představiteli MongoDB a například Cassandra. Tyto NoSQL databáze nepoužívají tabulková schémata jako relační databáze, jejich využití v dnešní době získává velkou podporu v oblasti Big–Data. Ve zpracování diplomové práce jsem používal relační databázi PostgreSQL.

Jak uvádí (Gála, a další, 2015 str. 56), pro modelování struktury databáze slouží ER (Entity Relationship) diagram. Výstupem je popis logické struktury databáze, který obsahuje entity (objekty), atributy (prvky, které charakterizují entity), dále relace (určují vztah mezi dvěma entitami) a kardinality vztahu (v jakém vztahu jsou entity, např. „1:1, 1:N, M:N").

3.5.7 Testování

Testování je nedílnou součástí životního cyklu vývoje software. Existuje několik typů testování, které popisuji níže. Vycházel jsem z páté kapitoly (Roudenský, a další, 2017).

Funkční testy – FAT

Funkční testy (Factory Acceptance Tests – FAT) jsou testy prováděné na straně dodavatele. Testuje se správné plnění všech úkolů podle požadavků zákazníka.

Integrační testování

Integrační testování spočívá v testování komunikace mezi jednotlivými komponentami uvnitř aplikace, případně komunikaci s operačním systémem nebo hardwarem. Například chceme otestovat, zda se správně dotazujeme do databáze (tím pádem k testům potřebujeme databázi). Ověřujeme také, jestli se metody volají ve správném pořadí. Integrační testy zpravidla nepřipravuje programátor, ale tým testerů. Testy mohou být manuální i automatizované.

End to end testy

Jejich principem je testování kompletních procesů a životních cyklů produktu. Výsledkem testů je nejen znalost kvality jednotlivých součástí, ale i ověření, zda jsou podporovány potřeby uživatele. Provedení tohoto testu se obvykle provádí až na závěr všech testů, zahrnuje jak samotný systém, tak i sub-systémy. Pokud některý z nich selže, může selhat i samotný systém. Pomocí tohoto testování tomu můžeme předejít.

Unit testy

Unit testy, případně testování jednotek, znamenají testování malých jednotek funkcionality. Často se testuje jedna metoda nebo třída.

Akceptační testování – UAT

Akceptační testy (User Acceptance Test – UAT) jsou akceptační testy na straně zákazníka. Ten si se svými testery obvykle provede akceptační testy, které jsou připraveny vzájemnou spoluprací s dodavatelem.

Progresní a regresní testy

Progresní testy se využívají při testování nových vlastností aplikace, případně kontrole nových funkcí.

Regresní testy se využívají při opětovném testování funkcí a vlastností aplikace. Smyslem je ověření, že provedené změny v aplikaci nemají vliv na stávající funkce.

Výkonnostní testování

Výkonnostní testování se snaží simulovat přístup mnoha uživatelů k systému zároveň a zkoumá dobu odezvy a chování systému.

PRAKTICKÁ ČÁST

V praktické části diplomové práce jsem se zaměřil na analýzu požadovaného informačního systému. Analýza obsahuje popis současné situace, na kterou navazuje návrh systému a celou kapitolu uzavírá část implementace a testování. Zároveň jsou zde zmíněna rizika celého projektu a také možnosti dalšího vylepšení systému.

4 Analýza a návrh software

Cílem business analýzy je porozumění hlavním problémům zadavatele a nalezení možností, jak tyto problémy vyřešit a splnit jeho vizi. Úkolem business analýzy je popsat a namodelovat realitu jednoduše, aby to bylo pro každého pochopitelné. Existují různé pohledy na danou věc a analýza toto musí reflektovat. Informace musejí být správné, aktuální, úplné a relevantní.

Při zpracování praktické části jsem využíval nástrojů Enterprise Architect pro tvorbu modelů, dále nástroj Intellij IDEA pro tvorbu kódu a nástroj pgAdmin pro vytvoření databáze. Pro správu projektu jsem vytvořil repozitář na portálu GitHub. Pro mapování projektu jsem použil nástroj Trello a vytvořenou roadmapu projektu.

4.1 Zadání

Zadáním bylo analyzovat a navrhnout informační systém pro vnitřní potřeby Masarykova ústavu vyšších studií ČVUT v Praze. Nový informační systém musí být inovační. Musí zamezit co největšímu plýtvání. A to jak nadbytečné práce, chybovosti, ale i nákladů. Zároveň implementovaný systém musí při zpracování uspořit čas všem jeho účastníkům. Důležitým faktorem je dostupnost v aktuální verzi všem uživatelům.

Informační systém povede evidenci předmětů, vyučujících, ale také závěrečných prací a výjezdů do zahraničí. Zároveň umožní evidenci publikací. Tyto veškeré vstupy se podle vnitřního předpisu MUVS využijí pro výpočet osobního ohodnocení zaměstnanců.

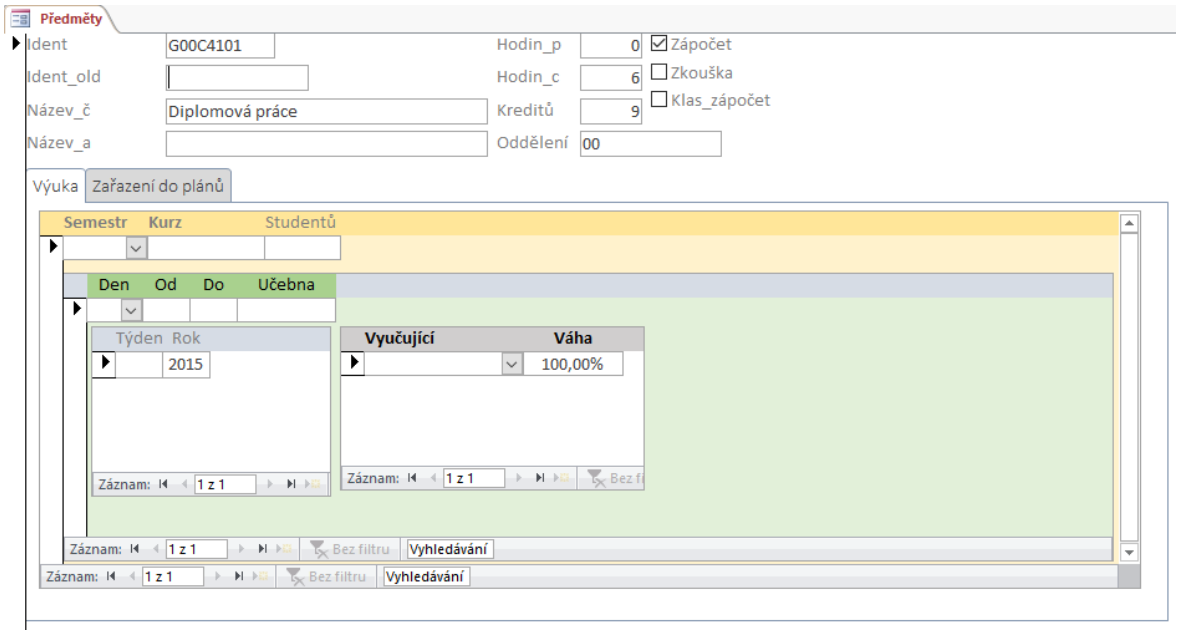
Požadavkem je přístupnost systému pro všechny zaměstnance, kteří k němu dostanou přístup. Podle přístupových rolí jim bude následně umožněna daná funkcionalita systému.

4.2 Současný stav

Analytická část začíná popisem současného stavu. Současný stav zahrnuje mnoho vstupů, které nemají jednotnou podobu. Tím dochází k velkému zpožďování v předávání informací. Data jsou předávána emailem, či tabulkovými soubory. Informace o aktivitách na MUVS jsou rozprostřené na mnoho míst, které spravují různí lidé. Dochází ke ztrátám, protože ne vždy se konkrétní informace dostane včas k uživateli. Zároveň neexistuje jednotný přehled kde, případě kolikrát byl daný pracovník ústavu na pracovní cestě. Tento prvek je také součástí osobního ohodnocení. Cílem je eliminace těchto jevů a použití jednotného systému pro evidenci oborů, aktuálně vyučovaných předmětů a vyučujících.

V současnosti se pro tvorbu studijních plánů a evidenci předmětů používá aplikace vytvořená v MS Access, která ukládá data do své interní databáze. Příklad formuláře v této aplikaci je uveden na obrázku 32.

Obrázek 32 Formulář v aplikaci MS Access *Zdroj: aplikace MÚVS (Švecová, 2017)*



Výhodou tohoto řešení je uživatelská přívětivost při tvorbě formulářů, výstupů, apod. Tvůrce nemusí být znalý programování, MS Access nabízí při vytváření průvodce jednotlivými kroky.

Řešení má ale nevýhody. Nikdo, kromě uživatele, který má tuto aplikaci na svém počítači, se k ní nedostane (i když to MS Access umožňuje, není to implementováno). Nemůže si tedy například ověřit správně zadaná data. Bez programování nelze vytvářet složité produkty, zajistit zabezpečení, apod.

Současný systém neobsahuje informace o publikacích, vedení závěrečných prací, výjezdech na služební cesty, ale také o roli vyučujícího v jednotlivých předmětech.
Zde by se dalo uvažovat o možnosti nasazení databáze na databázový server a přistupovat k ní vzdáleně. Tím by se muselo přenastavit připojení k databázi v aplikaci MS Access. Problémem však je, jak dostat aplikaci v aktuální verzi ke všem uživatelům, kterých může být více než 100. Výrazně by tím klesla uživatelská přívětivost, protože by při potřebě využít tento systém, musel vždy uživatel nejprve stáhnout aktuální verzi programu.
Tato výše popsaná možnost je ale nevyhovující, a proto jsem po konzultaci s vedoucí práce rozhodl o vytvoření webové aplikace.

4.3 Hodnocení výkonnosti zaměstnanců formou KPI

Hodnocení výkonnosti zaměstnanců na Masarykově ústavu vyšších studií upravuje (Švecová, 2016), který má platnost od 1. 3. 2016. KPI (Key Performance Indicators) jsou klíčové ukazatele výkonnosti. Mohou být finanční i nefinanční a používají se při kvantifikaci cílů pro vyjádření strategického výkonu organizace, která pomáhá organizaci definovat cíle a měřit průběh jejich plnění.

Při sledování výkonnosti pracovníků a jejich aktivit můžeme využít ukazatele s časovým nebo finančním hlediskem, případně například lidské zdroje pro určení optimálních personálních kapacit.

Ukazatele se týkají dotčených osob, kterými jsou:

- Vědecko – pedagogičtí pracovníci
    - Profesoři (P) – Předpokladem je jmenování prezidentem ČR
    - Docenti (D) – Předpokladem je udělený titul docent
    - Odborní asistenti (O) – Předpokladem je titul Ph.D. nebo jeho ekvivalent
    - Asistenti (A) – Předpokladem je vědecko-pedagogický růst a následné dosažení titulu Ph.D.

- Pedagogičtí pracovníci – Těmi jsou odborní asistenti jazykové výuky (J) a lektoři (L).
- Vědečtí pracovníci – Osoby související přímo s konkrétními výzkumnými projekty a jejich výstupy. Dokument se jimi nezabývá.

Předpis zároveň definuje hodnocení nových zaměstnanců, částečné úvazky a také změny zařazení zaměstnanců.

Principy hodnocení

Jak uvádí zmíněný předpis, hodnocení zaměstnanců se provádí v pravidelných semestrálních intervalech, kdy se vždy posuzují výsledky uplynulých dvanácti měsíců, tedy klouzavě období březen až únor, resp. září až srpen.

Kritéria hodnocení

- Kategorie A – Oblast, která se přímo týká pedagogické činnosti. Obsahuje výuku předmětů, dále zavedení nového předmětu, vedení kvalifikační práce, případně tvorbu učebních pomůcek nebo zajištění zkoušek.
- Kategorie B – Oblast, která obsahuje jednak vědeckou činnost, tvůrčí činnost a publikační činnost.
- Kategorie C – Oblast týkající se internacionalizace. Jedná se o zahraniční výjezd, mezinárodní projekty, zahraniční letní školy, případně výuku odborného cizího jazyka.
- Kategorie D – Je to oblast, která se zabývá vedoucí funkcí podle organizačního řád (člen vedení ústavu, vedoucí oddělení, garant oboru).

Jednotlivé kategorie se člení podle rozdělení akademických pracovníků. Výpočty v této diplomové práci neuvádím, protože v době psaní této práce procházejí úpravou a zjednodušením.

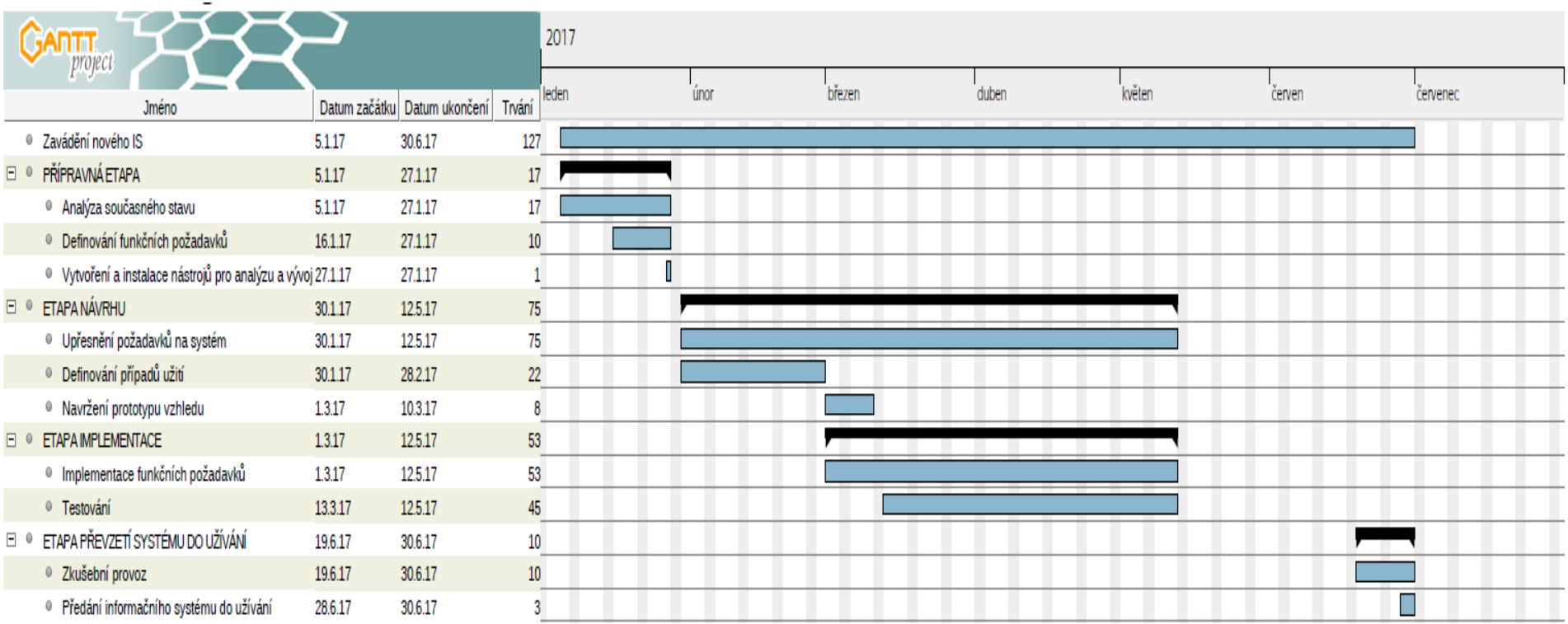
4.4 Analýza informačního systému

Při zpracování analýzy jsem si vytvořil časový plán projektu a analyzoval jsem jednotlivé požadavky. Na základě požadavků jsem vytvořil případy užití, na které navazuji navržením struktury aplikace v kapitole 4.5.

Časový plán projektu

V počátku projektu bylo potřeba stanovit časový plán jednotlivých etap. Etapy vycházejí z práce jednoho člověka (chybí týmová spolupráce). Obrázek 33 zobrazuje Ganttův diagram.

Obrázek 33 Ganttův diagram projektu *Zdroj: vlastní zpracování*

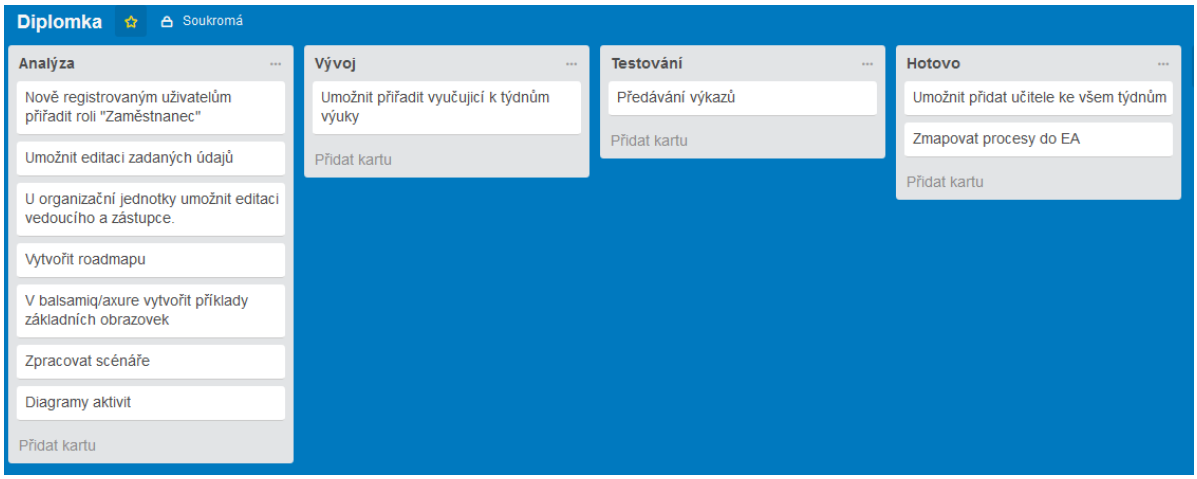


Jméno	Datum začátku	Datum ukončení	Trvání
Zavádění nového IS	5.1.17	30.6.17	127
PŘÍPRAVNÁ ETAPA	5.1.17	27.1.17	17
Analýza současného stavu	5.1.17	27.1.17	17
Definování funkčních požadavků	16.1.17	27.1.17	10
Vytvoření a instalace nástrojů pro analýzu a vývoj	27.1.17	27.1.17	1
ETAPA NÁVRHU	30.1.17	12.5.17	75
Upřesnění požadavků na systém	30.1.17	12.5.17	75
Definování případů užití	30.1.17	28.2.17	22
Navržení prototypu vzhledu	1.3.17	10.3.17	8
ETAPA IMPLEMENTACE	1.3.17	12.5.17	53
Implementace funkčních požadavků	1.3.17	12.5.17	53
Testování	13.3.17	12.5.17	45
ETAPA PŘEVZETÍ SYSTÉMU DO UŽÍVÁNÍ	19.6.17	30.6.17	10
Zkušební provoz	19.6.17	30.6.17	10
Předání informačního systému do užívání	28.6.17	30.6.17	3

Sledování stavu projektu

Pro správu projektu jsem využil nástroj Trello (viz Obrázek 34). Jedná se o systém nástěnek, seznamů a karet pro jednoduchý a rychlý přehled. Zároveň je nástroj pro jednotlivce zdarma. Grafické zpracování vychází z metody Kanban, kde jednotlivé položky (Kanban karty) se přesouvají mezi jednotlivými fázemi (sloupci).

Obrázek 34 Nástroj Trello *Zdroj: vlastní zpracování*



4.4.1 Analýza požadavků

Pro získávání požadavků jsem použil metodu konzultací s doc. Švecovou, ředitelkou MÚVS. Poptávaný software je realizován jako projekt, jehož hlavním výstupem je funkční řešení splňující uvedené základní požadavky.

Funkční požadavky

Funkční požadavky, jak bylo uvedeno v kapitole 3.2.7, reprezentují, co má být obsahem systému. Evidencí je myšleno přidávání, editace a mazání údajů. Obrázek 35 zobrazuje funkční požadavky.

Obrázek 35 Funkční požadavky *Zdroj: vlastní zpracování*

Nefunkční požadavky na systém

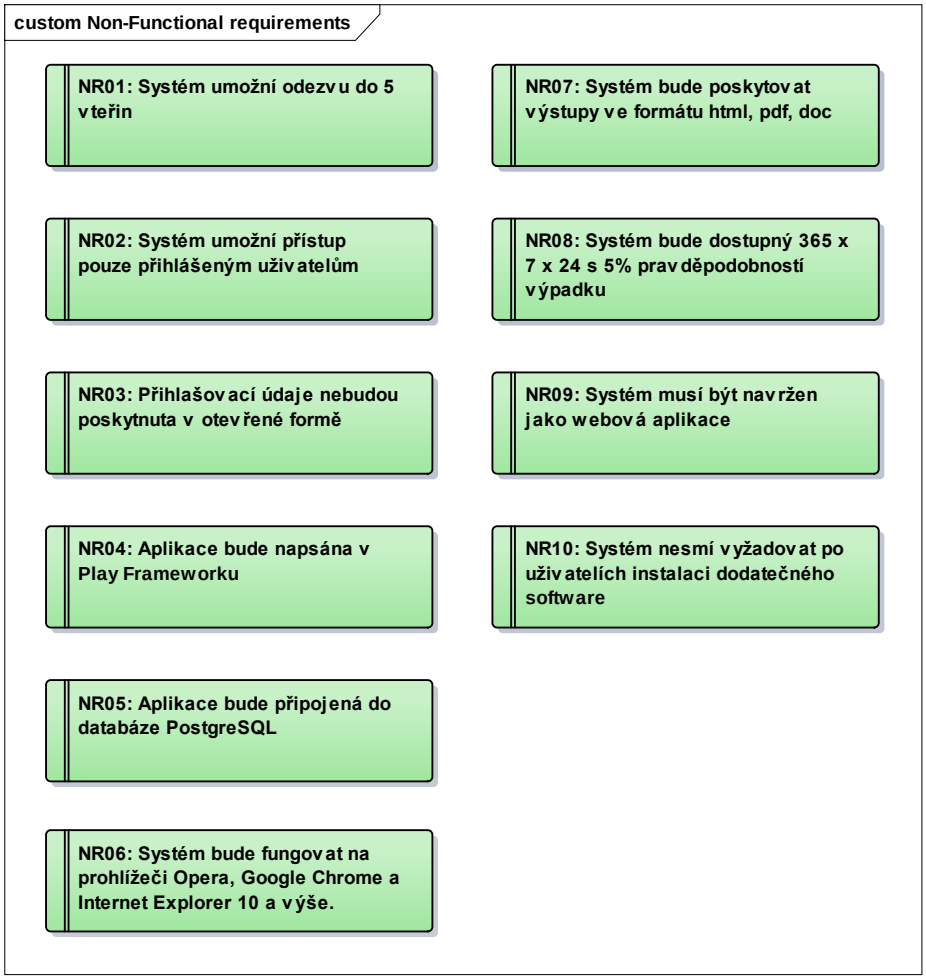
Systém musí být navržen jako webová aplikace s přístupem pro všechny uživatele z internetu.

Systém nesmí vyžadovat instalací dodatečného software.

Mezi další nefunkční požadavky patří bezpečnost systému. Nebude umožněn vstup uživatelům, kteří neznají uživatelské jméno a heslo. Uživatelským jménem je e-mail, který je unikátní. Každý uživatel má správcem systému (administrátorem) předem definované role, které vymezují oprávnění k jednotlivým krokům. Základním oprávněním je uživatelská role, která umožňuje prohlížet si své údaje a celkové studijní plány. Pro roli zadavatele a další role je nutné kontaktovat administrátora.

Odezva a dostupnost systému závisí na podmínkách Play Frameworku a nastavení serveru. Od serveru požadujeme dostupnost 365 x 7 x 24 s 5 % pravděpodobnosti výpadku (tedy kdy systém může být nedostupný). Tyto výpadky můžou být způsobeny údržbou serverů, případně neočekávanými technickými problémy s infrastrukturou. Obrázek 36 zobrazuje strukturu.

Obrázek 36 Nefunkční požadavky *Zdroj: vlastní zpracování*



Očekávané přínosy

Při splnění základních požadavků přinese implementace pozitiva ve formě snazší evidence zaměstnanců, vyučovaných předmětů a vytvořených studijních plánů. Zároveň to urychlí provádění změn, kdy jsou data centrálně uložena. Standardizuje se zadávání do systému.

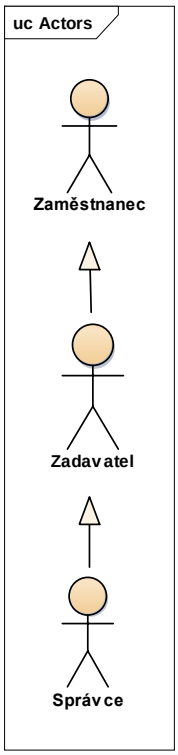
4.4.2 Případy užití a aktéři systému

Jak popisuje kapitola 3.2.4, diagram případu užití vizualizuje funkční požadavky na systém. Tyto požadavky jsem rozpracoval do logického celku, jak zobrazuje obrázek 38. Každý případ užití má svého aktéra, které jsem popsal níže.

Aktéři systému

Systém bude obsahovat tři aktéry systému, které zároveň definují role pro přístup do systému. Obrázek 37 zobrazuje strukturu.

Obrázek 37 Aktéři systému *Zdroj: vlastní zpracování*



Zaměstnanec – role, které po přihlášení může na základě oprávnění si prohlížet své údaje, měnit je a také si prohlížet výstupy určené jeho osobě. Jedná se o základní roli.
Zadavatel – role, která umožňuje zadávat předměty do jednotlivých semestrů a studijních plánů, nemůže ale editovat údaje zaměstnanců.

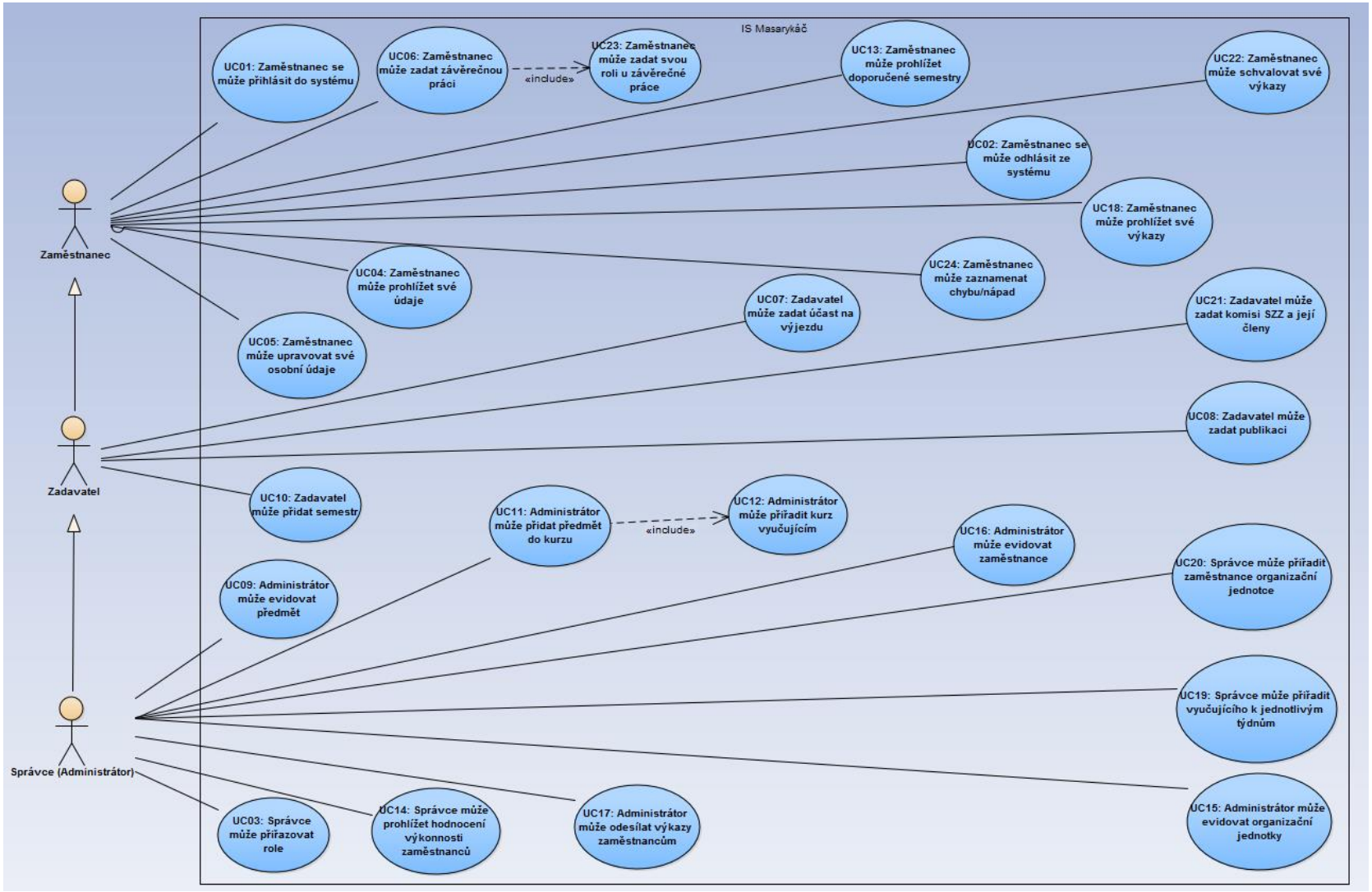
Správce (Administrátor) – role, která má přístup ke všem funkcím v systému a má povolené veškeré operace. Může přidávat nové uživatele a přiřazovat jim role. Může měnit údaje stávajících uživatelů. Mezi jeho další kompetence patří tvorba a editace organizačních jednotek.

Případy užití

Cílem modelu případů užití není detailně popsat veškeré činnosti, ale pouze ty hlavní. V tomto případě se jedná o zadávání uživatelů a editaci dat. Evidenci činností zaměstnanců a prohlížení výstupů a schvalovací proces výkazů. Na obrázku 38 je model případu užití pro tuto analýzu.

Na obrázku 39 je zobrazena matice vztahů pro případy užití a funkční požadavky. Matice dokazuje, že každý případ užití je vztažen k alespoň jednomu funkčnímu požadavku a nebyl opomenut.

Obrázek 38 Model případů užití *Zdroj: vlastní zpracování*



Obrázek 39 Vztahová matice případů užití a funkčních požadavků *Zdroj: vlastní zpracování*

Source \ Target	FR::FR01: Systém bude evidovat předměty	FR::FR02: Systém bude evidovat zaměstnance	FR::FR03a: Systém umožní přiřadit vyučujícího	FR::FR03b: Systém umožní přiřadit vyučujícího k	FR::FR04: Systém umožní evidenci publikací za	FR::FR05: Systém umožní evidenci závěrečných	FR::FR06: Systém umožní vedoucímu registraci	FR::FR07: Systém umožní vedoucímu přiřazovat	FR::FR08: Systém umožní evidenci komisí státní	FR::FR09: Systém bude poskytovat výstupy stud	FR::FR10: Systém bude poskytovat výstupy pro j	FR::FR11: Systém umožní automaticky vygenero	FR::FR12: Systém umožní přihlášení do systému	FR::FR13: Systém bude evidovat organizační je	FR::FR14: Systém umožní přiřadit zaměstnance	FR::FR15: Systém umožní informovat zaměstnar	FR::FR16: Systém umožní přihlášenému uživate	FR::FR17: Systém umožní přihlášenému uživate	FR::FR18: Systém umožní uživateli změnit si hes	FR::FR19: Systém umožní evidenci místností	FR::FR20: Systém umožní evidenci komisí SZZ a	FR::FR21: Systém umožní evidenci chyb/návrhů	FR::FR22: Systém umožní evidenci výjezdů	FR::FR23: Systém umožní přihlášenému uživate	FR::FR24: Systém umožní u přidávání předmětu	FR::FR25: Systém umožní evidenci výkazů zamě	FR::FR26: Systém umožní schvalování výkazů	FR::FR27: Systém umožní evidenci semestrů
Use Cases::UC01: Zaměstnanec se může přihlásit do systému													↑																
Use Cases::UC02: Zaměstnanec se může odhlásit ze systému																									↑				
Use Cases::UC03: Správce může přiřazovat role								↑																					
Use Cases::UC04: Zaměstnanec může prohlížet své údaje											↑							↑											
Use Cases::UC05: Zaměstnanec může upravovat své osobní údaje																			↑	↑									
Use Cases::UC06: Zaměstnanec může zadat závěrečnou práci						↑																							
Use Cases::UC07: Zadavatel může zadat účast na výjezdu																									↑				
Use Cases::UC08: Zadavatel může zadat publikaci					↑																								
Use Cases::UC09: Administrátor může evidovat předmět	↑																												
Use Cases::UC10: Zadavatel může přidat semestr																													↑
Use Cases::UC11: Administrátor může přidat předmět do kurzu																						↑					↑		
Use Cases::UC12: Administrátor může přiřadit kurz vyučujícím			↑																										
Use Cases::UC13: Zaměstnanec může prohlížet doporučené semestry										↑																			
Use Cases::UC14: Správce může prohlížet hodnocení výkonnosti zaměstnanců																												↑	
Use Cases::UC15: Administrátor může evidovat organizační jednotky														↑															
Use Cases::UC16: Administrátor může evidovat zaměstnance		↑					↑																						
Use Cases::UC17: Administrátor může odesílat výkazy zaměstnancům																												↑	↑
Use Cases::UC18: Zaměstnanec může prohlížet své výkazy																	↑												
Use Cases::UC19: Správce může přiřadit vyučujícího k jednotlivým týdnům				↑								↑																	
Use Cases::UC20: Správce může přiřadit zaměstnance organizační jednotce															↑														
Use Cases::UC21: Zadavatel může zadat komisi SZZ a její členy									↑														↑						
Use Cases::UC22: Zaměstnanec může schvalovat své výkazy																												↑	↑
Use Cases::UC23: Zaměstnanec může zadat svou roli u závěrečné práce						↑																							
Use Cases::UC24: Zaměstnanec může zaznamenat chybu/nápad																								↑					

4.5 Návrh informačního systému

Po analýze navazuje návrh systému. Navrhovaný systém musí být bezpečný, popisuji tedy jeho zabezpečení. Následně navazují jednotlivé oblasti, kde navrhuji tok aktivit a událostí v procesech. Poté jsem vytvořil prototypové obrazovky, tzv. wireframy. Pro ujasnění logické struktury tříd následuje jejich diagram s krátkým popisem těch nejdůležitějších. Jejich implementaci popisuje následující kapitola.

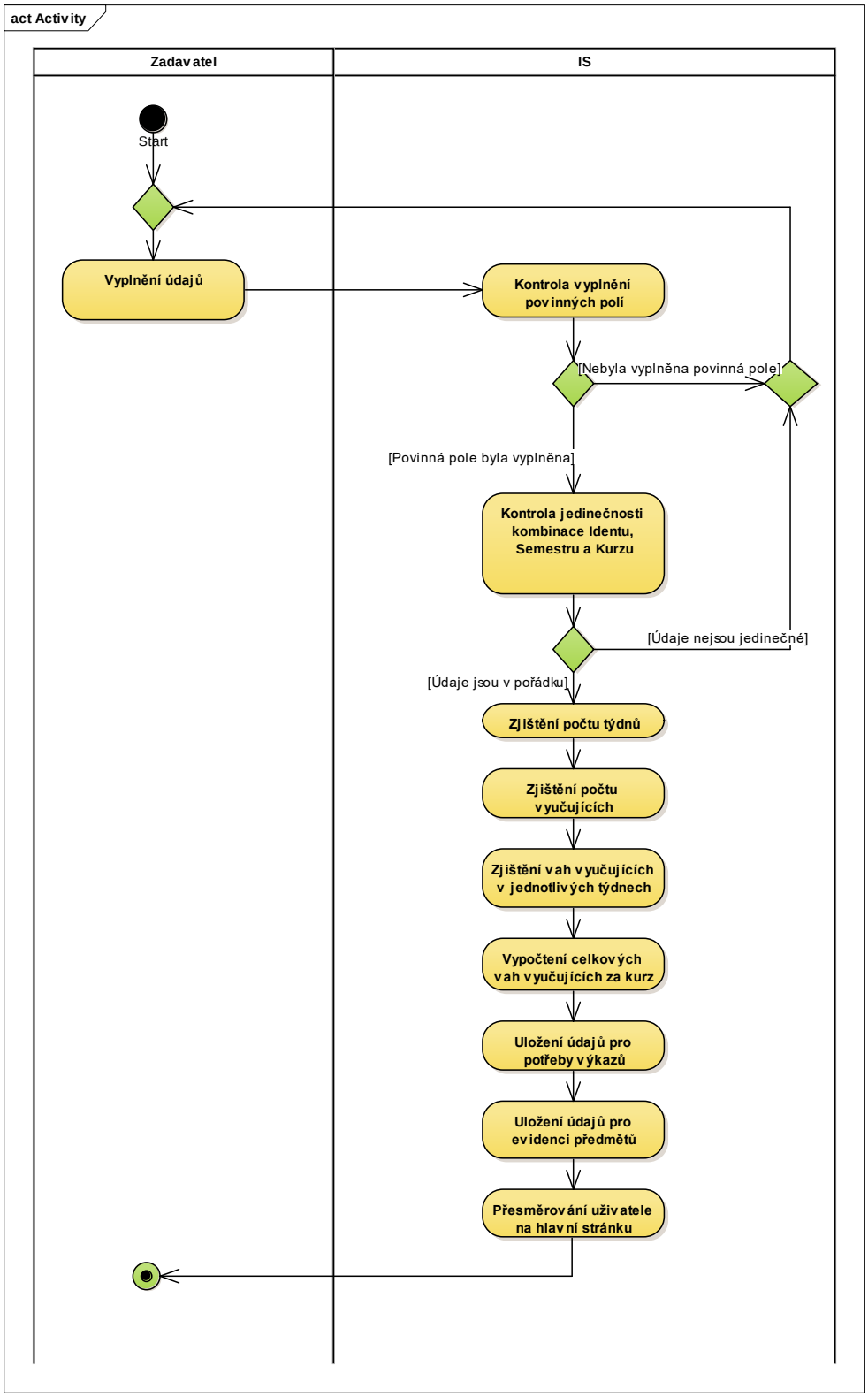
4.5.1 Navržení zabezpečení systému

Zabezpečení na straně klienta se skládá ze tří částí. První část požaduje, aby každý uživatel se přihlásil (autentizoval) zadáním uživatelského jména a hesla. Hesla jsou ukládána pomocí Bcryptu (viz kapitola 3.5.3). Zadané údaje jsou předány na server, který je ověří a v případě úspěchu je klientovi předán jedinečný identifikátor, kterým se v systému identifikuje (uživatel ale tento identifikátor nevidí). Pokud tedy nepřihlášený uživatel se pokusí dostat do zabezpečené části aplikace, je přesměrovaný vždy na přihlašovací stránku.
Dále má každý uživatel definovaná přístupová práva, která mu umožní operace se systémem. Jsou mu zobrazeny pouze ty aktivity, ke kterým má práva. Další částí je zabezpečení stránek tak, aby uživatel bez příslušného oprávnění na ně nemohl vstoupit, když by zadal adresu ručně do prohlížeče.

4.5.2 Zadávání předmětů

Při zadávání předmětů, kterým přiřazujeme vyučující, studijní plány a další položky, dochází na pozadí celého procesu ke kontrole již zadaných údajů. U zadávání předmětů do kurzů a zadání vyučujících, dochází ke kontrole kombinace identu (identifikátor předmětu), semestru a názvu kurzu. Proces je zachycen na obrázku 40.

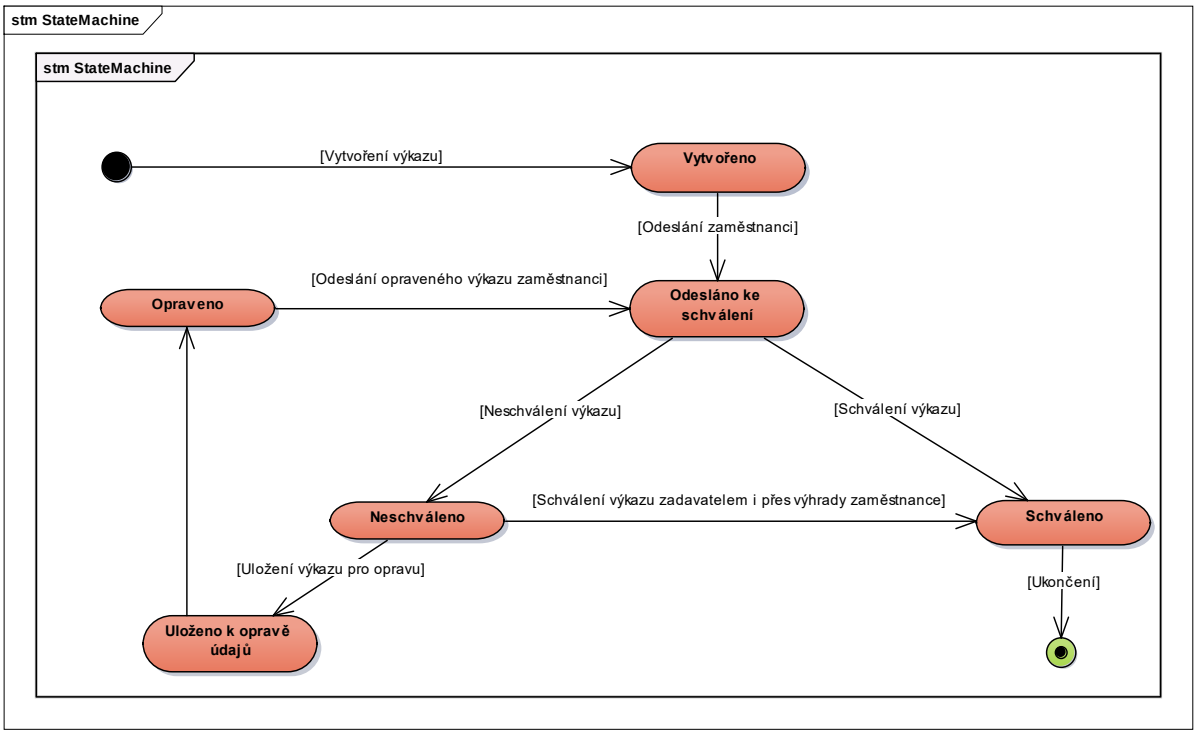
Obrázek 40 Proces zadávání předmětů *Zdroj: vlastní zpracování*



4.5.3 Tvorba výkazů a jejich vystavování

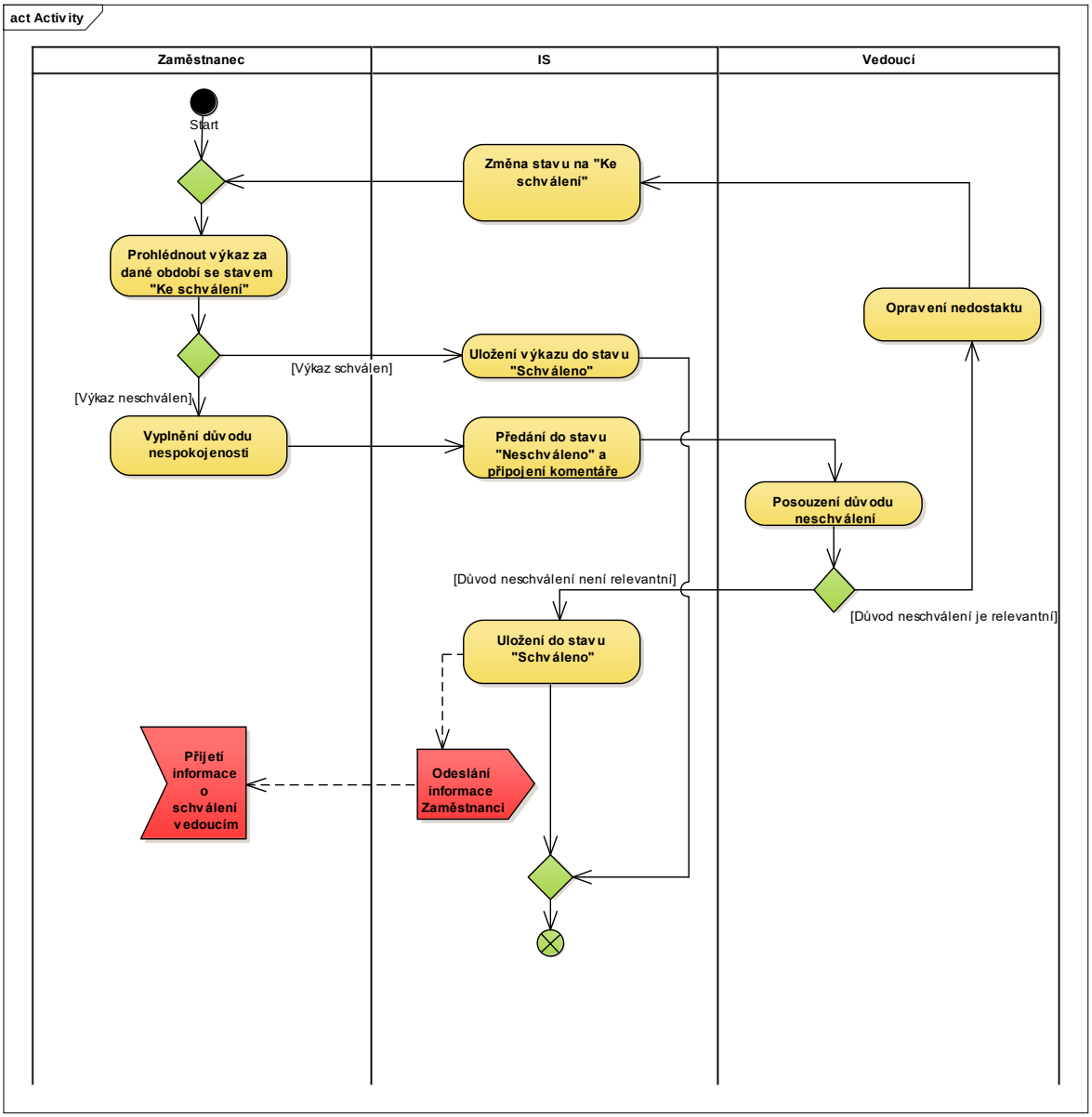
Vedoucí pracovník má možnost provádět k určenému datu výkazy zaměstnanců, které obsahují jejich aktivitu. Stavy výkazů a přechody mezi stavy ukazuje obrázek 41, na kterém je stavový diagram.

Obrázek 41 Stavový diagram výkazů *Zdroj: vlastní zpracování*



Proces schvalování zobrazuje obrázek 42. Systém umožňuje na jednom místě zobrazit tyto údaje a odeslat je zaměstnanci. Tomu se po přihlášení na jeho ploše zobrazí notifikace o novém výkazu, který je potřeba zkontrolovat. Při kontrole může být vše v pořádku a zaměstnanec tuto skutečnost u každé oblasti vyznačí. V případě problému u dané oblasti vyznačí nesouhlas a do textového pole napíše důvod nesouhlasu a odešle zpět vedoucímu pracovníkovi. Jeho úkolem je zkontrolovat výhrady a rozhodnout, zda jsou opodstatněné. Oprávněné nedostatky musí odstranit (například opravit překlepy) a takto opravený výkaz odešle znovu zaměstnanci pro kontrolu. A celý proces schvalování probíhá dokola, dokud není výkaz schválen.

Obrázek 42 Proces schvalování výkazů *Zdroj: vlastní zpracování*



Organizační jednotky

Každý vyučující je zařazený do organizační jednotky (OJ). Ta představuje úsek ve struktuře Masarykova ústavu vyšších studií ČVUT v Praze. Každá OJ má definovaný název, svého vedoucího a zástupce vedoucího.

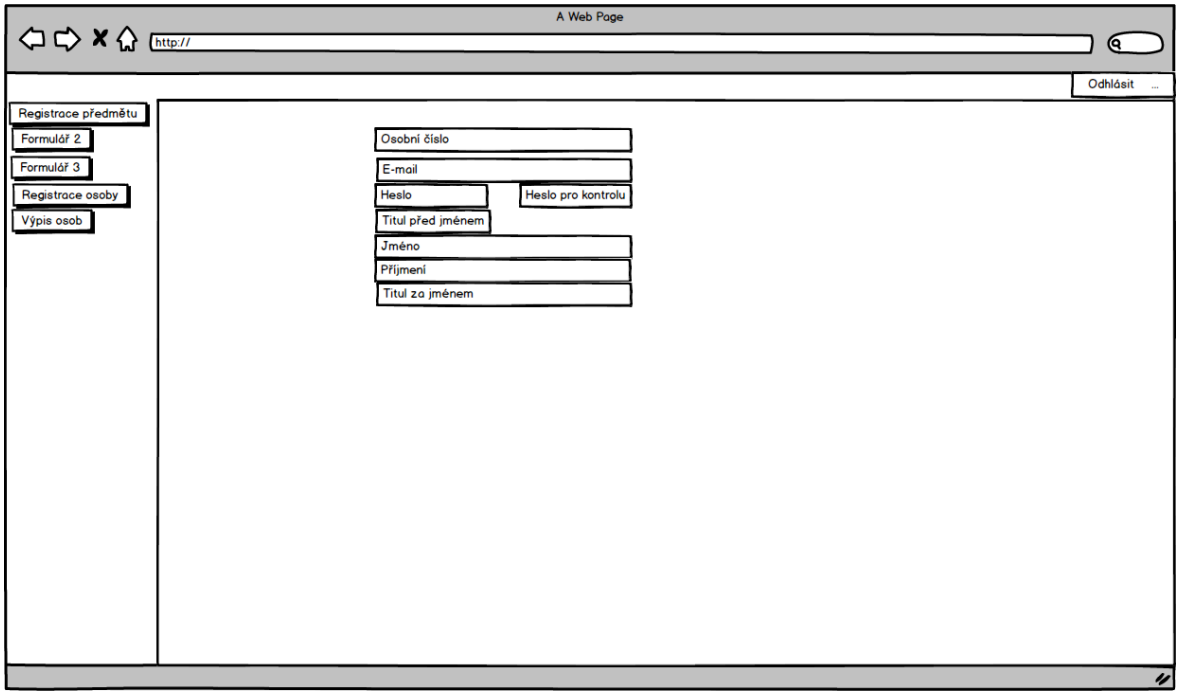
4.5.4 Ukázka prototypů pomocí wireframů

Wireframy pomáhají ujasnit si, jak bude vypadat struktura aplikace, jak budou rozmístěné jednotlivé prvky na stránce, apod.

Protože mnoho stránek, kde se zadávají údaje, případně kde jsou vypisovány, vypadají velice podobně, přikládám na ukázku od každého jeden vzorek.

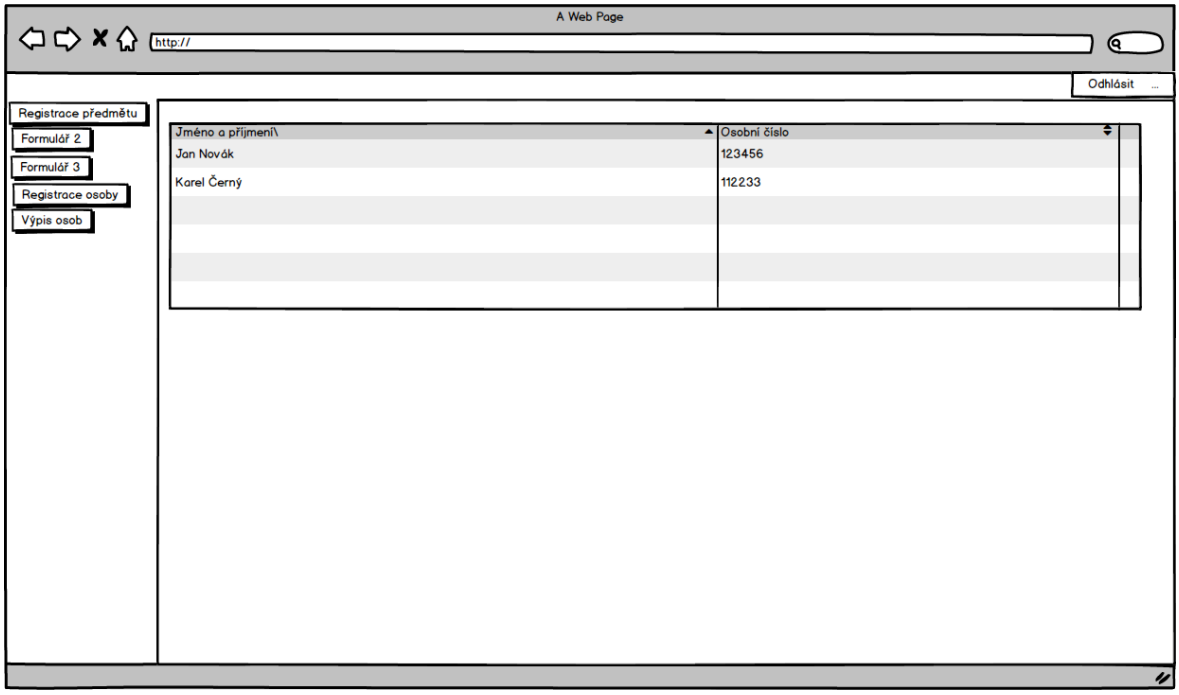
Zadávací formulář na obrázku 43 je prvním příkladem.

Obrázek 43 Návrh zadávacího formuláře *Zdroj: vlastní zpracování*



Druhým je na obrázku 44 zmiňovaný výpis údajů, zde konkrétně se jedná o návrh výpisu zaměstnanců.

Obrázek 44 Návrh výpisu zaměstnanců *Zdroj: vlastní zpracování*

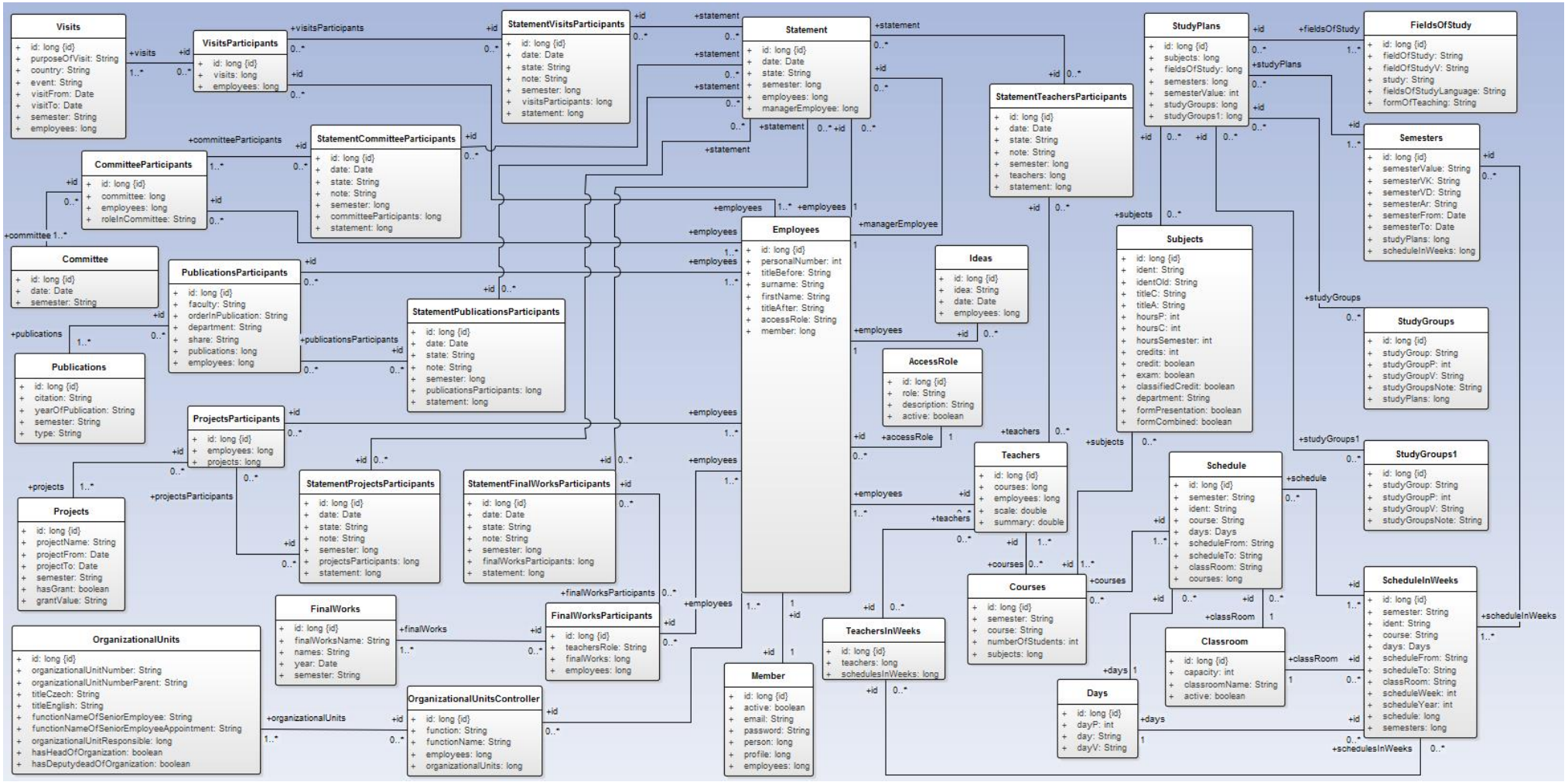


Zadávacích formulářů a výpisových stránek je zde větší množství, vzhledově vypadají podobně, ale vnitřní logika se liší.

4.6 Diagram tříd

Statický diagram tříd vychází z případů užití a scénářů, které nám pomohou určit jednotlivé třídy. Systém obsahuje 36 základních tříd. Většina tříd obsahuje další metody, které ale z důvodu přehlednosti v diagramu neuvádím. Na obrázku 45 jsou pouze třídy z části *Model* Play Frameworku, které zároveň reprezentují datový model databáze. Ústředním motivem je zaměstnanec, kolem kterého se vážou aktivity. Uživatel zadává veškeré údaje, vyučuje předměty, píše publikace, či například vede závěrečné práce.

Obrázek 45 Diagram tříd z části Model *Zdroj: vlastní zpracování*



4.7 Popis tříd

Diagram tříd na obrázku 45 je velmi rozsáhlý, proto uvedu pouze jejich stručný popis.

Třída *Employees* definuje zaměstnance na MÚVS. Definuje jeho osobní číslo a váže na sebe další třídy, které reprezentují činnosti, které jsou součástí osobního hodnocení každého zaměstnance definovaného v kapitole 4.3.

Třída *Member* zpracovává přihlašovací údaje do systému, kterými jsou e-mail a heslo. Heslo je zahashováno pomocí Bcryptu popsaného v kapitole 3.5.3.

Třída *Teachers* obsahuje informace o vyučujícím a kurzech, na kterých se podílel. Zároveň obsahuje váhu, která říká, jakým procentem toto zapojení probíhalo. Kurzy reprezentuje třída *Courses*, obsahující informaci o předmětech reprezentující třídou *Subjects*. Každý kurz je součástí rozvrhu, který představují třídy *Schedule* a *ScheduleInWeeks*.

Zadané předměty jsou součástí studijních plánů (třída *StudyPlans*), ty se konají v semestrech (třída *Semesters*) a obsahují studijní skupiny (třída *StudyGroups*). Každý studijní plán patří k oboru studia (třída *FieldsOfStudy*).

U vyučujících je umožněno, že se účastní pouze některých týdnů výuky daného předmětu, pro tyto účely slouží třída *TeachersInWeeks*, která ukládá informace, ve kterém týdnu výuky daného předmětu daný vyučující učil.

Třídy *Committee* a *CommitteeParticipants* obsahují informace o účastech a rolích v komisích u Státních závěrečných zkoušek.

Třídy *Publications* a *PublicationsParticipants*, jsou třídy, které zaznamenávají informace o publikační činnosti zaměstnanců.

Třídy *Projects* a *ProjectsParticipants*, uchovávají informace o projektech, na kterých zaměstnanci pracovali a také, zda a jaký grant projekt dostal.

Třídy *FinalWorks* a *FinalWorksParticipants*, slouží k uložení záznamů o spolupráci na závěrečných pracích studentů, jako je například vedení nebo oponentura.

Třídy *Visits* a *VisitsParticipants*, obsahují informace o výjezdech na služební cesty, konference, apod.

Třídy *OrganizationalUnits* a *OrganizationalUnitsParticipants*, slouží k zaznamenání zaměstnance do organizační jednotky. Organizační jednotky, jejich názvy a názvy vedoucích funkcí jsou uvedeny v příloze k Organizačnímu řádu Masarykova ústavu vyšších studií ČVUT v Praze.

Třídy *Statement*, *StatementCommitteeParticipants*, *StatementFinalWorksParticipants*, *StatementProjectsParticipants*, *StatementPublicationsParticipants*, *StatementTeachersParticipants*, *StatementVisitsParticipants*, sbírají informace o činnosti pracovníků a slouží k tvorbě a schvalování těchto výkazů. Výkazy se tvoří za semestry a vedoucí pracovník má možnost je odeslat danému pracovníkovi ke schválení.

Další třídy jako jsou *Days*, *Classroom*, *AccessRole*, slouží jako číselníky. Tedy obsahují hodnoty, které se používají ve formulářích pomocí rozbalovacích seznamů. Uživatel tedy nemůže zadat jinou hodnotu a navíc tato hodnota má stejnou podobu napříč ukládanými daty (například první písmeno velké).

5 Implementace

Tato kapitola se věnuje výběru použitých technologií při implementaci analyzovaného systému a také popisu implementace funkčních požadavků. Po implementaci jsem provedl nasazení na server heroku.com, na kterém jsem testoval zátěž pro různý počet uživatelů. Porovnání stavu před a po implementaci a zhodnocení časové úspory je uvedeno v závěru kapitoly.

5.1 Použité technologie

Při implementaci jsem zvolil následující technologie:

- Databázová vrstva
    - Databáze Postgresql
- Aplikační vrstva
    - Play Framework s podporou portu 9000
- Prezentační vrstva
    - Play Framework
    - HTML, CSS, JavaScript
    - Pro vzhled systému jsem použil Twitter Bootstrap ve třetí verzi, který obsahuje nástroje pro úpravu vzhledu formulářů, tlačítek a dalších komponent.

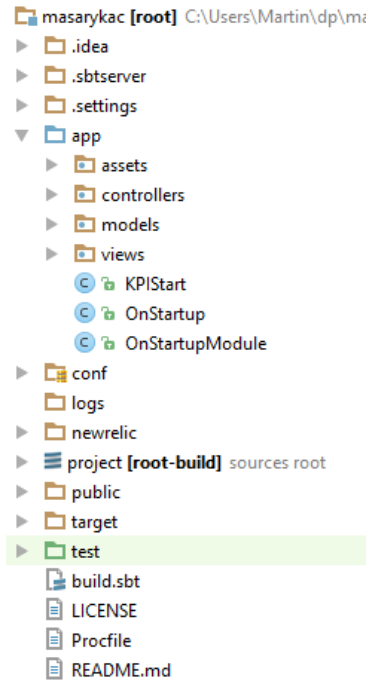
5.2 Výběr frameworku

V rámci realizace této diplomové práce jsem zvolil open source Play Framework. Jedná se o framework založený na jazyku Java a Scala. Vývojář tak může zvolit, který jazyk je mu bližší. Framework je založen na principu REST (více viz REST v kapitole 3.5.2) a nabízí řešení pro tvorbu škálovatelných aplikací. Je rychlý při zpracování požadavků, a protože podporuje asynchronní komunikaci, neblokuje cenné výpočetní zdroje.
Zvolil jsem jej jednak z předchozí osobní zkušenosti, a také protože jej používají nadnárodní společnosti, například služba LinkedIn (Lightbend, 2017). Při vývoji jsem použil jazyk Java pro tvorbu *modelů* a *controllerů*, pro *views* jsem využil kombinaci jazyka Scala a HTML.
Framework je dostupný s instalačním prostředím TypeSafeActivator, které automaticky nainstaluje samotný framework, Scalu a komponenty Akka a Activator.
Akka je systém, který se používá pro asynchronní zpracování, práci na pozadí a rozložení náročných úloh mezi více serverů a tím umožní dostupnost aplikace.

Activator je platforma, kterou framework používá pro sestavení, spuštění a ladění aplikace.

Play Framework vychází z MVC architektury (viz MVC v kapitole 3.5.2), kde má každá z vrstev svůj balíček v adresáři *app*. Zároveň je zde balíček *assets*, kde jsou další soubory a knihovny potřebné pro vnitřní fungování aplikace. V mém případě se jedná o soubory definující vzhled stránek a javascriptové soubory pro dynamické akce na webové stránce. Strukturu souborů zobrazuje obrázek 46.

Obrázek 46 Struktura souborů *Zdroj: vlastní zpracování*



Framework podporuje objektově – relační mapování, kdy v modelech lze mapovat třídy na tabulky v databázi. V Javě to má na starosti knihovna Ebean, která pomocí anotace *@ + klíčové slovo* definuje činnosti. Pro vygenerování tabulky ze třídy se použije anotace *@Entity* a pro primární klíč se použije *@Id*. Dále je možné pomocí anotace vytvořit například relační vazbu mezi třídami. Ukázku poskytuje následující obrázek 47.

Obrázek 47 Ukázka modelové třídy *Zdroj: vlastní zpracování*

```
@Entity
public class Employees extends Model {

    public static Finder<Long, Employees> find = new Finder<~>(
            Employees.class);

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long id;

    @Formats.NonEmpty
    @Constraints.Required(message = "Vyplňte osobní číslo")
    public int personalNumber;

    public String titleBefore;

    @Constraints.Required(message = "Vyplňte příjmení")
    @Formats.NonEmpty
    @Constraints.MinLength(message = "Zadejte alespoň 2 znaky", value = 2)
    public String surname;

    @Constraints.Required(message = "Vyplňte křestní jméno")
    @Formats.NonEmpty
    @Constraints.MinLength(message = "Zadejte alespoň 2 znaky", value = 2)
    public String firstName;

    public String titleAfter;

    public String accessRole;
```

Obvykle se při tvorbě aplikací za pomocí jazyka Java používá pro atributy modifikátor přístupu *private*. Tedy, kdo všechno může daný atribut vidět z okolních tříd. Modifikátor *private* značí, že je viditelný pouze ve třídě, ve které je definovaný. Pro přístup z okolních tříd se využívá get metod pro získání hodnoty a set metod pro definování hodnoty, které je nutné nadefinovat. Play Framework je ale výjimka, protože atributy mají modifikátor *public* a framework až v průběhu kompilace vytváří set a get metody.

Soubory použité pro část *View* se starají o vykreslování uživatelských obrazovek. Nachází se v nich HTML, CSS a javascriptový kód. Zároveň je možné do těchto souborů definovat parametry z vrstvy Controller. Získáme tím dynamické fungování stránky. Například pomocí podmínky lze vypisovat jen určité údaje. Na obrázku 48 je ukázka části kódu v sekci *view.*

Obrázek 48 Ukázka kódu v části View *Zdroj: vlastní zpracování*

```
@(semesterForm: Form[Semesters])
@import helper._
@localScripts = {
}
@main(scripts = localScripts) {
    <div class="row">
        <div class="col-lg-12">
            <h1 class="page-header">Registrace</h1>
        </div>
    </div>
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-md-offset-2">
                <div class="panel-body">
                @helper.form(action = routes.SemestersController.save) {
                    <fieldset>
                        <legend>Zadání předmětu</legend>
                        <div class="row">
                            <div class="col-md-12">
                            @inputText(
                                semesterForm("semesterValue"),
                                '_label -> "Semestr",
                                'class -> "form-control",
                                '_help -> "",
                                '_error -> semesterForm.globalError
                            )
                            </div>
                        </div>
                        <div class="row"...>
                        @inputText(
                            semesterForm("semesterAr"),
                            '_label -> "Akademický rok",
                            'class -> "form-control",
                            '_help -> "",
                            '_error -> semesterForm.globalError
                        )
                        <div class="row"...>
                    </fieldset>
                    <div class="actions">
                        <input type="submit" class="btn btn-lg btn-success btn-block" value="Uložit">
                        <a href="@routes.Application.index()" class="btn">Zrušit</a>
                    </div>
                }
                </div>
            </div>
        </div>
    </div>
}
```

Business logika aplikace, tedy to, co se má stát s objekty definovanými ve vrstvě Model, je obsažena ve vrstvě Controller. Reaguje na události z vrstvy View a zpracovává je, protože obvykle vyvolají změnu na modelové vrstvě. Play Frameworku podle HTTP požadavků definovanými v souboru *routes*, pozná, o jaký požadavek se jedná. Typicky jde o požadavky popsané v kapitole REST.

Řízení práv pro role

Jak již bylo řečeno výše, systém poskytuje dvě úrovně zabezpečení podle rolí. První část je na straně *view*, kde je zobrazení odkazů podmíněno rolí uživatele. Uživatel, který potřebnou roli nemá, nevidí odkaz. Příkladem je přidávání nových předmětů do kurzu, které může pouze vedoucí MUVS, zobrazuje obrázek 49.

Obrázek 49 Řízení přístupu na straně View *Zdroj: vlastní zpracování*

```
@if(session().get("role").equals("director")) {
    <li>
        <a href="@routes.SubjectPlanTeachingController.index()">Přidání předmětu do kurzu</a>
    </li>
}
```

Druhou částí je řešení přístupu na úrovni *Controlleru*, kde je u každé akce rozlišen přístup pro uvedenou roli. Pokud uživatel roli nemá, je přesměrován na stránku s přihlášením. Obrázek 50 zobrazuje situaci pro zobrazování místností.

Obrázek 50 Řízení přístupu na straně Controlleru *Zdroj: vlastní zpracování*

```
public Result listClassrooms() {
    if (Check.isDirector(Member.findByEmail(request().username()))) {
        return ok(views.html.tables.tableClassroom.render(Classroom.search()));
    }
    notAccess();
    return redirect(routes.Application.index());

}
```

Mazání dat

Často je potřebné zadaná data smazat, jedná se například o odcházejícího zaměstnance. Vyskytuje se však problém, co se stane s předměty, které tento zaměstnanec vyučoval. Co se stane s publikacemi, které napsal. Takovýchto problémů by šlo napsat více. V praxi se proto používá řešení, kdy tyto záznamy (v tomto případě zaměstnanec) se stanou neaktivní. V databázi jsou stále obsaženi, ale již není možné používat je pro nové zadávání. Důvodem je zachování historie tak, aby činnosti zůstaly provázané a prokazatelné. Můžeme tak stále tvořit výstupy do historie, či statistiky v čase.

5.3 Využití Kaizen a Kanban a Poka-Yoke

Při analýze a implementaci jsem využil softwarového nástroje Trello a metody Kanban. Vhodnější by zde byla agilní metoda Scrum, ale z důvodu existence pouze autora práce a vedoucí této diplomové práce, jako členy týmu, ji nešlo využít.

Kaizen jsem využil při konzultacích s vedoucí práce. Ukázaná část aplikace obdržela zpětnou vazbu, podle které se naplánoval nový budoucí stav, který jsem vytvořil, a znovu jsme vše zkonzultovali.

Použití Poka-Yoke je v aplikaci časté, protože jsem chtěl uživateli zpříjemnit její používání. Chybám předcházím následovně:

- Tam kde jsou povolená pouze čísla (například podíl na vyučovaném předmětu), nejsou povoleny jiné znaky.
- U zadávání datumů se zobrazuje kalendář, ze kterého si uživatel vybere.
- Některá pole umožňují vybrat hodnoty z předem definovaného číselníku, není tak možné v aktuálním formuláři zadat špatná data.
- Povinná pole ve formulářích obsahují symbol hvězdičky, a dokud není zadán alespoň nějaký znak, jsou ohraničena červeně.
- Dalším omezením je maximální počet znaků, kdy například číslo oddělení má povolené dva znaky.

5.4 Implementace funkčních požadavků

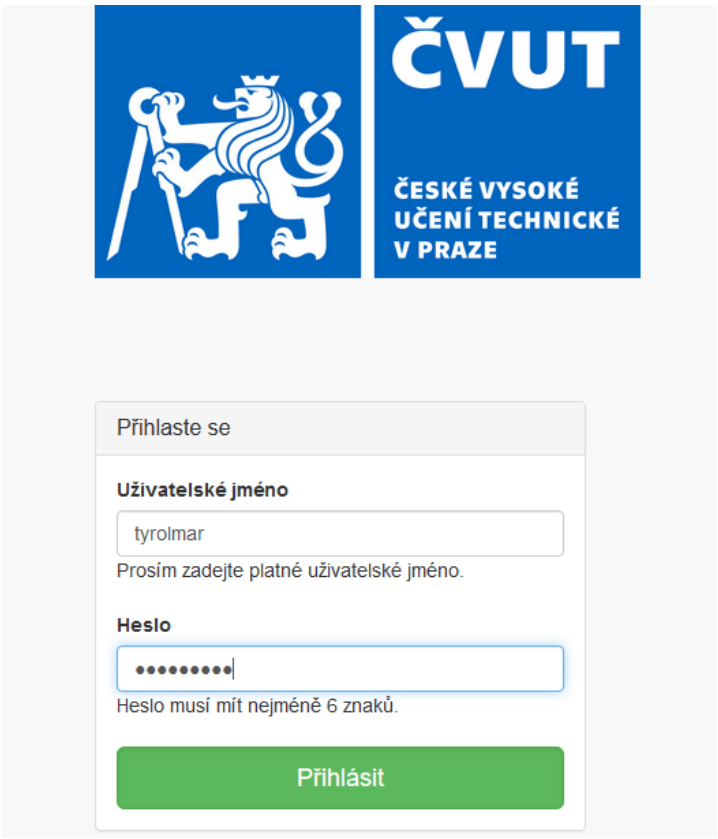
Na základě analyzovaných požadavků jsem provedl jejich implementaci do nového informačního systému. Na implementaci navazuje testování a nasazení.

Implementace funkčních požadavků souvisí se zadanými případy užití. Každý funkční požadavek musí být obsažen v alespoň jednom případu užití. Jednotlivé případy užití jsem implementoval do systému a jejich popis následuje níže. Znázorněné oblasti obsahují fiktivní uživatele, aby si čtenář mohl udělat představu o funkčnosti systému. Popisuji hlavní oblasti implementace. U zadávání a výstupů je princip velice podobný a není nutné všechny uvádět.

Oblast přihlášení

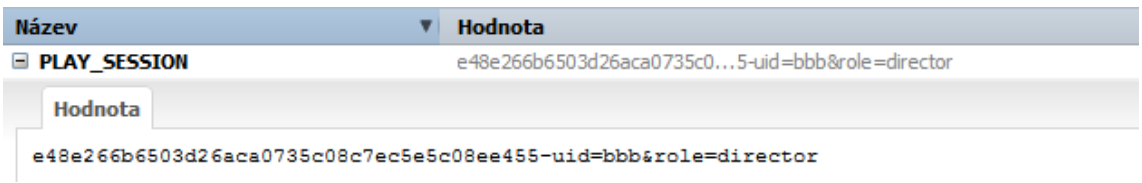
Úvodní obrazovka k přihlášení do systému. Je potřeba zadat platné uživatelské jméno, kterým je jednotné UID používané na ČVUT a heslo. Bez platných údajů není možné pracovat se systémem. Obrázek 51 tuto skutečnost zobrazuje. V tuto chvíli ale aplikace není propojená s ověřovacím serverem ČVUT a uživatelské jméno a heslo je potřeba přiřadit uživateli při jeho registraci do tohoto systému administrátorem.

Obrázek 51 Úvodní obrazovka systému *Zdroj: vlastní zpracování*



Po přihlášení je uživateli vygenerován jedinečný identifikátor do proměnné session (viz kapitola 3.5.3). Následně je uživateli zobrazena hlavní plocha systému. Příklad identifikátoru zobrazuje obrázek 52.

Obrázek 52 Příklad identifikátoru *Zdroj: vlastní zpracování*



Oblast hlavní plochy

Hlavní plocha je ústřední obrazovkou celého systému, jak zobrazuje obrázek 53. Rozděluje se na tři oblasti.

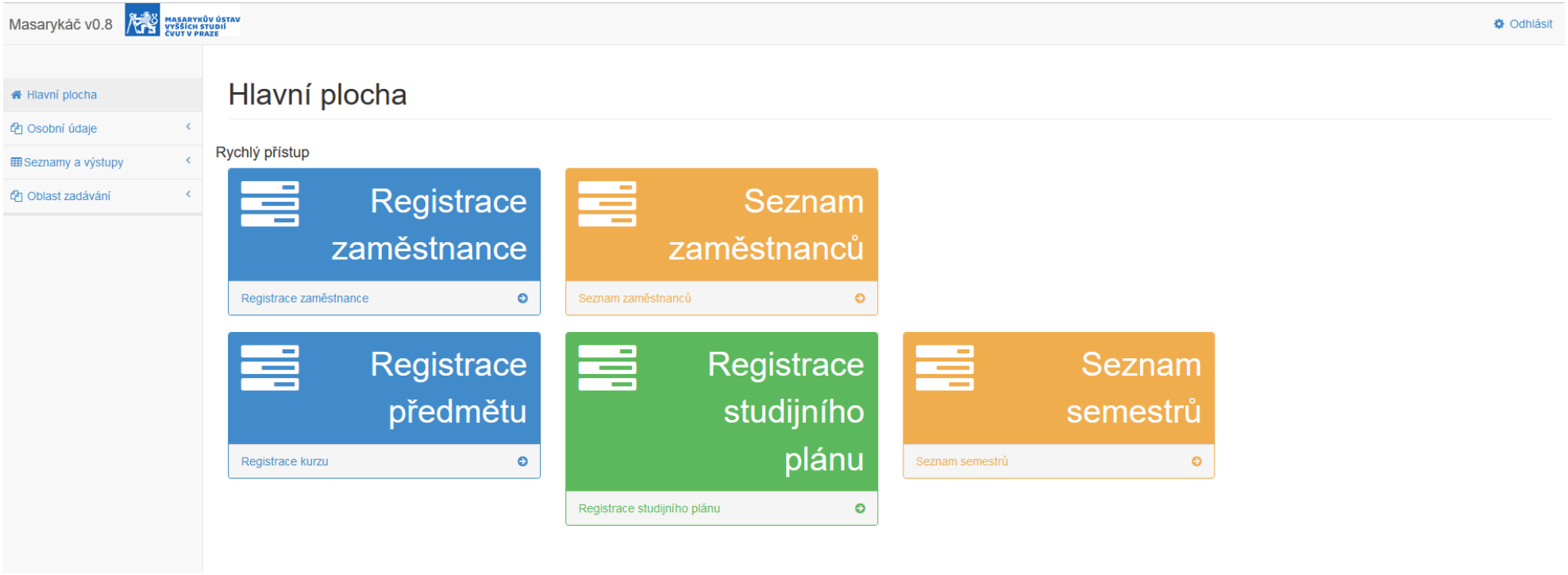
Vrchní panel umožňuje odhlásit uživatele. Menu v levé části zobrazuje akce, které má uživatel povolené. Dělí se na sekce:

- Zadávání údajů
- Prohlížení údajů formou seznamů
- Osobní údaje

Třetí částí, která zabírá většinu zobrazované plochy, je samotná plocha, která nabízí rychlý přístup k nejčastějším akcím.

Při vytvoření výkazu odpovědným pracovníkem a následném odeslání k hodnocenému pracovníkovi, zobrazí se hodnocenému tato informace na ploše. Následně si může výkaz otevřít a provést příslušné operace.

Obrázek 53 Hlavní plocha *Zdroj: vlastní zpracování*



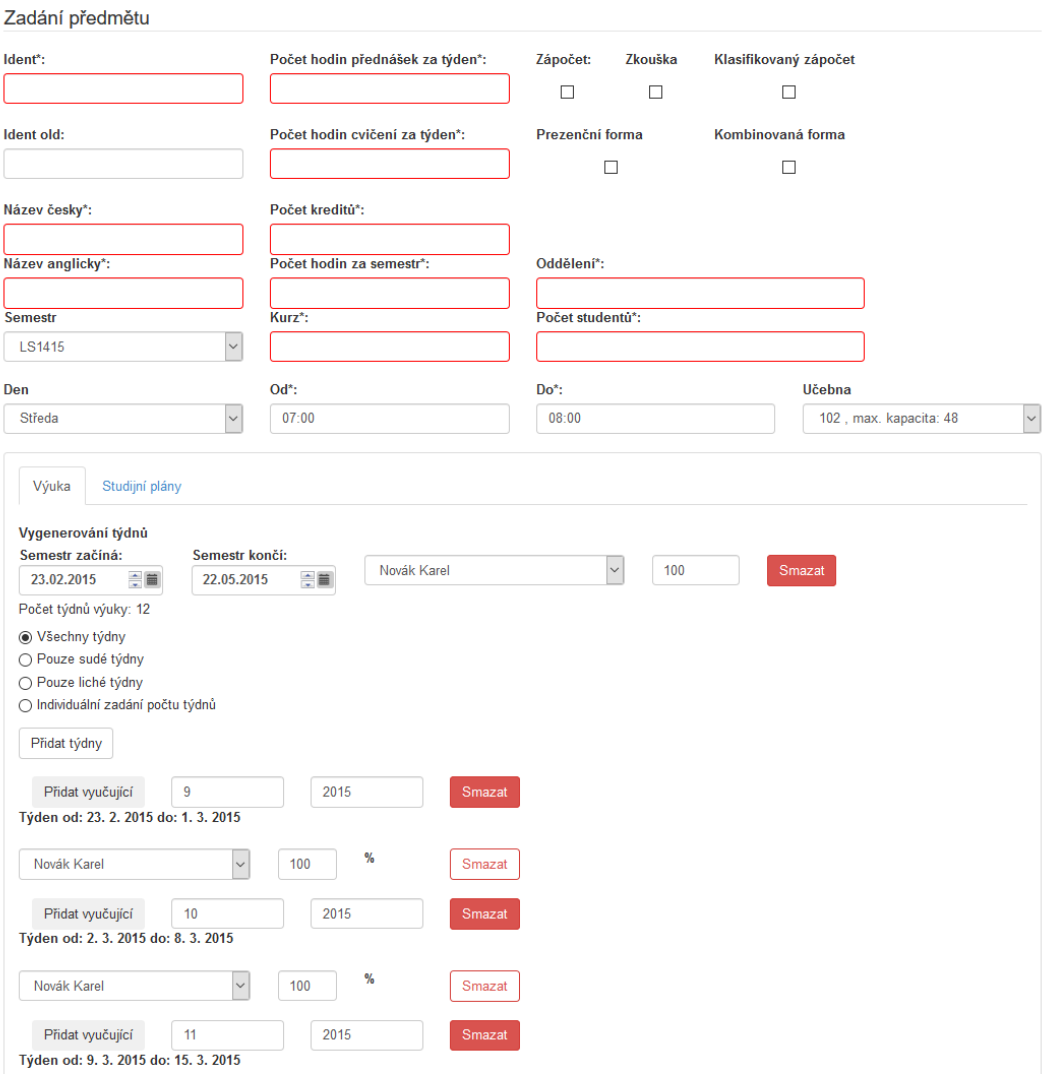
Oblast zadávání předmětů

Při zadávání předmětů jsem využil Poka-Yoke, kde uživatel je informován o povinných polích, která musí vyplnit.

Předmět vyučují vyučující, které je možné přidat hromadně ke všem týdnům výuky nebo pouze ke konkrétním týdnům. U každého vyučujícího se zadává váha, kterou se daný týden podílel na výuce.

Každý předmět se zařadí podle určení ke studijním plánům. Obrázek 54 reprezentuje formulář pro zadávání předmětů.

Obrázek 54 Zadávání předmětů *Zdroj: vlastní zpracování*



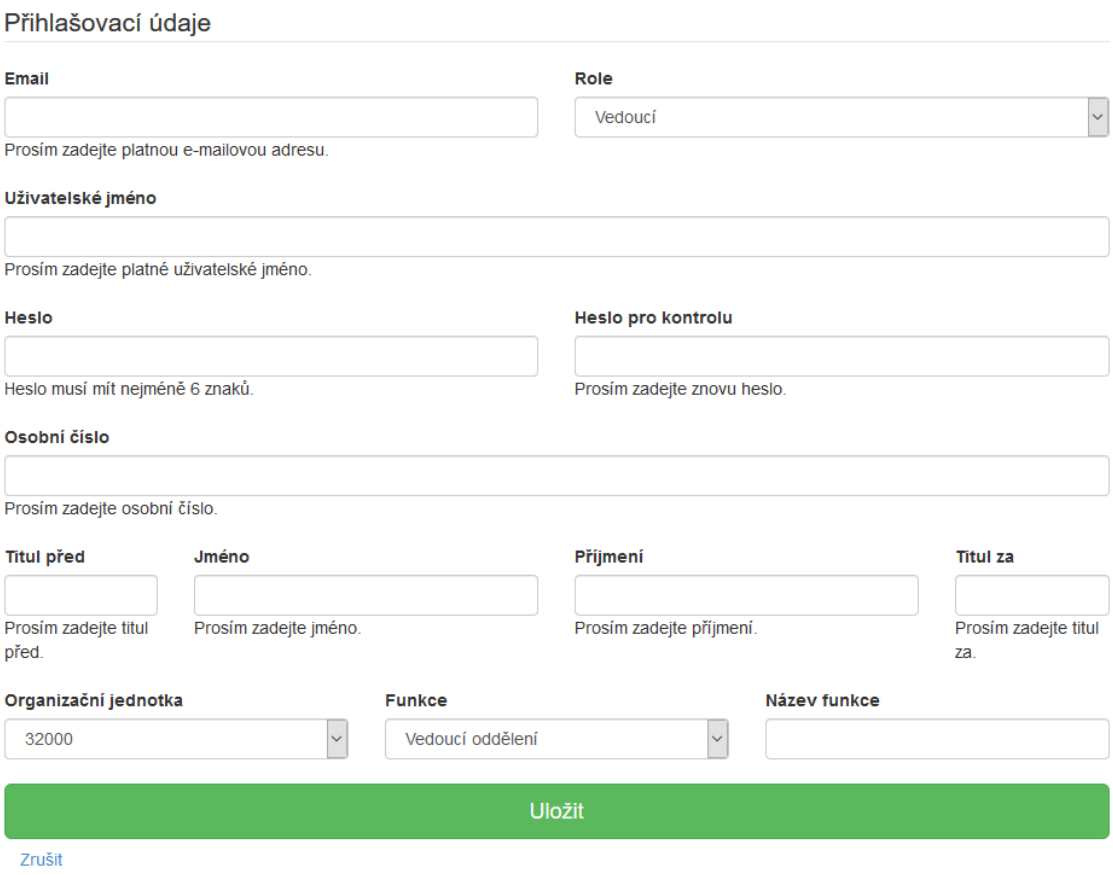
Při zadávání nového předmětu do kurzu systém musí pohlídat, aby již v databázi nebyl uložený předmět se stejným *identem*, nebyl učen ve stejném předmětu a neměl stejný *kurz*. Když by nastala tato situace, systém neprovede uložení.

Na pozadí ukládacího procesu proběhne přepočet váhy za jednotlivé uživatele a týdny výuky na celkovou váhu za každého uživatele a předmět. Tento výstup je dále použitý pro semestrální výkazy.

Oblast zadávání uživatelů

Každý nový uživatel, který bude přidán do systému, musí mít definovanou roli a organizační jednotku.

Obrázek 55 Formulář pro zadání nového uživatele *Zdroj: vlastní zpracování*



Obrázek 55 ukazuje tuto oblast zadávání. Heslo se musí zadávat dvakrát, aby se zamezilo překlepům.

Oblast výstupů

Mezi výstupy patří doporučené časové plány v semestrech, výkazy a seznamy. Seznamy obsahují předměty, vyučující a další oblasti, které poskytují informace.

Každý z výstupů se zobrazuje ve formě tabulky a vybrané výstupy je možné stáhnout do formátu pdf nebo doc. Takto vytvořené a stažené soubory obsahují veškeré informace, ale designově se mohou lišit od podoby tabulky. Některé výstupy obsahují tlačítka pro další akce, jako je mazání a upravování.

Příkladem výstupu je seznam studijních skupin na obrázku 56. Jak je vidět, dlouhý text se automaticky zalamuje tak, aby nebyl sloupec příliš široký.

Obrázek 56 Seznam studijních skupin *Zdroj: vlastní zpracování*

Studijní skupiny

Výpis studijních skupin

+ Přidat studijní skupinu | Generovat PDF | Generovat DOC

Zobrazit 10 záznamů Hledat:

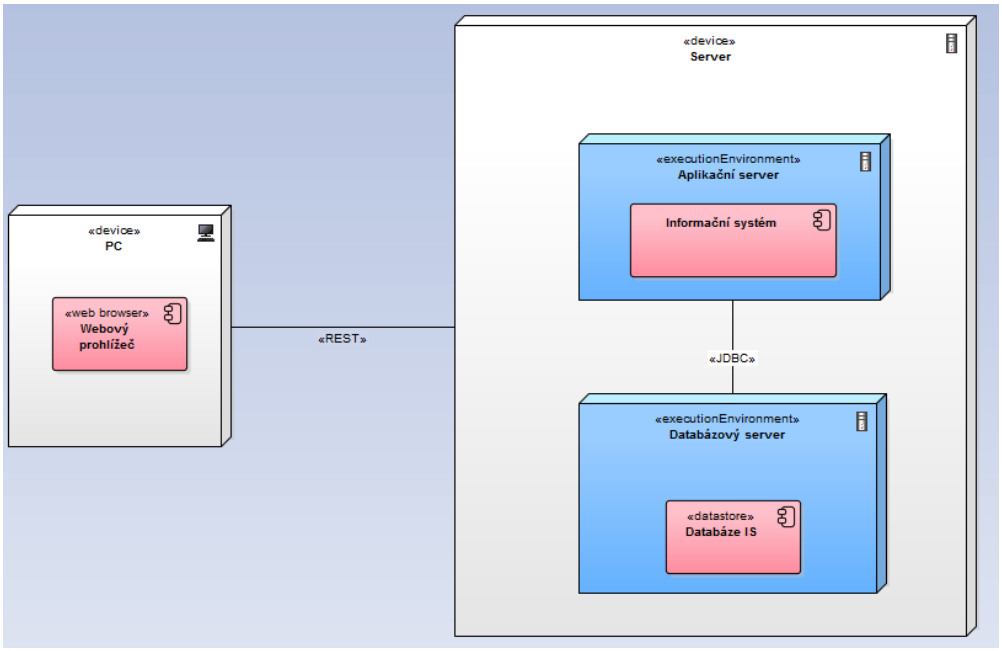
Skupina	Skupina_P	Skupina	Poznámka	Upravit
G_PVT	3	Povinně volitelné technické předměty	Studenti si volí z nabídky povinně volitelných technických předmětů tak, aby celkem za studium získal 6 kreditů. Předmět G77C0003 a G77C0004 lze studovat i na jiné fakultě ČVUT po předchozím souhlasu garanta oboru (bude řešeno uznáním vystudovaného předmětu). Nabídka předmětů bude průběžně aktualizována.	Upravit
U_PVa_6s	5	Povinně volitelné předměty	Student volí předměty tak, aby celkem získal 24 kreditů. Předmět U77C0005 lze studovat i na	Upravit
U_PV_6s	5	Povinně volitelné předměty	Student volí předměty tak, aby celkem za studium získal 6 kreditů. Nabídka předmětů bude průběžně aktualizována, předměty nemusí být vypsány v každém semestru.	Upravit
U_PVb_6s	5	Povinně volitelné předměty	Student volí předměty tak, aby celkem za studium získal 16 kreditů. Nabídka předmětů bude průběžně aktualizována, předměty nemusí být vypisovány každý semestr. Předměty U88E0201 až U88E0204 jsou předměty vyučované zahraničním profesorem na ČVUT či v zahraničí. Pro zápis předmětů začínající U88 je nutný předchozí souhlas.	Upravit

5.5 Nasazení

Informační systém je aktuálně nasazený na serveru heroku.com. Jedná se o cloudové řešení pro Play Framework. Pokud aplikace není delší dobu používána uživateli, server ji uspí a probuzení trvá jednotky vteřin. Pro potřeby MÚVS je to ale dočasné řešení, než se přesune vše na interní servery. Aktuální verze nasazeného systému je dostupná přes http://adaro.eu/dp.html.

Server, na kterém bude systém nasazen, musí mít nainstalovanou databázi PostgreSQL ve verzi 9.6.2 a novější. Obrázek 57 zobrazuje diagram nasazení a jednotlivé komponenty. Pro používání aplikace je klientská část dosažitelná pomocí webového prohlížeče s podporou javascriptu.

Obrázek 57 Diagram nasazení *Zdroj: vlastní zpracování*



5.6 Testování

Testování výsledné aplikace probíhalo za pomocí řady testů. Od testování jednotlivých tříd a metod, až po testování celých procesů. Jsou to nejen end-to-end testy, ale i progresní a regresní testy popsané v kapitole 3.5.7.

K výkonnostnímu testování jsem použil aplikaci Apache JMeter, která simuluje přístup nastaveného počtu uživatelů na zadaný zdroj a sleduje odezvu od tohoto zdroje. Pro měření jsem využil nasazeného systému na serveru heroku.com. Měření je tak objektivní, oproti měření na lokálním počítači, na kterém autor práce systém vyvíjel. Pro každý případ měření jsem provedl deset opakování, abych co nejvíce minimalizoval výkyvy internetového připojení a propustnosti serveru heroku.com.

Základní měření jsem provedl pro jednoho uživatele, za minutu se jednalo průměrně o 209 zaslaných požadavků s průměrnou dobou odezvy 288 ms.

Pro další měření jsem zvolil pět uživatelů. Jde o odhad počtu uživatelů, kterým by mohl být systém pravidelně zatížen. Průměrná doba odezvy činila 289 ms při 1031 průměrně zaslaných požadavků.

Pro extrémní zátěž jsem zvolil 200 uživatelů. Jedná se o případ, když by chtěli všichni zaměstnanci (počítáno i s rezervou) přistoupit najednou do systému. Průměrná odezva činila při 12 751 průměrně zaslaných požadavcích 968 ms. Je zde tedy nárůst v době odezvy, ale stále se při tolika zaslaných požadavcích jedná o velmi dobrou hodnotu.

Výsledné hodnoty pro každý případ měření jsou přijatelné, uživatel si ani nevšimne rozdílu podle počtu přihlášených uživatelů a jeho požadavek je splněn ve velmi krátké době.

5.7 Návrh možného rozšíření

Možné rozšíření souvisí s tím, že Masarykův ústav vyšších studií je součástí ČVUT. Nabízí se tedy propojení s dalšími systémy. Zároveň existuje možnost využití jednotného přihlašování a nabídky rolí. Další možností na zlepšení je angažování grafického konzultanta, který by pomohl s vizuálním stylem. Pro vylepšování a hlášení chyb jsem pro tuto činnost vytvořil proces, ve kterém uživatel zadá daný problém a systém jej uloží s aktuálním datem. Na vedoucím pracovníkovi je pak posouzení, zda je tento záznam relevantní nebo ne.

5.8 Výhody inovativního informačního systému

Výhodou nového, inovativního systému, je sjednocení zadávání a poskytování informací. Data jsou centralizována, provázána a jednotně zadávána. Jedná se o webovou aplikaci, která podporuje responzivní návrh webu (stránka se přizpůsobí zařízení, na kterém je zobrazena), je tak možné přistupovat k informacím opravdu odkudkoliv.

Ohromnou inovativní výhodou je eliminace plýtvání ve formě zbytečných prostojů a čekání na informace a také nákladů na tisk. Již není nutné každý výstup tisknout, uživatelé systému podle svého oprávnění k nim mají přístup. Nemusí se také čekat na dodání výstupu nebo jeho schválení.

Dalším benefitem je eliminace nekompletních informací. Osoba s dostatečným oprávněním má jasný přehled o vyučovaných předmětech a také rozvrhu, kde vidí, ve které učebně a v jakém časovém rozsahu probíhá výuka.

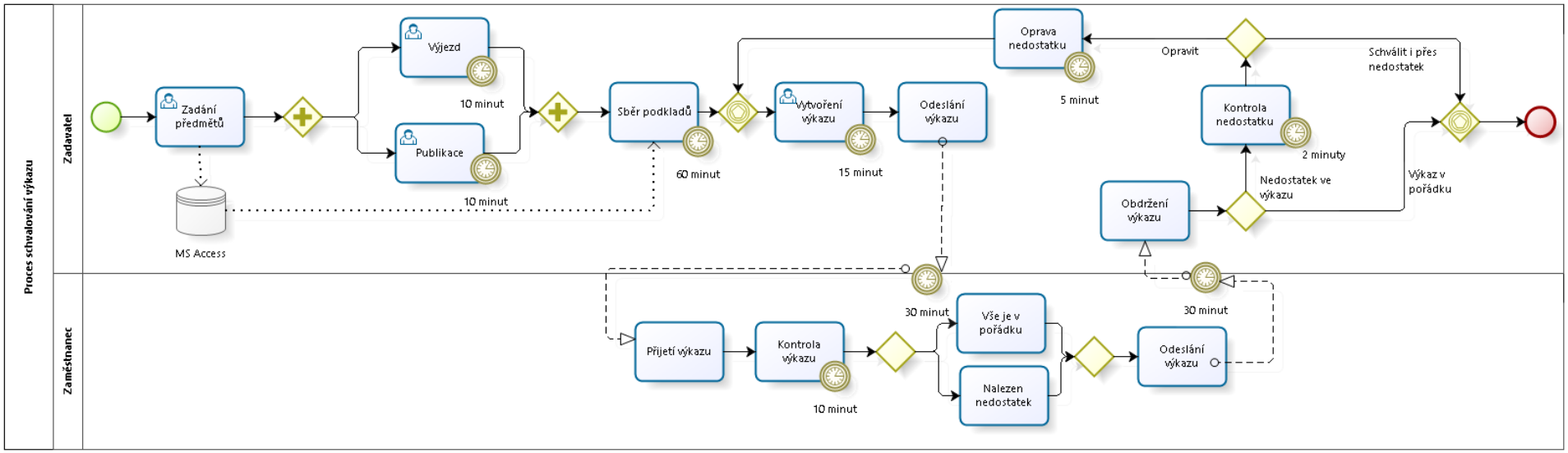
Zvýší se produktivita práce, protože zaměstnanec již nebude trávit čas činnostmi, které způsobují plýtvání v oblasti evidenci popisovaných činností v této diplomové práci.

5.8.1 Porovnání časové úspory

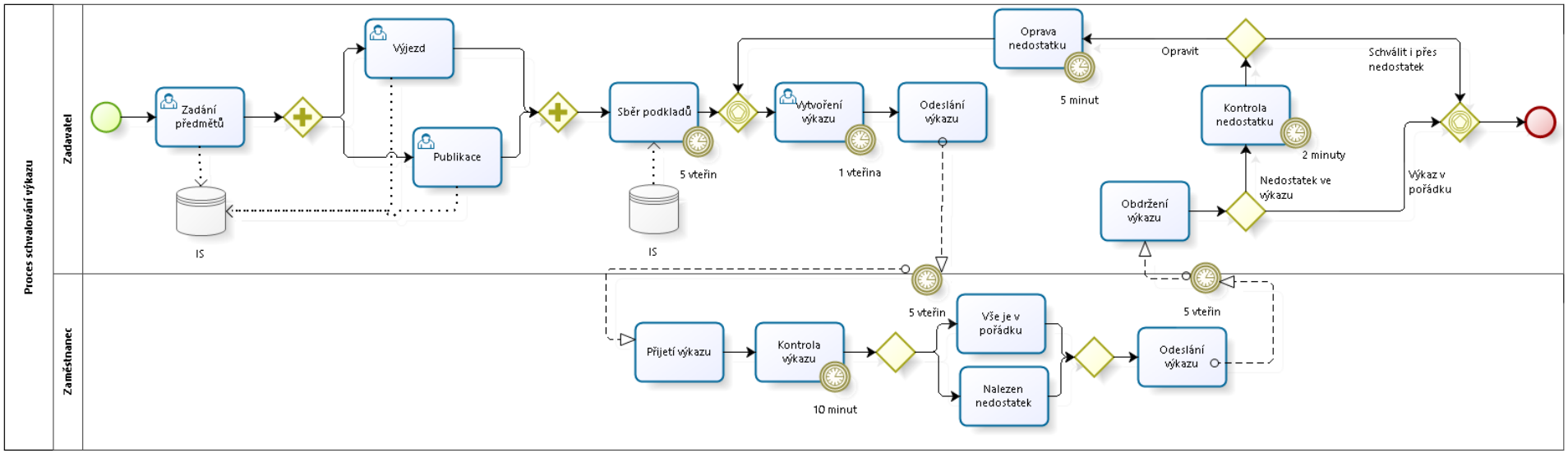
Tato kapitola demonstruje rozdíl při použití stavu před implementací a po implementaci nového informačního systému formou časového srovnání. Vyhodnocení pomocí finančních ukazatelů v tomto případě není relevantní. Software nemá investici v podobě financí.

Pro názorný příklad jsem si zvolil výstup zaměstnance, který vyučoval 5 předmětů, publikoval 1 článek a absolvoval výjezd na 1 konferenci. Předpokladem je, že nejdříve zadavatel v průběhu sledovaného období zadá data do příslušných systémů (pokud jsou k dispozici) a na konci sledovaného období provede hodnocení a předá jej zaměstnanci ke kontrole, který najde chybu v názvu předmětu (je jím překlep). Tuto chybu je nutné opravit, znovu předat výstup k akceptaci. Následně je již vše schváleno. Proces před implementací zobrazuje obrázek 58 a proces po implementaci je zachycen na obrázku 59.

Obrázek 58 Proces před implementací *Zdroj: vlastní zpracování*



Obrázek 59 Proces po implementaci *Zdroj: vlastní zpracování*



Experiment probíhal při zadávání následovně. Začátek byl vždy kliknutím na tlačítko otevírající příslušný formulář. Konec měření znamenalo uložení případu. Pro zadání předmětů, publikace a výjezdu předpokládám, že jsou k dispozici údaje (název, semestr, apod.).

Před implementací musel vedoucí nejprve sesbírat veškeré podklady. Sběr podkladů zabral čas, který definuji, jako konstantu *p*. Tato konstanta obsahuje žádost o poklady a čekání na jejich dodání. Výstup o vyučovaných předmětech zaměstnancem je konstanta *v*, která definuje čas pro filtrování podkladů v aplikaci MS Access do tabulkového procesoru. Protože stav před implementací neumožňuje evidovat publikace a výjezdy, jsou pro oba parametry vytvořené konstanty, *sp*, resp. *sv*. Definuji zde konstantu *sb* reprezentující čas, který je potřebný pro dodání výstupu konkrétnímu zaměstnanci. Obsahuje čas na tisk, čekání a osobní převzetí. Konstantu *schb* představuje čas, který potřebuje zaměstnanec pro kontrolu výstupu. Oba stavy mají pro tento úkol stejný čas. Pro vyhodnocení chyby je časová konstanta *sh*. Konstanta *so* reprezentuje čas pro opravu podkladu.

Pro tento případ jsem si definoval tyto hodnoty v tabulce 2.

Tabulka 2 Stanovení konstant *Zdroj: vlastní zpracování*

Konstanta	Hodnota času ve formátu hh:mm:ss
p	01:00:00
sp	00:10:00
sv	00:10:00
v	00:15:00
sb	00:30:00
schb	00:10:00
so	00:05:00
sh	00:02:00

Následující tabulka 3 porovnává přístup k informacím a časovou úsporu. Pro objektivitu experimentu provedl měření nejdříve autor práce, a poté nezávislý uživatel, který viděl oba stavy před implementací a po implementaci poprvé.

Tabulka 3 Časové porovnání *Zdroj: vlastní zpracování*

Parametr / Zadavatel	Stav před implementací [hh:mm:ss]		Stav po implementaci [hh:mm:ss]	
	Autor práce	Nezávislý uživatel	Autor práce	Nezávislý uživatel
Zadání jednoho nového předmětu Zadání 5 předmětů	00:02:11 00:10:55	00:03:46 00:18:50	00:02:35 00:12:55	00:04:36 00:23:00
Zadání publikace	sp = 0:10:00	sp = 0:10:00	00:01:16	00:01:59
Zadání výjezdu	sv = 0:10:00	sv = 0:30:00	00:01:27	00:02:04
Sběr podkladů pro tvorbu výkazu	p = 01:00:00	p = 01:00:00	00:00:05	00:00:05
Výstup ve formě výkazu za-městnance	v = 00:15:00	v = 00:15:00	00:01:00	00:01:00
Předání zaměstnanci ke kontrole	sb = 00:30:00	sb = 00:30:00	0:00:05	0:00:05
Kontrola zaměstnancem – nalezení chyby	schb = 00:10:00	schb = 00:10:00	schb = 00:10:00	schb = 00:10:00
Předání zpracovateli	sb = 00:30:00	sb = 00:30:00	00:00:05	00:00:05
Vyhodnocení chyby	sh = 00:02:00	sh = 00:02:00	sh = 00:02:00	sh = 00:02:00
Oprava podkladů	so = 00:05:00	so = 00:05:00	so = 00:05:00	so = 00:05:00
Vytvoření nového výkazu	v = 00:15:00	v = 00:15:00	v = 00:01:00	v = 00:01:00

Předání zaměstnanci ke kontrole	sb = 00:30:00	sb = 00:30:00	00:00:05	00:00:05
Kontrola zaměstnancem – již vše v pořádku	schb = 00:10:00	schb = 00:10:00	schb = 00:10:00	schb = 00:10:00
Předání zpracovateli	sb = 00:30:00	sb = 00:30:00	00:00:05	00:00:05
Výsledný čas	04:27:55	04:35:50	00:45:03	00:56:28

Jak plyne z tabulky 3 uvedené, hlavním přínosem je úspora času, která se ale projeví až v ostrém provozu. Časová úspora je přibližně tři a půl hodiny, jak u tvůrce této práce, tak u druhého uživatele, ve prospěch nového systému

U zadávání předmětu ve stavu po implementaci je nárůst času způsobený větší variabilitou při zadávání, jedná se o možnost přidat rozdílného vyučujícího ke každému týdnu výuky a také zadání rozsahu výuky.

Výsledkem je splnění zadání diplomové práce, které požadovalo inovaci ve formě úspory času. S přihlédnutím k tomu, že systém bude využívat desítky až stovky uživatelů, jde o zajímavou časovou úsporu v průměrné výši 81%.

Závěr

Cílem této diplomové práce bylo analyzování a navržení software na Masarykově ústavu vyšších studií na ČVUT pro evidování podkladů k hodnocení pracovníků. Webová aplikace vytvořená za pomoci Play Frameworku slouží k evidenci zaměstnanců, vyučovaných předmětů a dalších aktivit, které vstupují do výkazů pro hodnocení zaměstnanců. Každý zaměstnanec, který vykázal činnost za sledované období, má poté povinnost si tento výkaz prohlédnout a rozhodnout o jeho správnosti.

Při zpracování jsem nejprve analyzoval potřeby pro nový systém a po jejich vyhodnocení identifikoval klíčové oblasti. U nich jsem analyzoval funkční požadavky.

Na základě požadavků jsem provedl návrh budoucího stavu systému, který obsahuje nejen UML diagramy pro pochopení struktury, ale také popis rolí v novém systému a návrh obrazovek. Za využití on-line softwarového nástroje Trello jsem si vytvořil agilní tabuli, podle které jsem si řídil analýzu, vývoj a testování.

Výsledný informační systém splňuje analyzované požadavky a je přizpůsobený skutečným potřebám ústavu. Pomocí implantace navrženého software dojde k omezení plýtvání informací.

Inovací je nejen omezení plýtvání, ale nově také přístup všech zaměstnanců k aktuálním informacím, a to odkudkoli s připojením k internetu. Poskytovaný software obsahuje data na jednom místě a oproti původnímu řešení je eliminováno plýtvání v podobě času, který je potřebný pro zadání údajů a tvorbu výstupů. Úspora činí přibližně 81%.

Přínosem mé diplomové práce je pochopení průběhů jednotlivých procesů a jejich implementace do informačního systému.

Seznam použité literatury

ARLOW, Jim a NEUSTADT, Ila. 2007. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky.* 2., aktualiz. a dopl. vyd. Brno : Computer Press, 2007. 978-80-251-1503-9.

Bauer, Miroslav. 2016. *KAIZEN - Cesta ke štíhlé a flexibilní firmě.* místo neznámé : BizBooks, Albatros Media a.s., 2016. ISBN: 8026504194, 9788026504191.

Bejčková, Jana. 2015. Štíhlá administrativa - základ prosperující společnosti (2. část). [Online] 27. Říjen 2015. [Citace: 10. Květen 2017.] http://www.e-api.cz/25773n-stihla-administrativa-zaklad-prosperujici-spolecnosti-2.-cast.

Best, Kathryn. 2006. *Design Management: Managing Design Strategy, Process and Implementation.* místo neznámé : AVA Publishing, 2006. ISBN: 978-2940373123.

Bollinger, Gary a Natarajan, Bharathi. 2003. *JSP - Java Server Pages: podrobný průvodce začínajícího tvůrce webu.* místo neznámé : Grada Publishing a.s., 2003. ISBN: 9788024703404.

Boswell, Dustin. 2012. Storing User Passwords Securely: hashing, salting, and Bcrypt. [Online] 18. Červen 2012. [Citace: 26. Únor 2017.] http://dustwell.com/how-to-handle-passwords-bcrypt.html.

Božek, Ondřej. 2012. Data management, Architektury systémů. [Online] 2012. [Citace: 8. Březen 2017.] http://statnice.dqd.cz/__media/mgr-szz:in-ins:3__mvc.jpg.

Briol, Patrice. 2013. *BPMN 2.0 Distilled: The Business Process Modeling Notation.* 2013. ISBN 9781446104064.

Bruckner, Tomáš, Voříšek, Jiří a Buchalcevová, Alena. 2012. *Tvorba informačních systémů: Principy, metodiky, architektury.* místo neznámé : Grada Publishing a.s., 2012. ISBN: 8024779021, 9788024779027.

Buchalcevová, Alena. 2005. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky.* místo neznámé : Grada, 2005. ISBN: 9788024710754.

Čada, Ondřej. 2009. *Objektové programování: naučte se pravidla objektového myšlení.* místo neznámé : Grada Publishing a.s., 2009. ISBN: 9788024766997.

Čermák, Miroslav. 2010. IT Governance. [Online] 2. Červenec 2010. [Citace: 4. Březen 2017.] http://www.cleverandsmart.cz/it-governance/.

Danel, Roman. 2011. Agilní metodiky vývoje software. [Online] 2011. [Citace: 3. Březen 2017.] http://homel.vsb.cz/~dan11/is__skripta/IS%202011%20-%20Softwarove%20inzenyrstvi%20-%20agilni%20metodiky.pdf.

Dohnal, Jindřich. 2016. Jak na lean management v administrativě a službách. [Online] 22. Červen 2016. [Citace: 2. Březen 2017.] https://www.linkedin.com/pulse/jak-na-lean-management-v-administrativ%C4%9B-slu%C5%BEb%C3%A1ch-jindrich-dohnal.

Fowler, Martin. 2009. *Destilované UML.* 2009. ISBN: 978-80-247-2062-3.

Gála, Libor, Šedivá, Zuzana a Pour, Jan. 2015. *Podniková informatika: Počítačové aplikace v podnikové a mezipodnikové praxi - 3., aktualizované vydání.* místo neznámé : Grada Publishing a.s., 2015. ISBN: 9788024799186.

Gube, Jacob. 2010. What Is User Experience Design? Overview, Tools And Resources. [Online] 2010. [Citace: 13. Březen 2017.] https://www.smashingmagazine.com/2010/10/what-is-user-experience-design-overview-tools-and-resources/.
Hejna, Dalibor. 2007. *Diplomová práce.* Brno : autor neznámý, 2007.
Hrušková, Ela. 2017. TOGAF Část 3. [Online] 2017. [Citace: 7. Březen 2017.] http://slideplayer.cz/slide/2789653/.
Chování.eu. 2017. Řízení inovací. [Online] 2017. [Citace: 5. Duben 2017.] http://www.chovani.eu/img/images/203.png.
IIBA. 2015. *A Guide to the Business Analysis Body of Knowledge® (BABOK® Guide): 3.* místo neznámé : International Institute of Business Analysis, 2015. ISBN: 978-1927584026.
Jak na webové stránky. 2016. Bcrypt. [Online] 22. Listopad 2016. [Citace: 27. Únor 2017.] http://timehosting.cz/bcrypt/.
Janiš, Petr. 2015. PMI-PMBOK® 5th edition in pictures. [Online] 18. Červen 2015. [Citace: 8. Březen 2017.] http://www.projectman.cz/en/clanky/posts/66-pmi-pmbok-5th-edition-in-pictures.
Jonák, Stanislav. 2016. Pracovní doba v agilním světě. [Online] 12. Červenec 2016. [Citace: 4. Březen 2017.] http://www.middleware.cz/projektove-rizeni0/31-pracovni-doba-v-agilnim-svete.
Julinek, Bc. Pavel. 2008. Použití RUP pro malé SW projekty. [Online] 2008. [Citace: 2. Březen 2017.] http://is.muni.cz/th/72639/fi_m/diplomova_prace.pdf.
Keřkovský, Miloslav a Valsa, Ondřej. 2012. *Moderní přístupy k řízení výroby, 3. doplněné vydání.* místo neznámé : C.H.BECK, 2012. ISBN: 978-80-7179-319-9.
Košturiak, Ján a Frolík, Zbyněk. 2006. *Štíhlý a inovativní podnik.* místo neznámé : Alfa Publishing, 2006. ISBN: 80-86851-38-9.
Kotačka, Vít. 2014. Kanban, lehký úvod . [Online] 4. Březen 2014. [Citace: 25. Březen 2017.] http://www.sw-samuraj.cz/2014/03/kanban-lehky-uvod.html.
Lapoš, Jozef a Kostka, Robert. 2006. Zabezpečení dat v manažerských informačních systémech. [Online] 2006. [Citace: 10. Květen 2014.] https://www.systemonline.cz/it-security/zabezpeceni-dat-v-manazerskych-informacnich-systemech.htm.
Lightbend. 2017. Lightbend. *The Play Framework at LinkedIn.* [Online] 2017. [Citace: 1. Duben 2017.] https://www.lightbend.com/resources/case-studies-and-stories/the-play-framework-at-linkedin.
MÁČEL, Lukáš. 2009. *Modelování podnikových procesů.* Fakulta informačních technologií ústav počítačových systémů, Vysokého učení technického v Brně. Brno : Fakulta informačních technologií, 2009. Seminární práce.
Máchal, Pavel, Kopečková, Martina a Presová, radmila. 2015. *Světové standardy projektového řízení: pro malé a střední firmy.* místo neznámé : Grada Publishing a.s., 2015. ISBN: 9788024797052.
Malkusová, Tereza. 2016. Je lepší vyvíjet vodopádem nebo scrumem? [Online] 23. Květen 2016. [Citace: 4. Březen 2017.] https://www.aitom.cz/co-je-noveho/je-lepsi-vyvijet-vodopadem-nebo-scrumem.

Malý, Martin. 2011. Několik poznámek k heslům. [Online] 20. Červenec 2011. [Citace: 27. Únor 2017.] https://www.zdrojak.cz/clanky/nekolik-poznamek-k-heslum/.
ManagementMania. 2016. DMAIC - cyklus zlepšování (Improvement Cycle). [Online] 22. Červen 2016. [Citace: 03. Březen 2017.] https://managementmania.com/uploads/article_image/image/44/.png.
Mašín, Ivan. 2003. *Mapování hodnotového toku ve výrobních procesech.* Liberec : Institut průmyslového inženýrství, 2003. ISBN: 80-902235-9-1.
MBI. 2013. Metoda : Project Management Body of Knowledge (PMBOK). [Online] 2013. [Citace: 8. Březen 2017.] http://mbi.vse.cz/public/cs/obj/METHOD-13.
Mordánky. 2011. Proč používat framework. [Online] 20. Září 2011. [Citace: 10. Květen 2017.] http://mordanky.blogspot.cz/2011/09/proc-pouzivat-framework.html.
Page, Alan, Rollison, Bj a Johnston, Ken. 2017. *Jak testuje software Microsoft.* místo neznámé : Computer Press, Albatros Media a.s., 2017. ISBN: 9788025140574.
Pavlík, Michal. 2012. Vývojové modely. [Online] 2012. [Citace: 20. Únor 2017.] http://www.umel.feec.vutbr.cz/bdts/images/mmu/modelVodopad.PNG.
Plechatý, Radek. 2013. Analýza požadavků a návrh IS - případová studie. [Online] 16. Červen 2013. [Citace: 3. Březen 2017.] https://www.vse.cz/vskp/36586_analyza_pozadavku_a%C2%A0navrh_is___pripadova_studie.
Programujeme vlastní sociální síť. Peacock, Michael. 2016. místo neznámé : Computer Press, Albatros Media a.s., 2016. ISBN: 9788025140901.
Rábová. 2016. Informačná systémy. [Online] 2016. [Citace: 1. Březen 2017.] https://is.mendelu.cz/eknihovna/opory/index.pl?cast=4758.
Radigan, Dan. 2017. The kanban methodology. *Kanban.* [Online] 2017. [Citace: 4. Duben 2017.] https://wac-cdn.atlassian.com/dam/jcr:1b826f38-fb78-45ce-b352-03ccb108e3ba/Create-non-JIRA-specific-kanban-board-image-%2520(1).png?cdnVersion=dz.
Rejnková, Petra. 2009. Příklady použití diagramů UML 2.0. *UML.* [Online] 2009. [Citace: 17. 8 2016.] http://uml.czweb.org/index.html.
Roudenský, Petr a Havlíčková, Annd. 2017. *Řízení kvality softwaru.* místo neznámé : Albatros Media a.s., 2017. ISBN: 9788025145197.
Řepa, Václav. 2007. *Podnikové procesy: procesní řízení a modelování.* 2., aktualiz. a rozš. vyd. Praha : Grada, 2007. str. 281. 978-80-247-2252-8.
Schwalbe, Kathy. 2016. *Řízení projektů v IT.* místo neznámé : Computer Press, Albatros Media a.s., 2016. ISBN: 9788025147788.
Sommerville, Ian. 2017. *Softwarové inženýrství.* místo neznámé : Computer Press, Albatros Media a.s., 2017. ISBN: 9788025145258.
Stachecki, Filip. BPMN 2.0 dla Analityków Biznesowych. [Online] [Citace: 10. Květen 2017.] https://training-course-material.com/images/7/7d/BPMNBasicSymbols.png.
Svozilová, Alena. 2011. *Projektový management: Systémový přístup k řízení projektů.* místo neznámé : Grada Publishing a.s., 2011. ISBN: 8024774283, 9788024774282.
—. 2011. *Zlepšování podnikových procesů.* místo neznámé : GRADA Publishing, a.s., 2011. ISBN: 978-80-247-3938-0.

Šedivá, Zuzana. 2009. *Podniková informatika - 2., přepracované a aktualizované vydání.* místo neznámé : Grada Publishing a.s., 2009. ISBN: 9788024789354.
Šimerda. 2005. Principy UML. [Online] 2005. [Citace: 27. Únor 2017.] http://fei.mtrakal.cz/materialy_public/7.semestr/%5B2010-2011%5DINPSW_Simerda/prednasky/01_D_UMLprinciples.pdf.
Šimon, Michal a Miller, Anotnín. Kanban – výroba tahem. [Online] [Citace: 10. Květen 2017.] https://www.systemonline.cz/rizeni-vyroby/kanban-vyroba-tahem.htm.
Šochová, Zuzana a Kunce, Eduard. 2014. *Agilní metody řízení projektů.* místo neznámé : Computer Press, 2014. ISBN: 978-80-251-4194-6.
Švecová, Lenka. 2016. *Hodnocení výkonnosti akademických pracovníků Masarykova ústavu vyšších studií ČVUT v Praze - kvantifikovatelné ukazatele.* Praha : Masarykův ústav vyšších studií, 2016. Hodnocení výkonnosti akademických pracovníků Masarykova ústavu vyšších studií ČVUT v Praze - kvantifikovatelné ukazatele.
Toyota Material Handling CZ s.r.o. 2010. Výrobní systém Toyota TPS a jeho přínosy pro podnikání. [Online] Duben 2010. [Citace: 5. Duben 2017.] http://www.toyota-forklifts.cz/sitecollectiondocuments/tps_nahled.pdf.
Vašíček, Petr. 2008. 3. část: Úvod do BPMN. [Online] 2008. [Citace: 10. Květen 2017.] http://bpm-sme.blogspot.cz/2008/03/3-uvod-do-bpmn.html.
Veber, Jaromír. 2006. *Řízení jakosti a ochrana spotřebitele.* místo neznámé : Grada Publishing a.s., 2006. ISBN: 9788024766645.
Vochozka, Marek a Mulač, Petr. 2012. *Podniková ekonomika.* místo neznámé : GRADA Publishing, a.s., 2012. ISBN: 978-80-247-4372-1.
Žoltá, Lucie. 2016. UML (Unified Modeling Language). [Online] 2016. [Citace: 03. Březen 2017.] http://lucie.zolta.cz/images/skola/Softwarovelng/uml.jpg.

Seznam obrázků

Seznam tabulek

Evidence výpůjček

Prohlášení:
Dávám svolení k půjčování této diplomové práce. Uživatel potvrzuje svým podpisem, že bude tuto práci řádně citovat v seznamu použité literatury.

Jméno a příjmení: Martin Tyrol
V Praze dne: 19. 05. 2017 Podpis:

Jméno	Oddělení/ Pracoviště	Datum	Podpis