VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

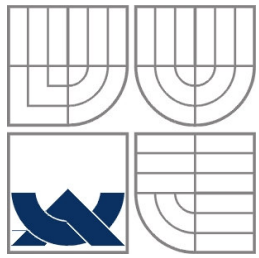
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÝ SYSTÉM ŠKOLIACEHO STREDISKA

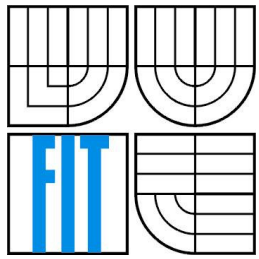
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. LADISLAV RUTTKAY
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM ŠKOLICÍHO STŘEDISKA

INFORMATION SYSTEM OF A LEARNING CENTER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. LADISLAV RUTTKAY

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADOMÍR KUREČKA

BRNO 2007

Zadání diplomové práce

Informační systém školicího střediska

Learning Center Information System

Vedoucí:

Kurečka Radomír, Ing., UIFS FIT VUT

Oponent:

Květoňová Šárka, Ing., UIFS FIT VUT

Přihlášen:

Ruttkay Ladislav, Bc.

Zadání:

1. Seznamte se s požadavky na typický informační systém školicího střediska.
2. Navrhněte třívrstvou architekturu tohoto systému.
3. V prostředí Microsoft ASP.NET implementujte tento systém včetně zákaznické části.
4. Navrhněte a zrealizujte vybraná možná propojení s externími systémy.
5. Zhodnoťte výsledný systém z hlediska praktického použití v reálném prostředí.

Část požadovaná pro obhajobu SP:

1. Seznam funkčních požadavků na systém.
2. Návrh třívrstvé architektury.

Kategorie:

Softwarové inženýrství

Implementační jazyk:

C#

Operační systém:

Windows XP

Komerční software:

Visual Studio .NET 2005

Literatura:

- Kačmář, D.: Programujeme .NET aplikace, Computer Press 2001
- Bishop, J., Horspool, N.: C# Concisely, Pearson Addison Wesley, 2004

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologii Vysokého učeni technického v Brně.

Abstrakt

Diplomová práca popisuje realizáciu informačného systému školiaceho strediska. Pojednáva o kompletnom priebehu vývoja systému a perspektívach jeho rozšírení. Podstatná časť práce je venovaná návrhu architektúry systému, špecifikácii zadávateľa, vytvoreniu modelov, a diagramov. Nemenej významná časť je venovaná samotnej implementácii, využívaniu špeciálnych programovacích prostriedkov a postupov. V neposlednom rade je vyzdvihnutá štandardizácia pracovného postupu implementácie konkrétnych častí systému, vďaka ktorej je systém jednoducho pochopiteľný a poskytuje možnosť rozšírenia funkčnosti aj v budúcnosti. Niektoré kapitoly taktiež pojednávajú o vytvorení rozhrania, poskytujúcom základnú funkčnosť, ktoré môže byť použité aj v iných informačných systémoch. Práca sa okrem iného zmeriava aj na spoluprácu s externými systémami a na využitie technológií na podporu ich vzájomnej komunikácie.

Klíčová slova

Informační systém, Objektový přístup, Vrstvová architektura, Školící středisko, .NET Framework

Abstract

This work describes the realization of information system of a learning center. It handles about the development and perspectives of its further development. The main part of it is dedicated to the development of system architecture, specification, creation of models and diagrams. An other part handles about the implementation, usage of programming methods and techniques. Highly stressed is the standardization of programming workflow, which enables easy understanding and future development of the system. Some parts handle about the developed framework, which provides the basic functionality, that can be used in other information systems as well. The work also describes the solution of the cooperation with other external systems and the used technology.

Keywords

Information system, Object architecture, Layer architecture, Learning center, .NET Framework

Citace

Ladislav Ruttkay: Informačný systém školiaceho strediska, diplomová práce, Brno, FIT VUT v Brně, 2007

Informačný systém školiaceho strediska

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Radomíra Kurečku.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ladislav Ruttkay
14.5.2007

Obsah

1 Úvod

Informačné systémy predstavujú podstatné odvetvie informatiky. S ich využitím sa stretávame pomerne často a v rôznych podobách. Hlavnou úlohou každého informačného systému je priniesť svojim nasadením isté výhody, či sa už jedná o konkurenčné, finančné alebo riadiace. Firmy implementujú a nasadzujú informačné systémy v súlade so svojou firemnou a konkrétne informačnou stratégiou. Pre každú firmu, ktorá uvažuje o nasadení informačného systému je dôležité analyzovať a správne konkretizovať požiadavky aby nedošlo k zbytočnému nadhodnoteniu alebo podhodnoteniu informačného systému. Toto odvetvie má v súčasnosti v rámci informatiky značnú popularitu o čom svedčí taktiež počet študentoch na odbore informačných systémov.

Informačný systém o ktorom pojednáva táto práca, predstavuje obdobu systémov navrhovaných firmami na zákazku. Tieto systémy sú často pod náročným časovým harmonogramom čo prináša sebou unáhlený design bez predošlej dôkladnej analýzy. Implementácia takéhoto systému potom je často robená v editore, ktorý sám generuje kód podľa definovaného užívateľského rozhrania. Tento prístup však sebou nesie značné nedostatky v oblasti rozšíriteľnosti a bezpečnosti výsledného systému. V tu popisovanom systéme sa jedná o značne uvedomelý prístup, ktorý nesie sebou prepracovaná analýzu, design a taktiež kódovú architektúru systému. Tieto výhody zabezpečujú značnú rozšíriteľnosť a taktiež použiteľnosť implementovaných častí, ktoré tvoria istú podobu frameworku, znovu nasaditeľného v iných informačných systémoch.

Diplomová práca predstavujú pokračovanie bakalárskej práce, ktorá obsahovala konceptuálny návrh informačného systému vo svojej základnej funkčnosti. Práca už pojednáva o nasadení tohto systému a riešení problémov, ale taktiež predovšetkým o následnej implementácií a udržiavaní systému. Taktiež rieši spoluprácu s ďalšími externými systémami, a vytvorenie znovu použiteľného frameworku. V oblasti procesov sa taktiež zameriava na náročnú implementáciu zákazníckej časti a vyhoveniu reálnym požiadavkám zákazníkov školiaceho strediska.

Cieľom práce bolo vytvorenie reálneho, značne rozsiahleho informačného systému. Práve jeho nasadenie prinieslo a prináša sebou značné praktické problémy, ktoré je nutné riešiť, často pod časovým tlakom. Tieto všetky faktory usmerňovali vývoj projektu za účelom ich efektívneho splnenia. Navrhnutý spôsob riešenia práve umožňuje efektívnu realizáciu údržby, pridania ďalších modulov, ale taktiež tvorby ďalších systémov.

Faktormi vstupujúcimi do riešenia danej problematiky, boli procesy fungujúce v reálnom školiacom stredisku a taktiež požiadavky na ich zlepšenie a zefektívnenie. Taktiež nutná bola kooperácia a podpora už fungujúcich systémov. Výhodou bolo, že žiaden systém nebol nasadení a preto nebolo nutné riešenie prevodu dát na nový systém.

Písomná časť diplomovej práce sa skladá zo šiestich základných tematických častí. Prvá sa zameriava na použité technológie a postupy. Ďalšia časť sa zameriava na analýzu procesov vo firme, ktoré mali byť implementované. Nasledujúca časť pojednáva o realizácií projektu a o samotnej implementácií za použitia popísaných technológií. Ďalšie dve časti sa zaoberajú návrhom a implementáciou frameworku a taktiež spolupráci s externými systémami. Posledná časť sa venuje úspešnosti nasadenia.

Kapitoly:

1. **Úvod**

   Uvedenie do problematiky a popis kapitol.

2. **Jazyk UML**

   Kapitola sa zoberá stručným popisom častí UML diagramov, ktoré budú používané v celej práci.

3. **Objektový prístup**

   Hlavnou témou tejto časti je vyzdvihnutie objektového prístupu, popis paradigiem a ich prínos do architektúry systému.

4. **Project management**

   Kapitola pojednáva o teoretických základoch návrhu projektu, predovšetkým postupnosti krokov, vedúcich k úspešnému výsledku.

5. **Využívané technológie**

   Na tomto mieste sa nachádza prehľad a popis využívaných prostriedkov pri realizácii projektu. V tejto časti sa nachádza taktiež popis technológie webových služieb (web services), ktorá bola využitá na spoluprácu s externými systémami.

6. **Analýza a riešenie procesov vo firme**

   Kapitola sa zameriava na analýzu pracovných procesov a reálnych postupov vo firme, ktoré majú byť implementované a taktiež ich riešením, prípadne zefektívnením zavedením informačného systému.

7. **Realizácia projektu**

   Kapitola už pojednáva o konkrétnej realizácií. Obsahuje praktickú ukážku použitia návrhových technológii popísaných v predošlých kapitolách. Jedna sa predovšetkým o špecifikáciu požiadaviek projektu, jeho analýzu a design. Kapitola sa taktiež zaoberá samotnou implementáciou projektu, návrhom štruktúry programového riešenia a použitých techník.

8. **Framework systému**

   Zameranie tejto časti spočíva v popise vytvoreného rozhrania, ktoré v sebe zahrňuje základné triedy, ktorých funkčnosť môže byť použitá v ďalších projektoch a tým značne zrýchliť ich realizáciu.

9. Spolupráca s externými systémami

Časť je zameraná na popis vytvorenej spolupráce s externými systémami, ktoré často používajú iné technológie.

10. Nasadenie systému

Kapitola pojednáva o procesných problémoch odhalených počas ostrého behu systému a taktiež a forme jeho nasadenia do prevádzky.

11. Záver

Kapitola pojednáva o zatiaľ dosiahnutých výsledkoch projektu, získaných poznatkoch plánoch ďalšieho rozvoja projektu.

2 Jazyk UML

V softwarovom inžinierstve, UML (Unified Modeling Language) je jazyk modelovania a špecifikácie. UML nie je obmedzený iba na modelovanie softwaru, ale umožňuje vytváranie taktiež hardwarových návrhov, pracovných procesov a štruktúr organizácie. Jazyk poskytuje zjednotené výrazové prostriedky, ktoré uľahčujú a štandardizujú komunikáciu medzi osobami podieľajúcimi sa na projekte. UML umožňuje modelovať jednoduché i zložité aplikácie pomocou rovnakého formálneho syntaxu. V súčasností sa začína presadzovať UML verzie 2, ktoré prináša sebou niektoré nové typy diagramov.

Účelom tejto kapitoly je predstavenie silného nástroja, ktorého použitie značne zefektívňuje návrh informačných systémov a jeho pochopenie je podstatné a vyžadované v praxi. Pri implementácií tohto projektu som tieto nástroje využíval. Niektoré návrhy je možné vidieť v ďalších kapitolách prípadne prílohách.

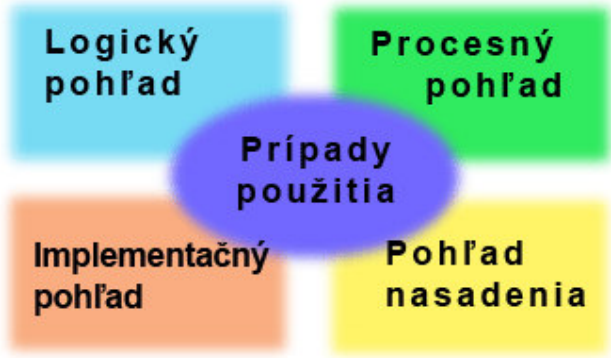
2.1 Pohľady na systém

Model UML sa skladá z rôznych diagramov, ktoré predstavujú pohľady na časti sémantického základu navrhovanej aplikácie. Sémantickým základom sa chápe súhrn špecifikácií aplikácie, vymedzujúci teritórium, v ktorom sa môže návrh pohybovať. Diagram vo svojej vizuálnej podobe sa zoberá konkrétnou časťou aplikácie. Žiadny dvojrozmerný diagram však nemôže zachytiť komplexnú aplikáciu v celku, ale sústredí sa stále na jeden dôležitý aspekt. Jazyk UML rozoznáva päť základných pohľadov na systém. [Stein-2005]

- ***Pohľad prípadov použitia:***
  V prípadoch použitia sú vyjadrené základné požiadavky kladené na systém. Všetky ďalšie pohľady sa môžu pohybovať len v medziach prípadov použitia.
- ***Logický pohľad:***
  Ukazuje pomocou objektov a tried statickú štruktúru programového riešenia. Obsahuje Základné moduly a väzby medzi modulmi a tým identifikuje vrstvy a ich jednotlivé rozhrania.
- ***Procesný pohľad:***
  Sa sústreďuje na chovanie systému, ktoré musí spĺňať požiadavky a obmedzenia prípadov použitia, ktoré sú kladené na priebeh procesov. Popisuje štruktúru procesov a paralelizmi v spracovaní.
- ***Implementačný pohľad:***
  Je zameraný na fyzické rozdelenie aplikácie na komponenty a ich vzájomné závislosti.

- ***Pohľad nasadenia:***

  Určuje fyzickú topológiu systému – procesory a zariadenia.



Obrázok 2.1 Prehľad pohľadov na UML

Pohľady sú konkretizované v typoch diagramov, ktorými sa bude zaoberať nasledujúca kapitola.

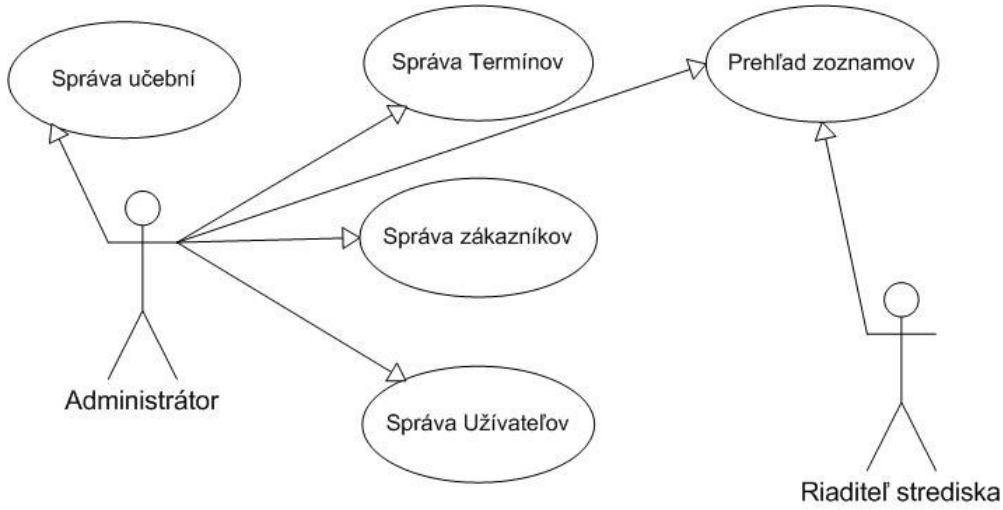
2.2 Typy diagramov

V tejto časti sa nachádza prehľad základných diagramov používaných v UML. Podrobnejšie je význam jednotlivých diagramov pre návrh systému rozobraný v kapitole *Project management.* [Ariad-2005] [Zendu-2006]

2.2.1 Diagram prípadov použitia

Diagram prípadov použitia je opisom chovania systému z pohľadu užívateľa. Tento graf predstavuje značnú pomoc počas analýzy, kde vývoj diagramov prípadov použitia pomáha porozumieť požiadavkám zadávateľa.

Diagram umožňuje vďaka svojej jednoduchosti, analytikom, designerom, programátorom, testerom a zákazníkovi spolu komunikovať.



Obrázok 2.2 Diagram prípadov použitia

Aj keď sa zdajú diagramy prípadov použitia elementárne, predstavujú veľmi silnú a nenahraditeľnú časť návrhu. Diagramy prípadov požití je často nutné rozložiť na podrobnejšie prípady použití. Tomuto procesu sa hovorí granularita. Je však nutné granularita správne odhadnúť aby nedošlo k zneprehľadneniu diagramu. Taktiež je možné sa často stretnúť s popisom prípadu použitia pomocou diagramu aktivít, alebo modelom prípadu použitia, ktorý je reprezentovaný tabuľkou zobrazujúcou hlavný a alternatívne toky.

Charakteristiky diagramov prípadov použití:

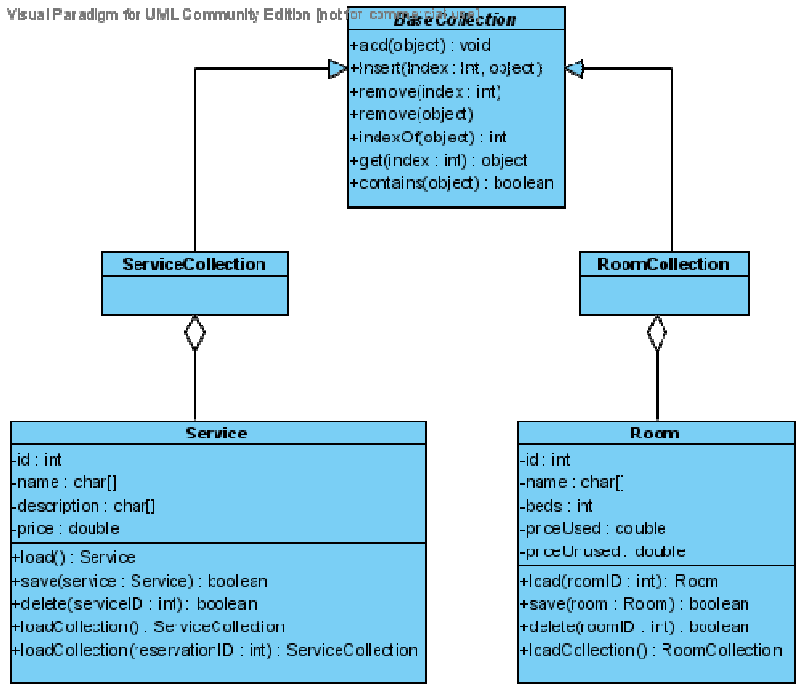
- Diagramy prípadov použitia definujú rozsah systému. Umožňujú vizualizáciu veľkosti návrhu projektu.
- Sú podobné požiadavkám, ale na rozdiel od nich sú viac formálne a prehľadné.
- Súčet všetkých diagramov použitia predstavuje kompletný systém, čo znamená, že všetko čo nie je v diagrame, je mimo vyvíjaného systému.
- Vďaka svojej jednoduchosti umožňujú komunikáciu so zákazníkom
- Vedú team počas procesu vývoja a mali by predstavovať kostru systému, na ktorú sa bude odkazovať každá činnosť.
- Majú využitie pri plánovaní časového rozvrhu projektu.
- Tvoria základ pre testovanie.
- Pomáhajú vytvoriť užívateľskú príručku systému.

2.2.2 Diagram tried

Diagram tried je nevyhnutným aspektom objektovo orientovaného návrhu. Slúži ako návrh programovej štruktúre, ktorá ho bude prakticky kopírovať, ale vo fáze vývoja diagramu tried sa ešte programovou štruktúrou nezaoberáme. Diagram tried je používaný pre plánovanie konceptov, ktorým

zákazník rozumie, preto sa mu niekedy hovorí *Konceptuálny model*. Spolu s diagramom prípadov použitia predstavuje konceptuálny model silný nástroj projektovej analýzy.

Diagram tried však samozrejme nemusí korešpondovať s dátovým návrhom schémy databáze. Tento diagram často predstavuje logický pohľad na funkčnú architektúru systému, ktorá je navrhnutá na podporu prípadov použitia. Ich funkčná realizácia a implementácia sa rieši až v ďalších fázach vývoja projektu. Niektoré triedy dokonca ani nemajú svoju fyzickú realizáciu v databáze, ako je napr. v nasledujúcom obrázku trieda *ServiceCollection, RoomCollection,* alebo abstraktná trieda *BaseCollection.*



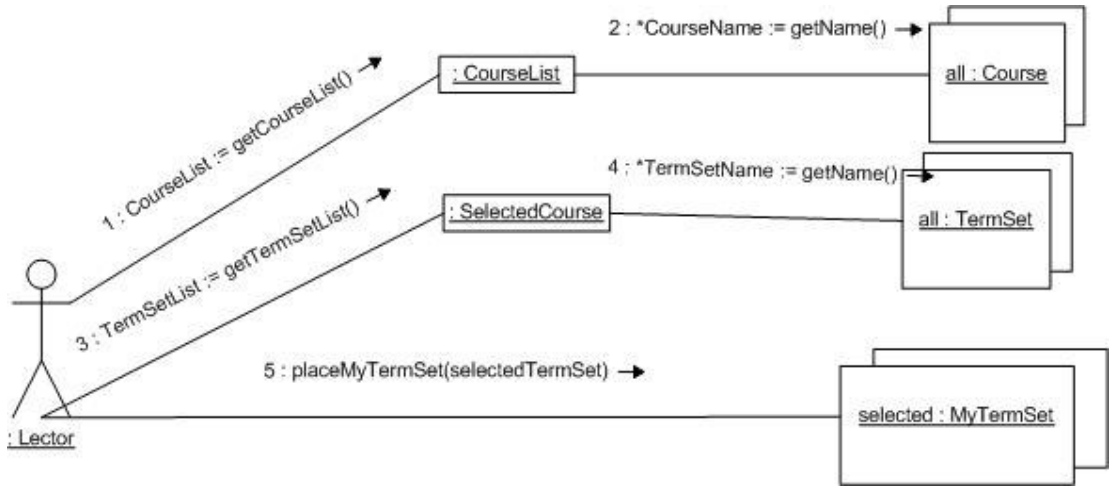
Obrázok 2.3 Konceptuálny model

2.2.3 Diagramy interakcií

Predstavujú hlavnú techniku úrovne návrhu. Pri analýze požiadaviek ešte nie sú triedy presne zrejme a špecifikované a preto je ich nutné spolu so vzájomnou komunikáciou za zachovania požadovanej funkčnosti navrhnúť. UML 2.0 rozlišuje štyri typy diagramov interakcii: *Diagram komunikácie, Sekvenčný diagram, Prehľadový diagram interakcií, Diagram časovania.* Práve diagram komunikácie nahradil diagram spolupráce (Collaboration diagram) z verzie UML 1.x.

2.2.3.1 Diagram komunikácie

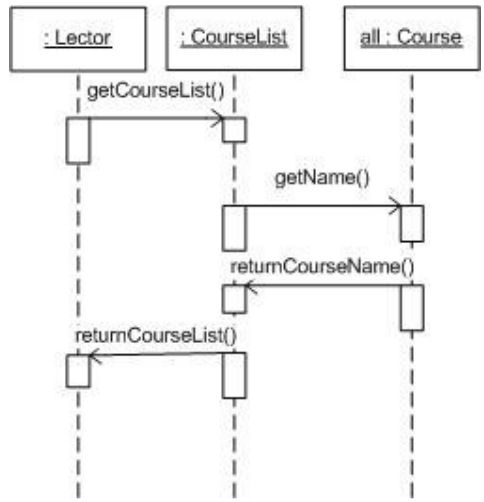
Vzhľadom na to, že vyvíjame objektovo orientovaný software, všetky funkcie softwaru je nutné získať ako výsledok vzájomnej spolupráce objektov. Diagram komunikácie znázorňuje aké máme požiadavky na kooperáciu objektov. Na rozdiel od *Sekvenčného diagramu* reprezentuje statickú štruktúru, ktorá sa využije pri interakcii k dosiahnutiu požadovaného chovania daného napr. prípadom použitia.



Obrázok 2.4 Diagram komunikácie

2.2.3.2 Sekvenčný diagram

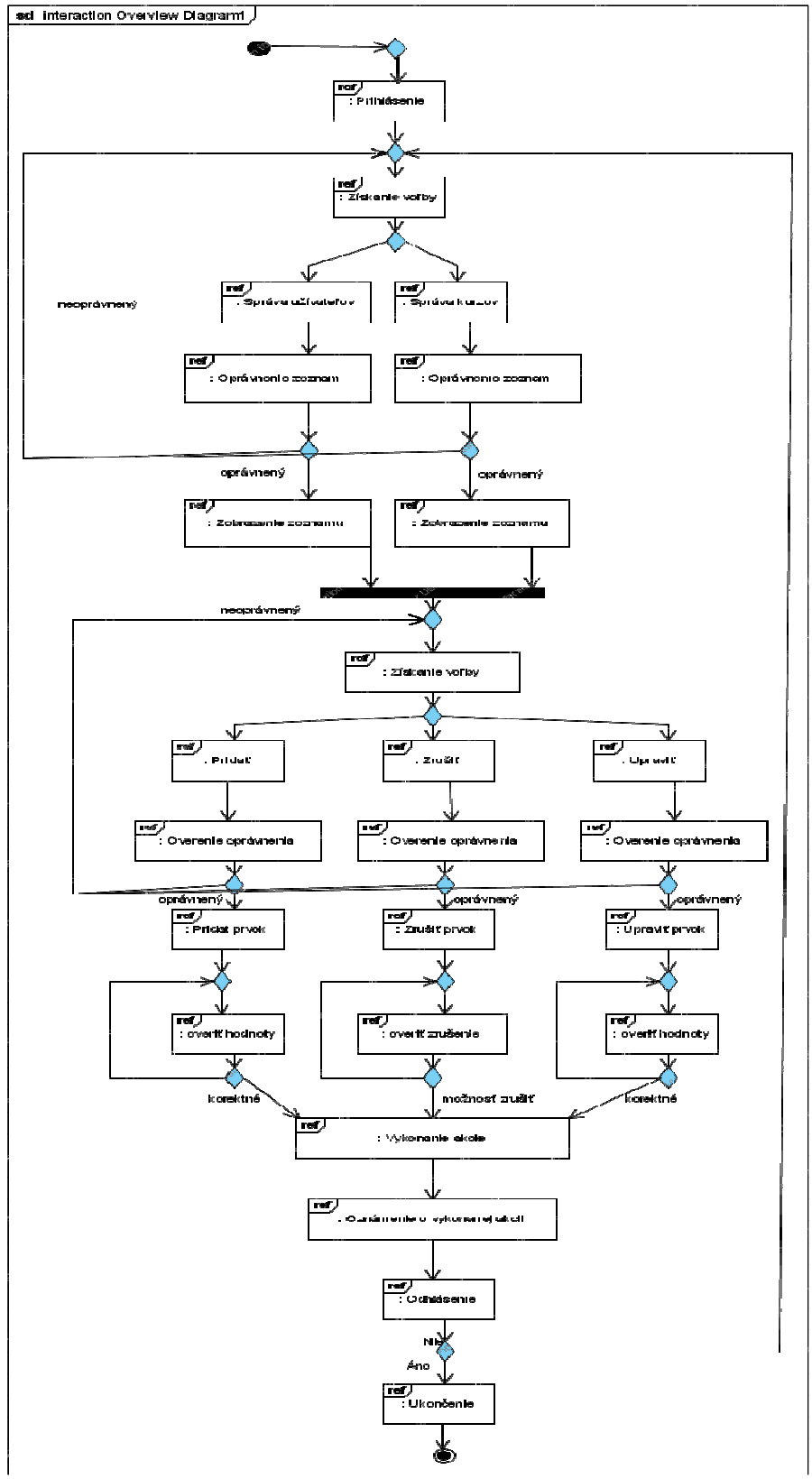
Sekvenčný diagram je priamo spojený s *diagramom komunikácie* a v podstate znázorňuje rovnaké informácie použitím inej formy. Diagram zobrazuje interakciu objektov v čase, kde čiarkované zvislé čiary znázorňujú čas.



Obrázok 2.5 Sekvenčný diagram

2.2.3.3 Prehľadový diagram interakcií

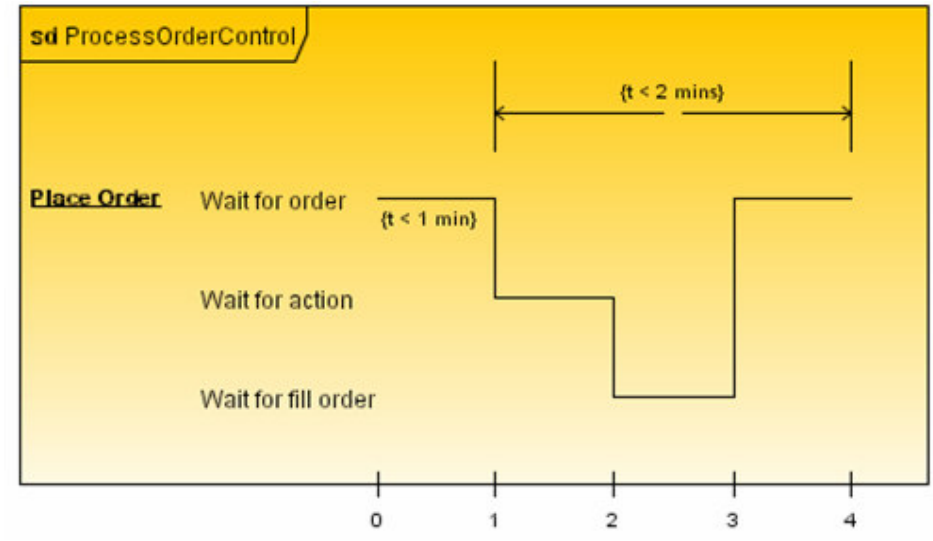
Predstavuje špeciálnu formu diagramu aktivít, ale ukazuje interakcie a ich výskyt a slúži predovšetkým na modelovanie toku riadenia. Predovšetkým je vhodný na modelovanie následnosti jednotlivých prípadov použitia a ich vzájomnej interakcie. Často sa používa taktiež na modelovanie jednotlivých prechodov medzi obrazovkami systému, ktoré reprezentujú jednotlivé prípady použitia. Takýmto diagramom, môže byt napr. reprezentovaná aktivita užívateľa, ktorý chce vytvoriť kurz, ktorá bude reprezentovaná postupnosťou: *Prihlásenie do systému, Voľba nový kurz, Vyplnenie údajov, Voľba uložiť kurz, Odhlásenie zo systému.* Prehľadový diagram interakcií používa rovnaký syntax ako diagram aktivít.



Obrázok 2.6 Prehľadový diagram interakcií

2.2.3.4 Diagram časovania

Predstavuje nový typ diagramu v UML 2.0. Má za úlohu odstrániť nedostatky predchádzajúcich verzií UML pri zobrazovaní behu systémov v reálnom čase. Reprezentuje časový priebeh ukazujúci doby trvaní určitých stavov objektu a taktiež dokáže reprezentovať vzájomné interakcie objektov v čase a vplyv na ich stav.

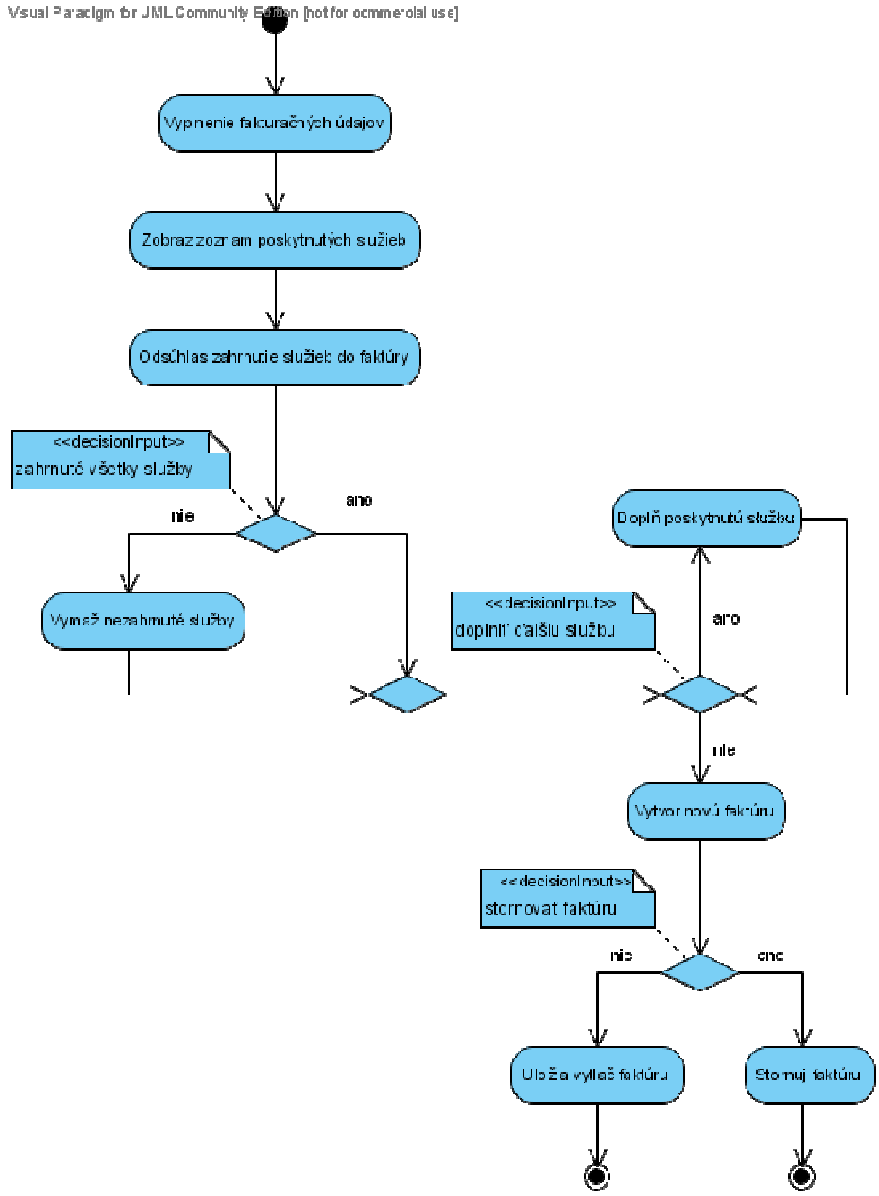


Obrázok 2.7 Diagram časovania[1]

[1] Zdroj: 24.4.2007 http://www.visual-paradigm.com

2.2.4 Diagram aktivít

Predstavuje objektovo orientovaný vývojový diagram. Vo verzií UML 1.x bol reprezentovaný schémou automatu, ale vo verzii 2.0 sa reprezentuje prostredníctvom Petriho sietí. Často sa používa na modelovanie business procesov, workflow, dátových tokov a operácií. Taktiež sa uplatňuje pri zobrazení toku aktivít konkrétneho prípadu použitia. Na rozdiel od iných diagramov, nereprezentuje len statická štruktúru objektov a tried, ale popisuje aj ich chovanie.

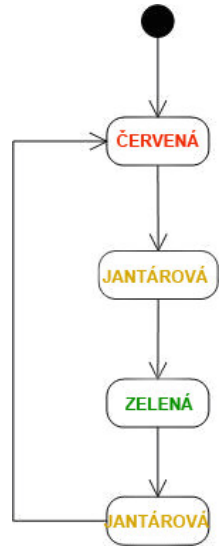


Obrázok 2.8 Diagram aktivít

2.2.5 Stavový diagram

Niektoré objekty môžu byť v čase v určitom stave. Napr. semafor môže nadobúdať jeden zo stavov: Vypnutý (Off), Červená, Jantárová, Zelená.

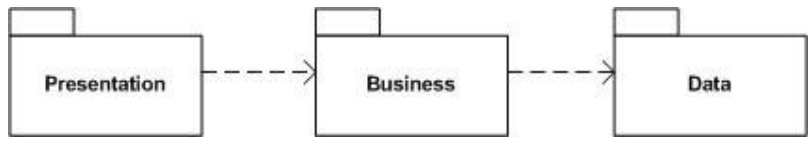
Sekvencia prechodu objektu do jednotlivých stavov môže byť značne komplexná. V našom prípade nesmie semafor zmeniť svoj stav zo „Zelená“ na „Červená“ ale musí prejsť cez stav „Jantárová“. V niektorých prípadoch je zachovávanie správnej postupnosti stavov nevyhnutné i keď príklad so semaforom vyzerá triviálne.



Obrázok 2.9 Stavový diagram

2.2.6 Diagram balíkov

Rozsiahlejšie, netriviálne systémy je nutné rozdeliť do menších, ľahšie pochopiteľnejších jednotiek (package, balíkov). Tieto časti majú v sebe združené triedy, ale diagram v UML znázorňuje len ich zabalenie a vzájomnú závislosť jednotlivých balíkov. Tento diagram je prostriedkom zoskupovania prvkov modelov. Každý balíček má vlastný menný priestor a v jeho rámci musia byť mena unikátne. Diagram balíkov predstavuje však logické zoskupenia, na vyjadrenia fyzického zoskupenia je nutné použiť *diagram komponent*.

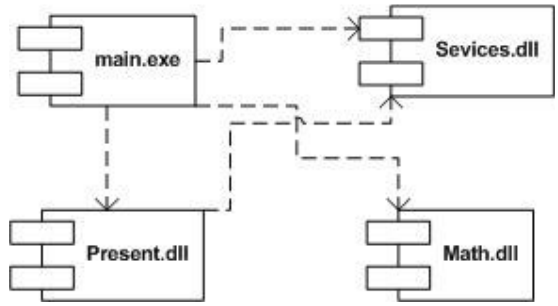


Obrázok 2.10 Diagram balíkov

2.2.7 Diagram komponent

Diagram komponent je podobný *Diagramu balíkov* a podobne ako on znázorňuje rozdelenie systému na jednotlivé časti a ich závislosti. Tento graf však znárodňuje závislosti fyzických softwarových komponent (súbory, knižnice, balíky). Podľa UML 2.0 komponenta reprezentuje modulárnu časť

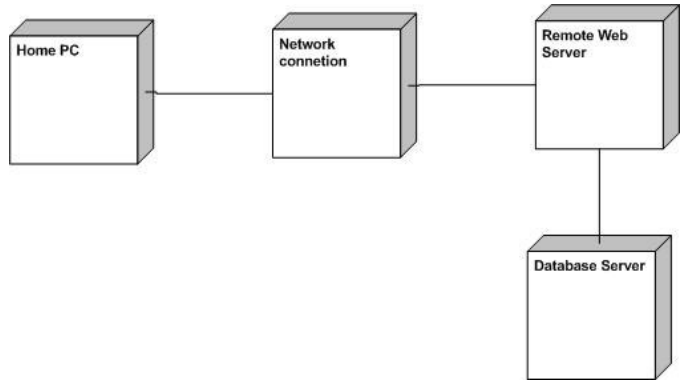
systému, ktorá zapuzdruje svoj obsah a jej prejav je nahraditeľný v jej okolí. Komponentu je možné si predstaviť ako čiernu skrinku, ktorej vonkajšie chovanie je reprezentované rozhraním, ktoré poskytuje a reprezentuje. Komponenta teda môže byť nahradená inou, ktorá podporuje rovnaké rozhranie.



Obrázok 2.11 Diagram komponent

2.2.8 Diagram nasadenia

UML týmto diagramom umožňuje návrh reálneho rozmiestenia častí softwaru. Diagram nasadenia v UML 2.0 ukazuje rozmiestnenie artefaktov a nie komponent. Každý artefakt však môže v sebe obsahovať komponenty alebo celý diagram komponent.



Obrázok 2.12 Diagram nasadenia

3 Objektový prístup

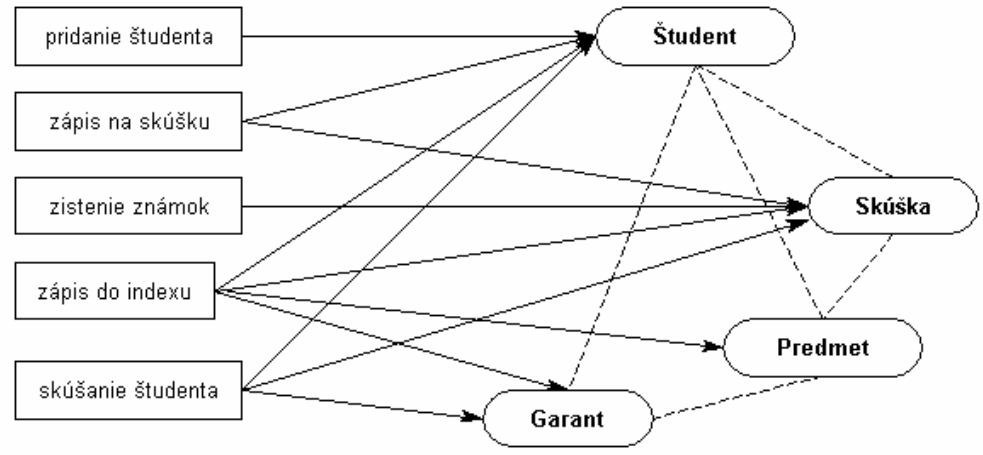
V kapitole sú vyzdvihnuté vlastnosti objektového prístupu v problematike informačného systému. Nachádza sa tu taktiež porovnanie z často, najmä v minulosti používaným, štruktúrovaným programovaním. [Ariad-2005]

3.1 Štruktúrované programovanie

Podstatou štruktúrovane orientovaného programovanie je analýza problému a navrhnutie sady funkcii, ktoré ho riešia. Tento prístup je výhodný pri malej funkcionalite a jednoduchých systémoch. Pri väčších projektoch však návrh znemožňuje v svojom rozsahu teamovú spoluprácu a chápanie komplexného systému.

Väčšia funkcií vyžaduje dáta špecifického typu s ktorými pracuje. Problém však nastáva vo vzájomnej závislostí funkcii a dát.. Ak potrebujeme zmeniť typ dát, tak nutne musíme zmeniť tieto záznamy vo všetkých funkciách. Systém sa pridávaním funkcií stáva značne komplexným a neprehľadným.

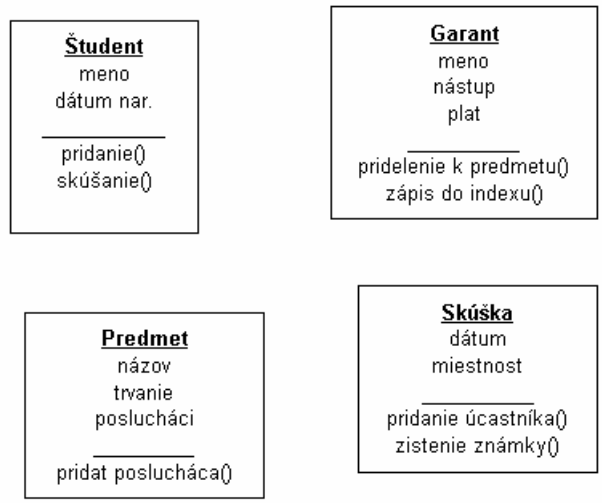
Nasledujúci príklad predstavuje štruktúrovaný prístup k systému predmetov, študentov, garantov a skúšok. Prerušované čiary predstavujú vzájomné vzťahy, kde jedna skupina dát je závislá na druhej. V následnom príklade sú uvedené funkcie, ktorá ma systém vykonávať. Z obrázku je zrejmá značná previazanosť funkcií s dátami.



Obrázok 3.1 Štruktúrované programovanie

3.2 Objektovo orientovaný prístup

Snahou objektového prístupu je riešiť uvedený problém kombináciou dát a funkcii do jedného modulu. Už v predošlom uvedenom návrhu existuje funkcia *Pridanie študenta*, ktorá sa týka len dát študenta. Podobné funkcie je vhodné spojiť s dátami, nad ktorými sú vykonávané a tým dôjde ku vzniku, realite podobným, objektom.



Obrázok 3.2 Objektovo orientovaný prístup

Objektový prístup umožňuje existenciu viacerých inštancii jedného modulu počas behu programu. Na škole by existovala jedna špecifická inštancia „Študent“ pre každého študenta, pričom by každá mala svoj vlastný obsah, príslušný ku konkrétnemu študentovi. Moduly medzi sebou komunikujú volaním svojich metód (funkcia) navzájom.

3.2.1 Zapuzdrenie

Iba konkrétna inštancia, ktorá vlastní svoju položku dát, ju môže meniť alebo čítať. Preto nemôže napr.: „Garant“ zmeniť „dátum nar.“ v inštancii „Študent“. Jedinou možnosťou ako nastaviť vnútorne zapuzdrené dáta konkrétnej inštancie, je zavolať jej verejnú metódu, ktorá ich môže zmeniť. O tejto možnosti však rozhoduje programátor priamo pri implementácií systému, ktorý týmto spôsobom môže dohliadať na korektnosť napĺňaných dát.

3.2.2 Objekt

Doposiaľ sme používali v tejto kapitole ako označenia skupiny dát a príslušných metód pojem „modul“. Ak sa pozrieme na charakteristiku modulu, tak môžeme nájsť v reálnom svete množstvo paralel.

Objekty sú v skutočnosti charakterizované svojím stavom a správaním. Napr.: televízor má svoj stav, ktorý ho charakterizuje: frekvencia, zapnutý, program,... Správania ktoré menia stav sú: zapnutie, vypnutie, prepnutie programu,... Stav objektu predstavuje dáta modulu, a správanie je totožné s metódami. Z dôvodu paralelnosti modulov s reálnym svetom hovoríme o objektoch a objektovo orientovanom prístupe.

3.2.3 Terminológia

Dátam objektu hovoríme *Atribúty* a správaniu *Metódy*. *Trieda* predstavuje šablónu objektu, v ktorej je uvedené aké metódy a atribúty existujú pre všetky inštancie danej triedy. Objekt predstavuje konkrétnu inštanciu danej triedy, ktorá už má vlastné hodnoty atribútov.

3.2.4 Objektovo orientovaná stratégia

Predošlé kapitoly pojednávali len o najpodstatnejších častiach objektovo orientovaného prístupu, a ich prínose do budovania systému. Ďalšími charakteristikami prístupu sú najmä dedičnosť a polymorfizmus. Objektový prístup bude používaný pri vytvorení a práci s *Diagramom tried*.

4 Projekt management

Project management predstavuje disciplínu, ktorá sa zaoberá definíciou a dosiahnutím cieľov, za optimalizácia vynaložených prostriedkov (čas, peniaze, ľudia, priestor, a pod.). Úlohou projektového managementu je zabezpečiť dokončenie projektu v požadovanej kvalite a termíne. [Wikip-2005]

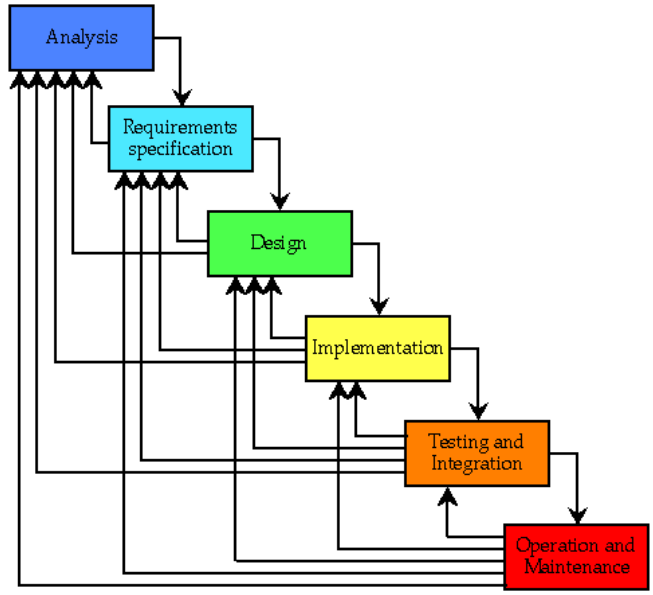
4.1 Kvalita aplikácie

Kvalitu aplikácie nie je možné empiricky zmerať alebo exaktne definovať postupy k jej dosiahnutiu. Je možné definovať len nároky kladené na aplikáciu. [Stein-2005]

- *Zhoda pôvodných požiadaviek zadávateľa s funkčnou aplikáciou*
  Snahou je zabezpečiť aby zákazník dostal takú aplikáciu, akú požadoval. Dodatočné implementovanie a úpravy sú náročné a môžu spôsobiť nekonzistentnosť celého systému.
- *Spracovateľnosť aplikácie*
  Aplikácia by mala byť jednoducho rozšíriteľná na základe nových požiadaviek zadávateľa.. Implementovanie nových častí a modulov neznamená kompletné prepísanie aplikácie ale malo by predstavovať len nenáročné pridanie prvkov systému, ktoré budú naviazané na už používaný systém.
- *Opakovaná použiteľnosť častí systému*
  Návrh niektorých častí aplikácie by mal byť natoľko abstraktný, že jednotlivé moduly budú použiteľné aj v ďalších aplikáciách. Väčšina softwarových firiem má už predvytvorené vlastné komponenty, ktoré implementuje do nových riešení, čím ušetrí značný čas tvorby systému.
- *Spoľahlivosť aplikácie*
  Spoľahlivosť sa definuje ako schopnosť reagovať na rovnaké vstupy rovnakým spôsobom.
- *Robustnosť aplikácie*
  Aplikácia sa musí zotaviť z výnimiek, bez úniku informácií, straty dát alebo výpadku aplikácie.
- *Bezpečnosť*
  V poslednom čase je na túto kvalitatívnu stránku aplikácia kladený dôraz. Zvýšenú bezpečnosť vyžadujú najmä aplikácie spravujúce tajné informácie alebo pracujúce s financiami. Na bezpečnosť aplikácie je nutné dbať už počas návrhu a celý design navrhovať podľa bezpečnostných zásad[2].

[2] S bezpečnosťou aplikácií súvisí pojem 3D, ktorý značí: secure by design, secure by default, secure by deployment.

4.2 Vodopádový model



Obrázok 4.1 Vodopádový model[3]

Princípom *vodopádového* modelu je nutnosť zachovania absolútnej postupnosti. Jedna fáza musí byť dokončená, aby mohla nasledovať nasledujúca. Systémy s väčšou komplexnosťou však na tomto modeli zlyhávajú. Rozsiahle projekty, môžu pri použití tohto modelu, práve kvôli nutnej postupnosti, trvať extrémne dlho.

Hlavný zdroj problémov predstavuje fakt, že systém musí byť plne pochopený a analyzovaný, aby bol možný prechod do ďalšej fázy. S rastúcou komplexnosťou je táto požiadavka pre vývojárov len veľmi ťažko splniteľná.

Nie menej podstatný, a často sa vyskytujúci problém pri danom návrhu je, že chyby sa objavia v neskoršej fáze projektu, mnoho ráz až pri integrácii. Ich riešenia si však vyžadujú vyššie náklady, ako problémy odhalené už pri fáze návrhu.

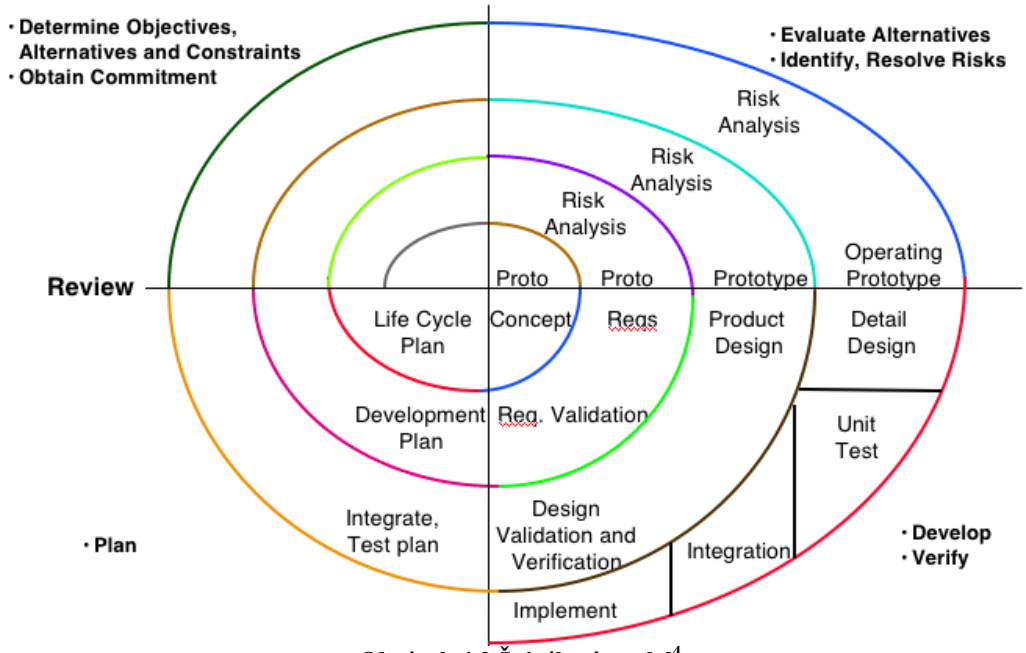
4.3 Špirálový model

Špirálový model poskytuje alternatívu k *vodopádovému* modelu. Pri tomto prístupe projekt podstupuje prechod kratšími časovými cyklami. V špirálovom modeli sa, oproti *vodopádovému*, môže celý team koncentrovať na všetky fázy. Výsledkom každého jedného cyklu je aplikácia, na ktorú má

[3] Zdroj: 25.4.2007 http://www.csse.monash.edu.au/~jonmc/CSE2305/Topics/07.13.SWEng1/html/text.html

možnosť reagovať zadávateľ. Týmto spôsobom sa predchádza vzniku aplikácie, nezhodnej s požiadavkami zákazníka.

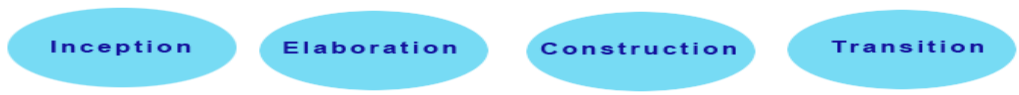
Nevýhodami tohto prístupu je niekedy príliš vysoká rýchlosť vývoja aplikácie, ktorá nie je často vo všetkých fázach uskutočniteľná. Ďalším problémom je management projektu, ktorý vyžaduje absolútne iný a zložitejší prístup ako *vodopádový* model.



Obrázok 4.2 Špirálový model[4]

4.4 Iterative, Incremental Frameworks

Tento model iteratívnej, postupnej konštrukcie je logickým rozšírením špirálového modelu, ale je viac formálny a presný. Model je rozdelený na štyri základné časti: Založenie (Inception), Rozpracovanie (Elaboration), Konštrukcia (Construction), Prechod (Transition). (viď. [Milto-2003] a [Ariad-2005])



Obrázok 4.3 Iterative, Incremental Frameworks

Vzhľadom na to, že tento prístup bol využívaný v projekte, bude v nasledujúcich kapitolách podrobnejšie prebraná jeho štruktúra a filozofia.

[4] Zdroj: 24.4.2007 http://www.kendall-consulting.com/projmgt.html

4.4.1 Inception

Vo fáze založenia projektu, dochádza predovšetkým k určeniu jeho rozsahu a celkovej vízie hotového riešenia. Pri malých projektoch postačuje získanie predstavy o systéme od zadávateľa a dohoda o začatí práce. V náročnejších riešeniach by výsledkom tejto fázy mali byť:

- dokument vízie projektu
- prvotný prieskum požiadaviek
- business plán, ktorý zahrňuje kritéria financovania, náklady a návrat investícií
- odhadnutie rizík projektu
- plán projektu

4.4.2 Elaboration

Vo fáze rozpracovanie sa stretávame s prieskumom problému vo väčších detailoch. Je nutné podrobne pochopiť požiadavky zákazníka a taktiež požiadavkou na úspešne zvládnutie projektu je porozumenie vzťahom a fungovaniu jeho procesov práce. Potrebujeme mať široký prehľad o kompletnom systéme, ako celku. Konkrétnymi podrobnosťami a detailmi sa však ešte nezaoberáme, tie budú riešené po častiach. V tejto fáze je nutné neuvažovať nad implementáciou.

Analýzu riskantných častí predstavuje prototypovanie. Výsledkom *Elabortation* fáze sú: *Diagram prípadov použitia*, a *Konceptuálny model*.

4.4.2.1 Prototypovanie

Predstavuje podstatnú časť *Elaboration* fázy. Jedná sa o vytýčenie riskantných častí systému. Je výhodné odhaliť riskantné miesta projektu už v tejto fáze, aby sa im predišlo pri designu, kde by si ich riešenie mohlo vyžiadať značné náklady a spôsobiť prípadné nutné prepracovanie návrhu.

4.4.2.2 Diagramy prípadov použitia

Úlohou v *Elaboration* fáze je identifikovať, podľa možnosti všetky *prípady použitia* systému. Podmienkou je, vyhnúť sa granularite[5], ktorá by mohla spôsobiť neúmerné zvýšenie komplexnosti a neprehľadnosti diagramu. Po vytvorení diagramu je potrebné overiť, či zahŕňa všetky požiadavky na systém.

Postup vytvorenia diagramu prípadov použitia:

1. Hľadanie všetkých možných *Aktérov* systému.
2. Nájdenie všetkých prípadov použitia (*Use Case*)

[5] V týchto prípadoch použitia je nutné zaznamenávať iba výsledok pre *Aktéra* prípadu použitia a nie kompletné fázy, ako k výsledku prišiel. Ak sa jedná napr.: o výber peňazí z bankomatu, tak nebude prípadom použitia: *vloženie karty, zadanie pinu*, *zadanie sumy*,..., ale len *výber peňazí*. Granularite sa venuje *Construction* fáza.

3. Krátky popis každého prípadu použitia
4. Grafické zachytenie do modelu

Tieto štyri fázy tvorenie diagramu sú vykonávané za častej komunikácie so zákazníkom. V poslednej dobe za začína používať na túto činnosť tzv. *workshop*, kde všetci kompetentný vyjadrujú svoje predstavy o systéme a zároveň sú tieto myšlienky spracovávané do diagramu.

Je ideálne nájsť v tejto fáze všetkých *Aktérov* a *Prípady použitia*. Ak sa však neodhalia všetky, najčastejšie vystúpia v neskorších fázach návrhu a integrujú do systému.

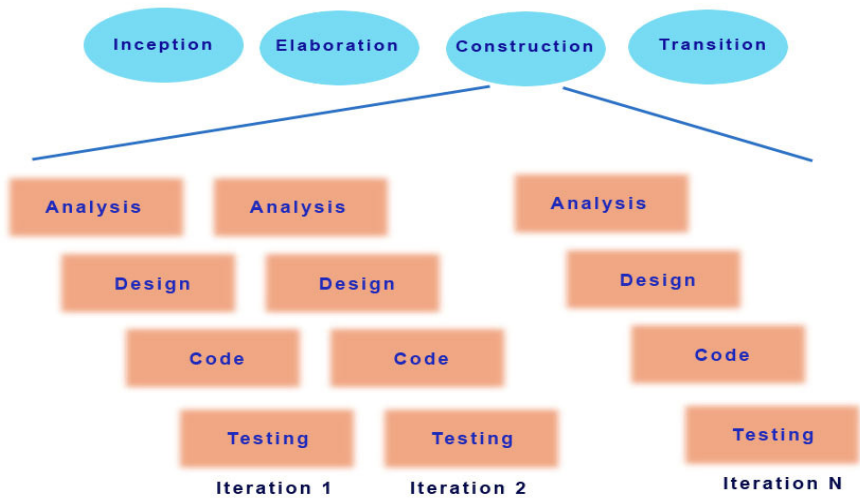
4.4.2.3 Konceptuálny model

Konceptuálne modelovanie je činnosť, ktorá vedie k nájdeniu podstatných konceptov projektu. Tento spôsob nám umožňuje širšie chápať systém a uvedomovať si deje vykonávané zákazníkom. UML nehovorí kedy a ako konceptuálne modelovať, ale poskytuje len syntax *Diagram Tried*, ktorým je možné danú problematiku formalizovať.

Z *Konceptuálneho modelu* vychádza štruktúra databáze a referenčná integrita.

4.4.3 Construction

V tejto fáze dochádzka k samotnému vytváraniu projektu. Projekt je realizovaný podobným spôsobom, ako pri *špirálovom* modeli, kde sa jedná o skupinu samostatných iterácií. Každá z týchto iterácií predstavuje samostatný jednoduchý *vodopádový* model. Ak vhodne rozdelíme problematiku na krátke iterácie, týkajúce sa jednotlivých častí projektu, tak predídeme problémom spojeným s komplexným návrhom projektu *vodopádovým* modelom. Na konci každej iterácie by mala vzniknúť jednoduchá verzia funkčnej časti programu.



Obrázok 4.4 Rozdelenie Construction na Iterácie

4.4.3.1 Analýza

Pri analýze je nutný spätný pohľad na diagramy prípadov použitia, zameranie sa na ich podrobnosti a kompletnú funkčnosť. V každej iterácií je bezpodmienečné sa zamerať iba na niekoľko prípadov použitia, aby sme sa vyhli komplexnosti a problému vzniku veľkého *vodopádového* modelu. Stále je nutné sa zaoberať len problémom a nie konkrétnym riešením a implementáciou.

Výsledkom tejto fázy sú podrobne popísané prípady použitia, ktoré by mali spĺňať obsahovo nasledujúcu štruktúru.

Názov:	Názov prípadu použitia.
Popis:	Krátky popis prípadu použitia, o čom pojednáva.
Aktéri:	*Aktéri* prípadu použitia, ktorých sa *Use Case* jedná.
Vstupné podmienky:	Podmienky, ktoré musia byť splnené, aby mohlo dôjsť k vykonaniu prípadu použitia. Napr.: *Prihlásený užívateľ*,
Výstupné podmienky:	Hovoria o výslednom stave systému, ktorý nastane pri úspešnom vykonaní prípadu použitia.
Hlavná cesta:	Predstavuje najpravdepodobnejší a najčastejší postup pri vykonávaní prípadu použitia.
Alternatívna cesta:	Predstavuje menej častý ale legitímny postup. Najčastejšie pri použití alternatívnej cesty dochádza k zmene výstupných podmienok.
Záznamy:	Tu je uvedené či daná akcia vyžaduje záznam o vykonaní, prípadne nevykonaní.
Poznámky:	Dodatočné záznamy potrebné pre ďalší návrh.

Tabuľka 4.1 Podrobný popis prípadu použitia

4.4.3.2 Design

Po vykonaní všetkých predošlých fáz, už existuje kompletná štruktúra programu, ktorú je nutné v danej iterácii riešiť. Úlohou je navrhnúť riešenie pre podrobne popísané prípady použitia, a rozhodnúť o interakciách objektov, nutných pre splnenie daného problému.

UML poskytuje na vyjadrenie interakcie objektov *Diagram komunikácie* a *Sekvenčný diagram,* ktoré sú si veľmi blízke. Keď sa jedná o objekty, tak je nutné dokumentovať práve ich triedy. Zachytenie týchto informácií poskytuje *Diagram tried*, vychádzajúci priamo z *Konceptuálneho modelu*, ktorý bol vytvorený v predošlej fáze. V designe sa niekedy používa taktiež *Stavový diagram.*

4.4.3.3 Implementácia

Jedná sa už o implementáciu v konkrétnom programovacom jazyku. Je nutné vytvoriť *Diagram komponent* a *Diagram balíkov*. Týmto vznikne napríklad vrstvová architektúra systému. Pri prvej iterácií musíme vytvoriť triedy[6], ktoré budú používané ostatnými prvkami, v nasledujúcich iteráciách.

Programovaním sa prakticky *Diagram tried* premietne do reálnej podoby a taktiež sa vyjadria jeho interakcie. Musíme sa zaoberať konkrétnym problémom komplexne, od databázy až po užívateľské rozhranie, pretože výsledkom tejto fázy by mala byť jednoduchá testovateľná časť systému.

Tu je nutné navrhnúť aj štruktúru databázy, v ktorej ide o premietnutie *konceptuálneho modelu* do tabuliek a ich vzťahov. K databáze je nutný komplexný prístup a mali by sme mať predstavu o systémy, aby sme si implementáciou konkrétnej časti nezamedzili integrovanie ostávajúcich. Štruktúra databáze vychádza z K*onceptuálneho modelu*, z neho sa určia komplexné vzťahy, a integrita. Z *Diagramu tried* sa tabuľkám v databáze priradia stĺpce, ktoré musia obsahovať.

4.4.3.4 Testovanie

V každej iterácií je nutné otestovať funkčnosť podľa konkrétnych prípadov použitia. Pri zistení nedostatkov je nutný návrat do niektorej z predchádzajúcich častí.

Po tejto fáze, už môže byť konkrétna časť predvedená zákazníkovi, ktorý by mal posúdiť, či projekt smeruje správnym smerom.

4.4.4 Transition

Finálna fáza ktorá predstavuje postúpenie výsledného produktu zákazníkovi. Na rozdiel od vodopádového modelu, by mal byť produkt v tejto fáze už plne funkčný a otestovaný.

Typickými prvkami *Transition* časti sú:

- vydanie beta verzii
- paralelne testovanie so starým systémom
- importovanie a konvertovanie starej databázy do novej
- zaučenie užívateľov
- marketing, distribúcia a predaj

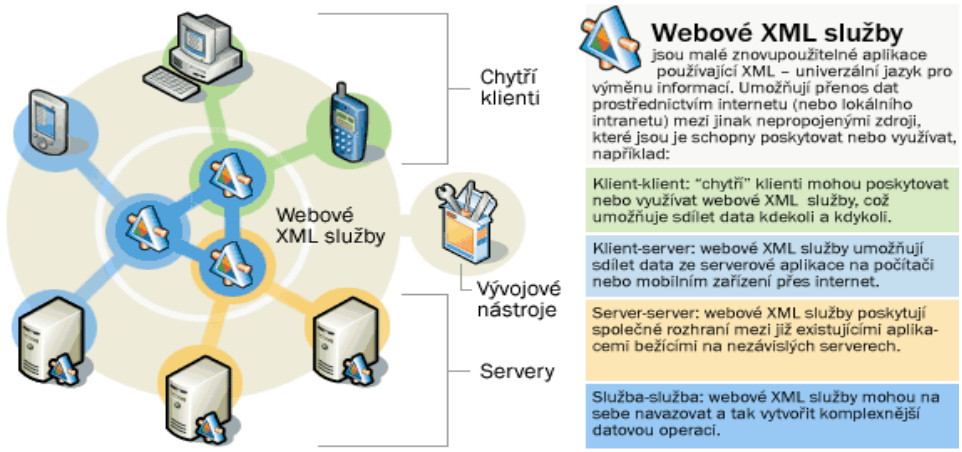
[6] Najčastejšie ich predstavujú balíky *Common* a *SystemFramework*

5 Využívané technológie

5.1 Microsoft .NET

Microsoft® ,NET predstavuje sadu softwarových technológií pre prepojovanie sveta ľudí, informácií a systémov. Jeho snahou je odpútať sa od klasických postupoch vývoja softwaru a zapojiť Internet. Umožňuje vysoký stupeň integrácie programov, použitím technológie webových XML služieb.

Webová služba predstavuje aplikáciu bežiacu na Internete, ktorá poskytuje webové metódy klientom na Internete. K výmene dát sa používa formát XML. Postupným rozširovaním webových služieb sa Internet stane softwarovou platformou s podstatne širším rozhraním API, aké dokáže poskytnúť operačný systém. (viď. [Micro-2005])

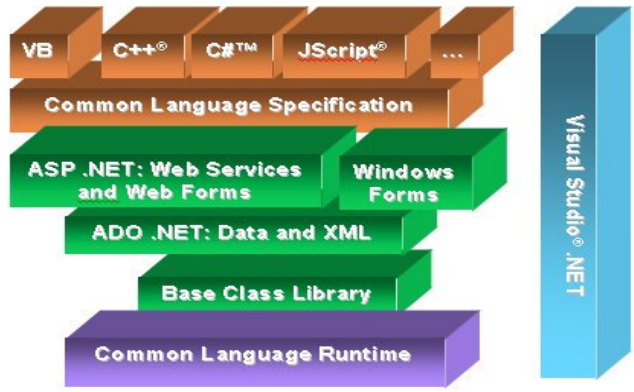


Obrázok 5.1 Microsoft® .NET[7]

[7] Zdroj: 25.4.2007 http://www.microsoft.com/cze/net/basics/whatis.asp

5.2 .NET Framework

.NET Framework je integrovaná komponenta Windows, ktorá umožňuje vytváranie a beh softwarových aplikácií a webových služieb. (viď. [Meliš-2005] a [Prosi-2003])



Obrázok 5.2 NET Framework[8]

- **Visual Studio .NET**

Predstavuje spojovaciu komponentu medzi ostatnými prvkami. Je to programovacie rozhranie pre niekoľko jazykov: C#.NET, Visual Basic .NET, Visual C++ .NET, ASP .NET, J# .NET. Ich spoločnou črtou je striktné dodržiavanie princípov objektového programovania. Neexistuje obmedzenie a je možné pridať ďalšie jazyky do rozhrania. V súčasnosti je dostupná verzia Microsoft Visual Studio .NET 2005.

- **Common Language Specification**

Spoločný modul jazykovej špecifikácie, predstavuje komponentu, ktorá zaisťuje možnosť kompilácie rôznych jazykov do jedného univerzálneho kódu, zdieľaného vo vnútri .NET. Sada obsahuje pravidlá pre tvorbu zdieľaných knižníc a pravidla pre vytváranie kódu použiteľného v každom z podporovaných vývojových jazykov.

- **ASP .NET**

Predstavuje rozšírenú verziu skriptovacieho jazyka ASP (Active Server Pages), ktorý sa používa pre vytváranie webových aplikáciu nad webovým servrom. Poskytujú možnosť vývoja dynamických webových stránok.

- **Windows Forms**

Je sada, predom vytvorených funkcií a objektov, ktoré zjednodušujú vytváranie aplikácií, pracujúcich priamo pod systémom Windows.

- **ADO .NET**

Komponenta, ktorá umožňuje sprístupnenie dát. Nová verzia .NET zabezpečuje prístup nie len k databáze ale ku všetkým zdrojom dát, čo je dané tým, že pracuje s dátami vo formáte XML.

- **Base Class Library**

[8] Zdroj: 25.4.2007, http://notebooky.idnes.cz/obrazek/schema.jpg

Knižnica tried rámca (Base Class Library[9]) poskytuje objektovo orientované rozhranie API, do ktorého riadené aplikácie zapisujú. Predstavuje náhradu API Windows, COM, ATL, MFC a ďalších.

- **Common Language Runtime**

Spoločný jazykový modul behu, ja jadrom všetkých aplikácií napísaných v tomto prostredí. Zaisťuje vlastný beh programu – alokácia pamäti, prístupové práva, spracovanie výnimiek.

5.3 Webové služby

V posledných rokoch podmienka spojenia ľudí, softwaru, procesov zmenila spôsob akým je software vyvíjaný. Úspešné systémy vyžadujú spoluprácu naprieč platformami a flexibilné služby, ktoré sa môžu jednoducho vyvíjať a obmieňať. To viedlo k prijatiu XML ako univerzálneho jazyka pre reprezentáciu a prenos štruktúrovaných dát nezávislých od programovacieho jazyka, softwarovej platformy a hardwaru.

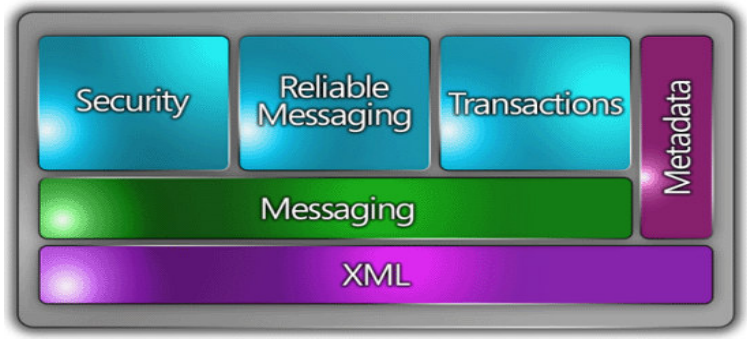
Vystavené na rozsiahlej akceptácií XML, predstavujú webové služby aplikácie, ktoré používajú štandardný transport, kódovanie a protokoly na výmenu informácií. So širokou podporou poskytovateľov a businessu, umožňujú webové služby komunikovať systémom na rôznych platformách cez internet, aj extranet, za podpory end-to-end bezpečnosti, spoľahlivého prenosu správ a distribuovaných transakcií.

Webové služby sú založené na jadre štandardov, ktoré popisujú syntax a sémantiku softwarovej komunikácie. XML poskytuje známy syntax pre reprezentáciu dát. Simple Object Access Protocol (SOAP) poskytuje sémantiku pre výmenu dát. Web Services Description Language (WSDL) poskytuje mechanizmus na popis spôsobilosti webových služieb. Ďalšie špecifikácie, nazývané ako WS-* architektúra, definujú funkčnosť pre nadviazanie spojenia, eventing, prílohy, bezpečnosť, spoľahlivú komunikáciu, transakcie a management.

Od zavedenia webových služieb bola vynakladaná snaha Microsoftu práve na podporu tejto technológie a jej integrovanie do platformy .NET. V budúcnosti sa predpokladá, že práve webové služby budú ešte viac integrované do systémov a bude stále viac ich poskytovateľov. Preto práve z tohto dôvodu je nutné informačný systém vyvíjať týmto smerov spolupráce s externými systémami.

Daná interoperabilita zabezpečuje rozšíriteľné spojenie aj s ďalšími systémami v budúcnosti bez značných zásahov do samotnej architektúry systému.

[9] Niekedy je používaný pojem Framework Class Library - FCL



Obrázok 5.3 Architektúra WS-*[10]

5.4 MS SQL

MS SQL predstavuje technológiu databázových systémov od firmy Microsoft. Jedná sa o relačnú databázu za splnenie jej základných paradigiem. MS SQL poskytuje referenčnú integritu a možnosť zachovania jej kontroly. Nad databázou sa programuje v jazyku T-SQL (*Transact SQL*), ktorý poskytuje možnosti písanie serverových databázových procedúr (*Stored Procedure*).

Technológia .NET je použitím svojej komponenty ADO .NET uspôsobená na MS SQL a poskytuje jednoduchý prístup k dátam.

[10] Zdroj: 25.4.2007 http://www.microsoft.com

6 Analýza a riešenie procesov vo firme

Požiadavka vytvorenia informačného systému vystála predovšetkým z náročných procesov vo firme, ktorých prevádzka bola značne časovo nákladná a neprehľadná. Jednalo sa predovšetkým o implementáciu niekoľkých základných a pre funkčnosť firmy podstatných procesov, ktorými sú napr. prihlasovanie lektorov na jednotlivé kurzy, a prihlasovanie účastníkov na školenia. Samozrejmosťou je správa všetkých používaných prostriedkov, ako sú užívatelia, lektori, zákazníci, kurzy, termíny, učebne a pod. V rámci správy týchto prostriedkov je možné rozumieť aj zabezpečenie aplikácie a oprávnení užívateľov, ale taktiež editáciu kurzov a vytváranie termínov a priradzovanie voľných učební a pod.

V tejto kapitole sú dôkladnejšie rozobrané len dva zo základných procesov, ktorých opodstatnenosť a funkčnosť je zrejmá. Je nutné ale podotknúť, že bolo implementovaných väčšie množstvo procesov, a predovšetkým taktiež spravovacích nástrojov. Často využívané a taktiež prakticky používané sú predovšetkým rôzne spôsoby filtrovania a radenia objektov. Bezpečnostne rizikovým, ale nutným je zabezpečenie limitovaného prístupu pre rôznych užívateľov systému, kde je komerčne podstatné, aby sa určitým užívateľom zobrazovali len určité informácie. Práve forma tohto riešenia je značne podporovaná neskôr popisovaným frameworkom.

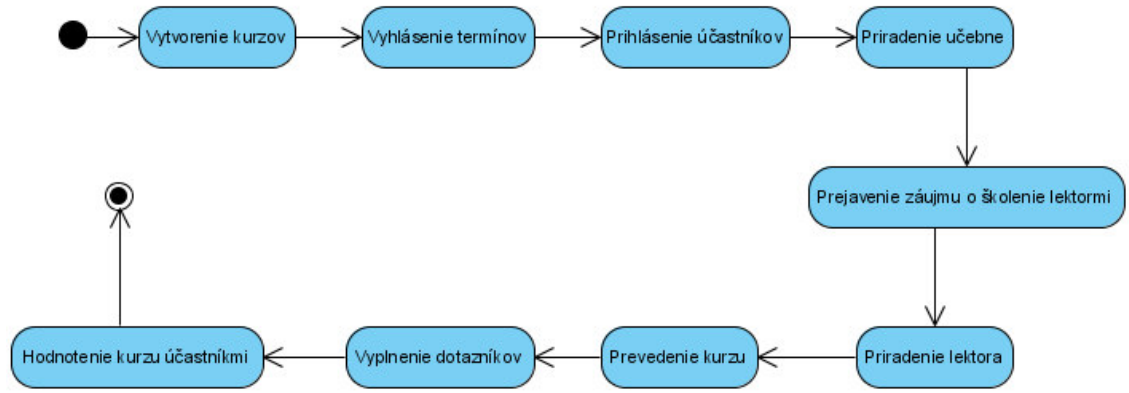
6.1 Proces kurzov

Školiace stredisko ponúka isté kurzy, ktoré majú presnú osnovy, doporučenú literatúru a pod. Tieto skutočnosti sú zverejnené na letákoch a bulletinoch firmy. K daným kurzom sú potom vyhlásené termíny, ktoré môžu byt verejne zobrazené na stránkach a potom sa účastníci prihlasujú na daný termín.

Proces prihlasovania bol však vykonávaný manuálne a zobrazený na statických web stránkach. Je samozrejme, že nie každý termín je plne obsadený a niekedy je preto zrušený. Ak sa však vedúci školení rozhodne, že termín je dostatočne obsadený, potom pridelí záväzne učebňu na daný termín a čas. Tento proces je náročný, pretože je nutné sledovať, či zvolená učebňa je v danom čase voľná a nie je obsadená iným školením. Ak je pridelená učebňa, potom je pridelený lektor. Školiace centrum, však nemá stálych lektorov ale niekoľko zmluvných, ktorý nepracujú výhradne v školiacom centre, alebo prípadne ešte študujú. Je samozrejme, že oslovený lektor nemusí mať čas a môže ponuku odmietnuť. Klasické riešenie bolo poslanie emailu lektorom s ponukou školení na celý mesiac. Lektor ku každému termínu uviedol, či školiť môže alebo nie. Časovo náročné však bolo spracovane týchto informácií. Vedúci strediska totiž musel prečítať a vyhodnotiť niekoľko emailov a prideliť školenia, v prípade znásobeného záujmu lektorov rozhodnúť, ktorý dané školenia dostane a oznámiť skutočnosť naspäť lektorovi. Tento proces bol značne časovo náročný a stával sa postupne

aj finančne nákladný, pretože vedúci strediska trávil podstatnú časť svojho času práve týmto procesom. Celá agenda bola vedená v programe MS Excel za podpory plánovania MS Outlook.

V implementovanom informačnom systéme bolo nutné práve tento proces vyriešiť a implementovať správny workflow, aby podľa možností vyhovoval uvedeným požiadavkám. Výsledná implementácia značne zjednodušuje prihlasovanie lektorov na kurzy. Každý lektor má prístup do vlastného profilu informačného systému, kde vidí vypísané, ešte neobsadené kurzy, o ktorých vyučovanie môže prejaviť záujem. Vedúci školení následne prehľadne vidí všetkých lektorov, ktorý prejavili záujem a môže termín niekomu prideliť. Lektorovi je poslaný email oznamujúci záväzné pridelenie školenia a taktiež je zobrazený v rozvrhu termínov lektora v jeho profile, kde si ho môže jednoducho importovať do kalendára programu MS Outlook.



Obrázok 6.1 Proces kurzov a termínov

6.2 Proces prihlasovania účastníkov na termíny

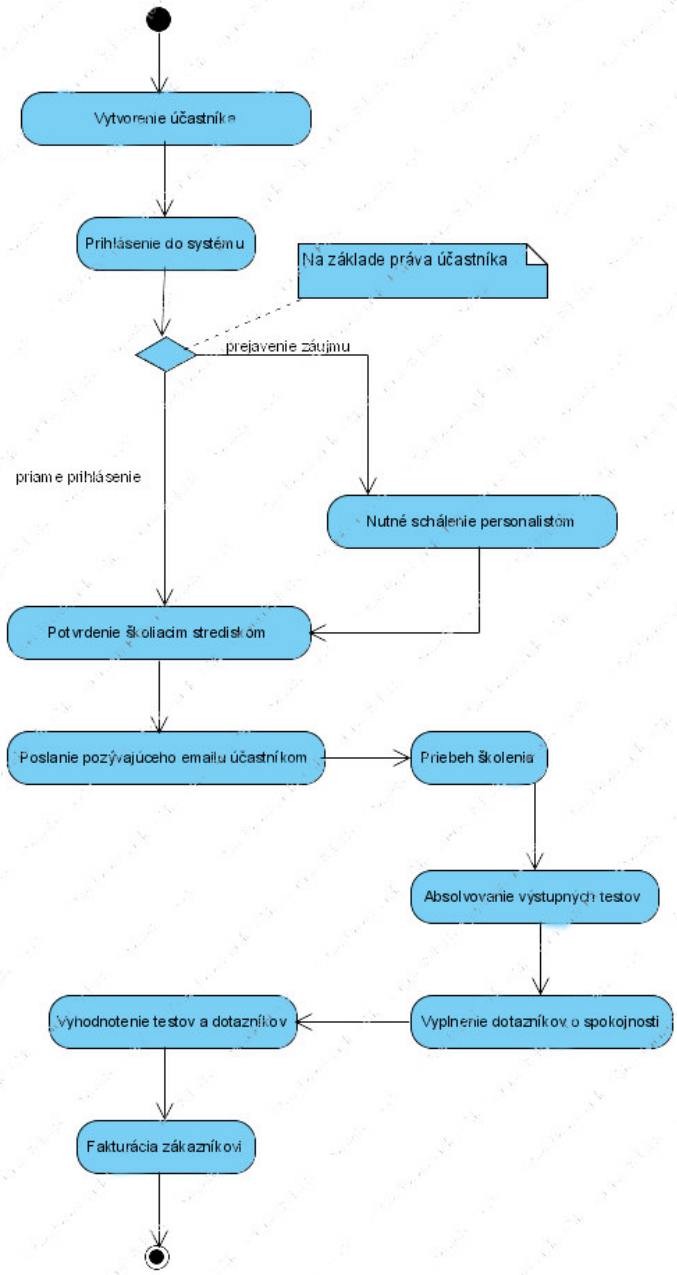
Aj tento proces patril pred zavedeným systému k značné náročným, ale taktiež ku kritickým, práve preto, že sa jednalo o zákazníka a komunikáciu s ním. Je jednoduchá predstava zákazníka ako individuálnej osoby, ktorá sa prihlási na kurz telefonátom do firmy. Podstatnejší sú však zákazníci, ktorý predstavujú veľké firmy a tým aj značné kontrakty školení. Takýto zákazník zvyčajne vyhlási výberové konanie, a školiace stredisko musí ukázať prečo je lepšie. Následne sa jedná s personalistom, ktorý často vyžaduje prehľadové tabuľky o záverečnom testovaní zamestnancov, o ich úspešnosti a očakáva, že skoliace centrum mu v každom prípade vyjde v ústrety. Taktiež je častou požiadavkou umožnenie účastníkom prejavovať záujem o konkrétne termíny, čím zákazníkovi odpadá agenda plánovania a prenáša sa na školiace centrum. Vyhovenie týchto požiadaviek však často znamená pre vedúceho strediska hodiny strávené kopírovaním a vyhodnocovaním dát a vytváranie tabuliek personalistovi.

Samotný proces je nutné rozdeliť podľa práv účastníka. Ten totiž môže byť individuálna osoba, ktorá sa hlási na kurz z vlastného záujmu a nie ako zamestnanec spoločnosti. Takýto účastník by mal mať právo voľnej registrácie termínov. Druhým špecifickým prípadom je účastník, ktorý je zamestnancom firmy, ktorá ho posiela na školenie. Je logické že tento účastník nemá voľnú voľbu kurzov, ale o jeho školení často rozhoduje a potvrdzuje personalista. Ten by mal mať možnosť takéhoto účastníka prihlásiť na zvolené a dojednané školenie, za často zjednaných podmienok, prípadne zliav. Je nutné uvažovať o tom, že účastník sa síce nemôže záväzne prihlásiť na kurz, ale môže vyjadriť záujem o kurz v určitom termíne a ten musí byť schválený personalistom.

Po prihlásení účastníka na školenie, či už to urobil priamo účastník, alebo personalista, prichádza do procesu správca školení, ktorý vidí prehľad počtu účastníkov na daný termín. Práve podľa počtu rozhodne, či sa bude dané školenie konať alebo nie. Ak rozhodne o konaní, tak prihlásenie účastníkov potvrdí. V tom momente účastníci sa už nemôžu z daného školenia odhlásiť a sú záväzne prihlásený. Zároveň im je poslaný automatický email, ktorý ich upozorňuje na školenie, na miesto a čas jeho konania. Vedúci školení v tejto fáze taktiež pridelí učebňu a umožní prihlasovanie lektorov na termín. Tento značne komplikovaný proces je taktiež nutné vhodne implementovať v navrhovanom informačnom systéme.

Informačný systém sprehľadňuje prihlasovanie účastníkov a najmä zrýchľuje agendu a umožňuje prácu s väčším množstvom zákazníkov. Proces je často dolaďovaný kvôli požiadavkám zákazníka. Jedná sa často o detaily ako nezobrazenie ceny školenia účastníkov, prípadne vytváranie termínov a cien pre individuálnych zákazníkov, ktoré nebudú zobrazované v celkovej verejnej ponuke. Taktiež nutné bolo riešenie generovania loginov a hesiel účastníkom a ich automatické poslanie emailom.

Značnú organizačnú výhodu predstavuje práve umožnenie účastníkom prejavovať záujem o termín a následné potvrdenie záujmu personalistom. Po tomto potvrdení je účastník zobrazený v prehľade pre vedúceho školení, ktorý vidí obsadenosť kurzu. Tento proces je poskytovaný zákazníkom ako bonus a odľahčuje agendu personalistom. Práve táto skutočnosť nadštandardnej služby prináša školiacemu centru komerčnú výhodu pred konkurenciou.



Obrázok 6.2 Proces prihlasovania účastníkov na termíny

7 Realizácia projektu

V kapitole sa nachádza popis vlastnej realizácie projektu. Postupne sú preberané jednotlivé časti vývoja systému použitím *Iterative, Incremental Framework.* Jedná sa o návrh informačného strediska školiaceho strediska. Celú realizáciu je možné rozdeliť do dvoch základných častí kde sa jednalo o vytvorenie internej časti, určenej pre administrátora, lektorov, riaditeľa a pod. a časť externú, určenej pre zákazníkov, účastníkov, personalistov a pod.

Návrh projektu, ako aj jeho realizácia postupovala za vysokej interakcie zadávateľa, ale taktiež dotazov zákazníkov školiaceho centra. V poslednej, fáze vznikali verzie takmer každým dňom a zadávateľ zabezpečoval, aby vývoj napredoval správnym smerom.

7.1 Inception

7.1.1 Špecifikácia požiadaviek

Príloha: **Špecifikácia požiadaviek**

V tejto fáze boli spísané základné požiadavky zadávateľa k celému systému. Predstavujú len náčrt celej šírky systému a nezaoberajú sa detailmi, lebo tie sa budú riešiť jednotlivo. V uvedených požiadavkách je prehľad systému, ktorý však presahuje rozsah tu uvedenej implementácia. Požiadavky sú písané z veľmi širokého hľadiska, aby bola zabezpečená rozšíriteľnosť systému do budúcnosti. Môžeme vlastne hovoriť o víziách systému.

Práve kvôli požiadavkám ďalšieho vývoja projektu si prístup vyžadovať už od začiatku veľmi prehľadný prístup, ktorý je pochopiteľný nie len pre autora samotného projektu ale aj pre prípadných programátorov, ktorý by mohli daný systém rozšíriť.

7.1.2 Plán projektu

Dochádza k rozdeleniu projektu na dve časti z pohľadu vývoja. Jedná sa o *Zákaznícku* a *Administrátorskú* časť.

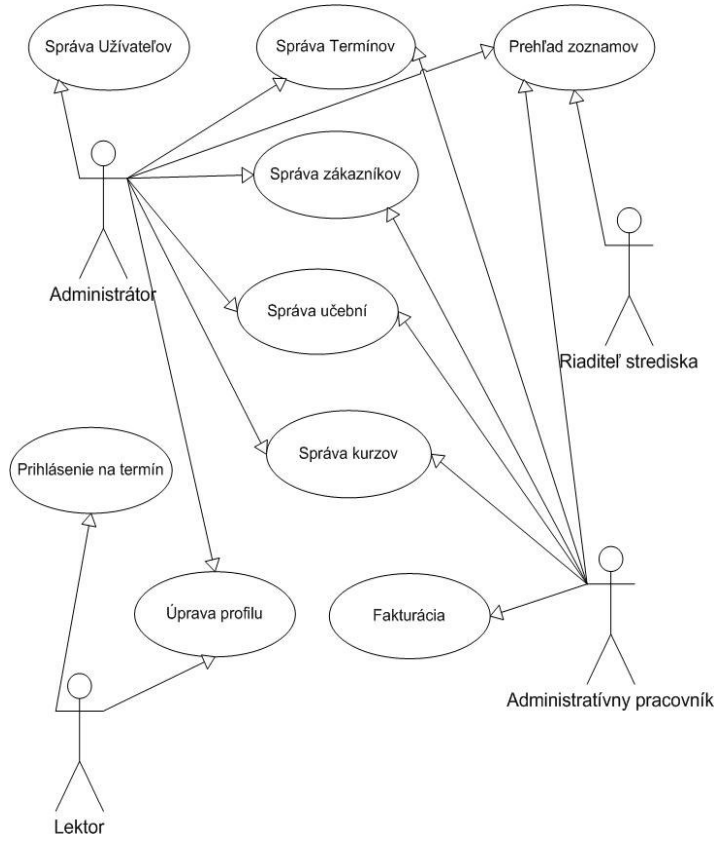
Ako pri každom projekte bol stanovený predbežný plán a postupnosť krokov, ktoré mali viesť k úspešnému systému a jeho reálnej časovej realizácií. V prvej fáze sa jednalo o štúdium používaných technológií, následne to bola analýza už konkrétnych požiadaviek, vybudovanie databázy, zostavenie architektúry a postupná implementácia požiadaviek a testovanie vytvorených častí. Vzhľadom na to, že niektoré časti projektu sú už nasadené v prevádzke, tak samozrejmosťou je udržovanie systému a odstraňovanie chýb.

7.2 Elaboration

Pre správne určenie požiadaviek zákazníka bola nutná častá komunikácia a zistenie fungovanie školiaceho strediska a procesov v ňom prebiehajúcich. Tieto základné implementované procesy boli popísané v predošlej kapitole *Analýza a riešenie procesov vo firme.*

7.2.1 Diagrami prípadov použitia

Na základe špecifikácie a podrobnejšej komunikácie som vytvoril *Diagram prípadov použitia,* ktorý zahŕňa najzákladnejšie požiadavky oboch častí systému.



Obrázok 7.1 Diagram prípadov použitia (interná časť)

Aktéri:

- *Administrátor:*

  Predstavuje užívateľa, ktorý ma absolútne práva. Jeho hlavnou funkciou je správa systému.

- *Lektor:*

  Užívateľ systému, ktorého úlohou je školiť niektoré kurzy. S tým súvisia aj jeho právomoci, ktoré sú obmedzené len na úpravu vlastného profilu a vyjadrovanie záujmu školiť nejaký termín.

- *Riaditeľ školiaceho strediska:*

Mali by mu byť zabezpečené prehľady zákazníkov, lektorov, fakturácií, prípadne v neskoršej fáze ekonomického riešenia o finančný obrat vo forme grafov.

- *Administratívny pracovník:*

  Jedná sa o užívateľa obmedzenými administrátorskými právami. Jeho úlohou je spravovať termíny a kurzy, prevažne sa zaoberať v zákazníckej časti fakturáciou a finančnými operáciami aj vo vzťahu k lektorom a zákazníkom

Prípady použitia:

- *Správa učební:*

  Predstavuje možnosť prehľadu učební, ich odstraňovania a pridávania do databázy.

- *Správa termínov:*

  Reprezentuje značne kritickú časť celej aplikácie. Termín je chápaný ako jeden neprerušený časový úsek počas ktorého trvá samotné školenie. Nastáva však problém so školeniami, ktoré trvajú viac dní. Z tohto dôvodu je nutné uvažovať o istých skupinách termínov, ktoré spolu súvisia. Napr.: Týždňové školenie sa skladá z piatich termínov od 9:00 do 16:00, ktorých spoločnou vlastnosťou je príslušnosť do jednej skupiny termínov. To môžeme nazvať granularitou termínov. Ku každému termínu musí byť možné priradiť učebňu a lektora, čo je možné previesť na celej skupine termínov, alebo aj na konkrétnom termíne. Uvedeným spôsobom je možné vyjadriť týždňové školenie, školené jedným lektorom, ale niektoré dni môže vďaka granularite zastupovať iný lektor. Z pohľadu lektora je nutné zabezpečiť vyjadrenie záujmu školiť a administrátor následne môže záujem potvrdiť. V danej problematike je integrovaná značná logika aplikácie.

- *Správa zákazníkov:*

  Predstavuje možnosť prehľadu zákazníkov, ich vytvárania a odstraňovanie, zaradzovanie do oblastí podnikania a vyjadrovanie platobnej morálky.

- *Správa užívateľov:*

  Jedná sa o prehľad a editáciu užívateľov, ktorými sú všetci aktéri vstupujúci do systému.

- *Správa kurzov:*

  Vytváranie, úpravy a odstraňovanie kurzov, aktuálne ponúkaných školiacim strediskom. Ku správe kurzov prináleží aj vytváranie nových termínov resp. skupín termínov, na ktoré sa potom môžu prihlasovať účastníci.

- *Úprava profilu:*

  Ak je lektor zamestnaný školiacim strediskom, tak je mu vytvorený užívateľský účet, pod ktorým sa môže prihlásiť a samozrejme meniť svoje heslo, adresu a pod.

- *Prihlásenie na termín:*

  Lektor v prostredí svojho profilu vyjadruje záujem učiť niektorý kurz a následne má prehľad termínov k danému kurzu. Má možnosť vyjadriť záujem daný termín školiť.

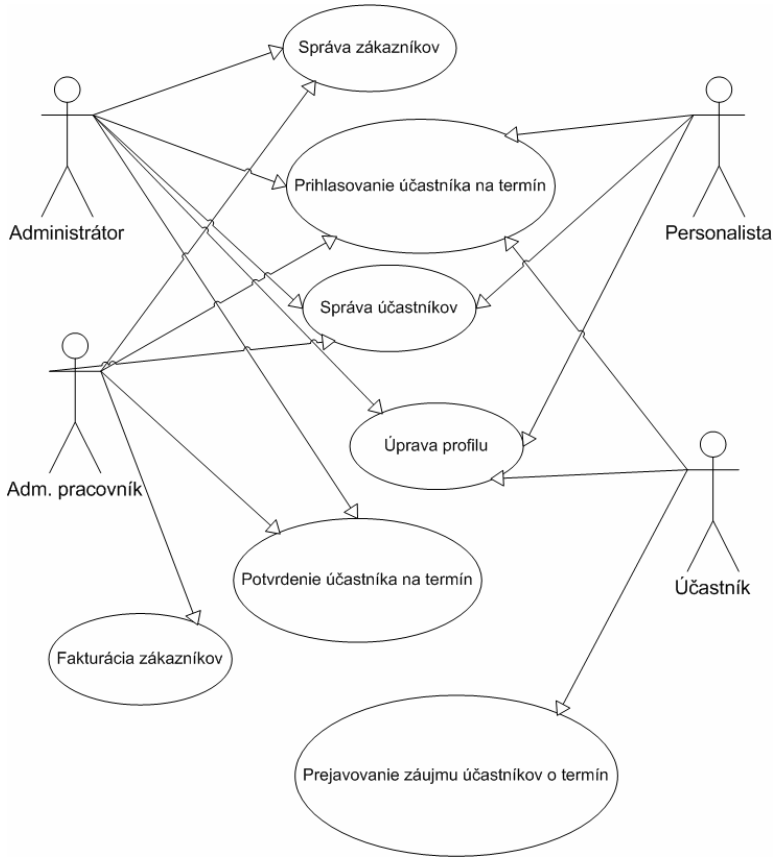
Záväzne školenie termínu je však až po odobrení administrátorom. Lektor sa z termínu, ktorý potvrdil administrátor nemá možnosť odhlásiť.

- *Prehľad zoznamov:*

  Jedná sa len o základné informácie o profile školení, termínov, lektorov, financovania,...

- *Fakturácia:*

  Ide predovšetkým o zobrazenie prehľadu termínov, ktoré daný lektor v istom časovom období školil, a prípadne o vytlačenie danej zostavy termínov v papierovej podobe.



Obrázok 7.2 Diagram prípadov použitia (externá časť)

Aktéri:

- *Administrátor:*

  Predstavuje užívateľa, ktorý ma absolútne práva aj v tejto externej časti. Jeho úlohou je predovšetkým správa a pridávanie zákazníkov a účastníkov.

- *Personalista:*

  Prestavuje externého užívateľa systému, ktorý ma značné administrátorské práva nad svojimi zamestnancami, ktorých predstavujú účastníci. Predovšetkým schvaľuje a potvrdzuje ich registráciu na termíny kurzov.

- *Účastník:*

Existujú dva základné typy tohto aktéra. Jedným je účastník – jednotlivec, ktorý sa registruje priamo na termín a ďalším je účastník – zamestnanec, ktorý vyjadruje iba záujem o termín a jeho registráciu musí potvrdiť personalista.

- *Administratívny pracovník:*

  Jedná sa o užívateľa obmedzenými administrátorskými právami. Jeho úlohou je spravovať zákazníkov a účastníkov ale taktiež potvrdzovať registráciu účastníkov a vo finálnej fáze vytvárať faktúru zákazníkom.

Prípady použitia:

- *Správa zákazníkov:*

  Predstavuje možnosť prehľadu zákazníkov, ich vytvárania a odstraňovanie, zaradzovanie do oblastí podnikania a vyjadrovanie platobnej morálky.

- *Správa účastníkov:*

  Jedná sa o prehľad a editáciu účastníkov, ktorý patria k istému zákazníkovi (často zamestnávajúca spoločnosť), prípadne sú jednotlivcami.

- *Úprava profilu:*

  Každý aktér vstupujúci do systému má vytvorený účet, pod ktorým sa môže prihlásiť a samozrejme meniť svoje heslo, adresu a pod.

- *Fakturácia zákazníkov:*

  Ide predovšetkým o zobrazenie prehľadu termínov, ktoré daný zákazník resp. účastník absolvoval a prípadne o vytlačenie danej zostavy termínov v papierovej podobe.

- *Prejavovanie záujmu účastníka o termín:*

  Účastník, ktorého registráciu na termín musí potvrdiť personalista, môže prejaviť záujem o niektoré z dostupných termínov školení. Tento záujem sa však ešte nezobrazuje žiadnemu aktérovi školiaceho strediska. Uvedená agenda umožňuje zákazníkom efektívnejšie plánovať a organizovať školenia.

- *Prihlasovanie účastníka na termín:*

  Tento prípad použitia môže využiť účastník, ktorého registráciu nemusí potvrdiť personalista. Jedná sa predovšetkým o jednotlivcov. U ďalšieho typu účastníkov práve v tomto prípade použitia potvrdzuje registráciu personalista, ktorý taktiež má finančný prehľad o kurzoch a ich dispozícií. Tento krok je nutný aby sa aktérom školiaceho strediska účastník zobrazil v prehľadoch a mohli potvrdiť jeho účasť na termíne.

- *Potvrdenie účastníka na termín:*

  Prípad použitia je riešený na strane školiaceho centra. Ide v podstate o zobrazenie zoznamu účastníkov, ktorý sa záväzne, po prípadnom potvrdení personalistom, prihlásili na termín. Aktér túto prihlášku potvrdí, na základe čoho je vygenerovaný pozývajúci email účastníkovi a ten, ani jeho personalista, sa už z termínu odhlásiť nemôže.

7.2.2 Prototypovanie

Vo fáze *Elaboration* je okrem určenia základných funkcií systému a ich reprezentácie prípadmi použitia, nutné identifikovať najriskantnejšie a najzložitejšie časti systému. Po úvahe nad prípadmi použitia som vytýčil, ako najzákladnejší problém, vytvorenie a spravovanie termínov z pohľadu externej aj internej časti. Zložitosť vystupuje pri nutnosti dodržaní postupnosti krokov. Administrátor vypíše termín na kurz, lektor prejaví záujem a následne si administrátor vyberie, ktorý lektor bude daný termín učiť. Taktiež kritickým je už popísaný proces prihlasovania účastníkov na termín.

7.2.3 Role a práva

Neodlučiteľnú, s diagramu prípadov použití však nie veľmi jasnú požiadavku, predstavuje správa a rozdelenie samotných aktérov systému. S týmto vystupuje otázka riešenie užívateľov. Zvolil som si formu granularity práv. V podstate ide o to, že v systéme nebudú existovať rôzni aktéri, ale len jeden, ktorý bude mat prisúdené role. Každá rola predstavuje súbor práv, čo užívateľ s danou rolou môže vykonať. Týmto riešením získam v podstate požadovaných aktérov. Výhody predstavuje možnosť dodatočne vytvárať ďalšie role, s konkrétnymi právami, prípadne priradzovať užívateľovi práva bez rolí.

Uvedené riešenie som musel navrhnúť pred samotným konceptuálnym modelom, práve kvôli jeho značnému zjednodušenie a väčšej podobnosti s následnou implemetáciou na programovej a databázovej vrstve. Riešenie ja natoľko abstraktné že je súčasťou vytvoreného frameworku a môže byť nasadené v ďalších systémoch.

Nasledujúca tabuľka predstavuje riešenie aktérov systému použitím granularity rolí.

Časť aplikácie	**Právo**	**Admin**	**Adm. Prac.**	**Lektor**	**Riaditeľ**
Systém	Prehliadanie logu	✓	✓		✓
	Nápoveda	✓	✓	✓	✓
Kurz	Insert	✓	✓		
	Delete	✓	✓		
	Modify	✓	✓		
	View	✓	✓	✓	✓
Termín	Insert	✓	✓		
	Delete	✓	✓		
	Modify	✓	✓		
	View	✓	✓	✓	✓
Učebňa	Insert	✓	✓		

	Delete	✓	✓		
	Modify	✓	✓		
	View	✓	✓	Iba seba	✓
Kvalifikácie	Insert	✓	✓		
	Delete	✓	✓		
	Modify	✓	✓		
	View	✓	✓	✓	✓
Užívatelia (admin, lector, director)	Insert	✓			
	Delete	✓			
	Modify	✓	Iba seba	Iba seba	
	View	✓	✓	Iba seba	✓

Tabuľka 7.1 Role a práva užívateľov (interná časť)

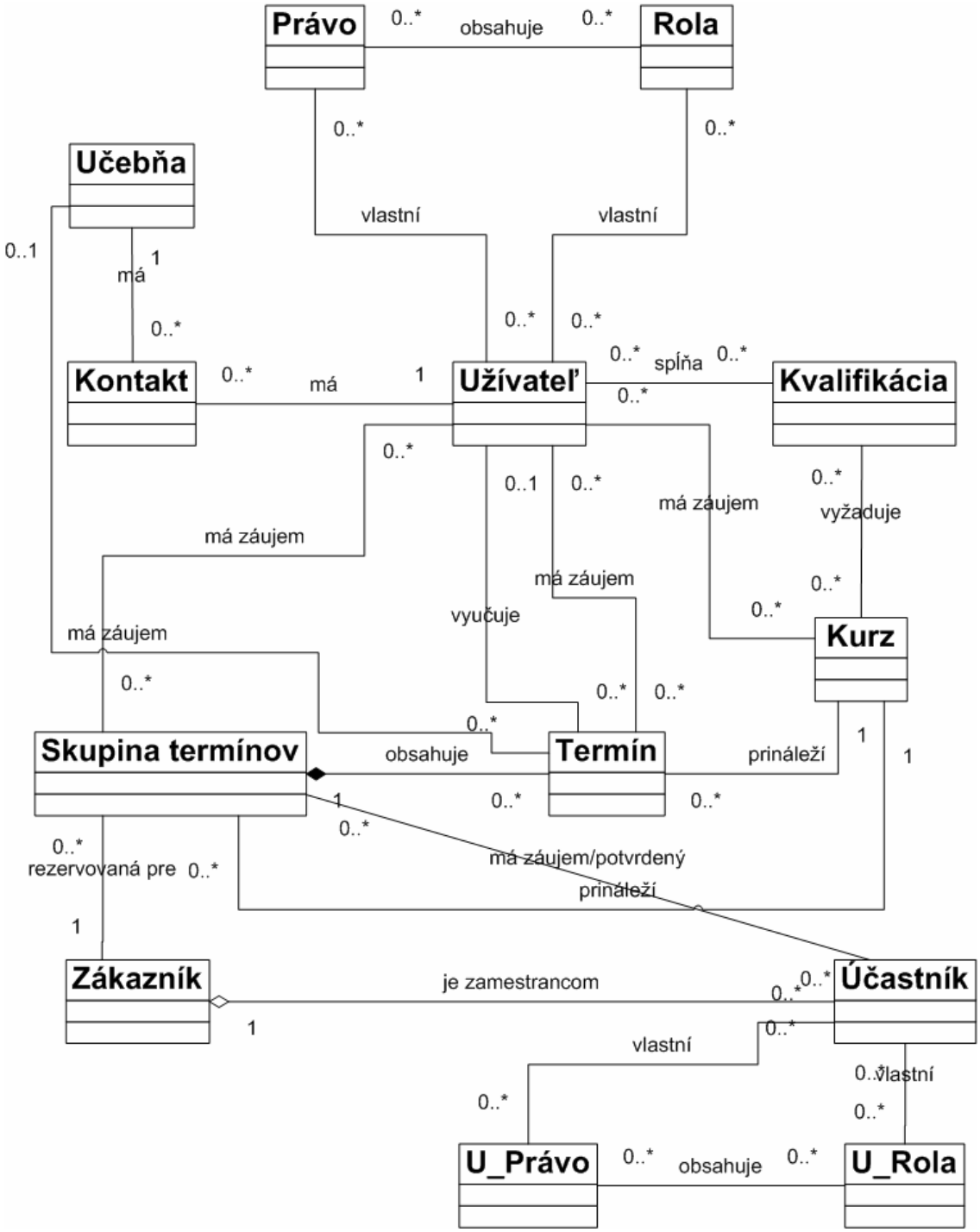
Časť aplikácie	**Právo**	**Admin**	**Adm. Prac.**	**Personalista**[11]	**Účastník**
Termín	Záujem				✓
	Registrácia	✓	✓	✓	✓
	Potvrdenie	✓	✓		
	Prehľad	✓	✓	✓	✓
Kurz	Insert	✓	✓		
	Delete	✓	✓		
	Modify	✓	✓		
	View	✓	✓	✓	✓
Účastníci	Insert	✓	✓	✓	
	Delete	✓	✓	✓	
	Modify	✓	✓	✓	
	View	✓	✓	✓	Iba seba
Zákazníci	Insert	✓	✓		
	Delete	✓	✓		
	Modify	✓	✓		
	View	✓	✓		

Tabuľka 7.2 Role a práva užívateľov (interná časť)

[11] Samozrejmosťou je, že personalista vidí iba profily svojich účastníkov a taktiež na termíny môže registrovať iba účastníkov príslušných danému zákazníkovi, ktorého je personalistom.

7.2.4 Konceptuálny model

V tejto fáze som vytvoril jednoduchý konceptuálny model, ktorý ešte neobsahuje všetky atribúty, ale slúži na pochopenie zložitých vzťahov v systéme. Predstavuje základnú kostru pre *Diagam tried.*



Obrázok 7.3 Konceptuálny model

Z uvedeného modelu bolo identifikovaných niekoľko podstatných entít:

- *Užívateľ:*

  Predstavuje entitu užívateľa systému, ktoré vlastní role, prípadne aj konkrétne práva. Reprezentantmi, tejto entity sú aktéri z *Diagramu prípadov použitia*: Administrátor, Administratívny pracovník, Lektor, Riaditeľ strediska.

- *Právo:*

  V každom jednom objekte v systému budú vystupovať štyri základné práva: *Insert, Delete, Modify, View*. Pri práci s entitami budú tieto práva overované. Prihlásenie do systému nepredstavuje právo užívateľa, ale ak nemá práva tak nemôže so systémom narábať a nebudú mu zobrazené ani odkazy na iné podstránky.

- *Rola:*

  Predstavuje súbor práv, ktoré môžu byť pomocou role, priradené užívateľovi naraz. Nie je možné pridať rolu a následne odobrať niektoré práva, ktoré rola so sebou pre užívateľa nesie. Danú požiadavku je možné riešiť priradením role s menej právami a následným pridaním ďalších práv. Rola môže zohrávať rolu vytvárania aktérov podľa špecifikácie zadania.

- *Učebňa:*

  Reprezentuje entitu, ktorá je nutné priradiť k termínu. Je samozrejme požadované udržiavať záznamy o základných informáciách učebne.

- *Kvalifikácia:*

  Označuje pomenovanú kvalifikáciu, ktorú lektor dosiahol. Kurz obsahuje odkaz na kvalifikácie, čím reprezentuje požiadavku na kvality lektora. Informácie o kvalifikácií však nepredstavujú obmedzenie prihlasovania lektorov na kurzy, resp. toto obmedzenie je možné zapnúť.

- *Kurz:*

  Reprezentuje predovšetkým informácie o ponúkanom školení ale nie termínoch. Predmetom entity kurzu je uchovávanie informácií, ako je osnova a obsah školenia.

- *Termín:*

  Predstavuje spojitý časový úsek vyučovania jedným lektorom, v jednej učebni. Na základe odučených termínov sa bude počítať pláca lektora. Každý termín musí byť nutne súčasťou jednej *Skupiny Termínov*. Účastníci sa nemôžu prihlásiť na termín, len na skupinu termínov.

- *Skupina termínov:*

  Označuje súbor termínov, ktoré spolu súvisia. Nepredstavuje však jednotku od ktorej by závisela pláca lektorov. Skupina termínov, môže existovať bez akéhokoľvek priradeného

termínu, avšak potom slúži len ako šablóna pre vytvorenie skutočných termínov, na ktoré sa môžu účastníci prihlásiť.

- *Zákazník:*

  Predstavuje entitu systému ktorá reprezentuje istú firmu, ktorá sa stala zákazníkom školiaceho centra. Je predovšetkým organizačnou entitou, ktorej súčasťou sa stávajú účastníci. Do systému sa môže prihlásiť len účastník, nie zákazník.

- *Účastník:*

  Predstavuje entitu užívateľa externého systému, ktoré vlastní role, prípadne aj konkrétne práva. Reprezentantmi, tejto entity sú aktéri z *Diagramu prípadov použitia*: Účastník, Personalista.

- *U_Právo:*

  Tieto práva sú priradzované účastníkom. Jedná sa taktiež o základné práva: *Insert, Delete, Modify, View*, ale okrem nich sa tu nachádzajú predovšetkým práva na registráciu a prejavovanie záujmu o školenia, ktoré sú podstatné pri implementácie uvedených procesov.

- *U_Rola:*

  Predstavuje súbor práv, ktoré môžu byť pomocou role, priradené účastníkovi naraz. Na základe príslušnosti účastníka do role sa môže stať jedným z aktérov: Účastník, Personalista. Prípadne môže byť vytvorená ďalšia rola kombináciou vhodných práv.

7.3 Construction

Táto fáza predstavovala časovo najnáročnejšiu časť projektu. Na jej začiatku som rozdelil celý projekt do niekoľkých, jednotlivo implementovaných iterácií, ktorých výsledkom bola zakaždým funkčná časť aplikácie.

V každej iterácií som musel vytvoriť, po podrobnej analýze požiadaviek a komunikácie so zadávateľom, predovšetkým dôkladný *Prípad použitia* a *Diagram tried*. *Prípad použitia* vystihuje následne implementovanú iteráciu a *Diagram tried* slúži ak kostra pre samotnú implementáciu od databázy až po užívateľské rozhranie.

Nasledujúci prehľad uvádza základné iterácie v ktorých bol vytváraný projekt. Vzhľadom na veľký rozsah projektu je pri každej iterácií uvedený zbežný postup a podstata jej implementácia. Po uvedenom prehľade je jedna iterácia rozobraná podrobne a sú ukázané jej jednotlivé časti.

Iterácie:

- *Autentizácia a práva:*

  Predstavuje vytvorenie tabuľky užívateľa, práv a rolí v databáze. Predovšetkým je tu riešená autentifikácia užívateľa voči systému. Tabuľku užívateľov v databáze som zatiaľ

editoval len v prostredí databázového servra a slúžila len ako zdroj základných dát: ID, login, heslo. Užívateľ pri prihlásení do systému získa práva, ktoré pochádzajú zo skupín, alebo konkrétne priradených práv. Správu skupín som implementoval aj do grafického rozhrania, ale práva je nutné editovať v databáze, pretože sú používané priamo v kóde, ako spôsob autorizácie užívateľa na konkrétny úkon. Užívateľ po prihlásené získa tzv. *Ticket* ktorý predstavuje jeho login a ID. Tieto informácie užívateľ vlastní počas celého spojenia a každá stránka overí na ich základe práva prihláseného užívateľa. Výsledkom fáze je funkčné prihlásenie užívateľa do systému, ktorá bolo zakomponované do frameworku a použité aj pri autententifikácií účastníkov.

- *Užívatelia:*

  Fáza sa zaoberá vytvorením a správou užívateľov. Predstavuje vkladanie, mazanie, zmenu a zobrazenie zoznamu užívateľov. Taktiež som vytvoril overovanie práv, na základe ktorých ma užívateľ možnosť užívateľa mazať, editovať a vkladať. V podstate predstavuje užívateľ entitu svojim obsahom podobnú, v následných iteráciach implementovaným, lektorom, učebniam, kurzom, kvalifikáciám, zákazníkom, účastníkom. Podobnosťou nemyslím obsah, ale formu. V každej z týchto entít sa jedná predovšetkým o ich správu a overovanie právomoci prihláseného užívateľa. V tejto fáze som sa snažil vytvoriť základnú grafickú podobu rozhrania a najmä jeho štandardu, ktorý bol následne využitý v ďalších fázach. Lektor predstavuje taktiež užívateľa, ktorý je v skupine Lektorov.

- *Kvalifikácie:*

  Predstavuje, podobne ako pri užívateľoch vytvorenie a správu kvalifikácie. Úlohou tejto časti je zabezpečiť pridávanie kvalifikácií do systému, ktoré budú neskôr priradzované školeniam a lektorom.

- *Učebňe:*

  V tejto fáze som sa zaoberal správou učební, čo predstavuje, ich mazanie, vkladanie, editáciu a zobrazenie zoznamu. Jedná sa opäť o komplenú implementáciu od databázy až po samotné zobrazenie. U učebne bolo nutné aj efektívne a prehľadné zobrazenie rozvrhu obsadenosti a taktiež pri vytváraní nových termínov bolo nutné zabezpečiť kontrolu, aby sa nekonali dva kurzy v rovnakom termíne na rovnakej učebni.

- *Kontakty:*

  Existovali dve možnosti riešenie kontaktov v systéme. Jednu alternatívu predstavovalo uloženie kontaktných informácii k užívateľovi, lektorovi, zákazníkovi, učebni,... Druhú, pokročilejšiu možnosť, predstavuje vytvorenie jednej tabuľky kontaktov pre všetky entity, a ich vzájomné spojenie. Výsledkom je možnosť pridelenie každej entite väčšieho počtu kontaktov. Taktiež špeciálna je implementácia v podobe *User Control*, čo predstavuje vytvorený modul správy kontaktov, ktorý stačí len pridať k ľubovolnej entite.

- *Kurzy:*

Iterácie predstavuje vytváranie a správu kurzov. Jedná sa predovšetkým o ukladanie informácií o školení. V tejto iterácii som sa nezaoberal správou termínov k danému kurzu.

- *Termíny:*

  Jedná sa o vytvorenie a správu termínov príslušných ku konkrétnemu kurzu. Ako už vyplýva z *Konceptuálneho modelu,* jedná sa nie len o vytvorenie jednoduchých termínov, ale predovšetkým o vytvorenie skupín termínov. Z toho vyplýva možnosť pri každom kurze vytvorenia novej skupiny termínov, kde je prípustné nastaviť časový rozsah a opakovanie v určitých dňoch. Skupina termínov predstavuje len šablónu konkrétneho školenie, teda ak nastavím všetky parametre ako napr.: cenu, lektora, učebňu,... tak sú vygenerované termíny, ktoré sú uložené do databázy. Na konkrétnom termíne nie je možnosť určenie jeho časového rozsahu v priebehu viacerých dní pretože musí predstavovať jeden spojitý časový úsek. Termíny ponúkajú možnosť nastaviť v konkrétny deň školenia inú učebni, iného lektora, prípadne upravený čas. Termín nie je možné vytvoriť samostatne, ale len ako súčasť skupiny termínov.

  Účastník sa prihlasuje len na kompletnú skupinu termínov a nemá možnosť sa prihlásiť napr.: iba na jeden deň týždňového školenia, čo by bolo neprípustné. Lektor je však vyplácaný za odučené termíny, nie skupiny termínov.

- *Termíny lektora:*

  V tejto iterácii som riešil prejavenie záujmu lektora o konkrétnu skupinu termínov a o termín. Lektor po vstupe do systému má možnosť výberu školení o ktoré má záujem a následne je mu zobrazená ponuka vypísaných skupín termínov a termínov. Lektor zaškrtne o ktorú skupinu termínov alebo termín má záujem. Ak zvolí skupinu termínov, tak automaticky sa predpokladá, že má záujem školiť každý termín obsiahnutý v školení. Ak lektor má záujem učiť len konkrétny termín, tak ho musí konkrétne v skupine termínov zvoliť.

- *Priradenie lektora:*

  Jedná sa o priradenie termínu lektorovi. Užívateľ s právami editácie termínu v možnosti „Lektor“ vidí záujemcov o daný termín, alebo celú skupinu termínov. Vyučovanie môže potvrdiť iba jednému záujemcovi, ktorému je následne celá skupina termínov, resp. termín, priradená. Lektorovi je poslaný email s informáciami o školení. Takto priradený termín lektor nemôže v svojom profile odmietnuť a ani odobrať záujem o učenie daného kurzu.

- *Zákazníci:*

  V tejto iterácií bola riešená správa zákazníkov, kde sa jednalo o zobrazenie zákazníkov podľa určitých filtrov. Podstatnou častou riešenou v tejto iterácií bola správa účastníkov konkrétneho zákazníka, kde sa jednalo taktiež o vytvorenie prehľadných zoznamov a možnosti hľadania účastníka podľa začiatočných písmen priezviska. Podobne ako v internej časti aj v tejto iterácií bolo nutné vyriešiť správu práv a rolí pre účastníkov.

- *Prihlasovanie účastníkov:*

  Celá uvedená iterácia sa zameriava na implementáciu procesu prihlasovania účastníkov na kurzy. Predovšetkým bolo nutné vytvoriť vlastnú časť systému v ktorej účastník vidí kurzy na ktoré je prihlásený a taktiež ma možnosť sa prihlásiť, resp. prejaviť záujem o určitý termín. Podstatné bolo nastavenie vhodných práv pre personalistov, aby mohli pracovať iba so svojimi účastníkmi a kurzami na ktoré majú právo prihlásenia.
- *Potvrdenie účastníkov:*

  Úlohou administrátora alebo adm. pracovníka je potvrdiť prihlásenie účastníka. Bolo nutné naimplementovať zobrazenie zoznamu skupín termínov a účastníkov, ktorý sú na daný termín registrovaný. Z takto priradenej skupiny termínov sa účastník už nemá právo odhlásiť, prípadne musí kontaktovať školiace centrum. Samozrejmosťou je upozornenie účastníka emailom na miesto a čas konania daného školenia.

7.3.1 Analýza

V týchto kapitolách uvediem ako príklad realizáciu iterácie týkajúcej sa užívateľa.

Pri analýze som už musel zozbierať konkrétne a presné požiadavky zadávateľa na danú iteráciu, z ktorých som vytvoril podrobný *Diagram prípadov použitia,* týkajúcich sa užívateľa.
Príloha: **Analýza Správy užívateľov**

7.3.2 Design

Fáza predstavuje prípravu ku konkrétnej implementácii danej iterácie. V tejto fáze som požiadavky zadávateľa premietol do diagramu tried, konkrétne som musel poznať všetky atribúty každej triedy a predovšetkým ich typ. V niektorých zložitejších iteráciách som vytvoril *Sekvenčný diagram*, prinajmenšom som si však rozvrhol postupnosť krokov a interakciu objektov, ktoré vedú k požadovanému výsledku, špecifikovanému v *Analýze.*

Najpodstatnejším výsledkom je *Diagram tried,* ktoré nepredstavuje len atribúty, ale taktiež metódy a dedičnosť. Väčšina tried patrí do jednej z dvoch základných skupín. Buď sa jedná o *základnú* triedu, ktorá nesie sebou atribúty objektu za účelom uchovania informácií, a metódy pre prácu s nimi. Druhu skupinu predstavujú triedy *kolekcie.* Jedná sa o zoznam objektov, prislúchajúcich k rovnakej *základnej* triede. *Kolekcie* obsahujú samozrejme vlastné metódy, ktorý zabezpečujú prácu s nimi.

Metódy základnej triedy:

- `Delete(ID)`

  Zabezpečí vymazanie objektu z databázy, najčastejšie podľa Identifikátora.
- `object Load(ID)`

Metóda, ktorá na základe ID vráti konkrétny naplnený objekt z databázy.

- `Save(object)`

  Metóda získa ako parameter odkaz na konkrétny objekt, ktorý ma uložiť do databázy. Ak ukladaný objekt ešte v databáze neexistuje, tak je vytvorený nový záznam v tabuľke, teda sa jedná o *Insert.* Ak však daný objekt už ID obsahuje, tak je vyvolaná funkcie *Update.*

- `LoadCollection(param, collection)`

  Metóda vráti kolekciu daných objektov na základe vstupného parametra, ktorým môže byť napr.: ID termínu, z čoho vyplýva, že chcem získať kolekciu všetkých objektov užívateľov, ktorý daný termín majú záujem školiť. Metód *LoadCollection* podľa rôznych parametrov existuje pre každú triedu zvyčajne väčšie množstvo a sú pridávané postupne, podľa vzniku požiadaviek v ďalších iteráciách.

Metódy triedy kolekcie:

- `Add(object)`

  Každá kolekcia obsahuje objekty len určitej triedy, táto metóda pridá do zoznamu objektov kolekcie nový objekt danej triedy.

- `Remove(object)`

  Objekt je odstránený z kolekcie.

- `Insert(index, object)`

  Metóda pridáva objekt na konkrétnu pozíciu *index* v kolekcii. Zoznam objektov v kolekcii predstavuje usporiadanie s počiatočným indexom 0.

- `int IndexOf(object)`

  Metóda vráti index dotazovaného objektu v databáze.

- `object this[index]`

  Z kolekcie vráti metóda objekt na danej pozícii *index.*

Nasledujúci graf a popis už pojednáva o konkrétnom riešení užívateľa, ktorého realizácia až do finálnej podoby nás bude sprevádzať týmito kapitolami.

User class:

Trieda užívateľa obsahuje všetky hore popísané metódy, ktoré obsahuje každá *základná* trieda. Okrem toho obsahuje metódy, ktoré boli pridané až v ďalších iteráciach a predstavujú autentifikáciu, získanie kolekcie užívateľov, ktorý majú záujem školiť termín a celú skupinu termínov, a metódu zabezpečujúcu reset hesla.

UserCollection class:

Predstavuje triedu, ktorej objekt obsahuje zoznam objektov užívateľov. Trieda poskytuje základné, už popísané metódy. Vlastnosti samotnej kolekcie sa dedia z abstraktnej triedy *CollectionBase.*

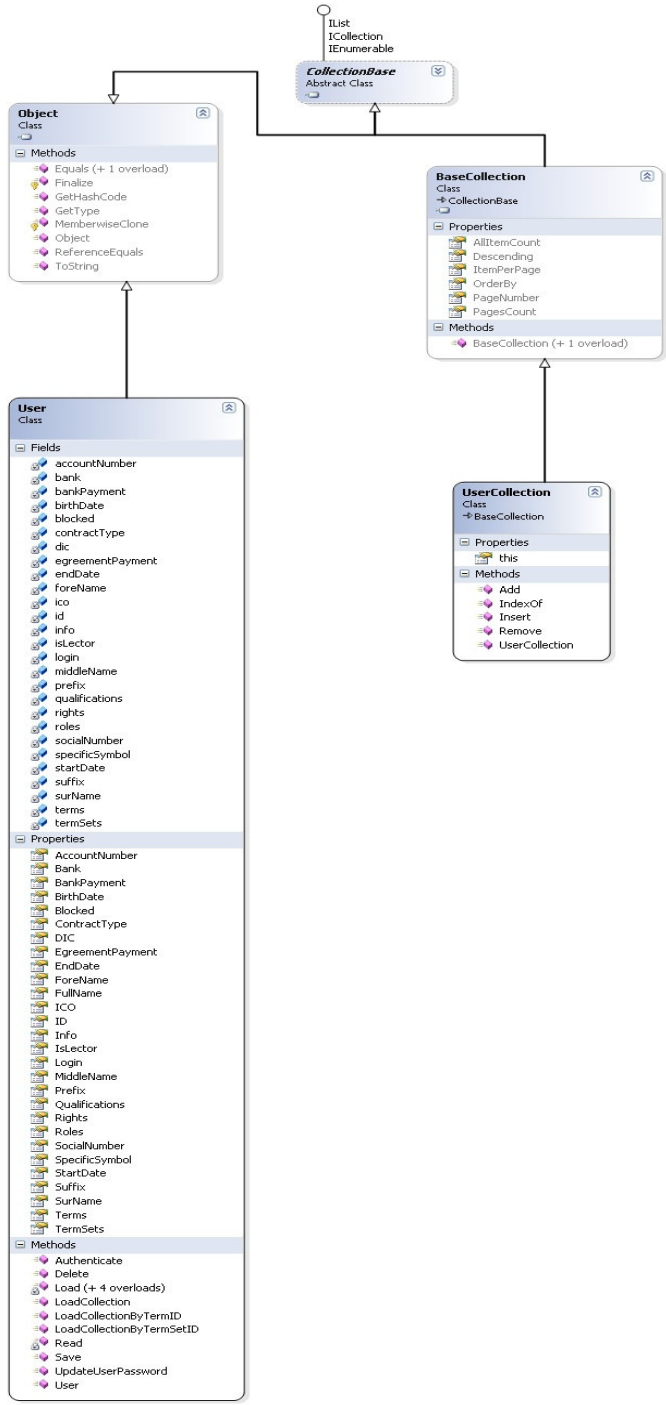
BaseCollection:

Jedná sa o mnou vytvorené triedu, ktorá je potomkom *CollectionBase* a všetky triedy kolekcii sú potomkami triedy *BaseCollection*. Jej podstatou je uchovávať hodnoty, prostredníctvom svojich verejných vlastností (Properties), ktoré sa týkajú zoznamu. Úlohou každej kolekcie je poskytnúť dátový základ pre zobrazenie zoznamu v užívateľskom rozhraní. Tento zoznam by mal byť stránkovaný a poskytovať možnosť radenia podľa stĺpcov. Vlastnosti na ktorej stránke zoznamu, z koľkých položiek sa zoznam skladá, podľa akého stĺpca radím, či vzostupne alebo zostupne si pamätá trieda *kolekcie* práve vďaka vlastnostiam zdedeným z tejto *BaseCollection.*

Položky *Fields* triedy predstavujú jej neverejné atribúty, ktoré slúžia na uchovanie konkrétnych vlastností objektu. K atribútom sa pristupuje pomocou vlastností, ku ktorým sa pristupuje pomocou *get* a *set,* kde *get* predstavuje možnosť získať hodnotu atribútu objektu a *set*, možnosť hodnotu atribútu zmeniť alebo nastaviť. Toto riešenie je podstatné pri zachovaní objektových paradigiem, a predstavuje zapuzdrenie objektu.

```
public string ForeName
{
    get { return foreName; }
    set { foreName = value; }
}
```

Obrázok 7.4 Vlastnosť užívateľa sprístupňujúca jeho meno



Obrázok 7.5 Diagram tried týkajúcich sa užívateľa[12]

7.3.3 Implementácia

Časť pojednáva už o konkrétnej programovej a databázovej realizácii. Musel som sa rozhodnúť pre určitý spôsob realizácie projektu a jeho rozvrhnutia. Zvolil som trojvrstvovú architektúru, ktorú

[12] Kompletný diagram tried (Class_diagram.png) sa nachádza na priloženom DVD v adresári Documents.

považujem na realizáciou tohto projektu za najvýhodnejšiu. Táto architektúra logicky rozdeľuje aplikáciu do vrstiev, ktoré sú navzájom nezávislé a komunikujú len prísne špecifikovanými povolenými spôsobmi. Model je výhodný najmä pre veľké projekty, kde je nutná efektívna spolupráca viacerých programátorov. Architektúra zabezpečuje možnosť vytvárania komponent v iných programovacích jazykoch ak ich vzájomná komunikácia bude napr. prostredníctvom XML, čo bolo neskôr uplatnené v spolupráci s externými systémami.

Model sa skladá z troch základných vrstiev:

1. **Presentation Layer**:

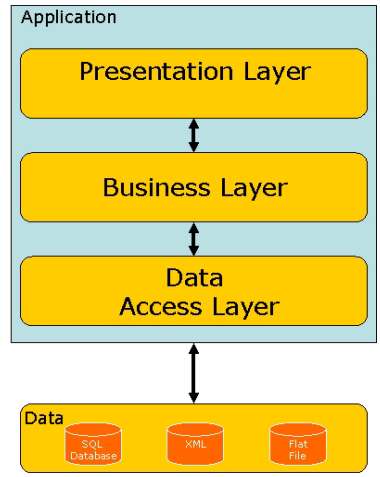
   Vrstva poskytujúca užívateľské rozhranie. Nesmie pristupovať priamo na dáta, tým sa zabezpečí jej nezávislosť. Pri webových aplikáciách môže byť medzi ňou a *Business* vrstvou napr. firewall. Táto vrstva sa môže jednoducho meniť na windowsovú, webovú, mobilnú aplikáciu, prípadne webovú službu.

2. **Business Layer**:

   Vrstva poskytujúca logiku aplikácie. Obsahuje rozhranie medzi *Data Access* a *Presentation*. Vykonáva všetky požiadavky na získavanie, úpravu a mazanie dát. Tieto požiadavky formalizuje presnejšie a dotazuje sa *Data Access* vrstvy. Mohla by obsahovať ešte tzv.: *Business Rules*, časť v ktorej by kontrolovala korektnosť požiadavkou na databázu, správnosť typov a podobne. Ja som logiku kontroly prenechal na samotnú databázu. *Business* vrstva, na rozdiel od *DataAccess,* nemá vystavané triedy nad všetkými tabuľkami pretože vo svojich metódach môže zjednocovať viaceré *DataAccess* triedy.

3. **Data Access Layer**:

   Vrstva poskytujúca prístup k databáze. Ona jediná môže volať SQL procedúry databázového servera a poskytovať im parametre, alebo len volať priamo operácie upravujúce dáta v databáze alebo XML súbore. Triedy tejto vrstvy sú postavené nad všetkými tabuľkami databázy, aj nad relačnými, ktoré len vyjadrujú vzťah N ku N. SQL procedúry zabezpečujú vykonávanie základných metód *DataAcces* vrstvy v databáze.



Obrázok 7.6 Trojvrstvová architektúra[13]

Uvedený trojvrstvový model je silne založený na objektovej architektúre, ktorá predstavuje pohľad na komunikáciu medzi vrstvami komponentovej architektúry. Podstata riešenia spočíva v uložení logického zoskupenia dát do objektu, získaného z *diagramu tried.* Objekt je logickým zjednotením metód práce s objektom a hodnôt z databázy, ktoré spolu súvisia a môžu byť aj vo viacerých fyzických tabuľkách. *Business* a *DataAccess* poskytujú základné metódy na vytváranie, mazanie a editovanie takýchto objektov. Vrstva *Business* používa metódy z *DataAccess.*

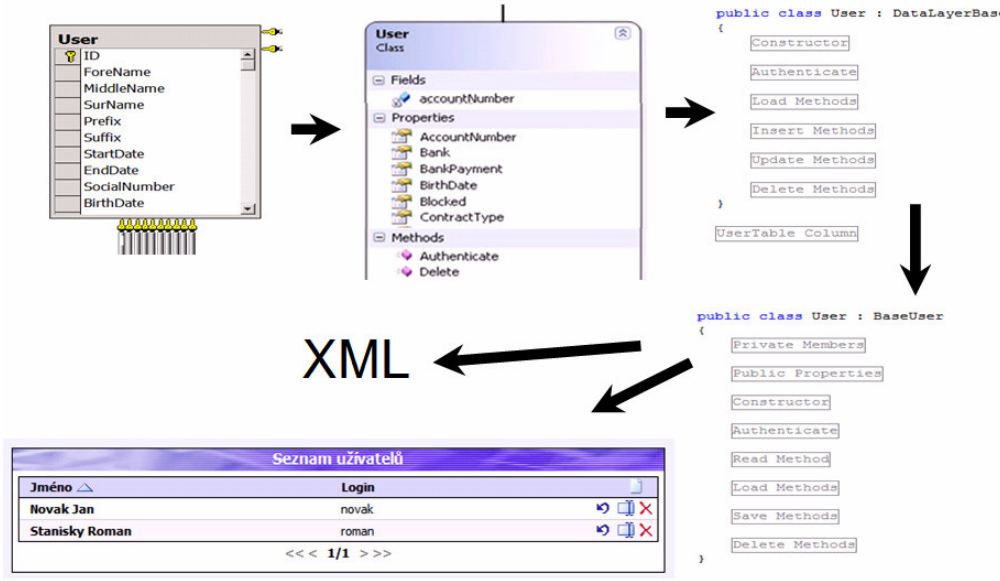
Prístup umožňuje viacerým programátorom súčasne implementovať všetky vrstvy. Nie je problémom pracovať na prezentačnej vrstve, aj keď *Business* ani *DataAccess* vrstva ešte nie je hotová. Stačí sa dohodnúť, že *Business* logika bude obsahovať metódy *Save*, *Load*, *Delete* pre daný objekt a programátora *Prezentačnej vrstvy* nemusí zaujímať riešenie *Business* vrstvy. On len vie, že má zavolať danú metódu s príslušnými parametrami, ktorých obsah získa napr. z formulára a *Business* logika sa už postará o bezpečné vytvorenie a uloženie objektu do databázy volaním metód z *Data Access.*

Uvedený prístup predstavuje elegantné riešenie problematiky. Reálnym problémom mojej implementácie, bola zjavná pomalosť napredovania prezentačnej vrstvy, pretože som pracoval na *Business logike* a *DataAccess* vrstve. Jednoduchšie riešenie tohto problému by priniesla realizácia v jednej vrstve, ktorú by som značne rýchlejšie vytvoril v grafickom vývojovom rozhraní. Tento druh realizácie by však poskytoval len obmedzené možnosti ďalšieho vývoja projektu, a prípadnej spolupráce viacerých programátorov na jeho realizácii. Riešenie neskôr zistených chýb by bolo náročné a mohlo by predĺžiť časť vývoja. Použitím vrstvovej architektúry môžem daný problém rýchlo lokalizovať a prehľadne vyriešiť.

---

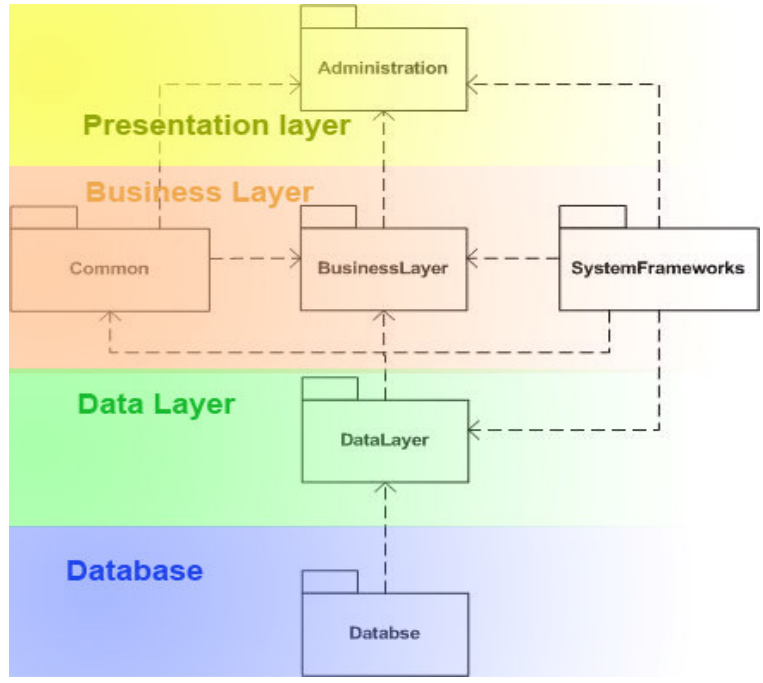
[13] Zdroj: 24.4.2007
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/FoodMovers3.asp

Čo sa týka použitia objektov v rámci trojvrstvovej architektúry, tak predstavujú základný nositeľ dát a metód krížom cez všetky vrstvy. Jediným obmedzením je použitie relačnej databáze, kde musí byť objekt v dátovej vrstve rozložený medzi jednotlivé relačné tabuľky.



Obrázok 7.7 Objektová realízácia naprieč vrstvami

Z uvedenej techniky prístupu som vytvoril *Package Diagram*, ktorý slúžil ako podklad pre navrhnutie projektu vo vývojovom prostredí: *Microsoft Visual Studio 2005*.



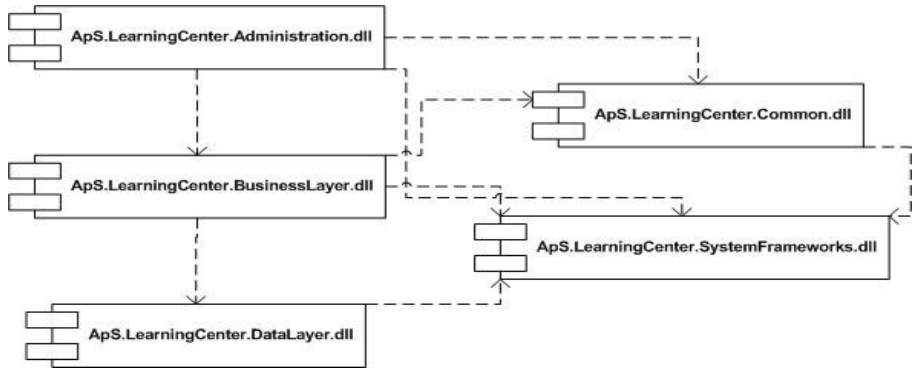
Obrázok 7.8 Package diagram projektu

Obrázok 7.9 Realizácia package diagramu

Common obsahuje triedu *BaseCollection*, z ktorej sú všetky kolekcie zdedené. Taktiež obsahuje podporné triedy pre prácu s databázou, a webovým rozhraním. Významné je podpora práce a obsah súboru *Strings.resx*, v ktorom sa nachádzajú všetky texty použité v užívateľskom rozhraní. Z uvedenej technológie prístupu vystupuje možnosť vytváranie jazykových mutácii danej aplikácie len preložením toho jedného súboru. Daný postup bol použití pri vytváraní možností jazykových mutácia v rámci viacpoužitelného frameworku.

SystemFramework obsahuje triedy poskytujúce konfiguráciu aplikácie získanú z *Web.config* a taktiež triedy s metódami pre zápis logovacích informácii.

K tejto fáze implementácie bezpodmienečne patrí aj *Diagram komponent*, ktorý predstavuje fyzickú závislosť častí aplikácie, ktoré vyplývajú z *Diagramu balíkov*.

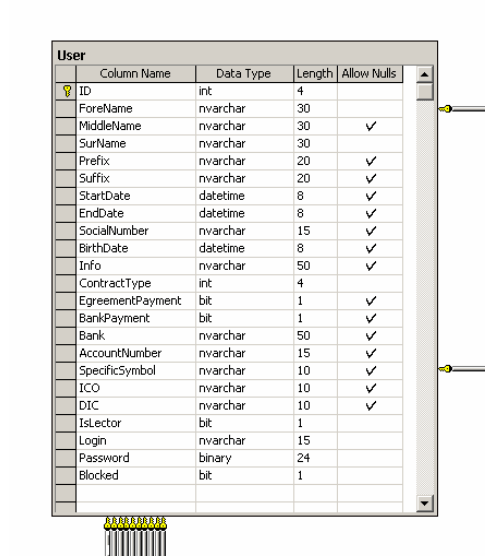


Obrázok 7.10 Diagram komponent

Ďalej uvediem konkrétny príklad riešenie práce s užívateľom na jednotlivých vrstvách.

Databáza:

Predstavuje vytvorenie tabuľky s požadovanými stĺpcami príslušných typov a napísanie základných databázových procedúr pre prácu s nimi.



Obrázok 7.11 Tabuľa užívateľa v databáze

Základné procedúry pre prácu s tabuľkou:

```
proc_UserDelete
proc_UserInsert
proc_UserUpdate
proc_UsetLoad
proc_UserLoadCollection
```

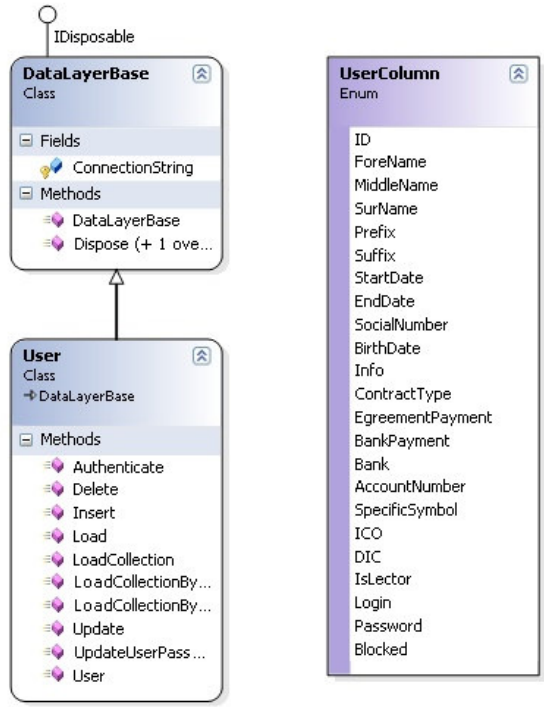
Data Layer:

Uvedená trieda *User* je zdedenou z *DataLayerBase*, ktorá poskytuje pripojenie k databáze prostredníctvom *ConnectionString*. Samotná trieda *User* obsahuje metódy pre volanie SQL procedúr uložených na serveri s príslušnými parametrami a návrat hodnôt z databázy.

Sú to predovšetkým metódy:

```
Delete
Insert
Update
Load
LoadCollection
```

V uvedenej vrstve som ku každej triede vytvoril výčtový typ, obsahujúci názvy parametrov procedúr v príslušnom poradí. Tento prístup poskytuje možnosť prehľadnej zmeny názvu parametru pri jeho úprave v databáze.



Obrázok 7.12 Triedy užívateľa v *DataLayer* vrstve

Business Layer:

Jedná sa o programovú realizáciu *Diagramu tried,* o implementáciu príslušných typov, vlastností, ktoré poskytujú k nim prístup a statických metód pre základné operácie s triedou.

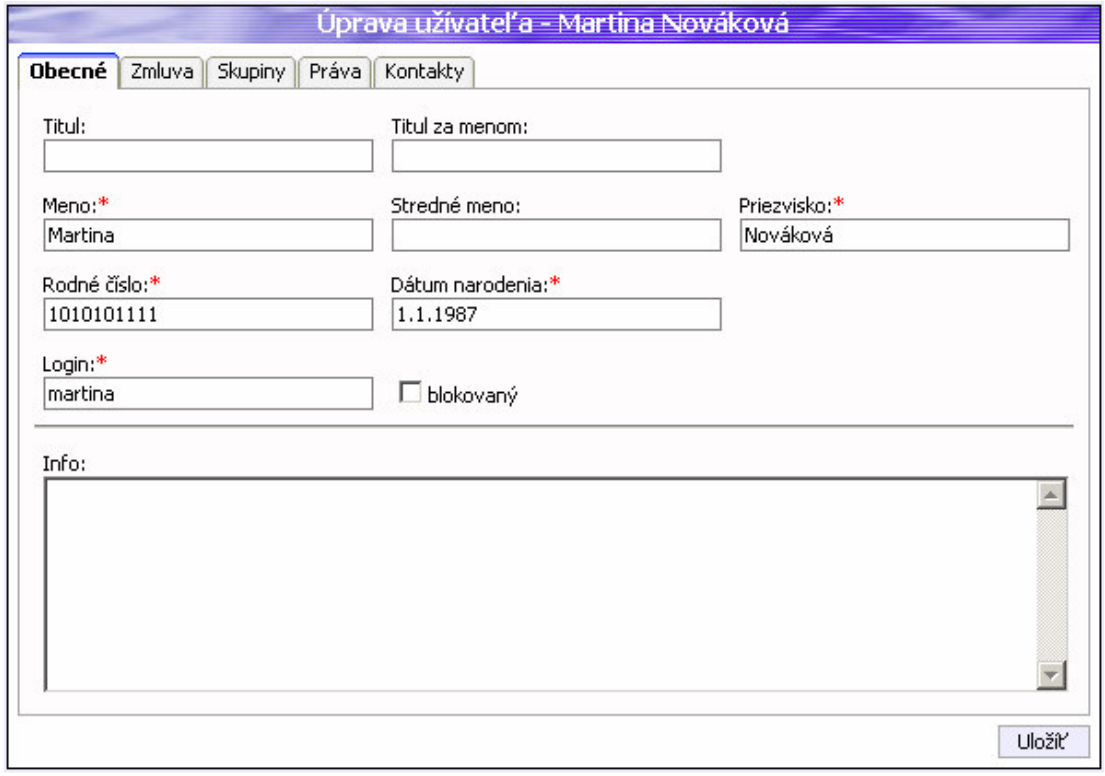
Presentation Layer:

V mojej implementácii sa jedná o *Administration package,* pretože som sa venoval časti správy systému internej ale aj externej. Na prezentačnej vrstve je možnosť v budúcnosti vytvoriť ďalšie balíky, ktoré by používali *Business Layer* a implementovaný framework.

Administration predstavuje v mojej implementácii webové užívateľské rozhranie. Použitie viacvrstvovej architektúry, však poskytuje možnosť vybudovanie balíka pre užívateľské prostredie Windows formulárov, mobilné zariadenia, alebo webové služby. Webové rozhranie je implementované v jazyku ASP.NET 2.0 a ako programovací jazyk na pozadí je použitý, podobne ako v celej aplikácii, C#. takmer každá trieda z *Business* vrstvy, v ktorej sa poväčšine nachádzajú objekty a kolekcie, je premietnutia do rozhrania v podobe formuláru, v prípade objektu, a do podoby zoznamu v prípade kolekcie. Podstatou rozhrania ja zobrazenie stránky, a korektné volanie vhodných metód z *Business* vrstvy. Pri realizácii som použil do najvyššej možnej miery technológiu *CSS* a v niektorých prípadoch som sa nezaobišiel bez použitia *JavaScriptov*.

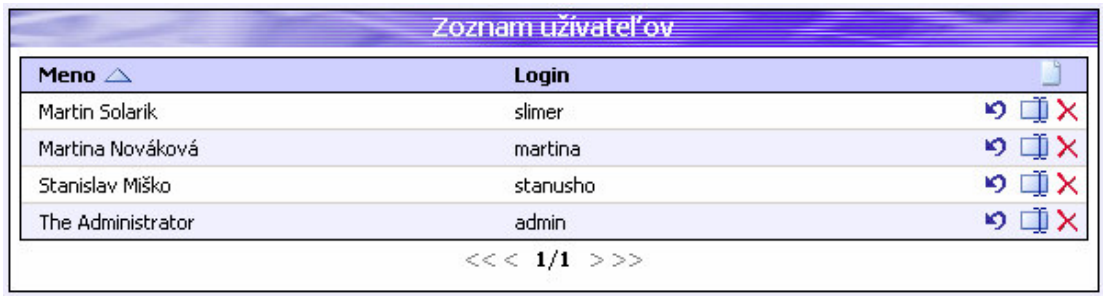
Výsledok tejto časti ako aj kompletne celej iterácie predstavuje funkčný formulár pre úpravu účastníka a prehľadný zoznam účastníkov.

Formulár: ***UserForm.aspx***



Obrázok 7.13 Formulár úpravy užívateľa

Formulár: ***UserList.aspx***



Obrázok 7.14 Formulár zoznam užívateľov

7.3.4 Testovanie

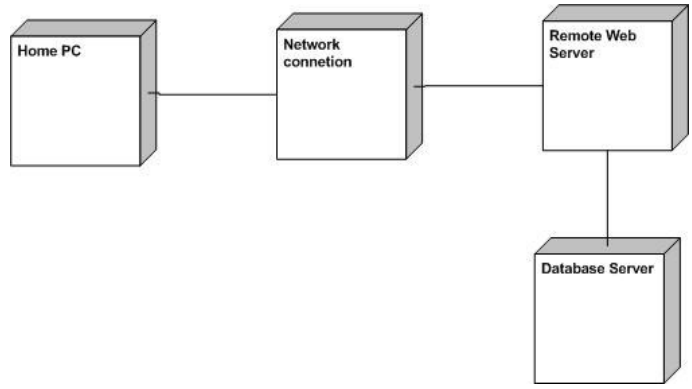
V tejto časti som testoval splnenie požiadaviek vyplývajúcich z *Diagramu prípadov použitia*, a testoval som možnosti, nevyplnenia údajov a reakciu systému na ne. Opätovne som sa vracal, najčastejšie do fázy *programovania*, kde som upravoval výslednú aplikáciu. Vďaka precíznemu návrhu, testovanie a opravy neboli časovo náročné.

7.4 Transition

7.4.1 Diagram nasadenia

Nasadenie projektu vychádza z požiadaviek zadávateľa a použitej technológie. Podstatné je pri to oddelenie databázového serveru od aplikačného webového serveru. Obe tieto služby sú outsourcované pri reálnom behu systému. Webový server poskytuje jednoduchý prístup vzdialeným klientom, ktorý môžu používať ľubovoľný internetový prehliadač na prístup do systému.

Výsledné nasadenie projektu zodpovedá uvedenému *diagramu nasadenia.*



Obrázok 7.15 Diagram nasadenia

8 Framework systému

Pri vývoji systému som sa nesnažil len splniť funkčné požiadavky zadávateľa na beh systému, ale taktiež predmetom mojej práce bolo vyvinúť abstraktné časti systému ktoré by bolo možné nasadiť aj v ďalších informačných systémoch. Výsledkom tohto úsilia bolo vytvorenie určitej množiny tried ktorá značne automatizuje a uľahčuje návrh systémov a tým zrýchľuje ich vývoj, čo je v reálnej komerčnej sfére vítaným faktorom, často rozhodujúcim o úspechu projektu.

8.1 Databáza

Výhodou trojvrstvovém architektúry je možnosť oddelenia objektov od samotného databázového zdroja. Poskytnutá implementáciu umožňuje pomerne jednoduchý prenos databázy so servera MS SQL na iný, za zmeny len niekoľkých tried dátovej vrstvy. Podstatná je práce s SQL procedúrami a využitie ich rýchlosti pri načítavaní zoznamov filtrovaných dát. Framework poskytuje metódy bezpečného spúšťania procedúr za dodania korektných parametrov správnych typov a odchytenia prípadných databázových výnimiek. Taktiež podstatné je pretypovanie niektorých dát, síce podobných, ale nie celkom rovnakých typov, ktorých rozdiel je spôsobený rozdielnou reprezentáciou daného typu v jazyku C# a jazyku T-SQL, používaným nad databázou.

8.2 Formátovanie XML

XML sa objevuje v systéme v niekoľkých rôznych podobách a prípadoch a preto vyvstala požiadavka štandardizácia prevodu objektov do formátu tohto jazyka. Tento prevod je preto riešený v abstraktnej triede *BaseCollection*, ktorá dokáže previesť celý zoznam objektov do formátu XML.

Jednoduchá reprezentácia XML sa používa pri niektorých komunikáciách s externými systémami, ktoré požadujú dáta v jednoduchom formáte XML, ktoré sú získané volaním webovej služby môjho systému.

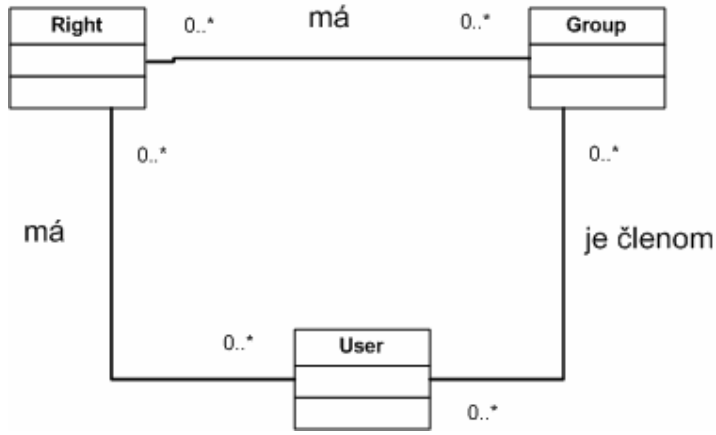
Ďalšie je použitie XML v rámci tlačových zostáv. Tu dochádza k formátovaniu XML pomocou XSLT šablóny, ktorú si samozrejme môže zákazník podľa potreby zmeniť. Toto formátovanie je čo najviac zobecnené a vložené do frameworku.

Posledné použitie formátovania predstavuje zasielanie HTML emailov. U toho bola požiadavka možnosti zmeny formátu emailu, pretože sa jedná o kontakt školiaceho strediska s klientmi, kde by bolo vhodné komunikáciu viesť pod formátovanou hlavičkou emailu, pripadne dodržiavať istý farebný štandard firmy.

8.3 Práva

Časť zabezpečenia, práv, rolí a užívateľov predstavuje najkomplexnejšiu a najzložitejšiu časť frameworku. Výhodou je značná komplexnosť, ktorá v mnohých faktoroch prekračuje požiadavky kladené na súčasný systém. Jedná sa predovšetkým o precíznu možnosť granularity práv, kde užívateľ na jednej strane získa práva zo všetkých rolí v ktorých je, ale taktiež mu môžu byť explicitne ďalšie práva priradené.

Celá funkčnosť je založená na troch tabuľkách v databáze vytvorených podľa konceptuálneho modelu.

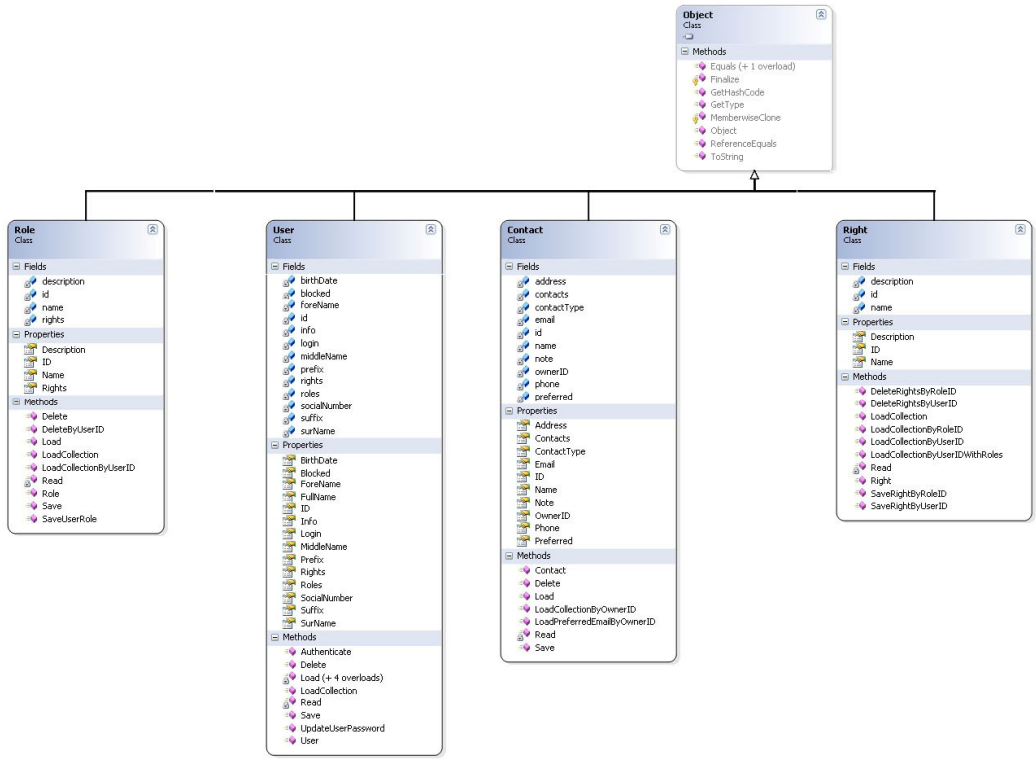


Obrázok 8.1 Konceptuálny model práv užívateľov

Z tohto modelu plynie, že užívateľ môže byť členom viacerých skupín zároveň a každej skupine môže byť priradených niekoľko práva. Užívateľovi môžu byť priradené práva aj bez členstva v určitej skupine.

Práva majú programátorom dané názvy a sú uložené aj s popisom v databáze. Samozrejmosťou je používanie výstižných mien ako napr.: *user_insert, course_view* a pod. Užívateľ sa do systému autentifikuje svojim loginom a heslom, ktoré je na strane databáze šifrované. Po úspešnej autentifikácii sa vytvorí prihlásenému užívateľovi tzv. ticket, ktorý nesie v sebe identifikátory všetkých práv, ktoré má užívateľ. Ticket je riešený použitím technológie *Session*, tá nesie informácie na serveri príslušné k danému internetovému spojenou klienta. Každá webová stránka ma určené, na ktoré práva sa má dotazovať a podľa ich prítomnosti zobrazí svoje časti prípadne sa nezobrazí vôbec. Napr. ak užívateľ nemá právo *course_delete*, ale len *course_view*, tak sa mu daná stránka zobrazí ale neuvidí tlačítka, ktoré umožňujú mazanie kurzov.

Princíp tohto frameworku už bol využití v dvoch ďalších nezávislých informačných systémov a značne urýchlil ich realizáciu.



Obrázok 8.2 Diagram tried frameworku práv

8.4 Jazykové mutácie

Pri implementácií systémov v dnešnej dobe sú práve často vyžadované ich rôzne jazykové mutácie. Z tohto dôvodu som v systéme vytvoril podporu, ktorá funguje na princípe súborov *.resx*, tzv. resources, ktoré predstavujú dvojicu kľúč a hodnota. Kľúč je vpísaný pri programovaní stránky na miesto, kde sa má lokalizovaný reťazec zobraziť. Z pohľadu prekladu je výhodné, ak sa všetky reťazce nachádzajú v jednom súbore. Mnou vytvorený framework tieto vlastnosti podporuje, a jazyková mutácia sa volí na základe výberu klienta z jazykového menu. Táto informácia je uložená a uchovávaná celé pripojenie použitím technológie *Session.*

Obrázok 8.3 Menu jazykov

8.5 Logovanie informácií

Framework poskytuje jednoduchá rozhranie pre programátorov, ako môžu prostredníctvom jedného riadku zapísať chybu do logovacieho súboru. Ten je vytvorený frameworkom na základe špecifikácií v konfiguračnom súbore *web.config*.

8.6 Posielanie Emailov

Posielanie emailov, prestavuje časť systému ktorú je výhodne implementovať ako znovunasaditeľnú a práve z tohto dôvodu boli vytvorené abstraktné triedy, ktorých vstupom metód je len adresa, pripadne adresy príjemca, predmet emailu a text, ktorý môže byť formátovaný do HTML prostredníctvom XML. Spojenie so SMTP serverom a odoslanie emailu už rieši vytvorený framework na základe špecifikácie v súbore *web.config*.

9 Spolupráca s externými systémami

Z nasadeným systémom je zviazaných niekoľko ďalších, ktoré vyžadujú vzájomnú spoluprácu. Prevažne je poskytovaná prostredníctvom zdieľaných metód so systémom nasadenom na rovnakom servery, alebo prostredníctvom webových služieb.

9.1 Verejné webové stránky

Systém verejných webových stránok bol vytvorená školiacim strediskom nad technológiou .NET Framework za použitia podporného modulového nástroja .NET Nuke. Jedná sa o web, ktorým sa firma prezentuje pred verejnosťou. Nachádzajú sa tam klasické informácie, ako napr: kontakty, adresa, mapa, ponuka školení, termíny, a pod. Pri implementácií tohto rozhrania vyvstala požiadavka spolupráce s mnou implementovaným systémom, práve za účelom získanie dostupných termínov a kurzov, ktoré sú vytvárané v internej časti implementovaného systému.

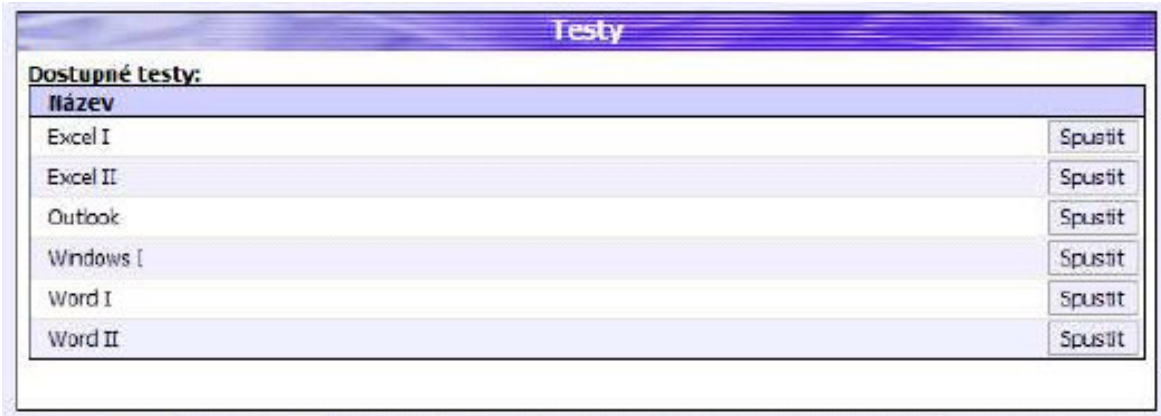
Pri riešení bolo nutné zvážiť niekoľko kritérií, ktoré implementáciu zjednodušovali. Bolo to použitie rovnakých technológii na tom istom webovom serveri. Z toho dôvodu som programátorovi týchto stránok iba odovzdal kompilované *.dll* súbory dátovej a business vrstvy a popísal metódy, ktoré môže používať. Následne si on vytvoril referenciu na dodané knižnice a mohol bez problémov pristúpiť k verejným metódam.

Touto implementáciou som dokázal jednoduchosť a jednu z praktických výhod objektového programovania, keď sa programátori iba dohodnú na názve a atribútoch určitých verejných metód a ich vlastná implementácia ich navzájom nezaujíma.

9.2 Systém testov

Po konci niektorých školení má zákazník podmienku aby všetci účastníci boli testovaný, aké znalosti získali. Samotný testovací systém je riešený v rozhraní napísanom v PHP a nemá spojenie na databázu, ktorú využívam ja v svojom systéme. Štandardne systém fungoval oddelene a účastníci pri začiatku testu zadali svoj email, prípadne meno a priezvisko. Následne na to sa im test spustil. Problém nastal, ako je popísané v kapitole *Nasadenie systému*, pri testovaní väčšieho počtu účastníkov, ktorý odhalili možnosť test vyplniť niekoľko krát, resp. nezadali korektné dáta autentifikácie a systém nemal možnosť overiť ich existenciu v databáze, nad ktorou beží môj informačný systém. Taktiež požiadavkou bolo, že nie každý zákazník môže nechať svojich účastníkov testovať každým, ale len vybranými testami.

Pri riešení toho problému bolo nutné vychádzať zo skutočnosti, že ani server ani technológia neboli rovnaké. Z tohto dôvodu som vytvoril v mojom systéme modul, kde ja možné zvoliť, ktoré testy sú pre zákazníka dostupné a taktiež na akej stránke. Následne sa účastník musí prihlásiť do externej časti systému, kde potrebuje login a heslo a nemôže informácie o sebe podvrhnúť resp. sa preklepnúť. Prihlásenému účastníkovi je poskytnutý zoznam testov, ktoré môže spustiť. Po kliknutí na určitú položku je mu zobrazená stránka testovacieho systému, ktorej sú informácie o teste a účastníkovi predané prostredníctvom parametrov webovej adresy. V okne testu nie je zobrazený adresový box, a preto tieto parametre nemôže jednoducho zmeniť. Pri veľkej snahe a použití technológie na odchytávanie HTTP komunikácie (napr. program Paros) by však mohol dané parametre podvrhnúť. Práve za účelom odstránenia danej hrozby sa práve pracuje na module frameworku, ktorý by parametre hashoval a tak účastníkovi znemožnil ich zmenu.



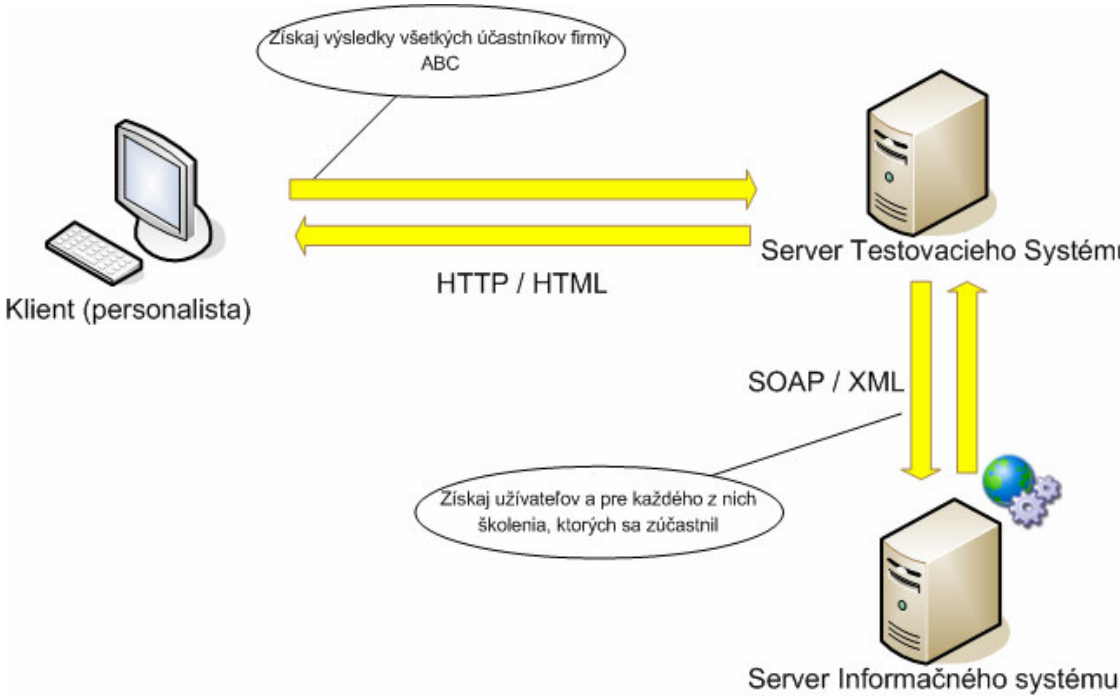
Obrázok 9.1 Odkazy na dostupné testy do testovacieho systému

9.3 Vyhodnocovanie testov

Po absolvovaní testu účastníkom nastáva proces spracovania výsledkov a ich zaslanie personalistovi. Jedná sa o proces značne automaticky, ale časovo náročný, kde ide prevažne o vytvorenie tabuliek výsledkov účastníkov. Z toho dôvodu vznikol opäť dotaz na spoluprácu systémov a automatizáciu vyhodnotenia testov.

Ako som už spomenul, testovací systém je napísaný v jazyku PHP a beží na oddelenom serveri. Preto sme sa v rámci spolupráce systémov rozhodli využiť technológiu webových služieb, ktoré používajú na výmenu informácií otvorený protokol XML a dokáže ich využívať aj PHP. Na mojej strane systému, som vytvoril webové služby, ktoré sú reprezentované metódami volateľnými cez internet z testovacieho systému. Dané služby vrátia zoznam všetkých termínov a ich popisov, ktorých sa daný účastník zúčastnil. Testovací systém dotazuje účastníkov na základe ich emailu, ktorý si eviduje vo vlastnej databáze.

Systém týmto spôsobom automatizuje a uľahčuje prácu vedúcemu školení, ktorý môže personalistom rýchlo vygenerovať aktuálne informácie o testovaní účastníkov. Opäť daná skutočnosť poskytuje školiacemu centru konkurenčnú výhodu.



Obrázok 9.1Spolupráca systémov využítím webových služieb

10 Nasadenie systému

Nasadzovanie systému prebehlo po postupnom otestovaní a paralelnom behu kontroly finančných procesov, ktoré sú pre firmu kritické. Pre lektorov však bol realizovaný prechod „veľkým treskom“, kde boli zaškolený do systému a nútený ho používať a sledovať. V súčasnej dobe prebieha postupne nasadenie zákazníckej časti, kde je predovšetkým nutné vytvoriť zákazníkom vhodné manuály a vysvetliť používané automatizované procesy.

10.1 Procesné problémy

Pri implementácii a aj integrácií boli odhalené niektoré problémy, ktoré neplynuli so samotnej implementácie, ale z používaných a navrhnutých procesov. Podstatná bola pri tom komunikácia so zadávateľom projektu a jeho komunikácia so zákazníkmi a prispôsobovanie sa potrebám týchto strán. Bolo to nutné z hľadiska praktickej funkčnosti projektu. V systéme sa často nachádzali implementované isté logické obmedzenia, ktoré bolo vo výslednom produkte nutné odstrániť.

Lektori nadobúdajú istú kvalifikáciu získavaním odborných certifikátov. Každý kurz vyžaduje určitú odbornosť lektora, ktorá by mala plynúť zo získaného certifikátu. Preto bolo pôvodne implementované obmedzenie, ktoré určovalo, že ak lektor nemá dostatočná kvalifikáciu na školenia, potom nemôže o dané školenie prejaviť záujem. Vzhľadom na to, že sa jedná o malú firmu, kde vedúci školení pozná odbornosť lektorov osobne, bolo práve toto obmedzenie zrušené a v podstate lektor sa môže prihlásiť na akékoľvek školenie.

Ďalším obmedzením bolo to, že vedúci školení nemohol na termín priradiť lektora, ktorý neprejavil záujem o školenie. Je to opäť značne logická implementácia, ale obmedzujúca. Toto obmedzenie bolo opäť nutné so systému odobrať.

Nejednalo sa samozrejme len o spomenuté procesy riadenia, ale často o zmeny ktoré vyplynuli až za behu systému a boli spôsobené ľudským faktorom. Jedná sa napr. o to, že lektor samozrejme môže vidieť sumu ktorú za daný kurz dostane a taktiež by mal mať možnosť vidieť ďalšie vlastnosti kurzu, ako sú osnova, literatúra a pod., ale nesmie vidieť cenu, ktorá je fakturovaná zákazníkovi, pretože sa jedná o značne citlivé údaje.

Ďalšou vlastnosťou podobného charakteru je, že lektor by nemal mať možnosť sledovať školenia, ktoré boli priradené kolegovi a ich sumu.

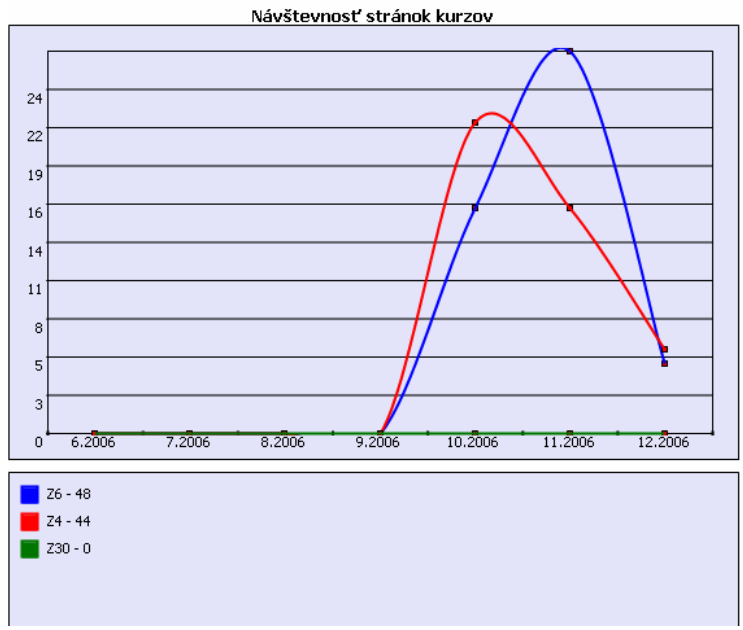
Zaujímavým je problém so zákazníkmi a testovacím rozhraním, ktorý bol nedávno riešený. Jednalo sa totiž o testovanie niekoľko stovky účastníkov. Testovacie rozhranie požaduje iba meno a email testovaného účastníka. Predpokladom funkčnosti je však korektnosť zákazníka a jeho ochota spolupracovať. Testovaný účastníci sa však často v emaile preklepli, alebo zadali schválne iné meno a email nezadali rovnaký aký je uvedený v informačnom systéme a aký bol dodaný zadávateľom

testovaní. Niektorý účastníci dokonca si zopakovali test stále pod iným menom. Z uvedeného vystal značný problém, keď nebolo možné jednoznačne účastníkov identifikovať. Preto bolo nutné okamžité riešenie problému, čo najrýchlejším zavedením autentizácie pri testovaní a spoluprácou s testovacím systémom.

10.2 Vyhodnotenie účinnosti

Systém už je v prevádzke takmer rok, a stále sa upravuje na podnet zadávateľa, lektorov alebo zákazníkov. Jeho funkčnosť priniesla značné časové úspory, ktoré môžu byť venované komunikácií so zákazníkom a získavaniu ďalších zákaziek. Systém svojou jedinečnosťou poskytuje firme konkurenčná výhodu pred ostatnými strediskami. Podstatné je napr. vyhodnotenie testov, ktoré ušetrí personalistovi hodiny práce, a taktiež možnosť plánovania účastníkov na kurzy.

Ďalšia podstatná výhoda a z ekonomického pohľadu zaujímavá časť, je vyhodnocovanie a zbieranie štatistík. Nad uvedeným informačným systémom beží aj verejná časť stránok školiaceho centra, ktorá získava dáta z databázy informačného systému. Jedná sa predovšetkým o ponúkané kurzy a termíny. Podľa toho, ako zákazníci prichádzajú na konkrétne webové stránky kurzov, sa dá určiť o ktoré kurzy je záujem a ktorých popis by mal byť vylepšený prípadne pridané termíny. Kurzy, ktorých stránky nie sú navštevované, zase naopak môžu byť zrušené a neevidované v systéme. Tieto informácie o návštevnosti taktiež prehľadne spracováva a analyzuje systém.



Obrázok 10.1 Štatistika návštevnosti kurzov

10.3 Riziká

Systém svojou integráciou a nasadením nepriniesol len výhody, ale aj niektorá nevýhody, ktorým je veľmi náročné sa vyhnúť.

Firma sa totiž výhodami informačného systému stala naň absolútne odkázaná a neeviduje ďalšie záznamy zdvojenou formou.

Ďalšou výhodou ale aj nevýhodou je, že na systéme sa stále pracuje a sa zdokonaľuje. Jednak na jeho softwarovej časti, ale aj na jeho databáze. Práve zásahy do databázy sú často kritické a je nutné ich vykonávať veľmi precízne aby neohrozili už existujúce dáta v databáze.

Veľmi náročné je otestovanie každej zmeny, či nepriniesla isté nekonzistencie do už fungujúceho systému. Len pre príklad spomeniem, že pridanie jedného stĺpca do tabuľky databázy a zabudnutie na prepísanie jednej procedúry, spôsobilo postupnú nefunkčnosť systému a kolaps objednávania školení. Samozrejme, keď sa na daný problém prišlo bolo nutné ho v priebehu hodín riešiť.

Už uvedená skutočnosť evokuje problém závislosti na autorovi, ktorý celý systém projektoval a vyvíjal. Z toho dôvodu som sa však snažil systém navrhnúť maximálne obecne s prehľadnou štruktúrou implementácie. Podstatné je, že systém používa rovnaký prístup ku všetkým objektom. Aj napriek trvaniu implementácie sa tento prístup nemenil, čím je zabezpečená konzistencie programového kódu. Svojím návrhom je systém pochopiteľný už pre mierne pokročilého programátora, ktorý po vhodnom zaškolení by bol schopný systém ďalej rozvíjať. Uvedení fakt štandardizácie kódu a prístupu je v praxi často veľmi podstatným.

11 Záver

Vzhľadom na to, že systém sa začal vyvíjať už v rámci bakalárskej práce, tak v súčasnosti už je v plnom nasadení. Ako posledný bol implementovaný značne obsiahli modul umožňujúci zákazníkom prihlasovanie na kurzy. Týmto sa otvoril systém verejnosti a bude nútený akceptovať značné vstupné, často nekorektné, dáta, čo predstavuje výzvu v jeho ďalšej implementácií.

Systém prináša jednoznačne úsporu prostriedkov, najmä času všetkým užívateľom. Taktiež začína poskytovať reprezentačnú a konkurenčnú výhodu školiacemu centru práve pre svoju jedinečnosť a funkčné prispôsobenia procesom firme za účelom ich zefektívnenia. Informačný systém svojím zavedením do praxe splnil očakávania, a množstvo nápadov a požiadaviek, nesvedči o jeho chybách, ale práve naopak o jeho opodstatnení a snahe rozvíjať a pridávať novú funkčnosť.

Značná časť vývoja je realizovaná na aktuálnej ostrej verzií, čo samozrejme sebou prináša riziká a zodpovednosť, ale taktiež umožňuje získavať stále nové skúsenosti.

V budúcnosti sa plánuje práve zdokonalenie zákazníckej časti a jej rozšírenie o rôzne prehľadové grafy a štatistické tabuľky vyžadované zákazníkmi. Taktiež sa predpokladá vytvorenie modulov vyhodnocujúcich finančnú stránku školení, náklady, fakturácie, zisky a pod. Aj v týchto častiach bude určite nutná spolupráca s externými systémami.

Pre ilustráciu mohutnosti systému je možné uviesť, že obsahuje vyše 200 súborov, cez 35000 riadkov kódu, 60 databázových tabuliek a 235 SQL serverových procedúr.

Literatúra

[Ariad-2005] Ariadne Training, *UML Applied*, 15.12.2006. Dokument dostupný na URL http://www.ariadnetraining.co.uk

[Hurwi-2003] Hurwitz, Dan, Liberty Jesse: *Programming ASP.NET*. druhé vydanie, O'Reilly 2003

[Meliš-2005] Meliška, Michal*: Od počátku až k .NET (4).* 15.12.2006, Dokument dostupný na URL http://notebooky.idnes.cz/technologie/operacni_systemy/operacni_systemy__windows476.html

[Micro-2005] Microsoft CZ: *Definice základních součástí .NET*. 16.12.2006, Dokument dostupný na URL http://www.microsoft.com/cze/net/basics/whatis.asp

[Milto-2003] Milton, D. Rosenau, Jr.: *Řízení projektu. druhé vydanie*, Brno, Computer Press 2003.

[PageJ-2001] Page-Jones, Meilir: *Základy objektově orientovaného návrhu v UML.* prvé vydanie, Praha, Grada 2001.

[Prosi-2003] Prosise, Jeff : *Programování v Microsoft .NET*. prvé vydanie, Brno, Computer Press 2003.

[Ruttk-2005] Ruttkay, L.: *Bakalárska práce*, Fakulta informačních technológií VUT, Brno 2005.

[Schmu-2001] Schmuller, Joseph: *Myslíme v jazyku UML.* prvé vydanie, Praha 2001.

[Stein-2005] Stein, René: *Návrh aplikací v jazyce UML.* 15.12.2006, Dokument dostupný na URL http://interval.cz/clanek.asp?article=2783

[Zendu-2006] Zendulka, J. a kol.: *Analýza a návrh informačních systému AIS,* Fakulta informačních technológií VUT, Brno 2006.

Zoznam použitých skratiek a symbolov

ADO	Active Data objects
API	Application Programming Interface
ASP	Active Server Pages
ATL	Active Template Library
COM	Component Object Model
FCL	Framework Class Library
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
MFC	Microsoft Foundation Class Library
PHP	Hypertext Preprocessor
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SOAP	Simple Object Access Protocol
T-SQL	Transact SQL
UML	Unified Modeling Language
WSDL	Web Services Description Language
XML	Extensible Markup Language

Zoznam príloh

Prílohy

Príloha 1. Špecifikácia požiadaviek

Základní datové struktury:

Kurz:

- název, stručná charakteristika, podrobná charakteristika, literatura, typická doba trvání, typická cena

Termín:

- konkrétní dny (začátek – konec), doba (např. ráno, dopoledne, odpoledne, podvečer, uživatelská doba), kurz, lektor, učebna, účastníci,

Lektor:

- základní kontaktní údaje, které kurzy jsou schopni lektorovat, lektoři pracují buď na Fakturu (pak by systém měl evidovat jejich fakturační údaje) nebo na Dohodu o provedení práce

Učebna:

- název, umístění, počet míst, cena za pronájem
- typy:
  1. stabilní (tzn. vlastní středisku)
  2. u zákazníka (tzn. školení se koná přímo u zákazníka)
  3. dočasné (pronájem) – lze suplovat stabilním typem

Zákazník

jedná se většinou o firmu – nejde o fyzické účastníky v kurzu (i když někdy je to možné)

- název, IČ, DIČ, kontaktní osoba + kontakt. údaje, poznámky
- ideálně CRM údaje: velikost zákazníka, způsob jeho získání (= jak se o nás dozvěděl), platební morálka, stav (aktivní/neaktivní), hodnocení, případně kategorie

Účastník

jde o fyzického účastníka, který musí mít vazbu na daného zákazníka (tzn. např. firma vyšle 5 osob na školení, musí se spárovat se zákazníkem)

- jméno, příjmení, adresu (nepovinné, pokud je stejné jako u Zákazníka), odkaz na zákazníka

Výplaty

je nutno evidovat výplaty jednotlivých lektorů za dané období (měsíc). Lektoři si buďto podají fakturu (a ta se eviduje s odkazem na jednotlivé termíny) nebo na konci měsíce se na mzdové oddělení odevzdává přehled odměn na Dohody o provedení práce. Systém v tom případě eviduje která faktura, resp. DOP je pro konkrétní termíny.

Popis funkcí:

Z hlediska administrativních pracovníků střediska:

Systém musí v první řadě zajistit organizaci kurzu, tzn. musí umožnit zadat konkrétní termín daného kurzu, přihlášení účastníků, zadání či přihlášení lektora, zadání či výběr učebny, fakturace.

zadání konkrétního termínu: - po volbě začátku kurzu by systém měl primárně nabídnout konec kurzu dle jeho délky. Možné stavy:

1. kurz se koná normálně, tzn. v nominální době a vcelku
2. kurz je zkrácen/prodloužen, avšak v celku
3. dtto 1. a 2., ale kurz není v celku (např. Po, Út a Pá)
4. individuální školení – je nutno evidovat nejen konkrétní termíny, ale rovněž počet odučených hodin
5. pravidelně se opakující kurzy – např. podvečerní, či FITu (ideálně – možnost zadat opakování jako v Outlooku). Přičemž některé termíny mohou z různých důvodů vypadnout (stát. svátek atd.) a kurz se může prodloužit.

Přihlášení účastníků: - zadám konkrétní osobu a odkaz do tabulky zákazník. Ideálně pro jednoho zákazníka zadat více účastníků najednou, možnost zvolit individuální cenu či slevu kurzu. Kontrola, zda-li byla provedena fakturace (Ano/Ne) pro všechny účastníky u každého zákazníka.

V každém stavu se účastník ve vztahu na daný běh kurzu může nacházet v některém z násl. stavů:

1. má zájem projevil zájem o konkrétní kurz, ale ještě se závazně nepřihlásil
2. přihlásil se poslal objednávku, závazně potvrdil účast atpod.
3. absolvoval kurzkurz absolvoval a byl mu vydán certifikát
4. na kurz se nedostavil ačkoli se přihlásil, nedostavil se na kurz (např. onemocněl atpod.)
5. případně další stavy, které se vyskytnou

Tyto stavy by měly být v systému ve formě katalogu (tedy uživatelsky doplňován).

Zadání lektora: - z tabulky Lektor si vyberu lektora a zadám jeho dohodnutou odměnu.

Zadání učebny: - z tabulky Učebna si vyberu konkrétní učebnu, systém přednastaví cenu za jeho pronájem, kterou mohu změnit.

Fakturace: systém mi pro každého zákazníka daného termínu zobrazí přehled fakturovaných částek (dle údajů u jednotlivého účastníka) a seznam účastníků. Systém umožní u každého zákazníka označit, zda-li v konkrétním termínu proběhla fakturace.

Dále systém by měl umožnit:

- tisk prezenční listiny
- tisk osvědčení o absolvování kurzu
- evidenci zájemců o zaslání katalogu a tisk obálky

Z hlediska zákazníka:

Systém musí umožnit přes web se zaregistrovat na konkrétní termín kurzu, systém poté zákazníkovi zašle potvrzující e-mail.

ideálně by systém mohl umožnit individuální přistup pro zákazníka (vždy na danou kontaktní osobu) s přehledem historie kurzů, které absolvovali jeho pracovníci (se seznamem), fakturovaných částek a registrovaných kurzů.

Z hlediska účastníka:

Systém musí umožnit se přes web zaregistrovat, ideálně přeregistrovat a dívat se na své zaregistrované kurzy, případně historie.

Z hlediska lektora:

Systém musí umožnit změnu kontaktních údajů, zobrazit přehled jeho kurzů (tj. kurzů na které je zaregistrován jako lektor) a neobsazených kurzů a umožnit mu se na neobsazený kurz zaregistrovat.

Ideálně by systém mohl na požádání lektorovi vyexportovat seznam jeho kurzů do rozumného formátu pro import do Outlooku, mobilních zařízení atpod.

Z hlediska ředitele střediska:

Jde především o řadu přehledů:

- seznam kurzů v nejbližší době + export do Outlooku
- seznam neobsazených či málo obsazených termínů
- obrat na jednotlivých Zákaznících, přehled jejich aktivit
- vytíženost jednotlivých učeben
- přehled odměn pro lektory

Spolupráce s externími systémy:

- Systém musí umožnit komunikaci s externími službami formou webových služeb. Jedná se především o informativní webové služby, které budou poskytovat údaje o účastnících školení, historii i budoucí plán jejich kurzů.
- Systém rovněž bude poskytovat externím systémům údaje o kurzech, jejich obsahovou náplň a koncové ceny pro běžné účastníky.
- Tyto webové služby budou chráněny proti zneužití a neoprávněném přístupu prostřednictvím standardních bezpečnostních opatření.

Príloha 2. Analýza Správy užívateľov

Správa Užívateľov:

Pridanie Užívateľa

Popis:
Umožňuje kompetentnému užívateľovi vytvoriť nového užívateľa s právami.

Vstupné podmienky:
1. Prihlásený užívateľ s právom *user_insert* a *user_view*.

Výstupné podmienky:
1. Vytvorený nový užívateľ

Hlavná cesta:
1. Užívateľ zvolí v možnostiach užívateľov „*pridať nového užívateľa*“
2. Vyplní minimálne: *Meno, Priezvisko, Login, Rodné číslo, Dátum narodenia.*.
3. Klikne „*Uložiť*“
4. Záznamy sú pridané do databázy. Heslo nového užívateľa je jeho rodné číslo. Pri prvom prihlásení do systému je nútený si heslo zmeniť.
5. Koniec *Prípadu použitia*.

Alternatívne cesty:
a. Nevyplnené dostatočné údaje, alebo údaje v chybnom formáte.
b. Login je už použitý pre iného užívateľa

Alternatívne cesty: Nevyplnené dostatočné údaje , alebo údaje v chybnom formáte
1. Zobrazí sa správa, ktorá informuje o chýbajúcom údaji, či formáte.
2. *Prípad použitia* sa pustí znova

Alternatívne cesty: Login je už použitý pre iného užívateľa
1. Zobrazí sa správa o chybe.
2. *Prípad použitia* sa pustí znova

Záznamy: Do Log súboru je pridaná záznam o vytvorení užívateľa.

Poznámky:

Odstránenie Užívateľa

Popis:
Umožňuje kompetentnému užívateľovi odstrániť existujúceho užívateľa

Vstupné podmienky:
1. Prihlásený užívateľ s právom *user_delete* a *user_view*.
2. Existujúci účet užívateľa, ktorý je určený k odstráneniu.

Výstupné podmienky:
1. Odstránený účet Užívateľa.

Hlavná cesta:

1. Kompetentný užívateľ v zozname užívateľov vyberie užívateľa, ktorého chce odstrániť a zvolí tlačítko „*Odstrániť užívateľa*“
2. Užívateľ je odstránený so systému a zároveň sú odstránený všetky položky v databáze, ktoré sa na neho odkazovali.
3. Koniec *Prípadu použitia.*

Alternatívne cesty:
a. Užívateľ nemôže odstrániť sám seba.

Alternatívne cesty: Užívateľ nemôže odstrániť sám seba
1. Zobrazí sa chybové hlásenie, že nie je možné odstrániť užívateľa.
2. *Prípad použitia* sa pustí znova

Záznamy: Do Log súboru je pridaná záznam o odstránení užívateľa.

Poznámky:

Upravenie existujúceho užívateľa

Popis:
Umožňuje upraviť profil existujúceho užívate

Vstupné podmienky:
1. Prihlásený užívateľ s právom *user_update* a *user_view*.
2. Existujúci účet užívateľa, ktorý je určený k úprave.

Výstupné podmienky:

1. Zmenený profil užívateľa.

Hlavná cesta:
1. V zozname užívateľov je vybraný jeden konkrétny tlačítkom „*Upraviť užívateľa*“.
2. Je zobrazený formulár so všetkým nastaveniami užívateľa, ktoré sú zmeniteľné.
3. Po prevedený zmien stlačí „*Uložit*“
4. Zmeny sú uložené do databázy
5. Koniec *Prípadu použitia*.

Alternatívne cesty:

Poznámky: Úpravy užívateľa sa týka taktiež zmena jeho hesla. Ako bolo spomenuté, užívateľovi je pri jeho vytvorení pridelené ako heslo jeho rodné číslo a pri prvom prihlásení je nútený ho zmeniť. Užívateľ s právom *user_reset* môže zmeniť heslo užívateľa spať na rodné číslo, avšak k pôvodnému heslu sa nedostane. Tým je zabezpečené, aby ani administrátor nemohol zistiť heslo a prihlásiť sa v kontexte iného užívateľa.

Zobrazenie zoznamu užívateľov

Popis:
Zobrazuje zoznam všetkých užívateľov, okrem lektorov, v systéme.

Vstupné podmienky:
1. Prihlásený užívateľ s právom *user_view*

Výstupné podmienky:

Hlavná cesta:

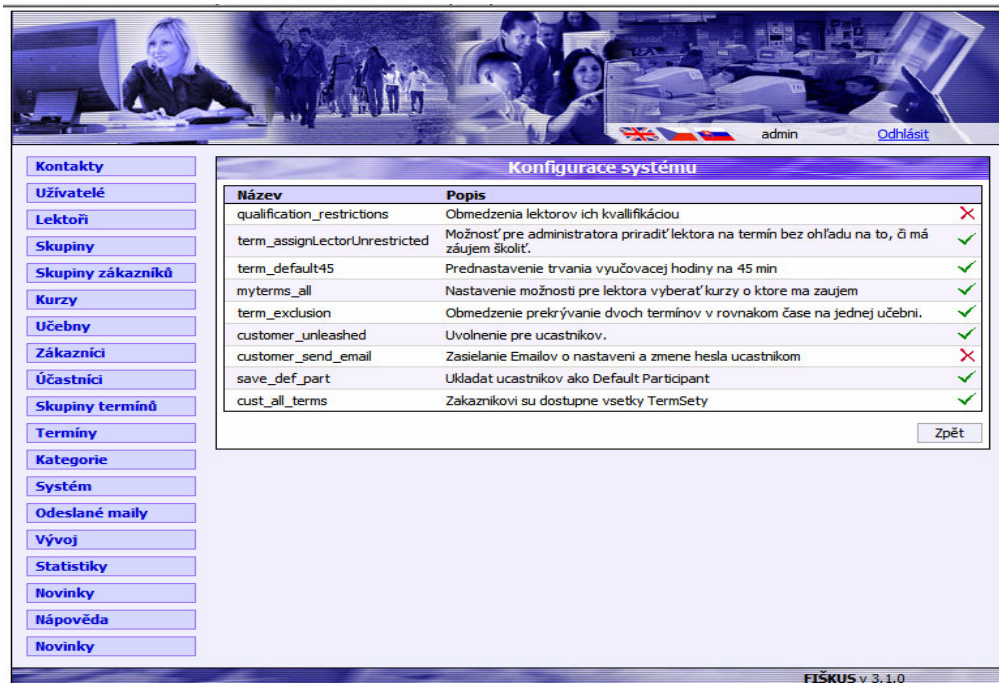
1. Zobrazí sa stránkovaný prehľad užívateľov: *Meno, Login, Blokovanie účtu.*
2. Prehľad je možné radiť podľa *mena* aj *loginu*.
3. Podľa príslušných práv prihláseného užívateľa sú zobrazené tlačítka odkazujúce sa na odstránenie, editáciu a pridanie nového užívateľa.
4. Koniec *Prípadu použitia*.

Alternatívne cesty:

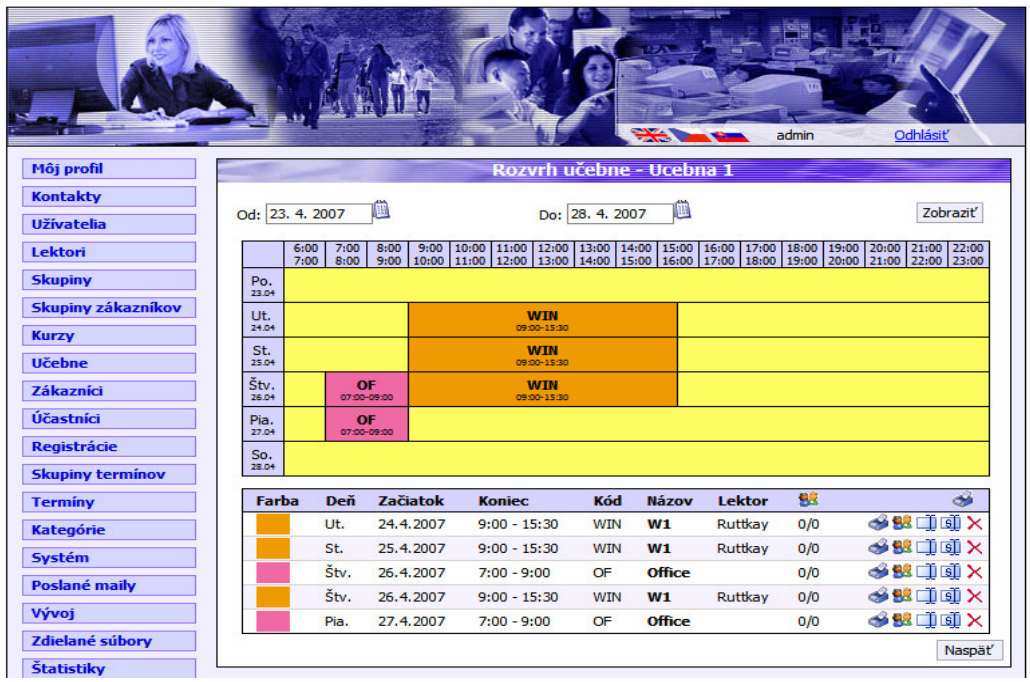
Poznámky:

Príloha 3. Ilustračné obrazovky systému

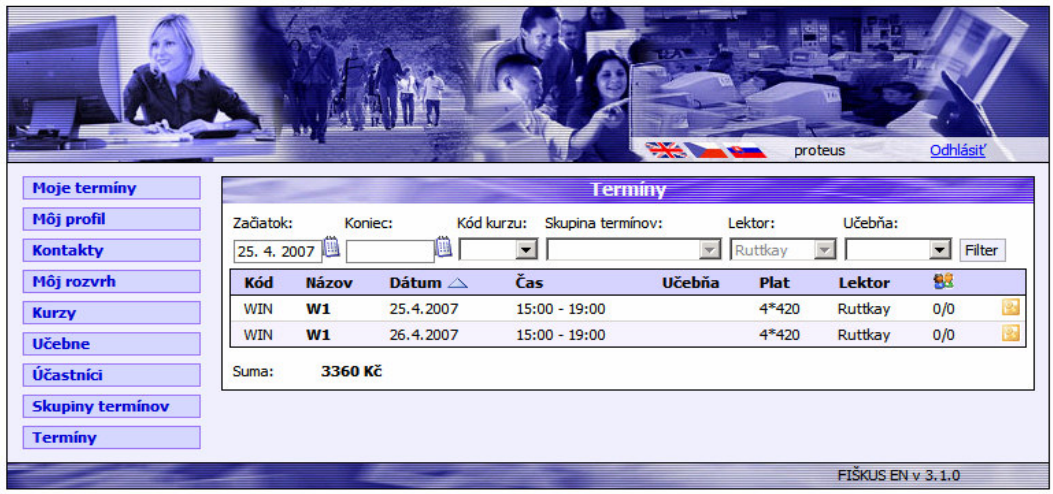
Administrátorská časť:



Správa učební:



Lektorská časť:



Zákaznícka časť: