FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
BRATISLAVA

Zdieľanie informácií o osobách v informačnom systéme Univerzity Komenského

Diplomová práca

Ján Terkanič **2006**

Zdieľanie informácií o osobách v informačnom systéme Univerzity Komenského

DIPLOMOVÁ PRÁCA

Ján Terkanič

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Katedra informatiky

Informatika

Školiteľ diplomovej práce
Mgr. Pavol Mederly

BRATISLAVA 2006

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry.

V Bratislave 15. 5. 2006 _____

Ján Terkanič

Ďakujem svojmu vedúcemu diplomovej práce, Mgr. Pavlovi Mederlymu, za mnoho cenných rád, podnetov a pripomienok pri písaní tejto práce.

Abstrakt

Predkladaná práca sa zaoberá riešením konkrétneho problému integrácie aplikácií na Univerzite Komenského s cieľom umožniť zdieľanie údajov o osobách medzi zdrojovými a cieľovými systémami a zároveň poskytnúť používateľom na univerzite možnosť tieto údaje prezerať, prípadne s nimi inak pracovať.

Práca obsahuje prehľad vybranej literatúry z oblasti integrácie aplikácií, analýzu určených systémov so zameraním na údaje v nich uložené, detailnú analýzu rôznych možností prepojenia týchto systémov a podrobný návrh realizácie vybranej možnosti spočívajúcej vo vytvorení centrálnej databázy osôb (CDO).

Pri práci sme vychádzali zo skúseností s návrhom a implementáciou aplikácií a tiež zo skúseností s prevádzkou a vývojom prototypu CDO. Cenným zdrojom informácií bola aj literatúra zaoberajúca sa popisom rôznych návrhových vzorov aplikovateľných pri tvorbe podnikových systémov. Pri návrhu CDO sme vychádzali z reálnych požiadaviek, či už používateľských alebo iných, vyplývajúcich z potrieb a možností jednotlivých integrovaných systémov tvoriacich IS univerzity.

Hlavným prínosom práce je ukážka riešenia reálneho problému integrácie aplikácií prechádzajúceho všetkými fázami vývoja softvéru až po fragment implementácie, pričom v každej fáze sú zvažované rôzne alternatívy a vybrané tie najlepšie vzhľadom k definovaným kritériám.

Kľúčové slová: integrácia aplikácií, zdieľanie údajov, centrálna databáza osôb.

Obsah

Úvod

Cieľom práce je vykonať analýzu potrieb a možností integrácie aplikácií na Univerzite Komenského v oblasti výmeny informácií o osobách a následne projekt integrácie čiastočne realizovať.

Uvedený cieľ práce je pomerne všeobecný a nehovorí príliš veľa o tom, aký konkrétny problém sa snažíme vyriešiť a čo budeme považovať za uspokojivý výsledok. V tejto úvodnej kapitole sa preto budeme venovať presnejšej formulácii zadania problému a stručne popíšeme súčasný stav, v ktorom sa informačný systém univerzity nachádza.

Pod pojmom „informačný systém organizácie" budeme v práci rozumieť súbor technických (hardvérových) a programových (softvérových) prostriedkov používaných na získavanie údajov potrebných na zabezpečenie čo najefektívnejšieho chodu organizácie a jej prezentáciu navonok, ich uchovávanie, spracovávanie, distribúciu a sprístupňovanie rôznym skupinám používateľov ([19]).

Pod pojmom „integrácia aplikácií" budeme v práci rozumieť prepájanie aplikácií za účelom výmeny informácií a vzájomného poskytovania vykonávaných funkcií. Presnejšie definícia tohto pojmu je prezentovaná v kapitole 1.1.

Kontext

Univerzita je veľká inštitúcia. Ako každá iná veľká organizácia potrebuje pre svoje správne a efektívne fungovanie počítačové aplikácie, ktoré podporujú a uľahčujú vykonávanie aktivít súvisiacich s jej činnosťou.

Postupom času bolo na univerzite uvedených do prevádzky niekoľko aplikácií, z ktorých každá podporuje činnosť univerzity v určitej oblasti. Ide o evidenciu študentov, evidenciu zamestnancov, knižničný systém a ďalšie systémy. Niektoré z týchto systémov vznikli ako výsledok vlastného vývoja na univerzite, iné boli zakúpené od externého dodávateľa. Možno povedať, že väčšina týchto aplikácií dostatočne dobre pokrýva špecifickú oblasť, na ktorú sú určené.

Každá veľká organizácia od istého času začne pociťovať potrebu vzájomnej spolupráce jednotlivých subsystémov[1] v rámci svojho informačného systému (ďalej aj IS). V tejto situácii organizácia zistí, že rozvoj a zdokonalenie služieb informačného systému sa už nedá riešiť jednoducho zakúpením resp. vyvinutím ďalšieho softvéru. Vo svetle nových „integračných" požiadaviek sa treba zamyslieť nad fungovaním súčasného komplexného IS a určiť spôsob ako „prepojiť" existujúce systémy tak, aby dokázali koordinovane spolupracovať a poskytovať vyššiu kvalitu služieb pre používateľov.

Problém

Úlohou je teda prepojiť určené aplikácie informačného systému UK. Integrácia aplikácií na UK je v súčasnej dobe v podstate len v začiatkoch. Momentálne hlavným zámerom pri prepájaní aplikácií je preto zabezpečiť efektívne zdieľanie resp. poskytovanie informácií, a to najmä údajov o osobách na univerzite.

Okrem tejto základnej požiadavky je cieľom aj splnenie používateľských požiadaviek, ktoré vychádzajú zo zámeru sprístupniť údaje uchovávané v IS UK používateľom a umožniť prácu s týmito údajmi. Ide hlavne o možnosť prezerať údaje o osobách tak, ako sú uložené v IS a prípadne hodnotu vybraných údajov (napr.

[1] pod subsystémom rozumieme aplikáciu, ktorá je súčasťou celkového informačného systému organizácie

kontaktných) upraviť. Ďalšie používateľské funkcie súvisia so správou univerzitných preukazov a s možnosťou zaregistrovať žiadosť o notifikáciu pri zmene nejakých údajov.

Zoznam atribútov, ktoré je možné zdieľať medzi systémami, vznikne analýzou možností týchto systémov. V súčasnosti nám ide najmä o zdieľanie nasledujúcich skupín atribútov:

- osobné údaje ako meno, priezvisko, tituly pred menom či za menom, rodné číslo, dátum narodenia, miesto narodenia,
- informácie o trvalom či prechodnom bydlisku,
- informácie o vzťahoch osoby k univerzite, či už študentských alebo zamestnaneckých,
- údaje o univerzitnom preukaze,
- informácie o ubytovaní osoby na internáte.

Ak by sme to chceli zhrnúť, „prepojiť určené aplikácie na UK“ znamená zabezpečiť efektívnu distribúciu údajov medzi jednotlivými systémami univerzitného IS podľa požiadaviek týchto systémov a v závislosti na podnikových procesoch v rámci univerzity. Toto umožní vzájomnú spoluprácu týchto systémov, čím sa skvalitnia a urýchlia služby pre používateľov.

Prínos práce

Hlavným prínosom tejto práce je ukážka riešenia reálneho problému integrácie aplikácií. Predpokladá sa, že výsledky tejto práce budú základom systému použitého v ostrej prevádzke. Riešenie prezentované v práci prechádza všetkými fázami vývoja softvéru – od špecifikácie požiadaviek až po fragment implementácie. V každom kroku sú zvažované viaceré alternatívy a vybrané tie najvhodnejšie vzhľadom k formulovaným kritériám. Veľkú časť práce tvorí analýza problému a rozdiskutovanie rôznych možností jeho riešenia.

Konkrétnych prínosov navrhnutého riešenia integrácie na UK je mnoho a vyplývajú z potreby automatizácie vykonávania procesov týkajúcich sa osôb na univerzite. Ide o procesy presahujúce rámec jedného subsystému IS. Dobrým príkladom je prijatie nového študenta. Ak po zápise príde napríklad do knižnice, knižničný systém by už mal mať všetky relevantné údaje o danom študentovi k dispozícii. Tu sa ukazuje jeden z veľkých problémov, ktorý integrácia dokáže riešiť, a síce viacnásobné zadávanie tých istých údajov do systému. Pri tomto zadávaní totiž spravidla vzniká relatívne veľké množstvo chýb, ktoré môžu neskôr spôsobiť problémy.

Obsah práce

Práca pozostáva zo siedmich kapitol a dvoch príloh. Stručný popis obsahu jednotlivých kapitol je nasledovný:

- Kapitolu 1 tvorí prehľad vybranej literatúry zaoberajúcej sa problematikou integrácie aplikácií.
- V kapitole 2 je prezentovaná analýza systémov, ktorých prepojenie je cieľom tejto práce. Zameriavame sa hlavne na popis integračných zámerov a údajov poskytovaných resp. požadovaných týmito systémami s ohľadom na podnikové procesy, ktoré príslušný systém podporuje resp. ktorými je daný systém ovplyvnený.

- V kapitole 3 prezentujeme možnosti riešenia prepojenia systémov analyzovaných v predchádzajúcej kapitole, pričom berieme do úvahy aj používateľské požiadavky na prácu s údajmi o osobách.
- Kapitola 4 je venovaná formulácii požiadaviek na centrálnu databázu osôb – používateľských a tiež ďalších funkčných a nie funkčných požiadaviek zásadne ovplyvňujúcich návrh CDO.
- Kapitola 5 obsahuje návrh CDO, v ktorom sa snažíme plniť formulované požiadavky. Kapitola zahŕňa riešenie flexibility a bezpečnosti, návrh architektúry CDO a návrh hlavnej časti databázového modelu. Súčasťou kapitoly je popis technológií použitých pri implementácii.
- V kapitole 6 sa snažíme porovnať vývoj „integračnej aplikácie“ akou je CDO s vývojom klasickej podnikovej aplikácie určenej na riešenie jednej oblasti fungovanie organizácie.
- Kapitola 7 predstavuje zhrnutie celej práce v podobe záveru.
- Prílohy práce tvoria dve časti – špecifikácia CDO a popis rozhraní medzi CDO a prepájanými systémami. Popis rozhraní sa zaoberá aj technickými otázkami ako napr. frekvencia, spôsob prenosu a formát údajov.

1 Literárny prehľad

Oblasť integrácie aplikácií zaznamenala v posledných rokoch dynamický rozvoj, ktorý pretrváva dodnes a predpokladá sa, že bude pokračovať aj v budúcnosti. Hlavnou motiváciou rozvoja v tejto oblasti je reálna potreba organizácií, aby jednotlivé systémy dokázali zdieľať informácie a spoločné procesy. V prípade správne a vhodne riešenej integrácie totiž organizácia získa výhodu vo zvýšenej kvalite služieb, ktoré poskytuje.

Vzhľadom na šírku záberu témy nebolo možné vypracovať obsiahly prehľad literatúry z oblasti integrácie. Rozsahom by sa pravdepodobne vyrovnal samostatnej prehľadovej práci. Rozhodli sme sa preto vybrať len niektoré zdroje, o ktorých si myslíme, že sú najpodstatnejšie a zároveň najreprezentatívnejšie.

1.1 Úvod do integrácie aplikácií

Táto kapitola je motivovaná informáciami uvedenými v [1] a [2].

1.1.1 Všeobecne o integrácii

V [1] je integrácia aplikácií definovaná ako strategický prístup k prepojeniu informačných systémov, tak na informačnej úrovni ako aj na úrovni služieb. Týmto prepojením sa podporí schopnosť aplikácií vymieňať si informácie a využívať procesy v reálnom čase. Výhoda pre organizáciu je očividná. Automatizáciou procesov sa zabezpečí urýchlenie ich vykonávania a zníženie prácnosti.

Integrácia aplikácií môže mať mnoho foriem, z ktorých spomenieme dve najznámejšie. Interná integrácia aplikácií sa nazýva „enterprise application integration“ (EAI), externá forma sa označuje ako „business to business application integration“ (B2B). Aj keď každá z týchto foriem má svoje špecifiká, možno medzi nimi nájsť veľa spoločných znakov a vzorov. Napríklad, vždy musí obsahovať technológiu na transformáciu údajov, na smerovanie informácií správnym konzumentom a tiež spracovanie pravidiel definujúcich integračné väzby.

Jednoducho povedané, integrácia aplikácií je komplexný problém. Rozložením na menšie komponenty sa samozrejme riešenie problému stáva menej náročným. Dôležité však je porozumieť najprv potrebám, potom požiadavkám a nakoniec tomu, ako vyriešiť problém pre danú doménu.

1.1.2 Potreba integrácie

Veľké organizácie v sebe typicky zahŕňajú stovky aplikácií, ktoré sú vyvinuté vlastnými silami, získané od externých dodávateľov alebo sú časťami zastaraných systémov. Tieto aplikácie operujú na rôznych vrstvách a rôznych platformách. Prirodzená je teda otázka: Ako si organizácia mohla dovoliť dostať svoj informačný systém do takého neprehľadného stavu? Kto je zodpovedný za vytvorenie takejto „špagetovej“ architektúry?

Po prvé, písanie podnikových aplikácií (business applications) je ťažké. Vytvoriť jedinú veľkú aplikáciu na podporu všetkých procesov v rámci organizácie je takmer nemožné, keď sa zamyslíme nad počtom jednotlivých požiadaviek veľkej organizácie.

Po druhé, rozdelenie podnikových funkcií medzi viacero systémov dáva organizácii výhodu v možnosti vybrať si ten „najlepší“ softvér pre každú konkrétnu oblasť jej fungovania. Obvykle nie je záujem o jeden veľký systém, ktorý dokáže podporovať všetky funkcie v organizácii.

Na podporu podnikových procesov a zdieľania údajov medzi aplikáciami musia byť tieto aplikácie integrované.

1.1.3 Výzvy integrácie

Integrácia aplikácií nie je ľahká úloha. Máme veľa systémov, na rôznych platformách, rozdistribuovaných na rôznych miestach. Aj keď sú k dispozícii EAI balíky (systémy na riešenie problému EAI), ktoré zabezpečujú integráciu nezávislú na platforme či programovacom jazyku, táto technická infraštruktúra rieši len malú časť tej zložitosti, ktorú integrácia prináša.

Integrácia vyžaduje značný posun vo firemnej politike. Úspešná integrácia musí zabezpečiť komunikáciu nie len medzi rôznymi systémami, ale aj medzi oddeleniami (či divíziami) v rámci organizácie. V integrovanom systéme už jednotlivé oddelenia nemajú takú kontrolu nad systémami, ktoré prevádzkujú. Tieto systémy sa totiž stali časťou celkového toku integrovaných aplikácií a služieb. Integračné riešenie sa od okamihu, kedy začne podporovať základné procesy v organizácii, stáva kritickým elementom pre fungovanie celej organizácie.

Dôležitým obmedzením pri vytváraní integračného riešenia je minimálna kontrola, ktorú majú vývojári nad jednotlivými integrovanými systémami. Vo väčšine prípadov ide o zastarané alebo komerčné aplikácie. Robiť zmeny vo vnútri týchto systémov za účelom prepojenia s inými aplikáciami, je spravidla ťažko (ak vôbec) realizovateľné.

Napriek značnému rozšíreniu integrácie v praxi, existuje v tejto doméne veľmi málo štandardov. Najväčší úspech štandardizácie sme zaznamenali v oblasti XML, XSL a webových služieb. Existujúce štandardy webových služieb pokrývajú len zlomok toho, čo by malo byť štandardizované.

Vývoj riešenia EAI je sám o sebe náročnou úlohou, prevádzka a údržba takéhoto riešenia môže byť náročná ešte viac. Množstvo rôznych technológií a distribuovaná podstata riešenia EAI robí z nasadenia, monitorovania a riešenia problémov komplexné úlohy, ktoré si často vyžadujú od človeka kombináciu viacerých schopností.

Medzi fundamentálne technické problémy, ktoré musí každé integračné riešenie nejakým spôsobom zvládnuť, zaraďujú autori v [2] nasledujúce:

- Siete sú nespoľahlivé. Pri integrácii sa presúvajú údaje z jedného počítača do iného cez sieť. V porovnaní s procesom bežiacim v jednom počítači musíme pri tejto distribuovanosti počítať s viacerými problémami. A to najmä pri geograficky vzdialených systémoch, keď sú údaje prenášané cez mnohé linky, aktívne sieťové prvky, segmenty siete LAN a podobne.
- Siete sú pomalé. Presun údajov po sieti je omnoho pomalší ako v operačnej pamäti počítača pri lokálnom volaní nejakej funkcie v aplikácii. Toto môže mať dopad na výkonnosť celého riešenia, keď berieme do úvahy širokú distribuovanosť.
- Ľubovoľné dve aplikácie sú rozdielne. Integračné riešenie prenáša informácie medzi systémami, ktoré používajú rôzne programovacie jazyky, platformy či formáty údajov.
- Zmena je nevyhnutná. Aplikácie sa postupom času menia a prispôsobujú novým požiadavkám. Integračné riešenie musí vedieť flexibilne reagovať na zmeny v aplikáciách a taktiež minimalizovať dopad zmien v jednej aplikácii na tie ostatné.

1.2 Prístupy k integrácii

Existuje viacero prístupov k integrácii, od najjednoduchších až po tie zložité. Každý prístup má svoje pozitíva či negatíva a o žiadnom z nich sa nedá povedať, že je univerzálne najlepší alebo najhorší. V každom konkrétnom prípade riešenia integrácie je potrebné zvažovať všetky alternatívy a zvoliť tú najvhodnejšiu. To nemusí byť nutne tá najzložitejšia – sú situácie, kedy je jednoduché riešenie integrácie úplne postačujúce. Dodajme, že vždy je dobré pri návrhu riešenia myslieť na rozširovanie sa problémovej domény v budúcnosti.

Linthicum vo svojej knižke ([1]) rozlišuje nasledovné prístupy k integrácii a zároveň poznamenáva, že pri riešení konkrétnej integrácie často kombinujeme rôzne prístupy.

1.2.1 Information-oriented application integration (IOAI)

Prvým a v mnohých prípadoch postačujúcim prístupom je integrácia na úrovni údajov. Tento prístup rieši problém integrácie pomocou výmeny údajov medzi jednotlivými databázami (alebo aplikačnými rozhraniami, ktoré produkujú údaje). Nezaoberáme sa teda žiadnymi procesmi ani aplikačnými službami.

Tento prístup ponúka isté výhody:

- Predovšetkým, zaoberáme sa tu jednoduchým prenosom informácií, takže zvyčajne nie je nutné meniť zdrojové ani cieľové systémy. Väčšina systémov, najmä relačné databázové systémy, dokáže produkovať a konzumovať jednoduchú informáciu.
- Nepotrebujeme sa zaoberať zložitosťou riadenia interakcie medzi systémami.
- Tento prístup je ľahko pochopiteľný a široko použiteľný.

Najväčšou výhodou tohto riešenia je teda minimálny, resp. niekedy žiadny, dopad na integrované aplikácie. Medzi ďalšie vlastnosti by sme mohli zaradiť aj jednoduchosť a nízke náklady na implementáciu. Realizácia pozostáva z kopírovania údajov, ktoré môže byť podľa potreby doplnené o transformácie údajov. Tieto sú vo väčšine prípadov nutné.

Pri tomto riešení potrebujeme detailne poznať všetky integrované aplikácie a to najmä z hľadiska vnútornej štruktúry údajov a toho, kedy a ako sa k nim dá pristupovať. Aplikačná sémantika robí tento problém ešte horším, keďže systémy spravidla nemajú tieto sémantiky kompatibilné. Preto integrácia na úrovni údajov neznamená len presúvanie údajov medzi databázami, ale aj zvládnutie rozdielov v schéme a obsahu týchto databáz.

Linthicum rozdeľuje systémy podľa spôsobu prístupu k údajom do týchto kategórií.

- relačná databáza prístupná cez SQL,
- aplikácia prístupná cez svoje aplikačné rozhranie (API),
- používateľské rozhranie automaticky spracovateľné technikou zosnímania obrazovky („screen scraping“),
- vstavané („embedded“) zariadenie.

V prípade aplikovania tohto prístupu k integrácii odporúča Linthicum vykonať nasledovné kroky.

- Identifikácia údajov.

V tomto kroku ide o zostavenie zoznamu kandidátskych systémov a zistenie informácií o ich vnútorných databázach – typy databáz, ich fyzická a logická štruktúra, schéma a tiež spôsob, akým sú informácie použité v kontexte relevantnej aplikácie.

- Katalogizácia údajov.

  Tu ide o proces získavania metadát a rôznych iných dát z danej problémovej domény. Výsledkom tohto kroku by mal byť čo možno najdetailnejší dátový slovník. Taktiež vznikne logický model vo forme entitno-relačného diagramu.

- Vytvorenie metadátového modelu celej organizácie.

  Tento model definuje nie len existujúce dátové štruktúry v organizácii, ale aj to, ako budú tieto štruktúry interagovať v rámci riešenia integrácie.

Náš postup riešenia integrácie na UK sa s týmto postupom prekrýva len v prvom kroku. Vytvorenie dátového slovníka by bolo užitočné, ale rozhodli sme sa viac zamerať na technický návrh riešenia než na rozsiahlu katalogizáciu údajov.

1.2.2 Business process integration-oriented application integration (BPIOAI)

Tento prístup je podľa [1] budúcnosťou integrácie aplikácií. V dnešnej dobe už možno potvrdiť, že vývoj v oblasti integrácie sa naozaj uberá týmto smerom. BPIOAI zavádza pojem spoločného modelu podnikových procesov (common business process model), ktorý riadi presun informácií a spúšťanie aplikačných služieb v rámci rôznych systémov.

Princíp je v podstate jednoduchý. Umiestnime vrstvu riadiacej logiky nad nejakú integračnú technológiu, ktorá tejto riadiacej logike umožní zviazať systémy do jednotného viackrokového podnikového procesu. Vznikne teda procesný model, ktorý prakticky riadi integráciu.

BPIOAI je v skutočnosti ďalšia vrstva nad ostatnými, tradičnejšími prístupmi k integrácii, vrátane IOAI či SOAI. Keď sa na to pozrieme z pohľadu vrstiev, nájdeme v rámci BPIOAI tri principiálne vrstvy:

- procesný model,
- transformácie, smerovanie a pravidlá,
- systém na doručovanie správ.

BPIOAI je tak stratégia popisujúca prístup k integrácii ako aj technológia, ktorá túto integráciu umožňuje. Z toho technologického hľadiska pozostáva tento prístup z nasledujúcich komponentov:

- grafický modelovací nástroj, v ktorom sa vytvára procesný model a definuje sa správanie,
- koordinátor podnikových procesov („business process engine“), ktorý riadi vykonávanie podnikových procesov, zachováva stav a zabezpečuje interakcie s middlewareom,
- rozhranie na monitorovanie podnikových procesov („business process monitoring interface“), ktoré umožňuje monitorovať vykonávanie procesov v reálnom čase,

- rozhranie koordinátora podnikových procesov („business process engine interface“), ktoré umožňuje ostatným aplikáciám prístup k nemu,
- integračná technológia („middleware“, ako napríklad integračný alebo aplikačný server), ktorá spája zdrojové a cieľové systémy.

1.2.3 Service-oriented application integration (SOAI)

Tento prístup nie je ničím novým. SOAI umožňuje organizáciám zdieľať spoločné aplikačné služby ako aj informácie. Toto zdieľanie možno docieliť definovaním aplikačných služieb, ktoré chceme zdieľať a vytvorením potrebnej infraštruktúry.

Z funkcií, ktoré poskytujú jednotlivé systémy, spravíme zdieľané služby. Tieto môže využívať ktokoľvek. Základnou črtou tohto riešenia je teda znovupoužiteľnosť. Príchod webových služieb (presnejšie povedané, štandardov spojených s webovými službami) umožnil lepšie použitie tohto prístupu k integrácii, keďže sú k dispozícii štandardy na popisovanie rozhraní služieb a na ich automatické vyhľadávanie.

Na využitie tejto paradigmy je takmer vždy nutný zásah do integrovaných systémov, čo logicky znamená oveľa väčšie náklady na implementáciu. Na druhej strane, skúsenosti ukazujú, že z dlhodobého hľadiska sú tieto investície opodstatnené a SOAI je prínosom pre organizáciu vzhľadom na širšie možnosti tohto riešenia.

Bližšiu charakteristiku SOA (service oriented architecture) uvádzame v podkapitole 1.5.1.

1.2.4 Portal-oriented application integration (POAI)

Portálové riešenie nám umožňuje pozerať sa na množinu systémov – interných aj externých – cez jednotné používateľské rozhranie. Pri tomto prístupe sa vyhneme „back-end“ integrácií systémov tým, že rozšírime používateľské rozhranie každého systému na jedno spoločné (agregované) používateľské rozhranie – čo najčastejšie býva webové rozhranie prístupné cez prehliadač.

POAI sa principiálne líši od predchádzajúcich prístupov, ktoré sme spomenuli. Tento prístup sa zaoberá sprístupnením informácií z mnohých systémov cez jedinú aplikáciu a rozhranie. Ostatné doteraz menované prístupy predstavovali udalostne riadenú (event-driven oriented) integráciu v reálnom čase.

V mnohých problémových doménach pri integrácii uprednostňujú používatelia interakciu so systémami cez používateľské rozhranie namiesto toho, aby si systémy sami v pozadí automaticky vymieňali informácie. Použitie portálov na integráciu má veľa výhod:

- Ide o neinvazívny prístup, keďže v skutočnosti netreba integrovať jednotlivé systémy, čo eliminuje riziko a znižuje celkovú cenu riešenia.
- Implementácia tohto riešenia je typicky rýchlejšia než zaistenie výmeny informácií medzi systémami v reálnom čase.
- Existuje vyspelá technológia, ktorá umožňuje realizáciu tohto prístupu.

Na druhej strane nevýhody tohto prístupu sú predovšetkým tieto:

- Vyžadovaná je ľudská interakcia, pretože údaje netečú v reálnom čase. V dôsledku toho, systémy nereagujú automaticky na podnikové udalosti v rámci organizácie (business events).

- Informácie musia byť vytiahnuté z jednotlivých systémov, typicky cez ďalšiu vrstvu aplikačnej logiky. Tým sa v niektorých prípadoch riešenie vlastne skomplikuje.
- Z hľadiska ochrany údajov je tu potenciálne riziko, keď sú údaje z vnútra organizácie poskytované používateľom cez web.

V našom prípade by sa uvedené negatíva mohli prejaviť pri používateľských funkciách – najmä pri prezeraní údajov cez webové používateľské rozhranie. Pri integrácií riešenej pomocou centrálnej databázy osôb sú eliminované prvé dva uvedené nedostatky, keďže informácie sú sústredené v jednom systéme a z neho získavané. Čo sa týka ochrany údajov, môžeme ju zabezpečiť použitím šifrovaných protokolov na prenos.

1.3 Vzory v oblasti EAI

O pokroku a rozvoji v oblasti integrácie aplikácií svedčí aj fakt, že existuje množina návrhových vzorov (design patterns) pre EAI. Z množstva riešených integračných projektov bolo teda možné vyabstrahovať základné princípy a elementy. Tieto boli sformulované do všeobecnejšie použiteľných vzorov.

Veľmi dobrou a komplexnou publikáciou o vzoroch v oblasti EAI je [2]. V tejto knihe popisujú autori veľké množstvo vzorov, od najzákladnejších až po tie zložitejšie. Jednotlivé vzory rozdelili do prehľadných kategórií podľa toho, aký problém sa snažia riešiť. Celá kniha sa primárne zaoberá integráciou pomocou posielania správ (enterprise integration using messaging), teda aj prezentované vzory sú aplikovateľné najmä ak integrujeme systémy posielaním správ.

Autori sa zameriavajú na pochopenie hlavných konceptov riešených pri integrácii pomocou posielania správ:

- výhody a obmedzenia integrácie prostredníctvom posielania správ v porovnaní s ostatnými integračnými technikami,
- ako určiť kanály pre správy (message channels), ktoré budú aplikácie potrebovať, ako sa vysporiadať s chybnými správami,
- kedy posielať správy, čo by mali obsahovať, ako použiť špeciálne atribúty v správe,
- ako smerovať správu k jej koncovému cieľu, keď odosielateľ nevie, kde sa tento cieľ nachádza,
- ako konvertovať správy, keď sa odosielateľ a príjemca nedohodli na spoločnom formáte,
- ako navrhovať kód, ktorý pripája aplikáciu k systému na doručovanie správ,
- ako spravovať a monitorovať celý systém na doručovanie správ, keď sa už stane súčasťou IS organizácie.

V [2] rozdeľujú autori návrhové vzory do skupín podľa toho, v ktorej oblasti riešenia integrácie posielaním správ sú aplikovateľné. Veľkým prínosom je aj navrhnutá grafická notácia pre jednotlivé vzory, čo následne umožňuje kresliť prehľadné diagramy celého riešenia s použitím týchto vzorov.

Tabuľka 1-1 Integračné návrhové vzory

Skupina	Vzor	Riešený problém
Messaging systems	Message channel	Ako aplikácie komunikujú pomocou posielania správ?
	Message	Ako si môžu dve aplikácie spojené komunikačným kanálom vymeniť informáciu?
	Pipes and filters	Ako môžeme spracovať správu tak, aby sme zachovali nezávislosť jednotlivých krokov spracovania a flexibilitu?
	Message router	Ako môžeme oddeliť jednotlivé výpočtové kroky tak, že správy môžu byť poslané cez rôzne filtre v závislosti na nejakej množine podmienok?
	Message translator	Ako môžu spolu komunikovať systémy používajúce rôzne formáty údajov?
	Message endpoint	Ako sa aplikácia pripojí ku komunikačnému kanálu, aby mohla prijímať a posielať správy?
Messaging channels	Point-to-point channel	Ako sa odosielateľ správy môžu uistiť, že ju prijme práve jeden príjemca?
	Publish-subscribe channel	Ako môže odosielateľ poslať správu všetkým príjemcom, ktorí o ňu majú záujem?
	Datatype channel	Ako môže aplikácia poslať údaj tak, aby príjemca vedel ako ho má spracovať?
	Invalid message channel	Ako môže príjemca korektne spracovať prijatie chybnej správy?
	Dead letter channel	Čo spraví systém na doručovanie správ so správou, ktorú nemôže doručiť?
	Guaranteed delivery	Ako sa odosielateľ môže uistiť, že správa bude doručená aj keď zlyhá systém na doručovanie správ?
	Channel adapter	Ako sa môže aplikácia pripojiť k systému na doručovanie správ za účelom prijímania a posielania správ?
	Messaging bridge	Ako môžeme prepojiť viacero systémov na doručovanie správ?
	Message bus	Aká architektúra umožní oddeleným aplikáciám spolupracovať tak, aby neboli navzájom zviazané a mohli byť ľahko pridávané a odoberané?
Message construction	Command message	Ako môžeme pomocou posielania správ vyvolať procedúru v inej aplikácii?
	Document message	Ako môžeme pomocou posielania správ prenášať dáta medzi aplikáciami?
	Event message	Ako môžeme pomocou posielania správ prenášať udalosti z jednej aplikácie do druhej?
	Request-reply	Keď aplikácia pošle správu, ako môže dostať odpoveď od jej príjemcu?
	Return address	Ako môže príjemca správy vedieť, kam má poslať odpoveď?
	Correlation identifier	Ako môže aplikácia, ktorá dostala odpoveď, identifikovať zodpovedajúcu požiadavku?

	Message sequence	Ako môže aplikácia pomocou správ poslať ľubovoľne veľké množstvo údajov?
	Message expiration	Ako môže odosielateľ správy indikovať, kedy má byť správa považovaná za starú a teda nemá byť spracovaná?
	Format indicator	Ako môže byť formát správy navrhnutý tak, aby umožňoval budúce zmeny formátu?
Message routing	Content-based router	Ako môžeme zvládnuť situáciu, keď je jedna logická funkcia implementovaná viacerými fyzickými systémami?
	Message filter	Ako sa komponent môže vyhnúť prijímaniu pre neho nezaujímavých správ?
	Dynamic router	Ako sa vyhnúť závislosti smerovača správ na všetkých možných cieľoch so zachovaním jeho efektívnosti?
	Recipient list	Ako môžeme doručiť správu dynamickému zoznamu príjemcov?
	Splitter	Ako môžeme spracovať správu obsahujúcu viac častí, z ktorých každá má byť spracovaná rozličným spôsobom?
	Aggregator	Ako môžeme skombinovať výsledky individuálnych ale súvisiacich správ tak, aby mohli byť spracované ako celok?
	Resequencer	Ako môžeme zoradiť sériu súvisiacich správ do správneho poradia?
	Composed message processor	Ako môžeme zachovať celkový tok správ pri spracovaní správ pozostávajúcej z viacerých elementov, z ktorých každý môže vyžadovať rozdielne spracovanie?
	Scatter-gather	Ako môžeme zachovať celkový tok správ keď správa musí byť poslaná viacerým príjemcom, z ktorých každý môže poslať odpoveď?
	Routing slip	Ako môžeme smerovať správu cez postupnosť krokov, ktorá nie je známa v čase návrhu a môže sa líšiť pre každú správu?
	Process manager	Ako môžeme smerovať správu cez viacero krokov, keď tieto kroky nie sú známe v čase návrhu a nemusia byť sekvenčné?
	Message broker	Ako oddeliť cieľ správy od odosielateľa a zachovať centrálnu kontrolu nad tokom správ?
Message transformation	Envelope wrapper	Ako sa môžu existujúce systémy zapojiť do výmeny správ, ktorá kladie špeciálne požiadavky na formát sprav?
	Content enricher	Ako môže aplikácia komunikovať s iným systémom, keď k dispozícií všetky požadované údaje?
	Content filter	Ako zjednodušiť spracovanie veľkej správy, keď nás zaujíma len niekoľko údajov?
	Claim check	Ako zredukovať množstvo údajov v správe tak, aby sa tieto údaje nestratili?

	Normalizer	Ako spracovať správy, ktorá sú sémanticky ekvivalentné, ale prichádzajú v rôznych formátoch?
	Canonical data model	Ako môžeme minimalizovať závislosti pri integrácii aplikácií, ktoré používajú rôzne formáty údajov?
Messaging endpoints	Messaging gateway	Ako oddeliť prístup k systému na doručovanie správ od zvyšku aplikácie?
	Messaging mapper	Ako presúvať údaje medzi doménovými objektmi a správami tak, aby tieto dve reprezentácie údajov zostali nezávislé?
	Transactional client	Ako môže klient kontrolovať svoje transakcie so systémom na doručovanie správ?
	Polling consumer	Ako môže aplikácia prijať správu vtedy, keď je pripravená ju spracovať?
	Event-driven consumer	Ako môže aplikácia prijímať správy hneď ako sú dostupné?
	Competing consumers	Ako môže klient spracovať viacero správ súčasne?
	Message dispatcher	Ako môžu viacerí príjemcovia na jednom kanáli koordinovať svoje spracovanie správ?
	Selective consumer	Ako si môže príjemca správ vybrať, ktoré správy chce prijímať?
	Durable subscriber	Ako sa príjemca (na kanáli typu „publish-subcribe“) môže vyhnúť strate správ v čase, keď na tieto správy nečaká?
	Idempotent receiver	Ako sa môže príjemca vysporiadať s duplicitnými správami?
	Service activator	Ako môže aplikácia poskytovať službu prístupnú pomocou posielania správ aj prostredníctvom iných prístupových techník?
System management	Control bus	Ako efektívne spravovať systém na doručovanie správ distribuovaný na viacerých platformách a širokej geografickej oblasti?
	Detour	Ako smerovať správy cez dodatočné kroky na vykonanie validácie, testovania a debugovania?
	Wire tap	Ako môžeme preskúmať správy prenášané cez komunikačný kanál typu „point-to-point“?
	Message history	Ako môžeme efektívne analyzovať a debugovať tok správ vo voľne zviazanom systéme?
	Message store	Ako môžeme sumarizovať informácie v posielaných správach bez narušenia systému na doručovanie správ?
	Smart proxy	Ako môžeme sledovať správy z aplikácie, ktorá publikuje odpovede do kanála určeného žiadateľom?
	Test message	Čo sa stane, ak komponent aktívne spracúva správy, ale na výstupe posiela chybné správy kvôli svojej vnútornej chybe?
	Channel purger	Ako môžeme uchrániť zvyšné správy v kanáli od

		rušivých testov alebo bežiacich systémov?

1.4 Metodiky pre EAI

Pri vyhľadávaní zdrojov o integrácii sme sa stretli s pokusmi popísať odporúčaný postup pri integrácii aplikácií. Ide o metodiky, ktoré podrobne určujú kroky nutné vykonať pri jej riešení. Toto možno považovať za ďalší dôkaz vyspelosti a rozvoja v oblasti EAI.

Bližšie sme sa pri písaní tejto práce oboznámili s iniciatívou OpenEAI. Projekt OpenEAI si kladie za cieľ preskúmať a zdokumentovať princípy a praktiky v oblasti integrácie aplikácií a následne tieto zistenia prezentovať, implementovať a rozširovať.

V rámci metodiky OpenEAI jej autori v [4] navrhujú pri integrácii nasledovnú postupnosť krokov:

- Vykonať analýzu.

  Vo všeobecnosti identifikovať systémy, ktoré majú byť integrované. Keď už existuje zoznam kandidátskych systémov, každý systém zanalyzujeme z pohľadu integrácie. Výsledkom je vyplnená šablóna pre integračnú analýzu („integration analysis template“), ktorá je tiež súčasťou OpenEAI a ktorú si organizácie môžu upraviť podľa vlastných potrieb.

- Zadefinovať správy.

  Zadefinujeme nové podnikové objekty (enterprise objects) a akcie, ktoré vyžadujú. Následne sa zadefinujú správy vo forme XML dokumentov a nové správy sa zaradia do spoločnej hierarchie správ v rámci organizácie.

- Vygenerovať objekty správ a príslušné artefakty.

  Implementovať definície správ ako objekty Java. Tento proces sa dá úplne automatizovať použitím na to určeného generačného nástroja.

- Vyvinúť, zdokumentovať a otestovať komunikujúce aplikácie (messaging applications).

  Detaily tejto fázy sa líšia od organizácie k organizácii. V skratke, toto je tá implementačná fáza, kde hlavnú úlohu hrajú vývojári a testeri. Vznikajú nové aplikácie a komunikačné rozhrania.

- Aktualizovať celopodnikovú dokumentáciu.

  Počas celého integračného procesu vzniká množstvo dokumentácie, definícií správ a iných artefaktov. Cieľom tejto fázy je zahrnúť tieto dokumenty do centrálneho skladu dokumentov organizácie a prípadne vytvoriť webový prístup, aby každý mohol k tejto dokumentácii ľahko pristupovať.

- Nasadenie hotového riešenia.

  V tomto kroku ide o zavedenie implementovaného integračného riešenia do prevádzky. OpenEAI poskytuje aj základné vzory pre nasadenie vyvinutého riešenia (deployment patterns).

- Sledovať vývoj.

  Najlepší spôsob ako sa vyhnúť zlyhaniam je sledovať vývoj a manažovať meniace sa požiadavky a očakávania.

Výhoda tohto štruktúrovaného prístupu je v tom, že proces integrácie je rozdelený na jednotlivé diskrétne úlohy, ktoré sa dajú lepšie riadiť a sledovať ich stav.

Pri uvažovaní nad postupom pri riešení integrácie sme sa čiastočne inšpirovali touto metodikou – najmä pri analýze a definícii tried jazyka Java pre jednotlivé správy. Náš postup sa v podstate zhoduje s touto metodikou, niektoré kroky sme však nerozpracovali úplne do detailu. OpenEAI predstavuje disciplinovaný prístup k integrácii aplikácií a na jeho použitie v praxi je potrebné zabezpečiť primerane veľký vývojový tím s rozdelení zodpovedností medzi jeho členmi. V našom projekte (kde riešitelia sú v podstate dvaja) sme zvolili menej formálny postup aj preto, aby sme sa mohli viac zamerať na samotné riešenie problému a vzniknutých komplikácií a nevenovali veľa času menej prioritným úlohám.

1.5 SOA a ESB

V súčasnosti sa v dostupných materiáloch a publikáciách z oblasti EAI čoraz častejšie objavujú skratky SOA (Service-Oriented Architecture) a ESB (Enterprise Service Bus). O týchto technológiách je k dispozícii veľmi veľa publikovaných článkov a kníh. Ako zdroje informácií pre túto kapitolu sme vybrali niektoré z nich.

1.5.1 Service-Oriented Architecture (SOA)

Relatívne podrobný prehľad internetových zdrojov o SOA sa nachádza v článku [5] Marka Doernhofera, ktorý mapuje rôzne zdroje z webu s cieľom odpovedať na otázku „Čo je to servisne orientovaná architektúra?“. Mnohé zo zdrojov publikovaných o SOA na internete sú dielom komerčných IT spoločností. Niektoré z nich sú napísané so zámerom propagovať príslušný softvérový produkt, ale napriek tomu ich považujeme za cenné zdroje, keďže ich vytvorili profesionáli v oblasti SOA.

Snáď v každej publikácii sa autori snažia zadefinovať SOA resp. podať stručné vysvetlenie toho, čo to vlastne SOA je. Existuje preto množstvo rôznych definícií. Uvedieme teraz niektoré z nich.

IBM uvádza v [6] nasledovnú definíciu. „Architektúra orientovaná na služby (SOA) je komponentový model, ktorý dáva do vzťahu rôzne funkčné moduly aplikácie, nazývané služby, cez dobre definované rozhrania a kontrakty medzi týmito službami. Rozhranie je definované neutrálnym spôsobom, ktorý by mal byť nezávislý na hardvérovej platforme, operačnom systéme a programovacom jazyku, v ktorom je služba implementovaná. To umožňuje službám, vytvoreným na rôznych takýchto systémoch, vzájomnú interakciu uniformným a univerzálnym spôsobom.“

O niečo kratšiu, a podľa nášho názoru aj presnejšiu, definíciu možno nájsť v [7]. „Architektúra orientovaná na služby (SOA) je architektonický štýl, ktorý podporuje voľne zviazané služby, aby umožnil flexibilitu podnikových procesov spôsobom umožňujúcim spoluprácu a nezávislým na technológii. SOA pozostáva z množiny služieb, ktoré umožňujú flexibilnú a dynamicky rekonfigurovateľnú realizáciu podnikových procesov použitím popisov služieb založených na ich rozhraniach.“

S podobnými definíciami by sme mohli pokračovať ďalej. Namiesto toho skúsime vysvetliť základné princípy SOA. Použijeme pri tom informácie hlavne z [8] a [9].

Na úvod treba povedať, že SOA nie je nový koncept. Je to znovuobjavenie myšlienky oddeliť rozhranie od jeho implementácie a definovať softvérovú službu len na základe popisu jej rozhrania. Tento prístup existoval už v technológiách ako J2EE, CORBA či COM. Rozdiel je, že v súčasnosti máme dostupné nové štandardy (ako

napríklad webové služby), ktoré značne uľahčujú implementáciu SOA v praxi. Samotné webové služby na implementáciu SOA síce nepotrebujeme, ale ich použitie je v realite odporúčané. Existujú samozrejme implementácie SOA bez použitia webových služieb.

SOA je architektonický štýl založený na koncepte služieb. Pre pochopenie SOA je teda nutné vedieť, čo sú to služby. Služby, ako základné stavebné bloky SOA, sú chápané ako podnikové služby. Teda napríklad aktualizácia nejakého záznamu v databáze nie je služba. Služby v rámci SOA musia vykazovať nasledovné základné spoločné vlastnosti.

- Služba zapuzdruje opakovane použiteľnú podnikovú funkciu (business function).

  Službou môže byť ľubovoľná podniková funkcia., ale je vhodnejšie, keď je to ozaj znovu použiteľná funkcia. V SOA môže byť služba používaná jedným alebo viacerými systémami, ktoré sú súčasťou architektúry. Služba by mala byť bezstavová a nemala by byť závislá na stave iných funkcií či procesov.

  Služba by mala príslušnú funkciu zapuzdrovať dostatočne dobre na to, aby bolo možné jej znovu použitie. Toto zapuzdrenie funkcie do služby spolu s jej definíciou cez rozhranie umožňuje nahrádzanie jednej implementácie služby inou.

- Služby sú definované explicitnými, implementačne nezávislými rozhraniami.

  Rozhranie by malo popisovať len tie aspekty procesu a správania, ktoré sa používajú pri interakcii poskytovateľa služby s jej konzumentom. Nemalo by teda obsahovať nič, čo by sa týkalo implementácie poskytovateľa či konzumenta služby. Takto sa získa veľká flexibilita pri zmene implementácie alebo identity oboch komunikujúcich systémov.

- Služby sú spúšťané cez komunikačné protokoly, ktoré zabezpečujú interoperabilitu a vzájomnú nezávislosť na umiestnení jednotlivých služieb.

  Pri rozhodovaní ako spojiť aplikácie máme niekoľko možností. Príkladmi protokolov resp. rozhraní, ktoré by sme mohli použiť, sú HTTP/S, JMS, CORBA alebo SMTP. Typicky, dokonca aj v rámci jednej organizácie, je použitých niekoľko rôznych techník, produktov a protokolov na riešenie rôznych integračných požiadaviek. Toto môže spôsobovať problémy, keď sa snažíme rozšíriť integráciu o aplikácie, ktoré nepoužívajú tie isté protokoly.

  SOA nešpecifikuje konkrétny protokol, ktorý by mal byť použitý na prístup k službe. Kľúčový princíp je, že služby nie sú definované komunikačným protokolom, ale naopak sú na protokole nezávislé. Teda rôzne protokoly môžu byť použité na prístup k tej istej službe.

Z istého pohľadu možno SOA považovať za evolúciu architektúry, nie revolúciu. Využíva mnohé dobré myšlienky („best practices“) iných architektúr, ako napríklad klient-server. Použitie SOA pri integrácii aplikácií má nesporné výhody, pretože heterogénne systémy môžu byť integrované vďaka implementačne nezávislým rozhraniam, ktoré popisujú služby. Ďalšie výhody sa skrývajú v znížených nákladoch na vývoj a údržbu, ale flexibilita je hlavným cieľom a prínosom SOA.

1.5.2 Enterprise service bus (ESB)

Úspešná implementácia SOA si vyžaduje podporu princípov SOA tak na strane aplikácií ako aj na strane infraštruktúry. Čo sa týka aplikácií, tie môžeme zapojiť do SOA

vytvorením servisných rozhraní k existujúcim alebo novým funkciám, ktoré tieto aplikácie poskytujú. K týmto rozhraniam by sa malo pristupovať použitím infraštruktúry, ktorá dokáže smerovať a prenášať požiadavky na služby k tým správnym poskytovateľom. Čím viac funkcií sprístupní organizácia v podobe služieb, tým je dôležitejšie, aby infraštruktúra podporovala manažment SOA resp. týchto služieb na úrovni organizácie.

ESB je komponent, ktorý umožňuje implementáciu SOA. Plní v nej práve funkciu infraštruktúry, pretože podporuje hlavné koncepty SOA:

- oddeľuje (decouple) pohľad konzumenta nejakej služby od jej skutočnej implementácie,
- izoluje technické aspekty interakcie služieb,
- integruje a manažuje služby v rámci organizácie.

Toto oddelenie značne zvyšuje flexibilitu celej architektúry, pretože dovoľuje nahradenie jedného poskytovateľa služby iným bez toho, aby konzument služby o tejto zmene vedel. Najlepší spôsob ako to dosiahnuť je nenechať služby komunikovať priamo, ale cez nejakého „prostredníka“.

ESB teda zaisťuje infraštruktúru, ktorá odstraňuje akékoľvek priame spojenia medzi poskytovateľmi a konzumentmi služieb. Konzumenti sa pripájajú na zbernicu a nie priamo na konkrétneho poskytovateľa požadovanej služby. Zbernica ako taká pridáva aj ďalšiu hodnotu napríklad v podobe centrálne riešenej bezpečnosti alebo spoľahlivého doručenia. Integrovanie a manažovanie služieb mimo ich samotnej implementácie pomáha zvýšiť flexibilitu a manažovateľnosť SOA.

Zbernica je z logického pohľadu jedna entita, ktorá poskytuje centrálne riadenie, ale v skutočnosti je to množina fyzicky distribuovaných komponentov. ESB je teda distribuovaná integračná infraštruktúra s konzistentným prístupom k administrácii a manažmentu.

K uvedenej ESB charakteristike spomenieme ešte nasledovné vlastnosti [10]:

- Integrácia založená na štandardoch.

  Toto je fundamentálny koncept ESB. JMS, JCA, SOAP, XSLT, XPath, XQuery, WSDL, BPEL4WS sú všetko štandardy podporované ESB na rôznych úrovniach. Od konektivity, cez transformácie údajov až po popis a vykonávanie podnikových procesov.

- Distribuované dátové transformácie.

  Kľúčová časť každej integračnej stratégie a schopnosť jednoducho konvertovať dátové formáty medzi aplikáciami. Transformácia údajov je súčasťou zbernice. Transformačné služby sú pripojené k zbernici a prístupné pre akúkoľvek aplikáciu.

- Rozšíriteľnosť cez ďalšie vrstvy služieb.

  Táto vlastnosť hovorí o ľahkej rozšíriteľnosti ESB o ďalšie požadované funkcie. Môžu to byť napríklad špecializované transformácie údaje dodané od tretích strán.

- Udalostne riadená SOA. (event-driven SOA)

  V SOA sú aplikácie a služby chápané ako abstraktné body (abstract service endpoints), ktoré môžu pohotovo reagovať na asynchrónne udalosti. SOA

zakrýva detaily konektivity. Implementácie služieb nemusia rozumieť konkrétnym protokolom. Služby nemusia riešiť smerovanie správ iným službám. Jednoducho prijmú správu zo zbernice ako udalosť a spracujú ju. ESB sa potom postará o doručenie správy kamkoľvek je to potrebné.

ESB umožňuje vytvárať vlastné služby a spájať ich do kompozitných služieb a procesov s cieľom automatizovať podnikové funkcie v reálnom čase.

- Definícia toku procesov. (process flow)

  ESB má schopnosť definovať procesy od jednoduchých sekvencií konečných krokov až po sofistikovanú orchestráciu podnikových procesov s paralelným vykonávaním aktivít.

2 Analýza prepájaných systémov

V tejto kapitole sa pozrieme na univerzitný informačný systém ako celok. Zmapujeme existujúce aplikácie, ktoré v súčasnej dobe zamýšľame prepojiť. Popíšeme hlavne údaje o osobách, ktoré sú tieto systémy schopné poskytnúť. Na základe získaných informácií sa v ďalšej kapitole pokúsime navrhnúť možné riešenia integrácie týchto aplikácií so zámerom zdieľania údajov o osobách medzi nimi a vybrať z nich to najvhodnejšie. Súčasťou tejto analýzy je aj príloha B, kde sú popísané komunikačné rozhrania na jednotlivé systémy – najmä popis integračného zámeru, formátu a sémantiky prenášaných údajov a spôsobu prenosu.

Proces integrácie by mal byť kontinuálny. V budúcnosti by sa mali do integrácie pripájať aj ďalšie aplikácie univerzitného IS (prípadne externé systémy), pri ktorých má zmysel uvažovať o ich prepojení s inými systémami.

Tabuľka 2-1 Zoznam integrovaných systémov s ich krátkym popisom

Systém[2]	**Popis**
Študent	študijná agenda – evidencia študentov, študijných programov, predmetov, prijímacie konanie, zápisy
Personalistika a mzdy (PaM)	kompletná evidencia zamestnancov, výpočet miezd
Finančný informačný systém (FIS)	ekonomická agenda – objednávky, faktúry, účtovníctvo, rozpočet, podnikateľská činnosť
Virtuálna knižnica (VIKUK)	katalógy kníh a publikačnej činnosti pracovníkov univerzity, evidencia čitateľov a ich výpožičiek
VoIP	systém na účtovanie telefónnych hovorov uskutočnených prostredníctvom univerzitnej siete VoIP
Prístupový systém (PS)	systém na automatické odomykanie dverí / turniketov na základe priloženia univerzitného preukazu
Univerzitné validačné terminály (UT)	aktualizácia informácií v čipových kartách (univerzitné preukazy študentov a zamestnancov)
EMcard	spoločnosť pôsobiaca ako systémový integrátor, ktorá zabezpečuje akceptáciu univerzitných preukazov u externých firiem (typicky ide o autobusovú a železničnú dopravu)
HelpDesk software (HDSW)	systém na podporu práce výpočtového strediska – eviduje počítače, používateľov a incidenty
Ubytovacie systémy (UBYT)	rôzne ubytovacie systémy prevádzkované na internátoch
CKM	spoločnosť, ktorá zabezpečuje rozvoj a distribúciu medzinárodných identifikačných preukazov študenta (ISIC) a učiteľa (ITIC)
Anketa	študentská anketa
EMtest	externý dodávateľ / výrobca univerzitných preukazov
Systém na správu preukazov (SUP)	kompletná evidencia vydaných preukazov, zabezpečuje aj tlač preukazov

Keď máme vybranú množinu aplikácií, ktoré chceme integrovať, môžeme pristúpiť k analýze údajov, ktorých prenos budeme chcieť automatizovať. Je potrebné získať popis

[2] kurzívou sú označené externé systémy, ktoré nie sú prevádzkované v rámci univerzitného IS

informácií uchovávaných v týchto systémoch. Pre každý typ informácie je nutné určiť autoritatívny zdroj. To je systém, ktorý primárne udržiava danú informáciu vo svojej databáze. Týmto spôsobom potom dokážeme povedať, aké údaje, odkiaľ a kam majú tiecť.

Predtým než sa budeme zaoberať prepájaním aplikácií s hlavným zameraním na údaje o osobách, musíme najprv vyriešiť problém identifikácie osôb. Je zjavné, že každej osobe je nutné priradiť nejaký jednoznačný identifikátor. Dôvodom je, aby sme ku každej osobe boli schopní jednoducho pridružiť všetky informácie, ktoré k nej prislúchajú. Taktiež, pri prenose údajov zo zdrojových systémov musíme určiť osobu, ku ktorej treba príslušný údaj priradiť. Z tohto vyplýva požiadavka, aby každý systém, ktorý poskytuje nejaké údaje o osobách, evidoval vo svojej vnútornej databáze identifikátor pre každú osobu.

V minulosti bolo rozhodnuté, že každej osobe sa bude priraďovať unikátne *univerzitné osobné číslo* (UOČ), ktoré bude identifikovať konkrétnu osobu. Osoba môže mať len jedno UOČ aj v prípade, že by mala viacero vzťahov k univerzite.

V nasledujúcej tabuľke uvedieme pre každý systém skupiny údajov, ktoré poskytuje resp. požaduje. Vynecháme zatiaľ konkrétne atribúty.

Tabuľka 2-2 Zoznam systémov s popisom poskytovaných a požadovaných údajov

Systém	**Poskytuje**	**Požaduje**
Študent	údaje o študentoch	-
Personalistika a mzdy (PaM)	údaje o zamestnancoch	-
Finančný informačný systém (FIS)	-	údaje o zamestnancoch[3]
Virtuálna knižnica (VIKUK)	-	údaje o študentoch a ich preukazoch
VoIP	-	údaje o zamestnancoch
Prístupový systém (PS)	-	údaje o osobách a ich preukazoch
Univerzitné validačné terminály (UT)	-	údaje o osobách a ich preukazoch
EMcard	-	údaje o osobách a ich preukazoch
HelpDesk software (HDSW)	-	údaje o osobách a ubytovaní
Ubytovacie systémy (UBYT)	údaje o ubytovaní osôb	údaje o osobách
CKM	-	údaje o študentoch, učiteľoch a ich preukazoch
Anketa	-	údaje o preukazoch
EMtest	údaje o vytlačených preukazoch	údaje o osobách
Systém na správu preukazov (SUP)	údaje o preukazoch	údaje o vytlačených preukazoch

2.1 Primárne zdroje údajov

Z vyššie uvedenej tabuľky vyplýva, že hlavných zdrojov údajov je v porovnaní s celkovým počtom systémov relatívne málo. Je teda viac systémov, ktoré budú pri integrácii vystupovať len ako „konzumenti“ údajov.

[3] v budúcnosti možno aj údaje o študentoch

2.1.1 Študent

Systém Študent je aplikácia na podporu študijnej agendy. Na každom študijnom oddelení je prevádzkovaná jedna inštancia tohto systému, ktorá pracuje nad svojou lokálnou databázou a je plne nezávislá od ostatných inštancií. Systém Študent je izolovaným systémom, ktorý nie je schopný samostatne komunikovať s inými systémami.

Inštancia systému Študent je autoritatívnym zdrojom údajov o študentoch príslušnej fakulty. Konkrétne atribúty, ktoré eviduje a je schopný poskytnúť, zhrnieme v tabuľke.

Tabuľka 2-3 Údaje poskytované systémom Študent

Skupina	Atribút
Identifikátory	univerzitné osobné číslo študenta
	univerzitný kód študenta
Osobné údaje	titul(y) pred menom
	krstné meno
	Priezvisko
	rodné priezvisko
	titul(y) za menom
	rodné číslo
	dátum narodenia
	miesto narodenia
Bydlisko	trvalé bydlisko – ulica a číslo domu
	trvalé bydlisko – PSČ
	trvalé bydlisko – názov obce + označenie pošty
Informácie o študijnom vzťahu	rok nástupu na vysokú školu
	kód študijného programu
	druh štúdia
	forma štúdia
	dátum zápisu (v aktuálnom roku)
	atribút štúdia (typicky dôvod ukončenia štúdia)
	dátum ukončenia štúdia
	dátum začiatku prerušenia štúdia
	dátum konca prerušenia štúdia
Preukaz študenta	číslo preukazu (ako ho evidujú na študijnom oddelení)
	spôsob validácie preukazu
	číslo ISIC (ako ho evidujú na študijnom oddelení)
Ochrana osobných údajov	súhlasy s poskytnutím / zverejnením údajov

2.1.1.1 Personalistika a mzdy

Tento systém slúži na kompletnú evidenciu zamestnancov a výpočet miezd. Opäť ide o decentralizovaný a izolovaný systém. Na každej fakulte univerzity sa používa jedna inštancia tohto systému. Samotná aplikácia neumožňuje žiadnu automatickú komunikáciu s ostatnými systémami za účelom výmeny informácií.

Atribúty poskytovaných údajov o zamestnancoch sú v tabuľke.

Tabuľka 2-4 Údaje poskytované systémom Personalistika a mzdy

Skupina	Atribút
Identifikátory	univerzitné osobné číslo
	osobné číslo zamestnanca
Osobné údaje	krstné meno

	priezvisko
	rodné priezvisko
	tituly pred menom
	tituly za menom
	dátum narodenia
	miesto narodenia
	rodné číslo
	pohlavie
Bydlisko	adresa trvalého pobytu
Informácie o zamestnaneckom vzťahu	katedra, resp. analogické pracovisko
	druh pracovno-právneho vzťahu
	dátum vzniku pracovno-právneho vzťahu
	dátum ukončenia pracovno-právneho vzťahu

2.1.1.2 Ubytovacie aplikácie

Na každom internáte sa používa nejaká aplikácia na evidenciu ubytovaných osôb a ubytovacích kapacít. Ide o navzájom nezávislé aplikácie, ktoré môžu ostatným systémom poskytnúť informácie o ubytovaní študentov či zamestnancov UK.

Tabuľka 2-5 Údaje poskytované ubytovacími aplikáciami

Skupina	Atribút
Čas ubytovania	dátum začiatku ubytovania
	dátum konca ubytovania
Miesto ubytovania	budova
	blok
	izba

2.1.1.3 Systém na správu preukazov (SUP)

Po zavedení používania univerzitných preukazov bolo potrebné mať systém, ktorý by zabezpečoval evidenciu vydaných preukazov, tlač preukazov a podporoval by všetky procesy súvisiace s ich správou. Je to teda systém, ktorý primárne uchováva informácie o preukazoch. Aj keď tento systém neexistuje ako samostatná aplikácia, pre naše účely má zmysel považovať ho za nezávislý komponent IS UK.

Tabuľka 2-6 Údaje poskytované systémom na správu preukazov

Skupina	Atribút
Identifikátory	UOČ osoby, pre ktorú bola karta personifikovaná
	číslo čipu
	číslo licencie ISIC/ITIC (ak je karta tohto typu)
Údaje uložené pri tlači do pamäte karty	verzia záznamu
	druh karty
	začiatok platnosti
	koniec platnosti
	identifikátor
	titul(y) pred menom
	krstné meno
	priezvisko
	titul(y) za menom
	dátum narodenia

	funkcia
	číslo fakulty, ktorá kartu vydala
Údaje vytlačené na preukaz	druh vydaného preukazu (vizuálna stránka)
	pozadie pre tlač
	šablóna pre tlač
	hodnoty, ktoré sa tlačia na kartu
Stav preukazu	dátum a čas personifikácie (vytlačenia) preukazu
	aktuálny stav preukazu

2.1.2 „Konzumenti“ údajov

Táto podkapitola je venovaná stručnému popisu tých systémov, ktoré pri integrácii neposkytujú žiadne údaje iným systémom (výnimkou je len systém EMtest). Pri ich popise sa zameriame na dôvod, prečo tieto systémy chceme zahrnúť do integrácie a tiež na dáta, ktoré požadujú.

Použitím automatického prenosu údajov zo zdrojových systémov získame minimálne jednu výhodu, a to elimináciu zbytočného viacnásobného zadávania údajov. To pomôže znížiť počet chýb, ktoré pri manuálnom zadávaní vznikajú. Zároveň to uľahčí prácu ľudí obsluhujúcich tieto systémy a v konečnom dôsledku to bude viesť k zrýchleniu príslušných univerzitných procesov.

2.1.2.1 Virtuálna knižnica (VIKUK)

Ide o systém, ktorý zabezpečuje kompletnú evidenciu čitateľov a ich výpožičiek. Taktiež eviduje katalógy kníh a publikačnú činnosť pracovníkov univerzity. Integračným zámerom je práve aktualizácia zoznamu čitateľov.

V súčasnej dobe VIKUK požaduje automatický prenos informácií o študentoch, ich študijných vzťahoch a preukazoch. Univerzitné preukazy sú používané na identifikáciu čitateľov v knižniciach, preto je potrebné poskytovať čísla preukazov systému VIKUK. Požadované atribúty opäť zhrnieme v tabuľke.

Tabuľka 2-7 Údaje požadované systémom VIKUK

Skupina	**Atribút**
Identifikátor	univerzitné osobné číslo študenta
Osobné údaje	krstné meno
	Priezvisko
	dátum narodenia
Bydlisko	trvalé bydlisko – ulica a číslo domu
	trvalé bydlisko – PSČ
	trvalé bydlisko – názov obce + označenie pošty
Údaje o preukaze	číslo karty
Informácie o študijnom vzťahu	skratka fakulty
	kód študijného programu
	atribút štúdia (typicky dôvod ukončenia štúdia)
	príznak, či už študent obhájil záverečnú prácu (prislúchajúcu k danému študijnému vzťahu)[4]

Príznak určujúci obhájenie záverečnej práce nie je momentálne evidovaný v systéme Študent. V súčasnej dobe teda nie sme schopní poskytnúť túto informáciu

[4] podľa typu štúdia to môže byť bakalárska práca, diplomová práca, …

systému VIKUK. Očakáva sa, že v novej verzii študijného systému bude tento atribút evidovaný a bude môcť byť poskytnutý iným systémom.

2.1.2.2 VoIP

Systém VoIP slúži na automatické účtovanie hovorov uskutočnených prostredníctvom univerzitnej siete VoIP. Vo svojej lokálnej databáze eviduje oprávnených používateľov systému VoIP. Títo používatelia sú všetci zamestnancami univerzity. Má teda zmysel synchronizovať údaje o používateľoch v systéme VoIP s aktuálnym zoznamom zamestnancov univerzity. Navyše, systém VoIP požaduje zoznam pracovísk (oddelení).

Tabuľka 2-8 Údaje o zamestnancoch požadované systémom VoIP

Skupina	Atribút
Identifikátor	univerzitné osobné číslo zamestnanca
Základné údaje o zamestnancovi	krstné meno
	priezvisko
	prihlasovacie meno
	e-mailová adresa
Pracovisko	identifikátor oddelenia

Tabuľka 2-9 Údaje o pracoviskách požadované systémom VoIP

Skupina	Atribút
Informácie o pracovisku	identifikátor oddelenia
	úplný názov oddelenia
	e-mailová adresa vedúceho oddelenia

2.1.2.3 Prístupový systém

Tento systém je určený na automatické riadenie vstupu do miestností (cez dvere alebo turnikety) na základe priloženia čipovej karty. V tomto systéme sú zadefinovaní používatelia a skupiny používateľov. Tieto informácie potom systém využíva pri odomykaní dverí alebo turniketov.

Zámerom integrácie je automatický prenos údajov o osobách a skupinách do prístupového systému. Prístupový systém používa skupiny používateľov na zjednodušenie administrácie prístupových oprávnení k miestnostiam – operátor môže prideľovať prístupové oprávnenia na úrovni skupín. Napríklad by mohol chcieť povoliť prístup to počítačovej učebne všetkým študentom príslušnej fakulty. Aby toto mohlo fungovať, prístupový systém potrebuje pre každú osobu poznať množinu skupín, do ktorých patrí. Zaujímavá je teda otázka definície skupín. Ako flexibilne riešiť otázku priraďovania osôb do skupín a na základe čoho je možné ku konkrétnej osobe určiť množinu skupín, do ktorých má patriť.

Tabuľka 2-10 Údaje o osobách požadované prístupovým systémom

Skupina	Atribút
Identifikátor osoby	univerzitné osobné číslo
Informácie o preukaze	číslo karty
Osobné údaje	tituly pre menom
	meno
	priezvisko
	tituly za menom
Vzťah k univerzite	typ primárneho vzťahu k univerzite

	1. „Zamestnanec“ 2. „Interný doktorand“ 3. „Študent“ 4. „Externý študent“
	názov súčasti UK

Tabuľka 2-11 Údaje o skupinách používateľov požadované prístupovým systémom

Skupina	Atribút
Informácie o skupine osôb	názov skupiny
	zoznam osôb patriacich do skupiny (zoznam čísiel UOČ)

Zostáva ešte vyriešiť problém s určovaním primárneho vzťahu osoby k univerzite v prípade, že má s univerzitou viac rôznych vzťahov. Napríklad osoba môže byť súčasne študentom aj zamestnancom univerzity.

Jedným z možných riešení je pevne definovať prioritu jednotlivým typom vzťahov. Táto definícia by bola uložená v externom súbore alebo databáze tak, aby pri zmene definície priorít nemuselo dôjsť k zmene aplikačného kódu. Pri posielaní údajov do prístupového systému by sa pre každú osobu vyhodnotili všetky jej univerzitné vzťahy a na základe definovanej priority by sa vybral primárny. Toto riešenie však predpokladá, že v momente generovania resp. posielania údajov do prístupového systému budú k dispozícii informácie o aktuálnych študentských a zamestnaneckých vzťahoch. Tieto sú uložené v dvoch nezávislých systémoch *Študent* a *Personalistika a mzdy*. Je tu potrebná istá miera synchronizácie.

O niečo závažnejším problémom je určenie súčasti univerzity prislúchajúcej primárnemu vzťahu osoby v prípade, že osoba má niekoľko vzťahov toho istého typu. Príkladom môže byť študent študujúci na dvoch fakultách. V tejto situácii je nerozhodnuteľné (bez zásahu človeka), ktorý vzťah je primárny.

2.1.2.4 Univerzitné validačné terminály

Validačné terminály slúžia na aktualizáciu informácií v čipových kartách (univerzitné preukazy študentov a zamestnancov). Terminál musí mať aktuálne údaje o preukazoch, aby po priložení preukazu dokázal tento rozpoznať a zapísať na čip korektné informácie. Požadované údaje o každom preukaze a jeho držiteľovi sú uvedené v tabuľke.

Tabuľka 2-12 Údaje požadované univerzitným validačným terminálom

Skupina	Atribút
Identifikácia preukazu	číslo karty
	druh karty
Platnosť preukazu	začiatok platnosti preukazu
	koniec platnosti preukazu
Identifikácia držiteľa preukazu	univerzitné osobné číslo
Osobné údaje držiteľa preukazu	tituly pred menom
	krstné meno
	priezvisko
	tituly za menom
	dátum narodenia
Bydlisko	ulica – trvalé bydlisko
	mesto – trvalé bydlisko
	PSČ – trvalé bydlisko

	ulica – prechodné bydlisko
	mesto – prechodné bydlisko
	PSČ – prechodné bydlisko
Atribúty označujúce použitie preukazu aj mimo univerzity	aktivovať autobusovú dopravnú časť
	aktivovať vlakovú dopravnú časť
	nesúhlas s poskytnutím osobných údajov
Fakulta	kód fakulty
	PSČ fakulty

Prechodné bydlisko nie je momentálne evidované v zdrojových systémoch Študent a PaM, a preto nemôže byť v súčasnosti poskytované univerzitným terminálom.

2.1.2.5 EMcard

Spoločnosť EMcard poskytuje služby systémového integrátora. Zabezpečuje akceptáciu študentských univerzitných preukazov u externých firiem - predovšetkým autobusová a železničná doprava. Vďaka službám EMcardu môžu študenti v prípade záujmu používať svoje univerzitné preukazy aj ako preukážky pri preprave autobusovou alebo vlakovou dopravou. EMcard teda potrebuje údaje o tých preukazoch študentov, ktorých držitelia majú záujem využívať preukaz aj mimo univerzity.

EMcard požaduje od univerzitného systému rovnaké atribúty ako validačné terminály[5], avšak s malými obmedzeniami (bez uvedenia UOČ, ktoré v tomto prípade nie je potrebné).

2.1.2.6 HelpDesk software (HDSW)

Systém HDSW slúži na podporu práce výpočtového strediska. Eviduje používateľov, počítače a incidenty a umožňuje vykonávať niektoré administrátorské úkony. V prevádzke je niekoľko inštancií tohto systému (každá na jednej konkrétnej súčasti UK).

Z hľadiska integrácie je zaujímavý zoznam používateľov evidovaných v tomto systéme. Zámerom je posielať každej inštancii systému relevantný zoznam osôb, vzhľadom na príslušnú súčasť.

Tabuľka 2-13 Údaje požadované systémom HDSW

Skupina	**Atribút**
Identifikátor osoby	univerzitné osobné číslo
Osobné údaje	titul(y) pred menom
	krstné meno
	priezvisko
	titul(y) za menom
Typ vzťahu k univerzite	vzťah k univerzite („S“ = študent, „Z“ = zamestnanec)
Umiestnenie	objekt/budova
	blok
	izba/miestnosť
Kontaktné údaje	e-mailová adresa
	telefónne číslo

[5] viď Tabuľka 2-12 Údaje požadované univerzitným validačným terminálom

Doplňujúce informácie	fotografia

Opäť tu vzniká problém s určením primárneho vzťahu k univerzite v prípade osôb, ktoré aktuálne majú viac rôznych vzťahov.

2.1.2.7 CKM

CKM 2000 Travel je spoločnosť, ktorá v rámci svojich aktivít zabezpečuje rozvoj a distribúciu preukazov ISIC a ITIC. Univerzitný preukaz študenta denného štúdia je riadnym preukazom ISIC. Podobne, univerzitný preukaz učiteľa na UK je riadnym preukazom ITIC. Aby CKM mohla poskytovať svoje služby držiteľom týchto medzinárodných preukazov, potrebuje mať k dispozícii aktuálne údaje o vydaných preukazoch ISIC/ITIC. Predovšetkým musí CKM poznať čísla pridelených licencií ISIC/ITIC.

Tabuľka 2-14 Údaje požadované spoločnosťou CKM

Skupina	**Atribút**
Informácie o preukaze	číslo licencie ISIC/ITIC
Osobné údaje držiteľa preukazu	meno
	priezvisko
	trvalé bydlisko – ulica
	trvalé bydlisko – mesto
	trvalé bydlisko – PSČ
	dátum narodenia
	označenie pohlavia

Atribút určujúci pohlavie osoby nie je momentálne explicitne evidovaný. Jeho hodnota sa odvodzuje z rodného čísla s tým, že problém vzniká pri zahraničných študentoch (prípadne učiteľoch), ktorí nemajú rodné číslo.

2.1.2.8 Anketa

Anketový systém je systém na podporu študentskej ankety. Študenti v nej hodnotia predmety, ktoré mali zapísané v danom semestri a učiteľov, ktorí ich vyučovali. Pred samotným hodnotením sa každý študent musí prihlásiť číslom preukazom. Anketový systém overuje, či preukaz patrí študentovi danej fakulty. Údaje o číslach preukazov spoločne s príslušnými fakultami anketový systém získa automatickým prenosom.

Tabuľka 2-15 Údaje požadované anketovým systémom

Skupina	**Atribút**
Informácie o preukaze	číslo preukazu
Údaj o fakulte	číslo fakulty (na ktorej má držiteľ príslušného preukazu otvorený študijný vzťah)

2.1.2.9 EMtest

Spoločnosť, ktorá vykonáva pre univerzitu hromadnú tlač preukazov. Univerzitný systém na správu preukazov poskytne EMtestu všetky údaje potrebné pre vytlačenie preukazov. Na druhej strane, EMtest poskytne čísla vytlačených preukazov. EMtest teda nie je len konzumentom údajov, hoci sme ho zaradili do tejto časti.

2.1.2.10 Finančný informačný systém (FIS)

FIS je systém na podporu finančnej agendy na univerzite – objednávky, faktúry, dodávatelia, odberatelia, prideľovanie finančných zdrojov a podobne. Integračným zámerom je pravidelná aktualizácia zoznamu osôb, ktoré vo FIS vystupujú ako dodávatelia resp. odberatelia. V súčasnosti sú tieto údaje do systému FIS zadávané ručne operátorom.

Tabuľka 2-16 Údaje zadávané do systému FIS

Skupina	**Atribút**
Identifikátor osoby	univerzitné osobné číslo
Osobné údaje	titul(y) pred menom
	krstné meno
	priezvisko
	titul(y) za menom
Adresa	ulica
	mesto
	PSČ

3 Návrh riešenia prepojenia systémov

V tejto kapitole prediskutujeme možnosti riešenia integrácie uvedených aplikácií na abstraktnej úrovni z pohľadu architektúry prepojenia týchto aplikácií. Zároveň uvedieme doplňujúce požiadavky, ktoré musí zvolené riešenie spĺňať, aby bolo možné integráciu uskutočniť. Nakoniec sa pokúsime vybrať najvhodnejšie riešenie pre univerzitný IS aj s ohľadom na požadované používateľské funkcie.

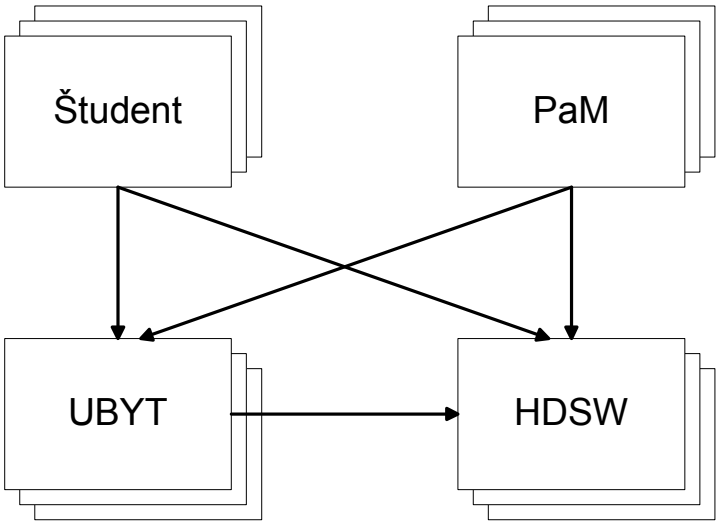
Kvôli prehľadnosti a zrozumiteľnosti budeme všetky alternatívy graficky ilustrovať na reprezentatívnej podmnožine prepájaných systémov.

3.1 Možnosti riešenia

Na základe analýzy potrieb a možností jednotlivých prepájaných aplikácií navrhneme niekoľko možností ako možno aplikácie integrovať. Navrhnuté riešenia možno rozlišovať na základe niekoľkých kritérií. Prvým z nich je existencia centrálneho prvku, ktorý by prepájal jednotlivé systémy a umožňoval by ich vzájomnú komunikáciu. Druhým kritériom, špecifickým pre IS univerzity, je existencia spoločnej centrálnej databázy údajov o osobách. Tretím kritériom je podpora implementácie používateľských funkcií (hlavne prezeranie údajov a notifikácie o ich zmene). Význam týchto kritérií sa viditeľnejšie prejaví pri jednotlivých alternatívach.

3.1.1 Integrácia „ad hoc“

Z istého pohľadu najjednoduchším, avšak nie veľmi premysleným riešením, by bolo implementovať akúsi „ad hoc“ integráciu bez použitia spoločného komunikačného prvku a tiež bez začlenenia centrálnej databázy údajov. Inými slovami, aplikácie, ktoré by potrebovali medzi sebou vymieňať údaje resp. poskytovať služby, by komunikovali priamo a nie prostredníctvom nezávislého „integračného prvku“.



Obrázok 3-1 Integrácia „ad hoc“

Implementácia tohto riešenia by bola neúmerne náročná, hlavne kvôli počtu prepojení. Uvedieme jeden príklad. Na obrázku vidieť, že potrebujeme zabezpečiť komunikáciu medzi systémom *Študent* a ubytovacími aplikáciami. Inštancií systému *Študent* je prevádzkovaných spolu 13 a ubytovacie aplikácie sú 3. Keďže nemáme žiadny centrálny prvok, je nutné zrealizovať 39 prepojení. Navyše je možné, že tieto prepojenia by

neboli všetky rovnaké, pretože ubytovacie aplikácie sú rôzne a navzájom nezávislé, a teda by mohli vyžadovať rôzne formáty údajov resp. spôsoby komunikácie.

Ukazuje sa tu značná komplexnosť a neprehľadnosť takto riešenej integrácie. Postupom času by sa do integrácie zapájali ďalšie aplikácie, čo by znamenalo ďalší relatívne rýchly nárast počtu integračných rozhraní. V istom bode by sa takáto integrácia stala len veľmi ťažko udržiavateľnou.

Z pohľadu jednej aplikácie, ktorá by sa chcela zapojiť do integrácie, sa tu vynára otázka nárokov kladených na túto aplikáciu. Čím viac prepojení by aplikácia mala, tým väčší by bol dopad integrácie na túto aplikáciu. Hlavne preto, že aplikácia musí sama riadiť a obhospodarovať všetky svoje prepojenia s ostatnými systémami. Každý typ prepojenia má špecifický formát údajov, ktorý aplikácia musí vedieť spracovať. Ak by sme navyše pripustili, že by inštancie toho istého systému mohli mať rôzne požiadavky na formát prenášaných údajov, situácia sa ešte skomplikuje. Toto kladie veľké nároky na prepájané systémy, ktoré si pri integrácii vo väčšine prípadov nemôžeme dovoliť.

Z hľadiska udržiavateľnosti nás môže zaujímať modifikovateľnosť. V prípade väčšej zmeny v jednej aplikácii, ktorá by ovplyvnila rozhranie s ostatnými systémami, by bolo nutné upraviť všetky jej prepojenia, čo v konečnom dôsledku znamená zmenu vo všetkých systémoch, s ktorými táto aplikácia komunikuje. Tento problém súvisí so zviazanosťou jednotlivých systémov, ktorá je pri tomto riešení dosť vysoká. Zviazanosť systémov je vlastnosť, ktorú sa pri riešení integrácie snažíme minimalizovať.

Z pohľadu prevádzkovania takejto integrácie je vhodné zamyslieť sa nad funkciami monitorovania a logovania, ktoré sú dôležité najmä pri diagnostike a riešení prípadných problémov. Je zjavné, že spomenuté funkcie sa tu nedajú riešiť konzistentne na jednom mieste, ale budú roztrúsené do jednotlivých systémov. Tento fakt znamená dve veci. Jednak je to ďalší nárok na prepájané systémy a tiež to výrazne sťaží monitorovanie fungovania celého integrovaného systému. To sa potom odrazí na zložitejšej diagnostike vzniknutých problémov a teda aj na zvýšenom čase potrebnom na riešenie problému.

Toto riešenie môže byť dostačujúce v niektorých situáciách. Typicky vtedy, keď chceme prepojiť malý počet aplikácií a do budúcnosti nie je predpoklad, že by sa ich počet mohol zvýšiť. A taktiež, keď neočakávame časté zmeny integračných nárokov jednotlivých aplikácií.

Ako vidno, takéto riešenie nevykazuje vlastnosti, ktoré očakávame od dobre integrovaného, robustného a flexibilného informačného systému. Jasne sa tu však ukazuje potreba akéhosi centrálneho prvku, ktorý by „riadil“ prepojenia medzi aplikáciami a zároveň by umožňoval dostatočne flexibilne tieto prepojenia definovať. Pre každý integrovaný systém by potom stačilo definovať a implementovať jedno rozhranie, cez ktoré by komunikoval s centrálnym prvkom.

Implementácia používateľských funkcií by si v tomto prípade vyžiadala vytvorenie samostatnej aplikácie, ktorá by tieto funkcie poskytovala. Táto aplikácia by takisto musela často komunikovať s ostatnými aplikáciami za účelom získavania údajov, čiže by prispela k zložitosti väzieb medzi systémami.

3.1.2 Použitie integračného nástroja

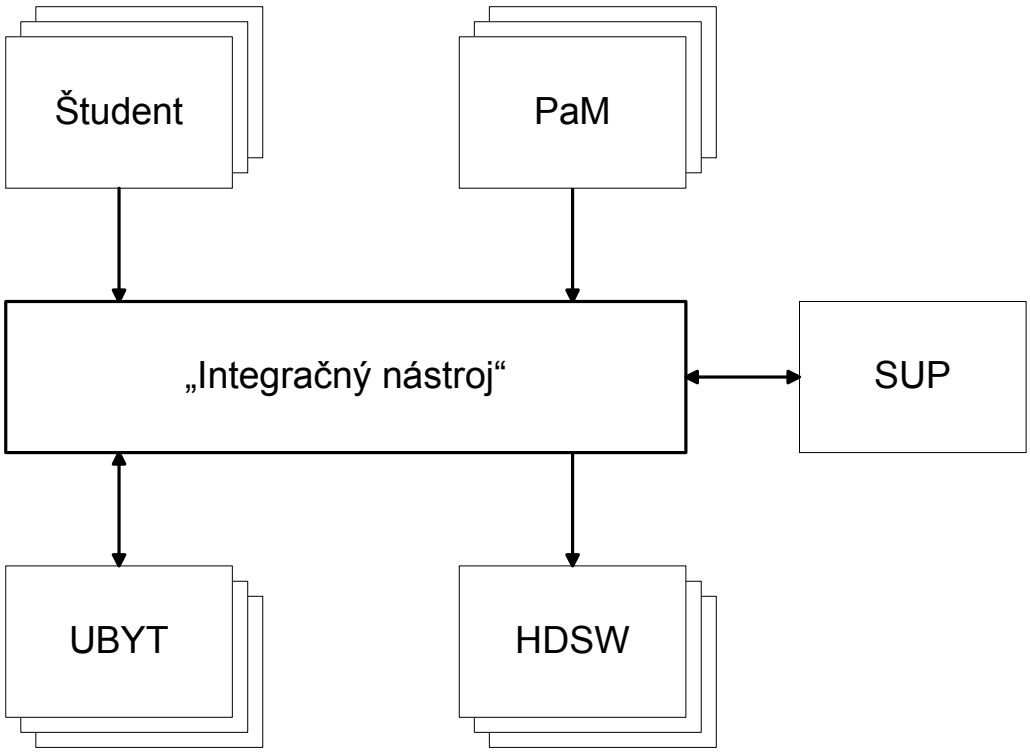
Ďalšou možnosťou ako integrovať určené aplikácie je použiť nezávislý integračný nástroj v úlohe centrálneho prvku. Pričom nepredpokladáme existenciu centrálnej databázy údajov o osobách.

Pod integračným nástrojom rozumieme aplikáciu špeciálne vyvinutú a určenú na riešenie integrácie aplikácií. Aplikácie tohto typu, nazývané aj middleware či podniková zbernica služieb (enterprise service bus, ESB), komplexne riešia integráciu aplikácií. Okrem toho, že tvoria komunikačnú vrstvu medzi prepájanými systémami, poskytujú aj ďalšie funkcie.

Výhody, ktoré tieto nástroje ponúkajú, sú viaceré:

- veľmi malá zviazanosť integrovaných systémov,
- centrálne riešené auditovanie a logovanie, čo umožňuje ľahšie monitorovať stav a diagnostikovať správanie sa celého komplexného distribuovaného systému,
- centralizovaná konfigurácia,
- transformácie údajov na základe pravidiel definovaných administrátorom systému,
- minimálne nároky kladené na prepájané systémy,
- spoľahlivá komunikácia medzi systémami,
- inteligentné smerovanie údajov,
- niekoľko spôsobov napojenia integrovaných systémov (webové služby, podpora C/C++ a Java aplikačného rozhrania, COM, veľa predpripravených adaptérov),
- dôsledné využívanie štandardov (XML, SOAP, WSDL, UDDI, HTTP/S),
- niektoré integračné nástroje sú už navrhnuté so zreteľom na známe a používané integračné vzory a tým uľahčujú aplikáciu týchto vzorov pri riešení konkrétnej integrácie.

Jednou z hlavných výhod je možnosť jednoduchého definovania integračných väzieb na vyššej úrovni. Pri tejto definícii sa uvedie odkiaľ, kam a kedy majú byť údaje prenesené, ako aj to, aké transformácie sa na tieto údaje majú aplikovať. Použitím takéhoto nástroja teda získavame dodatočnú flexibilitu.



Obrázok 3-2 Použitie integračného nástroja

Integračný nástroj dokáže odstrániť všetky nedostatky „ad hoc“ integrácie a zvykne sa často používať na riešenie integrácie. Netreba tu však zabúdať na zásadnú vlastnosť každého integračného nástroja a tou je jeho „bezstavovosť“. Je schopný prijať údaje z jedného systému a na základe definovaných pravidiel aplikovať potrebné transformácie a poslať iným systémom. Žiadne z týchto prenášaných údajov si neuchováva trvale. Samozrejme, ak je cieľový systém nedostupný, integračný nástroj dočasne uchováva údaje pre tento systém v záujme garantovania spoľahlivého prenosu. Keď sa však podarí údaje cieľovému systému doručiť, nie sú naďalej uchované v integračnom nástroji.

Popísané fungovanie je vo všeobecnosti akceptovateľné. Sú však situácie, kedy nám tento spôsob fungovania nevyhovuje, pretože nie je schopný uspokojiť integračné nároky všetkých aplikácií. V takejto situácii sa vo svojom súčasnom stave nachádza aj informačný systém univerzity.

Primárne a najdôležitejšie zdroje údajov o osobách, systémy *Študent* a *PaM*, sú v súčasnej dobe decentralizované a navyše nemôžu údaje poskytovať kedykoľvek, pretože nie sú v nepretržitej 24 hodinovej prevádzke. Na druhej strane, medzi integrovanými systémami sú aj také systémy, ktoré zo svojej podstaty vždy potrebujú dostávať kompletné údaje a nie len správy o zmenách v údajoch za nejaké časové obdobie. Takéto systémy integračný nástroj nedokáže obslúžiť, keďže potrebné údaje nemá stále k dispozícii. Navyše sú tieto systémy schopné údaje len prijímať a teda nemôžu použiť mechanizmus komunikácie „request-reply“, cez ktorý by si od zdrojového systému dokázali vyžiadať kompletné údaje. Problémom je teda decentralizovanosť a častá nedostupnosť zdrojových systémov.

Istým riešením by bolo požadovať od zdrojových systémov, aby vždy posielali kompletné údaje. Integračné väzby by boli v nástroji definované tak, aby sa tieto údaje automaticky posielali do špecifických cieľových systémov popísaných vyššie. Vzniká tu však nežiadúca závislosť, čo sa týka frekvencie prenosu údajov. Cieľové systémy si totiž

v tomto prípade nemôžu určiť kedy a ako často chcú prijímať údaje, pretože prenos by bol priamo viazaný na prenos údajov zo zdrojových systémov.

Alternatíva založená na použití integračného nástroja bez existencie centrálneho skladu údajov o osobách je teda použiteľná len s obmedzeniami uvedenými vyššie.

Z pohľadu požadovaných používateľských funkcií je v tomto prípade situácia rovnaká ako pri integrácii „ad hoc“ s tým rozdielom, že vytvorená používateľská aplikácia by nekomunikovala s ostatnými systémami priamo, ale prostredníctvom integračného nástroja. Frekvencia tejto komunikácie by však bola rovnako vysoká, pretože každý používateľom požadovaný údaj by musela získavať priamo od ostatných prepojených systémov.

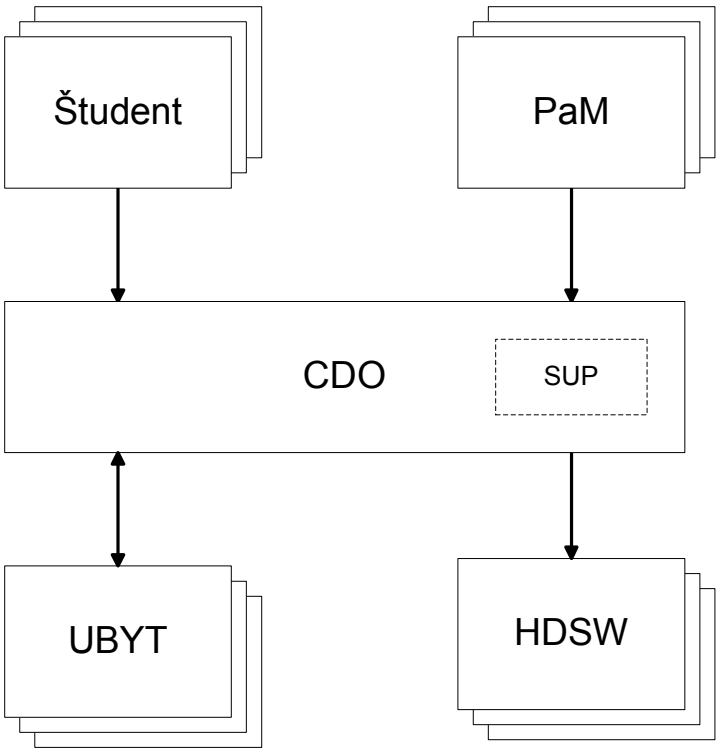
3.1.3 Použitie Centrálnej databázy osôb

Ďalšou možnosťou ako prepojiť aplikácie na UK je vytvoriť Centrálnu databázu osôb (CDO). Táto by nám pomohla vyriešiť problémy s decentralizovanosťou a nedostupnosťou zdrojových systémov, pretože by bola stavovým prvkom a uchovávala by vo svojej databáze údaje o osobách. Tieto údaje by získavala pravidelným prenosom zo zdrojových systémov. Teda v podstate by CDO bola akýmsi „stavovým integračným prvkom“, t.j. bol by to element so zdvojenou funkciou.

Konkrétne:

- CDO ako sklad údajov o osobách a preukazoch a zároveň systém poskytujúci ďalšie služby (generovanie UOČ, správa preukazov, notifikácie o zmene údajov),
- CDO ako integračný prvok, ktorý má v sebe zapuzdrené pravidlá odkiaľ, kam a aké údaje sa majú presúvať.

Medzi ďalšie služby, ktoré by CDO mohla poskytovať patria hlavne funkcie súvisiace so správou preukazov, evidencia a prideľovanie UOČ a v budúcnosti prípadne aj iné. Zároveň by sme CDO mohli využiť na implementáciu používateľských funkcií a používateľského rozhrania na prístup a prácu s uloženými údajmi. CDO by teda mohla vzniknúť v podstate rozšírením funkcionality systému na správu preukazov. Výhodou je obmedzenie komunikácie s inými systémami, pretože používateľské rozhranie (a jeho funkcie) budú pracovať s lokálnou databázou údajov o osobách.



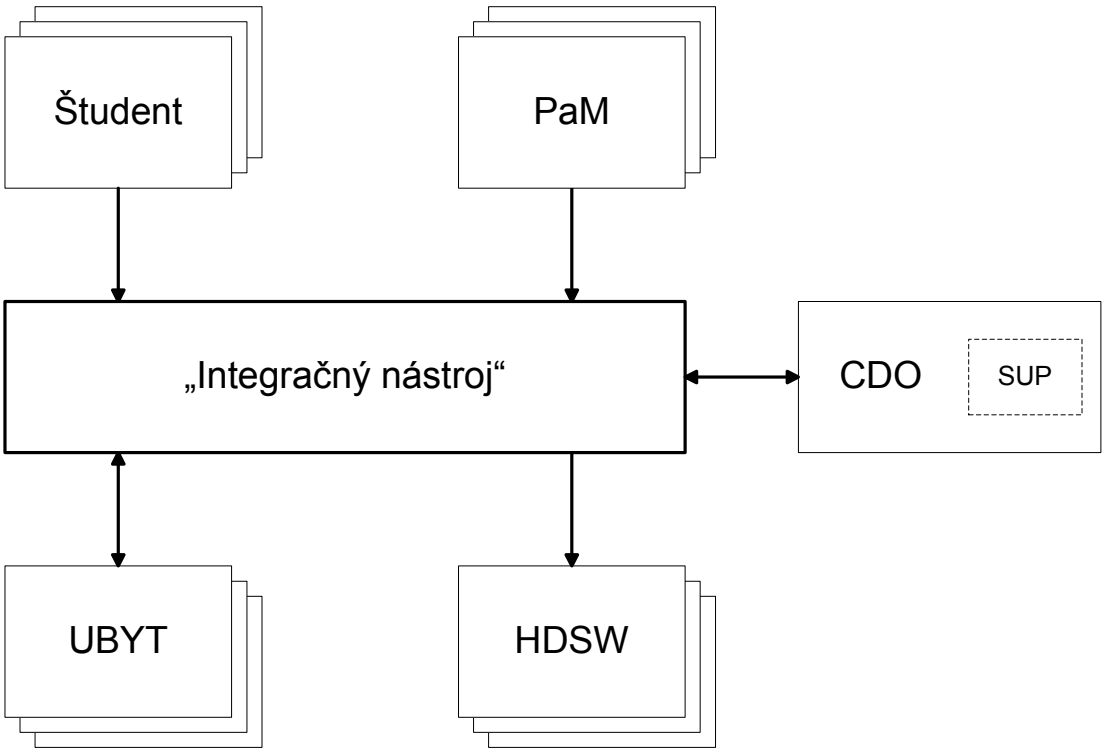
Obrázok 3-3 Integrácia len použitím CDO

Ak by bola CDO navrhnutá s dostatočnou flexibilnosťou, čo sa týka pridávania resp. modifikácie integračných rozhraní a čo sa týka evidovaných atribútov o osobe, získali by sme veľmi dobré integračné riešenie bez závažnejších nedostatkov.

Zamyslíme sa nad tým, či je žiadúce, aby CDO uchovávala všetky atribúty o osobách, ktoré cez ňu tečú. Výhodou je fakt, že CDO potom môže kedykoľvek poskytnúť tieto údaje iným systémom. Na druhej strane, množina evidovaných atribútov sa tým zväčšuje a niektoré z nich môžu byť len „okrajové“, v zmysle, že sú napríklad požadované len jedným systémom. Treba starostlivo zvažovať, ktoré atribúty budú v CDO uložené. Ak napríklad systém Študent bude plne „online“, môže si od neho CDO údaje vyžiadať kedykoľvek a bude mať veľký význam ukladať niektoré atribúty duplicitne v oboch systémoch.

3.1.4 Použitie integračného nástroja a Centrálnej databázy osôb

Poslednou alternatívou je kombinované riešenie, ktoré ako integračnú vrstvu používa samostatný integračný nástroj a zároveň predpokladá, že súčasťou architektúry je Centrálna databáza osôb. To z pôvodne bezstavového riešenia robí stavové, pričom máme zachované všetky výhody plynúce z použitia špecializovaného integračného nástroja.



Obrázok 3-4 Integrácia súčasným použitím integračného nástroja a CDO

Pri implementácii tohto scenára je vhodné sa krátko zamyslieť nad postavením a úlohou CDO v celkovej architektúre integrovaného systému. Z dôvodov uvedených vyššie potrebujeme mať CDO ako systém na kompletnú evidenciu údajov o osobách. Tieto údaje by plynuli zo zdrojových systémov do CDO, kde by boli uchovávané v čo možno najviac normalizovanom tvare a boli by kedykoľvek dostupné pre ľubovoľný iný systém. CDO si samozrejme naďalej zachová funkcie správy preukazov a prideľovania UOČ osobám, pričom v budúcnosti sa množina služieb, ktoré CDO poskytuje, bude pravdepodobne ďalej rozširovať.

3.2 Vyhodnotenie

V predchádzajúcej podkapitole boli prezentované štyri možnosti riešenia integrácie na UK. Pri každej boli rozdiskutované niektoré hlavné pozitíva a negatíva príslušného prístupu k riešeniu. Cieľom tejto podkapitoly je zhodnotiť dostupné alternatívy a zvoliť tú najvhodnejšiu.

3.2.1 Prijateľné možnosti riešenia

Prvé dve prezentované alternatívy, ktoré neuvažovali existenciu Centrálnej databázy osôb, nebudeme v tomto vyhodnotení brať do úvahy resp. vylúčime ich hneď na začiatku.

Z požiadaviek a z toho, čo bolo uvedené v predchádzajúcej časti, vyplýva, že v súčasnom stave univerzitného IS je existencia CDO nutnosťou. Keď v budúcnosti budú systémy *Študent* a *PaM* úplne „on-line“ a centralizované, úloha CDO sa pravdepodobne zredukuje, avšak naďalej bude tento systém potrebný ako poskytovateľ niektorých služieb pre ostatné systémy resp. pre používateľov. Je otázne, či si CDO ponechá funkciu centrálneho skladu údajov o osobách. Zatiaľ predpokladáme, že áno, pretože to zjednoduší implementáciu niektorých používateľských funkcií. Ako napríklad vyhľadávanie a zobrazovanie informácií o osobách. Množina evidovaných atribútov sa však

pravdepodobne zmenší. Navyše okrem spomínaných zdrojových systémov tu môžu byť ďalšie zdroje údajov, ktoré nie sú v nepretržitej prevádzke a teda informácie z nich musia byť uložené v permanentne dostupnom systéme, akým je práve CDO. Príkladom sú ubytovacie aplikácie.

Zostávajú nám teda na porovnanie dve alternatívy predpokladajúce použitie CDO. A síce, čisté použitie Centrálnej databázy osôb (ďalej aj „alternatíva CDO“) alebo použitie integračného nástroja v spolupráci s CDO (ďalej aj „alternatíva EAI“). Obidve sú v skutočnosti dobre použiteľné a netrpia závažnejším nedostatkom. Zdá sa, že obe alternatívy sú si veľmi podobné a jediný rozdiel je v tom, či si integračný element vytvoríme sami alebo radšej použijeme už hotový profesionálny nástroj. Zanalyzujeme teraz tieto riešenia z pohľadu oblastí, v ktorých by sa mohli odlišovať.

3.2.2 Kúpiť alebo vyvinúť?

Pri porovnávaní uvedených alternatív sa musíme zaoberať aj klasickým problémom typu „kúpiť alebo vyvinúť?“. Na strane „kúpiť“ je alternatíva EAI a na strane „vyvinúť“ zase alternatíva CDO. Toto rozdelenie však nie je úplne korektné, pretože aj pri použití alternatívy EAI bude nutné vyvinúť ďalšie komponenty – napr. adaptéry alebo komponenty na transformáciu údajov medzi rôznymi formátmi.

V nasledujúcich odstavcoch sa budeme všeobecnej zaoberať spomenutým problémom „kúpiť alebo vyvinúť?“. Odpoveď na túto otázku často nebýva jednoznačná. Vždy je potrebné brať do úvahy viac kritérií a výsledné rozhodnutie do veľkej miery závisí od priorít, ktoré týmto kritériám pridelíme.

Dostupnosť softvérového produktu spĺňajúceho všetky naše stanovené požiadavky nie je jediným kritériom pri rozhodovaní, či vyvinúť vlastnú aplikáciu alebo zakúpiť existujúcu od externého dodávateľa. Medzi ďalšie aspekty, na ktoré je nutné pri rozhodovaní myslieť, patria:

- kvalita softvéru,
- riziko,
- financie,
- stabilita a spoľahlivosť dodávateľa,
- podpora (support) a ďalší rozvoj či úpravy v budúcnosti.

Tieto aspekty spolu relatívne úzko súvisia. Finančná náročnosť sa často nedá dopredu presne odhadnúť. Je lacnejšie kúpiť softvér a potom následne platiť za jeho úpravy a údržbu alebo sa viac oplatí vyvinúť vlastnú aplikáciu?

Vlastný vývoj je vždy spojený s nezanedbateľným rizikom. Softvérový projekt totiž stojí organizáciu nemalé finančné prostriedky, pričom nie je záruka, že na konci vývoja získame kvalitný produkt spĺňajúci naše požiadavky. Ak ho aj získame, je možné, že to bude finančne náročnejšie než zakúpenie nejakej komerčnej aplikácie. Z tohto pohľadu je teda bezpečnejšie kúpiť hotový softvér, najlepšie taký, ktorý už prešiel niekoľkoročným vývojom a testovaním. V tom prípade je totiž predpoklad, že bude dostatočne kvalitný a odladený. Pri kúpe softvéru je zároveň nutné posúdiť stabilitu a spoľahlivosť dodávateľa tohto produktu. Je to najmä z dôvodu nevyhnutnej podpory v budúcnosti. Žiadna organizácia nechce vlastniť softvér, ktorého dodávateľ nie je schopný zaistiť ďalší rozvoj aplikácie.

V neprospech vlastného vývoja aplikácie hovorí ešte jedna nie celkom žiadúca vlastnosť. Aplikácie bývajú často vyvíjané jedným alebo dvoma ľuďmi a samotný vývoj sa neriadi žiadnym modelom procesu vývoja softvéru. Vo všeobecnosti sa pri týchto projektoch málo používajú vyspelé techniky softvérového inžinierstva. Má to dva nepríjemné následky.

Po prvé, vzniknutý produkt nie je spravidla tak kvalitný a udržiavateľný ako by mohol resp. mal byť. To pre organizáciu znamená ďalšie náklady spojené s údržbou a zvyšovaním kvality vyvinutého systému.

Po druhé, ďalší rozvoj aplikácie je priamo závislý na jednom človeku[6], pretože on jediný rozumie vnútornej štruktúre a fungovaniu systému. Čiastočne eliminovať sa to dá „zaškolením" ďalšieho pracovníka, ale to opäť znamená finančné a časové nároky.

Týmto problémom sa dá zabrániť, ak sa na ne myslí hneď na začiatku softvérového projektu a ak sa nezanedbáva tvorba dokumentácie. K tejto téme sa žiada ešte dodať, že v prípade obstarania softvéru od externého dodávateľa sa vystavujeme istej závislosti na tom konkrétnom dodávateľovi (tzv. „vendor lock-in"). Výber vhodného dodávateľa je tiež netriviálna záležitosť.

Ďalším argumentom proti vlastnému vývoju je dostupnosť niekoľkých kvalitných integračných nástrojov (implementácie na báze podnikovej zbernice služieb) z dielne veľkých a stabilných softvérových firiem.

Na druhej strane, v prípade úspešného vývoja vlastnej aplikácie by organizácia získala výhodu priameho dosahu na aplikáciu, čo by mohlo znamenať jednoduchšiu, rýchlejšiu a lacnejšiu údržbu vrátane zapracovávania prípadných nutných zmien do systému.

3.2.3 „Kúpiť alebo vyvinúť?" v kontexte UK

Prejdime teraz od všeobecnej diskusie k našej konkrétnej situácii a pokúsme sa na ňu aplikovať práve uvedené argumenty.

Vzhľadom na dostupnosť vhodných integračných nástrojov a na problémy spojené s vývojom vlastnej aplikácie sa prirodzene zdá byť výhodnejšia alternatíva EAI. Je otázne, či je vôbec v možnostiach univerzity vyvinúť CDO tak, aby sa kvalitou, funkcionalitou aj ostatnými nie funkčnými vlastnosťami vyrovnala niektorému pokročilejšiemu integračnému nástroju dostupnému na trhu. Je to síce reálne, ale vyžadovalo by to značné úsilie a veľmi veľa času. To by znamenalo ďalšie oddialenie termínu nasadenia hotového riešenia do praxe. Nehovoriac o tom, že na univerzite chýbajú dostatočne kvalifikovaní odborní pracovníci, ktorí by dokázali vytvoriť CDO (predovšetkým jej integračnú časť) s parametrami porovnateľnými s komerčnými riešeniami a s použitím najmodernejších technológií. Ak by sme sa teda rozhodli pre alternatívu CDO, tak integračná časť CDO by pravdepodobne neponúkala taký komfort a flexibilitu pri definovaní integračných väzieb ako profesionálny integračný nástroj. Toto sú ďalšie argumenty proti alternatíve CDO.

Preto sa z tohto hľadiska radšej prikláňame k alternatíve EAI a teda k obstaraniu komerčného nástroja na riešenie integrácie. Ako už bolo povedané v úvode tejto kapitoly, je síce pravda, že pri oboch alternatívach je nutné spraviť nejaký vlastný vývoj, ale rozdiel je v rozsahu. V prípade alternatívy EAI, implementovať spoľahlivo CDO bez funkcií integračného nástroja je v prostredí univerzity ešte reálne.

[6] prípadne na veľmi malej skupine ľudí

Ak teda chceme použiť alternatívu EAI, je vhodné sa na tomto mieste zamyslieť nad množstvom implementácie, ktoré budeme musieť vykonať.

- V prvom rade sú to úkony súvisiace s nasadením a nakonfigurovaním samotného integračného nástroja. Sem patrí aj zadefinovanie integračných väzieb na jednotlivé systémy, transformácií údajov a tiež vytvorenie komunikačnej infraštruktúry.
- V druhom kroku je nutné napojiť integrované systémy na zbernicu služieb. Možnosti napojenia sme už uviedli vyššie. Pre každý systém treba individuálne zvážiť a vybrať najvhodnejší spôsob komunikácie so zbernicou. Pri starších nesieťových aplikáciách (vyvinutých v nejakom dnes už nepoužívanom programovacom jazyku) sa nevyhneme vytváraniu vlastných adaptérov, ktoré umožnia ich napojenie do komunikačnej štruktúry. Na druhej strane, dodávatelia integračných nástrojov sú spravidla schopní ponúknuť veľmi širokú paletu už pripravených adaptérov (dokonca aj pre DBASE či Foxpro).
- Ďalej netreba zabudnúť na CDO, ktorá tvorí významný prvok celej architektúry. Návrh a implementácia CDO si vyžaduje netriviálny čas a zo všetkých činností bude pravdepodobne táto trvať najdlhšie.

Presný rozsah prác bude závisieť aj od konkrétneho použitého integračného nástroja a od toho, aké adaptéry bude štandardne poskytovať.

Do úvahy pri rozhodovaní treba ešte zobrať jeden aspekt. Každý vyspelejší integračný nástroj je komplexný systém poskytujúci veľa možností a funkcií. V súčasnej dobe sa integrácia aplikácií na univerzite zameriava hlavne na oblasť zdieľania informácií o osobách medzi aplikáciami. Teda nie veľmi na vzájomné poskytovanie funkcionality resp. služieb. Integračné nástroje, z ktorých jeden plánujeme použiť, implementujú servisne orientovanú architektúru (service-oriented architecture, SOA) prostredníctvom podnikovej zbernice služieb (enterprise service bus, ESB). Otázkou je, či a do akej miery využijeme možnosti takéhoto EAI nástroja. Ak nie, potom by sme pochopiteľne mali radšej zvoliť CDO alternatívu a naimplementovať celé integračné riešenie vlastnými silami.

Skúsme teda zhrnúť, ktoré vlastnosti integračného nástroja dokážeme v našej situácii využiť:

- kompletne vyriešená komunikácia so spoľahlivým doručovaním správ,
- jednoduché a flexibilné definovanie toku údajov medzi systémami,
- rovnako jednoduché definovanie transformácií údajov pri ich prenose medzi systémami,
- inteligentné smerovanie správ (na základe hlavičky, obsahu alebo itinerára),
- dobre riešené monitorovanie, logovanie a správa distribuovaného systému,
- možnosť inkrementálneho nasadzovania (postupné zapájanie jednotlivých systémov),
- decentralizovaná architektúra (čo sa týka komunikácie, nemáme tým pádom kritický bod, na ktorom závisí fungovanie celého systému - „single point of failure“).

Z uvedeného zoznamu vlastností môžeme usúdiť, že využitie integračného nástroja je na UK opodstatnené a jeho nasadenie nebude zbytočným, komplikovaným krokom.

Navyše integrácia aplikácií na UK sa bude v budúcich rokoch rozširovať na ďalšie systémy a očakávame, že výhody integračného nástroja sa postupom času v plnej miere prejavia.

3.2.4 Architektúra

Ďalší rozdiel je trochu technickejšieho charakteru a týka sa výslednej architektúry. Pri alternatíve CDO vzniká silno centralizovaná architektúra typu „hub and spoke“, kde centrum tvorí pochopiteľne CDO. Toto je výhodné z hľadiska centralizovanej správy distribuovaného systému. Naopak nevýhodou je fakt, že v tejto architektúre sa CDO stáva kritickým bodom z pohľadu fungovania celého systému. V prípade zlyhania CDO prestanú fungovať všetky integračné väzby. Táto architektúra taktiež nie je schopná podporovať lokálne integračné domény (nezávislé množiny integrovaných aplikácií). Tieto fakty znamenajú zvýšený nárok na spoľahlivosť a dostupnosť CDO.

V prípade alternatívy EAI s použitím nástroja na báze ESB sa presúvame od centralizovanej architektúry k úplne decentralizovanej. Jednotlivé integrované systémy sa za účelom komunikácie a poskytovania služieb pripájajú k spoločnej virtuálnej zbernici služieb, ktorá je fyzicky decentralizovaná (i keď z logického pohľadu tvorí jeden celok).

3.2.5 Záver

Na základe argumentov uvedených v tejto kapitole si myslíme, že alternatíva EAI je pre UK prínosnejšia hlavne z hľadiska budúcnosti. Na realizáciu súčasných integračných požiadaviek (t.j. zdieľanie informácií o osobách a prípadne ďalšie spomenuté služby a používateľské funkcie) postačuje samotná CDO. Predpokladáme však, že v budúcnosti budú nároky a požiadavky na integračné riešenie väčšie a zložitejšie. Implementáciu týchto požiadaviek zásadným spôsobom uľahčí použitie kvalitného integračného nástroja. Vývoj a dlhodobá prevádzka či údržba alternatívy CDO je ťažšie realizovateľná aj z dôvodu nedostatku zamestnancov kvalifikovaných v oblasti integrácie alebo programovania takéhoto systému.

Pragmatické dôvody viedli k rozhodnutiu, že integrácia na UK bude najprv riešená čisto pomocou centrálnej databázy osôb. Táto okrem centrálnej evidencie osôb a preukazov bude zabezpečovať aj prepojenie určených systémov univerzitného IS. Každý z týchto systémov bude nejakou formou rozhrania komunikovať s CDO. Priama komunikácia medzi aplikáciami bude zakázaná. V blízkej budúcnosti sa počíta so zakúpením integračného nástroja a s prechodom na druhú alternatívu. CDO musí však aj naďalej tvoriť súčasť IS, pretože bude poskytovať používateľom funkcie na prácu s údajmi v IS a ďalšie služby, ktoré nie sú implementované v žiadnom inom systéme v rámci IS UK – hlavne prideľovanie UOČ, operácie s univerzitnými preukazmi a notifikácie o zmene údajov.

4 Požiadavky na CDO

V závere predchádzajúcej kapitoly sme sa rozhodli pre alternatívu CDO s perspektívou prechodu na alternatívu EAI. V súčasnej dobe budeme teda integráciu riešiť pomocou centrálnej databázy osôb, ktorá bude plniť dve hlavné funkcie – uchovávať údaje o osobách a komunikovať s ostatnými systémami za účelom výmeny údajov.

Špecifikácia CDO spolu s prípadmi použitia (use case model) sa nachádza v samostatnom dokumente, ktorý tvorí prílohu A tejto práce. V prílohe B popisujeme detailne rozhrania medzi CDO a jednotlivými integrovanými systémami.

V tejto kapitole sa budeme skôr zaoberať tými požiadavkami, ktoré zásadne ovplyvňujú návrh a architektúru CDO. Kvôli prehľadu však uvedieme aj zhrnutie ostatných funkčných požiadaviek na CDO:

- uchovávanie údajov,
- prenos údajov zo zdrojových systémov a ich poskytovanie cieľovým systémom,
- generovanie univerzitného osobného čísla,
- správa preukazov (najmä aktivácia a deaktivácia preukazov),
- prezeranie uložených údajov používateľmi,
- notifikácie o zmenách údajov.

4.1 Používateľské funkcie

Hlavné používateľské funkcie, ktoré má CDO poskytovať, sú dve (pozri aj prílohu A – „Špecifikácia CDO“):

- prezeranie údajov,
- notifikácia o vzniku, zmene alebo zániku údajov.

CDO by mala používateľom umožniť prezeranie údajov uložených v databáze a to v závislosti na používateľových oprávneniach. Každá osoba by mala byť oprávnená prezerať si všetky údaje, ktoré sú o nej v CDO evidované. Používatelia s vyššími oprávneniami budú môcť vidieť aj údaje iných osôb. Napríklad referentka študijného oddelenia môže vidieť študentské záznamy všetky študentov príslušnej fakulty.

CDO musí vedieť riadiť prístup k uloženým údajom na úrovni konkrétnych záznamov a jednotlivých atribútov. Pre každý atribút o osobe musíme mať možnosť definovať, kto a za akých podmienok môže hodnoty tohto atribútu vidieť.

Druhou používateľskou funkciou je notifikácia. Používatelia, ktorí na to budú oprávnení, budú môcť v CDO zaregistrovať žiadosti o notifikáciu. Tieto notifikácie sa môžu týkať vzniku, zmeny alebo zániku údajov. Ak nastane príslušná udalosť, CDO pošle notifikačný email na príslušnú definovanú adresu uvedenú v registračnej žiadosti.

Čo sa týka notifikácie o vzniku údajov, príkladom môže byť správca počítačovej siete na fakulte, ktorý by chcel byť informovaný o vzniku nového (študentského alebo zamestnaneckého) vzťahu na príslušnej fakulte, aby následne mohol napríklad vytvoriť na serveri používateľské konto pre túto novú osobu. V tomto prípade ide teda o notifikáciu viazanú na konkrétnu súčasť UK a nie na konkrétnu osobu.

Notifikácia o zmene údajov sa musí vzťahovať na konkrétnu osobu. Používateľ si môže zvoliť, či chce byť informovaný o zmene konkrétneho vzťahu danej osoby alebo ho zaujíma zmena ľubovoľného vzťahu tejto osoby. Pri tomto type notifikácie by malo byť možné určiť aj konkrétne atribúty, ktorých zmena vyvolá poslanie notifikácie používateľovi. Tu treba brať do úvahy autorizáciu – používateľ si môže zvoliť len tie atribúty, ktoré je oprávnený prezerať.

V prípade notifikácie o zániku údajov, používateľ určí konkrétnu osobu a konkrétny vzťah, ktorého zánik spôsobí zaslanie notifikácie používateľovi. Ďalšou možnosťou je notifikácia o zániku akéhokoľvek vzťahu prislúchajúceho ku konkrétnej súčasti UK. Je to analogický prípad ako pri notifikácii o vzniku údajov. Príkladom oblasti, kde sa dá využiť notifikácia o zániku pracovného vzťahu, sú pracovné skupiny. Jeden zamestnanec môže byť súčasne členom viacerých pracovných skupín, ale nemusí byť v bližšom kontakte s vedúcimi každej skupiny. Vedúci pracovnej skupiny by chcel byť notifikovaný, keď niektorý z jeho členov prestane byť zamestnancom fakulty (resp. univerzity).

4.2 Flexibilita

Najdôležitejším zámerom pri návrhu CDO je flexibilita a to hneď v niekoľkých oblastiach:

- Na dátovej úrovni požadujeme flexibilitu vzhľadom k uchovávaným údajom. Hlavnou úlohou CDO ako stavového elementu v rámci integrácie aplikácií je uchovávať vo svojej databáze atribúty týkajúce sa osôb. Integrácia je relatívne dlhodobý proces, do ktorého budú postupne zapájané ďalšie systémy a prípadne niektoré systémy budú nahradené novými. Z tohto dôvodu očakávame, že množina atribútov o osobe uchovávaných v CDO nebude stála. Potrebujeme teda predovšetkým možnosť jednoducho pridávať nové atribúty do databázy evidovaných osôb tak, aby to nemalo zásadný dopad na existujúcu funkčnosť CDO a aby nebolo nutné prepisovať veľké časti zdrojového kódu systému.

- V závere predchádzajúcej kapitoly sme konštatovali, že v prvej fáze integrácie na UK bude CDO vykonávať aj integračné funkcie – komunikácia so systémami za účelom prenosu údajov medzi nimi s vykonaním potrebných transformácií. V budúcnosti očakávame, že tieto funkcie na seba preberie integračný nástroj. Z tohto vyplývajú hneď dve ďalšie požiadavky na flexibilitu.

  Po prvé, požadujeme flexibilitu vzhľadom na definovanie (prípadne modifikáciu) integračných rozhraní CDO. Ako už bolo povedané, integrácia aplikácií je kontinuálny proces a zmeny sú nevyhnutné. „Záber“ integrácie sa bude postupne rozširovať, a preto potrebujeme, aby definícia rozhraní v integračnej časti CDO bola čo najviac pružná a jednoduchá. To znamená, že pridanie nového rozhrania alebo modifikácia existujúceho by mala byť realizovateľná bez príliš veľkých zásahov do kódu.

  Po druhé, integračné funkcie musia byť čo možno najviac nezávislé na zvyšku CDO. Tým sa zaručí, že nasadenie integračného nástroja v budúcnosti bude mať minimálny dopad na CDO a množstvo potrebných úprav nebude veľké.

4.3 Prenos údajov a sledovanie zmien

Podstatnou časťou funkcionality CDO je prijímanie a poskytovanie údajov a taktiež ďalšie služby požadované buď systémami v rámci IS UK alebo samotnými používateľmi.

Z toho možno odvodiť ďalšie požiadavky na CDO relevantné pri navrhovaní štruktúry systému. Čo sa týka evidovania zmien a histórie údajov, sú to hlavne tieto:

- Kvôli efektívnosti alebo spôsobu spracovania vyžadujú niektoré systémy inkrementálne exporty z CDO. To znamená, že CDO týmto systémom neposiela vždy kompletné údaje, ale len zmeny, ktoré sa udiali od posledného posielania údajov. Tento prístup môže byť navyše kombinovaný s mechanizmom komunikácie „request-reply“. Pri tomto spôsobe komunikácie CDO posiela externému systému údaje len vtedy, ak od neho dostane požiadavku. To predstavuje nutnosť uchovávať inkrementy až do doby, kedy príslušný systém pošle požiadavku na zaslanie údajov.
- Špecifickou požiadavkou niektorých systémov je zasielanie historických údajov. Teda okrem práve aktuálnych dát by v exporte pre príslušný systém mali byť aj dáta, ktoré boli platné v minulosti. Konkrétnym príkladom môže byť systém virtuálnej knižnice (VIKUK), ktorý okrem aktuálnych študentských vzťahov požaduje aj zasielanie vzťahov aktuálnych v predchádzajúcom akademickom roku.
- Požadovanou používateľskou funkciou je notifikácia o zmenách. Jednotliví používatelia systému by mali mať možnosť zaregistrovať si v CDO svoje kritériá na notifikáciu. Teda v podstate si určia atribúty o osobe, na ktorých zmenu chcú byť upozornení. CDO pri zmene niektorého z týchto atribútov zašle príslušným registrovaným používateľom notifikačný e-mail.

4.4 Bezpečnosť

CDO umožňuje prenos a prácu s osobnými údajmi študentov a zamestnancov univerzity. Keďže ide o osobné údaje, ktoré podliehajú ochrane, je nutné zabezpečiť, aby sa k týmto údajom nedostali neoprávnené osoby. Údaje je potrebné chrániť na dvoch úrovniach:

- Ochrana údajov pri samotnom prenose údajov, a to ako pri prenose medzi CDO a inými systémami tak aj pri prenose medzi CDO a (webovým) používateľským rozhraním.
- Obmedzenie prístupu jednotlivých používateľov systému k údajom o osobách. Granularita týchto obmedzení je na úrovni konkrétnych atribútov. Každý používateľ teda môže pracovať (prezerať, modifikovať) len s tými atribútmi, na ktoré má oprávnenie.

Čo sa týka oprávnení, je potrebné obmedzovať aj akcie (operácie), ktoré môžu jednotliví používatelia vykonávať. Ide predovšetkým o operácie ako aktivácia či deaktivácia univerzitného preukazu, riešenie konfliktov pri generovaní UOČ a podobne.

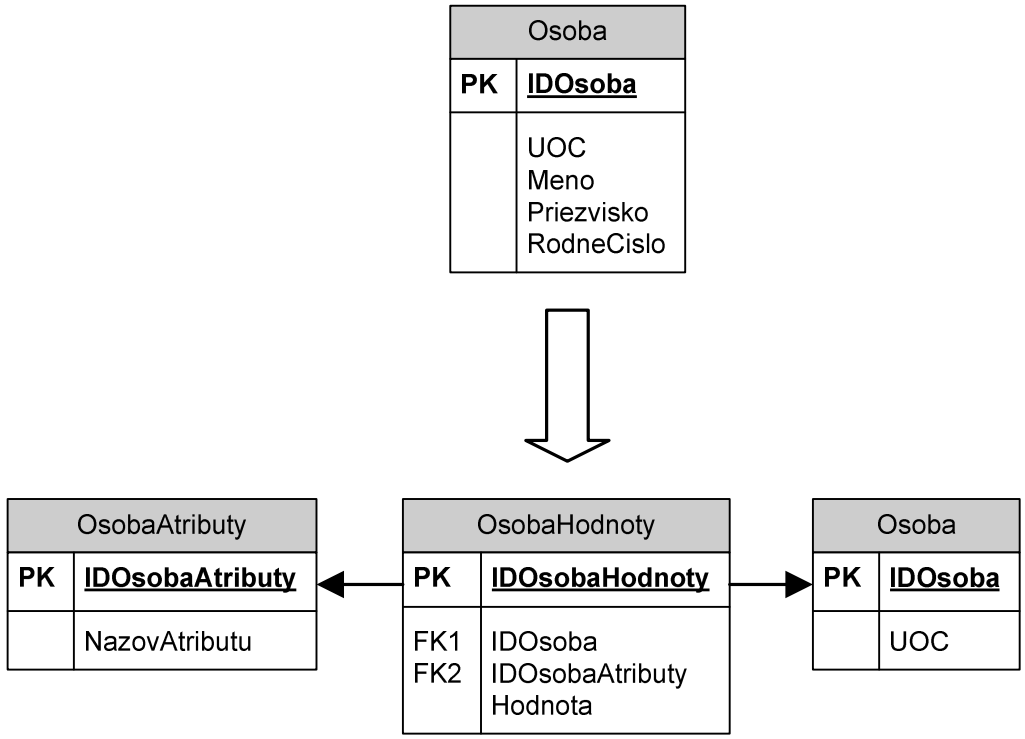
5 Návrh CDO

5.1 Flexibilita

Najzaujímavejšou otázkou je riešenie flexibility, čo sa týka evidovaných atribútov o osobách. Prvou alternatívou je zaviesť podporu pre túto flexibilitu už na úrovni databázového modelu. To znamená navrhnúť štruktúru tabuliek v databáze tak, aby pridávanie či odoberanie atribútov bolo možné bez zásahu do samotnej databázovej schémy.

Jeden z možných spôsobov ako to dosiahnuť je využiť abstrakciu atribútov (princíp „atribút ako entita“). Inšpiráciu sme čerpali z [12], v ktorej sú popísané často používané analytické dátové vzory. Pri abstrakcii atribútov sa atribút stáva samostatnou dátovou entitou. Umožňuje nám to definovať nové atribúty v čase vykonávania programu, avšak priamočiara implementácia býva značne neefektívna.

Ilustrujeme aplikáciu princípu abstrakcie atribútov na príklade tabuľky *Osoba*, v ktorej by sme chceli mať uložené nejaké atribúty o osobách.



Obrázok 5-1 Abstrakcia atribútov na úrovni databázy

V tabuľke *Osoba* zostane len niekoľko „pevných“ atribútov. Všetky ostatné atribúty budú uložené v tabuľkách *OsobaAtributy* a *OsobaHodnoty*. Prvá menovaná tabuľka obsahuje (dynamický) zoznam atribútov a druhá hodnoty týchto atribútov pre každú konkrétnu osobu.

Takéto riešenie je dosť flexibilné, umožňuje pridávať či odoberať atribúty bez nutnosti zasahovať do tried dátového modelu resp. do ich mapovania na databázu. Dopad zmeny na zvyšok aplikačného kódu je tiež malý, za predpokladu, že pri návrhu boli použité základné techniky (vzory) na dosiahnutie ľahkej modifikovateľnosti a znovupoužiteľnosti. Avšak riešenie prináša so sebou aj isté nevýhody:

- Na úrovni databázy strácame informáciu o dátových typoch jednotlivých atribútov.
- Zbytočne zložité riešenie, keďže hlavnou požiadavkou je možnosť pridávať nové atribúty a tento prístup rovnako dobre podporuje aj odoberanie či zmenu atribútov. To všetko dokonca aj za behu systému, čo tiež v CDO pravdepodobne nevyužijeme.
- Zložitý dátový model, a teda náročnejšia práca s takýmto modelom na aplikačnej úrovni. Takisto, zložitejšie SQL príkazy a z toho vyplývajúca vyššia časová náročnosť na ich vykonanie.

Z dôvodu uvedených negatív sme sa rozhodli nepoužiť túto alternatívu, t.j. flexibilitu atribútov zabezpečíme na úrovni aplikačnej logiky. Databázová štruktúra bude pevná, v zmysle, že nebudeme môcť dynamicky meniť atribúty tabuliek. Na druhej strane samotné pridanie ďalšieho atribútu do tabuľky nie je veľký problém. Treba však zaistiť, aby si takáto zmena v databáze nevyžiadala zásahy do existujúceho kódu resp. aby tieto zásahy boli minimálne. Toto možno dosiahnuť jednak celkovou architektúrou systému a tiež použitím návrhových vzorov známych ako vzory „Gang of Four“ ([14]).

Čo sa týka nezávislosti integračných funkcií od zvyšku CDO, zabezpečiť ju môžeme izoláciou týchto funkcií do samostatného modulu s jasne definovaným rozhraním. V celkovej architektúre CDO bude tento integračný modul súčasťou klientskej (resp. komunikačnej) vrstvy a eventuálne môže používať ostatné spoločné funkcie CDO. Flexibilitu pri definovaní rozhraní dosiahneme jednak celkovou architektúrou systému, použitím servisnej vrstvy a vhodným návrhom vnútornej štruktúry integračného modulu. Podrobnejšie sa architektúre CDO venujeme v kapitole 5.4.

5.2 Evidencia a sledovanie zmien

Ďalšia skupina požiadaviek uvedených vyššie súvisí so zmenami v evidovaných údajoch. Realizácia týchto požiadaviek si vyžaduje jednoduchú a efektívnu implementáciu sledovania zmien hodnôt atribútov o osobách.

5.2.1 Detekcia zmien

Ak chceme riešiť sledovanie zmien v dátach, musíme najprv rozhodnúť, ako sa dajú tieto zmeny detegovať. Zmeny v údajoch sa môžu udiať dvomi cestami.

Jednak pri importovaní novoprijatých údajov do CDO, keďže údaje sa do CDO dostávajú hlavne prenosom zo zdrojových systémov, pričom tieto systémy v súčasnosti posielajú vždy kompletné aktuálne údaje.

Taktiež sa počíta aj s možnosťou modifikovania niektorých atribútov priamo používateľmi cez používateľské rozhranie. Ide napríklad o zadávanie kontaktných údajov, ktoré by osoba mala o sebe vypĺňať samostatne.

Ak by sme nevyžadovali evidenciu zmien, mohli by sme importovanie do CDO jednoducho vyriešiť tak, že najprv zmažeme existujúce záznamy a následne tabuľku naplníme novými údajmi. Pre naše účely by sme mohli tento prístup modifikovať tak, že najprv vytvoríme kópiu starých údajov a potom staré údaje nahradíme novými. Ďalším krokom by bolo porovnanie starých a nových údajov v databáze, čím by sme získali zoznam zmien. Takéto triviálne riešenie však nevyhovuje, najmä z hľadiska efektivity spracovania a výkonnosti.

Efektívnejšou alternatívou je implementovať porovnávanie existujúcich (aktuálne v CDO) a nových (prijatých od externého systému) údajov „in situ“. Obe množiny dát sú v podstate zoznamy pozostávajúce z jednotlivých záznamov. Sekvenčným prechádzaním možno tieto dva zoznamy porovnať a výstupom bude rozdelenie záznamov do troch skupín – nové, (existujúce ale) modifikované a zmazané záznamy. Štvrtá skupina, t.j. existujúce ale nemodifikované záznamy, nie je z pohľadu riešenia zmien zaujímavá. Samotné porovnávanie dvoch záznamov na zhodu budeme implementovať na úrovni porovnávania hodnôt jednotlivých atribútov.

Z hľadiska výkonnosti tu možno vidieť komplikáciu pri sekvenčnom čítaní údajov z databázy. Problém to však nie je, pretože môžeme využiť techniky hromadného spracovania údajov podporované technológiou, ktorú plánujeme použiť na úrovni dátovej vrstvy (Hibernate). V pozadí sa využívajú napríklad kurzory na strane SQL servera (server-side cursors). Takisto možno spracovávať údaje po menších dávkach a tým optimalizovať veľkosť použitej operačnej pamäte.

Poslednou alternatívou, ktorá by uľahčila spracovanie údajov na strane CDO, je delegovať funkciu detekcie zmien v údajoch na zdrojové systémy. Pri posielaní údajov do CDO by zdrojový systém pri každom zázname označil, či ide o nový alebo modifikovaný záznam. Toto by samozrejme znamenalo väčší dopad na zdrojové systémy, významnejšie zásahy do kódu a prípadne aj do lokálnej databázy týchto systémov. Na druhej strane si treba uvedomiť, že všetky zdrojové systémy sú databázové aplikácie primárne slúžiace na evidenciu nejakých typov údajov o osobách (a na prácu s týmito údajmi). Používatelia v týchto systémoch priamo vyvolávajú funkcie vkladania nového záznamu resp. modifikácie existujúceho, čiže detekcia týchto udalostí by nebola veľmi zložitá.

5.2.2 Uchovávanie zmien

Keď už vieme zistiť, ktoré záznamy sa zmenili, treba ešte vyriešiť spôsob pamätania si týchto zmien. Najdôležitejšou požiadavkou je podpora inkrementálnych exportov z CDO do externých systémov. Pri týchto treba rozlíšiť dve formy komunikácie.

- Prvú skupinu tvoria systémy, ktoré potrebujú dostávať inkrementálne údaje „v reálnom čase“. Teda hneď ako sa v CDO udeje zmena, mali by sa zmenené údaje poslať príslušným systémom.
- Druhú skupinu tvoria systémy, ktoré chcú dostávať údaje z CDO na požiadanie, t.j. už spomenutý „request-reply“ model komunikácie. Čiže aj keď sa v CDO udeje niekoľko zmien v dátach, tieto budú poslané príslušnému systému až keď o to požiada.

Čo sa týka prvej skupiny, zrejme najjednoduchšie je posielať týmto systémom dáta už počas spracovania importných údajov zo zdrojových systémov prípadne na jeho konci, keď už budeme mať vytvorený zoznam modifikovaných (resp. nových) záznamov.

Pri systémoch z druhej skupiny je situácia zložitejšia, pretože si vyžaduje dočasne uchovať zmeny, kým nepríde požiadavka na zaslanie údajov. Jednotlivé inkrementy by sme mohli ukladať buď v externých súboroch na disku alebo v databáze s tým, že aplikačná logika spracujúca požiadavku by vedela, odkiaľ získať požadované inkrementálne údaje.

V prípade potreby uchovania viacerých inkrementov pre jeden konkrétny systém je treba zvážiť, či budú tieto inkrementy oddelené alebo sa budú spájať do jedného, ktorý sa pošle ako odpoveď pri najbližšej požiadavke. Prikláňame sa k druhej alternatíve, pretože je prehľadnejšia. Pri implementovaní prvej alternatívy by bolo otázne, čo spraviť, keď príde

požiadavka od systému a v jemu prislúchajúcom zásobníku inkrementov čaká päť správ. Poslať všetky alebo len jednu (tú najstaršiu)?

Implementácia uchovávania zmien použitím spomenutého „súborového prístupu“ má isté nedostatky:

- Práca so súbormi predstavuje v pohľadu aplikácie réžiu navyše, ktorú možno minimalizovať použitím databázy, ktorá predstavuje robustnejšie riešenie a ponúka väčšie možnosti.
- Spájanie dvoch inkrementov je zložitejšie pri súboroch, keďže záznamy v nich nie sú explicitne usporiadané podľa nejakého kľúča. Mohli by sme záznamy ukladať utriedené, ale to opäť znamená väčšie úsilie a dlhší čas spracovania.

Je teda lepšie pamätať si zmeny v databáze, kde potom môžeme využiť techniku časových pečiatok (timestamps) alebo verzií. Pri oboch možnostiach potrebujeme mať v databáze tabuľku externých systémov, ktorým bude CDO poskytovať údaje. Okrem základných atribútov (ako napríklad identifikátor, názov, príznak, či systém požaduje prenos v reálnom čase, ...) by táto tabuľka obsahovala jeden atribút označujúci buď čas posledného exportu alebo aktuálne číslo verzie.

Popíšeme najprv alternatívu využívajúcu časové pečiatky. Tabuľke, ktorej zmeny chceme sledovať, pridáme atribút *PoslednaZmena* hovoriaci o čase poslednej zmeny záznamu. Vždy keď urobíme zmenu nejakého riadku tejto tabuľky, nastavíme hodnotu atribútu *PoslednaZmena* prislúchajúceho tomuto riadku na aktuálny čas. V tabuľke externých systémov budeme mať pre každý systém uložený čas posledného exportu údajov do príslušného systému. Tento čas budeme aktualizovať po každom exporte. Export inkrementálnych údajov potom bude obsahovať všetky záznamy, ktorých časová pečiatka má väčšiu hodnotu ako čas posledného exportu do príslušného systému. Teda sú to presne tie záznamy, ktoré sa zmenili od posledného exportu do príslušného systému.

Druhá alternatíva je podobná, ale využíva verzie namiesto časových pečiatok. Predpokladáme, že na nejakom mieste (najlepšie priamo v databáze) máme uložené globálne aktuálne číslo verzie G_v. Vždy keď zmeníme riadok tabuľky (ktorej zmeny chceme sledovať), tak nastavíme jeho prislúchajúce číslo verzie na aktuálnu hodnotu G_v. V tabuľke externých systémov je pre každý systém uvedené maximálne číslo verzie v čase posledného exportu M_v. Do exportu sa potom dostanú záznamy s číslom verzie vyšším než má príslušný systém. Vždy po exporte údajov do nejakého systému sa číslo verzie tohto systému M_v nastaví na maximálnu hodnotu (spomedzi všetkých aktuálnych čísiel verzií priradených záznamom) a globálne číslo aktuálnej verzie G_v na hodnotu o jedna vyššiu než je toto maximum, t.j. $M_v + 1$. Jednoduchším a časovo menej náročným spôsobom aktualizácie čísiel M_v a G_v je po každom exporte inkrementovať G_v o 1 a M_v nastaviť na túto novú hodnotu G_v. V tomto prípade by sme potom do exportu zahŕňali všetky záznamy s číslom verzie väčším alebo rovným ako M_v.

Obe riešenia umožňujú evidovať informáciu o tom, ktoré záznamy sa zmenili a podporujú uchovávanie týchto inkrementálnych zmien pre individuálne systémy. Prvá alternatíva navyše zaznamenáva aj čas zmeny a je celkove menej náročná na spravovanie resp. implementáciu – netreba sa starať o aktualizáciu globálneho čísla verzie G_v. Nevýhodou časových pečiatok sú eventuálne problémy pri nesprávne nastavenom čase na SQL serveri resp. skôr pri jeho prestavovaní na správnu hodnotu – napr. pri synchronizácii s NTP serverom.

Dôležitým aspektom problému uchovávania informácie o zmene údajov je granularita. Obe prezentované riešenia uchovávajú informáciu o zmene na úrovni

záznamov v tabuľke. To je pre funkciu inkrementálnych exportov z CDO postačujúce, avšak požiadavka notifikácie o zmenách atribútov o danej osobe implikuje nutnosť zisťovať zmeny na úrovni jednotlivých atribútov. Na druhej strane uchovávanie týchto zmien nie je potrebné, keďže notifikácia je zo svojej podstaty služba vyžadujúca implementáciu v reálnom čase. CDO teda posiela notifikačné správy hneď ako zistí zmenu hodnoty sledovaného atribútu. Notifikáciami sa podrobnejšie zaoberáme v podkapitole 5.2.4.

5.2.3 História

Doposiaľ sme sa zaoberali problémom ako zistiť, či nastala zmena v údajoch a ako túto skutočnosť uchovať. Nezaujímali nás pritom konkrétne hodnoty atribútov a vždy sme v databáze ukladali len aktuálne hodnoty a príznak, či bol príslušný záznam zmenený. Tento koncept sa niekedy nazýva aj „snapshot table“ [13]. Hodnoty záznamov v tabuľke sú aktuálnym obrazom a nikde nemáme uložené temporálne dáta, t.j. hodnoty, ktoré boli platné v nejakom čase v minulosti.

V súčasnej dobe (na základe súčasných požiadaviek) nie je potrebné ukladať temporálne údaje v plnom rozsahu. Jedinou požiadavkou je schopnosť pamätať si a následne poskytovať informácie o bývalých (ukončených) študentských vzťahoch. Toto si však samo o sebe nevyžaduje ukladať históriu zmien jednotlivých atribútov (temporálne údaje), postačuje na to koncept evidovania akademického roku: Každý záznam v tabuľke študentských vzťahov bude mať priradený akademický rok, v ktorom bol resp. je príslušný vzťah platný.

5.2.4 Notifikácia o zmenách

Zatiaľ jedinou požadovanou používateľskou funkciou, ktorá si vyžaduje sledovanie zmien v údajoch, je notifikácia o zmenách. Túto funkciu sme popísali vyššie v časti 4.1 venovanej požiadavkám, detailne je táto funkcia zachytená v prílohe A.

CDO bude teda evidovať zoznam žiadostí o notifikáciu, pričom ku každej budú uvedené minimálne nasledovné informácie:

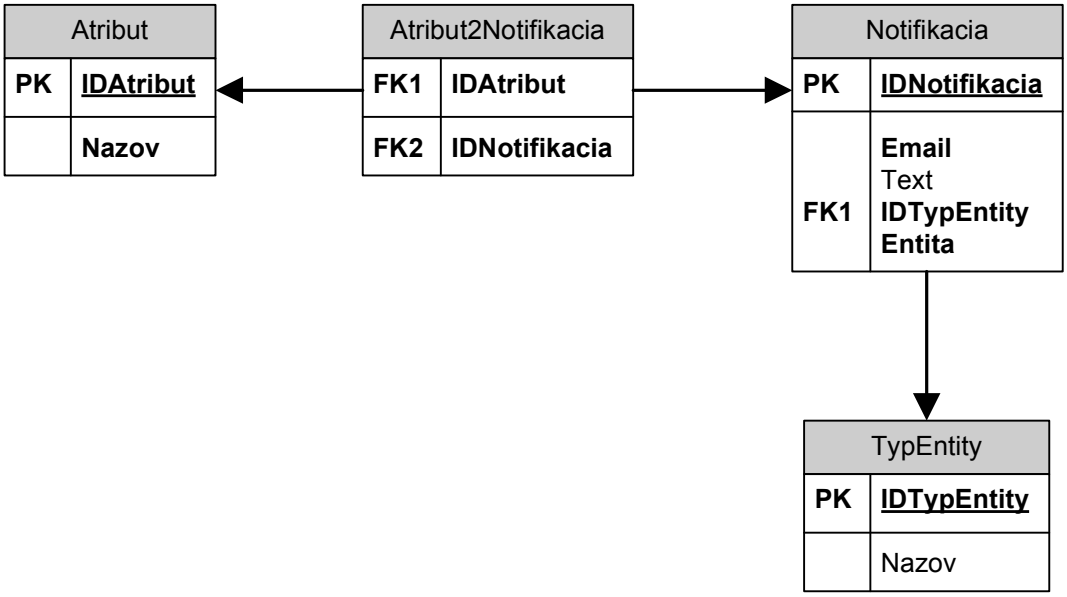
- identifikácia vlastníka žiadosti,
- jedinečný identifikátor žiadosti,
- emailová adresa, kam sa bude posielať prípadná notifikačná správa,
- text, ktorý bude uvedený v notifikačnej správe,
- určenie údajov, ktorých zmenu je potrebné sledovať.

Samotné evidovanie tohto zoznamu nepredstavuje žiadny problém. Zaujímavým problémom je spôsob pamätania si zoznamu sledovaných atribútov pre každú žiadosť o notifikáciu (v prípade notifikácie o zmene údajov). Pre každý z týchto atribútov je treba evidovať aj fakt, ku ktorej dátovej entite a ku ktorým záznamom sa vzťahuje. Zároveň, dátová štruktúra obsahujúca tieto údaje nesmie byť príliš zložitá, aby sme pri zmene hodnoty niektorého atribútu o osobe dokázali veľmi rýchlo zistiť, či existujú nejaké žiadosti o notifikáciu vzťahujúce sa k príslušnému atribútu.

Kvôli zníženiu zložitosti problému a taktiež z pragmatických dôvodov by sme mohli predpokladať, že jedna žiadosť o notifikáciu sa bude týkať len jedného typu údajov (študentské vzťahy, zamestnanecké vzťahy, preukazy, ...). V prípade, že by sme chceli

sledovať všetky zmeny údajov osoby, ktorá má v CDO viac druhov údajov, bolo by nutné vytvoriť samostatnú žiadosť pre každý druh. Takéto zjednodušenie je prijateľné, keď si uvedomíme, že v súčasnosti a v dohľadnej budúcnosti sú požadované len notifikácie o jednom druhu údajov - spomínané zamestnanecké vzťahy.

Uvedené úvahy a istá podobnosť s modifikovaným návrhovým vzorom „Observer“ [14] nás privádzajú k nasledovnému databázovému návrhu.



Obrázok 5-2 Databázový návrh na podporu notifikácií o zmene

Tabuľky *Atribut* a *Atribut2Notifikacia* nám umožnia jednoducho a rýchlo zistiť, či sú pre nejaký atribút registrované nejaké notifikácie. V tabuľke *Notifikacia* sú uložené dôležité atribúty o každej registrovanej žiadosti o notifikáciu. *TypEntity* predstavuje druh sledovaného záznamu, t.j., napr. „zamestnanecký vzťah“ alebo „študentský vzťah“. Atribút *Entita* obsahuje identifikátor konkrétneho záznamu, ktorého sa príslušná notifikácia týka.

Problém s atribútom *Entita* vzniká v prípade, že niekto bude požadovať sledovanie všetkých entít príslušného typu. Toto môžeme vyriešiť v princípe dvoma spôsobmi. Buď použijeme nejakú špeciálnu hodnotu atribútu *Entita* na indikáciu takéhoto prípadu. To však skomplikuje prácu s touto štruktúrou, hlavne databázové dotazy. Alebo môžeme pri spracovaní takejto požiadavky na notifikáciu vyhľadať všetky entity príslušného typu pre danú osobu a vytvoriť jednu notifikáciu pre každú nájdenú entitu. Nevýhodou tohto prístupu je fakt, že používateľ si zadá jednu žiadosť a eventuálne mu ich vznikne niekoľko. Toto by sa dalo čiastočne eliminovať tým, že takto „automaticky“ vzniknuté žiadosti by prislúchali k jednej notifikácii. Napr. by mali rovnaký identifikátor.

Celkove je tento návrh použiteľný, ale má jeden nedostatok. Je zrejmé, že najčastejším databázovým dotazom v oblasti sledovania zmien bude otázka „Existuje pre danú entitu a daný atribút nejaká registrovaná žiadosť o notifikáciu?“. Alebo ešte obmena v podobe „Aké registrované žiadosti o notifikáciu sa vzťahujú k danej entite a danej podmnožine jej atribútov?“. Takéto dotazy si v navrhnutej štruktúre vyžiadajú spojenie všetkých štyroch tabuliek, čo bude časovo náročné najmä pri vyššej frekvencii týchto dotazov. Počet spojení možno zredukovať o jeden zakomponovaním typu entity do zoznamu atribútov. Teda ten istý atribút použitý v rôznych tabuľkách (t.j., v rôznych druhoch údajov) by sme považovali za rôzny.

Ide v podstate o klasický problém pri normalizácii databázovej štruktúry – treba nájsť kompromis medzi úrovňou normalizácie tabuliek a efektívnosťou dotazov nad nimi.

Zatiaľ sme sa zaoberali len notifikáciami o zmene údajov. Do úvahy však treba brať aj ďalšie dva typy notifikácií. Toto situáciu ešte viac skomplikuje a je netriviálne nájsť jednotné riešenie podporujúce efektívne všetky typy notifikácií. Detailné riešenie problému notifikácií nechávame na budúcnosť, najmä z dôvodu rozsahu a nižšej priority tohto problému.

5.3 Bezpečnosť

Čo sa týka ochrany údajov pri ich prenose, bude nevyhnutné použiť šifrovanie. V tejto oblasti sú vyvinuté dostatočne vyspelé technológie a tieto plánujeme zakomponovať aj do implementácie CDO a komunikácie. Využijeme hlavne štandardné SSL, ktoré je možné použiť ako nadstavbu nad mnohými komunikačnými protokolmi.

Webové používateľské rozhranie bude komunikovať protokolom HTTPS. Zabezpečenie komunikácie CDO a ostatných zdrojových či cieľových systémov je individuálne podľa možností jednotlivých systémov. Systémy schopné prenášať údaje prostredníctvom JMS (Java Message System) nepredstavujú z hľadiska bezpečnosti prenosu problém, keďže SSL dokážeme jednoducho použiť pri väčšine implementácií JMS. Pri iných druhoch prenosu (FTP, zdieľaný adresár, email) treba použiť na zabezpečenie iné riešenie. To je však možné až po dohode so správcami príslušných externých systémov a zvážení reálnych možností týchto systémov.

Oveľa zložitejším problémom je autorizácia používateľov pri práci s CDO a pri prezeraní údajov. Zdôraznime, že tento problém sa týka len používateľov, ktorí po autentifikácii môžu vykonávať nejaké operácie nad údajmi uloženými v CDO. Externé systémy komunikujúce s CDO nepovažujeme za používateľov. Ich autentifikáciu ako aj autorizáciu riešime na úrovni prístupu ku komunikačnému rozhraniu a v prípade potreby dodatočnou validáciou prenášaných údajov.

Zhrňme najprv konkrétne požiadavky, ktoré musí náš model pre autorizáciu spĺňať.

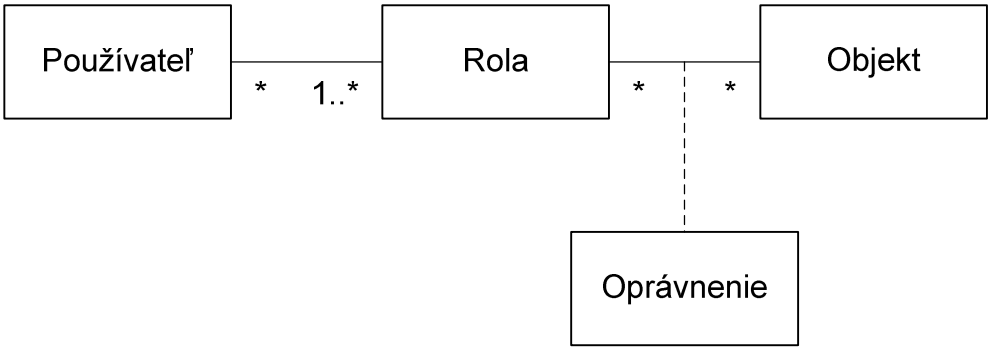
- Každému používateľovi musíme byť schopní priradiť množinu oprávnení, ktoré určujú, aké operácie môže vykonávať. Priradenie oprávnení používateľom musí byť jednoducho zmeniteľné.
- Pre každého používateľa je nutné jednoznačne určiť, ktoré atribúty o osobe si môže prezerať resp. s nimi inak pracovať – napr. nastaviť si na niektoré z nich notifikáciu o zmene. Uvažujeme aj možnosť modifikácie údajov, ktoré budú primárne uchovávané v CDO. Sú to napr. kontaktné údaje, ktoré by si každá osoba mala aktualizovať sama.

K takto formulovaným požiadavkám je treba ešte dodať, že v nich nie je zachytený spôsob vyhodnocovania povolených oprávnení v konkrétnom prípade autorizácie. Uvedieme pre ilustráciu krátky príklad. Majme študenta, ktorý súčasne študuje na dvoch fakultách – napr. FMFI a PriF. Referentka na študijnom oddelení FMFI si prezerá údaje o tomto študentovi. Mala by vidieť všetky atribúty jeho študentského vzťahu na FMFI, ale len niektoré atribúty jeho vzťahu na PriF. Teda v niektorých prípadoch na určenie atribútov, ktoré je používateľ autorizovaný vidieť, je potrebná ešte dodatočná informácia a aplikovanie dodatočných autorizačných kritérií.

Zaužívaným základným prístupom k riešeniu autorizácie prístupu je zoskupovanie používateľov do rol (skupín) podľa ich oprávnení. Každý používateľ môže mať

priradených viacero rol. Následne priradenie oprávnení bude na úrovni rol a nie jednotlivých používateľov. Je to výhodné, keďže predpokladáme existenciu skupín používateľov, ktorí majú rovnaké nastavenie oprávnení (študenti, referentky študijných oddelení, VoIP operátori, ...). Zavedenie rol je teda prínosom minimálne z hľadiska jednoduchosti administrácie. Z príkladu uvedeného vyššie vyplýva, že rola môže byť viazaná ku konkrétnej súčasti UK. To znamená, že na vyhodnotenie povolených oprávnení používateľa v príslušnej roli musíme brať do úvahy aj súčasť UK, ku ktorej sa rola viaže.

Nasledujúci obrázok ukazuje všeobecné riešenie priraďovania oprávnení z koncepčného hľadiska.



Obrázok 5-3 Koncepčný návrh autorizácie

Každý používateľ patrí do jednej alebo viacerých rol. Medzi rolou a nejakým objektom (ku ktorému chceme riadiť prístup) je definovaná množina oprávnení. Objektmi v prípade prezerania a práce s údajmi sú konkrétne atribúty o osobách. V prípade operácií sú objektmi jednotlivé záznamy o osobách. Napr. oprávnenie aktivovať preukaz sa nevzťahuje na žiadny konkrétny atribút o osobe. Zjednotiť dve vyššie menované požiadavky nie je triviálne, keďže každá z nich sa vzťahuje na iný typ objektu.

Na aplikačnej úrovni je vo všeobecnosti ľahké popísať fungovanie autorizácie. Algoritmus dostane na vstupe používateľa (resp. proces, ktorý koná v mene príslušného používateľa) a objekt, ku ktorému žiada prístup. Na výstupe algoritmu je potom množina povolených oprávnení. Algoritmus vlastne počíta prvky matice označovanej ako matica na riadenie prístupu (access control matrix) [16].

Uložiť túto maticu v databáze nie je prakticky možné z dvoch príčin. Po prvé, ako sme už povedali, objekty, ku ktorým riadime prístup, sú dvoch rôznych typov. Po druhé, aj keby sme tieto dva typy zjednotili, prístupová matica by mala veľmi veľké rozmery a bolo by značne neefektívne ju celú uložiť do databázy, nehovoriac o ďalšom spravovaní takejto štruktúry. Táto štruktúra by pre každého používateľa (resp. pre každú rolu) musela obsahovať referencie na všetky konkrétne objekty (záznamy o osobách resp. ich atribúty), ku ktorým má mať nejakú formu prístupu. Potom by sme napr. po každom importe údajov z nejakého zdrojového systému museli prepočítavať túto maticu a aktualizovať hodnoty jej prvkov v databáze.

V snahe vyhnúť sa príliš častému prepočítavaniu prvkov prístupovej matice a tiež z hľadiska efektívnosti celej autorizácie budeme v databáze na podporu procesu autorizácie ukladať len „statické“ informácie – také, ktoré sa priamo neviažu na konkrétne záznamy o osobách, preukazoch alebo iných dátových entitách, ktoré podliehajú častej zmene. Na úrovni implementácie v aplikácii sa budú prvky matice počítať dynamicky na požiadanie, eventuálne s čiastočným ukladaním výsledkov do pamäte.

Uvedená relatívne jednoduchá koncepčná schéma je teda použiteľná, ale musíme rozhodnúť, ktoré informácie potrebné pre vyhodnocovanie autorizácie je nutné ukladať v databáze.

Už sme spomenuli, že roly sú viazané k nejakej súčasti. Ku každej roli si teda budeme pamätať identifikátor súčasti UK, ku ktorej sa viaže. Na aplikačnej úrovni musíme potom zabezpečiť, aby sa tento atribút použil pri samotnej autorizácii. V budúcnosti možno pribudnú k rolám ďalšie doplnkové atribúty.

Najdôležitejšou časťou je spôsob uchovávania oprávnení a toho, k čomu sa tieto oprávnenia vzťahujú. V tabuľke uvádzame základné oprávnenia a informácie, ktoré sú potrebné na ich vyhodnotenie.

Tabuľka 5-1 Zoznam oprávnení a informácií potrebných na ich vyhodnotenie

Oprávnenie	Problém	Potrebné informácie
čítanie/zápis záznamu[7]	Ktoré atribúty daného záznamu je používateľ oprávnený prezerať/meniť?	• identifikátor záznamu, • súčasť UK, • definície pravidiel na prístup k atribútom.
aktivácia/deaktivácia preukazu	Ktoré preukazy je používateľ oprávnený požadovať aktiváciu/deaktiváciu?	• identifikátor záznamu, • súčasť UK.
notifikácia o zmene údajov	Na ktoré osoby a konkrétne atribúty si používateľ môže nastaviť notifikáciu?	• identifikátor záznamu, • súčasť UK, • definície pravidiel na prístup k atribútom.

Čítanie a zápis údajov

Problémom pri autorizácii čítania (príp. zapisovania) údajov je nutnosť aplikovať dodatočné autorizačné pravidlá, ktoré určia množinu zobrazených atribútov. Viď príklad študijnej referentky FMFI uvedený vyššie v tejto kapitole. V definícii prístupu k atribútom nestačí pre každú rolu uviesť, či má alebo nemá oprávnenie čítať (zapisovať) príslušný atribút. Jednoduchá hodnota typu „áno/nie“ nepostačuje a je potrebné na jemnejšej úrovni rozlíšiť situácie, kedy toto oprávnenie v konkrétnom prípade skutočne vlastní.

Z dôvodu lepšieho pochopenia procesu priraďovania oprávnení v prípade prezerania údajov zavedieme najprv niekoľko neformálnych definícií. Všetky sa budú vzťahovať k nejakej osobe *O* a jej roli *R*.

Def.: Pridružená osoba k osobe *O* je taká osoba, ktorá má ľubovoľný vzťah na tej istej súčasti UK ako má osoba *O* (resp. ako má definovanú rola *R*).

Def.: Majme osobu *O* a jej pridruženú osobu *P*. **Pridruženým vzťahom** osoby *O* je taký vzťah osoby *P*, ktorý je na tej istej súčasti UK ako osoba *O* (resp. ako má definovanú rola *R*).

Teraz môžeme zhrnúť jednotlivé typy zobrazovaných údajov, ktoré môže osoba *O* eventuálne vidieť:

1. vlastné údaje
2. údaje o pridružených osobách

[7] študentský záznam, zamestnanecký záznam, záznam o preukaze, záznam o ubytovaní, ...

a. pridružené vzťahy
b. ostatné vzťahy

3. údaje o ostatných osobách
    a. pridružené vzťahy
    b. ostatné vzťahy

Opäť na príklade študijnej referentky, ktorá je v roli „Študijné odd. FMFI“:

1. údaje o nej samotnej
2. údaje o študentoch FMFI
    a. študentské vzťahy na FMFI
    b. študentské alebo zamestnanecké vzťahy na iných súčastiach
3. údaje o iných osobách (všetci okrem študentov FMFI)
    a. zamestnanecké vzťahy na FMFI
    b. študentské alebo zamestnanecké vzťahy na iných súčastiach

Vo všeobecnosti sme teda identifikovali päť rôznych typov zobrazení údajov. Každé z týchto zobrazení má vlastnú množinu zobrazovaných atribútov, ktorá záleží na autorizácii konkrétnej osoby prezerajúcej si tieto údaje.

Definované zobrazenia možno zoradiť do hierarchie podľa ich reštriktívnosti vzhľadom na zobrazované atribúty. Na najnižšej úrovni bude zobrazenie, v ktorom je odfiltrovaných najviac atribútov. A postupne čím vyššia úroveň, tým viac zobrazovaných atribútov. Vzhľadom na definovanú hierarchiu zobrazení určíme pre každý atribút minimálne zobrazenie, v ktorom je povolené príslušný atribút ukázať používateľovi.

Pred zobrazením údajov používateľovi sa najprv na základe jeho roly (resp. rol) zistí, či má potrebné práva a následne sa aplikujú dodatočné autorizačné obmedzenia. Výsledkom tohto procesu je určenie druhu zobrazenia, ktoré sa má aplikovať na údaje prípadne úplné zamietnutie zobrazenia. V prvom prípade sa používateľovi zobrazia atribúty, ktorých definované minimálne zobrazenie je menšie alebo rovné ako určené zobrazenie. Pri filtrovaní údajov, ktoré je používateľ oprávnený modifikovať, možno použiť rovnaký princíp.

Operácie

Oprávnenia vzťahujúce sa k vykonaniu nejakých operácií (napr. aktivácia preukazov) zatiaľ navrhujeme evidovať jednoducho ako zoznam. Ku každej roli bude teda uvedený zoznam oprávnení, ktoré budú určovať povolené operácie. Ak sa v budúcnosti ukáže potreba evidovať tieto oprávnenia zložitejším spôsobom, nebude problém navrhnutú databázovú štruktúru rozšíriť.

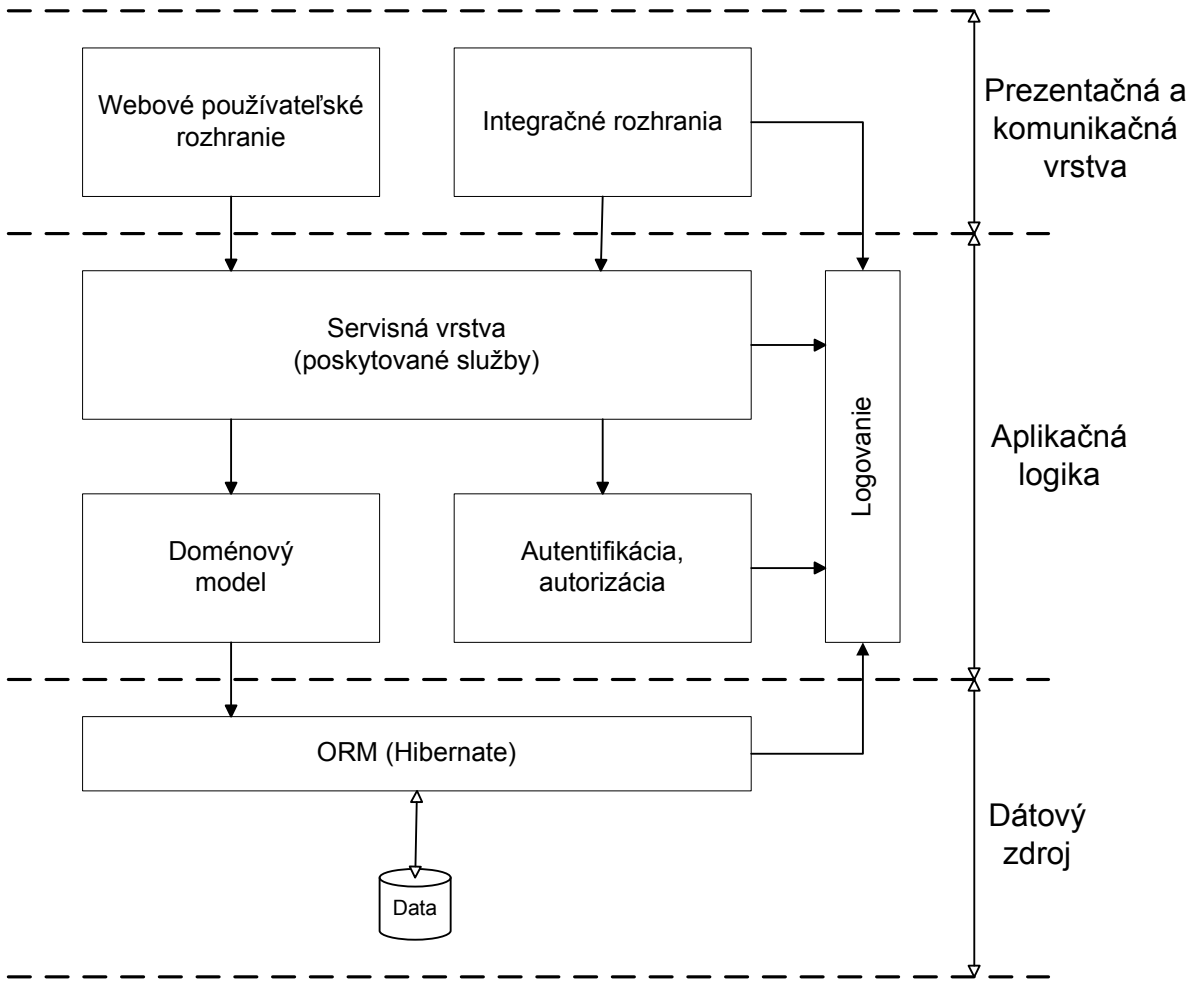
5.4 Architektúra CDO

Pri uvažovaní nad architektúrou CDO sme využili knižku Martina Fowlera ([11]), v ktorej sú popísané návrhové vzory aplikovateľné pri tvorbe veľkých podnikových aplikácií (enterprise applications). Druhým zdrojom informácií je [17]. Cieľom je navrhnúť architektúru dostatočne pružnú najmä vzhľadom k budúcemu nasadeniu integračného nástroja, ktorý prevezme od CDO všetky integračné funkcie.

Z hľadiska architektúry bude CDO postavená na vrstvovom modeli, tak ako väčšina veľkých aplikácií. V tomto modeli sú jednotlivé aspekty aplikácie zapuzdrené do samostatných vrstiev. Tým zároveň získavame možnosť izolovať integračnú funkčnosť CDO do jedného oddeleného celku, čím splníme jednu z požiadaviek flexibility.

Ďalej sme sa rozhodli použiť návrhový vzor „service layer“ [11]. Tento vzor hovorí o organizácii najvrchnejšej vrstvy aplikačnej logiky, kde sú jednotlivé funkcie aplikácie prístupné vo forme služieb. Servisná vrstva definuje hranicu aplikácie a množinu ňou poskytovaných operácií z perspektívy klientskych vrstiev, ktoré podľa potreby volajú tieto operácie. Tento prístup je výhodný vzhľadom na to, že CDO musí svoje funkcie a dáta poskytovať cez viacero rozhraní, či už používateľských alebo programových. Napriek tomu, že tieto rozhrania sa líšia, často potrebujú spoločnú interakciu s aplikáciou na prístup k údajom a aplikačnej logike.

Na obrázku je architektúra CDO na hrubej úrovni.



Obrázok 5-4 Architektúra CDO na hrubej úrovni

Aplikačná logika sa zvykne v mnohých prípadoch rozdeľovať na doménovú a servisnú logiku [17]. Doménová logika obsahuje základné pravidlá príslušnej domény. Servisná logika zahŕňa kód špecifický pre danú aplikáciu – infraštruktúrny kód (riadenie transakcií, autentifikácia, autorizácia a podobne), pracovné postupy (workflow), integrácia s inými systémami. Pri použití doménového modelu na implementáciu celej aplikačnej logiky sú doménová a servisná logika popreplietané, čo vedie k horšej udržiavateľnosti a znovupoužiteľnosti kódu. Má teda zmysel tieto dve časti aplikačnej logiky oddeliť. Na

toto oddelenie sa dá použiť servisná vrstva. Práve na úrovni tejto vrstvy bude riešené riadenie transakcií, autorizácia a z tejto vrstvy už budú volané metódy doménového modelu, ktoré zapuzdrujú len funkcie týkajúce sa príslušnej domény.

Služby servisnej vrstvy je vhodné zoskupovať kvôli prehľadnosti do logických celkov. To sa zvyčajne implementuje jedným z dvoch spôsobov. Buď vytvorením tried, z ktorých každá sprístupňuje niekoľko príbuzných služieb, alebo zapuzdrením každej služby do samostatnej triedy.

Smerom ku klientskej vrstve môžeme vystaviť servisnú vrstvu ako množinu služieb alebo použiť jednu prístupovú metódu (funkciu). Druhé riešenie je výhodnejšie z pohľadu riešenia autorizácie, auditovania a logovania na jednom mieste, teda máme menšiu duplicitu kódu a z toho vyplývajúcu väčšiu udržiavateľnosť. Parametrom prístupovej funkcie je spravidla názov požadovanej služby. Problémom pri tomto riešení je posúvanie údajov medzi servisnou a klientskou vrstvou. Vyplýva to práve z toho, že existuje len jedna prístupová funkcia, a preto jej rozhranie musí byť dostatočne všeobecné, aby vyhovovalo požiadavkám všetkých služieb. V prípade potreby zložitejšej interakcie medzi vrstvami pri vykonávaní jednej transakcie vzniká problém s implementáciou tohto prístupu, pretože je väčšinou potrebné viackrát presúvať rôzne údaje medzi vrstvami.

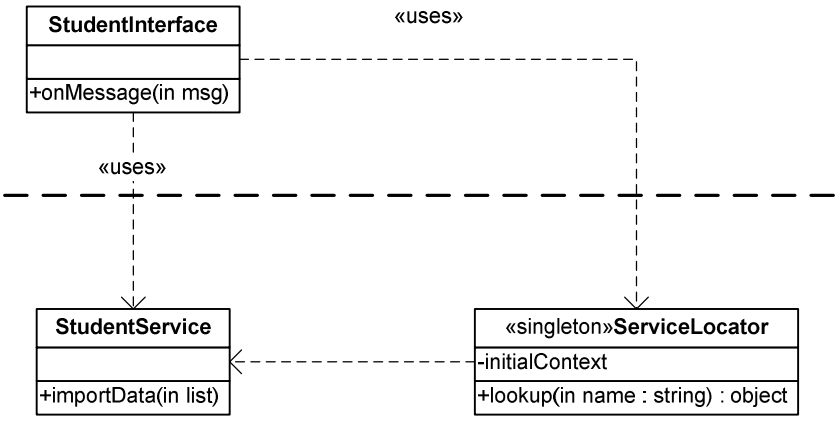
Rozhodli sme sa, že organizácia servisnej vrstvy bude založená na množine tried, ktoré budú logicky zoskupovať poskytované služby. Kritérium, podľa ktorého sa služby rozdeľujú do jednotlivých servisných tried, nie je nikdy jednoznačné. Používajú sa v podstate štyri základné prístupy [11]:

- jediná servisná trieda (použiteľné pre malé aplikácie),
- samostatná trieda pre každý subsystém,
- samostatná trieda pre každú významnú časť doménového modelu,
- samostatná trieda pre každú množinu príbuzných funkcií aplikácie.

V tomto prípade neexistuje jednoznačné rozhodnutie, pretože každý z menovaných prístupov má výhody aj nevýhody. Zohľadnením faktu, že CDO má plniť aj integračnú funkciu, pričom pridávanie a modifikovanie rozhraní má byť maximálne flexibilné (t.j. s minimálnym dopadom na existujúci kód), rozhodli sme sa vytvoriť samostatnú servisnú triedu pre každý integrovaný systém. Navyše budeme mať v modeli ešte jednu servisnú triedu pre používateľské rozhrania.

Aby sme klientskym vrstvám zjednodušili vyhľadávanie požadovaných servisných tried, implementujeme v našom riešení aj vzor „Service locator“ [15]. Zjednotí sa tým spôsob prístupu k servisnej vrstve. Klientske vrstvy budú vyhľadávať služby podľa mena a následne už len použijú metódy nájdenej servisnej triedy.

Na obrázku je znázornená ukážka návrhu fungovania servisnej vrstvy. Trieda *StudentInterface* patrí do klientskej vrstvy a predstavuje integračné rozhranie na systém Študent. Po prijatí správy a jej prípadnom počiatočnom spracovaní si pomocou *ServiceLocator* vyhľadá inštanciu servisnej triedy *StudentService* a zavolá na nej požadovanú službu (metódu).



Obrázok 5-5 Ukážka fungovania servisnej vrstvy

Čo sa týka implementácie v prostredí J2EE, najlepšou (a mnohokrát preferovanou [11]) možnosťou je realizovať každú servisnú triedu ako stateless session bean. Umožňuje to potom pri implementácii využiť služby aplikačného servera ako napr. kontajnerom riadené transakcie.

5.4.1 Implementácia vrstiev

Cieľom tejto podkapitoly je navrhnúť architektúru CDO na jemnejšej úrovni v podobe balíkov (package), hlavných tried a vzťahov medzi nimi. Východiskom bude uvedený model architektúry CDO na hrubej úrovni a formulované požiadavky – predovšetkým flexibilita.

Čo sa týka názvov balíkov a ich štruktúry, všetky balíky implementujúce CDO budú združené v balíku sk.uniba.cdo – pre názvy balíkov používame teda štandardnú konvenciu programovacieho jazyka java[8]. V nasledujúcej tabuľke uvedieme zoznam hlavných balíkov spolu so stručným popisom funkcií, ktoré v nich budú realizované.

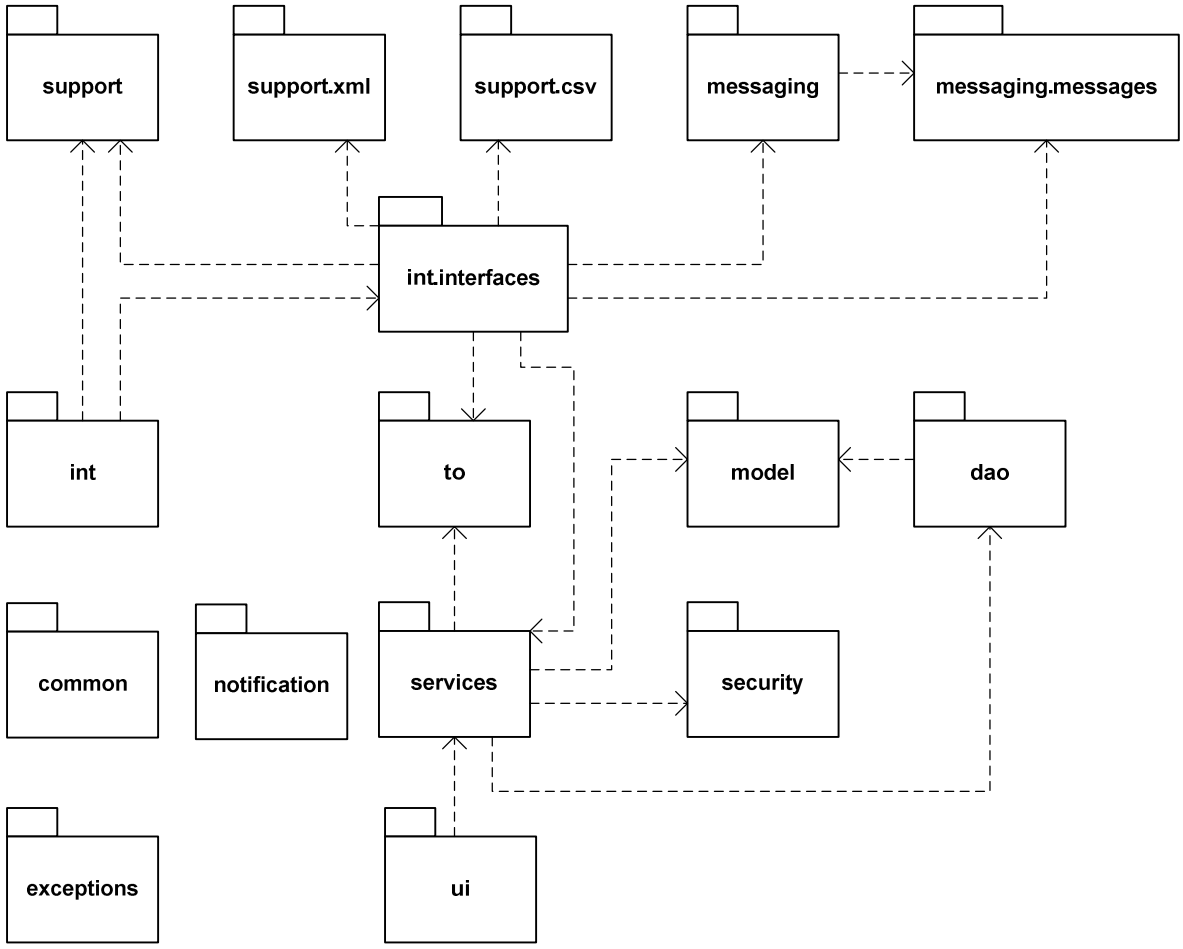
Tabuľka 5-2

Vrstva	Názov balíka[9]	Popis realizovaných funkcií
prezentačná a komunikačná vrstva	ui	webové používateľské rozhranie
	int	integračné funkcie
	int.interfaces	integračné (komunikačné) rozhrania na jednotlivé systémy
aplikačná logika	services	servisná vrstva – servisné triedy pre všetky systémy a používateľské rozhranie
	model	doménový model
	to	triedy pre transferové objekty slúžiace na prenos informácií medzi vrstvami
	messaging	triedy realizujúce asynchrónne posielanie správ
	messaging.messages	triedy predstavujúce jednotlivé typy prenášaných správ (pozri aj prílohu B „Špecifikácie rozhraní“)
	security	bezpečnostné funkcie – autentifikácia, autorizácia, filtrovanie výstupných dát
	support	rôzne pomocné triedy – napr. implementácia vzoru „Service locator“
	support.csv	parsovanie a zapisovanie CSV súborov

[8] http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html
[9] prefix sk.uniba.cdo neuvádzame kvôli prehľadnosti

	support.xml	parsovanie a zapisovanie XML súborov
	common	rôzne spoločné funkcie – konverzie medzi znakovými sadami, formátovanie/parsovanie dátumov, operácie s reťazcami a podobne
	notification	notifikácie o zmene – najmä zisťovanie, aké notifikácie treba poslať
	exceptions	triedy pre aplikačné výnimky (chybové stavy)
dátová vrstva	dao	triedy na prístup k údajom v databáze (Data Access Objects)

Najdôležitejšie vzťahy medzi balíkmi sú znázornené na obrázku. Kvôli prehľadnosti sme vynechali tie vzťahy, ktoré nie sú významné z hľadiska hlavných funkcií CDO.



Obrázok 5-6 Diagram balíkov s vyznačením najvýznamnejších vzťahov

Popíšeme teraz detailnejšie zodpovednosti a funkcie tých najdôležitejších balíkov.

Balík sk.uniba.cdo.security zodpovedá za riešenie všetkých otázok bezpečnosti. Implementuje nasledovné funkcie:

- autentifikácia používateľov a v prípade úspešnej autentifikácie vytvorenie autorizačného objektu,
- autorizácia používateľov – či má daný používateľ právo vykonať požadovanú operáciu na danom objekte.

Autorizačné pravidlá budú definované v konfiguračných tabuľkách v databáze (alternatívne v konfiguračných súboroch). Množina povolených oprávnení pre daného používateľa a daný objekt sa bude dynamicky prepočítavať na aplikačnej úrovni na základe rol, v ktorých sa príslušný používateľ nachádza a prípadne využitím dodatočných rozhodovacích kritérií (ktoré nie sú súčasťou definície autorizačných pravidiel).

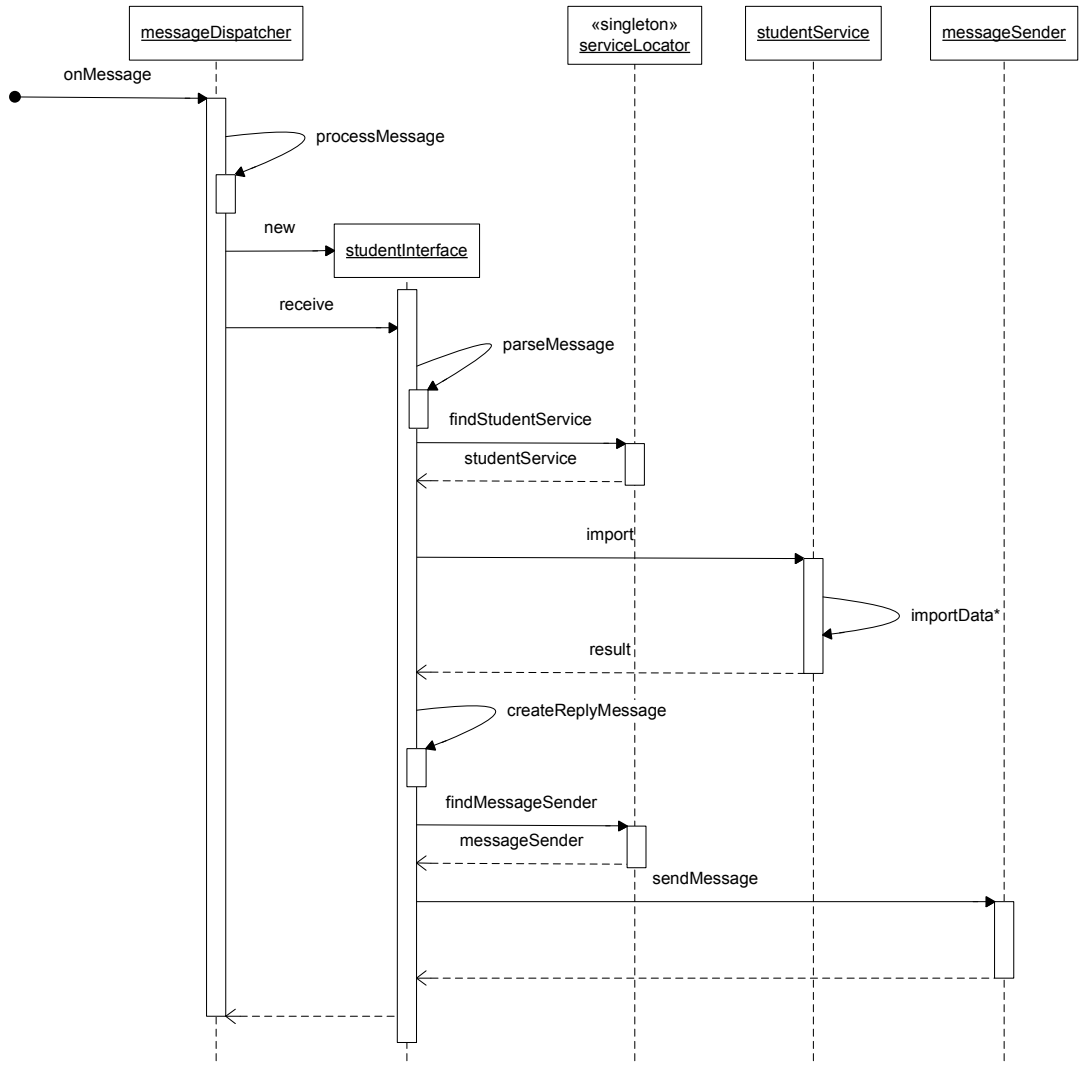
V module sk.uniba.cdo.services bude realizovaná servisná vrstva aplikácie. Dizajnové rozhodnutia týkajúce sa tejto vrstvy sme uviedli v úvode tejto kapitoly. Na ich základe bude v tomto balíku implementovaná množina servisných tried – jedna pre každý integrovaný subsystém.

Každá z týchto servisných tried bude poskytovať funkcie nutné na integráciu príslušného systému, predovšetkým je to import/export údajov a ďalšie služby poskytované CDO – napr. aktivácia preukazu či generovanie UOČ. Servisné triedy v prípade potreby využívajú služby autorizácie. Samotné vykonanie požadovanej podnikovej funkcie (business function) je delegované na doménový model, kde sú tieto funkcie implementované.

Klientska vrstva je realizovaná v sk.uniba.cdo.int.interfaces a sk.uniba.cdo.ui. Prvý menovaný balík má za úlohu komunikáciu s externými systémami, spracovanie prijatých správ do formy vhodnej na interakciu so servisnou triedou, vyvolávanie funkcií servisnej vrstvy a posielanie odpovedí na správy. Druhý menovaný balík implementuje používateľské rozhranie. Výmena údajov medzi klientskou a servisnou vrstvou je realizovaná prostredníctvom transferových objektov realizovaných v balíku sk.uniba.cdo.to.

Spomenieme ešte balík sk.uniba.cdo.messaging.*, ktorý má za úlohu posielanie správ. Prvou ideou je, že každá prijatá alebo odoslaná správa má zadefinovanú prislúchajúcu triedu. Táto trieda obsahuje všetky atribúty, ktoré majú byť uvedené v tele alebo hlavičke správy. Rozhranie triedy realizujúcej posielanie správ môže byť založené na týchto triedach pre správy a samotná metóda na posielanie správ sa tak stáva univerzálnejšou.

Na ukážkovom príklade prijatia údajov zaslaných zo systému *Študent* budeme teraz ilustrovať spôsob interakcie objektov jednotlivých vrstiev resp. balíkov. Zvolili sme formu sekvenčného diagramu UML.



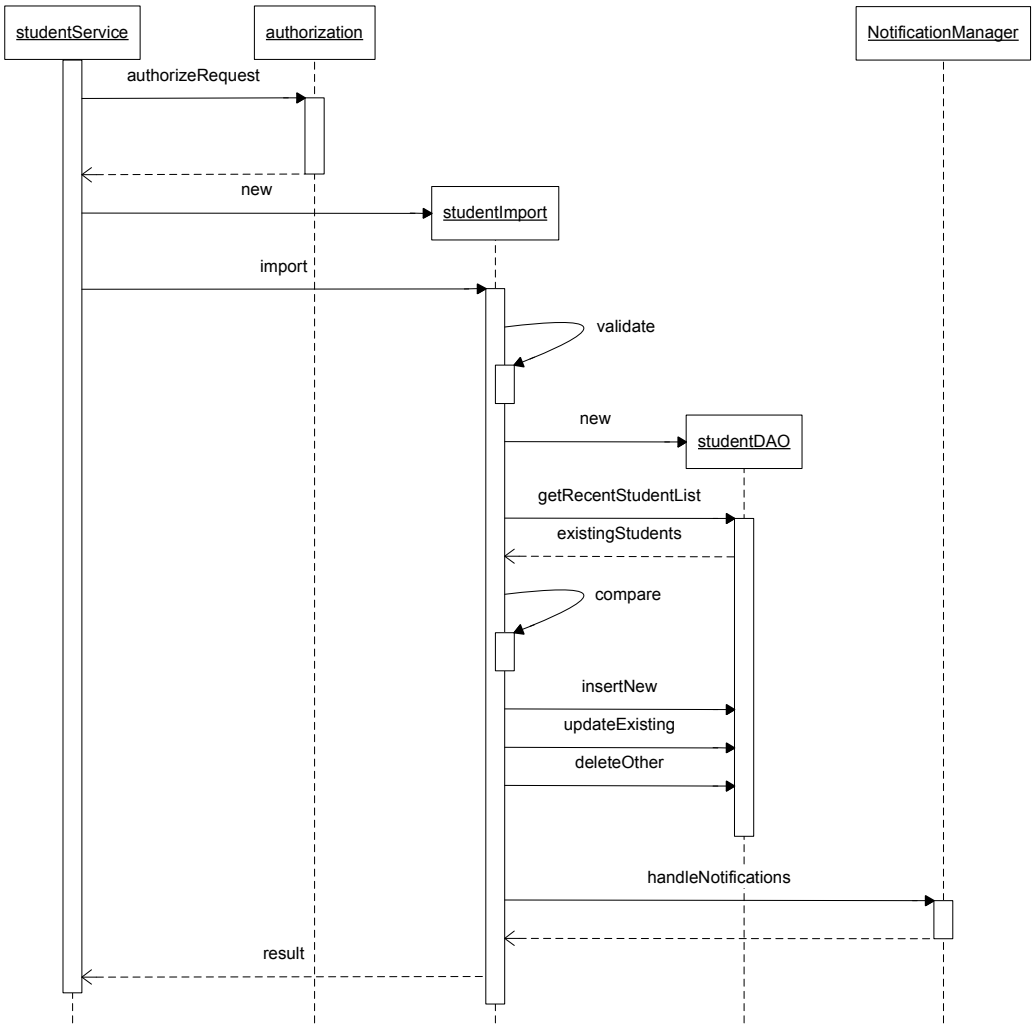
Obrázok 5-7 Časť sekvenčného diagramu – import údajov zo systému Študent

Komponent *MessageDispatcher* je zodpovedný za prijatie správy z globálnej fronty správ JMS, do ktorej sú posielané všetky správy pre CDO. Na základe atribútov v hlavičke správy zistí, o aký typ správy ide a skonvertuje správu do objektu príslušného typu – jedna z tried v sk.uniba.cdo.messaging.messages. Následne vytvorí inštanciu *studentInterface*, čo je trieda implementujúca integračné rozhranie na systém Študent a pošle tejto triede vytvorený objekt obsahujúci údaje z prijatej správy.

Objekt *studentInterface* spracuje telo správy a vytvorí transferový objekt (t.j. objekt určený na prenos údajov medzi vrstvami/modulmi aplikácie). Použitím *serviceLocator* získa referenciu na potrebnú servisnú triedu a vyvolá na nej funkciu importovania údajov. To, čo sa vykoná vnútri tejto metódy servisnej triedy (označená v diagrame ako „importData*“), nie je zachytené v tomto diagrame, ale bude uvedené nižšie. Importovacia funkcia po skončení vykonávania vráti výsledok, z ktorého vznikne objekt správy a ten sa pomocou *messageSender* pošle ako odpoveď.

Na tomto príklade je vidieť spoluprácu klientskej a servisnej vrstvy. Veľmi podobne budú realizované aj rozhrania na ostatné systémy.

Uvedieme ešte diagram popisujúci krok „importData*“ z predchádzajúceho diagramu.



Obrázok 5-8 Časť sekvenčného diagramu – import údajov z pohľadu servisnej vrstvy

Objekt *studentService* patrí do servisnej vrstvy aplikácie. Pred vykonaním operácie importu najprv overí autorizáciu. V tomto prípade je tento krok len ilustratívny, pretože import údajov sa nevykonáva v mene žiadneho používateľa. Autentifikácia a autorizácia je tu riešená na úrovni prístupu ku komunikačnému kanálu, ako sme povedali v kapitole 5.3.

Rozhodli sme sa implementovať servisnú vrstvu ako tenkú vrstvu nad zvyšnou aplikačnou logikou. Preto objekt *studentService* nevykoná import údajov sám, ale vytvorí na to objekt s*tudentImport* a deleguje na neho vykonanie importu údajov.

Objekt *studentImport* najprv vykoná validáciu údajov, ktorých import do CDO je požadovaný. Následne musí rozdeliť údaje (záznamy študentov príslušnej fakulty) do troch skupín – nové, modifikované a staré záznamy. K tomu potrebuje získať záznamy študentov príslušnej fakulty aktuálne uložené v CDO. Získanie týchto údajov deleguje do dátovej vrstvy na zodpovedný DAO (data access object) objekt *studentDAO*. Po získaní týchto záznamov z databázy, môže porovnávaním rozdeliť všetky záznamy do spomínaných troch skupín. Následne opäť pomocou *studentDAO* vloží nové záznamy, aktualizuje modifikované záznamy a vymaže všetky ostatné. Vyrobené tri zoznamy sa ešte pošlú

objektu *notificationManager*, ktorý zistí, či treba poslať nejaké notifikácie a zabezpečí poslanie notifikačných emailov (samozrejme použitím ďalších objektov).

5.5 Databázový model

V tejto kapitole navrhneme pre CDO model relačnej databázy – databázové tabuľky a vzťahy medzi nimi. Databázový model musí dosť robustný na to, aby dokázal podporovať všetky požiadavky kladené na CDO.

Prvým krokom pri návrhu databázy býva vytvorenie entitno-relačného modelu, v ktorom sú zachytené všetky dátové entity spolu s ich atribútmi a relácie medzi týmito entitami. Z entitno-relačného modelu sa potom vytvára konkrétny návrh tabuliek v databáze.

CDO však primárne nie je aplikácia na podporu činnosti organizácie v nejakej oblasti. Pripomeňme, že hlavnou úlohou CDO je prijímať a poskytovať údaje o osobách a umožniť používateľom prácu s týmito údajmi. Dátový model pre CDO je v prvom rade určený:

- dátovými modelmi zdrojových a cieľových aplikácií,
- používateľskými požiadavkami na CDO.

Pri vytváraní databázového modelu pre CDO sa najprv treba pozrieť na analýzu zdrojových a cieľových systémov a z nej identifikovať dáta, ktoré sú zdrojové systémy schopné poskytnúť. Ďalším dôležitým zdrojom informácií pri návrhu databázy sú popisy rozhraní medzi CDO a integrovanými systémami. Z týchto rozhraní dokážeme určiť tie atribúty, ktoré bude treba v CDO o každej osobe uchovávať, aby následne mohli byť poskytované cieľovým systémom. Zo špecifikácií rozhraní vyplýva aj spôsob fungovania komunikácie CDO s ostatnými systémami, z čoho možno tiež odvodiť ďalšie požiadavky na dátový model. Príkladom môže byť inkrementálny prenos údajov vyžadovaný niektorými systémami prípadne operácie ako aktivácia/deaktivácia preukazu alebo generovanie UOČ. Aby sme v CDO túto funkciu vedeli zrealizovať, potrebujeme na to podporu v databázovej štruktúre.

Ďalšou skupinou informácií, ktoré majú dopad na databázový návrh, sú používateľské funkcie CDO. Ide napríklad o autentifikáciu, autorizáciu, používateľské notifikácie alebo riešenie konfliktov pri generovaní UOČ. Opäť na implementáciu týchto funkcií je nutná podpora databázy.

Popísali sme hlavné zdroje, z ktorých budeme vychádzať pri návrhu databázového modelu. Základný model pre CDO berúci do úvahy len funkciu prenosu údajov medzi tými najdôležitejšími systémami je v princípe jednoduchý. Obsahuje tieto dátové entity:

- osoba,
- študentský vzťah,
- zamestnanecký vzťah,
- univerzitný preukaz.

Každá osoba má na univerzite aspoň jeden študentský alebo zamestnanecký vzťah. Pripúšťame aj možnosť, že má týchto vzťahov viac. Osoba teda môže mať niekoľko študentských alebo niekoľko zamestnaneckých vzťahov. Dokonca môže mať aj oba druhy vzťahov zároveň – osoba študujúca na niektorej z fakúlt a súčasne zamestnaná na UK.

Zároveň každá osoba má vydaný jeden univerzitný preukaz, nezávisle na počte vzťahov k univerzite. Univerzitný preukaz sa však vždy vzťahuje k niektorému (potenciálne minulému) vzťahu príslušnej osoby.

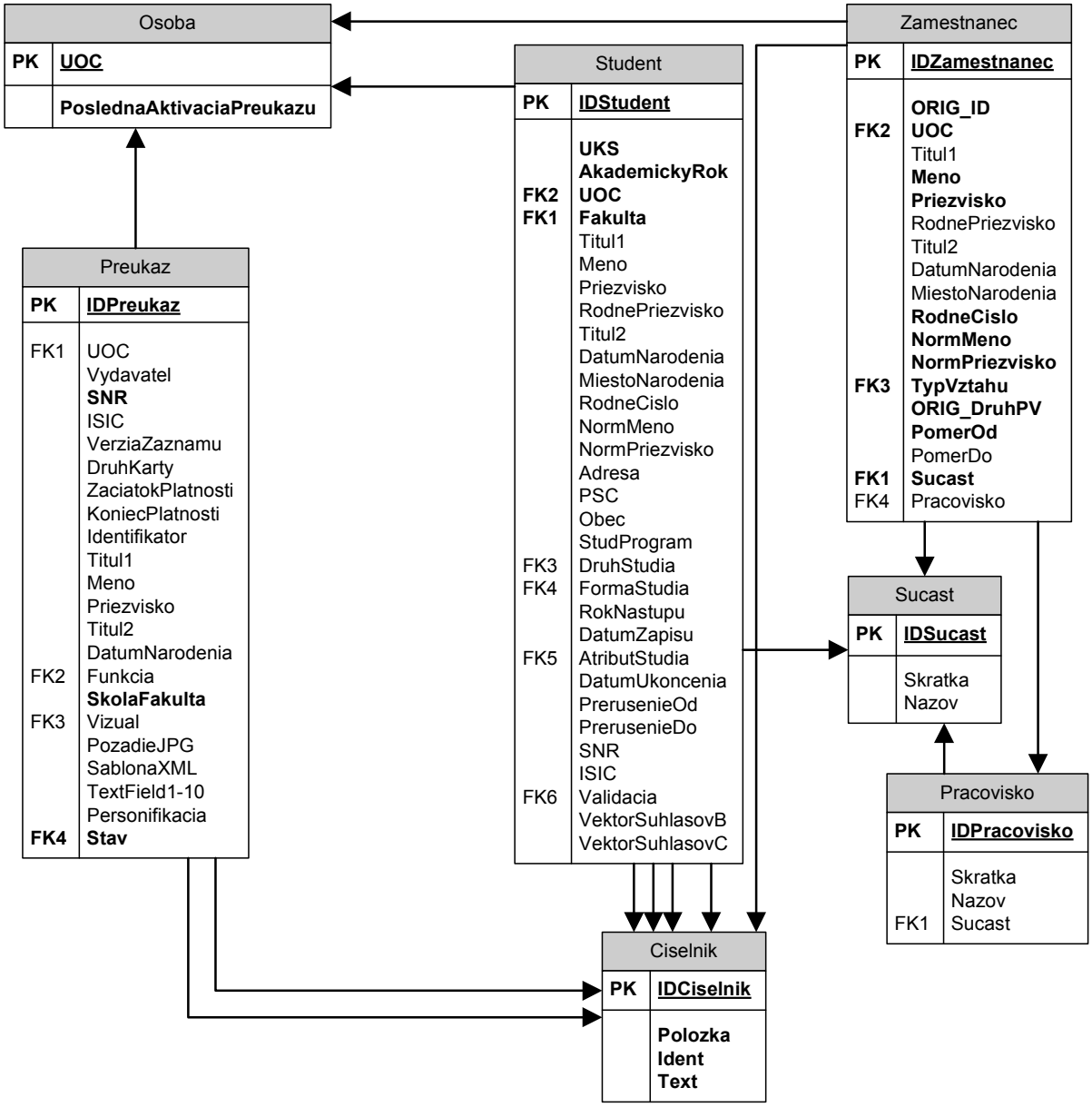
Informácie o všetkých študentských vzťahoch sa do CDO dostávajú zo systému *Študent*. Podobne, údaje o všetkých zamestnaneckých vzťahoch zo systému *Personalistika a mzdy* (*PaM*). Údaje o preukazoch sú primárne uložené priamo v CDO. Vidieť tu jasne vzťah medzi dátovými entitami a zdrojovými systémami. Osoba ako taká nie je v žiadnom zo zdrojových systémov definovaná ako samostatná entita. Práve z toho dôvodu musí koncept osoby a k nej prislúchajúcich údajov vzniknúť v rámci CDO.

5.5.1 Evidencia osôb a vzťahov

Na obrázku nižšie je časť dátového modelu, ktorá zachytáva uvedené základné entity informačného systému UK. Atribúty tabuliek *Student* a *Zamestnanec* sú práve tie, ktoré je CDO schopná získať prenosom z uvedených zdrojových systémov. Tabuľka *Preukaz* obsahuje informácie o univerzitných preukazoch. Z tejto tabuľky sa záznamy fyzicky nezmazávajú. Pre jednu osobu môže existovať viacero záznamov, avšak len jeden z nich je aktuálny resp. aktuálne používaný pri exporte do iných systémov.

Tabuľka *Osoba* by mala uchovávať atribúty, ktoré sa vzťahujú k osobe ako takej a nie sú špecifické pre nejaký vzťah k univerzite alebo pre preukaz. Taktiež, z hľadiska normalizácie, by v tejto tabuľke mali byť atribúty spoločné pre všetky tri typy záznamov. Tu však narážame na problém so sémantikou a formátom údajov. Každý z externých systémov má definovanú vlastnú sémantiku a vlastný formát ukladaných údajov. Pri importe dát do CDO sa často nedá rozhodnúť, ktorá z hodnôt nejakého atribútu je správna (platná). To samozrejme nastáva len v prípade, že ide o osobu evidovanú vo viacerých zdrojových systémoch a hodnoty niektorého zo spoločných atribútov sú rôzne.

Ako príklad uveďme študenta FMFI, ktorý súčasne pracuje pre PriF ako administrátor počítačovej siete. Jeho údaje sa nachádzajú aj v systéme *Študent* aj v systéme *PaM*. Nech v tabuľke *Osoba* máme atribút *RodneCislo*. CDO dostane údaje zo systému *Študent* a uloží ich do svojej databázy. Následne dostane údaje z *PaM* a pri ich importovaní zistí, že rodné číslo nášho študenta sa líši. Nedá sa rozhodnúť, ktoré z nich je správne. Ak prepíšeme správnu hodnotu nesprávnou, prejaví sa to v každom exporte údajov, ktorý by obsahoval záznam o tomto študentovi. Bez logovania zmien by sme nevedeli zistiť pôvod tejto chyby.



Obrázok 5-9 Hlavná časť databázového modelu

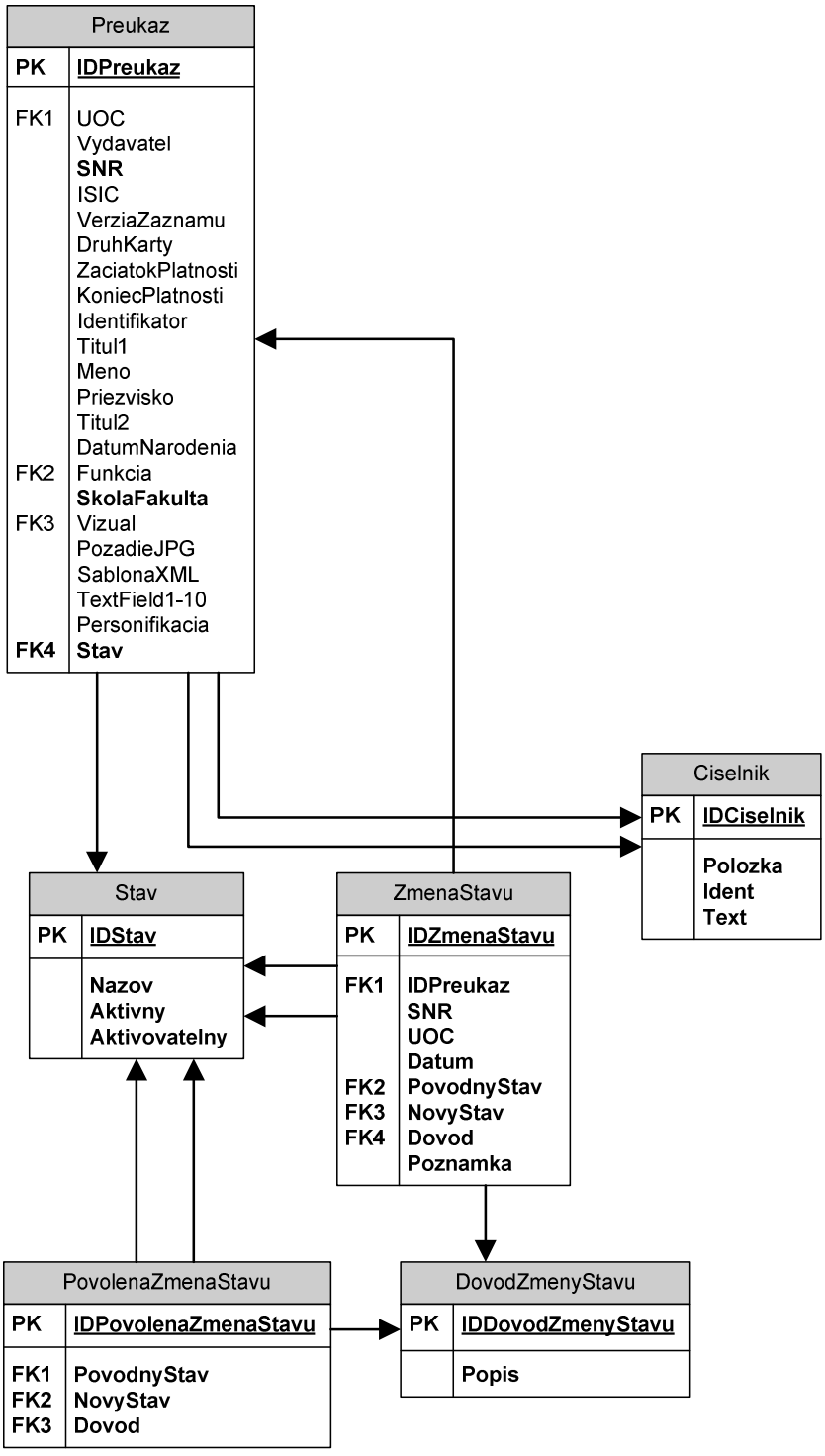
5.5.2 Evidencia preukazov

Ďalšia významná množina funkcií CDO sa týka podpory práce s univerzitnými preukazmi. Ako už bolo povedané, CDO má byť primárnym zdrojom údajov o preukazoch a tiež má poskytovať všetky funkcie spojené so správou týchto preukazov. Detailnejší popis sa nachádza v prílohe A - „Špecifikácia CDO“.

Na nasledujúcom obrázku je zobrazená štruktúra databázových tabuliek, ktoré umožnia evidenciu preukazov (tabuľka *Preukaz*) a ich spravovanie. Najdôležitejším úkonom je sledovanie stavu jednotlivých preukazov a zabránenie nedovolených prechodov medzi niektorými stavmi. Najčastejšie vykonávanými operáciami sú aktivácia a deaktivácia preukazov. Tomuto sa podriaďuje aj databázový návrh. A keďže tu nie je priama väzba na zdrojové systémy, nie sme pri návrhu obmedzení v normalizácii tabuliek.

V modeli máme zaznamenané prípustné stavy preukazov, povolené prechody medzi týmito stavmi a taktiež tabuľku na uchovanie histórie vykonaných prechodov medzi stavmi

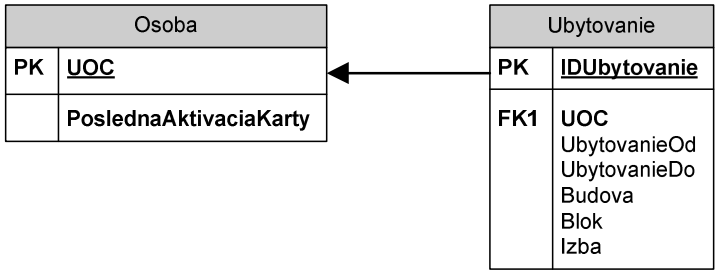
(tabuľka *ZmenaStavu*). História prechodov je veľmi dôležitá pri riešení prípadných problémov pri správe preukazov.



Obrázok 5-10 Databázové tabuľky na podporu správy preukazov

5.5.3 Evidencia ubytovania osôb

V súčasnej dobe postačuje na základe požiadaviek jednoduchá evidencia ubytovania osôb na internátoch. Pre každú osobu evidujeme, či a kde je ubytovaná. Tieto informácie pochádzajú z ubytovacích aplikácií prevádzkovaných na internátoch.



Obrázok 5-11 Jednoduchá evidencia ubytovania osôb

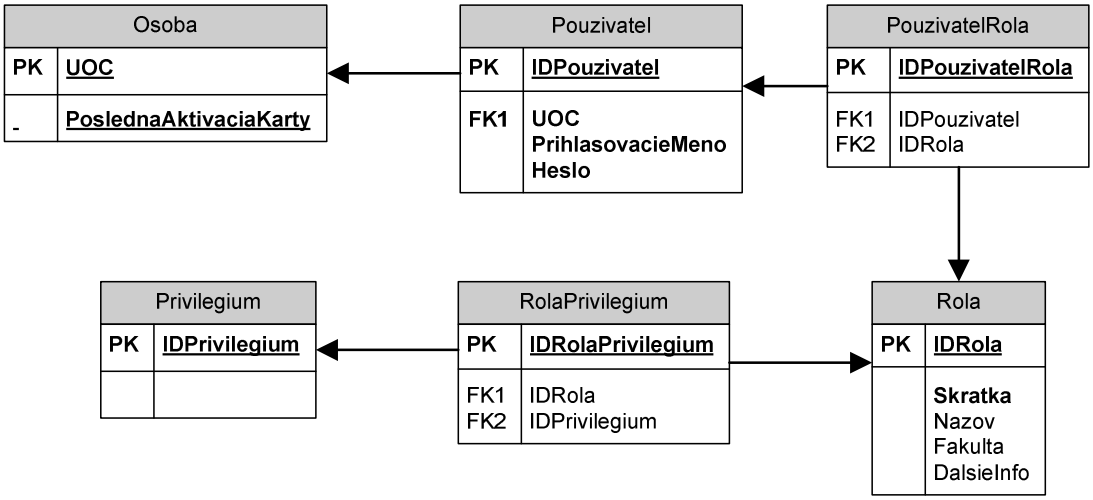
V budúcnosti sa počíta so zavedením číselníka budov, čo umožní miernu dodatočnú normalizáciu tabuľky *Ubytovanie*.

5.5.4 Evidencia používateľov a oprávnení

CDO by mala za účelom umožnenia práce s uloženými údajmi evidovať aj databázu používateľov spoločne s ich oprávneniami, ktoré by určovali, aké operácie môžu vykonávať a ktoré údaje (a konkrétne atribúty) sú autorizovaní prezerať, prípadne meniť.

Požadujeme, aby evidencia používateľských kont bola čo najviac nezávislá od zvyšku databázovej štruktúry. V budúcnosti sa totiž očakáva použitie nezávislého centrálneho komponentu zabezpečujúceho autentifikáciu používateľov (JAS – Jednotný Autentifikačný Systém).

Každý používateľ bude mať vytvorené najviac jedno používateľské konto, ale bude môcť patriť súčasne do viacerých rol (skupín oprávnení).



Obrázok 5-12 Základný návrh databázy na podporu autentifikácie a autorizácie

5.6 Technológie

V tejto časti krátko spomenieme technológie, ktoré použijeme pri implementácii CDO a dôvody, ktoré ovplyvňovali výber týchto technológií.

5.6.1 Platforma

Čo sa týka platformy pri realizácii veľkých podnikových aplikácií, zvažujú sa väčšinou dve hlavné alternatívy – Sun Java/J2EE (Java 2 Enterprise Edition) a platforma Microsoft .NET. Rozhodnutie v prospech J2EE bolo spravené už dávnejšie. Napriek tomu sa s týmto rozhodnutím stotožňujeme – najmä kvôli multiplatformovosti a dostupnosti Javy. Hlavným dôvodom zamietnutia platformy .NET bola pravdepodobne finančná

stránka a taktiež istá nestálosť licenčnej politiky spoločnosti Microsoft. Všeobecne je na univerzite presadzovaný trend využívania voľne dostupných (prípadne aj „open-source“) alternatív. Tieto sú často kvalitou porovnateľné s komerčnými produktmi a v niektorých prípadoch aj kvalitnejšie.

J2EE je vyspelá platforma, ktorá už prešla dlhým vývojom a vlastne sa stále intenzívne pracuje na jej ďalšom zlepšovaní. Vývoju tejto platformy sa okrem pracovníkov spoločnosti Sun venuje aj veľká komunita iných ľudí. Všetky súčasti javy vychádzajú zo štandardov a špecifikácií, ktoré Sun poskytuje zdarma rovnako ako ich referenčnú implementáciu. Výhodou javy je dostupnosť veľkého množstva dokumentácie, návodov a iných dokumentov, ktoré sa dajú použiť pri riešení prípadných problémov.

Relatívne nedávno bola uvoľnená nová verzia javy s číslom 1.5 a s ňou aj zmena názvu. Desktopová verzia javy je teraz označovaná ako JSE 5 (Java Platform Standard Edition 5) a enterprise verzia ako JEE 5 (Java Platform Enterprise Edition 5), ktorá však ešte nie je vo finálnej verzii – používame teda zatiaľ staršiu J2EE 1.4.

Na prevádzkovanie aplikácií na báze J2EE je nutné použiť aplikačný server, v ktorom sú takéto aplikácie spúšťané. Nebudeme sa teraz venovať popisu funkcií aplikačného servera ani tomu, prečo je výhodné ho použiť. Opäť je k dispozícii niekoľko aplikačných serverov, či už voľne dostupných alebo komerčných (platených). V minulosti bolo rozhodnuté, že sa nasadí (najprv v testovacej a neskôr v ostrej prevádzke) referenčná implementácia J2EE od spoločnosti Sun v podobe voľne použiteľného produktu Sun Java System Application Server Platform Edition 8.2. Sú aj iné voľne dostupné alternatívy, v budúcnosti pravdepodobne dôjde k preskúmaniu týchto možností.

Veľmi dôležité z hľadiska programátora je mať k dispozícii dobré vývojové prostredie. Pre javu existovali v začiatkoch projektu CDO dve voľne dostupné prostredia – Netbeans a Eclipse. Rozhodli sme sa pre Eclipse kvôli jeho väčšej vyspelosti, rozšíriteľnosti a podpore pluginov. Z dnešného pohľadu sa toto rozhodnutie ukazuje ako správne. Prostredie Eclipse sa za ten čas ešte viac zdokonalilo a dokonca sú na ňom založené viaceré komerčné vývojové prostredia.

5.6.2 Databáza

Rozhodnutie v tejto oblasti bolo jednoznačné, keďže na univerzite je dostupný databázový server Microsoft SQL Server, ktorý sa využíva aj na iné projekty. Možnosti tohto databázového servera sú pre účely CDO viac ako postačujúce.

5.6.3 Mapovanie objektov do databázy

Jednou z prvých požiadaviek týkajúcich sa implementácie CDO bolo automatické mapovanie objektov do databázy. To znamená zakomponovať do návrhu CDO vrstvu, ktorá by sa starala o automatickú perzistenciu objektov doménového modelu. Tento princíp je známy ako ORM (Object relational mapping).

Hlavné výhody takéhoto riešenia sú:

- zapuzdrenie všetkých operácií s databázou do jednej vrstvy,
- žiadna priama práca s príkazmi SQL v aplikačnom kóde,
- perzistentné objekty sú jednoduché triedy Java,
- automatické zisťovanie zmien v perzistentných objektoch a následné (tiež automatické) ukladanie do databázy,

- veľa možností mapovania objektov na databázu vrátane asociácií, dedenia, kompozície, kolekcií a podobne,
- dotazovací jazyk s veľkými možnosťami na vyhľadávanie objektov v databáze.

Na základe prieskumu možností sme získali zoznam troch kandidátskych riešení, z ktorých sme vyberali:

- Oracle TopLink,
- Entity EJB (Enterprise Java Beans),
- Hibernate.

Oracle TopLink je síce komerčný produkt, ale bol zvažovaný ako alternatíva, pretože je voľne šíriteľný. Licenčné podmienky ho však umožňujú zdarma použiť len na vývoj aplikácie a nie pri jej ostrej prevádzke. To bol hlavný dôvod zamietnutia tejto možnosti.

Entity EJB predstavovali použiteľné riešenie, avšak v čase nášho rozhodovania sa v tejto oblasti poskytovali málo možností mapovania a taktiež bol vo viacerých zdrojoch spomínaný problém s výkonnosťou pri väčšom počte objektov (entitných beanov). V súčasnej dobe je EJB vo verzii 3.0, ktorá už poskytuje podstatne viac možností a sú vyriešené problémy predchádzajúcich verzií.

Poslednou alternatívou je voľne dostupný open source nástroj na prístup k databáze a automatické mapovanie objektov s názvom Hibernate, ktorý sme napokon vybrali ako najlepšie riešenie.

Hibernate je profesionálny, vysoko výkonný a veľmi rozšírený nástroj na riešenie perzistencie objektov. Medzi hlavné charakteristiky patrí [18]:

- prirodzený programovací model – Hibernate podporuje objektovo orientované princípy programovania (dedenie, polymorfizmus, kompozícia),
- podpora veľmi detailných (fine-grained) doménových modelov – bohaté možnosti mapovania kolekcií a závislých objektov,
- extrémna škálovateľnosť – Hibernate je výkonný a má dvojvrstvovú architektúru vyrovnávacej pamäte,
- možnosti dotazov – veľa možností na vyjadrenie dotazov na databázu,
- transparentná perzistencia s automatickým zisťovaním zmien,
- mapovanie definované v externých XML súboroch.

Považujeme sa zbytočné uvádzať tu kompletný popis možností tohto nástroja, prípadných záujemcov odkazujeme na kvalitnú a rozsiahlu dokumentáciu [18].

Ďalšou veľkou výhodou v prospech Hibernate je zásuvný modul (plugin) do prostredia Eclipse, ktorý uľahčuje prácu s Hibernate – hlavne s definíciou mapovania jednotlivých objektov na tabuľky v databáze. To značne prispieva k rýchlosti vývoja aplikácie.

6 Špecifiká vývoja „integračného softvéru“

Cieľom tejto kapitoly je len orientačne porovnať vývoj „integračnej aplikácie“ typu CDO s vývojom klasických podnikových aplikácií a identifikovať ich podobnosti či odlišnosti.

Prvým rozdielom, ktorý si všimneme ešte predtým než vôbec samotný proces vývoja softvéru začne, je dôvod vzniku novej aplikácie. Podnikové aplikácie vznikajú zvyčajne za účelom automatizácie nejakých procesov v organizácii alebo na podporu činností týkajúcich sa nejakej oblasti v organizácii – napr. evidencia objednávok a faktúr, mzdové účtovníctvo, evidencia zamestnancov a podobne. Čiže vznikne aplikácia orientovaná na používateľa, v zmysle, že uľahčuje alebo podporuje činnosti vykonávané ľuďmi – zamestnancami prípadne aj zákazníkmi. Na druhej strane, pri integračnom projekte sa nekladie dôraz na používateľské funkcie, ale skôr na komunikáciu a spoluprácu jednotlivých systémov, ktoré sa zapájajú do integrácie. Účelom je hlavne zdieľanie údajov a funkcií, ktoré tieto systémy poskytujú.

Toto platí aj o CDO. Avšak špecifikom je fakt, že súčasťou CDO sú aj používateľské funkcie a to z dvoch príčin. Jednak chceme používateľom umožniť pracovať s údajmi, ktoré sa v CDO kumulujú alebo sú v nej primárne uložené. Navyše sú požadované funkcie, ktoré neposkytuje žiadny iný systém v rámci IS UK – najmä generovanie UOČ (nutné pre fungovanie zdieľania údajov a CDO ako takej) a aktivácia/deaktivácia univerzitných preukazov. Toto však nenarúša hlavný zámer vzniku CDO a možno sa na tieto funkcie pozerať ako na doplnkové vzhľadom k prenosu údajov medzi systémami.

Vývoj integračnej aplikácie rovnako ako vývoj každej väčšej podnikovej aplikácie prechádza jednotlivými etapami – špecifikácia požiadaviek, analýza, návrh, implementácia a testovanie.

Špecifikácia požiadaviek sa v prípade integračného projektu zameriava hlavne na určenie integračných zámerov, t.j. toho, čo chceme touto integráciou dosiahnuť. V špecifikácii by mala byť uvedená množina aplikácií, ktoré je treba prepojiť. Rovnako by nemal chýbať popis dôvodu, z ktorého vyplýva nutnosť zapojiť príslušný systém do integrácie. Naproti tomu pri bežnom podnikovom systéme sa špecifikácia požiadaviek skladá hlavne z modelu prípadov použitia (use cases) a ďalších popisov funkčných a nie funkčných požiadaviek. Veľká časť z týchto popisov prípadov použitia býva typu „vytvor, čítaj, uprav, zmaž“ (v angl. známy akronym CRUD, „Create, Read, Update, Delete“).

V analýze integračného projektu sa predovšetkým zaoberáme detailným rozborom možností jednotlivých systémov. Snažíme sa zistiť aké dáta a aké funkcie sú tieto systémy schopné poskytnúť. Pri analýze dát je nutné zistiť formát a sémantiku jednotlivých atribútov. Po dokončení analýzy systémov sa pozrieme na formulované integračné požiadavky a snažíme sa zistiť, či je možné ich splniť na základe identifikovaných možností externých systémov. Ak nie, treba hľadať inú cestu alebo od príslušnej požiadavky (aspoň dočasne) upustiť.

Špecifickou vlastnosťou pri integračnom softvéri je relatívne vysoká pravdepodobnosť zmien v budúcnosti. Vyplýva to jednak z prípadných zmien v integrovaných systémoch a tiež z toho, že proces integrácie je kontinuálny v tom zmysle, že postupne sa budú pripájať ďalšie systémy, ktoré budú mať svoje špecifické nároky na integráciu. Vo fáze návrhu systému teda na toto treba myslieť a zabezpečiť ľahkú

modifikovateľnosť a rozšíriteľnosť. Zmeny sú samozrejme nevyhnutné v každom softvéri, ale pri integrácií bývajú častejšie.

Pri implementácii žiadne rozdiely nenájdeme, ale zaujímavé je zamyslieť sa nad testovaním takéhoto „integračného softvéru“. Služby poskytované aplikáciou a jej vnútorné funkcie (napr. validácia a transformácia prijatých dát) sa dajú testovať štandardne technikou „black box“, t.j. zavolaním príslušnej funkcie na pripravených testovacích údajoch a následnou kontrolou výstupu funkcie. Otázkou je ako testovať integračné napojenia na externé systémy a korektnosť komunikácie medzi nimi a integračnou aplikáciou – najmä či korektne reaguje na správy a posiela správne odpovede.

Jednou z možností je nahradiť reálne systémy špeciálne pripravenými aplikáciami (stub), ktoré by emitovali testovacie správy a kontrolovali správnosť odpovedí na ne.

7 Záver

Cieľom tejto práce bolo vyriešiť reálny problém integrácie aplikácií na UK so zameraním na zdieľanie informácií o osobách. Tento integračný projekt je jedným z prvých krokov k výraznejšej integrácii aplikácií na univerzite. Okrem zabezpečenia výmeny údajov o osobách medzi jednotlivými univerzitnými systémami bolo cieľom sprístupniť tieto údaje rôznym používateľom v rámci univerzity a umožniť vykonávať základné operácie s nimi.

V práci je prezentovaná analýza vybraných systémov tvoriacich informačný systém univerzity. Táto analýza obsahuje základný popis systémov, dôvod integrácie s inými systémami a hlavne popis údajov, ktoré tieto systémy poskytujú resp. požadujú.

Na základe analýzy prepájaných systémov a tiež požiadaviek na používateľské funkcie sme rozobrali viaceré možnosti prepojenia určených systémov. Pri rozhodovaní nad správnou alternatívou riešenia bola dôležitá robustnosť a podpora budúcich zmien. Skonštatovali sme, že v súčasnej situácii je potrebná existencia centrálnej databázy osôb (CDO), ktorá by zabezpečila nepretržitú dostupnosť údajov o osobách. A to hlavne tých, ktoré sú uložené v systémoch nedostupných v ľubovoľnom čase. Zároveň bolo rozhodnuté, že CDO bude riešiť aj integračné väzby na jednotlivé systémy. Bude teda komunikovať s týmito systémami za účelom prijímania a poskytovania údajov a tiež bude poskytovať systémom ďalšie služby podľa potreby – napr. prideľovanie UOČ, overovanie čísiel preukazov a podobne.

Perspektívou do budúcnosti je použitie integračného nástroja v spolupráci s centrálnou databázou. Integračný nástroj umožní jednoduchšie a pružnejšie splnenie integračných požiadaviek a podporí rozširovanie integrácie aplikácii na UK.

Veľká časť práce sa zaoberala podrobným popisom CDO – od špecifikácie požiadaviek cez návrh architektúry aplikácie a štruktúry databázy až po popis technológií použitých pri implementácii. Pri návrhu sme sa zamerali na riešenie hlavných požiadaviek a požadovaných funkcií – prenos údajov, sledovanie zmien, bezpečnosť, používateľské funkcie (prezeranie údajov, notifikácie o zmenách). Súčasťou práce je aj fragment implementácie a je predpoklad, že sa stane základom aplikácie nasadenej v ostrej prevádzke.

Niektorým oblastiam sme sa pri návrhu CDO venovali najmä z časových dôvodov len okrajovo. Ide hlavne o návrh riešenia funkcie notifikácie a prípadné upravenie modelu pre autorizáciu kvôli lepšej podpore vyhodnocovania autorizačných pravidiel. V budúcnosti by takisto bolo možné sa bližšie venovať problému generovania UOČ – predovšetkým návrhu vhodného algoritmu na jednoznačné prideľovanie týchto identifikátorov osobám na univerzite.

Po nasadení integračného nástroja a sprevádzkovaní nových informačných systémov na evidenciu študentov a zamestnancov bude potrebné prehodnotiť množinu funkcií poskytovaných CDO. Množinu atribútov evidovaných v CDO bude pravdepodobne tiež potrebné upraviť, keďže mnohé z nich sa stanú nepretržite dostupnými priamo zo zdrojových systémov.

8 Zoznam použitej literatúry

[1] LINTHICUM, David S. *Next Generation Application Integration : From Simple Information to Web Services*. 1st ed. Boston : Addison-Wesley, 2003. 512 p. ISBN 0201844567.

[2] HOHPE, G., WOOLF, B. *Enterprise integration patterns : Designing, Building, and Deploying Messaging Solutions*. 1st ed. Boston : Addison-Wesley, 2003. 736 p. ISBN 0321200683.

[4] JACKSON, T., MAJUMDAR, R., WHEAT, S. *OpenEAI Methodology* [online]. Dostupné na internete: <http://xml.openeai.org/site/doc/Methodology.htm>. Dostupné tiež v PDF a RTF verziách na internete: <http://www.openeai.org/site/public.live?document=Documentation.xml&focus=N30>

[5] DOERNHOFER, M. Surfing the Net for Software Engineering Notes. In *ACM SIGSOFT Software Engineering Notes* [online]. New York : ACM Press. 2005, vol. 30, no. 6, pp. 5-13. Dostupné na internete <http://delivery.acm.org/10.1145/1110000/1102116/p5-doernhoefer.pdf?key1=1102116&key2=4085427411&coll=portal&dl=ACM&CFID=71067021&CFTOKEN=36913182>. ISSN 0163-5948.

[6] IBM developerWorks. *New to SOA and Web services* [online]. Dostupné na internete: <http://www-128.ibm.com/developerworks/webservices/newto>.

[7] REYNOLDS, J. *The SOA Elevator Speech* [online]. Dostupné na internete: <http://weblogs.java.net/blog/johnreynolds/archive/2005/01/the_soa_elevato.html>.

[8] KEEN, M., ADINOLFI, O., HEMMINGS, S., HUMPHREYS, A., KANTHI, H., NOTTINGHAM, A. *Patterns : SOA with an Enterprise Service Bus in WebSphere Application Server V6* [online]. 1st ed. IBM Redbooks, 2005. 404 p. ISBN 073849058X. Dostupné na internete: <http://www.redbooks.ibm.com/redbooks/SG246494/wwhelp/wwhimpl/java/html/wwhelp.htm>. Dostupné tiež v PDF verzii na internete: <http://www.redbooks.ibm.com/redbooks/pdfs/sg246494.pdf>.

[9] ERL, T. *Service-Oriented Architecture : Concepts, Technology, and Design*. 1st ed. Prentice Hall, 2005. 792 p. ISBN 0131858580.

[10] CHAPPELL, D. *Enterprise Service Bus*. 1st ed. O'Reilly, 2004. 352 p. ISBN 0596006756.

[11] FOWLER, M. *Patterns of Enterprise Application Architecture*. 1st ed. Boston : Addison-Wesley, 2002. 560 p. ISBN 0321127420.

[12] ŠEŠERA, Ľ., MIČOVSKÝ, A., ČERVEŇ, J. *Architektúra softvérových systémov : Analytické dátové vzory*. 1. vyd. Bratislava : Slovenská technická univerzita, 2000. 179 strán. ISBN 8022713587.

[13] SNOGRASS, R. T. *Developing Time-Oriented Database Applications in SQL*. 1st ed. San Francisco : Morgan Kaufmann Publishers, 2000. 504 p. ISBN 1558604367.

[14] GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. *Design Patterns : Elements of Reusable Object-Oriented Software*. 1st ed. Boston : Addison-Wesley, 1995. 395 p. ISBN 0201633612.

[15] ALUR, D., CRUPI, J., MALKS, D. *Core J2EE Patterns : Best Practices and Design Strategies*. 2nd ed. Santa Clara : Prentice Hall, 2003. 650 p. ISBN 0131422464.

[16] TANENBAUM, A. S., VAN STEEN, M. *Distributed Systems : Principles and Paradigms*. 1st ed. Prentice Hall, 2002. 803 p. ISBN 0130888931.

[17] BLŠTÁK, P., ŠEŠERA, P. Modelom riadený vývoj softvéru s využitím vzorov. In *Proceedings of the Annual Database Conference DATAKON 2005*. edited by Tomáš Hruška. Brno : Masarykova univerzita, 2005. pp. 93-122.

[18] *Hibernate manual* [online]. Dostupné na internete: <http://www.hibernate.org>.

[19] Mederly, Peter, Mederly, Pavol Stručný úvod do problematiky počítačových sietí a informačných systémov. In *Management vysokých škôl IV "Informačné systémy na vysokých školách"*. Academia Istropolitana, 1995. pp. 4-34.

A Centrálna databáza osôb – špecifikácia požiadaviek

A.1 Úvod

Centrálna databáza osôb (CDO) má byť jedným zo základných elementov integrovaného informačného a komunikačného systému Univerzity Komenského (IIKS UK). CDO je krokom k integrácii aplikácií v rámci IIKS, pričom hlavným zámerom v tejto fáze je zdieľanie údajov o osobách medzi jednotlivými systémami v IIKS.

CDO by mala evidovať údaje o študentoch a zamestnancoch univerzity, o ich vzťahoch k univerzite a o ich preukazoch. Okrem samotného získavania a poskytovania údajov by CDO mala svojim používateľom ako aj integrovaným systémom ponúkať ďalšie služby ako napríklad aktiváciu/deaktiváciu preukazov, prideľovanie UOČ alebo notifikáciu o zmene údajov. Navyše by CDO mala mať aj používateľské rozhranie umožňujúce pracovať s uloženými údajmi, vykonávať potrebné operácie a diagnostikovať rôzne problémy.

A.2 Popis požiadaviek

V tejto časti stručne popíšeme všetky používateľské funkcie, ktoré od Centrálnej databázy osôb očakávame.

A.2.1 Generovanie UOČ

CDO musí poskytovať službu generovania univerzitných osobných čísiel pre osoby na univerzite.

UOČ je unikátny identifikátor každej osoby na univerzite. CDO sa musí starať o centrálnu evidenciu a prideľovanie týchto identifikátorov tak, aby nedochádzalo k akýmkoľvek anomáliám (napr. jedna osoba s dvoma UOČ).

CDO musí taktiež umožniť operátorovi systému jednoduché riešenie problémov (konfliktov) pri prideľovaní UOČ cez príslušné používateľské rozhranie.

A.2.2 Správa preukazov

Súčasťou CDO má byť aj kompletne riešená správa preukazov. To znamená evidencia všetkých vydaných univerzitných preukazov a podpora operácií pracujúcimi s databázou preukazov – hlavne aktivácia/deaktivácia/blokovanie. CDO musí vedieť prijímať a automatizovane spracovávať požiadavky na aktiváciu preukazov.

CDO musí tiež v maximálnej možnej miere automatizovať/podporovať proces výroby nových preukazov, a to pre preukazy vyrábané na UK ako aj preukazy vyrábané a dodávané externým dodávateľom.

Táto požiadavka súvisí aj s vytvorením vhodného používateľského rozhrania pre operátora systému automatickej identifikácie osôb (SAIO), cez ktoré by mohol jednoducho vykonávať väčšinu úkonov spojených s výrobou resp. správou preukazov.

A.2.3 Vyhľadávanie a prezeranie informácií

CDO má slúžiť ako hlavný zdroj údajov o osobách na univerzite a je preto ideálnym miestom pre implementáciu funkcie prezerania údajov o osobách. CDO musí umožniť používateľom po ich autentifikácii prezerať uložené údaje.

Každý používateľ by mal mať možnosť prezerať údaje, na ktoré má oprávnenie. Toto oprávnenie závisí od jeho roly v systéme. Každá osoba by mala mať možnosť vidieť všetky údaje, ktoré sú o nej v systéme evidované. Ďalej by mala existovať možnosť definovať skupiny používateľov, ktoré budú mať vyššie oprávnenia a budú môcť prezerať aj informácie o iných vybraných osobách. Množina údajov, ktorú je používateľ oprávnený vidieť, by mala byť relatívne ľahko definovateľná.

A.2.4 Notifikácie

CDO musí používateľom umožniť zaregistrovať žiadosti o notifikáciu. Notifikácie sa budú vzťahovať na tri základné udalosti:

- vznik nového záznamu,
- modifikácia existujúceho záznamu,
- zmazanie existujúceho záznamu.

Notifikácia sa môže týkať prakticky ľubovoľného typu záznamu – študentský záznam, zamestnanecký záznam, záznam o preukaze. Konkrétna žiadosť o notifikáciu sa vzťahu vždy na jeden konkrétny typ záznamu.

Pri registrovaní notifikácie o vzniku nového záznamu bude mať používateľ dve možnosti:

- Používateľ určí konkrétnu súčasť UK a bude požadovať zaslanie notifikácie, ak pre akúkoľvek osobu vznikne nový záznam vzťahujúci sa k príslušnej súčasti.
- Používateľ určí konkrétnu osobu a chce byť notifikovaných o všetkých nových záznamoch tejto osoby (príslušného typu).

Podobne pri notifikácii o zmazaní záznamu by používateľ mal mať analogické dve možnosti. K nim sa ešte pridáva tretie možnosť, pri ktorej používateľ určí konkrétnu osobu a konkrétny záznam príslušného typu. Notifikačný email sa posiela len v prípade zániku tohto konkrétneho záznamu.

Používateľ registrujúci si žiadosť o notifikáciu zmeny údajov musí presne určiť osobu, konkrétny záznam resp. všetky záznamy a množinu tých atribútov, o ktorých zmene chce byť notifikovaný. Pri zmene ľubovoľného z týchto atribútov by CDO mala poslať notifikačný email.

Vo všetkých troch prípadoch bude v notifikačnej žiadosti uvedená emailová adresa, na ktorú sa budú notifikácie eventuálne posielať a text notifikačného emailu.

A.2.5 Logovanie

Pre administrátora CDO musí udržiavať kompletný log o všetkých udalostiach, ktoré sa v systéme udiali. Aspoň počas pilotnej prevádzky by tento log mal byť úplne detailný, dokonca by systém mal archivovať aj všetky správy, ktoré prijme resp. pošle.

Systém by mal definovať niekoľko úrovní logovania, pričom aktuálne použitá úroveň detailu logovania by mala byť nastaviteľná v konfiguračnom súbore.

V prípade zlyhania niektorej dôležitej funkcie by systém mal byť schopný poslať notifikačný e-mail na vopred definované adresy (v konfiguračnom súbore).

A.3 Prípady použitia (use case model)

A.3.1 Generovanie UOČ

Funkciu generovania UOČ môže iniciovať:

- operátor manuálne, v prípade, že potrebuje vygenerovať UOČ pre jednu alebo niekoľko osôb,
- jeden zo subsystémov zaslaním správy so zoznamom osôb, pre ktoré požaduje vygenerovať UOČ,
- samotné CDO pri importe údajov z nejakého subsystému za účelom aktualizácie pomocných údajov.

Názov	manuálne generovanie UOČ	
Identifikátor	UC001	
Cieľ	vygenerovať UOČ pre jednu osobu	
Vstupné podmienky	operátor je prihlásený do systému	
Výstupné podmienky v prípade úspechu	systém vráti UOČ pre danú osobu	
Výstupné podmienky v prípade neúspechu	žiadna zmena, systém vypíše chybové hlásenie	
Primárni a sekundárni účastníci (Actors)	Operátor	
Popis	**Krok**	**Akcia**
	1	operátor sa rozhodne manuálne vygenerovať nové UOČ pre nejakú osobu
	2	operátor vo svojom používateľskom rozhraní zvolí funkciu generovania UOČ
	3	systém zobrazí formulár, kde operátor musí zadať určujúce údaje pre danú osobu
	4	systém zadané údaje zobrazí ešte raz operátorovi kvôli kontrole
	5	operátor skontroluje správnosť zadaných údajov
	6	systém odošle údaje do CDO, kde sa spustí generátor UOČ v režime 1[10]
	7	vygenerované alebo nájdené UOČ je zobrazené operátorovi (v prípade, že bolo vygenerované nové UOČ, je zapísané v CDO)
Alternatívy	**Krok**	**Akcia**
	5	ak údaje nie sú správne, operátor má možnosť ich zmeniť v tomto kroku, ešte pred definitívnym odoslaním
	7	ak pri generovaní vzniknú konflikty, vykoná sa UC002 a podľa výsledku sa buď zobrazí vygenerované UOČ alebo chybové hlásenie

Názov	riešenie konfliktu pri generovaní UOČ

[10] popis fungovanie generátora UOČ sme z priestorových dôvodov vynechali

Identifikátor	UC002	
Cieľ	rozhodnúť o riešení konfliktu vzniknutom pri generovaní UOČ pre nejakú osobu	
Vstupné podmienky	operátor je prihlásený do systému	
Výstupné podmienky v prípade úspechu	systém vykoná akciu podľa rozhodnutia operátora	
Výstupné podmienky v prípade neúspechu	-	
Primárni a sekundárni účastníci (Actors)	Operátor	
Popis	**Krok**	**Akcia**
	1	systém zobrazí operátorovi údaje o danom konflikte (pôvodne zadané určujúce údaje ako aj možnosti ponúkané generátorom UOČ)
	2	operátor prezrie ponúkané záznamy, pričom má možnosť pozrieť si doplňujúce údaje
	3	operátor rozhodne, ktorý z nájdených záznamov určuje tú istú osobu
	4	systém zaznamená rozhodnutie operátora a pošle ho CDO, ktoré podľa toho vykoná príslušnú akciu
Alternatívy	**Krok**	**Akcia**
	3	operátor sa môže rozhodnúť, že nevie odpovedať, aj táto možnosť je prípustná, generátor UOČ potom nezapíše nič a UOČ pre túto osobu nie je vygenerované (CDO si to však uloží do logu)

Názov	automatické generovanie UOČ	
Identifikátor	UC003	
Cieľ	vygenerovať UOČ pre požadované osoby	
Vstupné podmienky	-	
Výstupné podmienky v prípade úspechu	systém vráti zoznam UOČ pre dané osoby	
Výstupné podmienky v prípade neúspechu	žiadna zmena, systém pošle naspäť príslušnú odpoveď	
Primárni a sekundárni účastníci (Actors)	subsystém (ktorý žiada generovanie UOČ)	
Popis	**Krok**	**Akcia**
	1	subsystém pošle CDO zoznam osôb, pre ktoré požaduje vygenerovanie UOČ (v tomto zozname uvedie pre každú osobu jej určujúce údaje)
	2	CDO správu prijme, spracuje a pre každú osobu na zozname zavolá generátor UOČ (režim 1)
	3	generátor UOČ vždy pre danú osobu vygeneruje UOČ
	4	CDO takto získa zoznam UOČ pre dané osoby

	5	CDO nakoniec pošle zoznam UOČ späť subsystému, ktorý poslal požiadavku
	6	subsystém získané údaje príslušne spracuje (zrejme import do svojej vnútornej databázy)
Alternatívy	**Krok**	**Akcia**
	3	ak vznikne konflikt pri generovaní UOČ pre danú osobu, ktorý vyžaduje zásah operátora, CDO si zapamätá všetky údaje o tomto konflikte a pokračuje ďalej v generovaní
	3a	v odpovedi CDO uvedie len tie UOČ, ktoré sa podarilo úspešne vygenerovať a vzniknuté konflikty pošle operátorovi na rozhodnutie (UC002)
	3b	keď operátor rozhodne, môžu byť prípadné ďalšie UOČ dodatočne poslané žiadajúcemu subsystému

A.3.2 Správa preukazov

Názov	tlač preukazov	
Identifikátor	UC004	
Cieľ	vyrobiť preukazy pre vybrané osoby	
Vstupné podmienky	operátor je prihlásený do systému	
Výstupné podmienky v prípade úspechu	v databáze preukazov vzniknú záznamy pre vytlačené preukazy	
Výstupné podmienky v prípade neúspechu	systém vypíše chybové hlásenie, eventuálne nastane chybový stav vyžadujúci zásah operátora do databázy preukazov	
Primárni a sekundárni účastníci (Actors)	Operátor	
Popis	**Krok**	**Akcia**
	1	operátor vo svojom používateľskom rozhraní zvolí funkciu tlače preukazov
	2	operátor si zvolí typ preukazov, ktoré chce tlačiť (študentské, zamestnanecké)
	3	systém zobrazí vhodné GUI na vyhľadávanie/filtrovanie osôb, kde operátor vyberie osoby, pre ktoré sa budú tlačiť preukazy
	4	systém zobrazí zoznam vybraných osôb
	5	operátor kliknutím na niektorú položku zoznamu môže zmeniť všetky údaje o danej osobe, okrem UOČ[11]
	6	systém automaticky doplní čísla ISIC/ITIC licencií, v prípade, že typ tlačených preukazov tieto čísla vyžaduje
	7	systém automaticky k daným osobám vyhľadá fotografie v databáze fotografií

[11] avšak s výrazným upozornením!

	8	systém vytvorí MDB súbor a skopíruje ho do prednastaveného zdieľaného adresára, potom systém čaká na pokyn operátora
	9	operátor spustí externý program na tlač preukazov (VSPrint), ktorý preukazy vytlačí a do MDB súboru doplní čísla týchto preukazov
	10	po skončení VSPrint systém znovu otvorí MDB súbor, na pokyn operátora
	11	systém zobrazí zoznam osôb v tomto MDB súbore
	12	operátor má možnosť ľubovoľný z týchto záznamov vyradiť zo zoznamu[12]
	13	systém naimportuje zostávajúce záznamy do CDO
Alternatívy	**Krok**	**Akcia**
	7	ak sa nepodarí nájsť všetky fotografie, systém zobrazí zoznam takýchto osôb
	7a	operátor musí doplniť chýbajúce fotografie alebo odstrániť záznam zo zoznamu, ak k nemu nemá fotografiu
	9	operátor v tomto kroku môže zvoliť, že ide o hromadnú tlač a chce využiť služby externého dodávateľa
	9a	systém v tom prípade odošle MDB súbor ako aj všetky potrebné fotografie externému dodávateľovi (FTP prenos)
	9b	tým sa proces preruší, kroky 10-13 sa vykonajú, až keď bude k dispozícii doplnený MDB
	9c	systém si tento stav pamätá a operátor potom iniciuje manuálne vykonanie krokov 10-13 s tým, že poskytne požadovaný MDB súbor
	13	v prípade zlyhania importu systém vypíše chybové hlásenie so zoznamom tých záznamov, pre ktoré import zlyhal

Názov	oprava záznamu o preukaze
Identifikátor	UC005
Cieľ	opraviť chybný záznam o preukaze v CDO
Vstupné podmienky	operátor je prihlásený do systému
Výstupné podmienky v prípade úspechu	chybný záznam bude blokovaný a vznikne nový záznam so správnymi údajmi
Výstupné podmienky v prípade neúspechu	systém vypíše chybové hlásenie, žiadna zmena v databáze nenastane
Primárni a sekundárni účastníci (Actors)	Operátor

[12] napr. v prípade, že sa vyskytla nejaká chyba pri tlači preukazu, a teda príslušný záznam nie je platný

Popis	**Krok**	**Akcia**
	1	operátor vo svojom používateľskom rozhraní zvolí funkciu opravy preukazu
	2	systém zobrazí vhodné GUI na vyhľadávanie/filtrovanie záznamov o preukazoch, kde operátor vyhľadá záznam, ktorý chce opraviť
	3	systém zobrazí všetky údaje o danej karte
	4	operátor môže zmeniť ľubovoľný údaj a následne potvrdí tieto zmeny
	5	systém zablokuje starý preukaz (t.j. starý záznam bude označený ako poškodený) a vytvorí nový záznam použitím operátorom upravených údajov
Alternatívy	**Krok**	**Akcia**
	5	ak bol preukaz aktivovaný, tak je rozhodnutie na operátorovi, ktorý má možnosť nový záznam o preukaze hneď aj aktivovať (implicitne však aktivovaný nebude)

Názov	automatická aktivácia preukazov	
Identifikátor	UC006	
Cieľ	vykonať hromadnú aktiváciu preukazov	
Vstupné podmienky	-	
Výstupné podmienky v prípade úspechu	preukazy budú v CDO označené ako aktivované	
Výstupné podmienky v prípade neúspechu	žiadna zmena v databáze nenastane	
Primárni a sekundárni účastníci (Actors)	subsystém (ktorý žiada aktiváciu preukazov)[13]	
Popis	**Krok**	**Akcia**
	1	subsystém pošle CDO zoznam osôb, pre ktoré požaduje aktivovať preukaz
	2	CDO sa pre každú osobu pokúsi preukaz aktivovať a výsledkom každej aktivácie je stavová informácia o úspešnosti aktivácie
	3	CDO pošle tieto stavové informácie ako odpoveď žiadajúcemu subsystému
Alternatívy	**Krok**	**Akcia**

Názov	manuálna aktivácia preukazu
Identifikátor	UC007
Cieľ	vykonať aktiváciu preukazu
Vstupné podmienky	používateľ je prihlásený do systému a má oprávnenie aktivovať preukazy

[13] v súčasnosti len systém Študent

Výstupné podmienky v prípade úspechu	príslušný preukaz bude v CDO označený ako aktivovaný	
Výstupné podmienky v prípade neúspechu	žiadna zmena v databáze nenastane	
Primárni a sekundárni účastníci (Actors)	operátor / pracovník štud. oddelenia / pracovník pers. oddelenia[14]	
Popis	**Krok**	**Akcia**
	1	používateľ sa rozhodne aktivovať preukaz pre nejakú osobu
	2	za pomoci vhodného GUI rozhrania umožňujúceho vyhľadávanie podľa rôznych kritérií vyhľadá preukaz, ktorý chce aktivovať
	3	používateľ si vyžiada aktiváciu daného preukazu
	4	systém pošle požiadavku na aktiváciu CDO
	5	CDO vykoná aktiváciu preukazu a vráti stavovú informáciu
	6	systém zobrazí stavovú informáciu o úspechu aktivácie používateľovi
Alternatívy	**Krok**	**Akcia**

A.3.3 Vyhľadávanie informácií a diagnostika problémov

Názov	základné informácie o sebe	
Identifikátor	UC008	
Cieľ	osoba chce zistiť základné informácie o sebe	
Vstupné podmienky	-	
Výstupné podmienky v prípade úspechu	systém zobrazí množinu základných informácií o danej osobe	
Výstupné podmienky v prípade neúspechu	-	
Primárni a sekundárni účastníci (Actors)	ľubovoľná osoba (používateľ)	
Popis	**Krok**	**Akcia**
	1	študent alebo zamestnanec sa rozhodne zistiť informácie o svojej osobe z CDO
	2	používateľ zadá svoje meno a priezvisko do systému (cez webové rozhranie)
	3	systém podľa zadaných údajov vyhľadá danú osobu v CDO
	4	systém zobrazí nájdené základné informácie o tejto osobe (meno, priezvisko, stručný zoznam vzťahov[15])
Alternatívy	**Krok**	**Akcia**
	4	v prípade, že podľa zadaných údajov nebola nájdená žiadna osoba, systém zobrazí chybové

[14] v popise prípadu použitia (use case) ľubovoľného z týchto účastníkov nazývame "používateľ"
[15] pre študentský vzťah: fakulta, druh štúdia, študijný odbor; pre zamestnanecký vzťah: fakulta, katedra

		hlásenie

Názov	úplné informácie o sebe	
Identifikátor	UC009	
Cieľ	osoba chce zistiť úplné informácie o sebe	
Vstupné podmienky	používateľ je prihlásený do systému	
Výstupné podmienky v prípade úspechu	systém zobrazí všetky informácie o danej osobe	
Výstupné podmienky v prípade neúspechu	-	
Primárni a sekundárni účastníci (Actors)	ľubovoľná osoba (používateľ)	
Popis	**Krok**	**Akcia**
	1	študent alebo zamestnanec sa rozhodne zistiť kompletné informácie o svojej osobe z CDO
	2	používateľ požiada o výpis kompletných údajov o sebe (cez webové rozhranie)
	3	systém vyhľadá danú osobu v CDO podľa jej UOČ[16]
	4	systém zobrazí všetky nájdené informácie o tejto osobe (všetko o danej osobe, jej vzťahoch a preukazoch)
Alternatívy	**Krok**	**Akcia**
	4	v prípade, že vyhľadávanie z nejakých príčin zlyhá, systém zobrazí chybové hlásenie

Názov	prezeranie údajov o osobách a preukazoch	
Identifikátor	UC010	
Cieľ	používateľ chce vyhľadávaním a prezeraním údajov o osobe resp. preukaze zistiť príčinu nejakého problému	
Vstupné podmienky	používateľ je prihlásený do systému	
Výstupné podmienky v prípade úspechu	-	
Výstupné podmienky v prípade neúspechu	-	
Primárni a sekundárni účastníci (Actors)	operátor / pracovník CIT / pracovník štud. oddelenia / pracovník pers. oddelenia[17]	
Popis	**Krok**	**Akcia**
	1	používateľ má možnosť vyhľadávať v databáze osôb a preukazov podľa rôznych kritérií (ako napr. meno a priezvisko, UOČ, číslo preukazu)
	2	systém na základe zadaných atribútov vyhľadá všetky relevantné záznamy v CDO a zobrazí ich v prehľadnej forme (vhodne štruktúrované

[16] keďže používateľ je prihlásený do systému, jeho UOČ už systém pozná

[17] rozdiel medzi týmito používateľmi je v tom, akú množinu atribútov o osobe/karte sú oprávnení vidieť pri prezeraní údajov

		tabuľky, hyperlinkové odkazy na detaily a pod.)
	3	systém pred zobrazením nájdených údajov eventuálne vyfiltruje niektoré atribúty na základe oprávnenia práve prihláseného používateľa
Alternatívy	**Krok**	**Akcia**
	2	v prípade, že vyhľadávanie z nejakých príčin zlyhá, systém zobrazí chybové hlásenie

A.3.4 Notifikácie

Názov	registrácia žiadosti o notifikáciu	
Identifikátor	UC011	
Cieľ	používateľ chce nadefinovať novú žiadosť o notifikáciu	
Vstupné podmienky	používateľ je prihlásený do systému a je oprávnený zaregistrovať žiadosť o notifikáciu	
Výstupné podmienky **v prípade úspechu**	-	
Výstupné podmienky **v prípade neúspechu**	-	
Primárni a sekundárni účastníci (Actors)	ľubovoľná osoba (používateľ)	
Popis	**Krok**	**Akcia**
	1	používateľ si zvolí typ údajov, o ktorých zmene chce byť notifikovaný
	2	používateľ si vyberie typ notifikácie (vznik, zmena, zánik)
	3a	ak je požadovaná notifikácia o vzniku, systém umožní buď vybrať konkrétnu súčasť alebo vyhľadať a vybrať konkrétnu osobu
	3b	ak je požadovaná notifikácia o zániku, systém umožní buď vybrať konkrétnu súčasť alebo vyhľadať a vybrať konkrétny záznam príslušného typu
	3c	ak je požadovaná notifikácia o zmene, systém umožní používateľovi vybrať osobu, následne buď určí konkrétny záznam príslušného typu alebo zaznačí, že má záujem sledovať všetky záznamy príslušného typu, nakoniec určí konkrétne atribúty, o ktorých zmene chce byť notifikovaný
	4	používateľ zadá emailovú adresu, kam majú byť poslané prípadné notifikácie
	5	používateľ zadá text, ktorý má byť uvedený v notifikačnom emaily
Alternatívy	**Krok**	**Akcia**

B Popis rozhraní

V tejto prílohe sú uvedené popisy jednotlivých komunikačných rozhraní medzi CDO a ostatnými integrovanými systémami. Navrhli sme vhodnú formu zápisu týchto rozhraní. Niektoré rozhrania už boli do veľkej miery dané – tieto sme len spísali do navrhnutej formy. Niektoré rozhrania sme mohli ovplyvniť či už z hľadiska formátu prenášaných údajov alebo spôsobu ich prenosu.

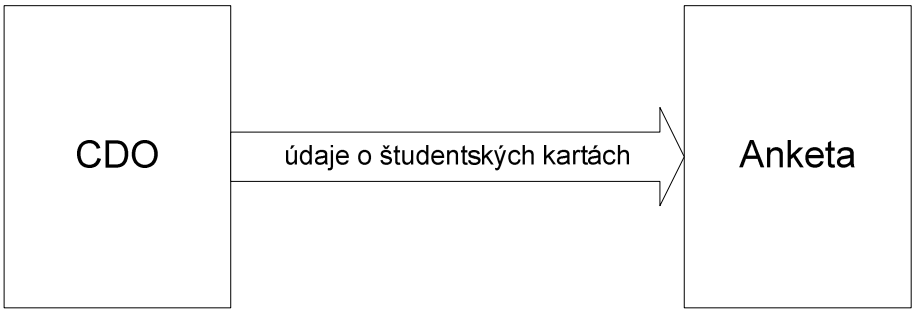
B.1 Anketa a Centrálna databáza osôb

B.1.1 Anketa

Každoročne sa na fakultách UK organizuje študentská anketa, aby vedenia fakúlt mali spätnú väzbu od študentov. Študenti v tejto ankete hodnotia vyučované predmety, ktoré absolvovali a taktiež samotných vyučujúcich. Anketa sa vypĺňa elektronicky prostredníctvom webových formulárov. Pred samotným hodnotením sa každý študent musí prihlásiť číslom karty. Anketový systém overuje, či karta patrí študentovi danej fakulty. Údaje o číslach kariet spoločne s príslušnými fakultami anketový systém získa prenosom z centrálnej databázy osôb.

B.1.2 Tok údajov

Táto časť obsahuje popis interakcie medzi integrovanými systémami len na hrubej úrovni, kvôli získaniu prehľadu.



Anketa

Prijíma údaje o študentských kartách z CDO, ale neposiela žiadne údaje ani požiadavky.

CDO

Generuje exportný súbor pre Anketu s dohodnutým formátom.

B.1.3 Posielané správy a realizácia komunikácie

Komunikácia CDO so systémom Anketa je len jednosmerná. Operátor CDO manuálne spustí proces generovania exportného súboru, ktorý má svoj špecifický formát. Tento súbor je potom poslaný do Ankety.

Na ten samotný exportný súbor sa možno pozerať ako na správu, ktorú CDO posiela do systému Anketa.

Prenos údajov o kartách do Ankety

`sk.uniba.cdo.messages.AnketaExport`

Túto správu generuje CDO na požiadanie (manuálne operátorom) a obsahuje zoznam údajov o študentských kartách.

B.1.3.1 Logika generovania požiadaviek

Anketový systém negeneruje žiadne požiadavky.

B.1.3.2 Logika prijímania údajov

`sk.uniba.cdo.messages.AnketaExport`

Túto správu generuje CDO len explicitne, keď operátor túto funkciu spustí. Výsledkom procesu exportu údajov, je textový exportný súbor.

Tento exportný súbor obsahuje len neblokované študentské karty s neprázdnym UOČ také, že daný študent má otvorený vzťah k univerzite, a síce, že je študentom danej fakulty v aktuálnom akademickom roku.

Exportný súbor:

- textový súbor s kódovaním je Windows-1250,
- vo formáte CSV,
- obsahuje hlavičku,
- oddeľovač je čiarka,
- jednotlivé hodnoty musia byť v úvodzovkách.

Atribúty prenášaného CSV súboru sú nasledovné:

Názov	Typ	Veľkosť	Význam	Neprázdne
SNR	char	10	sériové číslo preukazu	A
fakulta	char	4	číslo fakulty (na ktorej má držiteľ karty (s uvedeným SNR) otvorený študijný vzťah)	A

B.2 CKM a Centrálna databáza osôb

B.2.1 CKM

Spoločnosť CKM 2000 Travel v rámci svojich aktivít zabezpečuje aj rozvoj a distribúciu medzinárodných identifikačných preukazov:

- Medzinárodný identifikačný preukaz študenta – ISIC
- Medzinárodný identifikačný preukaz učiteľa – ITIC

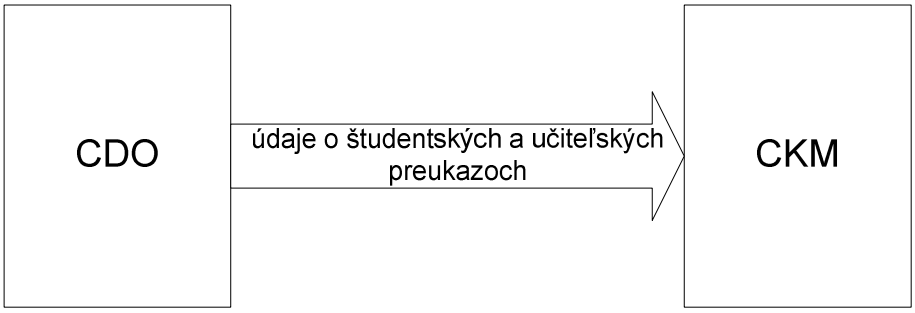
Preukaz denného študenta univerzity je v skutočnosti ISIC preukaz. Podobne, preukaz učiteľa na univerzite je platným ITIC preukazom.

CKM poskytuje univerzite zoznam voľných ISIC resp. ITIC licencií, ktoré sú použité pri výrobe uvedených preukazov. Bez týchto čísiel licencií nemôžu byť vyrobené preukazy považované za platné ISIC/ITIC preukazy.

CKM, v záujme poskytovania svojich služieb, požaduje pravidelné posielanie základných údajov o osobách (študenti, učitelia) a číslach im pridelených ISIC/ITIC licencií. Posielanie týchto informácií z CDO do CKM je hlavným zámerom komunikačného rozhrania, ktoré ďalej podrobne popíšeme.

B.2.2 Tok údajov

Táto časť obsahuje popis interakcie medzi integrovanými systémami len na hrubej úrovni, kvôli získaniu prehľadu.



CKM

Prijíma údaje o vydaných preukazoch denných študentov a učiteľov, ale neposiela žiadne údaje ani požiadavky.

CDO

Generuje exportný súbor pre CKM s dohodnutým formátom a dohodnutými údajmi.

B.2.3 Posielané správy a realizácia komunikácie

Komunikácia CDO so spoločnosťou CKM je len jednosmerná. Proces prenosu údajov z CDO do CKM sa vykonáva automaticky pravidelne raz mesačne alebo aj manuálne na požiadanie operátora SAIO. Operátor SAIO môže manuálne spustiť proces generovania exportného súboru so špecifickým formátom, ktorý ďalej popíšeme. Tento súbor je potom poslaný do CKM (prostredníctvom e-mailu).

Na ten samotný exportný súbor sa možno pozerať ako na správu, ktorú CDO posiela do CKM.

Prenos údajov o študentských a učiteľských preukazoch do CKM

`sk.uniba.cdo.messages.CKMExport`

Túto správu generuje CDO na požiadanie (manuálne operátorom) a obsahuje zoznam základných údajov o preukazoch denných študentov a učiteľov.

B.2.3.1 Logika generovania požiadaviek

CKM negeneruje žiadne požiadavky. V súčasnosti pracovník z CKM e-mailom alebo telefonicky požiada o zaslanie údajov.

B.2.3.2 Logika prijímania údajov

`sk.uniba.cdo.messages.CKMExport`

Táto správa obsahuje exportný súbor s údajmi o vydaných ISIC/ITIC preukazoch. Správa sa do CKM posiela e-mailom.

Exportný súbor obsahuje v súčasnosti len údaje o aktívnych ISIC preukazoch (t.j., validácia je ISIC, karty boli aktivované a neboli deaktivované).

Exportný súbor:

- textový súbor s kódovaním je Windows-1250,

- vo formáte CSV,
- obsahuje hlavičku,
- oddeľovač je čiarka,
- jednotlivé hodnoty musia byť v úvodzovkách.

Atribúty prenášaného CSV súboru sú nasledovné:

Názov	**Typ**	**Veľkosť**	**Význam**	**Neprázdne**
CisloPreukazu	char	10	číslo ISIC/ITIC licencie	A
Meno	char	25	meno	A
Priezvisko	char	25	priezvisko	A
Ulica	char	50	ulica – trvalé bydlisko	A
Mesto	char	50	mesto – trvalé bydlisko	A
PSC	char	5	PSČ – trvalé bydlisko	A
DatumNarodenia	char	10	dátum narodenia (DD.MM.YYYY)	A
Pohlavie	char	1	označenie pohlavia „M“ = muž „Z“ = žena	A

B.3 EMcard a Centrálna databáza osôb

B.3.1 EMcard

Každá univerzitná karta môže byť použitá aj na rôzne iné účely (externá funkcionalita), ako napríklad:

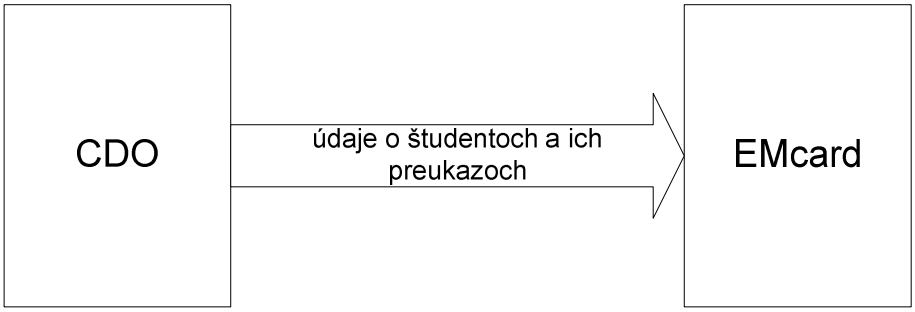
- verejná doprava prímestská, diaľková, MHD
- „elektronická peňaženka“
- časové lístky MHD
- železničná doprava – žiacke preukazy
- prístupové systémy

Spoločnosť EMcard sa poskytuje služby systémového integrátora, ktorý zabezpečuje akceptáciu bezkontaktných čipových kariet (akými sú aj univerzitné preukazy) v spoločnostiach poskytovateľov služieb navzájom a čipových kariet vydávaných na univerzitách u poskytovateľov služieb.

Práve z dôvodu, aby študentské univerzitné preukazy mohli byť akceptované v dopravných podnikoch a v železničnej doprave, je nutné, aby EMcard mal aktuálne informácie o všetkých vydaných preukazoch. Tieto informácie poskytuje CDO spoločnosti EMcard prostredníctvom rozhrania, ktoré teraz bližšie popíšeme.

B.3.2 Tok údajov

Táto časť obsahuje len rámcový popis interakcie medzi integrovanými systémami.



EMcard

Prijíma údaje o študentoch a ich kartách z CDO, ale neposiela žiadne údaje ani požiadavky.

CDO

Generuje exportný súbor pre EMcard a prenáša tento súbor na FTP server EMcardu.

B.3.3 Posielané správy a realizácia komunikácie

Komunikácia CDO so spoločnosťou EMcard je len jednosmerná. CDO v pravidelných časových intervaloch (raz týždenne) spustí proces generovania exportného súboru, ktorý má špecifický formát. Tento súbor je potom automaticky prenesený na FTP server EMcardu.

Na exportný súbor sa možno pozerať ako na správu, ktorú CDO posiela EMcardu.

Prenos údajov o študentoch a ich preukazoch do EMcardu

`sk.uniba.cdo.messages.EMcardExport`

Túto správu generuje CDO v pravidelných intervaloch a obsahuje zoznam údajov o študentoch a ich preukazoch.

B.3.3.1 Logika generovania požiadaviek

EMcard negeneruje žiadne požiadavky.

B.3.3.2 Logika prijímania údajov

`sk.uniba.cdo.messages.EMcardExport`

Túto správu generuje CDO v pravidelných časových intervaloch, raz týždenne. Následne sa použitím FTP protokolu pošle vygenerovaný exportný súbor na FTP server EMcardu.

Tento exportný súbor obsahuje len údaje o študentoch, ktorí dali súhlas s poskytnutím údajov podnikom SAD a DPB resp. ŽSR.

Exportný súbor:

- textový súbor s kódovaním ISO-8859-2,
- každá položka musí byť zakončená nulou (0x0),
- obsahuje hlavičku,
- obsahuje jeden záznam pre každého študenta (resp. preukaz).

Konkrétne dátové štruktúry použité na prenos údajov neuvádzame, pretože sú predmetom duševného vlastníctva spoločnosti EMcard.

B.4 HelpDesk software a Centrálna databáza osôb

B.4.1 HelpDesk software (HDSW)

HelpDesk software je systém pre podporu práce výpočtového strediska, ktorý eviduje počítače, používateľov a incidenty a umožňuje vykonávať niektoré administrátorské úkony v rámci počítačovej siete.

V súčasnej dobe existujú 2 verzie HDSW:

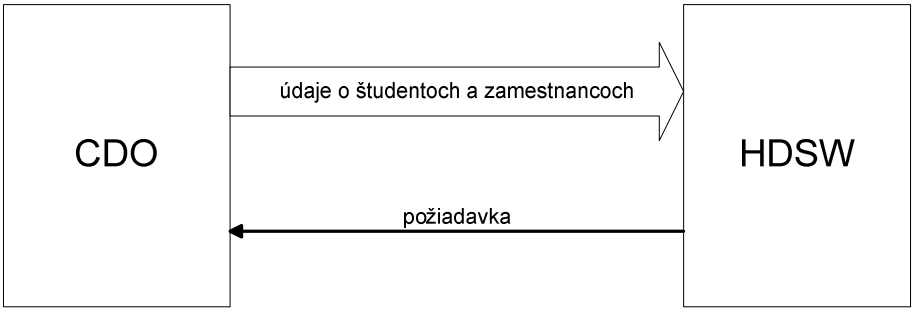
- HDSW pre ubytovaných, ktorý je prevádzkovaný na internátoch
- HDSW pre zamestnancov, ktorý je prevádzkovaný na LF UK a Rektoráte UK

Do budúcnosti sa plánuje integrácia týchto dvoch (dosť podobných) verzií do jednej spoločnej, ktorá bude konfigurovateľná v závislosti na tom, kde sa inštancia tohto softvéru nasadí. Na každej súčasti univerzity (resp. na väčšine) by mala byť prevádzkovaná jedna inštancia HDSW.

Z hľadiska integrácie aplikácií je našim cieľom automaticky poskytovať údaje o osobách z CDO do HDSW.

B.4.2 Tok údajov

Táto časť obsahuje popis interakcie medzi integrovanými systémami len na hrubej úrovni, kvôli získaniu prehľadu.



HDSW

- Posiela požiadavku na zaslanie údajov do CDO.
- Prijíma z CDO správy obsahujúce základné údaje o študentoch a zamestnancoch.

CDO

- Prijíma a spracováva požiadavky na zaslanie údajov od inštancie HDSW.
- Posiela údaje o študentoch a zamestnancoch, ktoré sa do CDO dostali z primárnych zdrojov týchto údajov[18], do HDSW.

[18] systém Študent, Personalistika a Mzdy, UBYT

B.4.3 Posielané správy a realizácia komunikácie

Komunikácia CDO a systému HDSW bude implementovaná použitím nasledujúcich správ, detailná definícia správ spolu s popisom logiky generovania požiadaviek a prijímania odpovedí na ne nasleduje ďalej.

Prenos údajov o osobách z CDO do HDSW

`sk.uniba.cdo.messages.HDSWRequest`[19]

Správa predstavuje požiadavku HDSW na zaslanie údajov o osobách z CDO.

`sk.uniba.cdo.messages.HDSWReply`

Správa obsahuje požadovaný zoznam údajov o osobách.

V uvedeného vyplýva, že komunikácia je riadená na strane HDSW, ktoré môže posielať požiadavky do CDO v ľubovoľnom okamihu. Predpokladá sa, že bude tieto požiadavky generovať pravidelne v istých časových intervaloch (ktorý by mohol byť konfigurovateľný podobne ako ďalšie parametre HDSW).

Výhodou tohto riešenia je flexibilita. Každá inštancia HDSW si môže samostatne určovať ako často chce prijímať údaje z CDO. Navyše sa tým zjednodušuje proces pridávania nových inštancií HDSW, lebo na strane CDO bude potrebná len minimálna zmena.

Formát posielaných údajov navrhujeme nasledovný:

- textový súbor s kódovaním je Windows-1250,
- vo formáte XML.

XML je pravdepodobne najlepšou voľbou, pretože do budúcnosti umožňuje použitie prípadné automatických transformácií cez XSLT. Navyše v tomto prípade nie sme obmedzení vo výbere formátu ani samotným integrovaným systémom ani nejakým rozhodnutím, ktoré eventuálne mohlo byť spravené v minulosti.

B.4.3.1 Logika generovania požiadaviek

`sk.uniba.cdo.messages.HDSWRequest`

Túto správu posiela HDSW do CDO ako požiadavku na zaslanie údajov.

Inštancia HDSW je povinná v hlavičke správy, okrem iných atribútov, uviesť aj *identifikátor súčasti univerzity*, na ktorej je daná inštancia prevádzkovaná. Tento atribút využije CDO pri generovaní odpovede na túto požiadavku. Telo správy bude prázdne.

Zámerom je, aby CDO poslala naspať do HDSW iba relevantné záznamy, t.j., len údaje o tých osobách, ktoré majú „vzťah“ k danej súčasti univerzity. V prípade internátov sú to napríklad len osoby ubytované na danom internáte. V prípade fakúlt sú to zase len zamestnanci danej fakulty.

B.4.3.2 Logika prijímania odpovedí na požiadavky

`sk.uniba.cdo.messages.HDSWReply`

Túto správu generuje CDO ako odpoveď na požiadavku, ktorú prijala od konkrétnej inštancie HDSW.

[19] konvencia: štandardné pomenovanie používané v jave, cdo = názov celého projektu, slovo za poslednou bodkou je samotný názov správy, pričom prvé slovo je meno systému a zvyšok označuje funkciu

Správa obsahuje zoznam údajov o relevantných osobách, ktorý CDO vygenerovala na základe identifikátora súčasti UK získaného z prijatej požiadavky. (Viď popis predchádzajúcej správy.)

O každej osobe sú prenášané atribúty popísané v nasledujúcej tabuľke.

Názov	**Typ**	**Veľkosť**	**Význam**	**Neprázdne**
uoc	char	10	univerzitné osobné číslo	A
titulypred	char	15	titul(y) pred menom	N
meno	char	25	krstné meno	A
priezvisko	char	25	priezvisko	A
titulyza	char	15	titul(y) za menom	N
vztah	char	1	vzťah k univerzite („S“ = študent, „Z“ = zamestnanec)	A
objbud	char	20	objekt/budova	N
blok	char	20	blok	N
izbamiestnost	char	20	izba/miestnosť	N
email	char	50	e-mailová adresa	N
telcislo	char	15	telefónne číslo	N
foto				A

Štruktúra posielaného XML súboru je popísaná v nasledujúcej tabuľke.

Nadradený element	**Element**	**Význam**	**Neprázdne**
Osoba	UOC	univerzitné osobné číslo študenta	A
	Meno	krstné meno	A
	Priezvisko	priezvisko	A
	TitulyPred	tituly pred menom	N
	TitulyZa	tituly za menom	N
	Vztah	vzťah k univerzite	A
	ObjektBudova	objekt/budova	N
	Blok	blok	N
	IzbaMiestnost	izba/miestnosť	N
	Email	e-mailová adresa	N
	Telefon	telefónne číslo	N
Osoby	Osoba	element obsahujúci údaje o jednej osobe	-

Príklad posielaného XML súboru:

```
<?xml version="1.0" encoding="windows-1250"?>
<Osoby>
     <Osoba>
     <UOC>12346262</UOC>
     <Meno>Jožko</Meno>
     <Priezvisko>Mrkvička</Priezvisko>
     <TitulyPred>Mgr.</TitulyPred>
          <TitulyZa>PhD</TitulyZa>
          <Vztah>Z</Vztah>
          <ObjektBudova>Rektorát UK</ObjektBudova>
```

```
            <Blok></Blok>
            <IzbaMiestnost>SB102</IzbaMiestnost>
        <Email>Jozko.Mrvicka@rec.uniba.sk</Email>
        <Telefon>123123123</Telefon>
        </Osoba>
        <!—dalsie osoby–>
</Osoby>
```

B.5 Personalistika a mzdy a Centrálna databáza osôb

B.5.1 Personalistika a mzdy (PaM)

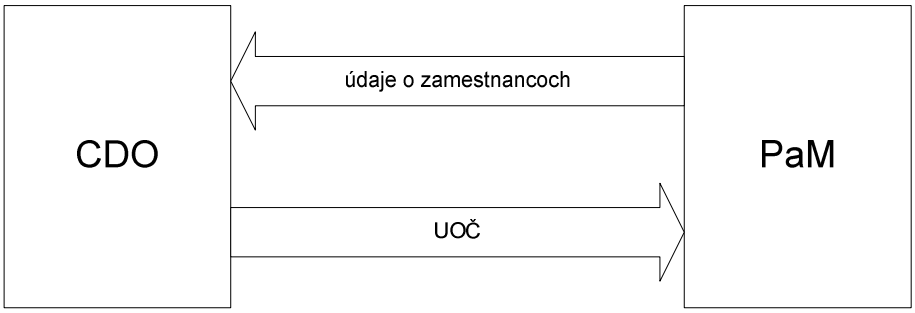
Personalistika a mzdy je systém na kompletnú evidenciu údajov o zamestnancoch – osobné údaje, ukončené vzdelanie a prax, druh a trvanie pracovného pomeru, pracovné a mzdové zaradenie, dovolenky. V súčasnosti je na každej fakulte prevádzkovaná jedna inštancia tohto systému.

Z integračného hľadiska má toto rozhranie slúžiť na:

- prenos údajov o zamestnancoch do CDO (odkiaľ sú tieto údaje poskytované ďalším systémom)
- generovanie UOČ pre zamestnancov a následný prenos týchto čísiel do systému PaM

B.5.2 Tok údajov

Táto časť obsahuje len rámcový popis interakcie medzi integrovanými systémami.



PaM

- Ako primárny zdroj údajov o zamestnancoch musí PaM pravidelne (a prípadne pri nejakej zmene v údajoch) posielať synchronizačné správy, aby v CDO boli vždy aktuálne údaje.
- PaM posiela CDO požiadavky na vygenerovanie UOČ a spracováva odpovede.

CDO

- CDO spracováva synchronizačné správy od PaM za účelom zabezpečenia aktuálnosti údajov o zamestnancoch v CDO.
- CDO obsluhuje požiadavky na generovanie UOČ a posiela príslušnú odpoveď.

B.5.3 Posielané správy a realizácia komunikácie

Komunikácia CDO a systému PaM bude implementovaná použitím nasledujúcich správ. Detailná definícia správ spolu s popisom logiky generovania požiadaviek a prijímania odpovedí na ne nasleduje ďalej.

Synchronizácia údajov (prenos údajov o zamestnancoch do CDO)

`sk.uniba.cdo.messages.PaMExport`[20]

Správa obsahuje údaje o zamestnancoch, ktoré majú byť prenesené do CDO.

Generovanie UOČ

`sk.uniba.cdo.messages.PaMGenerateUOCRequest`

Správa obsahuje zoznam zamestnancov, pre ktorých systém PaM požaduje vygenerovanie UOČ.

`sk.uniba.cdo.messages.PaMGenerateUOCReply`

Správa obsahuje zoznam požadovaných vygenerovaných UOČ. CDO posiela túto správu ako odpoveď na požiadavku.

Tento pohľad na komunikáciu CDO so systémom PaM je len logický. Lepšie a prehľadnejšie dokumentuje funkcie, ktoré sú od rozhrania medzi týmito systémami očakávané. Pri skutočnej fyzickej realizácii však požadujeme jednoduchosť, efektívnosť a čo možno najmenší dopad na integrovaný subsystém. Preto sa reálne posiela zo systému PaM len jeden typ správy, ktorý obsahuje všetky atribúty potrebné na to, aby CDO mohla vykonať všetky požadované funkcie. Túto správu označíme ako `sk.uniba.cdo.messages.PaMRequest`.

CDO posiela ako odpoveď len jeden typ správy. Formálne ju pomenujeme `sk.uniba.cdo.messages.PaMReply`. Je to správa, ktorá obsahuje tie isté atribúty a záznamy ako pôvodná požiadavka zo systému PaM s tým, že zamestnanci, pre ktorých bolo vygenerované UOČ, ho v tejto správe majú doplnené.

Údaje sa z PaM do CDO prenášajú vo forme DBF súborov. Naopak, z CDO do PaM, sú údaje prenášané formou CSV súborov.

Fyzická realizácia rozhrania:

Približne raz za mesiac sa zo všetkých fakúlt zoberú databázy zamestnancov vo forme DBF súborov. V súčasnej dobe sú údaje v týchto databázach v nenormalizovanom tvare, preto jeden pracovník CIT manuálne doplní do každého DBF súboru 3 atribúty – meno, priezvisko, UOČ. Tieto hodnoty získa zo samotného DBF súboru, ale postará sa o „normalizáciu“ dát do správneho formátu. Ide predovšetkým o doplnenie slovenskej diakritiky, očistenie údajov od rôznych poznámok a podobne. Následne sa tieto databázy pošlú správcovi centrálnej databázy osôb, ktorý zaistí ich naimportovanie do CDO a taktiež vygenerovanie nových UOČ pre zamestnancov, ktorí ho ešte nemajú. Vygenerované UOČ sú potom príslušnými pracovníkmi CIT nahraté do pôvodných fakultných databáz.

B.5.3.1 Logika generovania správ

`sk.uniba.cdo.messages.PaMExport`

[20] konvencia: štandardné pomenovanie používané v jave, cdo = názov projektu, slovo za poslednou bodkou je názov správy, pričom prvé slovo je meno systému a zvyšok označuje funkciu správy

Systém PaM generuje túto správu, ak je potrebné vyexportovať údaje o zamestnancoch a poslať ich do CDO. Tento prenos sa uskutočňuje v určených časoch (približne raz mesačne).

Táto správa vždy obsahuje celú databázu danej inštancie systému Mzdy, t.j., údaje o všetkých aktuálnych zamestnancoch na danej fakulte, ktorí už majú pridelené UOČ.

Údaje sa do CDO importujú tak ako sú, nie sú aplikované žiadne transformácie (nanajvýš ak normalizácia hodnôt niektorých atribútov). V CDO je teda presná kópia údajov v systéme PaM.

Atribúty posielaného súboru sú nasledovné:

Názov	**Typ**	**Veľkosť**	**Význam**	**Neprázdne**
databaza	char	10	názov zdrojovej databázy (názov DBF súboru)	A
oscisl	char	10	osobné číslo zamestnanca (ORIG_ID)	A
stred	char	10	katedra (ORIG_Pracovisko)	N
meno	char	50	krstné meno (normalizované, doplnené manuálne)	N
priezvi	char	50	priezvisko (normalizované, doplnené manuálne)	N
cislo	char	10	UOČ (normalizované, doplnené manuálne)	N
priez	char	100	krstné meno a priezvisko (formát: M+“ „+P)	A
adrmat	char	10	UOČ	A
povpriez	char	50	rodné priezvisko	N
titul	char	25	tituly pred menom	A
vedhod	char	25	tituly za menom	A
dtnar	date	10	dátum narodenia	A
miestnarn	char	100	miesto narodenia	N
rodcis	char	10	rodné číslo	A
druhpv	number	2	druh pracovno-právneho vzťahu	A
dtvzpv	date	10	dátum vzniku pracovno-právneho vzťahu	A
dtukpv	date	10	dátum ukončenia pracovno-právneho vzťahu	A
trvpoba	char	100	adresa trvalého pobytu	N
pohlav	char	1	pohlavie	A

`sk.uniba.cdo.messages.PaMGenerateUOCRequest`

Tento typ správy generuje systém PaM v prípade, že potrebuje vygenerovať UOČ pre nejakých zamestnancov.

V správe je uvedený zoznam zamestnancov, pre ktorých je generovanie UOČ požadované. Pre každého zamestnanca musia byť uvedené relevantné atribúty, ktoré CDO používa v algoritme generovania UOČ.

Formát posielaného súboru je v nasledujúcej tabuľke.

Názov	**Typ**	**Veľkosť**	**Význam**	**Neprázdne**
databaza	char	10	názov zdrojovej databázy (názov DBF súboru)	A
oscisl	char	10	osobné číslo zamestnanca	A

rodcis	char	10	rodné číslo	A
meno	char	50	krstné meno (normalizované, doplnené manuálne)	N
priezvi	char	50	priezvisko (normalizované, doplnené manuálne)	N
priez	char	100	krstné meno a priezvisko (formát: M+“ „+P)	A
povpriez	char	50	rodné priezvisko	N
dtnar	date	10	dátum narodenia	A
miestnarn	char	100	miesto narodenia	N

B.5.3.1.1 Mapovanie `PaMRequest` na logické správy

Názov	**PaMExport**	**PaMGenerateUOCRequest**
databaza	X	X
oscisl	X	X
stred	X	
meno	X	X
priezvi	X	X
cislo	X	
priez	X	X
adrmat	X	
povpriez	X	X
titul	X	
vedhod	X	
dtnar	X	X
miestnarn	X	X
rodcis	X	X
druhpv	X	
dtvzpv	X	
dtukpv	X	
trvpoba	X	
pohlav	X	

B.5.3.2 Logika prijímania odpovedí na požiadavky

V tejto časti popíšeme odpoveď generovanú CDO, ktorú musí byť systém PaM schopný spracovať, ako aj spôsob jej spracovania.

`sk.uniba.cdo.messages.PaMGenerateUOCReply`

Túto správu generuje CDO ako odpoveď pri funkcii generovania UOČ pre zamestnancov. UOČ nie je uvedené, ak ho z nejakých príčin nebolo možné vygenerovať.

Systém PaM musí túto správu prijať a spracovať adekvátnym spôsobom. Formát prenášaných údajov je CSV. Kvôli uľahčeniu spracovania sa v tejto správe nachádzajú aj nadbytočné atribúty.

Názov	**Typ**	**Veľkosť**	**Význam**	**Neprázdne**
databaza	char	10	názov zdrojovej databázy (názov DBF súboru)	A
oscisl	char	10	osobné číslo zamestnanca	A
uoc	char	10	vygenerované UOČ	N
stred	char	10	katedra (ORIG_Pracovisko)	N
meno	char	50	krstné meno	N

priezvi	char	50	priezvisko	N

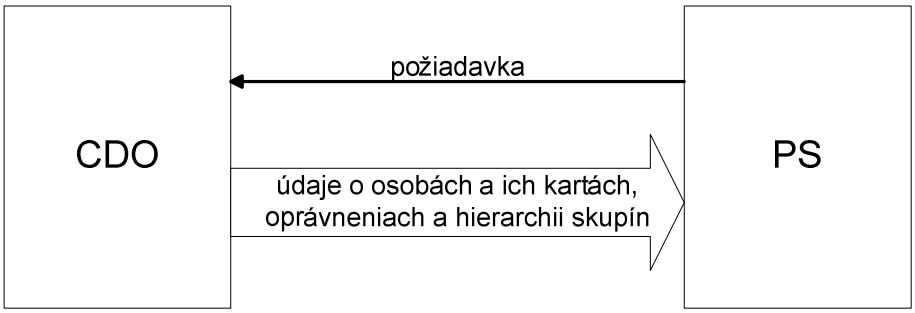
B.6 Prístupový systém UK a Centrálna databáza osôb

B.6.1 Prístupový systém (PS)

Prístupový systém je systém na automatizované odomykanie dverí oprávneným osobám na základe priloženia bezkontaktnej čipovej karty. Tento systém pre svoje správne fungovanie potrebuje mať k dispozícii základné informácie o osobách a ich preukazoch. Ďalej potrebuje poznať zaradenie jednotlivých osôb do skupín určujúcich oprávnenia na vstup. Navyše existuje hierarchia skupín, ktorú prístupový systém musí poznať, aby ju napríklad vedel prezentovať operátorovi systému. Požaduje sa flexibilnosť pri definovaní skupín a pravidiel, kto do nich patrí.

B.6.2 Tok údajov

Táto časť obsahuje len rámcový popis interakcie medzi integrovanými systémami.



PS

- Posiela správu s požiadavkou na zaslanie údajov.
- Prijíma správu:
  - s údajmi o osobách a číslach ich preukazov,
  - s definíciami skupín osôb,
  - s definíciou hierarchie skupín.

CDO

- Prijíma požiadavku od PS na zaslanie údajov, ktorú musí obslúžiť.
- Posiela údaje požadované PS. (viď predchádzajúce tri body)

B.6.3 Posielané správy a realizácia komunikácie

Komunikácia CDO s PS je obojsmerná, založená na mechanizme „request-reply“. To znamená, že adaptér na strane PS rozhoduje o tom, kedy budú údaje posielané. Údaje sa presúvajú z CDO do PS. Správa, ktorú posiela PS (presnejšie, adaptér na strane PS), je len prázdnou správou. Jej poslanie je pre CDO indikáciou faktu, že je požadovaný prenos údajov. Z formálnych dôvodov zadefinujeme meno tejto správy ako `sk.uniba.cdo.messages.PSDataRequest`.

Nasleduje logický pohľad na komunikáciu CDO s PS, zadefinujeme správy použité na realizáciu tejto komunikácie.

Prenos údajov o osobách a ich preukazoch do PS

`sk.uniba.cdo.messages.PSUsersExport`

Táto správa obsahuje zoznam osôb, ktoré budú využívať služby prístupového systému. Pre každú osobu sú uvedené základné atribúty a číslo preukazu.

Prenos údajov o skupinách osôb

`sk.uniba.cdo.messages.PSGroupsExport`

Táto správa obsahuje definíciu skupín. Pre každú skupinu je uvedený názov a zoznam osôb, ktoré do nej patria.

`sk.uniba.cdo.messages.PSGroupsHierarchyExport`

Táto správa obsahuje definíciu hierarchie skupín.

Údaje sa medzi týmito systémami prenášajú vo forme CSV súborov, pričom:

- kódovanie je Windows-1250,
- neobsahuje hlavičku,
- oddeľovač je bodkočiarka,
- jednotlivé hodnoty nie sú v úvodzovkách.

B.6.3.1 Logika generovania požiadaviek

`sk.uniba.cdo.messages.PSDataRequest`

Prístupový systém generuje túto správu ako požiadavku na zaslanie údajov z CDO. Táto správa má prázdne telo.

B.6.3.2 Logika prijímania údajov

`sk.uniba.cdo.messages.PSUsersExport`

CDO v tejto správe uvedie všetky osoby, ktoré majú aktívnu kartu a súčasne majú otvorený vzťah k univerzite (študentský alebo zamestnanecký).

Atribúty vytvoreného CSV súboru sú nasledovné:

Názov	**Typ**	**Veľkosť**	**Význam**	**Neprázdne**
UOC	char	10	univerzitné osobné číslo	A
SNR	char	11	sériové číslo karty	A
Titul1	char	15	tituly pre menom	A
Meno	char	25	meno	A
Priezvisko	char	25	priezvisko	A
Titul2	char	15	tituly za menom	A
PrimVztah	char	20	primárny vzťah k univerzite „Zamestnanec“ „Študent“ „Interný doktorand“ „Externý študent“	A
SkolaFakulta	char	50	úplný názov fakulty	A

`sk.uniba.cdo.messages.PSGroupsExport`

V tejto správe je uvedený zoznam skupín spoločne s osobami, ktoré do nich patria. Jedna osoba môže súčasne patriť do viacerých skupín.

Atribúty posielaného CSV súboru sú:

Názov	Typ	Veľkosť	Význam	Neprázdne
skupina	char	100	názov skupiny	A
UOC	char	10	UOČ osoby, ktorá patrí do tejto skupiny	A

`sk.uniba.cdo.messages.PSGroupsHierarchyExport`

CDO v tejto správe posiela definíciu hierarchie skupín osôb. Graf hierarchie je strom, ktorý popisujeme dvojicami tvaru (otec, syn).

Atribúty posielaného CSV súboru sú:

Názov	Typ	Veľkosť	Význam	Neprázdne
otec	char	100	názov nadradenej skupiny	A
syn	char	100	názov skupiny, ktorá je v hierarchii umiestnená hneď pod uvedenou nadradenou skupinou	A

B.7 Systém Študent a Centrálna databáza osôb

B.7.1 Systém Študent

Systém Študent je aplikácia na podporu evidencie študentov. Na študijnom oddelení každej fakulty je v prevádzke jedna inštancia tohto systému, spolu je ich teda 13.

Údaje o študentoch sa do tohto systému dostávajú manuálnym zadávaním referentkami. Určitou výnimkou je len prideľovanie UOČ študentom. Tieto údaje v súčasnosti importuje do jednotlivých inštancií správca systému Študent. V budúcnosti bude samozrejme tento proces automatický a nebude vyžadovať prácu správcu.

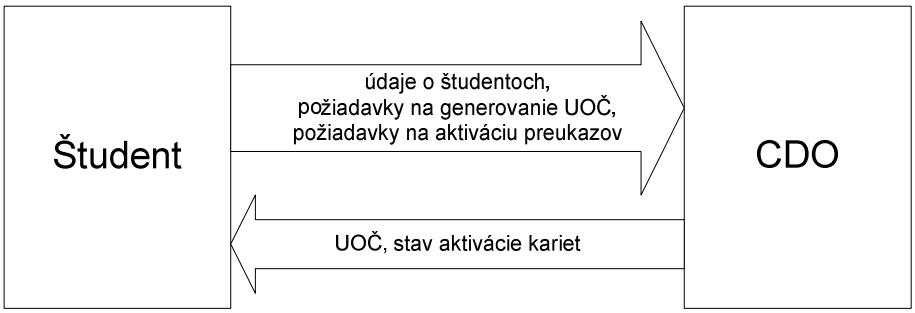
Študent je teda primárnym zdrojom údajov o študentoch. Z hľadiska integrácie má zmysel poskytovať tieto údaje iným systémom, prostredníctvom CDO.

Na druhej strane, Študent požaduje od CDO nasledujúce služby:

- generovanie UOČ pre študentov,
- aktiváciu preukazov študentov.

B.7.2 Tok údajov

Táto časť obsahuje len rámcový popis interakcie medzi integrovanými systémami.



Študent

- Ako primárny zdroj údajov o študentoch musí Študent pravidelne (a prípadne pri nejakej zmene v údajoch) posielať synchronizačné správy, aby v CDO boli vždy aktuálne údaje.
- Študent posiela CDO požiadavky na vygenerovanie UOČ a spracováva odpovede.
- Podobne, Študent generuje požiadavky na aktiváciu študentských preukazov a spracováva odpovede od CDO.

CDO

- CDO spracováva synchronizačné správy od Študenta za účelom udržiavania aktuálnych údajov o študentoch.
- CDO obsluhuje požiadavky na generovanie UOČ a na aktiváciu preukazov a posiela príslušné odpovede.

B.7.3 Posielané správy a realizácia komunikácie

Komunikácia CDO a systému Študent bude implementovaná použitím nasledujúcich správ. Detailná definícia správ spolu s popisom logiky generovania požiadaviek a prijímania odpovedí na ne nasleduje ďalej.

Synchronizácia údajov (prenos údajov o študentoch do CDO)

`sk.uniba.cdo.messages.StudentExport`[21]

Správa obsahuje údaje o študentoch, ktoré majú byť prenesené do CDO.

Generovanie UOČ

`sk.uniba.cdo.messages.StudentGenerateUOCRequest`

Správa obsahuje údaje o študentoch, pre ktorých systém požaduje vygenerovanie UOČ.

`sk.uniba.cdo.messages.StudentGenerateUOCReply`

Správa obsahuje zoznam požadovaných vygenerovaných UOČ, ako odpoveď na požiadavku.

Aktivácia preukazov

`sk.uniba.cdo.messages.StudentActivateCardRequest`

Správa obsahuje údaje o študentoch, pre ktorých systém Študent požaduje aktiváciu univerzitných preukazov.

`sk.uniba.cdo.messages.StudentActivateCardReply`

Správa obsahuje informáciu o úspešnosti aktivácie preukazov študentov, pre ktorých to bolo požadované v požiadavke.

Toto je logický pohľad na prenášané správy. Pri fyzickej realizácii komunikácie sa používa len jeden typ správy namiesto trojice správ StudentExport, StudentGenerateUOCRequest, StudentActivateCardRequest.

Fyzickú správu nazývame `sk.uniba.cdo.messages.StudentRequest` a množina jej atribútov je zjednotením atribútov v spomínaných troch správach.

[21] konvencia: štandardné pomenovanie používané v jave, cdo = názov projektu, slovo za poslednou bodkou je názov správy, pričom prvé slovo je meno systému a zvyšok označuje funkciu správy

Podobne, pri fyzickej realizácii, CDO posiela jeden typ správy `sk.uniba.cdo.messages.StudentReply`.

Údaje sa medzi týmito systémami prenášajú:

- vo formáte CSV,
- kódovanie je Windows-1250,
- oddeľovačom hodnôt je čiarka,
- úvodzovky ohraničujúce hodnoty atribútov sú povinné,
- CSV hlavička je povinná.

B.7.3.1 Logika generovanie požiadaviek

`sk.uniba.cdo.messages.StudentExport`

Systém Študent generuje túto správu, ak je potrebné vyexportovať údaje o študentoch a poslať ich do CDO. Toto sa deje jednak automaticky v určených časoch alebo manuálne na požiadanie študijnej referentky.

Táto správa obsahuje celú databázu študentov jednej inštancie systému Študent, t.j., údaje o všetkých zapísaných študentoch na príslušnej fakulte v aktuálnom akademickom roku, ktorí už majú pridelené UOČ.

Údaje sa do CDO importujú tak ako sú, nie sú aplikované žiadne transformácie (nanajvýš ak normalizácia hodnôt niektorých atribútov). V CDO je teda presná kópia údajov v systéme Študent.

Atribúty posielaného CSV súboru sú nasledovné:

Názov	**Typ**[22]	**Veľkosť**	**Význam**	**Neprázdne**
uks	char	8	univerzitný kód študenta (prvé 2 znaky je fakulta v zmysle nižšie uvedenej tabuľky, zvyšok je kód študenta v rámci fakulty)	A
ttlp	char	15	titul(y) pred menom	N
meno	char	25	krstné meno	N
przv	char	25	priezvisko	A
rodm	char	25	rodné priezvisko	N
ttlz	char	15	titul(y) za menom	N
rcis	char	10	rodné číslo[23]	A
dnar	date	10	dátum narodenia	A
mnar	char	25	miesto narodenia	N
statp	char	50	štátna príslušnosť	A
ulcd	char	25	trvalé bydlisko – ulica a číslo domu	A[24]
psc	char	6	trvalé bydlisko – PSČ	A
obec	char	25	trvalé bydlisko – názov obce + označenie pošty	A
rnvs	char	4	rok nástupu na vysokú školu	N
kspg	char	10	kód študijného programu	A
drst	char#	1	druh štúdia	A

[22] # = pre daný atribút existuje číselník
[23] v tvare 9999999999, s tým, že niekedy posledné štvorčíslie chýba
[24] v súlade s neformálnymi podmienkami EMcard musí byť zadané minimálne buď ulcd a obec, alebo psc a obec

fost	char#	1	forma štúdia	A
dzap	date	10	dátum zápisu (v aktuálnom roku)	N
atst	char	1	atribút štúdia (typicky dôvod ukončenia štúdia)	N
duks	date	10	dátum ukončenia štúdia	N
dpod	date	10	dátum začiatku prerušenia štúdia	N
dpdo	date	10	dátum konca prerušenia štúdia	N
cups	char	10	číslo preukazu (ako ho evidujú na študijnom oddelení)	N
vups	char#	1	spôsob validácie preukazu	N
isic	char	10	číslo ISIC, ako ho evidujú na študijnom oddelení	N
vxsb	char	2	súhlasy s poskytnutím/zverejnením údajov (podľa formulára, ktorý sa odovzdáva pri zápise)[25]	N
vxsc	char	4		N
uoc	char	10	univerzitné osobné číslo študenta	A

Dohodnuté číselníky pre niektoré atribúty:

Fakulta (prvé 2 znaky UKS)	
Kód	**Význam**
51	Lekárska fakulta UK
52	Jesseniova lekárska fakulta UK (Martin)
55	Farmaceutická fakulta UK
61	Právnická fakulta UK
64	Filozofická fakulta UK
14	Prírodovedecká fakulta UK
11	Fakulta matematiky, fyziky a informatiky UK
74	Fakulta telesnej výchovy a športu UK
71	Pedagogická fakulta UK
31	Fakulta managementu UK
02	Evanjelická bohoslovecká fakulta UK
01	Rímskokatolícka cyrilo-metodská bohoslovecká fakulta UK
68	Fakulta sociálnych a ekonomických vied

DruhStudia	
Kód	**Význam**
M	magisterské
B	bakalárske
D	doktorské
P	doktorandské

FormaStudia	
Kód	**Význam**
d	denná
e	externá

[25] viď prílohu tejto špecifikácie

Validacia	
Kód	**Význam**
i	predlžovacia známka ISIC
u	univerzitná predlžovacia známka
f	karta akceptovaná z inej fakulty UK
NULL	validácia zatiaľ neprebehla

`sk.uniba.cdo.messages.StudentGenerateUOCRequest`

Tento typ správy generuje systém Študent v prípade, že potrebuje vygenerovať UOČ pre nejakých študentov.

V správe je uvedený zoznam študentov, pre ktorých je generovanie UOČ požadované. Pre každého študenta musia byť uvedené relevantné atribúty, ktoré CDO používa v algoritme generovania UOČ. Formát posielaného CSV súboru je v nasledujúcej tabuľke.

Názov	Typ	Veľkosť	Význam	Neprázdne
uks	char	8	univerzitný kód študenta	A
ttlp	char	15	titul(y) pred menom	N
meno	char	25	krstné meno	N
przv	char	25	priezvisko	A
rodm	char	25	rodné priezvisko	N
ttlz	char	15	titul(y) za menom	N
rcis	char	10	rodné číslo	A
dnar	date	10	dátum narodenia	A
mnar	char	25	miesto narodenia	N
statp	char	50	štátna príslušnosť	N

`sk.uniba.cdo.messages.StudentActivateCardRequest`

Systém Študent eviduje pre každého študenta aj číslo jeho študentského preukazu, ak mu bol vydaný. Aby preukaz mohol byť používaný, musí sa najprv aktivovať v CDO. Požiadavku na aktiváciu karty posiela študijné oddelenie po tom, čo skontrolovalo, že preukaz ozaj patrí príslušnému študentovi a môže ho používať.

V tejto správe systém Študent uvedie zoznam študentov, pre ktorých požaduje aktiváciu karty. Posielaný CSV súbor obsahuje nasledovné atribúty.

Názov	Typ	Veľkosť	Význam	Neprázdne
uks	char	8	univerzitný kód študenta	A
cups	char	10	číslo preukazu (ako ho evidujú na študijnom oddelení)	A
vups	char#	1	spôsob validácie preukazu	A
isic	char	10	číslo ISIC (ako ho evidujú na študijnom oddelení)	N
dpak	date	10	dátum vzniku požiadavky na aktiváciu (NULL ak aktivácia nie je požadovaná)	N
uoc	char	10	univerzitné osobné číslo študenta	A

B.7.3.1.1 Mapovanie `StudentRequest` na logické správy

Názov	StudentExport	StudentActivateCardRequest	StudentGenerateUOCRequest
uks	X	X	X
ttlp	X		X
meno	X		X
przv	X		X
rodm	X		X
ttlz	X		X
rcis	X		X
dnar	X		X
mnar	X		X
ulcd	X		
psc	X		
obec	X		
rnvs	X		
kspg	X		
drst	X		
fost	X		
dzap	X		
atst	X		
duks	X		
dpod	X		
dpdo	X		
cups	X	X	
vups	X	X	
isic	X	X	
vxsb	X		
vxsc	X		
uoc	X	X	
dpak		X	
statp			X

B.7.3.2 Logika prijímania odpovedí na požiadavky

V tejto časti popíšeme odpovede generované CDO, ktoré musí byť systém Študent schopný spracovať, ako aj požadovaný spôsob spracovania.

`sk.uniba.cdo.messages.StudentGenerateUOCReply`

Túto správu generuje CDO ako odpoveď pri funkcii generovania UOČ pre študentov. UOČ nie je uvedené, ak ho z nejakých príčin nebolo možné vygenerovať resp. zistiť.

Systém Študent musí túto správu prijať a spracovať adekvátnym spôsobom.

Atribúty posielaného CSV sú v tabuľke.

Názov	Typ	Veľkosť	Význam	Neprázdne
uks	char	8	univerzitný kód študenta	A
uoc	char	10	vygenerované/zistené UOČ	N

`sk.uniba.cdo.messages.StudentActivateCardReply`

Tento typ správy posiela CDO systému Študent ako odpoveď pri funkcii aktivácie preukazov pre študentov. Atribút „stav“ obsahuje textovú správu pre používateľa systému Študent, v ktorej je uvedené, či sa daný preukaz podarilo aktivovať a ak nie, tak z akých dôvodov.

Systém Študent musí vedieť túto správu spracovať a zobraziť informáciu uvedenú v atribúte „stav“ používateľovi. V tabuľke je opäť zoznam atribútov posielaného CSV súboru.

Názov	**Typ**	**Veľkosť**	**Význam**	**Neprázdne**
uks	char	8	univerzitný kód študenta	A
uoc	char	10	UOČ	A
cups	char	10	číslo aktivovaného preukazu[26]	A
isic	char	10	číslo licencie ISIC	N
stav	char	150	správa pre používateľa systému Študent	N

Popis významu atribútu DPAK a práce s ním

Atribút *dpak* sa používa na signalizáciu toho, či systém Študent požaduje aktiváciu preukazu. CDO si v databáze pamätá dátum poslednej aktivácie preukazu pre každého študenta, ktorého eviduje. Porovnaním tohto uloženého atribútu s atribútom *dpak* získaným z tela správy vie CDO určiť, či je potrebná aktivácia.

CDO pracuje s hodnotou *dpak* získanou zo správy nasledovne:

- Ak *dpak* je NULL, potom CDO nevykoná aktiváciu preukazu pre príslušného študenta.
- Ak *dpak* nie je NULL, potom je jeho hodnota porovnaná s atribútom uloženým v CDO. V prípade, že je *dpak* v správe aktuálnejší, CDO vykoná aktiváciu preukazu. Ak aktivácia prebehne úspešne, CDO zapíše aktuálny *dpak* do svojej databázy. V atribúte *stav* vráti informáciu o úspešnosti aktivácie, prípadne dôvod neúspechu.

Na strane systému Študent je predpokladané spracovanie atribútu *dpak* nasledovné:

- Študent nastaví *dpak* na aktuálny dátum, ak požaduje aktiváciu preukazu pre príslušného študenta.
- Na základe odpovede od CDO (konkrétne z atribútu *stav*) sa Študent rozhodne, či vynuluje hodnotu *dpak* alebo nie.
- Ak aktivácia prebehla úspešne, potom Študent vynuluje atribút *dpak*, teda v ďalších posielaných správach už nebude požadovaná aktivácia preukazu.
- Ak aktivácia neprebehla úspešne, dôvod neúspechu je popísaný v atribúte *stav*. Túto informáciu zobrazí Študent používateľovi a nechá *dpak* nastavený. Pri ďalšom posielaní údajov bude v *dpak* opäť aktuálny dátum.

[26] v prípade neúspechu aktivácie je to číslo preukazu tak, ako ho CDO prijalo zo systému Študent

B.8 Ubytovací softvér a Centrálna databáza osôb

B.8.1 Ubytovací softvér (UBYT)

V rámci UK existujú tri vysokoškolské internáty. Fungujú nezávisle na sebe a na každom z internátov je prevádzkovaná nejaká aplikácia na evidovanie ubytovaných osôb (čo nie sú nutne len študenti) a samotných ubytovacích kapacít. Jedná sa teda o tri rôzne ubytovacie aplikácie, ale pre potreby integrácie s inými systémami (CDO, HDSW) možno od tohto faktu abstrahovať, pokiaľ všetky tri aplikácie dokážu poskytnúť požadované údaje v spoločnom formáte.

Na internátoch sú okrem študentov a zamestnancov univerzity ubytovaní aj iné osoby, ktoré nemajú žiadny vzťah k univerzite. Údaje o týchto osobách nebudú prenášané z ubytovacej aplikácie do CDO. Hlavne z dôvodu identifikácie týchto osôb, keďže nemajú pridelené UOČ.

Ubytovacia aplikácia je primárnym zdrojom údajov o ubytovaných osobách, predovšetkým sú to údaje o dobe a konkrétnom mieste ubytovania.

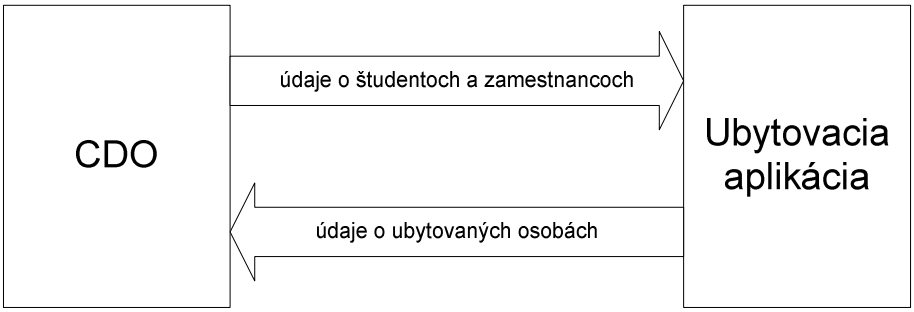
V koncepčného hľadiska potrebujeme zrealizovať nasledovné prepojenia:

- automatický prenos údajov o študentoch zo systému Študent do UBYT
- automatický prenos údajov o zamestnancov zo systému PaM do UBYT
- automatický prenos informácií o ubytovaní osôb z UBYT do CDO
- automatický prenos údajov o ubytovaných osobách do HDSW (verzia pre ubytovaných)[27]

Keďže CDO je hlavným integračným prvkom, všetky údaje sa prenášajú cez CDO a nie priamo medzi integrovanými systémami. V tomto prípade, systém Študent, PaM, UBYT a HDSW.

B.8.2 Tok údajov

Táto časť obsahuje popis interakcie medzi integrovanými systémami len na hrubej úrovni, kvôli získaniu prehľadu.



Ubytovacia aplikácia

- Ako primárny zdroj informácií o ubytovaní osôb musí UBYT pravidelne (a prípadne pri nejakej zmene v údajoch) posielať synchronizačné správy.
- Prijíma od CDO správy obsahujúce základné údaje o študentoch a zamestnancoch.

CDO

[27] popis rozhrania s HDSW a prenosu údajov do HDSW je v samostatnom dokumente

- Spracováva synchronizačné správy od ubytovacej aplikácie, aby mala čo najaktuálnejšie údaje a mohla ich ďalej poskytovať.
- Posiela údaje o študentoch a zamestnancoch, ktoré sa do CDO dostali z primárnych zdrojov týchto údajov[28], do ubytovacej aplikácie.

B.8.3 Posielané správy a realizácia komunikácie

Komunikácia CDO a ubytovacej aplikácie bude implementovaná použitím nasledujúcich správ, detailná definícia správ spolu s popisom logiky generovania požiadaviek a prijímania odpovedí na ne nasleduje ďalej.

Synchronizácia údajov (prenos údajov o osobách z CDO do UBYT)

`sk.uniba.cdo.messages.UBYTExport`[29]

Správa obsahuje zoznam osôb, ktoré majú byť prenesené a naimportované do ubytovacej aplikácie.

Synchronizácia údajov (prenos údajov o ubytovaní z UBYT do CDO)

`sk.uniba.cdo.messages.UBYTImport`

Správa obsahuje zoznam ubytovaných osôb, v ktorom sú uvedené základné atribúty o ich ubytovaní.

Formát prenášaných údajov je nasledovný:

- textový súbor v kódovaní Windows-1250,
- formát CSV,
- oddeľovač hodnôt je čiarka,
- úvodzovky sú povinné,
- CSV hlavička je povinná.

Tento formát bol vybraný kvôli jeho najmenšiemu dopadu na ubytovacie aplikácie, keďže už v súčasnosti sú schopné pracovať so súbormi v CSV formáte. Zároveň toto rozhodnutie prispeje k rýchlejšej implementácii tohto komunikačného rozhrania.

B.8.3.1 Logika generovania správ

`sk.uniba.cdo.messages.UBYTImport`

Túto synchronizačnú správu generuje ubytovacia aplikácia v pravidelných časových intervaloch a je v nej uvedený zoznam ubytovaných osôb spolu so základnými atribútmi o ich ubytovaní. Uvádzané sú len univerzitné osoby, ktoré majú pridelené UOČ. Mimouniverzitné osoby ubytované na internáte v posielanom zozname osôb nie sú.

CDO musí vedieť takúto správu spracovať a uchovať tieto údaje vo svojej vnútornej databáze, aby ich mohla poskytnúť iným systémom (napr. HDSW).

Atribúty posielaného CSV súboru sú nasledovné:

Názov	Typ	Veľkosť	Význam	Neprázdne
uoc	char	10	univerzitné osobné číslo	A

[28] systém Študent, Personalistika a Mzdy

[29] konvencia: štandardné pomenovanie používané v jave, cdo = názov celého projektu, slovo za poslednou bodkou je samotný názov správy, pričom prvé slovo je meno systému a zvyšok označuje funkciu

ubytovanyOd	date	10	dátum začiatku ubytovania	A
ubytovanyDo	date	10	dátum konca ubytovania	A
budova	char	20	miesto ubytovania – budova	A
blok	char	20	miesto ubytovania – blok	A
izba	char	20	miesto ubytovania – izba	A

B.8.3.2 Logika prijímania správ

`sk.uniba.cdo.messages.UBYTExport`

Túto správu generuje CDO v prípade, že chce poslať aktuálne údaje o osobách do ubytovacej aplikácie. Správa teda obsahuje zoznam aktuálnych študentov a zamestnancov, ktorí majú otvorený vzťah k univerzite.

Prostredníctvom tejto správy sa zároveň rieši problém prenosu UOČ čísiel do lokálnych databáz ubytovacích aplikácií.

Ubytovacia aplikácia musí túto správu prijať a spracovať adekvátnym spôsobom.

Atribúty posielaného CSV sú v tabuľke.

Názov	**Typ**	**Veľkosť**	**Význam**	**Neprázdne**
uoc	char	10	univerzitné osobné číslo	A
titulypred	char	15	titul(y) pred menom	N
meno	char	25	krstné meno	A
priezvisko	char	25	priezvisko	A
rodnep	char	25	rodné priezvisko	N
titulyza	char	15	titul(y) za menom	N
rodnecislo[30]	char	10	rodné číslo	N
miestonar	char	50	miesto narodenia	A
pohlavie	char	1	pohlavie („M“ = muž, „Z“ = žena)	A
cisloopp	char	15	číslo občianskeho preukazu / pasu	A
ulica	char	25	trvalé bydlisko – ulica a číslo domu	A
psc	char	6	trvalé bydlisko – PSČ	A
mesto	char	25	trvalé bydlisko – názov obce + označenie pošty	A
statp	char	50	štátna príslušnosť	A
vztah	char	1	vzťah k univerzite („S“ = študent, „Z“ = zamestnanec)	A
sucast	char	50	súčasť UK (tá, z titulu ktorej bolo osobe pridelené ubytovanie na internáte)	A

B.9 Univerzitný validačný terminál a Centrálna databáza osôb

B.9.1 Univerzitný validačný terminál (UT)

Univerzitný terminál je zariadenie, ktoré slúži všetkým držiteľom preukazov na UK. Zamestnanci a študenti môžu priložením preukazu zistiť aké údaje majú uložené v čipe karty. Pre študentov má terminál aj ďalšie funkcie. Na začiatku každého akademického roku musí študent prísť k validačnému terminálu, aby mu na čip karty boli

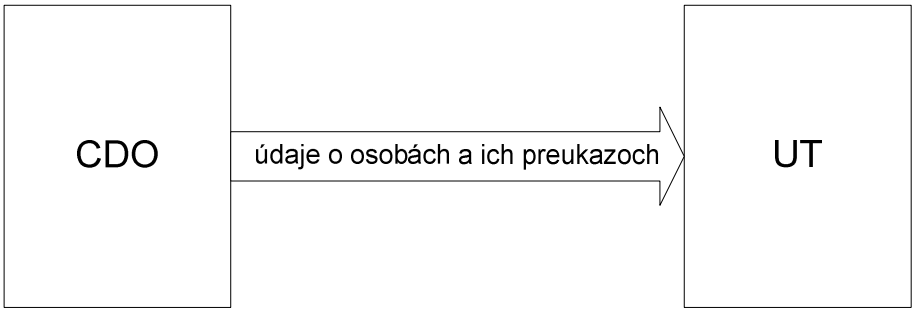
[30] v ubytovacej aplikácii internátu Družba je rodne číslo primárny kľúč tabuľky a teda sa nedá meniť !

zapísané aktuálne údaje. Študentské preukazy sú využiteľné aj ako preukážky na autobusovú a železničnú dopravu. To sú ďalšie údaje, ktoré terminál na kartu zapisuje.

Aby validačný terminál mohol správne fungovať, musí mať aktuálne informácie o študentoch a ich preukazoch. Tieto údaje sa do univerzitných terminálov dostávajú pravidelným presunom dát z CDO.

B.9.2 Tok údajov

Táto časť obsahuje len rámcový popis interakcie medzi integrovanými systémami.



UT

Prijíma údaje o študentoch, zamestnancoch a ich preukazoch z CDO, ale neposiela žiadne údaje ani požiadavky.

CDO

Generuje exportný súbor pre UT a prenáša tento súbor do každého terminálu (použitím na to určenej aplikácie EMWriteUT).

B.9.3 Posielané správy a realizácia komunikácie

Komunikácia CDO s univerzitným terminálom je jednosmerná. CDO v pravidelných intervaloch zisťuje, či nastala zmena v databáze študijných vzťahov a preukazov. V prípade, že nastala, tak sa spustí proces generovania exportného súboru pre univerzitné terminály, ktorý má špecifický formát. Tento súbor je potom prenesený do každého terminálu.

Na exportný súbor sa možno pozerať ako na správu, ktorú CDO posiela do univerzitných terminálov.

Prenos údajov o osobách a ich preukazoch do UT

`sk.uniba.cdo.messages.UTExport`

Túto správu generuje CDO v pravidelných intervaloch a obsahuje zoznam údajov o študentoch, zamestnancoch a ich preukazoch.

B.9.3.1 Logika generovania požiadaviek

Univerzitné validačné terminály negenerujú žiadne požiadavky.

B.9.3.2 Logika prijímania údajov

`sk.uniba.cdo.messages.UTExport`

Túto správu generuje CDO v pravidelných časových intervaloch, ak nastala zmena v databáze osôb a preukazov. Následne sa použitím aplikácie EMWriteUT (vyvinutá

externým dodávateľom EMcard) nahrá vygenerovaný exportný súbor do každého terminálu.

Exportný súbor obsahuje všetky platné karty také, že držiteľ danej karty má otvorený vzťah k univerzite relevantný vzhľadom k vizuálu danej karty (viď nasledujúca tabuľka).

Vizuál	**Požadovaný vzťah k univerzite**
ISIC	musí existovať aspoň jeden otvorený vzťah „denný študent“
Externý študent	musí existovať aspoň jeden otvorený vzťah „externý študent“ a nesmie existovať vzťah „denný študent“
Zamestnanec	musí existovať aspoň jeden otvorený vzťah „zamestnanec“
ITIC	musí existovať aspoň jeden otvorený vzťah „zamestnanec“
Dôchodca	musí existovať aspoň jeden otvorený vzťah „zamestnanec“

Exportný súbor:

- textový súbor s kódovaním ISO-8859-2
- každá položka musí byť zakončená nulou (0x0)
- obsahuje hlavičku
- obsahuje jeden záznam pre každého študenta (resp. preukaz)

Konkrétne dátové štruktúry použité na prenos údajov neuvádzame, pretože sú predmetom duševného vlastníctva spoločnosti EMtest.

B.10 Systém virtuálnej knižnice UK a Centrálna databáza osôb

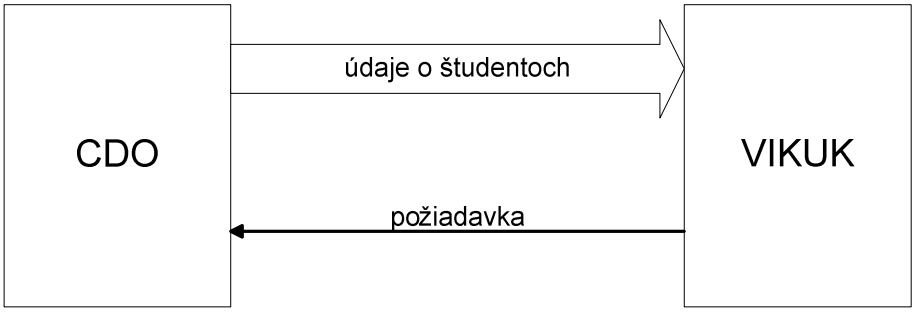
B.10.1 Systém virtuálnej knižnice UK (VIKUK)

Virtuálna knižnica UK sa zameriava na automatizáciu klasických knižničných pracovných procesov a na zabezpečenie prístupu k externým informačným zdrojom. V rámci Virtuálnej knižnice je prevádzkovaný knižnično-informačný systém VTLS/VIRTUA.

VIKUK zodpovedá za evidenciu čitateľov a ich výpožičiek. Má teda zmysel automaticky posielať niektoré údaje o študentoch do systému virtuálnej knižnice. Minimálne je to meno, priezvisko, fakulta a číslo preukazu. Taktiež pri objednávaní literatúry cez webové rozhranie slúži číslo karty ako používateľské meno pri prihlásení. Keď príde čitateľ do knižnice, zosnímaním čísla jeho preukazu (ktoré je vytlačené na preukaze ako čiarový kód) ho môžu rýchlo vyhľadať v databáze čitateľov.

B.10.2 Tok údajov

Táto časť obsahuje popis interakcie medzi integrovanými systémami len na hrubej úrovni, kvôli získaniu prehľadu.



VIKUK

Generuje požiadavky na zaslanie údajov o študijných vzťahoch a posiela ich CDO na spracovanie.

CDO

Prijíma požiadavky od systému VIKUK a po spracovaní posiela odpovede obsahujúce požadované údaje o študijných vzťahoch.

B.10.3 Posielané správy a realizácia komunikácie

Komunikácia CDO a systému Študent bude implementovaná použitím nasledujúcich správ, detailná definícia správ spolu s popisom logiky generovania požiadaviek a prijímania odpovedí na ne nasleduje ďalej.

Komunikácia medzi CDO a systémom VIKUK je založená na request-reply mechanizme. VIKUK pošle správu s požiadavkou, v ktorej je uvedený typ funkcie (full, differential). CDO túto požiadavku prijme, podľa požadovanej funkcie vygeneruje údaje a pošle ich naspať ako odpoveď na požiadavku.

Prenos údajov o študentoch a ich študijných vzťahoch do systému VIKUK

`sk.uniba.cdo.messages.VIKUKDataRequest`

Systém VIKUK pošle túto správu CDO, keď žiada o zaslanie údajov. V správe uvedie či požaduje údaje o všetkých aktuálnych študentoch alebo len zmeny, ktoré sa udiali v databáze študentských vzťahov (novopridaní, zmazaní študenti alebo študenti, pri ktorých sa zmenil nejaký atribút)[31].

`sk.uniba.cdo.messages.VIKUKDataReply`

Táto správa je odpoveďou na požiadavku o zaslanie údajov. Obsahuje zoznam informácií o študijných vzťahoch.

Údaje sa medzi týmito systémami prenášajú vo forme XML súborov s pevne definovanou štruktúrou, ktorú teraz popíšeme. Použité kódovanie je Windows-1250.

B.10.3.1 Logika generovania požiadaviek

`sk.uniba.cdo.messages.VIKUKDataRequest`

Túto správu generuje systém VIKUK, ak chce požiadať CDO o zaslanie údajov o študentoch a ich študijných vzťahoch k univerzite. Jediný atribút, ktorý v tejto správe musí uviesť, je určenie požadovanej funkcie. Teda, či žiada o kompletné údaje o študentoch alebo len o zmeny v databáze, ktoré sa udiali od posledného zaslania údajov.

[31] všetky zmeny od posledného zaslania údajov

Pre jednoduchosť a rýchlosť spracovania požiadavky je tento atribút určujúci požadovaný typ exportu uvedený v hlavičke posielanej správy a nie v tele samotnej správy. Telo správy teda zostáva prázdne.

Názov	Typ	Veľkosť	Význam	Neprázdne
differential	boolean	1	určenie požadovaného typu exportu údajov ak hodnota je „1“, potom je požadované len zaslanie zmien, ku ktorým došlo od posledného poslania údajov ak hodnota je „0“, potom sú požadované kompletné údaje	A

B.10.3.2 Logika prijímania odpovedí na požiadavky

`sk.uniba.cdo.messages.VIKUKDataReply`

Túto správu generuje CDO ako odpoveď na požiadavku o zaslanie údajov. Systém VIKUK ju musí prijať a správne spracovať.

Atribúty resp. elementy posielaného XML súboru sú nasledovné:

Nadradený element	Názov	Význam	Neprázdne
Student	UOC	univerzitné osobné číslo študenta	A
	Meno	krstné meno	A
	Priezvisko	priezvisko	A
	DatumNarodenia	dátum narodenia	A
	Adresa	trvalé bydlisko – ulica a číslo domu	A
	PSC	trvalé bydlisko – PSČ	A
	Obec	trvalé bydlisko – názov obce + označenie pošty	A
	SNR	číslo karty, ako ho evidujú na študijnom oddelení	A
	StudVztah	element obsahujúci údaje o jednom študijnom vzťahu tohto študenta	-
StudVztah	Fakulta	skratka fakulty	A
	StudProgram	kód študijného programu	A
	AtributStudia	atribút štúdia, typicky dôvod ukončenia štúdia	A[32]
	PracaObhajena	príznak, či už študent obhájil záverečnú prácu (prislúchajúcu k danému študijnému vzťahu)[33]	A
Studenti	Student	element obsahujúci informácie o jednom študentovi a jeho študijných vzťahoch	-

Príklad posielaného XML súboru:

```
<?xml version="1.0" encoding="windows-1250"?>
<Studenti>
```

[32] avšak samotná hodnota tohto atribútu v databáze môže byť prázdny reťazec
[33] ide o “boolean” atribút, prípustné hodnoty sú “0” alebo “1”

```
<Student>
<UOC>12346262</UOC>
<Meno>Jožko</Meno>
<Priezvisko>Mrkvička</Priezvisko>
<DatumNarodenia>1.2.1983</DatumNarodenia>
<Adresa>Veternicová 33</Adresa>
<PSC>84105</PSC>
<Obec>Bratislava 4</Obec>
<SNR>1216387420</SNR>
<StudVztah>
     <Fakulta>PriF</Fakulta>
     <StudProgram>3.uMACH</StudProgram>
          <AtributStudia></AtributStudia>
          <PracaObhajena>0</PracaObhajena>
</StudVztah>
<!–ďalšie štud. vzťahy–>
</Student>
<!–ďalší študenti–>
<Studenti>
```

V rámci jedného elementu `Student` môže byť viacero elementov `StudVztah`. To je prípad, keď daný študent má viac študijných vzťahov.

V prípade, že VIKUK žiada od CDO len zmeny (diferenciálny mód), je potrebné pri zmazaných záznamoch nejako indikovať túto skutočnosť. CDO v tom prípade v elemente `Student` uvedie len UOČ a žiadne iné údaje. To bude pre VIKUK znamenať, že študent s týmto UOČ bol zmazaný z centrálnej databázy a viac ho neevidujeme.

B.11 VoIP a Centrálna databáza osôb

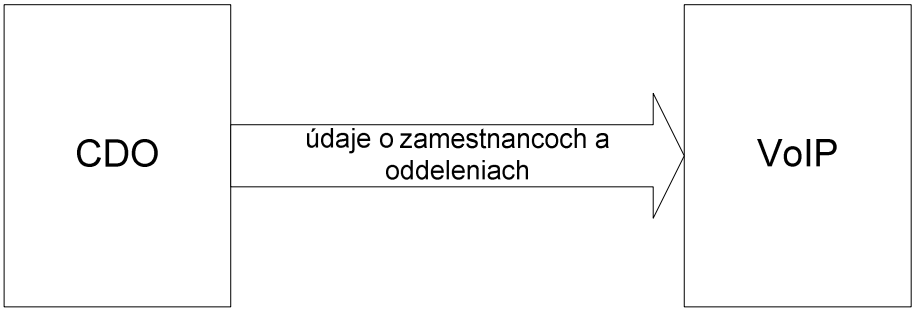
B.11.1 VoIP

Systém VoIP sa používa na evidenciu oprávnených používateľov IP telefónie a na samotné účtovanie hovorov, ktoré sa cez VoIP uskutočňujú. Používateľmi VoIP sú najmä zamestnanci univerzity.

CDO môže automatickým presunom údajov synchronizovať aktuálnu databázu zamestnancov s vnútornou databázou VoIP systému. Ďalším typom údajov, ktoré CDO môže systému VoIP poskytovať je zoznam oddelení (pracovísk) univerzity, keďže každý používateľ VoIP musí patriť do nejakého oddelenia a taktiež každé oddelenie musí mať svojho vedúceho. Všetky tieto údaje potrebuje VoIP k svojmu správnemu fungovaniu.

B.11.2 Tok údajov

Táto časť obsahuje popis interakcie medzi integrovanými systémami len na hrubej úrovni, kvôli získaniu prehľadu.



VoIP

Prijíma údaje o zamestnancoch a oddeleniach z CDO, ale neposiela žiadne údaje ani požiadavky.

CDO

Pravidelne posiela VoIP údaje o zamestnancoch a oddeleniach.

B.11.3 Posielané správy a realizácia komunikácie

Komunikácia CDO so systémom VoIP je len jednosmerná. CDO v pravidelných intervaloch exportuje požadované údaje a posiela ich do systému VoIP.

Prenášajú sa 2 druhy údajov, jednak údaje o osobách a jednak údaje o oddeleniach. Nie je však nutné definovať 2 typy správ, najmä ak sa ako formát na prenos údajov použije XML. Zadefinujeme teda jeden typ správy, ktorý v sebe bude niesť záznamy oboch typov.

Prenos údajov o osobách a oddeleniach do VoIP

`sk.uniba.cdo.messages.VoIPExport`

V tejto správe CDO posiela zoznam osôb (zamestnancov) oprávnených používať služby VoIP a súčasne zoznam pracovísk, do ktorých sú tieto osoby v rámci univerzity zaradené.

B.11.3.1 Logika generovania požiadaviek

VoIP systém negeneruje žiadne požiadavky.

B.11.3.2 Logika prijímania údajov

`sk.uniba.cdo.messages.VoIPExport`

Túto správu generuje CDO automaticky pravidelne v dohodnutých časových intervaloch. Telo správy obsahuje zoznam osôb, ktoré sú oprávnené používať VoIP systém a taktiež zoznam pracovísk týchto osôb.

Čo sa týka zamestnancov, správa obsahuje zoznam všetkých tých zamestnancov, ktorí majú platný otvorený pracovný vzťah k univerzite.

Čo sa týka pracovísk, správa obsahuje zoznam tých pracovísk, pre ktoré existuje aspoň jeden zamestnanec v uvedenom zozname zamestnancov.

Formát posielaných údajov je nasledovný:

- textový súbor s kódovaním je Windows-1250,
- vo formáte XML.

O každom zamestnancovi sú prenášané nasledovné atribúty:

Názov	Typ	Veľkosť	Význam	Neprázdne
UOČ	char	10	univerzitné osobné číslo zamestnanca	A
meno	char	100	krstné meno	A
priezvisko	char	100	priezvisko	A
prihlMeno	char	100	prihlasovacie meno zamestnanca	A
e-mail	char	100	e-mailová adresa	N
oddelenie	char	6	identifikátor oddelenia[34]	A

Podobne, o každom pracovisku sú prenášané nasledovné atribúty:

Názov	Typ	Veľkosť	Význam	Neprázdne
id	char	6	identifikátor oddelenia	A
nazov	char	100	úplný názov oddelenia	A
boss	char	100	e-mailová adresa vedúceho oddelenia	N

Štruktúra posielaného XML súboru je popísaná v nasledujúcej tabuľke.

Nadradený element	Element	Význam	Neprázdne
Person	UOC	univerzitné osobné číslo zamestnanca	A
	Name	krstné meno	A
	Surname	priezvisko	A
	Username	prihlasovacie meno	A
	Email	e-mailová adresa	N
	Department	identifikátor oddelenia (ID)	A
Department	ID	identifikátor oddelenia	A
	Name	úplný názov oddelenia	A
	Email	e-mailová adresa vedúceho oddelenia	N
Persons	Person	element obsahujúci údaje o jednej osobe	-
Departments	Department	element obsahujúci údaje o jednom oddelení	-

Príklad posielaného XML súboru:

```
<?xml version="1.0" encoding="windows-1250"?>
<Persons>
     <Person>
     <UOC>12346262</UOC>
     <Name>Jožko</Name>
     <Surname>Mrkvička</Surname>
     <Username>Mrkvicka1</Username>
     <Email>Jozko.Mrvicka@rec.uniba.sk</Email>
     <Department>10101</Department>
     </Person>
     <!—more persons–>
</Persons>
<Departments>
```

[34] oddelenie = pracovisko zamestnanca; v prípade, že má zamestnanec viac vzťahov k univerzite, tak sa tu uvedie pracovisko prislúchajúce jeho primárnemu vzťahu

```
<Department>
     <ID>10101</ID>
     <Name>Centrum informačných technológií UK</Name>
     <Email>Domany@rec.uniba.sk</Email>
</Department>
     <!—more departments–>
</Departments>
```