FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
BRATISLAVA

Proces integrácie aplikácií

Diplomová práca

Ondrej Svačina **2007**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky
Katedra informatiky

Proces integrácie aplikácií

DIPLOMOVÁ PRÁCA

Ondrej Svačina

informatika

Vedúci práce: Mgr. Pavol Mederly

BRATISLAVA 2007

Knižničné údaje

Svačina, Ondrej
Proces integrácie aplikácií (diplomová práca)

Obsahuje bibliografické referencie (formát ISO 690).

Kľúčové slová: integrácia aplikácií, proces, prípadová štúdia (application integration, process, case study)

Čestne vyhlasujem, že som túto záverečnú prácu vypracoval samostatne, len s použitím uvedenej literatúry.

V Bratislave, X 2007

.....

Ondrej Svačina

Na tomto mieste by som sa rád poďakoval vedúcemu mojej diplomovej práce, Mgr. Pavlovi Mederlymu, za pripomienky a rady, ktoré viedli k jej nepochybnému skvalitneniu.

Okrem toho vďaka patrí aj pracovníkom Univerzity Komenského, s ktorými sme na mnohých stretnutiach analyzovali a riešili otázky integrácie ich aplikácií.

Abstrakt

Autor:	Ondrej Svačina
Názov práce:	Proces integrácie aplikácií
Škola:	Univerzita Komenského
Fakulta:	Fakulta matematiky, fyziky a informatiky
Vedúci diplomovejpráce:	Mgr. Pavol Mederly
Rok:	2007
Počet strán vrátane príloh:	136

Diplomová práca sa zaoberá procesom integrácie aplikácií, teda činnosťami, ktorých cieľom je vytvoriť integrovaný informačný systém – informačný systém, v ktorom si aplikácie navzájom poskytujú potrebné dáta a procesy, komunikujú efektívne a s minimálnou údržbou.

Prezentovaný proces nezávisí na žiadnej technológii, dokonca nie je závislý ani na žiadnom konkrétnom prístupe k integrácii aplikácií. Cieľom bolo navrhnúť dostatočne univerzálny proces, ktorý by bol schopný zachytiť celý životný cyklus integračných projektov, od prvotných krokov cez analýzu, návrh až po odovzdanie do prevádzky. Možnosti jeho ďalšieho rozpracovania sú načrtnuté v závere. Popri procese zmieňujeme aj kritériá kvality a riadenie rizík v kontexte integrácie aplikácií.

V druhej časti ilustrujeme proces integračným projektom na Univerzite Komenského, kde uvádzame aj čiastkové výsledky. V integračnom projekte sme zanalyzovali potreby integrácie vybraných systémov, navrhli čiastočnú integračnú architektúru a pripravili návrh integrácie.

Súčasťou práce je aj stručný úvod do problematiky integrácie aplikácií, prehľad integračných prístupov a ich súvis s technologickými aspektmi.

Kľúčové slová: integrácia aplikácií, proces, prípadová štúdia

Predhovor

Cieľom práce bolo navrhnúť proces pre integráciu aplikácií vhodný do podmienok podobných prostrediu Univerzity Komenského, čo sa týka rozsahu (počtu aplikácií) a zložitosti informačného systému. S tým veľmi úzko súvisiaca úloha bola zanalyzovať potreby integrácie vybraných aplikácií na Univerzite Komenského a aspoň čiastočne navrhnúť riešenie.

Prínos tejto práce vidíme v dvoch rovinách: je to jednak samotný proces integrácie aplikácií a jednak práce vykonané v rámci integračného projektu na Univerzite Komenského, ktoré tu taktiež prezentujeme. Obe časti sa navzájom významne ovplyvňujú – prvé náčrty procesu vznikali zo skúseností z integračného projektu, následne sa ním overovali, keď sme postupovali v prácach podľa neho, a znovu projekt poskytoval spätnú väzbu do procesu, ktorý sme na základe nej korigovali, čím sa kruh uzatvoril. Zdôrazňujeme však, že proces nie je viazaný žiadnym spôsobom na Univerzitu Komenského a je možné ho použiť v ľubovoľnom prostredí.

Vychádzali sme predovšetkým zo skúseností v oblasti budovania klasického softvéru a literatúry, ktorá procesy tvorby softvéru a veľkých podnikových aplikácií popisuje. Ako sme spomenuli vyššie, prax v oblasti integrácie aplikácií sme získali integráciou aplikácií na Univerzite Komenského.

Po krátkom úvode práca začína prehľadom prístupov a technológií, ktoré sa pri integrácii aplikácií používajú. Tretia kapitola popisuje detailne navrhnutý proces integrácie aplikácií, spolu s jeho východiskami. Štvrtá kapitola je súhrnom prác, ktoré sme urobili v rámci integračného projektu na UK a slúži zároveň ako ilustrácia procesu a prípadová štúdia. V prílohách sú uvedené výstupy integračného projektu.

Obsah

Zoznam obrázkov

Zoznam tabuliek

1. Úvod

Každý podnik či organizácia si dnes uvedomuje možnosti informatizácie, pričom vie, že bez informačného systému by bol len sotva konkurencieschopný. Viac ako kedykoľvek predtým, zvlášť v našom regióne po viacerých zmenách v spoločnosti a na trhu, si podniky ostro konkurujú v poskytovanej kvalite svojich výrobkov či služieb. V rovnakej situácii sa nachádza aj neziskový sektor a školstvo. Naše univerzity a vysoké školy si nielen konkurujú navzájom, ale oveľa väčších súperov máme v podobe zahraničných inštitúcií mnohokrát s vlastným know-how a vysokou kvalitou.

Veľký problém však nastáva, keď informačný systém namiesto toho, aby znižoval náklady, naopak, spotrebúva viac zdrojov, ako je pridaná hodnota, ktorú prináša. Jeden z hlavných dôvodov môže byť slabá previazanosť jeho jednotlivých aplikácií. Keď aplikácie navzájom o sebe nevedia, vzniká tendencia duplikovať funkčnosť, sú nutné pravidelné ručné zásahy do ich činností (napr. v podobe exportov a importov dát) alebo sa jednoducho neexistujúca komunikácia nahradí činnosťou používateľa, ktorý potom musí prepisovať dáta z formulára v jednej aplikácii do podobného v druhej. V každom prípade takýto stav prináša iba neúmerné náklady spočívajúce v zlej udržiavateľnosti a nutnosti zamestnávať inak zbytočné ľudské kapacity, nehovoriac už o používateľskom diskomforte. Hovoríme, že informačný systém je dezintegrovaný a našou snahou je ho povýšiť na integrovaný, teda dosiahnuť stav, v ktorom aplikácie navzájom komunikujú a automatizovane prenášajú medzi sebou dáta a zdieľajú svoje procesy.

K dezintegrovanému informačnému systému buď prídeme jeho prirodzeným vývjom alebo dezintegrovaním po nejakej udalosti. Prvý prípad je typický pre organizácie, ktoré svoj informačný systém ešte len budujú a pre organizácie pôsobiace už veľmi dlho. Postupne, ako začali počítače prenikať z výskumných centier do komerčnej sféry, začali sa programovať rozličné aplikácie, ktoré pomáhali podnikom v ich bežnej agende, typicky to boli účtovné softvéry, balíky na podporu predaja, aplikácie pre sklady a pod. Tieto aplikácie boli vytvárané samostatne, bez ohľadu na iné aplikácie, pretože v tých časoch ani ľahká komunikácia možná nebola. S podobným výsledkom sa však stretneme i dnes, keď vývoj alebo nákup softvérových balíkov nie je koordinovaný a je v kompetencii každého oddelenia (učtáreň, predaj, sklad a pod.).

Niektorí dodávatelia softvéru si postavili celý svoj obchod na práve opísaných ťažkostiach. Presvedčiť zákazníkov prevádzkujúcich desiatky izolovaných aplikácií rôznej kvality, aby použili jeden už integrovaný balík ich výroby, ktorý zvláda všetko od účtovníctva až po evidenciu natankovaných pohonných hmôt referentských vozidiel, nebolo ťažké. Podobné univerzálne balíky väčšinou označujeme ako ERP (enterprise resource planning). Okrem vysokej ceny však ďalšiu prekážku predstavuje ich pomerne komplikované nasadenie. Otázka stojí, či napriek svojej univerzálnosti naozaj zvládajú všetko. Odpoveď je skoro vždy záporná a tak budeme musieť popri balíku ERP udržiavať ďalšie špecializované aplikácie – a sme opäť pri probléme integrácie aplikácií. Navyše je predsa vždy výhodné, keď si spoločnosť môže vybrať z každej kategórie toho najlepšieho dodávateľa a tú najkvalitnejšiu aplikáciu a nemusí sa viazať na

jeden obrovský ERP balík od jedného dodávateľa, z ktorého vyniká možno iba účtovníctvo, ale ostatné moduly má len priemernej či podpriemernej kvality.

Aj dokonale integrovaný informačný systém sa môže dezintegrovať, najmä z obchodno-politických dôvodov. Najčastejšie sú prípady akvizícií a fúzií spoločností – v týchto prípadoch nová spoločnosť dostane „do vienka“ aplikácie z oboch informačných systémov a musí si pripraviť stratégiu ich zlúčenia, teda „po našom“ môžeme povedať integrácie.

2. Integrácia aplikácií

Informácie uvedené v tejto kapitole sú súhrnom všeobecných poznatkov v oblasti integrácie aplikácií a poznatkov, ktoré už boli opísané v literatúre venujúcej sa integrácii aplikácií (napr. (Linthicum, 2003), (Hohpe, a iní, 2003), (Chandra, a iní, 2003)).

Ako sme načrtli v úvode, moderný podnik potrebuje mať dobre zabezpečenú vnútornú organizáciu, musí rýchlo, efektívne a spoľahlivo vykonávať svoje podnikové procesy, čo v súčasnosti bez informačných technológií vo väčšine prípadov nie je možné. Žiadané údaje musí mať pracovník, ktorý ich potrebuje, bez zbytočného zdržania a čo možno najčerstvejšie. Je zrejmé, že bez toho, aby informačný systém navonok vyzeral ako jeden dobre fungujúci celok, to bude zložité. Naším cieľom je z jednotlivých kúskov – aplikácií – informačného systému v podniku poskladať takýto celok – integrovaný informačný systém. Budeme sa v práci zaoberať práve týmto skladaním – integráciou aplikácií.

Presnejšie, pod **integráciou aplikácií** rozumieme proces *riadeného zdieľania dát a procesov* medzi softvérovými entitami. Výsledný produkt nazývame **integrovaná aplikácia** alebo **integrovaný informačný systém.**

Softvérové entity v zásade môžu reprezentovať akékoľvek externe rozpoznateľné časti informačného systému ako celku, ktoré dokážu zdieľať dáta alebo procesy, alebo na druhej strane ich dokážu využívať. Väčšinou pôjde o aplikácie ako také, avšak entitami sú aj samostatné databázy, súborové servery a procesné servery, aplikačné servery, nasadené webové služby a podobne, ale napríklad aj externí partneri (väčšinou nás pri nich zaujíma iba rozhranie, cez ktoré komunikujeme, ale nie jeho implementácia). Softvérové entity, ak budú súčasťou podnikového informačného systému, budeme niekedy nazývať aj súčasťami informačného systému a niekedy jednoducho systémami.

V závislosti od šírky záberu entít, ktoré integruje, rozlišujeme medzi vnútropodnikovou integráciou (EAI) a (medzi)podnikovou integráciou (B2B):

Enterprise application integration (EAI) je integrácia vnútropodnikových aplikácií (entít). V prípade, že aplikácie (entity) sa nachádzajú na úrovni dvoch alebo viacerých podnikov, hovoríme o integrácii aplikácií **Business To Business** (B2B).

V zásade sú princípy oboch typov rovnaké, avšak pri integrácii B2B vystupujú do popredia ďalšie faktory:

- **technické** – zabezpečenie komunikačných liniek medzi podnikmi, niekedy na veľké vzdialenosti, ich spoľahlivosť, aké protokoly sa použijú a pod.,
- **bezpečnostné** – zabezpečiť dôvernosť, integritu a dostupnosť, rozhodnúť cez ktoré rozhrania budú partneri pristupovať,
- **politické** – pre ktorých partnerov zabezpečiť aké dáta a aké procesy, v ktorú dobu (napr. iba pred zdaňovacím obdobím).

V práci sa špecifickým otázkam podnikovej integrácie (B2B) nebudeme ďalej venovať, nakoľko, ako sme už spomenuli, základ majú rovnaký a v integračnom projekte UK sme zatiaľ neintegrovali systémy externých partnerov.

2.1. Modely prístupov k integrácii aplikácií

K riešeniu problému neintegrovaného informačného systému môžeme pristúpiť viacerými spôsobmi. My budeme rozlišovať prístupy k integrácii aplikácií primárne na základe jednotiek funkčnosti, ktoré sa prostredníctvom integrácie zdieľajú a vymieňajú (inými slovami, „čo“ sa zdieľa, obsah). Na tomto rozdelení sú nezávislé použité technológie, ktoré túto komunikáciu umožňujú (povedané inak, „ako“ sa zdieľa, forma).

V krátkosti takto rozoznáme tieto prístupy k integrácii aplikácií:

- dátový (výmena dát) – základná jednotka výmeny sú iba dáta bez funkčnosti,
- aplikačná – aplikácie nezdieľajú priamo svoje dáta, ale poskytujú časti svojej aplikačnej funkčnosti externým entitám,
- úroveň služieb – aplikácie zdieľajú služby.

Od tohto rozdelenia je úplne nezávislá technológia, pomocou ktorej sa prenášajú alebo zdieľajú dáta a procesy (funkčnosť) – tzv. middleware. Najčastejšie sa v praxi môžeme stretnúť s týmito druhmi middlewaru:

- softvéry ETL (extrakcia, transformácia, načítanie), CLI (Call Level Inteface),
- vzdialené volania procedúr a metód (napr. RPC, RMI, .NET Remoting),
- distribuované objekty (napr. DCOM/COM+, CORBA),
- message-oriented middleware (MOM),
- webové služby.

Vo väčšine (ak nie vo všetkej) dostupnej literatúre sa stretneme s rozdeleniami, ktoré zmiešavajú a stavajú na jednu úroveň obsahovú aj technickú stránku – napr. (Hohpe, a iní, 2003) hovorí o prenose súborov, zdieľaných databázach, vzdialenom volaní procedúr a správach, (Chandra, a iní, 2003) rozoznáva štyri „úrovne“: prezentačnú, podnikovo-procesnú, dátovú a komunikačnú, pričom prezentačná a komunikačná sú viac technické, dátová úroveň je „schopnosť porozumieť dátam“ a podnikovo-procesná je o integrácii podnikových procesov.

2.2. Výmena dát

Zjednodušene môžeme povedať, že softvérové aplikácie zvyšujú kvalitu života človeka práve automatizovaným spracovaním dát. Je preto prirodzené, že prvé kroky pri dosahovaní spolupráce medzi viacerými aplikáciami sa orientovali práve na výmenu dát. Tento spôsob ostáva stále veľmi rozšírený a dá sa povedať, že je akousi „integráciou prvého kontaktu“.

Hoci výmena dát predstavuje pomerne jednoduchý koncept, množstvo a štrukturálna zložitosť týchto dát v niektorých prípadoch vôbec nemusí byť triviálna. Môže ísť o údaje zo stoviek databázových tabuliek pospájaných stovkami relácií. Zanalyzovať problém a navrhnúť integračné riešenie môže zabrať veľa času.

Vo väčšine prípadov sú však takéto riešenia oproti ostatným vytvárané rýchlejšie a jednoduchšie (Linthicum, 2003). Avšak práve sústredenie sa na čisté dáta prináša nevýhody v podobe možnej straty celkového pohľadu na podnik a jeho procesy, ktoré sa snažíme eliminovať posunutím na ďalšie úrovne popísane nižšie.

Softvérovú entitu, ktorá dáta poskytuje, nazývame **producentom dát** alebo **dátovým zdrojom** a tú entitu, ktorá ich prijíma, **konzumentom dát.** V kontexte integrácie aplikácií výmenou dát môžu byť producenti a konzumenti dát, ako i metódy extrakcie dát, rozmanité. Tie najbežnejšie popíšeme podrobnejšie.

2.2.1. Extrakcia, transformácia a načítanie

Prvým krokom na ceste k úspešnej integrácii aplikácií výmenou dát je samotné získanie dát, teda *extrakcia*. Hocako sa to môže zdať samozrejmé, nemusí to byť jednoduché. Extrakciu môže sprevádzať viacero problémov od technologických (zložitosť samotného prístupu ku zdroju) až po problém súčasného prístupu viacerých používateľov k jednému zdroju a problémy aktuálnosti extrahovaných dát.

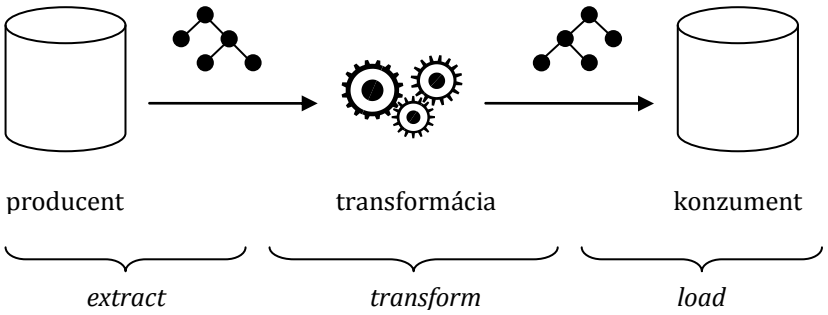
Vo väčšine prípadov sa nemôžeme uspokojiť s dátami v takej podobe, v akej prídu z dátového zdroja. Cieľová aplikácia ich totiž často očakáva v inom formáte a niekedy i inak štruktúrované, preto je nutná *transformácia* do podoby, v ktorej je ich cieľová aplikácia schopná prijať. Techniky transformácie dát popisujeme nižšie.

Po vykonaní potrebných transformácií si cieľová aplikácia *načíta* výsledné dáta. Konkrétny postup, podobne ako pri extrakcii, závisí od povahy konzumenta.

Celý proces nazývame podľa načrtnutých krokov extrakcia, transformácia a načítanie (angl. **extract, transform, load**, ETL). Frekvencia behu celého procesu môže byť rôzna v závislosti od požiadaviek. Bežne sa stretávame s

- *pravidelným ETL*, kedy sa proces opakuje v stanovený časový okamih (napr. každú hodinu, na konci fakturačného obdobia v mesiaci, na konci fiškálneho roku) alebo
- *aktualizačným ETL*, pri ktorom sú nastavené spúšťače, typicky v dátovom zdroji (spúšťač je asociácia udalosti v dátovom zdroji a odpovede na ňu, príkladom udalosti môže byť vloženie nového riadku do tabuľky `Faktúry`). Možnosti použitia aktualizačného ETL sú obmedzené podporou pre tvorbu spúšťačov v dátovom zdroji. Väčšina komerčných databázových systémov ich podporuje minimálne na úrovni tabuliek. V prípade, že zdrojom dát je súbor, dajú sa použiť štandardné notifikácie operačných systémov.[1]

[1] Napr. v operačných systémoch Microsoft Windows môžeme použiť API funkcie FindFirstChangeNotification a FindNextChangeNotification. Pri použití .NET Frameworku je k dispozícii trieda System.IO.FileSystemWatcher.



Obrázok 2-1 Proces extrakcie, transformácie a načítania dát

2.2.2. ETL a databázové systémy

Výhody databázových systémov sú nesporné a na riadenie dát sa používajú vo veľkej miere, preto i producenti a konzumenti dát sú pri ETL neraz databázové systémy.

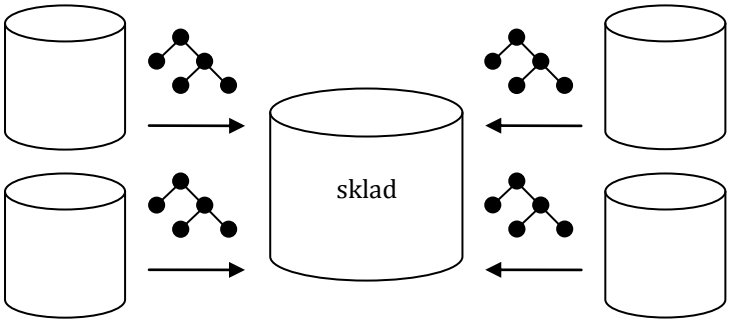
Výmena dát medzi jednotlivými databázami je tak často sa vyskytujúca úloha, že na trhu existuje pomerne veľa nástrojov a dokonca niektoré komerčné databázové systémy ponúkajú priamo podporu pre proces ETL, vrátane transformácií (napríklad Data Transformation Services v Microsoft SQL Server 2000 alebo SQL Server Integration Services v Microsoft SQL Server 2005).

V prípade, že nemáme k dispozícii žiadny špecializovaný nástroj, dávku ETL môžeme pripraviť aj ad hoc využitím štandardného aplikačného programového rozhrania pre prácu s databázovými systémami (napr. ADO.NET, ODBC, OLE DB, JDBC).

2.2.3. Ďalšie metódy spolupráce databázových systémov

Stáva sa, že vznikne požiadavka vybudovať novú aplikáciu, ktorá by používala dáta z viacerých dátových zdrojov. Ako príklad možno uviesť aplikáciu pre audit financií, ktorá potrebuje dáta zo samostatných databáz, v ktorých sú uložené faktúry (billing), údaje o zákazníkoch a údaje o dodávateľoch.

V tomto prípade do jednej databázy prúdia (prípadne transformované) dáta z niekoľkých databáz. Ešte častejší dôvod pre vznik takejto novej databázy je vykonávanie komplexných analýz dát a ich závislostí (OLAP, on-line analytical processing). Databáza, do ktorej sa sústreďujú dáta cez ETL, sa nazýva **dátový sklad (data warehouse).** Z výkonnostných dôvodov sú poväčšine dáta v sklade silne denormalizované, čo si môžeme dovoliť, pretože sklady sa zvyknú používať iba na čítanie. Viac informácií o dátových skladoch môže čitateľ nájsť napríklad v (Inmon, 2002).



Obrázok 2-2 Dátový sklad v kontexte ETL

Ďalším typickým príkladom sú distribuované databázy. Distribuovaná databáza nemá údaje uložené na jednom fyzickom mieste. Niekedy sa stretávame s tzv. horizontálnym delením tabuliek, pri ktorom sú riadky jednej tabuľky rozmiestnené na viacerých miestach, napr. centrála v Bratislava a pobočky v Žiline a Košiciach majú z výkonnostných dôvodov údaje o svojich zákazníkoch a realizovaných objednávkach vo svojich vlastných častiach príslušných tabuliek. Pri vertikálnom delení sú stĺpce tabuľky distribuované na rôznych miestach. V prípade, že v uzloch sú rôzne tabuľky, ale sú distribuované kompletné, môžeme postupovať ako pri čerpaní údajov z rôznych dátových zdrojoch (databáz).

Iným spôsobom zlúčenia dát môže byť vytvorenie **pohľadu nad viacerými databázami.** Takýto pohľad definujeme podobne ako klasický databázový pohľad, ale dáta môžu pochádzať z tabuliek z rozličných databáz; dokonca tieto databázy môžu byť postavané na fundamentálne rôznych dátových modeloch (napr. relačnom a objektovom). V prípade, že máme sofistikovaný nástroj na vytváranie takýchto pohľadov (alebo nám to dokonca umožňuje sám databázový systém), pokles výkonu nemusí byť dramatický (nástroj môže používať rozličné cachovanie výsledného pohľadu so závislosťami na zdrojových tabuľkách). Hlavný rozdiel oproti dátovému skladu je, že pohľad je čisto virtuálny a dáta dáva vždy aktuálne. Takýto systém môžeme použiť aj na zlúčenie vertikálne či horizontálne delených tabuliek. Konkrétna implementácia podobných myšlienok je napr. informačno-integračný systém *Infomaster* (Informaster: An Information System, 1997). Podobné koncepty sa niekedy označujú ako tzv. **databázová federácia** (Linthicum, 2003), (Heimbinger, a iní, 1985). Vo všeobecnosti je modifikácia dát v databázovom systéme ekvivalentná problému modifikácie pohľadov v relačných databázach (Heimbinger, a iní, 1985), a preto nie je široko podporovaná (Hull, 1997).

2.2.4. Kopírovanie súborov

Producenti a konzumenti dát nemusia byť iba databázové systémy. Najjednoduchším spôsobom výmeny dát je vlastne výmena najelementárnejších zväzkov údajov – súborov.

Prostredníctvom súborov sa dáta vymieňali už v časoch pred úspešným príchodom databáz. Dnes však vďaka jednoduchosti nie je takýto postup minulosťou, a to zvlášť aj vďaka tomu, že tento spôsob predstavuje istý najmenší spoločný menovateľ – takmer každá aplikácia nezávisle na svojom internom spôsobe ukladania údajov poskytuje možnosti importu či exportu vo forme

súborov, čo je mimoriadne užitočné pri potrebe neinvazívneho prístupu k integrácii.

Pri výmene súborov je kladený ešte väčší dôraz na transformácie, hlavne prevody formátov, keďže hlavne staršie aplikácie nepodporujú doménovo špecifické priemyselné štandardy formátov súborov pre prenos dát, ktoré sa za čas ich existencie vyvinuli. Ak takéto štandardy existujú a naše aplikácie ich poznajú, máme značne zjednodušenú úlohu.[2]

V dnešnej dobre badať výrazné presadzovanie metaformátu XML, ktorého výhody môže čitateľ nájsť napr. v (W3C, 2004) v kapitole 1.1 Origin and Goals. Jeho použitie umožňuje transformovať jednotlivé súbory prostredníctvom štandardných transformácií XML dokumentov (XSLT, pozri napr. (W3C, 1999)).

EDI (Electronic Data Interchange) bola snaha zjednotiť automatizovanú výmenu elektronických dokumentov.[3] Záber EDI bol pomerne široký. Dôraz bol kladený na zavedenie štruktúrovaných dokumentov v elektronickej podobe, ktoré boli ekvivalentom dovtedy používaných papierových formulárov posielaných prostredníctvom pošty, kuriérom a neskôr faxom, typicky faktúry a objednávky. Navyše však EDI opisoval aj spôsob výmeny dokumentov prostredníctvom komerčných tzv. VAN sietí (Value Added Network), hoci neskôr s extenzívnym využitím internetu prišli, prirodzene, korekcie.

Napriek tomu EDI nie je štandard. V minulosti existovali dva pokusy, jeden na úrovni ANSI, ktorý vyprodukoval štandard s označením X12 a druhý EDIFACT od ISO. ISO EDIFACT je čiastočne evolúciou ANSI X12, umožňuje v istej miere popisovať aj obchodné procesy, v ktorých centre však vždy stojí jeden dokument (teda popis sa týka viac-menej transformácie a posunu tohto dokumentu na obchodnej linke). Skutočnosť je však taká, že tak X12 ako aj EDIFACT majú v praxi iba charakter odporúčania a jednotlivé vertikálne odvetvia priemyslu používajú časti z nich mnohokrát s vlastnými pridanými rozšíreniami. Ďalšími problémami podľa (Chiu, 2002) sú minimálna penetrácia v malých a stredných podnikoch (pre ktoré môžu byť investície súvisiace so zavedením EDI značne vysoké a nemusia sa vrátiť v tolerovateľnej dobe) a problémy s udržaním tempa vývoja informačných systémov a obchodných procesov na niektorej strane linky (napr. nenastala úplná automatizácia alebo dokonca niektoré komunikujúce strany si prichádzajúce EDI správy vytlačili na papier a pracovali s nimi po starom). VAN siete, na ktorých sa dokumenty najčastejšie posielajú, sú veľmi drahé (prenájom alebo vybudovanie) a celkovo vďaka komplexnosti a rôznym interpretáciám sa EDI stáva drahým aj čo sa týka ľudských zdrojov (Newcomer, 2002). EDI dnes nahrádzajú modernejšie technológie postavené okolo webových služieb a ebXML.

2.2.5. Aplikačné programové rozhranie

Množstvo softvérových aplikácií je otvorených navonok iným programom prostredníctvom štandardných techník pre volanie externých procedúr. Aké procedúry konkrétne je možné volať, definuje aplikačné programové rozhranie danej aplikácie.

---

[2] Príkladom spomínaných štandardov sú de jure štandardy ako XMI (prenos všeobecných modelovacích metadát, používaný však výlučne pre UML modely) alebo de facto štandardy ako RTF (formátované texty; vyvinutý spoločnosťou Microsoft).

[3] Informácie tu uvedené o EDI čerpajú hlavne z (Chiu, 2002).

Aplikačné programové rozhranie (application programming interface, API) je množina signatúr operácií,[4] ktoré implementuje aplikácia a sú dostupné pre externé aplikácie (hovoríme, že sú verejné).

Počet operácií a miera, do ktorej je možné prostredníctvom ich volaní ovplyvňovať beh aplikácie či získavať z nej dáta, môže značne kolísať od jednej k druhej.

Samotná technika je závislá od použitých technológií. Môže ísť o klasické lokálne volanie v konvenciách jazyka C, transparentné COM volania až po webové služby.

Operácie sú pri výmene dát stavané väčšinou systémom *vráť dáta, zapíš dáta.* Napríklad pre systém správy faktúr jedno z rozhraní môže vyzerať takto:

```
GetCustomerLegalName(int customerId)
SetCustomerLegalName(int customerId, string legalName)
InsertNewInvoice(Invoice invoice)
```

Tieto operácie majú za úlohu čisto posielať dáta z aplikácie von a prijímať dáta dnu. Na rozdiel od použitia API pri vyšších úrovniach integrácie pri volaní týchto operácií nezačínajú žiadne zložité procesy, maximálne prebehnú validácie argumentov.

2.2.6. Snímanie obrazoviek

Najnepríjemnejšia situácia, ktorá sa nám môže pri integrácii prihodiť, je nutnosť zapojenia aplikácie, ktorá nemá žiadne použiteľné „programátorské" rozhrania a nemáme ani možnosť ju upravovať.

Hoci na prvý pohľad vyzeráme stratení, predsa len existuje jedno rozhranie, ktoré nás môže zachrániť – používateľské. Pri tomto extrémnom spôsobe integrácie sa snažíme simulovať prácu používateľa s aplikáciou vo forme umelo vygenerovaných stlačení kláves a pohybov myši a získať dáta z aplikácie snímaním a následným parsovaním kópie obrazovky.

Snímanie obrazoviek je niekedy jedinou možnou cestou pre veľmi staré (niekoľko desiatok rokov) aplikácie, ktoré sú však stále v prevádzke, pretože pracujú tak, ako majú, čo je typický prípad pre mainframové aplikácie v bankách. Na trhu sú k dispozícii rôzne softvérové nástroje pre emuláciu a snímanie obrazoviek z rozličných historických terminálov, napr. voľakedy populárnych IBM 3270.

2.2.7. Transformácie dát

Dáta len zriedkakedy priamo vyhovujú cieľovej aplikácii, a preto musia byť pred ich použitím prevedené do podoby, v ktorej ich je cieľová aplikácia ochotná použiť s minimálnym množstvom zmien a predpokladov o zdrojovej aplikácii. Dôvodov je niekoľko, ale v zásade môžeme hovoriť o nezhodách:

[4] Signatúra operácie definuje minimálne názov operácie, dátové typy parametrov a typ návratovej hodnoty. Ďalej môže špecifikovať volacie konvencie parametrov, prístup k operácii, vyžadovaný bezpečnostný kontext a pod.

- *syntaktických* – dáta majú nevhodný formát, ale ich význam po správnej interpretácii je rovnaký a
- *sémantických* – dáta sa odlišujú významovo, teda koncept, ktorý popisujú v zdrojovej aplikácii má významové odlišnosti od konceptu, s ktorým pracuje cieľová aplikácia.

Medzi typické a časté syntaktické nezhody patria odlišné formáty transportných médií (iné protokoly, iné typy súborov, napr. CSV a XML, iná organizácia dát), odlišné kódovanie čísel a odlišné kódovanie národných znakov.

Na príklad sémantickej nezhody by sme si mohli vybrať koncept študenta. Zdrojová aplikácia za študenta považuje všetkých riadnych študentov na univerzite; cieľová aplikácia za študenta považuje iba študentov denných, ale z tých aj mimoriadnych.

Oba typy nezhôd riešime *transformáciami* medzi zdrojom a cieľom. Syntaktické nezhody vieme prekonať transformáciami relatívne jednoduchšie ako sémantické.

Konkrétny spôsob transformácie dát závisí na použitej technológii komunikácie (middlewari). Niektoré aplikácie pre ETL dodávané s databázovými systémami (napr. DTS z Microsoft SQL Server 2000) vedia definovať a interpretovať jednoduché transformácie priamo a na zložitejšie je možné napísať obslužný kód v imperatívnom jazyku (v prípade DTS v jazyku VBScript). Ak používame na prenos dát formát XML, môžeme s úspechom použiť štandardný transformačný jazyk XSLT, ktorý nám dáva možnosť deklaratívnym spôsobom opísať transformáciu z XML súboru do ľubovoľného cieľového formátu.

Sémantické transformácie sú ťažšie. V jednoduchších prípadoch, keď sú vzťahy medzi rozdielnymi konceptmi statické, môžeme použiť fixné pravidlá, napr. vo vyššie spomenutom príklade vyradiť iných ako denných riadnych študentov a zaradiť mimoriadnych denných študentov navyše. Ak vzťahy nie sú statické, menia sa a prípadne koncepty pribúdajú, jednou z možností by mohlo byť vybudovanie ontologických slovníkov a koncepty medzi aplikáciami prekladať pomocou nich. Táto oblasť je ale zložitá a prakticky sú takéto nasadenia zriedkavé.

Transformácie nemajú vždy jednoduchú povahu prekladov typu „jedna k jednej“, ale patria k nim aj rozličné projekcie, normalizácie aj denormalizácie dát.

Transformácie dát nepoužívame iba pri syntaktických alebo sémantických transformáciách, ale niekedy pomocou nich pripravujeme aj dáta nové — často sa stretneme s rôznymi agregáciami (sumy, štatistické priemery, rozptyly a pod.) či poľami s hodnotami vypočítanými z ostatných dát.

2.3. Aplikačná úroveň

Na aplikačnej úrovni pristupujeme k integrácii napojením na logiku aplikácie, pričom aktiváciou zdrojovej aplikácie (napr. zavolaním metódy na zdieľanom rozhraní) dochádza k vyvolaniu viacerých súvisiacich akcií, ktoré spolu poskytujú podnikovú funkčnosť. Na rozdiel od integrácii na dátovej úrovni, kde jediným cieľom je zdieľať dáta bez akejkoľvek logiky na strane zdroja, pri integrácii na aplikačnej úrovni nám ide v prvom rade o zdieľanie aplikačnej logiky, hoci výsledkom invokácie môžu byť aj dáta.

Na aplikačnej úrovni najčastejšie integrujeme cez aplikačné programové rozhranie, pri ktorom máme na výber mnoho rôznych techník a middlewarov. V zásade ide skoro vo všetkých o vzdialené volanie operácií. Ucelený zoznam princípov v podobe komunikačných vzorov, ktoré stoja za týmito technikami, môže čitateľ nájsť vo (Völter, a iní, 2004). Konkrétnymi príkladmi sú CORBA, DCOM, JMI, .NET Remoting.

Na aplikačnej úrovni môžeme integrovať aj snímaním obrazoviek, väčšinou v takomto prípade by išlo o vyplnenie údajov na obrazovke a stlačenie potvrdzujúceho tlačidla v jednej integračnej jednotke.

2.4. Úroveň služieb

Pod **službou** máme na mysli sémanticky uzavretú explicitne popísanú jednotku funkčnosti, ktorú systém poskytuje konzumentom.

Služby poskytujú svoju funkčnosť zvyčajne na *hrubšej úrovni* ako komponenty, prevádzkuje ich niekto iný *a fyzicky ani nie sú súčasťou konzumenta* (typicky bežia dokonca na iných počítačoch), čím sa znižuje previazanosť aplikácií a zlepšuje ich udržiavateľnosť.

Služby sú takto veľmi vhodnými bodmi, cez ktoré môžeme integráciu uskutočňovať a v súčasnosti môžeme badať trendy ich rozsiahleho používania.

Najdôležitejšou implementáciou služieb sú webové služby, ktoré ako technológia sú navyše platformovo nezávislé (na rozdiel od napr. COM či RMI), čo je ďalším významných faktorom pri integrácii aplikácií.

Viac o výhodách a nevýhodách službách a hlavne webových službách sa dá nájsť napr. v (Hasan, 2004), (Marks, a iní, 2003), (Cerami, 2002) alebo (Arora, a iní, 2002).

V nasledujúcich dvoch kapitolách stručne popíšeme dve podľa nás veľmi zaujímavé a perspektívne myšlienky, ktoré sa dajú uplatniť pri integrácii aplikácií.

2.5. Automatizácia a riadenie procesov

Integrácia aplikácií implikuje na istej úrovni aj automatizáciu procesov, pretože za predpokladu, že aplikácie potrebujú medzi sebou zdieľať dáta a procesy a nie sú integrované, toto zdieľanie musí byť zabezpečené ručne človekom. Integrovaním aplikácií zdieľanie automatizujeme a tak sa automatizujú aj podnikové procesy (aj keď čiastočne), ktoré s touto komunikáciou súvisia.

Špeciálnou kategóriou integrácie, ktorá sa dostáva do popredia záujmu súčasnosti (aj vďaka rozmachu webových služieb a architektúry orientovanej na služby), je takmer úplná automatizácia a s tým súvisiace riadenie procesov prostredníctvom informačných technológií.

Pod **automatizáciou procesu** budeme v našej práci rozumieť maximalizáciu počtu činností (aktivít) podnikového procesu vykonávaných informačnými technológiami. **Riadenie procesu** je špecifická činnosť externá bežným podnikovým procesom, ktorej hlavnou úlohou je koordinovať aktivity podnikových procesov (v rámci nich aj medzi sebou).

Samotné riadenie procesu môže mať tiež rozličnú úroveň automatizácie – od žiadnej (podnikový proces je riadený ručne)[5] až po úplnú automatizáciu. Softvér špeciálne určený na riadenie procesov budeme nazývať **procesný stroj** alebo procesný server.

Poznámka: Niekedy sa koordinácia viacerých podnikových procesov označuje ako choreografia. Podobný a často zamieňaný pojem je orchestrácia. Pod orchestráciou sa chápe koordinácia implementácie služieb (napríklad jedna podniková služba môže byť implementovaná viacerými fyzickými službami). Viac informácií napr. v (Richards, 2006). Tieto pojmy však nie sú ustálené a v niektorých prácach, predovšetkým tých popisujúcich technológie, sa orchestrácia používa v zmysle centrálneho riadenia procesov a choreografia ako decentralizovaná koordinácia.

Požiadavky kladené na procesné stroje podľa (Chappell, 2005) sú:

- schopnosť robiť rozhodnutia na základe podnikových pravidiel,
- schopnosť komunikácie s externými softvérovými systémami,
- schopnosť komunikácie s ľuďmi,
- schopnosť uchovávať svoj stav počas behu inštancie procesu,

Okrem týchto základných požiadaviek to sú (Chappell, 2005):

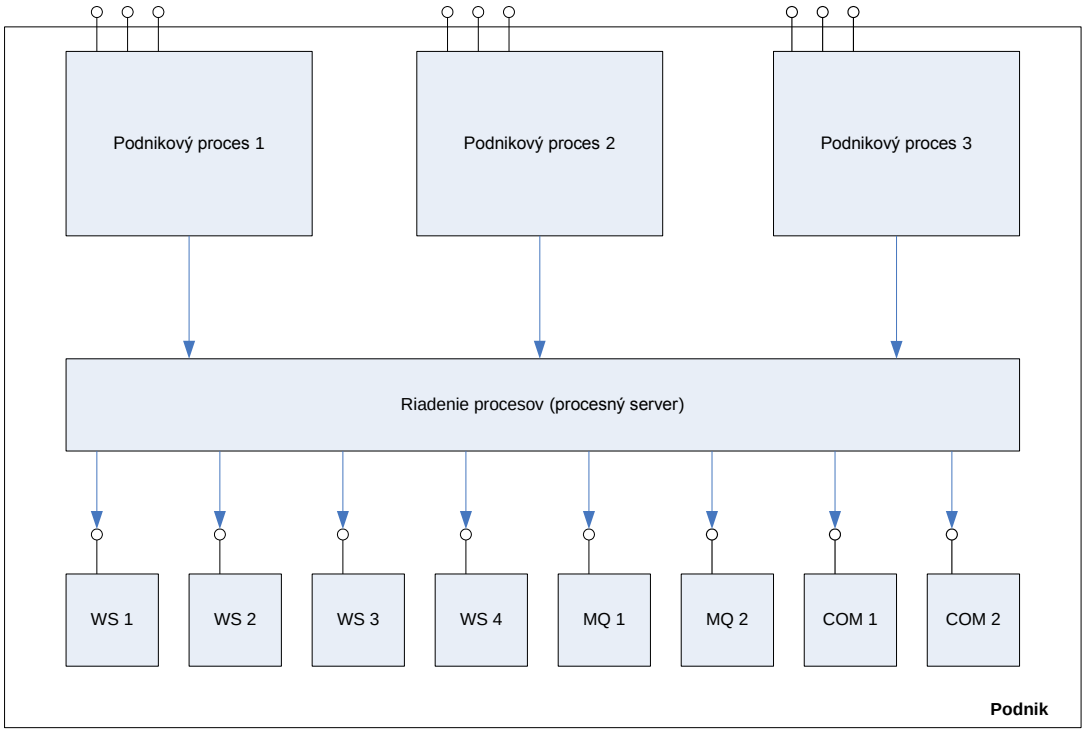
- komponentový prístup (na úrovni aktivít) a deklaratívne riadenie správania sa aktivít, napr. transakcie,
- nástroje na grafické vytváranie procesov,
- monitorovanie bežiacich inštancií procesov,
- možnosť zmeniť bežiacu inštanciu, napríklad pridaním kroku.

Procesný stroj (server) bude predstavovať v ďalej popísanom procese integrácie aplikácií jeden z integračných systémov.

Na obrázku 2-3 vidíme kontext procesného stroja v architektúre podnikového informačného systému. Podnik definuje tri podnikové procesy, ktoré majú explicitne popísané rozhrania. Procesy sú definované väčšinou v doménovo špecifickom jazyku procesného stroja, napr. BPEL.[6] Aktivity týchto procesov môžu byť volania ostatných aplikácií informačného systému pomocou rôznych technológií (na obrázku sú znázornené štvorcami dole, napríklad webové služby, messaging alebo COM).

[5] čo ale nevylučuje, že jednotlivé činnosti sú vykonávané automatizovane

[6] BPEL = Business Process Execution Language, vykonávateľný modelovací jazyk pre popis podnikových procesov.



Obrázok 2-3 Architektúra pri automatizovanom riadení procesov

2.6. Modelovo riadená integrácia

Takmer všetky procesy vývoja softvéru, ktorými sa riadime v súčasnosti, identifikujú niekoľko základných aktivít: analýza požiadaviek, návrh, implementácia, verifikácia a validácia a následná údržba. Pri analýze ako aj pri návrhu v tradičných procesoch vznikajú pomerne obsiahle modely, ktoré slúžia v prípade analytického modelu na zachytenie konceptov doménovej oblasti problému a súčasne pomáhajú pri komunikácii so zákazníkom ako aj pri zbere požiadaviek a v prípade návrhového modelu na dokumentáciu pripravovaného riešenia a jeho komunikáciu s implementačným tímom. Práve na rozhraní návrhu a implementácie vzniká problém: hneď ako sa skončí s implementáciou, mnohokrát tieto modely sa stávajú zbytočnými (stáva sa, bohužiaľ, že takýto stav nastane už počas implementácie), robia sa úpravy v kóde, ktoré sa nesynchronizujú s návrhovým modelom buď z časových dôvodov alebo jednoducho preto, lebo už analytici a návrhári pracujú na novom projekte. Návrhový model však obsahuje pomerne detailné informácie o riešení problému zákazníka a doplnením ďalších vhodných informácií na nižšej úrovni sa črtá možnosť eliminovať fázu implementácie (minimálne v podobe, v akej ju poznáme dnes) buď automatickým vygenerovaním modelu nižšej úrovne („zdrojového kódu“) alebo priamym spustením (interpretáciou) modelu. Výhody sú zrejmé: ušetrenie nemalých nákladov a zmenšením počtu chýb (chýb vzniknutých počas implementácie a chýb pochádzajúcich nesprávnou transformáciou návrhového modelu do programovacieho jazyka).

Model Driven Architecture (MDA) je jeden z konkrétnych prístupov modelovo riadeného vývoja. Pozostáva zo sady štandardov a odporúčaní, ktoré spravuje organizácia *Object Management Group* (OMG). V tomto prístupe sa väčšina práce sústreďuje do popisu samotnej funkčnosti systému v podobe modelu nezávislého od platformy (tzv. PIM – *Platform Independent Model*), ktorý sa

následne transformuje do jedného alebo viacerých modelov špecifických na platforme (PSM – *Platform Specific Model*), v ideálnom stave automaticky za použitia vhodných nástrojov. Po platformou sa tu myslí cieľové prostredie pre PSM, označované v MDA aj ako *middleware platform*, príkladom môže byť trojica C#, .NET a webové služby. Explicitné rozdelenie PIM a PSM prináša vlastne oddelenie podnikovej funkčnosti od technologických aspektov, ktoré sú vo väčšine klasických prístupov k architektúre úzko previazané (OMG, 2007). V prípade príchodu novej technológie nie je potrebné podnikovú funkčnosť modelovať znova. O MDA existuje viacero knižných publikácií, napr. (Kleppe, a iní, 2003) alebo (Mellor, a iní, 2004).

Jeden z problémov je, že organizácie často nevidia tento potenciál v modeloch alebo dokonca celé modelovanie je skôr záležitosťou niekoľkých náčrtov, pričom o formálnom modeli napríklad v jazyku UML sa nedá hovoriť (Zetie, 2005). Prieskum Forrester Research v roku 2005 ukázal pomerne prekvapivý fakt, že v skúmanej vzorke bolo iba menej ako 30% organizácií, ktoré generujú kód z modelu a následne ho aj udržiavajú aktuálnym po ručných zásahoch doňho (Zetie, 2005).

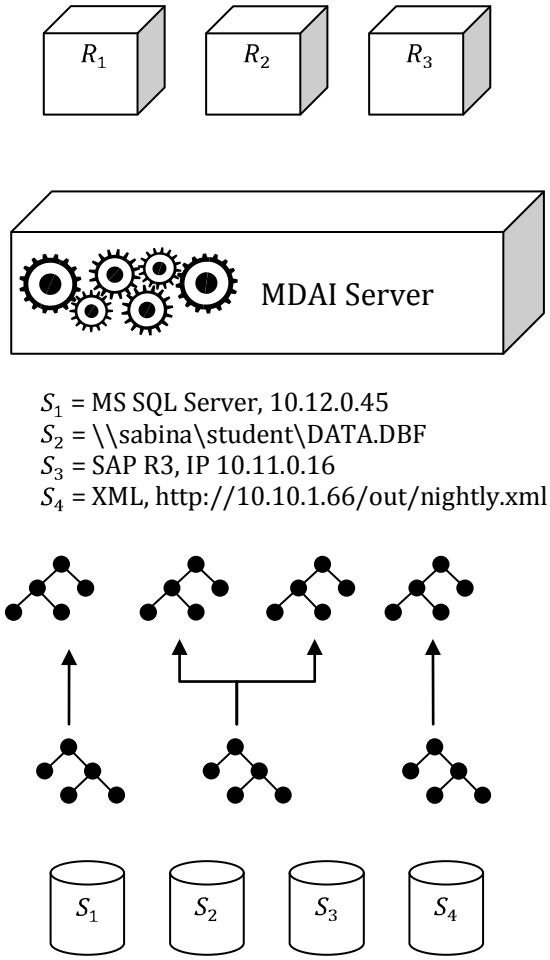
Koncept modelovo riadeného vývoja aplikácii sa ukazuje ako životaschopný, preto bola len otázka času, kedy sa podobné myšlienky začnú presadzovať aj na poli integrácii aplikácií. Modelovo riadená integrácia (MDAI) je vo svojej podstate pomerne priamočiare rozšírenie konceptov MDA na (zatiaľ) predovšetkým dátovo orientovanú integráciu aplikácii.

Základom modelovo riadenej integrácie je vytvorenie platformovo nezávislého modelu (PIM) verne zachytávajúceho štruktúru dát v existujúcich častiach informačného systému. Druhý krok predstavuje model pohľadu na dáta v štruktúre, o ktorú majú záujem cieľové aplikácie. Následne treba prepojiť obidva modely zodpovedajúcimi transformáciami a doplniť ich ďalšími informáciami potrebnými pre automatizáciu.

Na opačnom konci modelovo riadenej integrácie stojí súčasný spôsob – *programovo riadená integrácia*, pri ktorej sa aplikácie spájajú písaním kódu alebo konfigurovaním na relatívnej nízkej úrovni programátormi.

Koncept modelovo riadenej integrácie podlieha v súčasnosti intenzívnemu výskumu; pozri napríklad práce (Lang, a iní, 2002), (Zhu, a iní, 2004), (Zdun, a iní, 2006).

Na obrázku 2-4 sme sa pokúsili ilustrovať naše chápanie modelovo riadenej integrácie po preštudovaní viacerých materiálov. Obrázok zachytáva prechod informácií medzi zdrojovými aplikáciami (S_1 až S_4) a cieľovými aplikáciami (R_1 až R_3) zdola nahor. Návrh v modelovacom nástroji definuje transformácie dát medzi nimi a prípadnú koordináciu procesov a ďalej namapuje symbolické mená aplikácií na reálne umiestnenia. Následne sa tieto definície umiestnia na MDAI server, ktorý model interpretuje alebo alternatívne sa definície pretransformujú na kód, ktorý sa po skompilovaní vykoná.



Obrázok 2-4 Príklad prechodu informácií pri integrácii riadenej modelmi

2.7. Zhrnutie

V tejto kapitole sme zaviedli kľúčové pojmy, s ktorými budeme ďalej v práci pracovať. Okrem toho sme poskytli základný úvod do prístupov integrácie aplikácií a načrtli metódy automatizácie procesov a integrácie riadenej modelmi.

3. Proces integrácie aplikácií

V tejto kapitole popíšeme detailne všeobecný postup, pomocou ktorého je možné vybudovať integrovaný informačný systém.

Hlavný výstup (produkt) procesu integrácie aplikácií je integrovaný informačný systém.

Pod **integrovaným informačným systémom** máme na mysli taký informačný systém,

- v ktorom jeho časti (aplikácie) navzájom komunikujú (zdieľajú svoje dáta a procesy), podľa možností čo najviac automatizovane,
- bez zbytočných zdržaní,
- a ktorý môžeme kedykoľvek rozšíriť o novú súčasť (aplikáciu) alebo z neho kedykoľvek odobrať nasadenú viac nepotrebnú súčasť (aplikáciu) jednoduchým spôsobom.

Ciele procesu integrácie aplikácií (skrátene integračného procesu) sú teda:

- zintegrovať existujúce časti (aplikácie) informačného systému podniku, a to zdieľaním nových dát a procesov a zlepšovaním (skvalitnením) existujúcej komunikácie,[7]
- a uviesť výsledný už integrovaný informačný systém do ľahko udržiavateľného stavu, v ktorom nasadenie ďalších aplikácií nepovedie opäť k dezintegrovanému informačnému systému.

Integrácia aplikácií prebieha vždy ako projekt. Vysvetlenie pojmu projekt nájdeme takmer v každej knihe o projektovom manažmente, napr.:

Projekt je postupnosť jedinečných, komplexných a spojených aktivít majúcich jeden cieľ alebo zámer, ktorá musí byť ukončená vo (vopred) daný čas, v rámci (vopred) daného rozpočtu a podľa špecifikácie (Wysocki, a iní, 2003).

Projekt je na rozdiel od **operácií** teda jedinečný a nikdy sa neopakujúci.

Proces je množina aktivít, ktoré sú vykonávané s cieľom vytvoriť konkrétny produkt alebo poskytnúť konkrétnu službu. Každý proces je definovaný svojimi aktivitami. **Aktivitu** primárne charakterizujú výstupy, t.j. výsledky jej vykonania. Doplnkovou charakteristikou sú jej vstupy.

Pod modelom procesu chápeme jeho abstraktný popis.

V praxi sa modely procesov skladajú z rôznych slovných i grafických popisov na rôznych úrovniach.

Každá aktivita (krok) nami navrhnutého procesu je charakterizovaná nasledujúcimi časťami:

[7] Treba si uvedomiť, že aspoň z filozoficko-teoretického hľadiska, väčšina aplikácií už v spoločnosti „zintegrovaná“ bude. Málokedy sa stretneme so situáciou, že by sa ozaj nedostávali striktne žiadne informácie z jednej aplikácie do druhej, ale vo väčšine prípadoch sa bude na tejto výmene podieľať človek, v tej najtriviálnejšej forme pôjde napr. o ručné prepisovanie formulárov referentkami.

- *popis* vo voľnej textovej forme podáva celkový pohľad na krok,
- *účel a zdôvodnenie* je stručným súhrnom vysvetlenia zámeru a odôvodnenia potreby kroku v procese,
- *zúčastnené roly* vymenúvajú, ktoré roly sa podieľajú na vykonávaní kroku,
- *aktivity* a *tok práce* popisujú dekomponované aktivity nižšej úrovne, z ktorých sa krok skladá a ich vzájomné, predovšetkým sekvenčné, vzťahy. Aktivity zásadne pomenovávame názvami začínajúcimi neurčitkom slovesa, keďže reprezentujú pracovné jednotky,[8]
- *vstupy* a *výstupy* detailne popisujú vstupné, resp. výstupné artefakty (môžu byť hmotné i nehmotné, napr. znalosti). V prípade dokumentov sú priložené šablóny, kde je to možné,
- *podpora CASE*[9] rozoberá možné uplatnenie počítačových nástrojov v danom kroku, ktoré znižujú prácnosť a zrýchľujú vykonanie kroku,
- *odporúčania* sú súhrnom rád a heuristík, ktoré sa nám osvedčili pri praktickom overovaní kroku,
- *odkazy na prípadovú štúdiu UK.*

Prezentovaný proces sa dá chápať aj ako procesný rámec[10] – má explicitne definované miesta, do ktorých je možné doplniť vlastné postupy („prekryť"), ktoré môžu vyplývať napr. zo zavedeného systému riadenia kvality; príkladom je modelovanie podnikových procesov. Uvádzame však aj „predvolenú implementáciu", príp. odkazy na literatúru, ak sú takéto postupy už dobre zdokumentované.

Väčšina moderných modelov procesov, ktoré sa zaoberajú vývojom softvéru, sa snaží rozpoznávať dve hlavné etapy: konštrukciu a údržbu. Konštrukčná etapa má za úlohu pripraviť prvú verziu softvérového systému vhodnú do „ostrého" produkčného prostredia. Následná údržba vylepšuje vlastnosti kvality softvéru, čím ho ďalej približuje *očakávaniam* zákazníka (nutnosť údržby nemusí znamenať vždy len fakt, že softvér nebol úspešne dokončený – požiadavky zákazníka sa časom nevyhnutne menia s meniacim sa prostredím spoločnosti). Z tohto pohľadu ani proces integrácie aplikácií nie je výnimkou a musíme na to pamätať.

Budeme sa zaoberať aj rôznymi typmi architektúr. Architektúra objektu vo všeobecnosti znázorňuje jeho *štruktúru* na vyššej abstraktnej úrovni. Podľa (Bass, a iní, 2003) je „**softvérová architektúra** programu alebo výpočtového systému štruktúra alebo štruktúry systému, ktoré zahŕňajú softvérové elementy, ich externe viditeľné vlastnosti a vzťahy medzi nimi". My sa nebudeme zaoberať iba čisto softvérovými architektúrami, ale vo všeobecnosti architektúrami

[8] a neurčitkový tvar sa nám zdal vhodnejší ako rozkazovací spôsob

[9] Computer Aided Software Engineering (použitie softvérových nástrojov pri vývoji aplikácií, typicky kompilátory, generátory kódu, modelovacie nástroje a nástroje na riadenie zmien).

[10] Rámcom (angl. framework) označujeme v tvorbe softvéru rozpracovaný produkt slúžiaci ako podporná štruktúra, z ktorej doplnením vlastných častí vznikne finálny produkt.

podnikových systémov, ktoré môžu mať aj iné zložky ako len softvérové elementy, preto definíciu rozšírime takto:

Architektúra objektu je štruktúra alebo štruktúry objektu, ktoré zahŕňajú elementy, z ktorých je zložený, ich externe viditeľné vlastnosti a vzťahy medzi nimi.

Pri zmienke konkrétnych typov architektúry uvedieme presnejšie, ktoré elementy a aké vzťahy máme na mysli.

Vo svete modelov procesov vývoja klasického softvéru v súčasnosti badať jasné trendy v prospech iteratívno-inkrementálnych variantov (a ich agilných verzií), ktorých výhody oproti klasickému vodopádovému modelu sú zrejmé (Sommerville, 2004). Náš proces tieto myšlienky tiež podporuje a jeho iteratívno-inkrementálny potenciál môžeme vidieť v týchto dimenziách:

- integrácia „per partes“, teda integrujeme nie naraz všetky aplikácie v portfóliu podnikového informačného systému, ale postupne, v inkrementoch. Odporúčame však v jednom inkremente zintegrovať navzájom súvisiacu množinu aplikácií, nie viac-menej náhodné dvojice. Ideálne je zobrať aplikácie, ktoré pokrývajú implementáciu jedného alebo niekoľkých podnikových procesov – v takomto prípade sa neminieme cieľu vývoja po inkrementoch, a to postupného dodávania funkčnosti po zmysluplných a prevádzkyschopných celkoch,
- viaceré aktivity procesu majú samé o sebe iteratívnu povahu (modelovanie podnikových procesov, hľadanie a popisy podnikových objektov, návrh integračnej architektúry a iné).

3.1. Východiská procesu integrácie aplikácií

Prezentovaný proces integrácie aplikácií je výsledkom teoreticko-praktických úvah vychádzajúcich

- zo skúseností autora na viacerých komerčných projektoch v oblasti vývoja softvéru v pozíciách návrhára i vývojára,
- zo súčasných modelov procesu vývoja softvéru a budovania veľkých podnikových aplikácií,
- z práce na integračnom projekte, ktorý momentálne prebieha na UK.

Integračný projekt UK slúžil aj na spätné čiastočné overenie správnosti procesu a jeho vylepšenia (pozri kapitolu 4).

V čase písania tejto práce nám nebol známy žiadny iný formálne alebo poloformálne zachytený proces, ktorý by popisoval vo všeobecnosti integráciu aplikácií, okrem metodológie OpenEAI (Jackson, a iní, 2005). Náš proces nevychádza priamo z metodológie OpenEAI, nie je jeho vylepšením, ale samostatným procesom. Zásadné rozdiely medzi nimi sú:

- OpenEAI proces robí analýzu potrieb integrácie na dátovej úrovni (dátové objekty), pričom s nimi pracuje na veľmi nízkom stupni abstrakcie a technicky (pre každú aplikáciu detailne identifikuje tabuľky, dátové typy vrátane šírky, nullovateľnosť, a pod.), náš

proces ju robí prostredníctvom univerzálnych podnikových objektov na vysokej úrovni,

- OpenEAI nehovorí konkrétne z akých artefaktov a ako sa analyzujú potreby integrácie jednotlivých aplikácií (iba si kladie jednoducjú otázku „aké dáta bude aplikácie potrebovať od iných“ a „pre aké dáta bude táto aplikácia zdrojom“),
- je veľmi úzko previazaná s technológiami OpenEAI, náš proces je technologicky striktne neutrálny,
- OpenEAI počíta iba s integráciou prostredníctvom správ, náš proces umožňuje použiť ľubovoľný prístup aj typ middlewaru,
- náš proces poskytuje variabilitu pri analýze potrieb integrácie (modely podnikových procesov, procesno-systémové modely, používateľské popisy systémov),
- ak sa vychádza v našom procese z modelu podnikových procesov, explicitne sa oddeľuje implementácia od jadra procesu, čo umožňuje integrovať bez zmeny principiálnej stránky procesu (ktorý by mal byť definovaný procesným architektom a nemal by byť menený integračným tímom) a naopak, vykonávať vylepšovanie podnikových procesov počas analýzy potrieb integrácie,
- OpenEAI sa nezaoberá vôbec architektúrou riešenia.

Úzka previazanosť metodológie OpenEAI s technologickým balíkom OpenEAI na druhú stranu poskytuje možnosť automaticky generovať správy, ak sa návrhári zmieria so skutočnosťou, že ich budú definovať v jazyku XML.

OpenEAI sa zaoberá aj plánovaním projektu, ktoré my sme ešte nezapracovali do nášho procesu.

Zdalo sa nám, že v projekte OpenEAI sú dôležitejšie technológie a metodológia (proces) stojí viac na okraji.

3.2. Kvalita riešenia

3.2.1. Kritériá kvality produktu

Nech už vývoj alebo výroba dopadnú akokoľvek, konečný spotrebiteľ či zákazník bude hodnotiť kvalitu produktu alebo služby. Rozmer pojmu kvalita je však vždy trochu iný v závislosti na uhle pohľadu. Pre rozličných ľudí môže kvalita znamenať niečo iné. Počas pomerne krátkeho obdobia histórie odborov, ktoré sa zaoberajú vývojom softvéru, sa zozbieralo niekoľko menej či viac exaktných definícií kvality. Exaktnejšie vysvetlenia si berú na pomoc rozličné metriky, pomocou ktorých je možné objektívnejšie merať rozmanité aspekty systému. V zásade ale môžeme veľmi zjednodušene povedať, že kvalitný produkt je ten produkt, s ktorým je zákazník spokojný; ten, ktorý spĺňa jeho očakávania.[11]

Kvalita produktov je oblasť, do ktorej spoločnosti investujú nemalé finančné prostriedky, keďže vyššia kvalita prináša tak priamy výnos (vyššie tržby), ako

[11] Hlbšiu diskusiu o pojme kvalita a rôznych metrikách používaných pri vývoji klasického softvéru čitateľ môže nájsť napr. v (Kan, 2002).

i nepriamy (spokojnosť, referencie). Väčšina procesov vývoja softvéru explicitne spomína problém kvality priamo iba prostredníctvom verifikácie a validácie, ktoré prebiehajú prevažne formou testovania, čo je len jedno z opatrení na posilnenie kvality (a nanešťastie defenzívne). Riadenie kvality predstavuje samostatnú časť projektového riadenia zameranú práve na zvyšovanie úrovne kvality budovaných produktov. Pre viac informácií môže čitateľ konzultovať napr. (Horch, 2003) alebo (Tian, 2005).

Na základe našich skúseností sme zozbierali niekoľko kritérií kvality integrovaného informačného systému, ktoré sme rozdelili do klasických kategórií používaných pri kritériách kvality softvéru – FURPS (Functionality, Usability, Reliability, Portability, Supportability) – a pokúsili sme sa ich aplikovať s ohľadom na integráciu aplikácií. Treba si ale uvedomiť, že niektoré kritériá spadajú do viacerých kategórií a podľa nás nie je ani účelné sa držať nejakého striktného disjunktného delenia.

Integračné projekty sú navyše odlišné od projektov vývoja softvéru aj tým, že požiadavky na integrovaný informačný systém[12] sú skoro všetky typu, ktorý by sme označili v terminológii zberu požiadaviek vyvíjaného softvéru ako nie-funkčné (teda netýkajúce sa priamo služieb ním poskytovaných). Napriek tomu si dovolíme uviesť niekoľko kritérií v kategórii *funkčnosť*, ktoré považuje za *charakteristické pre integráciu aplikácií.*

Funkčnosť

Pri integrácii aplikácií funkčnosť znamená predovšetkým, že aplikácie (entity), ktoré disponujú potrebnými dátami alebo poskytujú potrebné procesy pre iné aplikácie (entity), ich zdieľajú a tie aplikácie (entity), ktoré ich naopak potrebujú, ich využívajú – veľmi jednoducho môžeme povedať, že je **entity spolu „rozumne“ komunikujú.**

Pre zníženie nákladov je dôležité, aby podľa možností čo najviac komunikácie prebiehalo automaticky, ideálne je, aby podnikové procesy boli tiež automatizované. Redundancia tam, kde nie je potrebná, je odstránená.[13] Zvlášť pri medzipodnikovej integrácii (B2B) je zaistená bezpečnosť.

Pri automatizácii dlhotrvajúcich obchodných procesov (niektoré môžu prebiehať aj týždne) máme stále zaistené uloženie ich stavu a pri transakciách medzi aplikáciami zaručené ACID vlastnosti.[14]

Používateľnosť

Používateľnosť označuje mieru jednoduchosti obsluhy používateľmi (teda nie vývojármi).

Pri integrovaných systémoch sa môžeme pozrieť na

---

[12] Ak sa odvíjame od situácie, keď integračný projekt rieši iba samotné zintegrovanie systému, teda neberieme do úvahy požiadavky na jednotlivé časti informačného systému.

[13] Cieľom integrácie aplikácií je, prirodzene, redundanciu minimalizovať, ale v niektorých prípadoch je účelná, napríklad pri zaisťovaní obnoviteľnosti údajov pri poruchách.

[14] ACID = Atomicity (atomickosť), Consistency (konzistentnosť), Isolation (izolácia) a Durability (trvácnosť), základné vlastnosti transakčného systému, používa sa hlavne v kontexte databázových systémov.

- používateľnosť systému *per se*: ide napríklad o možnosť sledovania stavu komunikácie či monitoring prebiehajúcich obchodných procesov v reálnom čase,
- používateľnosť systému koncovými používateľmi: používatelia nemusia rovnaké údaje vpisovať do dvoch alebo viacerých aplikácií, nemusia spúšťať ručné importy/exporty a pod. Systém sa pre nich tvári ako jeden celok.

Spoľahlivosť

Spoľahlivosťou rozumieme mieru odolnosti systému voči neočakávaným zásahom zvonka.

V kontexte integrácie aplikácií platia metriky používané pri softvéri, pretože spoľahlivosť závisí od použitých technologických prostriedkov (middlewaru).

Výkon

Výkon je všeobecne počet vykonaných operácií za jednotku času.

Pri integrácii aplikácií by sme mohli merať výkon ako:

- rýchlosť propagácie zmien v systéme (t.j. ako rýchlo sa zmeny vykonané v jednej aplikácií prejavia v ostatných),
- počet úspešne/neúspešne spracovaných správ, úspešne/neúspešne doručených správ, veľkosť správ, zdržanie správ a pod. (pri MOM),
- trvanie vykonania podnikovej transakcie (podnikového procesu bežiaceho ako transakcia).

Udržiavateľnosť

Udržiavateľnosť je miera, ako jednoducho sa systém darí udržovať v prevádzkyschopnom stave. Keďže požiadavky na systém sa čom zákonite menia vplyvom nevyhnutných zmien v podnikovom prostredí, zaraďujeme sem aj jednoduchosť úprav systému.

Pri integrácii aplikácií ide o:

- minimalizácia väzieb medzi aplikáciami, minimalizácia predpokladov, ktoré má aplikácia o svojich partnerských aplikáciách (znižovanie viazanosti – coupling),
- viditeľnosť (dovnútra) a analyzovateľnosť architektúry integračného riešenia,
- jednoduchosť pridávania nových aplikácií (súčastí) do integrovaného systému,
- komunikácia otvorenými štandardmi (podstatne môže zlepšovať interoperabilitu, zvlášť dôležité pri medzipodnikovej integrácii),
- schopnosť oddeliť podnikové (obchodné) pravidlá od zvyšku systému, centralizácia podnikových pravidiel, možnosť meniť podnikové pravidlá aj menej technicky zdatnými pracovníkmi,

- flexibilitu zámeny aplikácií (komunikácie cez rozhrania, zníženie viazanosti), pri medzipodnikovej integrácii flexibilita zámeny partnerov.

3.2.2. Kritériá kvality projektu

Na to, aby sme mohli prehlásiť projekt za úspešný, kandidátsky projekt:

- *musí byť ukončený načas* (teda stihnúť termíny),
- *nesmie prekročiť rozpočet* (teda realizovať s plánovanými finančnými prostriedkami, ľuďmi, zariadeniami a pod.),
- *musí spĺňať požiadavky* (väčšina ľudí povie jednoducho, že výsledný produkt projektu musí byť kvalitný). V krátkosti sme o tomto bode hovorili v predchádzajúcej podkapitole.

3.3. Celkový pohľad

Náš proces integrácie aplikácií pozostáva z niekoľkých krokov (ktoré budeme ďalej označovať aj ako aktivity) s naznačenou následnosťou. V krátkosti teraz tieto kroky popíšeme a ďalšie sekcie tejto kapitoly sa im budú venovať detailne.

Pripomíname, že každá spoločnosť si môže pripraviť vlastný model procesu integrácie aplikácií založený na našom príslušnými modifikáciami, ktoré zohľadnia zabehnuté pracovné postupy, štandardy a smernice.

Inicializácia integračného projektu v podniku pochádza zvyčajne od nespokojnosti so stavom informačného systému, ktorý vykazuje typické znaky dezintegrácie – pravidelné ručné zásahy v podobe importov/exportov, ktoré sú drahé a nespoľahlivé, obťažovanie používateľov udržiavaním rovnakých dát vo viacerých aplikáciách a z toho plynúce problémy so synchronizáciou, nízka udržiavateľnosť systému a podobne. Myšlienku posunúť informačný systém na novú úroveň integráciou väčšinou podávajú na rokovanie vedeniu spoločnosti architekti IT alebo iní pracovníci zodpovední za rozvoj informačných technológií v spoločnosti.

Podnik – budúci zákazník integračného dodávateľa – si pripraví **štúdiu realizovateľnosti**, v ktorej zhodnotí svoje možnosti a potenciálne benefity vyplývajúce z navrhovanej integrácie. V prípade nízkeho indexu pridanej hodnoty v porovnaní s predpokladanými nákladmi na predkladaný integračný projekt (zatiaľ len hrubo odhadnutými, väčšinou na základe skúseností a prieskumu trhu s integračnými dodávateľmi), vedenie spoločnosti sa môže rozhodnúť v ňom ďalej nepokračovať.

Po zaregistrovaní prvotného podnetu zákazníka dodávateľom integračných riešení začína **štúdia realizovateľnosti na strane dodávateľa.** V ňom zhodnotí svoje možnosti a či je schopný dodať navrhované integračné riešenie včas a za finančného ohodnotenia v medziach zákazníka.

Súbežne so štúdiou realizovateľnosti prebieha aj **výber kandidátskych systémov** – teda systémov z portfólia informačného systému, ktoré budú zapojené do integračného projektu. Zoznam týchto systémov nám poslúži aj ďalej ako

referencia a jeho vypracovanie nám dáva možnosť dôkladne nazrieť do architektúry podniku.

Po zahájení veľkých integračných projektov (viac ako 10 aplikácií, veľké spoločnosti, náročná podniková doména a pod.) sa začína s **modelovaním podnikových procesov.** Celý proces je zameraný na podnikové procesy ako jeden z hlavných artefaktov, pretože si myslíme, že práve podnikové procesy, ak sú správne nastavené, vypovedajú o tom, čo je pre spoločnosť dôležité a teda mala by sa ním riadiť aj integrácia aplikácií. Pri menších projektoch to odporúčame tiež, ale dá sa urobiť kompromis v podobe tzv. procesno-systémových modelov (pozri ďalej v popise aktivít), ktoré znázorňujú súčasnú implementáciu podnikových procesov s dôrazom na komunikáciu aplikácií. Model podnikových procesov by nemal obsahovať technologické aspekty implementácie, preto je vhodný aj ako zdroj pri súbežnom vylepšovaní procesov. Tento model nám môže priniesť aj zníženie práce, ak ho dokážeme vhodne využiť rozličnými transformáciami (pozri sekciu 3.7).

Z modelu podnikových procesov vieme určiť základné abstraktné koncepty, s ktorými podnik ako taký pracuje – osoby, veci, udalosti, pravidlá a pod., ktoré budeme nazývať **podnikové objekty.** Podnikové objekty unifikujú svojou abstrakciou zdroje, z ktorých pochádzajú a v ktorých sú o nich uchovávané informácie, ako aj ich reprezentáciu v nich, vďaka čomu sa vieme sústrediť na ich podstatu a nie sme zbytočne zaťažovaní nepodstatnými detailmi. Podnikové objekty takto predstavujú efektívny nástroj na zisťovanie potrieb integrácie aplikácií v spoločnosti.

Počas integračného projektu môžeme pristúpiť aj k **zlepšovaniu podnikových procesov.** Integrácia aplikácií môže prispieť k čiastočným zmenám v procesoch alebo tieto zmeny priamo vyvolať. Okrem toho existujúci model podnikových procesov dáva predpoklady na túto činnosť, ktorá by sa takto bez integrácie nemusela ani vykonať.

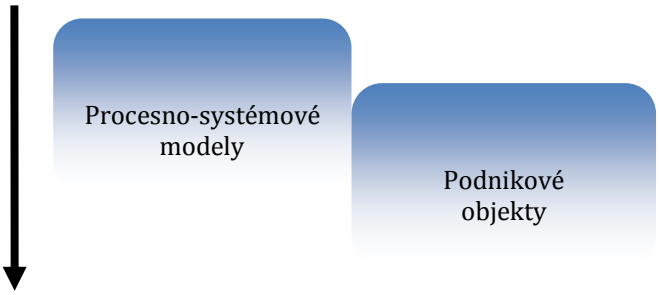
Analýza potrieb integrácie v podniku explicitne pokračuje mapovaním podnikových procesov na časti informačného systému. Pod časťami alebo entitami informačného systému budeme rozumieť samostatné celky, z ktorých sa skladá. Väčšinou pôjde priamo o aplikácie, ale niekedy môžeme vidieť aj databázy, ktoré nie sú súčasťou žiadnej aplikácie, rôzne služby využívané externými partnermi a pod. Pre podnikové objekty určujeme, ktoré entity ich ku svojej práci potrebujú a ktoré entity ich im môžu poskytnúť. Taktiež skúmame, ako sa mení význam a prípadne aj formát objektov medzi jednotlivými entitami a zaujímajú nás aj ich niektoré vlastnosti. Na takúto analýzu už potrebujeme mať aj podrobnejšie informácie o entitách, ktoré získavame pri **charakteristikách systémov.** Z niektorých výstupov dokážeme získať ďalšie podnikové objekty, predovšetkým z používateľských popisov systémov.

Architekti navrhnú integračnú **architektúru**, teda štruktúru komunikácie medzi jednotlivými entitami. Niekedy bude účelné zaradiť úplne nové systémy (entity) do informačného systému, špecifické pre integráciu, ktoré súhrnne nazývame **integračné systémy.** Najčastejšie pôjde o rôzne integračné servery, procesné stroje, ale aj špecializované celopodnikové služby zaisťujúce napríklad jednotné prihlasovanie alebo evidujúce osoby, partnerov, záväzky a organizačnú štruktúru. **Podnikové procesy** máme možnosť **automatizovať.**

Konkrétne prepojenia medzi aplikáciami navrhujeme v tzv. **komunikačnom návrhu.** Zapájame doň aj navrhnuté integračné systémy. V zásade postupujeme v súlade s podnikovými procesmi, analyzujeme, ktoré podnikové objekty sa pri nich vyskytujú a ktoré entity ich pri ich činnosti poskytujú a potrebujú. V komunikačnom návrhu je potrebná dostatočná miera aj technologických detailov, aby ich mohli implementátori v fáze implementácie bez väčších problémov zrealizovať – **implementovať.** Implementácia sa následne overuje štandardnými spôsobmi (**verifikácia a validácia**).

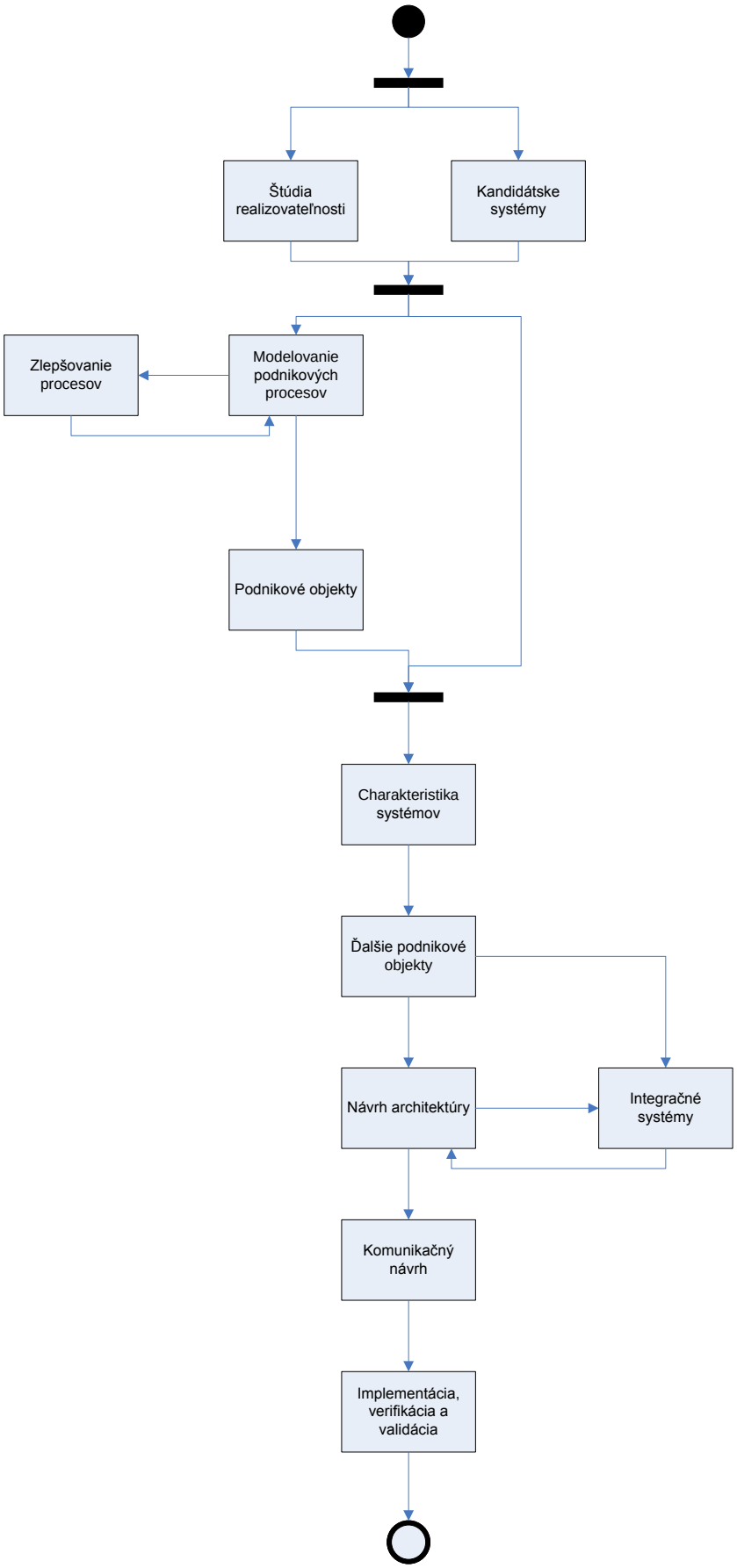
Na obrázku 3-3 sú práve popísané aktivity zachytené vo forme diagramu toku práce, ktorý nám názorne ukazuje aj vzťahy medzi nimi. Myslíme si, že pre čitateľa s praxou vo vývoji softvéru nebude problém tento a jemu podobné diagramy pochopiť. Viac informácií o nich je možné nájsť napr. v (Sharp, a iní, 2001).

Pripomíname, že tento model nemožno brať ako vodopádový model procesu vývoja softvéru, jednotlivé aktivity sa časovo prelínajú (tak ako je to uvedené aj v ich popisoch). Na obrázkoch 3-1 a 3-2 vidíme, aké je rozloženie prác vzhľadom na postupujúci čas na vertikálnej osi pri väčších, resp. menších integračných projektoch, pri ktorých sa tím nerozhodne vytvoriť celý model podnikových procesov, ale iba procesno-systémové modely. Bloky predstavujú prácu na príslušných aktivitách integračného projektu. Detailnejšie sa zmienime v popise aktivít a integračného projektu UK.



Obrázok 3-1 Časové rozloženie analytických prác na menších projektoch

Obrázok 3-2 Časové rozloženie analytických prác na väčších projektoch



Obrázok 3-3 Celkový pohľad na proces integrácie aplikácií

3.4. Roly v integračnom projekte

Rola v kontexte projektu je ucelená množina zodpovedností, ktorú vykonávajú účastníci projektu. Jednu rolu môže vykonávať jeden alebo aj viac ľudí a podobne jeden človek môže vykonávať viacero rolí.

Typicky sa na projektoch podieľa veľké množstvo ľudí s rozmanitými schopnosťami a zručnosťami. Podiel práce, ktorú odvedú, nie je rovnaký, a tak máme tendenciu zanedbávať mnoho rolí na projektoch, hoci ich správna identifikácia je mnohokrát pre projekt životne dôležitá.

Spoločnosti platiace za IT služby i spoločnosti tieto služby poskytujúce sa prirodzene snažia minimalizovať náklady svojich projektov. Súčasné dobre pozorovateľné trendy sú zákazkový vývoj, outsourcing (dodávka nekľúčových služieb externou spoločnosťou) a off-shore development (vývoj „za pobrežím“, na veľkú diaľku, typicky v Indii a Číne). Integrácia aplikácií veľmi dobre zapadá do týchto konceptov. Pre máloktorú organizáciu s výnimkou tých najväčších je výhodné vytvoriť a udržiavať celé oddelenie pre vývoj IT, nieto ešte zamestnávať pomerne úzko špecializovaných expertov v oblasti integrácie aplikácií na trvalý pracovný pomer. Z týchto dôvodov je potrebné explicitne rozlišovať roly v projekte, ktoré patria organizácii (prípadne organizáciám), v ktorých sa aplikácie budú integrovať (táto bude typicky aj objednávateľom prác) a roly patriace pod riešiteľský tím. Samozrejme je možné, že obe tieto skupiny budú spadať formálne pod jednu organizáciu, ak sa celý projekt bude riešiť in-house.

Konkrétne sme zadefinovali nasledovné roly, ktoré ďalej popíšeme:

Za riešiteľský tím: vedúci projektu, analytik, návrhár, implementátor, tester.

Za organizáciu: procesný architekt, podnikový architekt, správca IT

Manažment: vedúci projektu, obchodný manažér, sponzor, zadávateľ

3.4.1. Vedúci projektu

Vedúci projektu (projektový manažér) zodpovedá za celý projekt na strane dodávateľa. Jeho hlavnou úlohou je plánovať, prideľovať členom tímu prácu, kontrolovať a sledovať pokrok a tiež komunikuje a rokuje so zákazníkom na obchodnej úrovni. Vedúci projektu rieši prípadné konflikty medzi členmi tímu. Je prítomný počas celého projektu od jeho vzniku.

3.4.2. Analytik

Úlohou analytika je analyzovať problémy na strane zákazníka. V kontexte integračného projektu analyzuje súčasnú situáciu v podniku, podnikové procesy, požiadavky zákazníka a integračné možnosti.

Analytici môžu vykonávať aj vylepšovanie podnikových procesov v súvislosti s integráciou aplikácií (takúto špecializáciu môžeme nazývať procesný analytik).

3.4.3. Návrhár

Návrhár navrhuje riešenia zanalyzovaných problémov. Na základe analytického výstupu pripravuje integračné prepojenia jednotlivých softvérových

entít a v spolupráci s podnikovým architektom patrične upravuje podnikovú architektúru.

Na rozdiel od analytikov je ich práca technologicky orientovaná.

Niekedy sa môžeme stretnúť so situáciou, keď analytik je súčasne návrhár. V takom prípade si ale treba dať pozor, aby oddelil svoje zodpovednosti analytika v čase analýzy a návrhára počas návrhu.

3.4.4. Implementátor

V procese integrácii aplikácii je implementátor veľmi všeobecný pojem, pretože jeho presná úloha bude závisieť na zvolenom prístupe (prístupoch) k integrácii aplikácií, napr. v prípade dátovej integrácie to budú pravdepodobne databázoví špecialisti vytvárajúci dávky ETL, SQL skripty a dátové modely.

Vo všeobecnosti možno povedať, že implementátori realizujú návrhárom navrhnuté riešenie.

3.4.5. Kvalitár

V dnešných dobách vysokých nárokov na kvalitu je potrebné overiť produkt pred prebratím zákazníkom. Úlohou kvalitára (angl. v rôznych kontextoch *quality assurance*) je zaistiť potrebnú kvalitu produktu od zahájenia projektu.

3.4.6. Procesný architekt

Väčšie spoločnosti majú vyčlenenú špeciálnu pracovnú pozíciu procesného architekta. Procesný architekt navrhuje, optimalizuje a dohliada na realizáciu podnikových procesov. V menších spoločnostiach je táto funkcia zvyčajne kumulovaná s podobnými, napríklad funciou manažéra pre riadenie kvality.

3.4.7. Podnikový architekt

Podnikový architekt navrhuje, optimalizuje, aktualizuje a sčasti realizuje podnikovú architektúru spoločnosti, teda aplikácie, ktoré tvoria informačný systém, a ich vzťahy, predovšetkým ohľadom komunikácie. Podnikový architekt schvaľuje nákup nového softvéru a hardvéru a rozhoduje o vylepšení a vyradení existujúceho vybavenia informačnými technológiami. Podnikový architekt riadi strategickú a taktickú stránku problematiky informačných technológií v spoločnosti.

3.4.8. Správca IT

Správa informačných technológii (správa IT) zodpovedá za dennodenný chod informačného systému spoločnosti a riadi operatívnu stránku informačných technológií v spoločnosti. Funkčne podlieha podnikovému architektovi a zvyčajne je na čele oddelenia starajúceho ho sa o prevádzku informačného systému.

3.4.9. Sponzor

Sponzor zabezpečuje finančné a personálne krytie projektu na strane objednávateľa. Zvyčajne je zamestnaný ako funkčný manažér.

3.4.10. Zadávateľ

Zadávateľ v zastúpení objednávateľa komunikuje s dodávateľom ohľadne bežných vecí týkajúcich sa ich spoločného projektu a takisto podáva dodávateľskému tímu predstavy a požiadavky, ktoré má integračné riešenia spĺňať.

3.5. Štúdia realizovateľnosti

Niektorí dodávatelia softvérových produktov pred samotným prijatím objednávky vykonajú krátku analýzu so zámerom zistiť, či objednávka je pre ich spoločnosť naozaj výhodná a či ju dokážu vôbec splniť za ponúknutú odmenu. Štúdia realizovateľnosti by mala byť krátka a dať odpovedať na otázku, či má cenu v projekte ďalej pokračovať (Sommerville, 2004).

Podľa nás je vhodné, aby podobná štúdia prebehla tak na strane dodávateľa ako i na strane objednávateľa, hoci tradičné modely procesov vývoja softvéru vidia štúdiu realizovateľnosti len optikou zadávateľa (zákazníka). Myslíme si však, že tieto dve strany obchodného vzťahu majú čiastočne iný pohľad a ciele:

- pre objednávateľa by mala dať odpoveď, či vôbec je potrebné investovať do plánovaného zámeru, a či navrhovaný zámer vyrieši obchodné potreby spoločnosti,
- pre dodávateľa je dôležité, či je schopný splniť objednávku načas a v stanovenom finančnom limite.

Z hľadiska integrácie aplikácií sú zaujímavé hlavne nasledovné otázky:

- máme dostatok skúseností s konkrétnymi prístupmi a metódami integrácie aplikácií, ak ich zákazník požaduje?
- vieme ľahko zapojiť už integrované riešenia u zákazníka do integrovaných aplikácií?
- máme dostatok skúseností s nástrojmi, ktoré požaduje zákazník?
- sú popísané obchodné procesy zákazníka? Ak nie, sú jasne viditeľné alebo podnik funguje „na ľade“?
- existuje veľa v súčasnosti ručných krokov? Má zákazník záujem ich v rámci integračného projektu automatizovať?
- existuje popísaná podniková architektúra? Sú popísané dátové modely, ktoré sa používajú? Aká je ich kvalita?
- udržujú sa dokumentácie aktuálne?
- je v prevádzke veľké množstvo aplikácií? Ktoré z nich bude potrebné integrovať?
- sú tieto aplikácie otvorené? Pochádzajú od rôznych výrobcov? Títo výrobcovia ešte fungujú?
- sú schopné už nasadené aplikácie komunikovať s inými aplikáciami? Používajú otvorené štandardné formáty alebo uzavreté? Ak sú uzavreté, máme skúsenosti s integráciou takýchto aplikácií?

- aká je architektúra nasadených aplikácií?

Objednávateľ by si mal predovšetkým položiť nasledujúce otázky:

- reálne pociťujeme potrebu integrovať aplikácie alebo ideme integrovať aplikácie, pretože to je moderné/chceme sa zviezť na vlne moderných pojmov dneška, akými sú webové služby, XML, SOA, .../pretože aplikácie integruje náš najväčší konkurent?
- ako by fungoval náš podnik bez integrácie aplikácií ďalej?
- aké reálne náklady by sme vedeli odstrániť integráciou aplikácií?
    - ručné importy/exporty
    - vpisovanie údajov
    - synchronizácia dátových zdrojov
- reálne zrýchlime/vylepšíme podnikové procesy?
- vieme integrovať naše portfólio aplikácií v rámci podniku (in-house) alebo integráciu zveríme externému dodávateľovi?

3.5.1. Účel a zdôvodnenie

Kladie a dáva odpovede na otázky:

Zákazník: potrebujeme integrovať aplikácie a prinesie navrhované integračné riešenie spoločnosti pridanú hodnotu?

Dodávateľ: sme schopní navrhnúť a zrealizovať integračné riešenie za prijateľných podmienok pre zákazníka (čas, cena) a oplatí sa nám to (odmena, vstup na trh, dobré meno)?

Pričom máme možnosť projekt zastaviť, ak sa to neoplatí jednej alebo druhej strane (v takom prípade je možné navrhnúť iný projekt).

3.5.2. Roly

Zákazník

- sponzor
- zadávateľ
- podnikový architekt
- správca IT

Dodávateľ

- obchodný manažér
- vedúci projektu

3.5.3. Vstupy

- znalosti o dodávateľovi
- znalosti o zákazníkovi

- znalosti o doméne problému
- znalosti o informačnom systéme zákazníka, ktorý plánuje integrovať

3.5.4. Výstupy

Štúdia realizovateľnosti (dokument), ktorá zhrňuje celú prácu vynaloženú na štúdiách, zvlášť odpovede na otázky načrtnuté vyššie alebo im podobné. Najdôležitejší výstup je však jednoznačný záver v podobe *odpovede na otázku*, či strana (zákazník alebo dodávateľ) v projekte *bude pokračovať*.

3.6. Kandidátske systémy

Počas nasledujúcich krokov sa budeme často odkazovať na jednotlivé časti existujúceho informačného systému, preto je veľmi vhodné explicitne vytvoriť a udržovať ich zoznam spolu s najdôležitejšími informáciami o nich.

Pri tvorbe zoznamu je potrebné vyvinúť nemalú prácu pri hľadaní a zisťovaní informácií o súčasnom informačnom systéme, preto tento krok podporuje získanie znalostí o ňom (a to aj rozličných znalostí nezachytených v písomnej forme).

3.6.1. Účel a odôvodnenie

- vytvorenie referenčného zoznamu systémov,
- získanie prvých znalostí o systémoch a reálnom fungovaní spoločnosti,
- vrátane prvého zistenia a popísania úrovne existujúcej integrácie v spoločnosti.

3.6.2. Tok práce

Celá práca spočíva v postupnom hľadaní a spoznávaní súčastí (aplikácií, samostatných databáz, služieb, aplikačných serverov a pod.) informačného systému spoločnosti na spoločných stretnutiach zúčastnených strán. Prvú verziu môžeme začať vytvárať už počas prác na štúdii realizovateľnosti. Do zoznamu zaraďujeme len tzv. kandidátske systémy, teda systémy, s ktorými sa s veľkou pravdepodobnosťou počíta v integračnom projekte a budú súčasťou integrovaného informačného systému; presnejšie povedané, nezaraďujeme sem súčasti, pri ktorých vieme, že tento fakt je záporný.[15] Nesmieme zabudnúť, že tak ako takmer všetky artefakty, musíme aj tento zoznam udržiavať aktuálny a v prípade známej zmeny ju doňho premietnuť.

3.6.3. Zúčastnené roly

- podnikový architekt
- správca IT
- analytik

[15] Hlavne na začiatku projektu si nemôžeme byť úplne istí, či daná súčasť má význam pre integráciu, ale o veľa systémoch vieme, resp. zákazník vie, že tento význam nemajú alebo z napr. obchodno-politických dôvodov nemôžu byť súčasťou integrovaného riešenia.

3.6.4. Vstupy

- znalosti o stave informačného systému spoločnosti (nasadených súčastiach),
- predstava o podnikovej architektúre (nemusí byť zapísaná, pozri časť 3.10 o podnikovej architektúre),
- ak má zákazník jasnú predstavu, tak aj ktoré zo súčastí chce zapojiť do navrhovaného integračného projektu.

3.6.5. Výstupy

Zoznam súčastí (entít) *informačného systému* (informačných systémov), ktoré sú v prevádzke, príp. sa plánujú uviesť do prevádzky v čase odovzdania integračného projektu. Typické súčasti sú jednotlivé aplikácie, samostatné databázy, dátové sklady, integračné servery a pod.

Je možné použiť uvedenú šablónu:

Názov:			
Skratka:		**Umiestnenie:**	
Popis:			
Správca:		**Kontakt:**	
Typ:			
Architektúra:			
Softvérové súčasti:	1.		
	2.		
	3.		
Prepojenia:	1.		
	2.		
	3.		
Možnosti integrácie:			
Poznámky:			

Tabuľka 1 Šablóna kandidátskeho systému

Legenda tabuľky:

- *názov* je plný oficiálny názov súčasti,
- *skratka* je skratka alebo značka, pod ktorou sa budeme následne v projekte odvolávať na túto časť,
- *umiestnenie* je miesto, kde sa fyzicky nachádza (napr. M185, centrála BA),
- *popis* stručne popisuje účel časti v systéme,
- *správca* je meno a priezvisko osoby zodpovednej za súčasť a *kontakt* je jeho telefón, e-mail,
- *typ* popisuje druh súčasti, napr. dátový sklad, ERP modul, webová aplikácia
- *architektúra* stručne popisuje základnú architektúru, napr. trojvrstvová aplikácia, klient/server

- *softvérové súčasti* je zoznam všetkých softvérových produktov, z ktorých sa súčasť skladá, napr.
    1. Microsoft SQL Server 2000, verzia 8.0.384
    2. Microsoft IIS 6.0
    3. ASP.NET 2.0
- *prepojenia* je len zoznam identifikovaných existujúcich prepojení s ostatnými časťami (skratky),
- *možnosti integrácie* uvádzajú, akým spôsobom alebo prístupom je možné komunikovať s danou súčasťou (napr. SQL databáza – dátová integrácia, aplikačný server – webové služby).

Príklady môže čitateľ nájsť v prípadovej štúdii UK (príloha A.).

3.6.6. Podpora CASE

Referenčný zoznam systémov je vhodné uložiť do elektronickej podoby. Ak je k dispozícii aplikácia pre správu takýchto systémov, môžeme zvážiť jej použitie. V ostatných prípadoch je vhodné použiť tabuľkový procesor.

3.7. Modelovanie podnikových procesov

Základným kameňom integračného procesu, ako sme povedali v úvode, sú podnikové procesy. Od nich odvádzame celú ďalšiu prácu na integračnom projekte, preto ich musíme mať v dostatočnej miere zmapované. Pripomíname, že dostatočná miera neznamená príliš detailná!

Pod **podnikovým procesom** rozumieme množinu vzájomne prepojených aktivít, ktoré majú pre spotrebiteľa procesu hodnotu.

Aktivita[16] je množina jednotiek práce vykonávaných **aktérmi**, ktorú na *danej úrovni abstrakcie* nerozkladáme. Aktivita teda v takomto zmysle môže reprezentovať samostatný proces (a skladať sa z ďalších aktivít) na nižšom stupni abstrakcie. Proces je iniciovaný **udalosťou** (teda výskytom vopred danej špecifickej podmienky).

Upozorňujeme, že proces musí spĺňať svojmu spotrebiteľovi nejaký cieľ. Keďže analyzujeme procesy v rámci podniku, splnenie cieľa spotrebiteľovi musí prinášať hodnotu tomuto podniku.

Hodnota, ktorú získava podnik, môže byť *hmotná* (napr. finančná vo forme zisku), ale aj *nehmotná* (napr. získanie dobrého mena, získanie znalostí alebo splnenie zákonných povinností).

Spotrebiteľ procesu je jeden z jeho aktérov a nemusí byť iba jeden.

3.7.1. Účel a odôvodnenie

Identifikáciou podnikových procesov získame

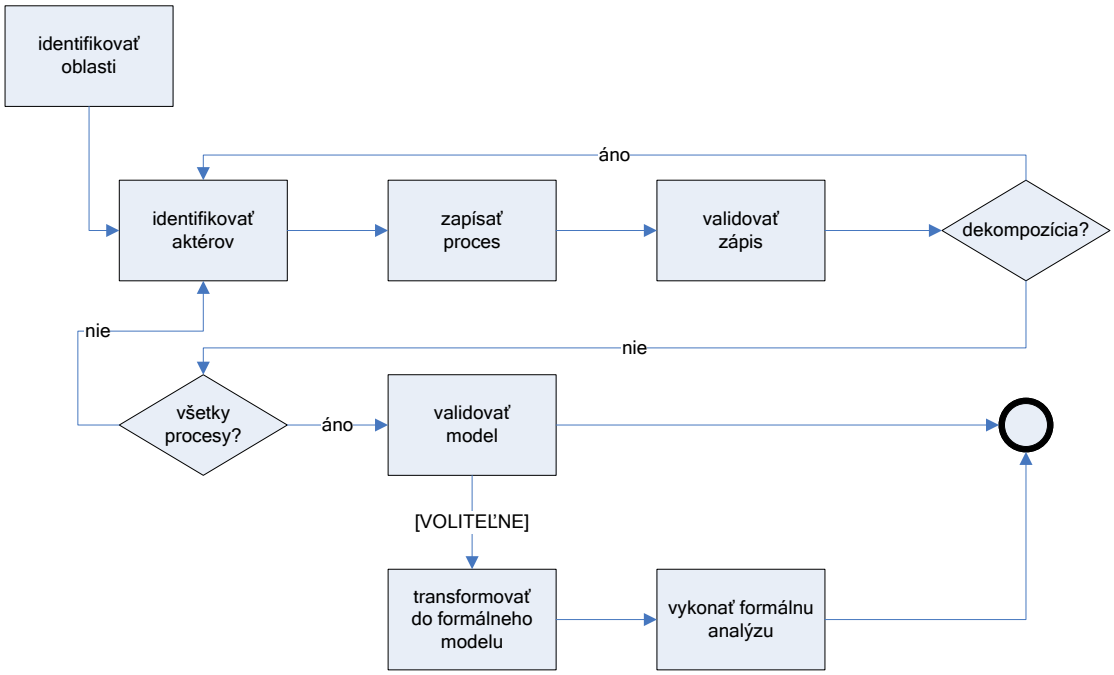
- pohľad na podnik z hľadiska jeho činností,
- explicitný vstup na ďalšie fázy integračného procesu.

[16] Niekedy sa nazýva aj *úloha* alebo *krok*.

3.7.2. Zúčastnené roly

- analytik,
- procesný architekt,
- podnikový architekt (sčasti).

3.7.3. Tok práce



3.7.4. Aktivity

Identifikovať oblasti

Prvý krok k zvládnutiu zložitosti analýzy fungovania väčších organizácií je postupovať po ucelených častiach. Ako tieto časti identifikujeme, záleží na podniku samotnom a odvetví jeho činnosti. Väčšinou identifikácia oblastí je priamočiara a v rovnakej priemyselnej doméne budú veľmi podobné.

Pravdepodobne sa nám podarí nájsť oblasti, ktoré súvisia priamo s predmetom činnosti podniku a oblasti, ktoré sú podporné.

V prípade poisťovne procesné oblasti môžu byť napríklad:

- životné poistenie,
- neživotné poistenie,
- poistné udalosti,
- zaistenie,
- underwriting.

Z doménových podporných oblastí to môžu byť:

- správa partnerov (klienti, makléri, agenti, ...),

- správa provízií,
- upomienkovanie a storná zmlúv,
- štatistika a analýza.

Zo všeobecných podporných oblastí:

- účtovníctvo, inkaso a exkaso,
- controlling,
- korešpondencia.

Ak je to potrebné, oblasti je možné ďalej deliť, napr. oblasť neživotného poistenia:

- poistenie majetku
  - zákonné poistenie motorových vozidiel (MTPL),
  - havarijné poistenie motorových vozidiel (KASKO),
  - poistenie škôd spôsobených živlom,
  - ...
- poistenie záujmov
  - poistenie úverov,
  - poistenie kaucií,
  - poistenie finančných strát,
  - ...
- ...

V tomto prípade doménové procesné oblasti sledovali pomerne priamočiaro štandardnú poistnú hierarchizáciu (poistený objekt a riziko).

Identifikovať aktérov

Pri funkčnom popise softvérových systémov (väčšinou prostredníctvom scenárov použitia, angl. *use case*) sa začína identifikáciou aktérov (Kroll, a iní, 2003), (Sharp, a iní, 2001). Dôvodom je, že „na to, aby sme sa dostali k tomu, čo systém robí, musíme sa najprv sústrediť na to, kto (alebo čo) systém používa“ (Bittner, a iní, 2003). S týmto sa stotožňujeme a môžeme túto myšlienku priamo preniesť na modelovanie podnikových procesov. Navyše podľa nás sa podnikové procesy hľadajú a popisujú ľahšie s nájdenými aktérmi, pretože tieto aktivity aspoň sčasti musíme vykonávať aj s pracovníkmi od zákazníka, ktorí sú väčšinou netechnickí (a nemajú vyvinuté dostatočne analytické myslenie) a pre nich je jednoduchšie myslieť v pojmoch funkčnosti systému ako celku pri obsluhe konkrétneho používateľa (aktéra).

Pripomíname, že aktéri nie sú iba ľudia vstupujúci do procesu, ale aj externé počítačové systémy a partneri ako takí (pri nich nie je dôležité a ani nás nezaujíma, aký systém používajú). Upozorňujeme, že sa nevnárame do technických záležitostí (aký formát dát, aká databáza a pod.), ide nám o procesy, ktoré prebiehajú v podniku na vysokom stupni abstrakcie.

Všetkých aktérov sa nám pravdepodobne nepodarí nájsť hneď, čo je v poriadku, pretože ďalších potrebných objavíme pri samotnom zápise procesu.

Zapísať proces

Samotný zápis procesu je daný zvoleným formátom (pozri časť Výstupy). Pri samotnom získavaní informácií od zákazníka neodporúčame formálne modelovacie techniky ani sa držať rigidnej syntaxe, skôr len náčrty s voľnou syntaxou v duchu klasických modelov toku práce a poznámky, kde je to potrebné. Kresliť priamo do softvérového nástroja je možné, ale nám sa skôr zdalo efektívnejšie a rýchlejšie kresliť náčrty na papier (príp. môžeme použiť tablet).

Finálne zápisy majú už väčšinou počítačovú podobu, ktorú získavame prekreslením diagramov a napísaním popisných textov do príslušného softvéru.

Validovať zápis

Procesy je nutné po zápise validovať, t.j. skontrolovať a opraviť prípadné nedostatky. Odporúčame overiť, či zapísaný proces je:

- *úplný*, teda sú v ňom zachytené všetky dôležité aktivity,
- *reálny*, teda či naozaj prebieha tak, ako je zapísaný (pozri aj časť Odporúčania),
- *konzistentný*, teda či jeho aktivity si navzájom neprotirečia,
- *neredundantný*, teda či nemá aktivity s rovnakou alebo veľmi podobnou náplňou práce.

Rozhodovanie Dekompozícia?

V prípade, že aktivity namodelovaného procesu sú zložité, nie sú to jednoduché elementárne úlohy, prípadne sa skladajú z viacerých takýchto úloh s netriviálnou následnosťou, môže byť vhodné túto aktivitu ďalej namodelovať ako samostatný proces. Takýmto spôsobom môžeme v prípade potreby dekomponovať aj skupinu aktivít (pozri napr. model podnikového procesu prijímacieho konania v podkapitole 4.9).

Nie je však úlohou integračného projektu zdokumentovať dôsledne všetky podnikové procesy do posledného detailu, preto s dekompozíciou musíme byť opatrní. Otázka, či máme robiť dekompozíciu, je komplikovaná a odpoveď nie je nikdy jednoznačná, ale treba brať do úvahy:

- ak sa rozhodneme robiť formálnu analýzu, nutnosť dekomponovať vyplýva z toho, či aktuálna úroveň nesie dostatok informácií pre ňu,
- ak model plánujeme použiť na transformáciu do modelu nižšej úrovne, prípadne z neho generovať kód, opäť to závisí od množstva a povahy detailov, ktoré chceme preniesť,
- ak modelujeme iba kvôli primárnemu účelu (nerobíme formálnu analýzu ani transformácie), úrovní dekompozície by nemalo byť viac ako dve, tri.

Validovať model

Validáciou modelu overujeme, či má model ako celok požadované atribúty kvality. Jednotlivé časti sme už overili (Validácia zápisu), tu nám ide o to, aby:

- jednotlivé časti boli konzistentné,
- časti boli funkčne disjunktné,
- pri dekomponovaní väčšieho procesu uviedli všetky jeho časti na nižšej úrovni.

Transformovať do formálneho modelu

Ak sa rozhodneme využiť niektorý z formálnych modelov, či už na generovanie kódu alebo formálnu analýzu (pozri stať Odporúčania), musíme výsledný model doňho preložiť. Konkrétna podoba závisí na cieľovom modeli a použitých nástrojoch. V niektorých prípadoch musíme model podnikových procesov dostatočne obohatiť niektorými metainformáciami, ktoré prekladač do formálneho modelu potrebuje.

Transformuje sa zvyčajne na (predpokladanom) konci analýzy, prípadne na konci analýzy väčšieho celku alebo plánovaného inkrementu.

Vykonať formálnu analýzu

Po transformácii môžeme spustiť všetky formálne analýzy na modeli podnikových procesov (pozri aj stať Odporúčania).

3.7.5. Vstupy

- znalosti o podniku a jeho procesoch,
- existujúce modely podnikových procesov (ak existujú) – zvážiť úpravy alebo prepis.

3.7.6. Výstupy

Výstupom je model podnikových procesov väčšinou reprezentovaný diagramami a doplnkovými popismi a prípadne implicitným modelom v pozadí v prípade použitia modelovacích softvérových nástrojov (pozri aj časť Odporúčania).

Každá procesná oblasť by mala mať aj úvod, ktorý zbežne formou voľného textu prezentuje, ktoré dôležité procesy a aktivity popisuje a vysvetľuje dôležité koncepty a pojmy, ktorých význam nemusí byť úplne jasný zvlášť ľuďom mimo zadávateľského tímu.

V prípade, že používame grafické alebo modelovacie nástroje, je grafická reprezentácia procesov vo forme diagramov obmedzená možnosťami týchto nástrojov. Väčšina podnikových analytikov používa tzv. formát plaveckých dráh (swimlane diagrams), ktoré poznáme s drobnými odchýlkami pod mnohými ďalšími názvami (napr. procesná mapa, workflow process model, responsibility process matrix, people-process chart) (Sharp, a iní, 2001). Z tohto konceptu vychádzajú aj diagramy aktivít jazyka UML (Booch, a iní, 1998). *Object Management Group* ponúka okrem diagramov aktivít v UML aj komplexnejší štandard notácie zameranej iba na modelovanie podnikových procesov – *Business Process Modeling Notation* (OMG, 2006).

Reálne príklady môže čitateľ nájsť v podkapitole 4.9.

3.7.7. Odporúčania

V prípade, že integračný tím sa podujme modelovať podnikové procesy pomocou počítačovej podpory, odporúčame použiť modelovací softvér. Modelovací softvér na rozdiel od obyčajných kresliacich nástrojov udržuje v pozadí model kreslených diagramov, teda zoznam všetkých nakreslených objektov, ich vlastností a vzťahov. V takomto prípade je neskôr možné takéto diagramy spracovávať automatizovane, napr. pri modelovo riadenej integrácii (pozri podkapitolu 2.6). V praxi sme už využili možnosti takéhoto spracovania pri automatickom preklade procesných diagramov do štyroch jazykov v nadnárodnom projekte (bol použitý softvér Microsoft Office Visio Professional 2003). Nevýhodou je väčšinou vyššia počiatočná investícia na zakúpenie licencie, ak ju nevlastníme.

Ďalšia veľmi zaujímavá možnosť, ktorú nám dávajú modelovacie nástroje, je počítačová analýza zachytených procesov. Analýza môže prebiehať po (automatickej) transformácii do úplne formálneho modelu, napr. Petriho sietí, v ktorých dokážeme ekvivalentne sformulovať niektoré problémy v oblasti podnikových procesov (Petri Nets: Properties, Analysis and Applications, 1989), (Störrle, 2004), (Meena, et al., 2005). V súčasnosti je realita taká, že dokážeme aplikovať tento model len na značne obmedzené problémy (napr. každý dostupný medzistav sa ukončí, každý tok riadenia je prístupný), preto výskum v tejto oblasti je veľmi aktívny (Zhu, a iní, 2004). Podľa nás bude zaujímavá hlavne detekcia rozličných konfliktov a úplnosti celého modelu podnikových procesov.

V súčasnosti sa veľkej obľube tešia rôzne vzory, ktoré v informatike spopularizovala predovšetkým kategória návrhových vzorov. Vzor predstavuje opakovane použiteľný náčrt riešenia často sa vyskytujúceho problému. Hoci väčšina vzorov nie je zložitá, ich poznanie prináša priame aplikovanie znalostí, ktoré reprezentujú, bez nutnosti ich explicitne objavovať znova. Rozpoznanie možnosti aplikovania istého známeho vzoru prináša aj uistenie, že podobný problém riešilo už mnoho ľudí (a pravdepodobne ho aplikovaním vzoru úspešne vyriešilo, keďže bol publikovaný). Doména modelovania procesov a tokov práce nie je výnimkou a objavili sa už prvé katalógy vzorov z tejto oblasti (Aalst, a iní, 2003), (Russel, a iní, 2006), (Russel, a iní, 2006), (Havey, 2005).

Niekedy má analytik problém zachytiť proces tak, ako prebieha v spoločnosti. Dôvodom môže byť buď zastaraná dokumentácia a neaktuálne procesné mapy alebo, čo je horšie fakt, že ľudia pracujú inak ako stanovujú podnikové procesy, ak ich organizácia má v písomnej podobe. Prvoradou úlohou je vôbec zistiť tento stav, v čom nám môžu pomôcť napríklad niektoré metodiky zberu požiadaviek nových softvérových systémov (etnografia, pozri (Sommerville, 2004) a iné). Ďalšie kroky už závisia na konkrétnej situácii, ale vždy je dôležité nájsť dôvod, prečo tak ľudia robia, aj keď to nemusí byť jednoduché, zvlášť v prípadoch, kedy pracovníci musia dodržiavať procesné dokumentácie a nerobia tak. Stane sa tak, že vlastne istým spôsobom vylepšia proces, ale toto vylepšenie nie je odrazené v dokumentácii, preto je veľmi dôležité vedieť, ako sa v podniku pracuje reálne.

Pokladáme za dôležité pridať aj varovanie, aby sme procesy nedekomponovali na príliš nízke úrovne a nepreplnili modely zbytočnými detailmi. Je tu ukryté riziko, pretože zbytočné detaily uberajú projektové prostriedky a hlavne pôsobia demotivujúco jednak na analytikov, ktorí modely pripravujú

a takisto na neskorších používateľov týchto modelov. Situácia, keď dôjde projekt na mŕtvy bod z dôvodu príliš detailnej a dlhotrvajúcej analýzy, je reálna (známy manažérsky anti-vzor „paralýza analýzou“ (Brown, a iní, 1998)).

3.8. Vylepšovanie procesov

Integrovaním aplikácií sa potenciálne môže otvoriť cesta na vylepšenie niektorých podnikových procesov. Automatizáciu činností (odstránenie ručného vpisovania údajov, ručných importov a exportov) považujeme za samozrejmosť.

Situáciu, kedy môže ovplyvniť integrácia aplikácií podnikový proces, by sme mohli ilustrovať na príklade zápisov predmetov s obmedzenou kapacitou na cudzích fakultách. Zápis takýchto predmetov funguje teraz tak, že študent si predmet zapíše do indexu na svojej fakulte (fakulta A) a svoju požiadavku oznámi študijnej referentke, ktorá mu predmet predbežne zapíše. Prednostné právo na predmety s obmedzenou kapacitou majú študenti domácej fakulty (fakulta B), aj tí sú vyberaní podľa istých kľúčov (napríklad podľa študijného priemeru alebo absolvovania predmetov podobného zamerania). Po ukončení zápisov na fakulte A sa posiela zoznam študentov fakulte B, ktorá pošle odpoveď o možnosti zápisu jej študentov. Napriek tomu, že na fakulte B zápis prebieha o tri týždne skôr, je tento postup nutný, pretože systémy medzi fakultami nekomunikujú. Integrovaním oboch systémov v takmer reálnom čase vieme vykonať overenie, či predmet na fakulte B má ešte voľné kapacity. Reálne sme zmenili podnikový proces zápisu externých predmetov – odpadlo dodatočné overenie, študent má zapísané svoje predmety ihneď a definitívne.

Keďže vylepšovanie procesov závisí silne nielen od podnikovej domény, ale aj od konkrétnych činností procesu, je veľmi ťažké popísať konkrétne kroky.

Odporúčame postupovať podľa niektorých zabehnutých praktík, ktoré čitateľ nájde napríklad v (Sharp, a iní, 2001) alebo (Persse, 2006).

3.9. Podnikové objekty

Častým problémom v organizáciách rôznej veľkosti je neúplná, nekvalitná či dokonca chýbajúca dokumentácia svojich informačných systémov. Na jednej strane je to logické, pretože aplikácie v informačnom systéme žijú často svoj vlastný život a udržujú ich rôzni ľudia, a preto sú zdokumentované rôzne. Okrem toho dokumentácia môže byť na rozmanitej úrovni.

Analýza požiadaviek pre integráciu si žiada rozumne jednoduchý pohľad, dostatočne vysokú abstrakciu na „veci“, ktoré jednotlivé aplikácie ponúkajú a ktoré naopak potrebujú.

Detailný dátový model (a fyzický aj mnohokrát logický také sú) je pre nás neužitočný, pretože nie je na dostatočne vysokom stupni abstrakcie pre riešenie tak komplexného problému, akým je integrácia aplikácií, ktorý vyžaduje pohľad nad celú podnikovú architektúru. Dátové modely nás zaťažujú svojimi detailmi, akými sú dátové typy atribútov, ich presnosť, počet znakov, kódovanie, názvové konvencie a podobne, ale aj technickými záležitosťami dobrého databázového návrhu (normalizácia, optimalizácia, many-to-many vzťahové tabuľky).

Podnikový objekt je každá osoba, miesto, udalosť, predmet, ..., o ktorej potrebujeme udržiavať informácie, aby podnik mohol správne a efektívne

fungovať. Podnikový objekt reprezentuje koncept v podnikovej doméne na vysokom stupni abstrakcie.

Dobrými príkladmi podnikových objektov v doméne študijnej agendy by mohli byť napr. študent, učiteľ (osoby), študijný program (predmet), učebňa (miesto), začiatok semestra (udalosť). Na zváženie sú napr. knihovník (naozaj potrebný pre študijnú agendu?), zoznam PSČ (na nízkej úrovni abstrakcie ako samostatný objekt, ale v prípade, že sa jedná o celopodnikový číselník, zvážiť integráciu, pozri aj 3.11.7), študent-telefón (vyzerá na prebratie vzťahovej tabuľky z databázy, ako samostatný objekt na nízkej úrovni, lepší je atribút telefón objektu študent, ale študent-predmet, hoci s vhodnejším pomenovaním – napr. zapísaný predmet, by samostatný objekt mohol byť, pretože reprezentuje dôležitý koncept v našej doméne).

Podnikové objekty budú predstavovať náš dôležitý zdroj pri úvahách o tom, čo je pred podnik dôležité a aké sú jeho potreby v oblasti integrácie. Základný zoznam podnikových objektov je možné pripraviť už po alebo počas modelovania podnikových procesov. V takom prípade nám iste pomôžu aj pri vylepšovaní týchto procesov, ak sa tak rozhodneme urobiť. Ďalšia analýza a ich rozpracovanie, predovšetkým otázky ich autoritatívneho zdroja, rozdielov v chápaní medzi jednotlivými systémami a pod., príde na rad po charakteristikách týchto systémov (pozri ďalej). V ďalšom texte sa môžeme odvolávať na podnikovú úroveň alebo systémovú úroveň, čo sa dá chápať v kontexte procesu aj ako miesto, kedy sa podnikovými objektmi zaoberáme (po alebo počas modelovania podnikových procesov, resp. po alebo počas charakteristík systémov).

Podnikové objekty sa používajú pri analýze a definovaní podnikových architektúr, odkiaľ pochádza aj naša inšpirácia (McGovern, a iní, 2004). Nové využitie pri analýze potrieb integrácie sa nám osvedčilo, preto je podstatnou súčasťou navrhovaného procesu integrácie aplikácií.

3.9.1. Účel a odôvodnenie

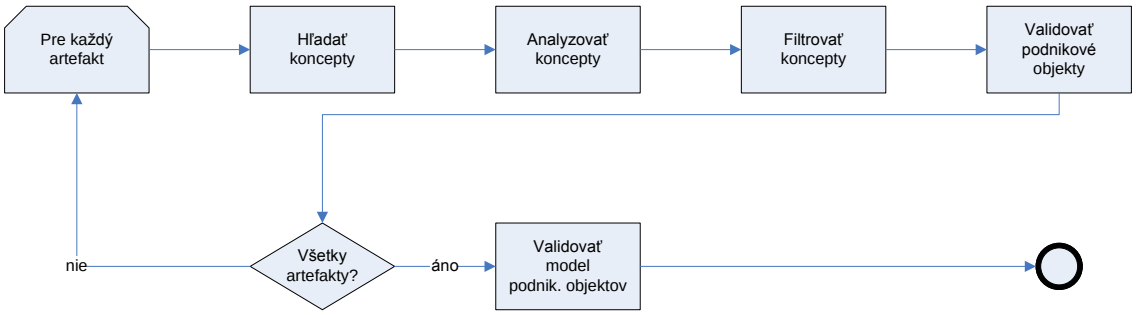
Identifikácia a zápis podnikových objektov nám umožňuje

- rýchlo a efektívne zistiť, ktoré podstatné informácie sú v podniku potrebné,
- ktoré aplikácie tieto informácie udržujú a ktoré by mohli mať o ne záujem,
- ktoré z týchto informácií sú už zdieľané.

3.9.2. Zúčastnené roly

- analytik,
- procesný architekt.

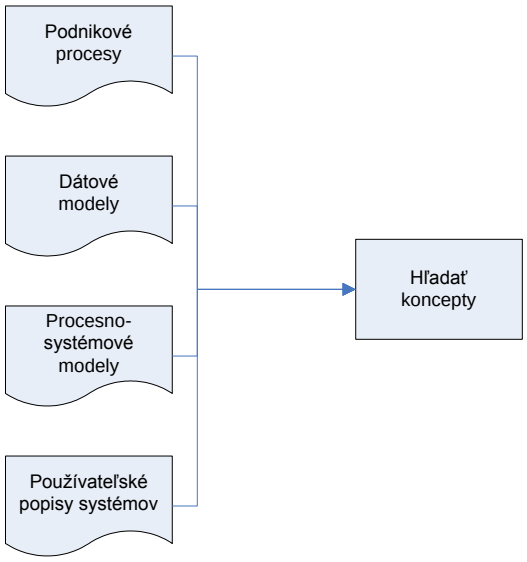
3.9.3. Tok práce



3.9.4. Aktivity

Vyhľadať koncepty

Prvá nosná aktivita pri zostavovaní podnikových objektov nápomocných pri integrácii, je ich samotná identifikácia. Keďže podnikové objekty reprezentujú koncepty s istými vlastnosťami, hľadáme vhodné koncepty. Mali by sme už tu myslieť na ich využitie pri integrácii, teda sú pre nás zaujímavé iba koncepty využiteľné v iných softvérových entitách (aplikáciách), prípadne koncepty pochádzajúce z iných entít. V tejto fáze však nejdeme do hlbšej analýzy, len si pripravujeme takpovediac pôdu pre ďalšie kroky.



Koncepty sa snažíme hľadať v štyroch hlavných zdrojoch (artefaktoch):

- podnikových procesoch, ktoré využijeme na začiatku,
- procesno-systémových modeloch, kde sa už ukážu na jemnejšej úrovni a v kontexte príslušnej entity (časti) informačného systému,
- dátových modeloch, v prípade, že entitou je surová databáza, príp. aplikácia integrovateľná len na dátovej úrovni,
- používateľských popisoch systémov, ktoré sú užitočné pri hľadaní konceptov pre jeden konkrétny systém.

V používateľských popisoch systémov majú väčšinou podobu podstatných mien.

Prvé koncepty však prichádzajú často aj z neformálnych alebo formálnych stretnutí s prevádzkovateľmi či používateľmi systémov – na toto nesmieme

zabúdať, hoci sa pravdepodobne budú upravovať, keď budeme mať k dispozícii vyššie spomenuté artefakty.

Analyzovať koncepty

Na podnikovej úrovni analýza v prvom rade znamená:

- overenie, či koncept je podnikovým objektom v zmysle našej definície (pozri aj nevhodné príklady vyššie),
- v prípade, že sa nepúšťame do vylepšovanie procesov, prvotné overenie, či podnikový objekt je zaujímavý z hľadiska integrácie (tzn. existuje reálny predpoklad, že iné aplikácie ho potrebujú na svoju prevádzku, príp. je vyžadovaný od nejakej inej aplikácie),
- zistiť aké sú jeho prirodzené identifikátory,
- zistiť aké sú jeho ďalšie vlastnosti na podnikovej úrovni.

Na systémovej úrovni ďalej analyzujeme:

- ktorá aplikácia v informačnom systéme môže poslúžiť ako autoritatívny zdroj podnikového objektu,
- ako sa líši podnikový objekt naprieč spektrom aplikácií, syntakticky aj sémanticky,
- ktoré aplikácie v informačnom systéme tento podnikový objekt potrebujú,
- aké informácie o podnikovom objekte tieto aplikácie potrebujú, ktoré z nich potrebujú, ak sú objekty vedené ako množina (napr. z osôb len učiteľov, len predmety z odboru informatika),
- akým spôsobom sa v súčasnosti tieto objekty zdieľajú.

Filtrovať koncepty

V integračnom projekte nás zaujímajú predovšetkým objekty majúce význam z hľadiska integrácie aplikácií, teda vylučujeme a zbytočne nezneprehľadňujeme model podnikových objektov objektmi, ktoré sa používajú výlučne v rámci kontextu jednej súčasti (aplikácie) informačného systému a nie sú ani potenciálne využiteľné mimo nej.

Validovať podnikové objekty

Validácia overuje, či práve prebiehajúci výsledok zberu a analýzy podnikových objektov z artefaktu je korektný, hlavne:

- podnikové objekty sú využiteľné z hľadiska integrácie,
- model je úplný (nezabudli sme na významný objekt),
- konzistentný (v zásade nemáme tie isté podnikové objekty pod inými názvami, „falošné“ podnikové objekty, ...),
- a dostatočne abstraktný (pozri úvodnú stať) .

Validovať model podnikových objektov

V podstate kontrolujeme veci ako pri validácii jedného artefaktu, ale zameriavame sa na celý model, teda odhaľujeme konflikty, nepresnosti a neúplnosť vzhľadom na všetky artefakty.

Navyše je veľmi vhodná v tomto kroku revízia objektov podnikovým architektom.

3.9.5. Vstupy

- informačný ekosystém,
- zoznam častí informačného systému,
- model podnikových procesov,
- dátové modely, procesno-systémové modely, používateľské popisy systémov (v neskorších fázach).

3.9.6. Výstupy

Základný zoznam podnikových objektov, ktorý obsahuje kľúčové informácie o každom identifikovanom podnikovom objekte:

- *názov* – všeobecne akceptovaný názov podnikového objektu,
- *popis* – všeobecne uznávaná stručná charakteristika podnikového objektu voľným textom,
- *identifikátory* – vlastnosti, umožňujúce identifikovať konkrétnu inštanciu podnikového objektu od iných. Tieto vlastnosti sú na vysokej abstraktnej úrovni (ako samotné podnikové objekty), nemusia to byť exaktne diferencujúce atribúty (ako napr. primárne kľúče v relačnej databáze).
- *ďalšie vlastnosti* – ostatné vlastnosti, ktoré je potrebné viesť v informačnom systéme pre správne fungovanie spoločnosti.
- *pozn.* – poznámky, väčšinou príklady a súvislosti.

3.9.7. Odporúčania

Ako sme spomenuli v úvode tejto kapitoly, pri menších integračných projektoch model podnikových procesov nemusí byť explicitne zapísaný a čiastočne ho nahrádzajú procesno-systémové modely – v týchto prípadoch začíname až po kroku Charakteristika systémov.

Ak sme zahájili hľadanie podnikových objektov už počas alebo hneď po dokončení modelu podnikových procesov, majme na pamäti, že zoznam, ktorý vytvoríme, určite nebude mať finálnu podobu, preto s ním nestrácame neúmerné množstvo času. V každom prípade však odporúčame takýto spôsob práce, pretože z modelu podnikových procesov sa podnikové objekty hľadajú veľmi jednoducho, vzhľadom na to, že oba modely sú na podobnej úrovni abstrakcie a keď sa pozrieme na situáciu z vhodného uhla, tak podnikové procesy vlastne dávajú formu množine podnikových objektov – na podnikové procesy sa dá hľadieť, akoby boli z nich zložené.

3.10. Charakteristika systémov

Doteraz bolo našou snahou zistiť, aké podnikové procesy riadia prácu v spoločnosti a ktoré koncepty sú pri tom používané. Výsledky týchto aktivít nezáviseli na žiadnej technológii ani na žiadnom konkrétnom systéme. Sú odrazom toho, ako *spoločnosť funguje bez ohľadu na to, ako sú v konečnej fáze implementované.* Ako sme už spomenuli, takéto oddelenie nám prináša čistejší pohľad na reálnu podstatu spoločnosti a dovoľuje nám sústrediť sa na to, čo naozaj potrebuje na splnenie cieľov.

V tomto kroku sa pokúsime vystihnúť, ako sú podnikové procesy reálne implementované a ako systémy nakladajú s identifikovanými konceptmi (podnikovými objektmi).

Navyše môžu existovať aplikácie, pre ktoré nemá zmysel modelovať podnikové proces, napr. dynamické webové prezentácie, ktoré nemajú interaktívnu podnikovú funkčnosť (zobrazujú iba informácie) alebo ako sme spomenuli, niekedy model podnikových proces netvoríme vôbec – v týchto prípadoch si vystačíme iba s ich charakteristikou.

3.10.1. Účel a zdôvodnenie

- prehľad o implementácii podnikových procesov,
- vstup pre mapovanie podnikových objektov k častiam (entitám, aplikáciám) informačného systému,
- vstup pre hľadanie ďalších podnikových objektov (ktoré nevieme nájsť v modeloch podnikových procesov alebo kde sme ich nerobili),
- vstup pre integráciu podnikových a univerzálnych dát a služieb,
- vstup pre návrh.

3.10.2. Zúčastnené roly

- analytik,
- zadávateľ,
- podnikový architekt,
- správca IT.

3.10.3. Tok práce

Konkrétny tok práce závisí od výstupov. Vo všeobecnosti sa postupuje vo viacerých iteráciách sedení zákazníkovho tímu s analytikmi. Pri používateľských popisoch systémov môžeme využiť už dokumentované techniky zberu požiadaviek pomocou scenárov použití (use case). Procesno-systémové modely pripravujeme na sedeniach priamo z príslušných modelov podnikových procesov (za predpokladu, že procesné modelovanie vykonávame).

3.10.4. Vstupy

- zoznam kandidátskych systémov,

- model podnikových procesov,
- znalosti používateľov a zákazníkovho tímu o systémoch a fungovaní informačného systému podniku.

3.10.5. Výstupy

Ako sme už načtli, fáza charakteristík systémov môže mať viacero výstupov. Podniková architektúra nám dáva „veľký obraz" nad aplikáciami a inými zložkami informačného systému. Používateľské popisy systémov charakterizujú aplikácie z hľadiska koncového používateľa. Databázy a dátové časti aplikácií zachytávajú klasické dátové modely. Implementáciu procesov popisujeme procesno-systémovými modelmi. Najdôležitejším výstupom je interakčná matica podnikových objektov. V ďalšom texte tieto výstupy popíšeme detailnejšie.

Podniková architektúra je typom architektúry, v ktorej ako elementy (pozri definíciu na začiatku kapitoly) vystupujú „časti", z ktorých sa skladá informačná infraštruktúra podniku. Nie je exaktne zadefinované, čo by malo byť a nemalo byť jej súčasťou, v zásade tu ponechávame voľnú ruku a všetko, čo pomôže ozrejmiť fyzický aj logický pohľad na informačné systémy v podniku, je vítané. Typicky sú tu zakreslené servery, databázy, pracovné stanice, aplikácie, faxy, ale aj používatelia, firewally či siete. Vidíme, že sa tu môže vyskytovať pestrá zmes fyzických aj logických konceptov, dokonca zmiešaných na jednom papieri (obrazovke).

Používateľské popisy systémov popisujú interakcie konkrétneho systému s používateľom. Odporúčame, aby mali podobu zápisu scenárov použitia (angl. use case). V prípade, že nie sú k dispozícii aktuálne (z časov, keď sa zbierali na systém požiadavky), môžeme ich napísať sami, na čo môžeme použiť patrične upravené klasické metódy zberu požiadaviek (pozri napr. (Sommerville, 2004), (IEEE, 1998), (Wiegers, 2003)). Ak je to vhodné, môžeme pridať aj use case diagramy, ktoré prezentujú obrazovou formou celok (kto a v akej veci komunikuje so systémom). Dôležité sú ale práve scenáre použitia, ktoré zapisujeme ako voľný text. Nasledovná šablóna sa nám osvedčila:

Názov:			
Úroveň:		**ID:**	
Systém:			
Iniciátor:			
Vstupná podmienka:			
Hlavný scenár:	1. *krok* 2. *krok* 3. *krok* 4. *krok*		
Rozšírenia:	1a: *podmienka alebo popis rozšírenia v 1. kroku hl. scenára* 1. *krok rozšírenia* 2. *krok rozšírenia* 1b: *podmienka alebo popis rozšírenia v 1. kroku hl. scenára* 1. *krok rozšírenia* 3a: *podmienka alebo popis rozšírenia v 3. kroku hl. scenára*		

	1. *krok rozšírenia* 2. *krok rozšírenia*

Tabuľka 2 Šablóna používateľského popisu systému

Legenda tabuľky:

- názov: stručný popis scenára, ktorý by mal odrážať cieľ iniciátora,
- úroveň: na ktorej úrovni sa scenár použitia nachádza, v prípade, že scenáre dekomponujeme,
- ID: identifikácia scenára kvôli referenciám,
- systém: interakciu s ktorým systémom popisuje (identifikácia zo Zoznamu kandidátskych systémov),
- iniciátor: aktér (výnimočne aktéri), ktorí iniciujú hlavný scenár,
- vstupná podmienka: podmienka, ktorá musí platiť pri začatí interakcie,
- hlavný scenár: kroky popisujúce interakciu so systémom. Kroky číslujeme a mali by mať formu jednoduchých viet.
- rozšírenia: odklonenia od hlavného scenára. Každé rozšírenie začína podmienkou, za ktorou sa aktivuje a alternatívnymi krokmi.

Môžeme použiť v prípade potreby aj ďalšie polia v tabuľke (napr. výstupné podmienky) a takisto niektoré techniky zo scenárov použitia, na ktoré sme zvyknutí (zovšeobecnenie, vkladanie, ...), ale opäť musíme zdôrazniť, že pri integračnom projekte nám nejde o detailné zdokumentovanie systémov, ale iba o podporu pre hľadanie podnikových objektov. My sme si na integračnom projekte UK vystačili viac-menej s touto šablónou.

Dátové modely zachytávajú štruktúru dát používaných v podniku, väčšinou v rámci jednej databázy či aplikácie. Takmer vždy majú podobu relačného dátového modelu, čo odráža realitu v implementačnom svete. Dátové modely, aj princípy dátového modelovania sú veľmi dobre pokryté literatúrou, preto sa im ďalej nebudeme venovať (pozri napr. (Muller, 1999) alebo (Simsion, a iní, 2005)).

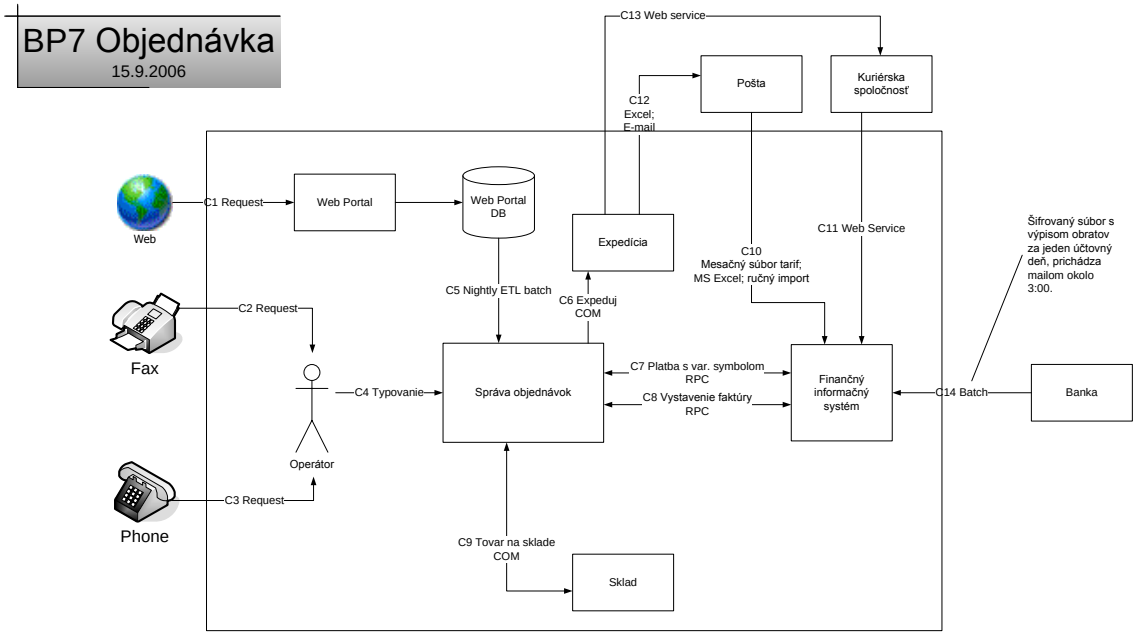
Procesno-systémové modely sú modely znázorňujúce implementáciu podnikových procesov softvérovou infraštruktúrou. Na rozdiel od modelov podnikových procesov, ktoré sú nezávislé na technológii a nemali by byť pri zmene podnikovej architektúry alebo zásahu do počítačových systémov zmenené, procesno-systémové modely sú odrazom toho, ako prebieha podnikový proces cez infraštruktúru podniku.

Procesno-systémový model pozostáva z popisu a diagramu, ktorý znázorňuje aktérov, súčasti (aplikácie) informačného systému, ktorými je proces podporovaný a prípadne iné objekty podľa uváženia, spojené komunikačnými väzbami. Každá väzba má svoj identifikátor a krátky popis, ktoré sú nakreslené na diagrame kvôli ľahšej orientácii (pozri obrázok 3-4). V popise je slovne zachytený kontext modelu, uvedené, ktorý proces znázorňuje a ďalej popísané komunikačné väzby.

Komunikačné väzby môžeme stručne popísať v tabuľke s nasledujúcou šablónou:

ID	Rozhranie	Popis

Príklad reálne vyplnenej tabuľky môže čitateľ nájsť v kapitole 4.10.



Obrázok 3-4 Príklad diagramu procesno-systémového modelu pre fiktívny podnikový proces vybavenia objednávky.

Interakčná matica podnikových objektov zachytáva nasledujúce vzťahy medzi podnikovými objektmi a systémami informačného systému (na pozícii danej súradnicami podnikový objekt *O* a systém *S*):

- systém *S* je autoritatívnym zdrojom objektu *O*,
- systém *S* je v súčasnosti konzumentom objektu *O*,
- systém *S* bude potenciálnym konzumentom objektu *O,*
- alternatívny názov pre objekt *O*, ktorý sa používa v kontexte systému *S*,
- odchýlky vo význame objektu *O* v kontexte systému *S*.

Interakčná matica môže mať prakticky tri podoby: priamy zápis, rez v dimenzii systémov a rez v dimenzii podnikových objektov.

Priamy zápis tvorí klasická tabuľka, ktorej riadky a stĺpce reprezentujú podnikové objekty a systémy. V zásade je jedno, do ktorej dimenzie umiestnime podnikové objekty a do ktorej systémy, ale vzhľadom na veľký počet podnikových objektov k systémom je praktickejšia verzia s podnikovými objektmi v riadkoch a systémami v stĺpcoch. V praxi na papieri alebo v tabuľkovom procesore máme pomerne obmedzený priestor na informácie, ktoré uchovávame v bunke na priesečníku podnikového objektu *O* a systému *S*.

Práve kvôli problémom s miestom sa nám osvedčilo na rozšírený zápis informácií o podnikových objektoch a systémoch použiť rezy v jednej alebo druhej dimenzii. Rez v dimenzii systémov zoskupuje všetky podnikové objekty vzťahujúce

sa k jednému systému. V takomto reze sa nám osvedčil nasledovný formát zápisu údajov:

Systém A (Názov systému)

Názov PO	Zdroj	Konzument	Odchýlky	Poznámky
objekt 1	Á			
objekt 2		Á		

Tabuľka 3 Šablóna rezu zoznamu podnikových objektov zhľadom na systém

Legenda tabuľky:

- *názov PO*: názov podnikového objektu, aký je v základnom zozname,
- *zdroj*: Á, ak je systém zdrojom pre tento objekt,
- *konzument*: Á, ak systém má o tento objekt záujem,
- *odchýlky*: syntaktické alebo sémantické odchýlky objektu v tomto systéme od všeobecne chápaného významu a formátu.

3.11. Návrh architektúry

Doteraz sme sa snažili zistiť, aké informácie a procesy vôbec majú byť zdieľané. Teraz sa zameriame na spôsob, akým budú zdieľané.

Integračná architektúra je typ architektúry, v ktorej elementy sú prvky (entity) podnikového systému (softvérové systémy, ľudia, externí partneri, ...) a porty, pričom vzťahy reprezentujú komunikačné linky, prostredníctvom ktorých dochádza k cielenému zdieľaniu dát a procesov. V integračnej architektúre teda nezakresľujeme konkrétnu výmenu dát či volanie procesov. Porty sú ukončenia komunikačných kanálov v entitách podnikového systému.

Integračná architektúra inými slovami popisuje štruktúru kanálov, prostredníctvom ktorých budú aplikácie zdieľať svoje dáta a procesy.

Okrem toho sú pre nás v integračnej architektúre dôležité aj typy portov, ktoré minimálne špecifikujú spôsob a úroveň integrácie (dátová, integrácia cez rozhrania služieb a pod.) – tieto sú externe viditeľnými vlastnosťami portov a nepriamo aj entít a kanálov, ktoré spájajú.

Návrh tejto architektúry je mimoriadne dôležitý, pretože priamo ovplyvňuje viacero kritérií kvality celého integračného riešenia. Ako príklad by sme mohli uviesť tzv. problém $O(n^2)$, ktorý spomína vari každá literatúra o integrácii aplikácií (autori knihy (Hohpe, a iní, 2003) majú na to pekný výraz – integračné špagety). V krátkosti sa pod týmto názvom označuje integračná architektúra, v ktorej existuje aspoň jeden komunikačný kanál medzi každou dvojicou entít, ktoré spolu komunikujú. Celkový počet komunikačných kanálov je tak $\frac{n(n-1)}{2}$ alebo $n(n-1)$ v prípade jednostranných kanálov v závislosti od počtu entít. Vidíme, že s rastúcim počtom entít stúpa počet kanálov kvadraticky, čo je veľmi zlé vzhľadom na vývoj i údržbu. Architektúry s podobnou tendenciou budeme volať $O(n^2)$ architektúry alebo architektúry bod-bod.

Riešením spomínaného problému je vloženie centrálneho komunikačného systému, ktorý sám rieši komunikáciu medzi jednotlivými zložkami podnikového systému. V takomto prípade stačí, ak bude existovať spojenie medzi entitou

a týmto centrálnym middlewarom (čo dáva $O(n)$ komunikačných kanálov). V praxi má tento centrálny komunikačný systém podobu buď hubu alebo zbernice (viac informácií o týchto modeloch napr. v (Throwbridge, a iní, 2004), porovnanie na (Bakker, 2005)).

Pri návrhu integračnej architektúry sa vlastne prvýkrát posúvame na technologickú úroveň. Návrhár (architekt) musí mať teda znalosti tak z technologickej, ako aj obchodno-analytickej oblasti.

Upozorňujeme, že návrh architektúry je krok pozostávajúci z viacerých postupných vylepšení (iterácií).

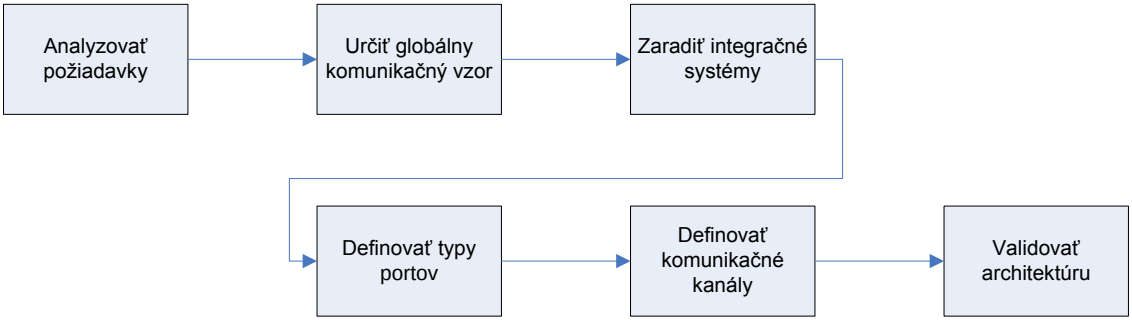
3.11.1. Účel a odôvodnenie

Architektúra má priamy i nepriamy dosah na množstvo z kritérií kvality, pričom robiť zmeny v nej po začatí implementácie je veľmi zložité.

3.11.2. Zúčastnené roly

- analytik (hlavne dodáva a popisuje vstupy),
- návrhár.

3.11.3. Tok práce



3.11.4. Aktivity

Analyzovať požiadavky

Ako sme už spomenuli, architektúra podstatnou mierou priamo ovplyvňuje kvalitu výsledného riešenia integračného projektu a súčasne je veľmi ťažké robiť zásahy do nej po nasadení, preto jej návrhu musíme venovať patričnú pozornosť už od prvých momentov.

V prípade, že máme vopred stanovené požiadavky, na ktoré architektúra musí zodpovedajúcim spôsobom reflektovať, musíme ich do návrhu zapracovať. Dôležité je pritom aj zistiť, či vôbec takéto požiadavky sú a identifikovať ich, pretože málokedy ich dostaneme explicitne spísané od zákazníka.

Z tejto fázy môže vystúpiť samostatný dokument kladúci požiadavky na architektúru.

Určiť globálny komunikačný vzor

Následne treba rozhodnúť o nasadení centrálneho komunikačného systému a v prípade kladnej odpovede aj o jeho type (hub alebo zbernica).[17]

Odporúčame voliť vzhľadom na výhody centralizovanej komunikácie[18] práve takýto komunikačný vzor, zvlášť keď systémov zapojených do integračného riešenia je príliš veľa alebo je odôvodnený predpoklad, že ich množstvo výrazne stúpne.

Zaradiť integračné systémy

Popri umiestnení existujúcich prvkov informačného systému zaraďujeme do architektúry i tzv. integračné systémy – nové prvky informačného systému, ktoré sme vytvorili alebo nasadili počas riešenia integračného projektu a riešia špecializované potreby, hlavne univerzálne a celopodnikové číselníky a služby, procesné servery, smerovacie a transformačné služby a pod. (pozri aj sekciu Odporúčania). Tieto časti sa môžu správať ako ostatné aplikácie a pristupujeme k nim v ďalších fázach podobne.

Definovať typy portov

Pod portom máme na mysli zabezpečený vstup do systému za účelom zdieľania dát alebo procesov alebo ich využívania. Typ portu špecifikuje konkrétny port, predovšetkým v kontexte informačnej architektúry povolené metódy prístupu k dátam a procesom. Možné typy portov sú napríklad DCOM, príjem súboru, webové služby. Všimnime si, že nepopisujeme presné rozhranie (tzn. napr. v prípade DCOM množiny operácií, v prípade príjmu súboru názvy a formát), pretože to je úloha, ktorú budeme riešiť až v ďalšom kroku. Typ portu zahŕňa aj prípadné adaptéry a fasády.

Teoreticky môže mať jeden účastník ukotvených aj viacero portov s rôznymi typmi, aj keď sa tejto situácii snažíme vyhnúť.

Typy portov, ktoré sú vhodné, závisia jednak od požiadaviek (napr. reakčný čas alebo bezpečnosť) a v neposlednom rade aj od technických možností systémov, na ktorých sú ukotvené (veľmi stará aplikácia nám nemusí umožniť iný spôsob komunikácie ako cez používateľské rozhranie).

Upozorňujeme, že významy portu a typu portu, s ktorými tu pracujú, nie sú úplne totožné s rovnomennými pojmami zo špecifikácie WSDL.

Definovať komunikačné kanály

Po určení spôsobu, akým sa budú systémy pripájať do zvyšku integrovaného informačného systému, definujeme komunikačné kanály, teda určujeme možnosti priamej komunikácie jednotlivých účastníkov. Komunikačný kanál je ukotvený v portoch účastníkov a v zásade môže byť typu bod-bod alebo bod-viacbod (point-to-multipoint).

---

[17] Zhodnotenie a klady a zápory pozri napríklad v (Throwbridge, a iní, 2004).
[18] (Throwbridge, a iní, 2004), (Bakker, 2005), (Kočí, 2007).

Validovať architektúru

Ako sme už viackrát zdôraznili, správna architektúru je z pohľadu výslednej kvality veľmi dôležitá a jej zmena je v neskorších fázach projektu veľmi náročná, preto je nevyhnutné navrhnutú architektúru overiť vzhľadom na zadané požiadavky a všeobecnú funkčnosť vôbec. Validáciu architektúry si vieme predstaviť ako viaczložkový proces pozostávajúci z:

- experimentu (proof of concept), ktorý simuluje integračné riešenie vytvorením produktu s navrhnutou architektúrou v zjednodušených podmienkach (napr. integrovaním testovacích aplikácií alebo integrovaním menšieho počtu jednoduchších aplikácií),
- skúseností, pozitívnych alebo negatívnych, s týmto typom integračnej architektúry alebo podobným, ktorý sme už sprevádzkovali, eventuálne sa pokúšali sprevádzkovať s neúspechom.

3.11.5. Vstupy

- podniková architektúra,
- zoznam systémov,
- interakčná matica podnikových objektov,
- technologické znalosti.

3.11.6. Výstupy

Požiadavky na integračnú architektúru (neopúšťa krok)

Model integračnej architektúry vo forme diagramu a doplnkového textového popisu.

3.11.7. Odporúčania

Globálne komunikačné vzory môžeme v prípade potreby aj vhodne kombinovať, napr. nosná časť komunikácie môže prebiehať na podnikovej zbernici služieb, ale niektoré aplikácie môžu komunikovať aj mimo nej (ako sme to urobili aj my pri návrhu architektúry v integračnom projekte UK; pozri kapitolu 4.7). Ak sa však rozhodneme pre podobné riešenie, majme k dispozícii dobré zdôvodnenie.

Počas integrácie sa stáva, že do portfólia aplikácií informačného systému pribudnú ďalšie, ktoré celkovú architektúru zjednodušia a výrazne uľahčia prácu pri integrácii. V zásade ide o technologickú podporu komunikácie (middleware), ktorej časti môžu predstavovať samostatné systémy v integračnej architektúre, alebo o nové systémy poskytujúce integrované dáta či služby.

Typickí reprezentanti prvej kategórie sú integračné huby a zbernice, procesné servery (stroje), smerovače, transformačné služby a pod. Do druhej kategórie patria systémy špeciálne budované kvôli integrácii s jediným cieľom zdieľať dáta a procesy medzi viacerými aplikáciami.

Procesné servery využijeme pri automatizovanom riadení podnikových procesov alebo ich častí (pozri kapitolu 2.5). V takomto prípade opäť môžeme

využiť oddelenie modelu podnikových procesov od ich implementácie a upravovať len procesno-systémové modely.

V prípade, že pri integrácii dátovým prístupom sú dáta na priame použitie nevhodné, navrhujeme vytvoriť tzv. ideálny dátový model. Takýto dátový model potom inkarnujeme do fyzickej podoby ako databázu vystupujúcu ako nový systém v rámci podnikovej architektúry. Ideálny dátový model nám dobre poslúži, ak:

- samotný dátový model je nevhodný,[19]
- dáta vykazujú anomálne stavy (neudržiavané, nekonzistentné, chýbajúce, ...),
- podnikové objekty sú reprezentované viacerými dátovými zdrojmi – v tomto prípade ide o zlúčenie dát.

Ak jedným z dôvodov sú problematické dáta, prevod existujúcich dát do ideálneho dátového modelu (tzv. migrácia) nám dáva možnosť analýzy a opráv prípadných chýb, či už automatizovane alebo poloautomatizovane.

Proces samotného dátového modelovania ideálneho dátového modelu tu nebudeme rozoberať, nakoľko v tejto oblasti existuje pomerne veľký výber literatúry (pozri napr. (Muller, 1999), (Simsion, a iní, 2005)).

Ideálny dátový model je jeden z integračných systémov, ktorý sme zaradili do procesu na základe spätnej väzby z integračného projektu UK, kde nám jeho použitie veľmi pomohlo (pozri v kapitole 4).

Podobne ako sme už spomínali návrhové vzory v tvorbe softvéru a vzory toku práce, existujú aj rozličné vzory vhodné pre dátové modelovanie. Dajú sa priamočiaro aplikovať mnohé z analytických vzorov (Fowler, 1996), užitočné sú aj katalógy univerzálnych dátových modelov (Silverston, 2001).

Okrem ideálneho dátového modelu sa v informačnom systéme môžu nachádzať aj iné dáta a procesy, ktoré nie sú špecifické pre jednu aplikáciu v zmysle, že by bola „prirodzeným zdrojom“ pre ne alebo že existujú iné zdôvodnenia, prečo by mohli byť vyňaté spod konkrétnej aplikácie (napríklad otázky bezpečnosti, lepšieho zabezpečenia aktuálnosti, ...).

Rozoznávame tzv. univerzálne dáta, teda dáta presahujúce rámec podniku, ktoré sú zvyčajne riadené vonkajšou autoritou. Celopodnikové dáta sú dáta, ktoré svojou podstatou v zmysle vyššie uvedeného presahujú rámec jednej aplikácie, ale riadi ich a využíva primárne len jedna spoločnosť. Veľmi typické pre dáta takýchto druhov sú číselníky.

Univerzálne číselníky môžu byť úplne štandardizované, kedy aj identifikátory položiek definuje externá autorita mimo podniku. Príkladom úplne štandardizovaného univerzálneho číselníka je číselník študijných odborov, ktorý vytvára a spravuje Ústav informácií a prognóz školstva alebo číselník MIME typov, ktorých identifikátory sú reťazce (zoznam MIME typov má pod kontrolou IANA, čo je organizácia patriaca pod ICANN – Internet Corporation For Assigned Names and Numbers, zoznam sa dá nájsť na (IANA, 2006)). Väčšinou sa ale stretávame

[19] Napr. nebol dobre navrhnutý už na začiatku alebo sa postupne zdegeneroval neriadeným vývojom.

s klasickými univerzálnymi číselníkmi, ktorých identifikátory nikto nespravuje – typické sú číselníky miest, obcí, krajov, štátov, poštových smerovacích čísel.

Číselníky majú veľký integračný potenciál, pretože každá netriviálna aplikácia v informačnom systéme ich používa aj rádovo niekoľko desiatok, pričom existujú sady vyskytujúce sa vo viac ako jednej aplikácií.

Typické univerzálne číselníky zahŕňajú:

- územno-správne členenie: poštové smerovacie čísla, obce, mestá, mestské časti, obvody, vojenské obvody, okresy, kraje, samosprávne kraje (vyššie územné celky), štáty, regióny. Môžeme sem zaradiť aj ulice, ale tie sa väčšinou nevyskytujú v číselníkovej forme.[20] Údaje v nich sa riadia príslušnou legislatívou na úseku verejnej správy. Pre zoznamy platné na území SR pozri (MV SR, 2006), číselníky PSČ v SR udržiava Slovenská pošta, š.p. (Slovenská pošta, 2007),
- národnosti,
- pohlavie,
- jazyky: odporúčame použiť štandardizované kódy podľa normy ISO 639,
- klasifikácia zamestnaní,
- právne formy organizácií.

Na Slovensku sú pre organizácie štátnej správy záväzné číselníky, ktoré centrálne spravuje Štatistický úrad SR (Štatistický úrad Slovenskej republiky, 2007). Podobne môžu niektoré úsekové (doménové) číselníky spravovať príslušné asociácie združujúce organizácie podnikajúce na danom úseku. Ďalšou kategóriou externe spravovaných číselníkov sú zoznamy publikované ako štandardy štandardizačných autorít (napr. štandardy STN alebo ISO).

Okrem číselníkov môžeme medzi časté celopodnikové dáta zaradiť dáta o partneroch (zamestnanci, klienti, ...) a záväzkoch, organizačnú hierarchiu, hierarchiu produktov a pod.

Podobne ako pojmy celopodnikové a univerzálne dáta definujeme aj procesy. Ich nasadenie v podobe služieb označujeme ako

- celopodnikové služby, ak ide o druh služieb, ktorých význam presahuje rámec jednej entity informačného systému, ale sú nasadené jednou spoločnosťou a ich konzumenti pochádzajú primárne tiež z tejto spoločnosti,
- univerzálne služby, teda služby, ktorých význam presahuje rámec podniku, sú nasadené a spravované mimo podniku a ich spotrebiteľmi sú aj iné podniky.

Konkrétnym príklad celopodnikovej služby by mohli byť služby centrálnej autentifikácie, autorizácie alebo služby pre výpočty daňového zaťaženia, poštovného a pod.

[20] Agendové aplikácie zvyčajne umožňujú priame vpísanie ulice používateľom.

Univerzálna služba môže byť univerzálna v rámci priemyslového odvetvia (napr. objednávky u dodávateľa agrochemikálií) alebo univerzálna ponad všetky odvetvia (príkazy na prepravu špedičných spoločností, zasielanie faktúr poštovej službe a pod.).

Univerzálne alebo celopodnikové dáta či služby môžu byť vystavené integráciou ako samostatné integračné systémy, v prípade dát zvyčajne ako samostatné databázy s prípadnými rozhraniami väčšinou v podobe služieb, v prípade procesov ako aplikačné servery, v ktorých je nasadená ich implementácia.

3.12. Komunikačný návrh

Posledným návrhárskym krokom je zadefinovanie konkrétnych prepojení medzi aplikáciami, určenie ktoré aplikácie (entity) budú zdieľať aké dáta a procesy a ktoré aplikácie ich naopak budú využívať a za akých podmienok.

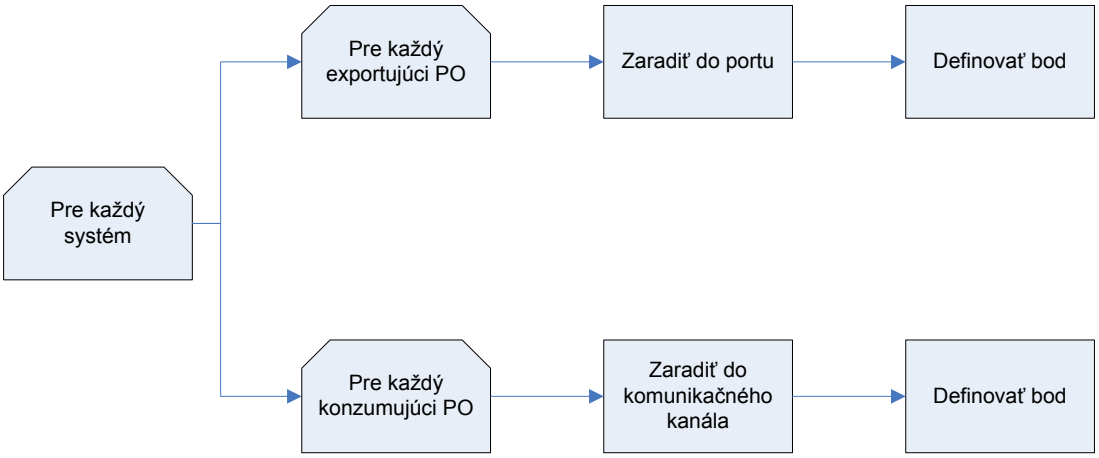
3.12.1. Účel a odôvodnenie

Návrh a popis konkrétnych bodov zdieľania dát a procesov.

3.12.2. Zúčastnené roly

- architekt,
- podnikový architekt,
- návrhár.

3.12.3. Tok práce



3.12.4. Aktivity

Hoci v diagrame toku práce je naznačený sekvenčný postup pre každý systém, prakticky to nie je úplne možné – ak to bude potrebné, budeme pracovať aj s viacerými systémami naraz, takejto situácii sa podľa nás nedá vyhnúť, zvlášť v prípadoch, keď zabezpečujeme obe strany komunikácie.

Odporúčame začať návrh komunikácie podľa podnikových procesov, teda navrhovať komunikačné body v súlade s aktuálnymi procesno-systémovými

modelmi. V prípade, že niektoré procesy riadime automatizovane, súčasťou komunikačného návrhu je aj tok riadenia procesu.

Zaradiť do portu (exportujúci PO)

Podnikový objekt zaradíme do niektorého z definovaných portov. Vo väčšine prípadov má exportujúci systém iba jeden port.

Definovať bod (exportujúci PO)

Bodom v tomto kontexte rozumieme miesto v porte, cez ktoré bude aplikácia podnikový objekt zdieľať. Znamená to nadefinovať v závislosti na použitej metóde integrácie (type portu) všetky potrebné parametre. V prípade potreby sa udávajú aj transformácie, ktoré je vhodné urobiť pred vyzdieľaním objektu smerom von.

Pre dátovú integráciu to môže znamenať:

- z ktorých dátových objektov (tabuliek, stĺpcov, pohľadov a pod.) je podnikový objekt dostupný vonkajšiemu svetu,
- aká je dostupnosť dát a aká je miera ich aktuálnosti,
- ktorí koncoví účastníci sú oprávnení a v akej miere (ktoré atribúty) získať (bezpečnosť),
- na akých fyzických adresách (UNC názov servera, IP adresy, ...) sú dáta dostupné,
- pod akými bezpečnostnými dokladmi (meno/heslo, certifikát, ...) sú dáta dostupné.

Zaradiť do komunikačného kanálu (konzumujúci PO)

Navrhujeme, cez ktorý komunikačný kanál bude zdieľaný podnikový objekt importovaný zo zdrojovej aplikácie.

Definovať bod (konzumujúci PO)

Definícia zahŕňa prípadné vyrovnanie sa so zmenami medzi zdrojom a potrebami cieľového účastníka (transformácie).

3.12.5. Vstupy

- integračná a podniková architektúra,
- model podnikových procesov,
- zoznam podnikových objektov,
- artefakty charakteristík systémov (podľa prístupu, procesno-systémové modely, dátové modely a pod.).

3.12.6. Výstupy

Náš proces nezávisí na konkrétnych technologických aspektoch, ako sme uviedli už v úvode. Formát výstupov tejto fázy však záleží na zvolenom spôsobe komunikácie (typoch portov) a čiastočne i na komunikačnom vzore. Uvedieme tu

preto príklady, ako by takéto výstupy mohli vyzerať pri integrácii na dátovej úrovni prostredníctvom dávok ETL v databázových systémoch a v prípade integrácie budovaním ESB.[21]

Integrácia na dátovej úrovni/ETL

Názov cieľovej tabuľky

Cieľový stĺpec	Zdroj	Poznámka
C1	*autogenerate*	PK
C2	S1.T1.C1	
C3	S1.T1.C2 + " - " + S1.T1.C3	
C4	S2.T2.C4	S2.T2 spojiť cez C1/S2.T2.C1
...	...	...
...	...	...

Tabuľka 4 Šablóna pre návrh dátovej integrácie

Štandardne zápis pozostáva z definičných popisov pre každú jednu cieľovú tabuľku. Z praxe máme poznatok, že väčšina dátových importov prebieha zo zdrojových tabuliek pre všetky riadky, preto stačí uvádzať v definícii, z ktorého stĺpca/stĺpcov ktorej tabuľky/tabuliek dáta pochádzajú a ako sa prípadne upravujú. Jednoduchšie transformácie zapisujeme priamo do stĺpca *Zdroj* vo forme výrazu, napr. pri zreťazení hodnôt z dvoch stĺpcov cez pomlčku `S1.T1.C2 + " - " + S1.T1.C3`. Zložitejšie transformácie píšeme do poľa *Poznámka*. V prípade, že dáta pochádzajú z viacerých tabuliek, musíme uviesť, vzhľadom na ktorý atribút ich treba spojiť. V prípade, že väčšina dát pochádza z jednej tabuľky (označme ju referenčná), túto informáciu môže zapísať lokálne pri zmienke ďalšej tabuľky v poli Poznámka v zmysle „spojiť cez `A/B`“, kde `A` je odkaz primárny kľúč referenčnej tabuľky a `B` je odkaz na kľúč druhej tabuľky (pozri príklad vyššie a ilustráciu na integračnom projekte UK v ďalšej kapitole).

Ak zo zdrojových tabuliek nevyberáme všetky riadky, uvedieme na začiatku tabuľky selekčnú podmienku, ktorej platnosť vzhľadom na záznamy zdrojových tabuliek určí, ktoré riadky sa majú spracovať a následne importovať.

Do tej istej cieľovej tabuľky môžu pochopiteľne prichádzať dáta aj z rôznych zdrojových tabuliek, prípadne ich kombinácií a selekčných podmienok. V takom prípade uvedieme importy príslušný počet krát.

Integrácia na úrovni služieb/ESB

Pri integrácii službami definujeme rozhrania služieb, v ktorých uvádzame názvy operácií, v krátkosti popis ich účelu a identifikátory správ, ktoré sa posielajú pri volaní operácie. Typicky to budú správy typu požiadavka a odpoveď, v prípade diffgramov (správ so zmenenou podmnožinou dát) iba odpoveď (pozri aj vzory komunikácie napr. v (Hohpe, a iní, 2003)).

Názov operácie	Popis	Správy
názov	popis operácie	správy, ktoré sa pri vyvolaní

[21] Dôvod je jednoduchý – tieto prístupy používame v integračnom projekte UK.

		vymieňajú

Tabuľka 5 Šablóna pre popis rozhrania služby

Správy potom môže špecifikovať prostredníctvom nasledovných šablón:

Id:		**Typ:**	Request
Názov:			
Popis:			
Parametre:			
Telo:			
Odpoveď:			
Výnimky:			

Tabuľka 6 Šablóna pre špecifikáciu správy typu požiadavka (request)

Id:		**Typ:**	
Názov:			
Popis:			
Telo:			

Tabuľka 7 Šablóna pre popis správy typu odpoveď (response)

3.13. Implementácia, verifikácia a validácia

Komunikačný návrh sa ďalej realizuje jeho implementáciou, teda vykonaním činností vedúcich k naplneniu navrhnutej integrácie. V modeloch procesu vývoja softvéru badať v súčasnosti popularitu rôznych metodík riadených testami. Myslíme si, že podobná myšlienka sa dá aplikovať aj pri integrácii aplikácií, a to pomerne priamočiaro. Súčasťou tohto kroku je teda aj verifikácia a validácia riešenia.[22]

3.13.1. Účel a odôvodnenie

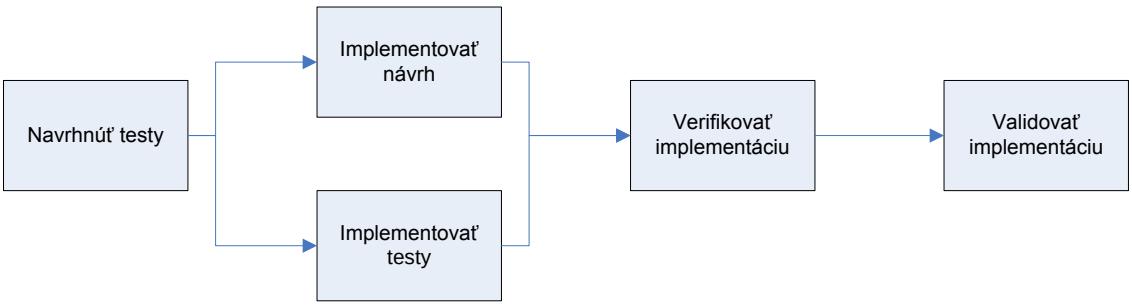
Realizovať na základe návrhu integráciu informačného systému a overiť jeho výslednú kvalitu.

3.13.2. Zúčastnené roly

- návrhár (komunikácia s implementátormi ohľadne návrhu, návrh testov),
- kvalitár (návrh testov, testovanie),
- implementátor.

[22] Verifikácia (overenie kvality produktu voči zozbieraným požiadavkám) a validácia (overenie kvality vzhľadom na reálne potreby zákazníka) sú pojmy prebraté z procesov vývoja softvéru, pozri (Sommerville, 2004).

3.13.3. Tok práce



3.13.4. Aktivity

Navrhnúť testy

Návrhár v spolupráci s kvalitárom popíše testy. Konkrétna forma závisí od zvolenej metodiky, prístupu a technológie integrácie. Cieľom testov je overenie, či implementácia je skutočne korektnou realizáciou komunikačného návrhu. Napríklad v prípade integrácie na úrovni služieb by sme navrhovali testy, ktoré vyvolajú príslušné služby s testovacími vstupmi a porovnajú ich výstupy so správnym výsledkom. V dátovej integrácii prostredníctvom ETL môžeme pripraviť testovaciu zdrojovú databázu, spustiť ETL a opäť porovnať výsledok. Automatizované riadenie procesu by sme mohli otestovať navrhnutím testovacích vstupov, vyvolaním procesu a kontrolou výstupov, pričom by sme mali testovať aj volanie jednotlivých aktivít procesu.

Pripomíname, že testy by mali byť navrhnuté pred implementáciou.

Implementovať návrh

Samotná implementácia návrhu je, pochopiteľne, silne závislá od prístupu a technológie a nedá sa o nej hovoriť vo všeobecnosti.

Implementovať testy

V prípade testov, ktoré sú vykonateľné automatizovane, sa tieto implementujú podľa návrhu. Opäť konkrétny postup je závislý na použitej technológii.

Verifikovať implementáciu

Po dokončení implementácie (alebo jej časti) je potrebné overiť korektnosť otestovaním podľa navrhnutých testov. V prípade nezhody je nutné implementáciu opraviť alebo ak príčina nezhody spočíva v nemožnosti implementácie či chybe v návrhu, musia sa urobiť príslušné zmeny v návrhových artefaktoch (nemali by sme zabudnúť aj upraviť testy).

Validovať implementáciu

Úspešnosť verifikačnej aktivity ešte neznamená zaručený úspech, pretože verifikácia overuje implementáciu voči navrhnutým testom, ktoré jednak nemusia testovať vyčerpávajúco všetky požadované vlastnosti (a zvyčajne ani netestujú) alebo môže byť chyba v nich samotných, prípadne chyba prenesená z vyšších artefaktov. Validácia zahŕňa overenie kvality implementácie zákazníkom.

3.13.5. Vstupy

- integračná architektúra,
- komunikačný návrh.

3.13.6. Výstupy

- návrh testov,
- výsledok implementácie (integrovaný systém).

3.13.7. Odporúčania

Pri špecifikácii komunikačného návrhu odporúčame oddeliť „logický“ a „fyzický“ pohľad. Vo fyzickom pohľade definujeme konkrétne vlastnosti závislé od nasadenia systémov, teda IP adresy, názvy severov, používateľské účty a pod. V logickom pohľade uvedieme potom všetko, čo od týchto vlastností nezávisí, teda v prípade dátovej integrácie názvy tabuliek, stĺpcov a ich transformácie. V prípadovej štúdii uvedenej v kapitole 4 v logickom návrhu používame symbolické názvy zdrojov (ST pre databázy systému Študent, IL pre informačné listy predmetov a pod.). Konkrétne mapovanie (fyzický pohľad) bude potom napísané v samostatnom dokumente.

3.14. Riadenie rizík v integračnom projekte

Rizikom nazývame v projektovom riadení udalosť s negatívnymi dôsledkami na projekt s pravdepodobnosťou výskytu p, $0 < p < 1$.

Riadenie rizík je systematický prístup k známym aj neznámym rizikám tak, aby ich vplyv na projekt bol čo najmenší.

Riadenie rizík je teda proces, ktorý väčšinou pozostáva z identifikácie rizík, ich analýzy a prípravy na ne a dodržiavania akcií na minimalizovanie ich vplyvu pri výskyte, pričom je dôležité, aby tento proces prebiehal sústavne počas celého projektu (postupne vznikajú nové riziká, menia sa vlastnosti nájdených rizík). Akcie na minimalizáciu rizík väčšinou spadajú do kategórií akceptovania rizika, vyhnutiu sa mu, monitorovania a kontingenčných plánov, transferu rizika a mitigácie rizika (Verzuh, 2005). Keďže o riadení rizík existuje veľa literatúry, nebudeme sa ním už podrobnejšie zaoberať (pomerne detailný rozbor o riadení rizík vo veľkých projektoch môže čitateľ nájsť napr. v (Cooper, a iní, 2004)).

(Sommerville, 2004) delí riziká na projektové (ovplyvňujúce projektové zdroje), produktové (týkajúce sa kvality produktu) a podnikové (majúce vplyv na dodávateľskú organizáciu).

V nasledujúcej tabuľke uvádzame niekoľko rizík, ktoré sme identifikovali v kontexte projektov integrácie aplikácií. Nevymenovávame klasické projektové a podnikové riziká totožné s rizikami pri vývoji softvéru (napr. odchod kľúčových zamestnancov, príliš optimistický časový plán a pod.).

Riziko	Kategória
podnikové procesy neodrážajú skutočnosť	produktové

podnikové procesy nie sú efektívne	produktové
middleware nespĺňa požiadavky naň kladené	produktové
vývoj nového systému počas integrácie	podnikové
neefektívna integračná architektúra	produktové
nezdokumentované dátové modely	produktové
veľmi staré aplikácie (legacy)	produktové
aplikácie bežia na starom hadvéri/operačných systémoch	produktové
vysoká previazanosť aplikácií	produktové
integrácia snímaním obrazoviek	produktové
integračná architektúra bod-bod	produktové
použitie neštandardných protokolov	produktové
málo skúseností s daným typom prístupu k integrácii alebo s middlewarom	projektové
prítomnosť dlhotrvajúcich podnikových transakcií	produktové
propagácia zmien v reálnom čase	produktové
neudržiavané dáta (možno bude potrebná analýza dát, očistenie a migrácia)	produktové, projektové
výrobca middlewaru alebo iného integračného systému už neexistuje	projektové

Tabuľka 8 Riziká pri integračných projektoch

4. Prípadová štúdia integračného projektu UK

V tejto kapitole popíšeme naše skúsenosti a postup prác pri realizácii integračného projektu na UK, čo slúži jednak ako vstup pre tvorbu procesu a jednak ako jeho praktické overenie.

Kvôli rozsahu práce, žiaľ, nemôžeme uviesť v plnom znení všetky artefakty, ktoré sme vyrobili pri práci na integračnom projekte (podnikové objekty a procesy, popisy systémov, diagramy, modely a pod.), preto sa pokúsime z nich vybrať aspoň tie reprezentatívne.

Proces budeme ilustrovať na integrácii študijnej agendy s webovou prezentáciou a anketovým systémom. Keďže pre webovú prezentáciu ako aj pre anketový systém si postačíme bez modelu podnikových procesov, ktorý by bol príliš „ťažký“, tento príklad je zároveň príkladom, kedy analyzujeme potreby integrácie z iných artefaktov. Kvôli názornosti však na konci uvádzame aj príklady modelu podnikových procesov, ktoré plánujeme použiť v blízkej budúcnosti, pričom načrtneme možnosti integrácie.

4.1. Východiskový stav

Univerzita Komenského začala automatizovať svoje podnikové procesy podobne ako väčšina iných podnikov a organizácii. Postupne sa vytvárali samostatné programy spracovávajúce časť agendy jednotlivých oddelení. V roku 1991 sa začala budovať na Univerzite Komenského počítačová sieť, čo dalo predpoklad k vzniku centrálneho informačného systému, známeho lokálne ako Informačný a komunikačný systém UK (IKS UK) (Devínsky, a iní, 1998). Vzhľadom na prudký rozvoj informačných technológií, ktorý nastal aj na území ponovembrovej republiky, bolo čoraz jasnejšie, že súčasný stav udržateľnosti IKS UK je nedostačujúci a v roku 1997 sa rozhodnutím rektora začalo s projektom *Integrovaného* informačného a komunikačného systému UK (IIKS UK), ktorého prvými krokmi bolo vypracovanie úvodnej štúdie a tzv. Strategickej štúdie IIKS UK (Devínsky, a iní, 1998). Tento materiál sa na vyše 250 stranách venuje úvodnému rozpracovaniu činností, ktoré na UK prebiehajú, technologickej infraštruktúre a riadeniu a plánovaniu ďalších fáz projektu IIKS UK. Netreba pripomínať, že desať rokov je v oblasti informačných technológií pomaly epochou a tak mnohé z vecí popísané v nej už nie sú aktuálne.

Za ten čas sa v rámci projektu IIKS vybudovali viaceré systémy, medzi nimi virtuálna knižnica, moderná webová prezentácia UK a jednotlivých fakúlt, finančný informačný systém, systém automatickej identifikácie osôb, systém pre telefonovanie prostredníctvom počítačovej siete a viaceré služby v oblasti infraštruktúry.

Každá moderná univerzita či vysoká škola musí v súčasnej dobe viesť údaje a spracovávať ich v počítačovej forme. Najvýznamnejším prvkom jej informačného systému je vždy systém na vedenie študijnej agendy, teda tá časť informačného systému, v ktorej sú evidované a spracúvané údaje o študentoch, učiteľoch, predmetoch a ich vzájomných vzťahoch – ktorí študenti majú zapísané ktoré predmety a ktorí učitelia ich vyučujú, aké hodnotenia študenti získali a pod. Tieto dáta spolu s procesmi prebiehajúcimi nad nimi sa vyskytujú vo veľkom počte

aplikácii na univerzite, preto je logické, že táto oblasť má najvyššiu prioritu a začali sme s ňou aj my.

4.2. Štúdia realizovateľnosti

Vzhľadom na skutočnosť, že integračný projekt UK sa realizuje aspoň zatiaľ výlučne vlastnými silami, integračný tím aj zadávateľ sú z jednej organizácie, preto postačuje jedna štúdia realizovateľnosti, ktorá však dáva odpovede na otázky oboch strán, zadávateľa i dodávateľa.

Za štúdiu realizovateľnosti môžeme približne pokladať práce, ktoré boli zosumarizované v (Devínsky, a iní, 1998).

4.3. Kandidátske systémy

Univerzita Komenského prevádzkuje viacero rozličných systémov v rámci svojho informačného systému. Ako sme spomenuli vyššie, v integračnom projekte UK sme sa rozhodli zamerať na všetky aspekty sústredené okolo hlavného podnikového procesu UK, teda študijnej agendy. Primárnym systémom pre integráciu je systém Študent. Následne sme sa rozhodli zaradiť do tejto ukážky webovú prezentáciu a anketový systém.

Výrez zo zoznamu kandidátskych systémov je uvedený v prílohe A.

Úroveň integrácie pri webovej prezentácii je veľmi nízka. Integrácia prebieha zvyčajne portálovým spôsobom, pri ktorej sa vyvinú nové aplikácie s dizajnom prezentačnej časti – takto funguje napríklad vyhľadávanie osôb na UK. Informácie o študijných programoch a predmetoch väčšinou nie sú integrované vôbec, ide o statické stránky, pri tvorbe ktorých v najlepšom pomáhajú exporty do HTML.

Do anketového systému sa pred začatím ankety jednorázovo ručne naimportujú dáta o študentoch a predmetoch. Dáta pochádzajú priamo zo systému Študent, v ktorom je úroveň ich čistoty rôzna. Po importe sa musia ručne doplňovať mnohé údaje, napríklad priradiť prihlasovacie údaje učiteľa k predmetom, ktoré vyučuje.

4.4. Charakteristiky systémov (používateľské popisy)

Ako sme spomenuli, webová prezentácia a anketový systém partia medzi časti informačného systému, ktorý nezasahuje „veľký podnikový proces“.

Pri analýze integračných potrieb týchto systémov vychádzame predovšetkým z používateľských popisov systémov, ktoré sme urobili na základe stretnutí s ich prevádzkovateľmi a osobnom vyskúšaní.

Reprezentatívne používateľské popisy (opäť kvôli obmedzenému priestoru) uvádzame v prílohe B.

4.5. Podnikové objekty

Na základe používateľských popisov systémov môžeme ľahko identifikovať podnikové objekty, s ktorými pracujú tie systémy, ktoré sa nezapájajú do „veľkých“ podnikových procesov alebo podnikové procesy, ktoré sa ich týkajú, sú vyslovene

lokálne. Typicky ide o systémy na zobrazovanie informácií (web) alebo systémy s veľmi úzkou profiláciou (anketa).

Keďže do integrácie na UK má byť zapojený aj web, aj anketový systém, zaradili sme do práce ukážky používateľských popisov systémov práve pre tieto dva systémy (príloha B). Z nich známym procesom hľadania podnikových objektov vyťažíme tieto objekty:

Názov	Popis	Identifikátory	Ďalšie vlastnosti	Poznámky	Zdroj	Konz.
študent		(osoba)	(osoba)			Á
učiteľ	hodnotené osoby	(osoba)	(osoba)	do ankety môžu študenti dopisovať učiteľov		Á
predmet	hodnotené objekty	kód predmetu, názov	vyučujúci, ideálne po formách vyúčby, študijný odbor a program			Á
vyučujúci na predmete		predmet, učiteľ		študent môže priradiť predmetu vyučujúceho		Á

Tabuľka 9 Podnikové objekty pre webovú prezentáciu

Všimnime si, že ako vlastnosti objektov uvádzame len tie, ktoré cieľový systém dokáže využiť (napr. pre anketu nie je zaujímavá osnova predmetu).

Podnikové objekty pre web:

Názov	Popis	Identifi-kátory	Ďalšie vlastnosti	Poznám-ky	Zdroj	Konz.
študijný odbor		kód	názov			Á
študijný program		kód	študijný odbor			Á
skupina predmetov	množina predmetov zoradená kvôli kategorizácii	rodičovská skupina predmetov, predmety	povinnosť voľby (pravidlá úspešného absolvovania tejto skupiny)			Á
predmet		kód, názov	počet kreditov, jazyk, osnova, forma			Á

			a rozsah výučby, literatúra, vyučujúci			
študent		(osoba)	(osoba)	pre web zoznam študujú cich podľa progra mov		Á
zapísané predmety	predmety, ktoré si študent zapísal	študent, predmety	semester, známky			Á

Tabuľka 10 Podnikové objekty pre anketový systém

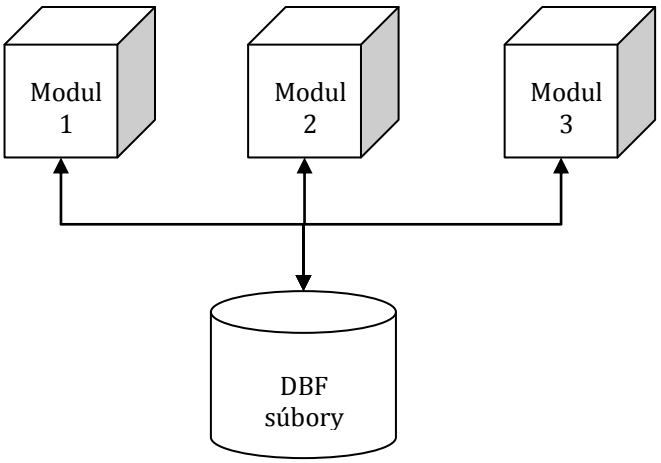
4.6. Charakteristiky systémov

V uvedených príkladoch požadovaných podnikových objektov vidíme, že sa týkajú predovšetkým študijnej agendy. Na fakultách UK je hlavným aplikačným systémom pre podporu študijnej agendy systém Študent, preto prvým krokom bolo hľadať zdroje týchto objektov práve tu.

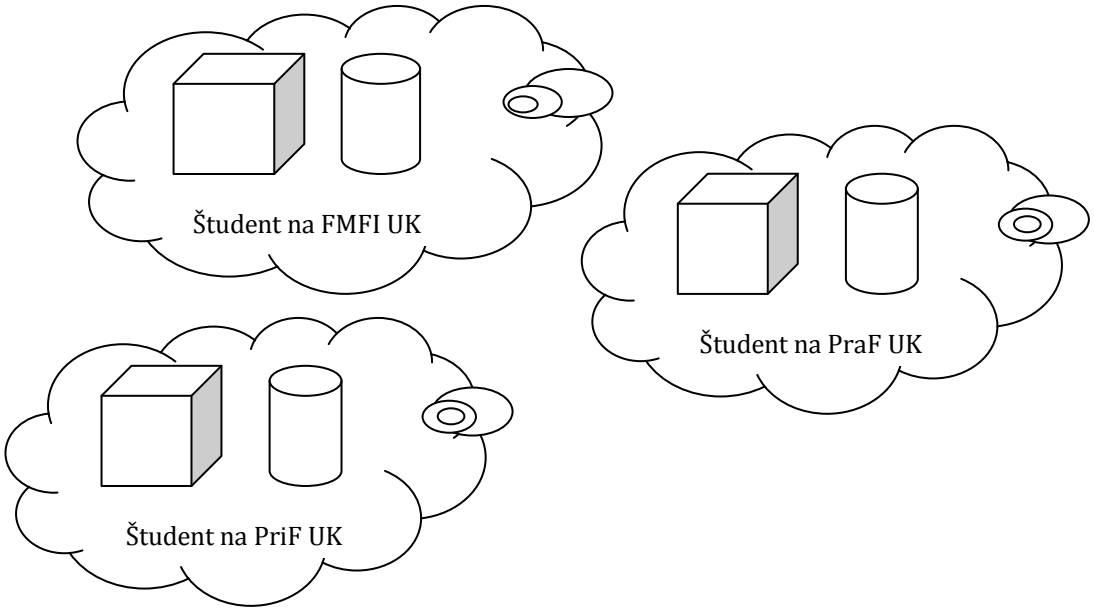
Na začiatku prebehlo niekoľko stretnutí s hlavným vývojárom Študenta, Ing. Jánom Petríkom, na ktorom sme sa venovali funkčnosti systému, jeho možnostiach integrácie a súčasnej prevádzke. Nasledujúce odstavce zhŕňajú podstatu úvodných stretnutí.

V súčasnosti sa všetkých fakultách UK používa na vedenie študijnej agendy systém Študent. Študent je aplikácia databázového systému Clipper s logikou písanou v rovnomennom programovacom jazyku Clipper. Práce na ňom začali začiatkom 90. rokov výlučne pre FMFI UK, ale neskôr sa jeho používanie rozšírilo na celú univerzitu. Univerzita Komenského bola čo sa týka počítačového a softvérového vybavenia vždy medzi poprednými univerzitami na Slovensku a systém vedenia študijnej agendy nebol výnimkou, čo paradoxne (ale na druhej strane prirodzene) viedlo k situácii, že dnes má v prevádzke systém, ktorého technologické riešenie i platforma, na ktorej je postavený, sú poplatné dobe svojho vzniku.

Čo sa týka architektúry, Študent ako taký pozostáva z viacerých modulov. Každý modul má zodpovednosť za inú časť študijnej agendy. Najpoužívanejšie moduly sú Evidencia študijných programov, Evidencia študentov, Zápisy, Štipendiá, Pôžičky a Podporné služby. Posledne menovaný modul generuje rozličné štatistiky a exporty a importy, väčšinou pripravované ad-hoc na požiadanie správcu iného systému, ktorý by rád zdieľal údaje so Študentom. Moduly sú fakticky samostatné aplikácie písané nad desktopovým databázovým systémom v jazyku Clipper a bežia nad spoločnou (logickou) databázou pozostávajúcou z viacerých fyzických súborov vo formáte DBF.



Obrázok 4-1 Architektúra systému Študent na najvyššej úrovni



Obrázok 4-2 Systémy Študent ako neintegrované ostrovčeky

Systém Študent je prevádzkovaný samostatne na každej fakulte zvlášť s vlastnou databázou, väčšinou na príslušnom študijnom oddelení. Jednotlivé nasadené systémy medzi sebou nekomunikujú. Situácii neprospieva ani fakt, že fakulty prevádzkujú rozličné kombinácie modulov a to i v rôznych verziách.

V súčasnosti prebiehajú na Univerzite Komenského prípravy nahradenia existujúceho systému Študent novou verziou (pracovný názov Študent II), ktorá má byť kúpená ako hotové riešenie od externého dodávateľa upravené pre podmienky UK. V súčasnej dobe existuje predbežné zadanie a bolo vyhlásené

výberové konanie. Celý proces potrvá podľa nás najmenej rok a nový systém nebude nasadený skôr ako v septembri roku 2008.

Predchádzajúcou analýzou potrieb integrácie sme zistili, že viaceré aplikácie by potrebovali údaje zo systému Študent. Ak máme jasno v tom, ktoré dáta budeme potrebovať, ďalším logickým krokom je sa pozrieť na tieto dáta detailne (charakteristiky systémov).

4.6.1. Analýza dát systému Študent

Z toho, že Študent je dvojvrstvová aplikácia postavená na dátovými súbormi a z povahy prostredia Clipper, v ktorom je naprogramovaný, vyplýva, že jediný rozumný spôsob sprístupnenia jeho údajov iným aplikáciám je integráciou na dátovej úrovni.

Základný artefakt pri charakteristike databázového systému je jeho dátový model, ktorého popis môže čitateľ nájsť v prílohe

Pri zostavovaní dátového modelu sme si zároveň všímali aj samotné dáta, hlavne v akom sú stave a aký majú vzťah k potrebným podnikovým objektom (odlišnosti, neúplnosti, nekonzistencie a pod.), teda robíme de facto analýzu dát.

Z dátového modelu sme postupom popísanom v 3. kapitole našli tieto podnikové objekty, ktoré poskytuje systém Študent:

Názov	Popis	Identifikátory	Ďalšie vlastnosti	Poznámky
študijný odbor		kód	názov	v systéme Študent sa nenachádza univerzálny „rezortný“ kód
študijný program	množina predmetov a pravidiel, ktoré tvoria jednu „implementáciu“ študijného odboru	kód, názov	študijný odbor, pravidlá, stupeň a forma štúdia, titul, štandardná dĺžka, profil absolventa	
predmet		kód, názov	anglický názov	
variant predmetu	inštancia predmetu s variabilnými atribútmi	predmet	počet kreditov, forma a rozsah výučby, odp. semester, hodnotenie, spôsob hodnotenia,	

			jazyk výučby, stav predmetu	
nasadenie predmetu	inštancia variantu v študijnom programe	variant predmetu, skupina predmetov, v ktorej je nasadený	odporúčaný semester zápisu, prerekvizity	
skupina predmetov	množiny skupín predmetov a predmetov	rodičovská skupina predmetov alebo študijný program, predmety, názov	voliteľnosť	
etapa študijného programu	časová jednotka študijného programu, skoro vždy jeden semester	študijný program, poradové číslo	podmienky na absolvovanie etapy	

Tabuľka 11 Podnikové objekty systémyu Študent

Po vykonaní analýzy dát modelu systému Študent sme našli niekoľko nedostatkov, ktoré sú dôsledkami podľa nás jednak obmedzení prostredia Clipper, zložitej udržiavateľnosti programov napísaných v ňom a nedostatkov prevádzkovateľov a používateľov systémov na jednotlivých fakultách. Toto bol jeden z motivujúcich faktorov, prečo sme sa rozhodli korigovať našu cestu v integrácii študijnej agendy čistým dátovým prístupom (ďalšie spomenieme o chvíľu). Namiesto toho sme navrhli tzv. ideálny dátový model (v ďalšom texte pod kódovým menom STIF, viac o ňom v ďalšej podkapitole). Ďalej uvádzame tie najpodstatnejšie nedostatky dát systému Študent a ako sme sa s nim vyrovnali v novom modeli:

Nenormalizované dáta. Niektoré dáta v pôvodnom systéme nevyhovujú ani prvej normálnej forme.[23] Príkladom porušenia tohto princípu sú napr. atribúty `ucitl` a `forov` tabuľky `RLP` systému Študent, ktorých hodnoty tvoria zoznam učiteľov, resp. formy výučby predmetu:

uipmt	stats	semes	poctv	forov	phodn	hkred	gespr	ucitl
410	a	Z	13	**K4**	50/50	5	KAGDM	**JdcO**
4n0	a	L	13	**K4**	50/50	5	KAI	**Ag**
620	a	Z	13	**K3**	50/50	4	KJFB	**PGFZ**
6Z0	a	L	13	**K2**	50/50	3	KZVI	**gbfl**
3t2	a	Z	13	**P3 C1**	20/80	6	KMANM	**Ji**
FI0	a	Z	13	**P2**	0/100	3	KJFB	**BFDHHy**
6Y0	a	L	13	**P1 C2**	60/40	4	KAI	**Rq**
4r0	a	Z	13	**P2 C2 L2**	60/40	6	KZVI	**Ei**

[23] Hovoríme, že relačná schéma je v **prvej normálnej form**e (1NF), ak hodnoty všetkých domén sú atomické; inými slovami, v atribútoch sa vyskytujú ďalej nedeliteľné hodnoty, pozri napr. (Gardarin, a iní, 1989) alebo (Silbershatz, a iní, 2002)

Napríklad hodnota P2 C2 znamená, že daný predmet je nasadený s dvojhodinovou prednáškou a dvojhodinovými cvičeniami. Hodnota `BFDHHy` v stĺpci `uctil` vyjadruje, že tento predmet vyučujú traja učitelia s kódmi `BF`, `DH` a `Hy`.

Tabuľky, ktoré nie sú v prvej normálnej forme, robia problémy pri všetkých základných databázových operáciách (výbere, úprave a vkladaní), keďže nie je daný jednoznačný spôsob zápisu hodnôt.

Tento problém sme odstránili v STIFe patričnými rozkladmi (pozri relácie `TeachersVariants` a `FormExtensVariants`).

Redundancia dát. Na viacerých miestach sa v systéme Študent vyskytujú redundantné (nadbytočné) údaje; príkladom je tabuľka `RLP` a jej atribút `phodn`, hodnoty ktorého vyjadrujú podiel záverečnej skúšky k priebežnému hodnoteniu na celkovom hodnotení vo formáte p/s, kde p je percentuálny podiel priebežného hodnotenia a s je podiel záverečnej skúšky. Keďže žiadne iné hodnotenia nie sú, implicitné ohraničenie hodnôt je, že ich súčet musí byť číslo 100, z čoho vyplýva, že jeden podiel je uvedený zbytočne a dá sa z druhého jednoducho dopočítať.

uimpt	stats	semes	poctv	forov	phod	hkred	gespr	ucitl
Hc0	a	L	13	L4	**100/0**	5	KEF	JL
9Q1	a	L	13	D6	**100/0**	2	KEF	
Gb2	a	Z	13	P3	**20/80**	3	KEF	Kd
C=1	a	Z	13	P1 C1	**50/50**	2	KAFZM	es
L20	a	L	13	P3 S2	**5/95**	6	KTFDF	GF
7s0	a	Z	13	K2	**35/65**	3	KAFZM	WU

Tento príklad predstavuje veľmi špecifický prípad výskytu nadbytočnej informácie v nenormalizovanej tabuľke, keďže sa jedná o jediný atribút. V STIFe udržujeme iba informáciu o podieli záverečnej skúšky.

Výskyt na jednom mieste. Ďalším príkladom, kde môžu vznikať v databáze anomálie pri vkladaní, mazaní či úprave, sú stĺpce uchovávajúce zo svojej podstaty konečný počet enumerovaných, väčšinou textových hodnôt. Tento druh nadbytočnosti nie je dôsledkom porušenia tradičných normálnych foriem relácií. Riešením je hodnoty uložiť do samostatnej tabuľky, každej priradiť kód a uvádzať namiesto hodnôt len tieto kódy ako cudzie kľúče. Vniknutá tabuľka sa zvykne označovať ako *číselník*.[24] Príklad zo systému Študent je tabuľka `SPG` a stĺpec `titul`:

idspg	nazov	sodb1	sodb2	stups	titul	sysst
AIN	aplikovaná informatika	251100		1	**Bc.**	kp
BMF	biomedicínska fyzika	116006	1111	1	**Bc.**	kp
D	doktorandské			4	**PhD.**	ka
D/x	doktorandské			4	**PhD.**	ka
EFM	ekonomická a finančná matematika	111304		1	**Bc.**	kp
FY	fyzika	116000		2	**Mgr.**	ka

[24] zriedkavejšie v slovenčine aj *kódovník*, v angličtine *lookup table* alebo *code table.*

Uvedený nečíselníkový spôsob má niekoľko zrejmých nevýhod:

- ak v systéme neexistuje záznam s niektorou hodnotou, nemáme ju v systéme vôbec (v našom príklade neexistuje žiadny program, vyštudovaním ktorého sa študentovi prizná titul Ing. – preto v systéme takýto titul ani nie je a nemôžeme ho ponúknuť napríklad pri zakladaní nového programu v kombinovanom boxe),
- preklepy a rôzne spôsoby zápisu v záznamoch znemožňujú jednoduché dopytovanie (napríklad ak by existoval záznam študijného programu s titulom Mgr (bez bodky), nevieme jednoduchým dotazom nájsť všetky magisterské programy),
- pri zmene hodnôt je nutné upraviť všetky záznamy s pôvodnou hodnotou (a stále hrozí riziko, že ich nenájdeme všetky – predchádzajúci bod),
- veľmi ťažká a neprehľadná lokalizácia (ak by sme chceli uchovať v systéme viacero jazykových verzií, týmto spôsobom by sme museli tabuľky rozširovať ďalšími stĺpcami – nezriedka majú kľúčové tabuľky aj 5 a viac stĺpcov z enumerovanej množiny, pri 5 jazykových mutáciách by musela mať 20 stĺpcov navyše).

Pri použití číselníkových tabuliek veľmi často vo väčších databázach ich vzniká pomerne veľa. Niekedy môžeme vidieť, že databázový návrhár sa s takouto situáciou vyrovná vytvorením jednej veľkej tabuľky so schémou, ktorá vyzerá podobne ako (**id_číselníka, kód**, hodnota). Táto tabuľka reprezentuje všetky číselníky v databáze. Ušetríme síce vytvorenie jednotlivých číselníkových tabuliek, čo môže znamenať sprehľadnenie dátového modelu, ale súčasne sa náš model stane trochu neprirodzený, ťažšie čitateľný a spôsobíme si s tým aj nutnosť uvádzať vo všetkých dotazoch ďalšiu podmienku navyše (id_číselníka), prípadne budeme nútení vytvoriť pohľady pre každý číselník, čo nás však trochu posúva späť do východiskovej situácie. Tento spôsob sa zvykne označovať ako *One True Lookup Table*, viac informácií o negatívach takéhoto riešenia je možné nájsť napr. na (Celko, 2005). Z uvedených dôvodov preferujeme samostatné číselníkové tabuľky.

Všetky náznaky enumerovaných hodnôt sme nahradili v STIFe zodpovedajúcimi číselníkovými údajmi.

Redundantné dáta predstavujú v databázach vždy riziko a ťažkosti. Nenormalizované databázy je vhodné prevádzkovať iba pri veľmi špecifických úlohách (napríklad v on-line dátových skladoch kvôli výkonu).

Neflexibilné hierarchie. Niekedy je dátový model navrhnutý s obmedzeniami, ktoré sa môžu zdať rozumné v dobe návrhu, ale časom sa stanú prekážkou. V modeli systému Študent môže mať napríklad jeden študijný program najviac 5 podblokov. Navyše hierarchia predmetov nie je jednotná – na najvyššej úrovni stoja študijné programy uložené v tabuľke SPG, pod nimi stoja tzv. bloky (tabuľka BPR) a bloky sa delia na tzv. subbloky (tabuľka SBP). V STIFe máme jednotnú hierarchizáciu predmetov – navrhli sme tzv. skupiny predmetov, čo sú univerzálne množiny predmetov bez ohľadu na ich pozícii v hierarchii a ich názov.

Navyše majú atribút povinnosti, čo sama o sebe je informácia, ktorá je v súčasnosti vedená na rozličných miestach.

Význam dát je závislý od usporiadania riadkov. Konkrétne v systéme Študent ide o tabuľku `SBP` (subbloky predmetov), pri interpretácii ktorej závisí na relatívnom usporiadaní riadkov.

Neudržiavané dáta. Ďalším veľkým problémom sú relácie, ktorých záznamy sa neudržiavajú. Systém Študent je, ako sme už spomenuli, prevádzkovaný na viacerých fakultách, a každá z nich je zodpovedná za stav údajov, ktoré sa v nej nachádzajú. Niektoré fakulty k tomu problému pristupujú menej zodpovedne ako iné. Môžeme si zobrať napríklad tabuľku UCP – Zoznam učiteľov. Niektorí učitelia sa v nej nachádzajú viackrát pod inými kódmi, čo znemožňuje napísať aj taký základný dopyt, ako je počet predmetov, ktoré daný učiteľ vyučuje. Nasadenie STIF umožní tieto dáta prefiltrovať a získať aj z iných, dôveryhodných zdrojov.

Názvové konvencie. Aj keď nie dôvod na zmenu existujúceho či tvorbu abstraktného dátového modelu, názvové konvencie vedia prácu uľahčiť, ale aj sťažiť. Existuje viacero metodík, ako tvoriť názvy objektov databáz (tabuliek, stĺpcov, indexov, ...), každá firma či dokonca projekt môžu mať vlastné, osvedčené názvoslovie. Osobný názor autora práce, ktorý môže ďalej odporučiť, je vyvarovať extrémne skráteným formám a nezabehnutým skratkám. V prostredí Clipper, v ktorom je Študent naprogramovaný, sú isté príliš strinktné obmedzenia na dĺžku identifikátorov, preto sú stĺpce pomenovávané ako `stats`, `forov` či `ucitl`. Vývojár v spleti desiatok tabuliek a stoviek stĺpcov nebude vedieť pri písaní rýchlo odvodiť správny názov (stĺpec so zoznamom učiteľov je nazvaný `ucitel`, `ucitelia`, `ucit`, `uc`, ...?). Dnešné databázové systémy nemajú problém s dĺžkou identifikátorov aj vo veľkosti rádovo až sto znakov, preto nie je dôvod na nezaužívané skratky (moderné vývojové prostredia poskytujú aj pomoc v podobe rozbaľovacieho zoznamu ako programátor píše identifikátory, preto argument, že by musel nadmerne vypisovať tiež odpadáva). Autor je tiež za zdržanie sa v podobe rôznych prefixov, hlavne tých označujúcich tabuľky a databázové procedúry (`T_Ucitelia`, `sp_ZapisPredmet`), ktoré zbytočne znečisťujú menný priestor modelu (moderné vývojové prostredia vedia filtrovať rôzne databázové objekty, takisto je možné písať dopyty, ktoré vyberú len tabuľky, len procedúry a pod.). Autor je ďalej za anglické názvy, ktoré sú medzinárodné (nikdy nevieme, či bude na nejakej časti pracovať zahraničný tím).

4.6.2. Ideálny dátový model

Kvôli uvedeným faktom sme pristúpili ku korekcii a rozhodli sme sa, že sa pokúsime vybudovať ideálny, očistený model študijnej agendy. Dôvodov bol viac – zhrnieme ich:

- analýza dát ukázala nedostatky modelu ako takého, ako aj problémy s čistotou dát,
- niektoré aplikácie požadujú dáta študijnej agendy, ktoré sa v Študentovi nevedú (napríklad to vidíme v atribútoch podnikového objektu predmet, aké potrebuje web – osnova, cieľ, jazyk výučby, literatúra – a podnikových objektov predmet, variant a nasadenie

predmetu, ktoré môže poskytnúť Študent), preto treba hľadať aj ďalšie zdroje – myslíme si, že vznikne prehľadnejšia a jednoduchšia integračná architektúra na jednej strane s importmi z viacerými zdrojmi a na strane druhej aplikácie, ktoré dáta požadujú, sa nebudú musieť o tieto zdroje zaujímať,

- čo fakulta, to inštancia Študenta a samostatná databáza, centrálny dátový model, ktorý navyše zintegruje aj údaje jednotlivých fakúlt, bude výhodou,
- Študent nemá historické údaje, každý rok sa robí kópia databázy a tá sa odloží, pracuje sa len s aktuálnymi údajmi – ADM umožní viesť aj históriu a bude možné kedykoľvek k nej pristupovať.

Po netriviálnom množstve času stráveného stretnutiami, na analytických a návrhárskych prácach, sme prišli k novému modelu pre študijnú agendu s kódovým menom STIF (Student Interface). Logický dátový model je uvedený na obrázkoch 7-12 a 7-13, fyzický na obrázku 7-14. Popisy relácií nájde čitateľ v prílohe E.

Chýbajúce informácie o predmetoch – jazyk výučby, osnovu, literatúru a cieľ predmetu – musíme získať inak, pretože sa v systéme Študentov nevyskytujú. Zistili sme, že na FMFI UK sa udržujú na súborovom serveri tzv. informačné listy predmetov, ktoré tieto údaje obsahujú. Informačné listy majú podobu dokumentov formátu Microsoft Word a RTF. Po menších testoch si myslíme, že sa budú dať spracovať ako dátové zdroje do STIF.

Zhrnuté predchádzajúce, náš zámer sa posunul k vybudovaniu centrálneho systému poskytujúcemu dáta z oblasti študijnej agendy z celej univerzity. Dáta by sa mali doňho dostať z týchto zdrojov:

- databázy systému Študent prevádzkovaných na konkrétnych fakultách,
- informačné listy predmetov (tu máme situáciu zatiaľ overenú len na FMFI UK).

Historizácia bude prebiehať v samostatnej historizačnej tabuľke pre všetky relácia (schéma názov tabuľky, názov stĺpca, stará hodnota, nová hodnota, dátum konca platnosti). Dôvod je ten, že iné riešenia by nám podstatne znečistili dátový model a priniesli technické ťažkosti (napríklad by sme museli predefinovať primárne kľúče na viaczložkové a zahrnúť do nich časové údaje). Takýto prístup si môžeme dovoliť, pretože väčšina dopytov bude prebiehať na aktuálnych údajoch.

4.7. Architektúra

V našom príklade budeme pokračovať navrhnutím architektúry a prezentujeme jej časť v kontexte tohto príkladu integrácie študijnej agendy.

Keďže plánované riešenie má byť riešením pre celú UK, univerzitu s používateľmi rádovo až v desiatkach tisícov, musíme zodpovedne pristupovať k integračnej architektúre aj čo sa týka požiadaviek na výkon, robustnosť a udržiavateľnosť.

Vzhľadom na to, že do projektu by sa mali časom zapojiť takmer všetky aplikácie na UK (v ďalších inkrementoch projektu), ktorých počet po zarátaní viacerých inštancií dosahuje rádovo až niekoľko desiatok (!), musíme navrhnúť architektúru, ktorá je veľmi prehľadná a dá sa riadiť na centrálnej úrovni, pričom je možné ju pozorovať a zasahovať do jej aktuálneho stavu. Ak postavíme architektúru na zbernici ako globálnom komunikačnom vzore, za predpokladu, že dobre zvolíme konkrétneho dodávateľa zbernice, máme možnosť tieto požiadavky naplniť, ako sme podotkli v 3. kapitole. Musíme však povedať, že argument za integračnú zbernicu bol v našom prípade aj silne ovplyvnený predchádzajúcim nákupom takého produktu (Sonic ESB).

Pokračujúc v našom príklade by sme chceli navrhnúť integračnú architektúru pre tieto systémy:

- Študent, resp. databázy systémov Študent na fakultách UK,
- informačné listy predmetov na FMFI UK,
- STIF,
- web UK,
- anketa FMFI UK.

Okrem týchto systémov sa na diagramoch vyskytnú aj aplikácie Ubytovanie, Knižnica, CDO a CTS, ktoré ale nie sú súčasťou tu popísaného výseku prác integračného projektu UK.

Rozhodli sme sa, že v záujme celistvosti a jednotnosti integračnej architektúry, budú všetky aplikácie, ktoré požadujú dáta študijnej agendy, komunikovať prostredníctvom integračnej zbernice.

Dáta do STIF z databáz Študenta a informačných listov však nebudú prúdiť cez zbernicu, ale vlastnými spojeniami typu ETL. Po zvážení všetkých argumentov pre a proti sme sa tak rozhodli, pretože:

- v prípade Študent –STIF ide o čistú integráciu na dátovej úrovni, na riešenie ktorej existujú sofistikované nástroje, ktoré nám umožnia lepšiu kontrolu a rýchlejší vývoj,
- nepotrebujeme výhody centralizovaného riešenia integračnej zbernice, pretože táto komunikácia je veľmi účelová, komunikujúce strany sú fixne dané a nemá žiadne predpoklady na rozširovanie.

Do tohto musíme zarátať aj plánovaný postupný prechod na novú generáciu systému Študent, ktorý je ešte v štádiu príprav, a tak sa dnes presne ani ešte nevie, akú veľkú časť študijnej agendy preberie. Navrhli sme teda, aby ostatné aplikácie komunikovali so STIFom výlučne prostredníctvom nasadených služieb vytvorením fasády služieb. Komunikácia tak bude prebiehať iba cez explicitne definované rozhrania služieb a aplikácie sa nebudú starať, či dáta pochádzajú z databázy, informačných listov, nového systému Študent či iného zdroja. Pozri obrázok 7-9 s diagramom navrhnutej integračnej architektúry a obrázok 7-11 po nasadení novej generácie Študenta.

Nasledujúca tabuľka uvádza navrhnuté typy portov, cez ktoré budú systémy komunikovať:

ID	Typ	Systém	Port
Študent1	Ex	Študent (I – stará generácia)	exporty dátových súborov DBF vo formáte CSV, pre každú tabuľku samostatný súbor, súbory sa budú ukladať na dohodnutý súborový server v časovom intervale 1 deň
IL1	Ex	Informačné listy predmetov FMFI UK	uložené v dohodnutej adresárovej štruktúre na súborovom serveri Gauss na dohodnutej ceste, pristupovať sa bude pod špeciálnym používateľským účtom
STIF 1	Im	STIF	priamy prístup do databázy MS SQL Server po špeciálnym používateľským účtom
STIF 2	Ex	STIF	fasáda služieb komunikujúcich XML správami
Web1	Im	Web	podľa ďalšej dohody s web mastermi buď sa bude pracovať priamo nad prijatými XML súbormi cez XSLT transformácie do HTML alebo dáta poputujú do vlastnej databázy
Anketa1	Im	Anketa FMFI	priamy prístup do databázy anketového systému

Tabuľka 12 Typy portov

Medzi týmito portami sme navrhli tieto komunikačné kanály:

ID	Port1	Port2	Popis
CC1	Študent1	STIF1	prichádzajú vybrané dáta exportované z databáz systémov Študent – fakticky sa jedná o viacero kanálov, pre každú fakultu jeden; implementácia: vlastný kód kombinovaný s MS SQL DTS
CC2	IL1	STIF 1	prichádzajú dáta z informačných listov predmetov do STIF; implementácia: vlastný kód pracujúci s objektovým modelom MS Word cez COM
CC3	STIF 2	Web1	volanie operácií na rozhraní STIF: GetSubjectsHierarchy, GetSubjectsInfo, GetStudyCourses, druhá fáza implementácie: GetEnrolledStudentsOnSubject, GetEnrolledSubjectsForStudent,
CC4	STIF 2	Anketa1	volanie operácií na rozhraní STIF: GetSubjectsHierarchy, GetEnrolledSubjectsForStudent.

Tabuľka 13 Komunikačné kanály

4.8. Komunikačný návrh

Rozhranie fasády služieb systému STIF je nasledovné:

Názov operácie	Popis	Správy
GetSubjectsHierarchy	požiadavka na strom predmetov v skupinách predmetov, študijných programoch a odboroch	M1, M1R
GetSubjectsInfo	požiadavka na informácie o predmetoch	M2, M2R
GetEnrolledStudentsOnSubject	požiadavka na zoznam zapísaných študentov na predmete	M3, M3R
GetEnrolledSubjectsForStudent	požiadavka na zoznam predmetov, ktoré si študent zapísal	M4, M4R
GetStudyCourses	požiadavka na strom študijných programov a odborov	M5, M5R

Popisy správ:

Id:	M1	**Typ:**	Request
Názov:	GetSubjectsHierarchy		
Popis:	požiadavka na zoznam predmetov zaradených hierarchicky v skupinách predmetov, študijných programoch a odboroch		
Parametre:	`faculty`: ID fakulty, na ktorú sa vykoná filter,		
	`branches`: ID odborov, na ktoré sa vykoná filter,		
	`courses`: ID št. programov, na ktoré sa vykoná filter,		
	`subjects`: ID predmetov, na ktoré sa vykoná filter,		
	`year`: za aký akademický rok sa očakáva odpoveď		
Telo:	NULL		
Odpoveď:	M1R		
Výnimky:	neplatné parametre		

Id:	M1R	**Typ:**	Reply
Názov:	SubjectsHierarchy		
Popis:	zoznam predmetov zaradených hierarchicky v skupinách predmetov, študijných programoch a odboroch		
Telo:	telo tvoria dáta v formáte XML ako v uvedenom príklade (Obrázok Obrázok 7-1)		

Id:	M2	**Typ:**	Request
Názov:	GetSubjectsInfo		
Popis:	požiadavka na detailné informácie predmetov		
Parametre:	`subjects`: zoznam ID predmetov, pre ktoré požadujeme informácie,		
	`year`: za aký akademický rok sa očakáva odpoveď.		
Telo:	NULL		

Odpoveď:	M2R
Výnimky:	neplatné parametre

Id:	M2R	**Typ:**	Reply
Názov:	SubjectsInfo		
Popis:	zoznam predmetov s detailnými informáciami o nich		
Telo:	telo tvoria dáta v formáte XML ako v uvedenom príklade (Obrázok Obrázok 7-2)		

Id:	M3	**Typ:**	Request
Názov:	GetEnrolledStudentsOnSubject		
Popis:	požiadavka na zoznam študentov, ktorí si zapísali predmet		
Parametre:	`subjectId`: ID predmetu,		
	`year`: za aký akademický rok sa očakáva odpoveď.		
Telo:	NULL		
Odpoveď:	M3R		
Výnimky:	neplatné parametre		

Id:	M3R	**Typ:**	Reply
Názov:	EnrolledStudentsOnSubjects		
Popis:	zoznam študentov, ktorí si zapísali predmet		
Telo:	telo tvoria dáta v formáte XML s koreňovým elementom `students` a vnorenými elementmi `student` s jediným atribútom `studentId`		

Id:	M4	**Typ:**	Request
Názov:	GetEnrolledSubjectsForStudent		
Popis:	požiadavka na zoznam predmetov, ktoré si študent zapísal		
Parametre:	`studentId`: ID študenta,		
	`year`: za aký akademický rok sa očakáva odpoveď.		
Telo:	NULL		
Odpoveď:	M4R		
Výnimky:	neplatné parametre		

Id:	M4R	**Typ:**	Reply
Názov:	EnrolledSubjectsForStudent		
Popis:	zoznam predmetov s detailnými informáciami o nich		
Telo:	telo tvoria dáta v formáte XML s koreňovým elementom `subjects` a vnorenými elementmi `subject` s jediným atribútom `subjectId`		

Id:	M5	**Typ:**	Request
Názov:	GetStudyCourses		
Popis:	požiadavka na strom študijných programov a odborov		
Parametre:	`faculty`: ID fakulty, na ktorú sa použije filter,		
	`branches`: zoznam ID odborov, na ktoré sa použije filter,		
	`courses`: zoznam ID programov, na ktoré sa použije filter,		
	`year`: za aký akademický rok sa očakáva odpoveď.		
Telo:	NULL		
Odpoveď:	M5R		
Výnimky:	neplatné parametre		

Id:	M5R	**Typ:**	Reply
Názov:	StudyCourses		
Popis:	zoznam študijných programov a odborov, do ktorých patria		
Telo:	telo tvoria dáta v formáte XML ako v uvedenom príklade (obrázok Obrázok 7-3)		

Okrem toho STIF disponuje servisným rozhraním, prostredníctvom ktorého aplikácie môžu žiadať o poskytnutie aktualizácií údajov prostredníctvom diffgramov. Diffgram je XML správa schémou totožná s pôvodnou správou, ale prenáša iba podmnožinu zmenených dát. Pre aplikácie, ktoré so systémom STIF komunikujú, nie je dôležité, ako je táto funkčnosť zabezpečená. Komunikácia prebieha vzorom publish-subscribe (pozri (Hohpe, a iní, 2003)).

4.8.1. Importy zdrojových dát do ADM

Zdroje dát pre ADM Študent predstavujú:

- systém Študent (prefix ST),
- informačné listy predmetov v podobe RTF súborov so záložkami (prefix IL, prepája sa s predmetmi adresárovou štruktúrou popísanou v prílohe F).

Prepoklady:

- na začiatku prebehne nárazový import zo všetkých databáz a informačných listoch,
- systém Študent sleduje zmeny v jednotlivých záznamoch a vie generovať zmeny (musí sa doprogramovať),
- na súborovom serveri beží v pozadí služba, ktorá notifikuje pri zmene informačného listu,
- každú noc (o 3:00) prebehne diffgramový import na základe zmien v Študentovi a notifikáciách na súborovom serveri. V prípade zaregistrovania zmien (publish/subscribe) vo fasáde služieb STIF, sa následne pošlú diffgramové správy.

Courses

Cieľový stĺpec	Zdroj	Poznámka
CourseId		auto
Code	St.SPG.idspg	
Name	St.SPG.nazov	
Degree	St.SPG.stups	
StudyForm	St.SPG.forms	Číselníkové hodnoty, v zdroji e = externá, d = denná.
StandardLength	St.SPG.stdls	
Title	St.SPG.titul	V zdroji prídu ako texty ("Mgr."), previesť do číselníkových kódov.
Branch1Id	St.SPG.sodb1	previesť na rezortný kód odboru, vzťah medzi ním a kódom v systéme Študent sa určí napevno a uloží do listu v MS Excel, odkiaľ sa bude brať
Branch2Id	St.SPG.sodb2	ako Branch2Id
RootSubjGroupId		FK na koreňovú skupinu predmetov. Táto skupina predmetov je umelá a treba ju vytvoriť.
StudentId	St.SPG.idspg	ID v systéme Študent

SubjectGroups

Údaje v tejto tabuľke sa naplnia nasledovne:

1. Pre každý študijný program príslušný záznam v relácii `St.SPG` špecifikuje tzv. bloky predmetov a atribútoch `blokp1`, `blokp2`, `blokp3`, `blokp4` a `blokp5`. Tieto bloky predmetov sa stanú skupinami predmetov, ktorých rodič je koreňová skupina predmetov pre daný študijný program.[25] Ich atribúty sa zoberú z tabuľky `St.BPR` (prepoja sa cez cudzí kľúč v `St.SPG.blokpX`). Pre atribúty platí: `Name = St.BPR.nazov`, `ParentSubjectGroupId = ID` na umelo vytvorenú koreňovú skupinu predmetov asociovanú s týmto študijným programom. Voliteľnosť tejto skupiny predmetov sa určí napevno a uloží sa do listu aplikácie Microsoft Excel, odkiaľ sa bude brať.
2. Nižšie úrovne prichádzajú z tabuľky `St.SBP`, a to nasledovne: záznamy sa spracovávajú postupne podľa atribútu `blokp`, ktorý je cudzím kľúčom pre nadradený blok predmetov. Záznamy v rámci jednej skupiny sa utriedia podľa atribútu idsbp vzostupne (t.j. najmenšie číslo nazačiatku). Postupuje sa procedurálne riadok za riadkom:

[25] Ako je uvedené v popise tabuľky SPG, tieto cudzie kľúče môžu obsahovať otáznikové zástupné znaky.

a. pre aktuálny záznam sa vytvorí nová skupina predmetov, ktorej rodič je, ak nie sme vnorení, skupina predmetov reprezentovaná blokom `blokp`.

b. ak aktuálny záznam má hodnotu atribútu atsbp rovnú znaku '#', pokračuje sa ďalej podľa bodu 3a. Ak má hodnotu '%', vnoríme sa, t.j. nasledujúcich `psbk` záznamov bude mať ako rodiča túto skupinu predmetov, okrem toho je spracovanie rovnaké ako v bode 3a.

Existujú dve špeciálne koreňové skupiny predmetov, ktoré nie sú asociované so žiadnym študijným programom:

- celofakultné predmety, v `BPR` kľúč `blokp = XXXX`,
- externé fakulty UK, v `BPR` kľúč `blokp = XFUK`.

Pre obe platí `Optionality = 99, OptParam = NULL`.

SubjectRealizations

Cieľový stĺpec	Zdroj	Poznámka
SubjectRealizationId		auto
VariantId	ID prevedené z ST.VEP.uipmt	ID variantu predmetu, ktorý vznikol z príslušného záznamu v tabuľke St.RLP
RcmdYears	St.VEP.orost	
Prereq	St.VEP.idprq	Všetky prerekvizity sú vedené ako logický súčin bez zátvoriek. Previesť do formátu id_1*id_2*...*id_n, kde id_1, id_2, ..., id_n sú ID predmetov v ADM.
RcmdPrereq	St.VEP.idprq, St.VEP.atprq	Zobrať ID predmetov z idprq na tých pozíciách, kde v atprq sa nachádza výkričník (!), previesť na ID predmetov v ADM a zreťaziť cez hviezdičku (*).
SubjGroupId	ID prevedené z St.VEP.idsbp	ID skupiny predmetov, ktorá vznikla zo záznamu v tabuľke SBP daná PK uvedeným v atribúte St.VEP.idsbp.

SubjectVariants

Cieľový stĺpec	Zdroj	Poznámka
VariantId		auto
SubjectId	ID prevedené z St.RLP.uipmt zobratím	ID predmetu, ktorý vznikol zo záznamu

	prvých dvoch znakov	v tabuľke St.PREDMETY
Active	St.RLP.stats	Previesť na číselníkový kód z TeachingStatus
Term	St.RLP.semes	Previesť na číselníkový kód z Terms.
ExamRatio	St.RLP.phodn	Hodnoty sú vo formáte D1/D2. Ako pumpovanú hodnotu zobrať D2.
Credits	St.RLP.hkred	
Language	IL.jazyk	previesť do číselníkového ISO kódu jazykov
DepartmentId	St.RLP.gespr	previesť na kód číselníka Departments

Subjects

Cieľový stĺpec	Zdroj	Poznámka
SubjectId		auto
Code	St.PRM.kod	
Name	St.PRM.nazov	
EnglishName	St.PRM.anglnazov	
StudentId	St.PRM.idprdmt	
Goal	IL.ciel	
Synopsis	IL.osnova	
Literature	IL.literatura	
FacultyId	ID fakulty, z ktorej pochádza tento import	

TeachersVariants

Pre každý záznam tabuľky St.RLP sa rozdelí zoznam učiteľov uvedený v atribúte uctil (formát je uvedený v popise tejto tabuľky). Každá dvojica variant-učiteľ predstavuje jeden záznam v tabuľke TeachersVariants.

Cieľový stĺpec	Zdroj	Poznámka
VariantId		
TeacherId		

FormExtentsVariants

Pre každý záznam v tabuľke St.RLP sa rozparsuje hodnota atribútu forov na jednotlivé formy výučby a rozsahy. Formát je uvedený v popise tabuľky St.RLP. Každá forma bude predstavovať samostatný záznam v tabuľke FormExtentsVariants.

Cieľový stĺpec	Zdroj	Poznámka
FormId		
VariantId		
Extent	Časť atribútu St.RLP.forov	
TimeUnit	Časť atribútu St.RLP.forov	

CoursesFaculties

Pri každom importe sa naplní ID programu a ID fakultou, z ktorej pochádza.

Cieľový stĺpec	Zdroj	Poznámka
CourseId		auto
FacultyId		

4.8.2. Slovensko-anglický slovník študijnej agendy

slovensky	anglický význam
forma výučby	teaching form
literatúra	literature
predmet	subject
prerekvizita	qualifying subject, prerequisite
rozsah výučby	teaching extent
osnova	curriculum
semester	semester, term *(ako všeobecná jednotka ak. roka)*
študijný odbor	branch of study, study branch
študijný program	course of study
záverečné hodnotenie	concluding exam

4.9. Podnikový model

Na konci tejto kapitoly ešte uvádzame, ako by sme postupovali pri analýze potrieb integrácie vychádzajúc z modelu podnikových procesov.

4.9.1. Identifikácia oblastí

Univerzita Komenského je štandardnou univerzitou, a preto sa jej podnikový model príliš nelíši od ostatných podobných univerzít a dá sa považovať viac-menej za štandardný. Hlavné procesné oblasti sú nasledovné:

- *Študijná agenda* – všetky aktivity súvisiace so štúdium na UK, sleduje celý životný cyklus študenta, od príprav na prijímacie skúšky až po ukončenie štúdia, vo všetkých stupňoch.
- *Finančná agenda a účtovníctvo* – štandardné ekonomické podporné procesy.
- *Knižničné služby* – procesy a evidencie pre knižnicu (nákupy knižničných jednotiek, výpožičky, vyraďovanie, ...), vrátane evidencie a zaraďovania záverečných prác študentov UK.
- *Granty* – procesy udeľovania grantov a čerpania prostriedkov z nich,
- *Študentská anketa* – procesy sledujúce spätnú väzbu na stav výučby na UK od študentov,
- *Ubytovanie* – procesy prideľovania ubytovania na internátoch UK,
- *Univerzitné a fakultné webové rozhrania* – webové sídla, slúžiace jednak ako prezentácia UK a jej fakúlt smerom von a jednak

poskytujú dôležitú komunikačnú bránu medzi univerzitou a členmi akademickej obce.

Ako zaujímavosť možno uviesť, že v súčasnosti je každá z týchto oblastí riešená samostatným softvérovým systémom, prípadne viacerými systémami, pretože fakulty majú značnú autonómiu aj v ich voľbe a správe.

4.9.2. Procesy

Ilustrovať budeme práce budeme ďalej na jednom z podnikových procesov týkajúcich sa študijnej agendy, konkrétne na procese prijímania študenta na univerzitu.

Proces BP1/Prijímacie konanie

Tento proces predstavuje súhrnný pohľad na proces prijatia na UK a je dekomponovaný procesmi BP1.1/Žiadosti o prijatie na štúdium, BP1.2/Priebeh prijímacích skúšok, BP1.3/Vyhodnotenie prijímacích skúšok a BP1.4/Odvolanie voči rozhodnutiu o prijatí/neprijatí. Nie sú dekomponované jednotlivé aktivity, ale skupiny aktivít.

Aktéri:

- uchádzač
- spracovateľ (typicky referenti študijného oddelenia)
- hodnotiaci – vyhodnocuje prijímaciu skúšku
- odvolacia autorita – rozhoduje o odvolaní voči výsledku prijímacej skúšky

Aktivity/Rozhodovania:

Vyplniť prihlášku. Podrobnosti v BP1.1.

Odoslať prihlášku. Podrobnosti v BP1.1.

Založiť prihlášku do evidencie. Podrobnosti v BP1.1.

Rozhodnutie Prijatý bez skúšok? Podrobnosti v BP1.2.

Pripraviť pozvánku na skúšky. Podrobnosti v BP1.2.

Zložiť skúšky. Podrobnosti v BP1.2.

Odoslať rozhodnutie. Podrobnosti v BP1.2 a Podrobnosti v BP1.4.

Vyhodnotiť skúšky. Podrobnosti v BP1.3.

Rozhodnutie Odvolanie? Podrobnosti v BP1.4.

Vyriešiť odvolanie. Podrobnosti v BP1.4.

Proces BP1.1/Žiadosti o prijatie na štúdium

Proces popisuje priebeh činností podania si žiadosti na štúdium na UK.

Aktéri:

- uchádzač (o štúdium na UK)
- spracovateľ (žiadosti) – typicky referent študijného oddelenia

Podporné systémy:

- systém študijnej agendy (Študent)

Aktivity/Rozhodovania:

Vyplniť prihlášku. Uchádzač vyplní prihlášku v štandardnom formáte poskytovanom MŠ SR. Pre každý stupeň štúdia existuje samostatná prihláška. V súčasnosti sú prihlášky iba v papierovej podobe.

Odoslať prihlášku. Uchádzať prihlášku odošle poštou alebo ju odovzdá osobe v podateľni príslušnej fakulty.

Zaplatiť poplatok. V zmysle platných predpisov sú fakulty oprávnené požadovať uhradenie poplatku, z ktorého sa hradia náklady pokrývajúce agendu prijímacieho konania. Uchádzač tento poplatok zaplatí buď poštovou poukážkou alebo prevodom na účet.

Zaevidovať prihlášku. Spracovateľ došlé prihlášky zaeviduje do systému študijnej agendy a fyzicky do príslušnej zložky.

Spárovať poplatok. Spracovateľ došlú platbu za poplatok spáruje so žiadosťou a poznačí si, že poplatok bol zaplatený.

Žiadosť o aktualizáciu údajov. Od uchádzača sa akceptujú zmeny ohľadom jeho údajov (súčasné priezvisko, bydlisko, telefónne číslo a pod.). Zmena odborov, o štúdium ktorých sa uchádza, nie je možná.

Aktualizovať údaje. Spracovateľ žiadanú zmenu vykoná, tak fyzicky v príslušnej zložke, ako aj v systéme študijnej agendy.

Proces BP1.2/Priebeh prijímacích skúšok

Proces popisuje vykonanie prijímacej skúšky a činnosti bezprostredne súvisiace s jej prípravou.

Aktéri:

- spracovateľ – typicky referent študijného oddelenia
- uchádzač (o štúdium na UK)

Aktivity/Rozhodovania:

Vyhodnotiť možnosť prijatia bez prijímacích skúšok. Uchádzačom s vynikajúcimi výsledkami na strednej škole môže dekan fakulty odpustiť prijímaciu skúšku (alebo jej časť) na niektoré odbory. Vo všeobecnosti platí pravidlo, že čím atraktívnejší študijný odbor, tým je menšia pravdepodobnosť odpustenia prijímacej skúšky. Presné kritéria sú platné na konkrétny akademický rok.

Rozhodnutie Prijatý bez skúšok? Áno, ak výsledkom predchádzajúcej aktivity pre uchádzať je „prijatý bez skúšok“, nie v opačnom prípade.

Rozdeliť do skupín a termínov. Spracovateľ uchádzačom pridelí kategóriu na základe podobnosti (rovnaký odbor, rovnaký test, ...) a týmto pridelí termín (preferuje sa, aby uchádzači z jednej kategórie absolvovali testy v jeden deň kvôli možnému ovplyvňovaniu).

Odoslať pozvánky. Spracovateľ rozošle pozvánky uchádzačom.

Vygenerovať, vytlačiť a rozdistribuovať čiarové kódy. V snahe zachovať čo najväčšiu anonymitu vyplnené testy neobsahujú meno uchádzača, ale iba čiarový kód. Tieto sa musia vygenerovať a vytlačiť.

Vylosovať čiarový kód. Uchádzač si na mieste konania testov vylosuje sadu čiarových kódov, ktoré ho budú anonymne reprezentovať na všetkých materiáloch.

Absolvovať PS. Uchádzať absolvuje testy, ktoré má v daný deň naplánované. Ich konkrétna podoba a náplň sa stanovuje na každý akademický rok osobitne. Uchádzať na každý vyplnený test nalepí svoj vylosovaný čiarový kód.

Zozbierať testy s kódmi. Spracovateľ (v zastúpení dozorom) zozbiera vyplnené testy od uchádzačov. Uchádzač navyše odovzdá jeden čiarový kód, ktorý sa priradí k jeho skutočnej identite – toto priradenie sa považuje za tajné a nemalo by sa dostať mimo úseku študijného oddelenia (v žiadnom prípade nie do rúk hodnotiacich). Jeden čiarový kód si uchádzač ponechá, aby si mohol čo najskôr pozrieť počet získaných bodov, ktoré sa objavia spolu s kódmi.

4.9.3. Podnikové objekty

Model podnikových procesov nám dáva prvú možnosť zozbierať dôležité podnikové objekty. Sústredíme sa, ako sme spomenuli v časti o procese integrácie, na prechody medzi aktérmi a aktivity, ktoré svojou povahou alebo priamo popísanou činnosťou implicitne či explicitne spomínajú iných aktérov alebo rovno systémy.

Pozrime sa bližšie na proces BP1.1/Žiadosti o prijatie na štúdium a podnikové objekty, ktoré sa v ňom vyskytujú so stručným odôvodnením, prečo sú pre nás zaujímavé z hľadiska integrácie:

- spracovateľ, uchádzač (aktéri)
- prihláška (prechod medzi aktérmi, artefakt medzi uchádzačom a UK),
- poplatok (prechod medzi aktérmi, uchádzač musí zaplatiť a musí sa niekde evidovať),
- aktualizácia údajov (udalosť od aktéra spôsobujúca zmenu údajov na UK)

To isté pre proces BP1.2/Priebeh prijímacích skúšok

- spracovateľ, uchádzač (aktéri),
- prijatie bez skúšok (podnikové pravidlo nad prihláškou a priebehom predchádzajúceho štúdia),
- priebeh predchádzajúceho štúdia (využíva podnikové pravidlo; súhrn prospechu na strednej škole, účastí a ocenení na súťažiach),
- pozvánka na PS (komunikácia medzi UK a uchádzačom),
- čiarový kód (pôjde späť do systému s vyplnenými testami; na základe čoho sa generuje – treba informácie z ostatných systémov?)
- rozhodnutie o prijatí (komunikácia medzi UK a uchádzačom),

- deň PS (udalosť)

Na základe toho môžeme pripraviť časť základného zoznamu podnikových objektov:

Názov	Popis	Identifikátory	Ďalšie vlastnosti	Poznámky
spracovateľ	zodpovedný za koordináciu a prípravu PS	(osoba)	(osoba)	
uchádzač	záujemca o štúdium na UK	(osoba)	(osoba)	
prihláška na štúdium	prihláška záujemcu o štúdium na UK	uchádzač	odbory, na ktoré sa hlási	
poplatok	úhrada za PS	uchádzač	suma	
aktualizácia údajov	žiadosť o zmenu údajov uchádzača	uchádzač	údaje, ktoré treba zmeniť	udalosť
prijatie bez skúšok	rozhoduje, či je možné prijať uchádzača na daný odbor bez PS	uchádzač, odbor	priebeh štúdia na SŠ	podnikové pravidlo
pozvánka na PS	písomné pozvanie na PS	uchádzač, termín, odbor	testy, pokyny	
priebeh štúdia na SŠ	dôležité výsledky počas štúdia na SŠ, ktoré je možné zobrať do úvahy na prijatie bez PS	uchádzač	prospech, účasť na olympiádach, účasť na súťažiach	
čiarový kód	kód, pod ktorým bude uchádzač vedený na testoch	uchádzač		väzba uchádzač – kód je počas hodnotenia testov tajná
rozhodnutie o prijatí	kladné alebo záporné	uchádzač, odbor		
deň PS	termín konania PS	uchádzač, odbor		udalosť

Tabuľka 14 Podnikové objekty pre proces prijímacieho konania

Všimnime si, že všetky podnikové objekty, ako aj ich udalosti, sú na pomerne vysokom stupni abstrakcie, tak ako sme to opísali v integračnom procese.

V prípade podnikových objektov spracovateľ a uchádzač je v poli identifikátory a ďalšie vlastnosti odkaz na podnikový objekt osoba (ktorý sme tu neuviedli).

4.10. Charakteristiky systémov

Pre základný životný cyklus študenta máme namodelované všetky podnikové procesy. Procesno-systémový model nám dáva pohľad, ako je daný podnikový proces implementovaný. Na základe rozhovorov a ukážok sme zostrojili procesno-systémové modely. Uvádzame ukážku pre podnikový proces BP1.1/Žiadosti o prijatie na štúdium. Diagram sa nachádza na obrázku 7-8.

Popis súčasných spojení:

ID	Rozhranie	Popis
C1	uchádzač/ podateľňa	uchádzač prinesie osobne papierovú prihlášku do podateľne
C2	uchádzač/pošta	uchádzač pošle papierovú prihlášku poštou na adresu fakulty
C3	podateľňa/spracovateľ	podateľňa doručí všetky prihlášky spracovateľovi
C4	pošta/spracovateľ	poštová služba doručí všetky prihlášky spracovateľovi
C5	spracovateľ/Študent	spracovateľ prepíše údaje z papierovej prihlášky ručne do Študenta
C6	uchádzač/pošta	uchádzač uhradí poplatok za náklady na prijímacie konanie poštovou poukážkou typu U, kde sa identifikuje prostredníctvom variabilného symbolu
C7	uchádzač/banka	uchádzač uhradí poplatok príkazom na úhradu na účet fakulty v Štátnej pokladnici, pričom sa identifikuje variabilným symbolom
C8	pošta/spracovateľ	spracovateľovi sa doručia fyzické časti poštových poukážok typu U od uchádzačov s poplatkami
C9	banka/spracovateľ	spracovateľ si pozrie výpise došlých prostriedkov na účte pre úhrady poplatkov spojených s prijímacím konaním
C10	uchádzač/Študent	spracovateľ na základe poštových poukážok a výpisov z účtu priradí došlé poplatky uchádzačom zaškrtnutím príslušného poľa v systéme Študent

Tabuľka 15 Prehľad spojení v súčasnej implementácii procesu prijímacieho konania

Spolu s podnikovým procesom máme k dispozícii kompletnú informáciu, ako sa daný proces realizuje. Modely podnikových procesov (aspoň tak, ako sme si ich zadefinovali) sú technologicky neutrálne, preto počas integračného projektu nám poskytujú vodiacu čiaru, ale ostávajú invariantné, ak si odmyslíme prípadné kroky k ich explicitnému vylepšovaniu (business process improvement). Časť, ktorá sa pri integrácii mení, je implementácia, konkrétne procesno-systémové modely.

Ďalej by sme pokračovali podľa načtnutého procesu. Za zmienku ešte stoja možnosti integrácie, ktoré sme už začali skúmať, a to integrácia s poštou ako externým partnerom, kedy automatizovaním došlých platieb eliminujeme nutnosť zasielania kópií poštových poukážok, ich párovania so študentmi a označením referentkou v systéme, či zaplatil poplatok.

5. Použité skratky

skratka	význam
B2B	Business To Business
BPEL	Business Process Execution Language
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus (podniková zbernica služieb)
ETL	Extract, Transform, Load (extrakcia, transformácia, načítanie)
FaF UK	Farmaceutická fakulta UK
FiF UK	Filozofická fakulta UK
FMFI UK	Fakulta matematiky, fyziky a informatiky UK
UK	Univerzita Komenského
XML	Extensible Markup Language (rozšíriteľný značkovací jazyk)
WS	Web Service (webová služba)
WSDL	Web Services Description Language (jazyk na popis webových služieb)

6. Záver

V práci sme prezentovali naše výsledky v oblasti integrácie aplikácií. Prínos práce spočíva v dvoch dimenziách: proces integrácie aplikácií a integračný projekt na UK.

Dnes považujeme za samozrejmosť, keď sa necháme viesť rozličnými procesmi pri vývoji softvéru, ktorých máme na výber veľmi veľa. Nepodstupujeme tak znovu riziká a nevydávame sa na neznáme chodníky, ktoré už prebádali autori týchto procesov. Budeme radi, ak náš proces integrácie aplikácií bude tiež podobne nápomocný. Nami navrhnutý proces integrácie aplikácií nie je viazaný na žiadnu technológiu, môžeme pri ňom použiť ľubovoľný typ middlewaru a dokonca si ho prispôsobovať „prekrývaním" vlastných krokov. V duchu moderných procesov vývoja softvéru má iteratívno-inkrementálny potenciál, čo ocenia hlavne vedúci projektov.

Okrem procesu práca prináša aj dokumentačnú časť práce na integračnom projekte UK, na ktorom sme si proces overovali, ale aj z ktorého sme čerpali vstupy pre proces. Zanalyzovali sme potreby integrácie v oblasti študijnej agendy a prezentovali sme jej výsek v podobe návrhu integrácie niektorých aplikácií, vrátane integračnej architektúry a komunikačného návrhu. Okrem toho sme navrhli nový centrálny systém, ktorý integruje väčšinu dát študijnej agendy zo všetkých fakúlt, pričom tieto dáta čerpá z viacerých zdrojov.

Podobne ako táto práca sledovala až do konca dva hlavné ciele – proces integrácie aplikácií a integračný projekt UK, tak sa môže ďalej rozvíjať v týchto dvoch smeroch.

Proces by sme chceli ešte doplniť pohľadmi vedúceho projektu, teda otázkami a ich riešením v kontexte riadenia integračného projektu, predovšetkým plánovania a kontroly pokroku, pretože aj tu si myslíme, že existujú isté rozdiely oproti vývoju klasického softvéru, ktoré sú príliš špecifické pre integráciu. Ďalším rozšírením by mohla byť kapitola o integrácii temporálnych (časových) dát a podstatné rozpracovanie kapitoly o automatizácii medziaplikačných procesov, na ktoré sme ešte nemali, žiaľ, príležitosť, pretože doteraz sme ju v integračnom projekte UK nemali možnosť reálne nasadiť.

V integračnom projekte UK je ďalšia práca daná – pokračovať v integrácii ďalších aplikácií, medzi ktoré sa časom zaradí aj nová generácia systému Študent, tak, aby bol informačný systém na UK konečne naozaj integrovaný. Dôležité je teda povedať, že projekt nekončí a prebieha aj po uzávierke tejto práce, ako aj proces v nej popísaný, ktorý tiež chceme ďalej vylepšovať a aktualizovať ako budú prebiehať ďalšie práce.

Dúfame, že navrhnutý proces integrácie aplikácií bude užitočný pri nových integračných projektoch a osvedčí sa aj v iných prostrediach.

7. Referencie

Aalst, W.M.P. van der, a iní. 2003. Workflow Patterns. 2003, 1, Hingham : Kluwer Academic Publishers, 2003, Distributed and Parallel Databases, Zv. 14, s. 5-51. ISSN 0926-8782.

Ahern, Dennis M., a iní. 2005. *CMMI® SCAMPI Distilled Appraisals for Process Improvement.* s.l. : Addison Wesley , 2005. ISBN 0-32-122876-6.

Arora, Geetanjali a Kishore, Sai. 2002. *XML Web Services Professional Projects.* Ohio : Premier Press, 2002. ISBN: 1931841365.

Bakker, Loek. 2005. Goodbye Hub-and-Spoke, Hello ESB? Integration Architecture With BizTalk 2004. *.NET Developer's Journal.* [Online] 12. septembra 2005. [Dátum: 15. októbra 2006.] http://dotnet.sys-con.com/read/121831.htm.

Bass, Len, Clements, Paul a Kazman, Rick. 2003. *Software architecture in practice.* Second Edition. s.l. : Addison Wesley, 2003. ISBN 0-321-15495-9.

Bittner, Kurt a Spence, Ian. 2003. *Use case modeling.* s.l. : Addison-Wesley, 2003. ISBN 0-201-70913-9.

Booch, Grady, Rumbaugh, James a Jacobson, Ivar. 1998. *The Unified Modeling Language User Guide.* s.l. : Addison Wesley, 1998. ISBN 0-201-57168-4.

Brown, William J., a iní. 1998. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis.* s.l. : John Wiley, 1998. ISBN 0-471-19713-0.

Celko, Joe. 2005. One True Lookup Table. *dbazine.com.* [Online] 2005. [Dátum: 19. februára 2007.] http://www.dbazine.com/ofinterest/oi-articles/celko22.

Cerami, Ethan. 2002. *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL.* s.l. : O'Reilly, 2002. ISBN 0-596-00224-6.

Cooper, Dale F., Grey, Stephen, Raymond, Geoffrey a Walker, Phil. 2004. *Project risk management guidelines: managing risk in large projects and complex procurements.* Chichester : John Wiley, 2004. ISBN 0-470-02281-7.

Date, C.J., Hugh, Darwen a Lorentzos, Nikos A. 2003. *Temporal Data and the Relational Model.* San Francisco : Morgan Kaufmann, 2003. ISBN 1-55860-855-9.

Devínsky, Ferdinand prof. Ing. DrSc., Mederly, Peter doc. RNDr. CSc. a Mederly, Pavol Mgr. 1998. *Strategická štúdia pre integrovaný a komunikačný systém Univerzity Komenského.* Bratislava : Univerzita Komenského, 1998.

Fowler, Martin. 1996. *Analysis Patterns: Reusable Object Models.* s.l. : Addison-Wesley, 1996. ISBN: 0201895420.

—. 2003. *UML distilled: a brief guide to the standard object modeling language.* Third Edition. s.l. : Addison Wesley, 2003. ISBN 0-321-19368-7.

Gardarin, Georges a Valduriez, Patrick. 1989. *Relational Databases and Knowledge Bases.* s.l. : Addison Wesley, 1989. ISBN 0-201-09955-1.

Gibbs, Dennis R. 2006. *Project Management with the IBM® Rational Unified Process®: Lessons from the Trenches.* s.l. : IBM Press, 2006. ISBN 0-321-33639-9.

Hasan, Jeffrey. 2004. *Expert Service-Oriented Architecture in C#: Using the Web Services Enhancements 2.0.* s.l. : Apress, 2004. ISBN 1-59059-390-1.

Havey, Mike. 2005. *Essential Business Process Modeling.* Sebastopol : O'Reilly, 2005. ISBN 0-596-00843-0.

Heimbinger, Dennis a McLeod, Dennis. 1985. A Federated Architecture for Information Management. ACM Transactions on Office Information Systems, júl 1985, Zv. 3, 3, s. 253-278.

Herbert, Andrew. 2004. Middleware? Muddleware! [ed.] Andrew Herbert a Karen Spärck Jones. *Computer Systems: Theory, Technology, and Applications.* New York : Springer-Verlag, 2004, s. 109-116.

Hohpe, Gregor a Woolf, Bobby. 2003. *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* s.l. : Addison Wesley, 2003. ISBN 0321200683.

Horch, John W. 2003. *Practical Guide to Software Quality Management.* 2. edícia. Boston : Artech House Publishers, 2003. ISBN 0-201-72915-6.

Hull, Robert. 1997. Managing Semantic Heterogenity in Databases: A Theoretical Perspective. [Online] 1997. [Dátum: 19. februára 2007.] http://www-db.research.bell-labs.com/user/hull/public_ftp/pods97-tutorial-slides.pdf.

Chandra, David, a iní. 2003. *Guidelines for Application Integration.* s.l. : Microsoft, 2003.

Chappell, David. 2005. Introducing Microsoft Windows Workflow Foundation: An Early Look. *MSDN (Microsoft Developer Network).* [Online] august 2005. [Dátum: 15. apríla 2007.] http://msdn2.microsoft.com/en-us/library/aa480215.aspx.

Chiu, Eric. 2002. *ebXML Simplified.* s.l. : Wiley, 2002.

Chrissis, Mery Beth, Konrad, Mike a Shrum, Sandy. 2003. *CMMI guidelines for process integration and product improvement.* s.l. : Addison Wesley, 2003. ISBN 0-321-15496-7.

IANA. 2006. MIME Media Types. [Online] 7. decembra 2006. [Dátum: 19. februára 2007.] http://www.iana.org/assignments/media-types/.

IEEE. 1998. *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements SpeciÞcations.* New York : IEEE, 1998. ISBN 0-7381-0332-2.

Informaster: An Information System. **Genesereth, Michael R., Keller, Arthur M. a Duschka, Oliver M. 1997.** 1997, Proceedings of the ACM SIGMOD Conference, May 1997, s. 539-542.

Inmon, William H. 2002. *Building the Data Warehouse.* 3. edícia. s.l. : Wiley, 2002.

Jackson, Tod, Majumdar, Rupa a Wheat, Steve. 2005. OpenEAI Methodology Version 1.0. [Online] 2005. [Dátum: 6. januára 2007.] http://www.openeai.org/cgi-bin/cvsweb.cgi/project/documentation/core/Methodology.rtf.

Kan, Stephen H. 2002. *Metrics and Models in Software Quality Engineering.* 2. edícia. s.l. : Addison Wesley, 2002. ISBN 0-201-72915-6.

Kleppe, Anneke, Warmer, Jos a Bast, Wim. 2003. *MDA Explained: The Model Driven Architecture: Practice and Promise.* s.l. : Addison Wesley, 2003. ISBN 0-321-19442-X.

Kočí, Vladimír. 2007. *Integrácia aplikácií pomocou podnikovej zbernice služieb (diplomová práca).* Bratislava : Fakulta matematiky, fyziky a informatiky UK, 2007. V tlači.

Kroll, Per a Kruchten, Philippe. 2003. *The Rational unified process made easy: a practitioner's guide to the RUP.* s.l. : Addison Wesley, 2003. ISBN 0-321-16609-4.

Lang, Micheal a Noggle, Brian J. 2002. Model-Driven Information Architecture. *The Data Administration Newsletter.* [Online] 2002. [Dátum: 25. februára 2007.] http://www.tdan.com/i020fe04.htm.

Linthicum, David S. 2003. *Next Generation Application Integration: From Simple Information to Web Services.* s.l. : Addison-Wesley, 2003. ISBN 0201844567.

Marks, Eric A. a Werrell, Mark J. 2003. *Executive's Guide to Web Services.* Hoboken : John Wiley, 2003. ISBN 0-471-26652-3 .

McGovern, James, a iní. 2004. *A practical guide to enterprise architecture.* New Jersey : Prentice Hall, 2004. ISBN 0-13-141275-2.

Meena, Hemant Kr., a iní. 2005. An Approach to Workflow Modeling and Analysis. 2005. Saniego : ACM, 2005. OOPSLA.

Mellor, Stephen J., a iní. 2004. *MDA Distilled: Principles of Model-Driven Architecture.* s.l. : Addison Wesley, 2004. ISBN 0-201-78891-8.

Microsoft. 2007. Word 2007: Rich Text Format (RTF) Specification, version 1.9. [Online] 9. januára 2007. [Dátum: 20. februára 2007.] http://www.microsoft.com/downloads/details.aspx?familyid=DD422B8D-FF06-4207-B476-6B5396A18A2B&displaylang=en.

Muller, Robert J. 1999. *Database design for smarties: using UML for data modeling.* San Francisco : Morgan Kaufmann, 1999. ISBN: 1-55860-515-0.

MV SR. 2006. Územné a správne usporiadanie Slovenskej republiky. *Sekcia verejnej správy MV SR.* [Online] MV SR, 2006. [Dátum: 25. marca 2007.] http://www.civil.gov.sk/p11/p11-01.shtm.

Newcomer, Eric. 2002. *Understanding Web Services: XML, WSDL, SOAP, and UDDI .* s.l. : Addison Wesley, 2002. ISBN 0201750813.

OMG. 2006. Business Process Modeling Notation Specification. [Online] február 2006. [Dátum: 12. marca 2007.] http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf.

—. 2007. Model Driven Architecture (MDA) FAQ. [Online] 2007. [Dátum: 5. marca 2007.] http://www.omg.org/mda/faq_mda.htm.

Persse, James R. PhD. 2006. *Process Improvement Essentials.* s.l. : O'Reilly, 2006. ISBN 0-596-10217-8.

Petri Nets: Properties, Analysis and Applications. **Murata, Tadao. 1989.** 4, apríl 1989, Proceedings of the IEEE, Zv. 77, s. 541-580.

Richards, Mark. 2006. Process Choreography and the Enterprise Service Bus. [ed.] Ford Neal. *No Fluff, Just Stuff Anthology: The 2006 Edition.* s.l. : The Pragmatic Programmers, 2006, 7, s. 104-116.

Russel, Nick a Hofstede, Arthur H.M. ter. 2006. *Workflow Control-Flow Patterns: A Revised View.* BPMcenter.org. 2006. BPM Center Report BPM-06-22.

Russel, Nick, Aalst, Wil M.P. van der a Hofstede, Arthur H.M. ter. 2006. *Exception Handling Patterns in Process-Aware Information Systems.* BPMcenter.org. 2006. BPM Center Report BPM-06-04.

Sharp, Alec a McDermott, Patrick. 2001. *Workflow modeling: tools for process impovement and application development.* Norwood : Artech House, 2001. ISBN 1-58053-021-4.

Silbershatz, Abraham, Korth, Henry F. a Sudarshan, S. 2002. *Database system concepts.* 4. vydanie. s.l. : McGraw-Hill, 2002. ISBN 978-0072554816.

Silverston, Len. 2001. *The Data Model Resource Book, Vol. 1: A Library of Universal Data Models for All Enterprises.* s.l. : Wiley, 2001. ISBN: 0471380237.

Simsion, Graeme, C a Witt, Graham, C. 2005. *Data modeling essentials.* Third Edition. San Francisco : Morgan Kaufmann, 2005. ISBN: 0126445516.

Slovenská pošta. 2007. PSČ obcí a ulíc. *PSČ obcí a ulíc.* [Online] 2007. [Dátum: 25. marca 2007.] http://www.slovenskaposta.sk/index.php?id=106.

Snodgrass, Richard T. 2000. *Developing Time-Oriented Database Applications in SQL.* San Francisco : Morgan Kaufmann, 2000. ISBN 1558604367.

Sommerville, Ian. 2004. *Software Engineering.* 7. edícia. Harlow : Addison Wesley, Pearson Education, 2004. ISBN 0-321-21026-3.

Störrle, Harald. 2004. Semantics of Control-Flow in UML 2.0 Activities. 2004. s.l. : IEEE, 2004, Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC'04).

Šešera, Ľubor, Mičovský, Aleš a Červeň, Juraj. *Architektúra softvérových systémov (Analytické dátové vzory).*

Štatistický úrad Slovenskej republiky. 2007. Databáza číselníkov. [Online] 2007. [Dátum: 25. marca 2007.] http://www.statistics.sk/wmetis/ciselniky/index.jsp.

Throwbridge, David, a iní. 2004. *Integration Patterns.* Redmond : Microsoft Press, 2004. ISBN 0-7356-1850-X.

Tian, Jeff. 2005. *Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement.* s.l. : IEEE Computer Society, John Wiley & Sons, 2005. ISBN 0-471-71345-7.

Univerzita Komenského. 2006. Príloha Štatútu UK č. 2. [Online] 2006. [Dátum: 19. februára 2007.] http://www.uniba.sk/fileadmin/user_upload/editors/subory/legislativa/Statut_UK_060502_priloha_2_1_.pdf.

Verzuh, Eric. 2005. *The fast forward MBA in project management.* 2. vydanie. Hoboken : John Wiley, 2005. ISBN 0-471-69284-0.

Völter, Markus, Kircher, Michael a Zdun, Uwe. 2004. *Remoting patterns: foundations of enterprise, internet and realtime distributed object middleware.* Hoboken : Wiley, 2004. ISBN 0-470-85662-9.

W3C. 2004. Extensible Markup Language (XML) 1.0 (Third Edition). [Online] 2004. [Dátum: 19. februára 2007.] http://www.w3.org/TR/2004/REC-xml-20040204/.

—. 1999. XSL Transformations (XSLT) Version 1.0. [Online] 1999. [Dátum: 19. februára 2007.] http://www.w3.org/TR/xslt.

Wiegers, Karl E. 2003. *Software Requirements.* Second Edition. Redmond : Microsoft Press, 2003. ISBN 0735618798.

Wysocki, Robert K. a McGary, Rudd. 2003. *Effective Project Management: Traditional, Adaptive, Extreme.* 3. edícia. s.l. : Wiley, 2003. ISBN 0-471-43221-0.

Zdun, Uwe a Dustdar, Schahram. 2006. Model-Driven and Pattern-Based Integrtation of Process-Driven SOA Models. 2006. Dagstuhl Seminar Proceedings.

Zetie, Carl. 2005. *"Model-Driven" Matters More Than MDA.* s.l. : Forrester Research, 2005.

Zhu, V., a iní. 2004. Model-driven business process integration and management: A case study with the Bank SinoPac regional service platform. september/november 2004, IBM Journal of Research and Development, Zv. 48, 5/6, s. 649-669.

Príloha A. Zoznam kandidátskych systémov

Poznámka: V práci neuvádzame mená správcov, ani ich kontakt (čísla kancelárií, telefónne čísla, e-maily).

Názov:	*Študent*		
Skratka:	Študent	**Umiestnenie:**	študijné oddelenia
Popis:	Správa študijnej agendy		
Správca:		**Kontakt:**	
Typ:	aplikácia pre koncových používateľov		
Dodávateľ:	FMFI UK		
Architektúra:	jednovrstvová, samostatné moduly riešia časti agendy		
Softvérové súčasti:	1. samostatné dátové súbory (DBF)		
	2. spustiteľné moduly desktopovej databázy Clipper		
Priorita:	*Vysoká* (predstavuje primárnu oblasť UK)		
Prepojenia:	1. CDO (nacionále študentov, ich študijný vzťah a informácie o preukazoch)		
	2. Centrálny register študentov		
	3.		
Možnosti integrácie:	dátová – DBF súbory		
Poznámky:	Na každej fakulty (študijnom oddelení) beží samostatná inštancia. Jednotlivé inštancie medzi sebou navzájom nekomunikujú.		

Názov:	*Internetová prezentácia UK*		
Skratka:	Web UK	**Umiestnenie:**	CIT UK
Popis:	Správa študijnej agendy		
Správca:		**Kontakt:**	
Typ:	webová aplikácia		
Dodávateľ:	open source komunita, obsah spravuje CIT UK		
Architektúra:	trojvrstvová s modulárnymi rozšíreniami		
Softvérové súčasti:	1. MySQL 5		
	2. CMS Typo3 s rozšíreniami		
Priorita:	*Stredná* (komunikačná brána medzi UK a okolitým svetom)		
Prepojenia:	(žiadne)		
Možnosti integrácie:	dátová – MySQL 5 „prezentačná“ – napísanie rozšírenia, vloženie vlastnej aplikácie s rovnakým dizajnom		
Poznámky:			

Názov:	*Centrálna databáza osôb*

Skratka:	CDO	**Umiestnenie:**	CIT UK
Popis:	Evidencia študentov, zamestnancov a ich preukazov		
Správca:		**Kontakt:**	
Typ:	Integračná služba s webovou aplikáciou na správu preukazov		
Dodávateľ:	UK		
Architektúra:	trojvrstvová		
Softvérové súčasti:	1 Java aplikačný server (Sun), JSP, EJB		
	2. MS SQL Server 2000, Hibernate		
Priorita:	*Stredná*		
Prepojenia:	1. importy údajov o osobách (Študent)		
	2. export do Ankety FMFI (čísla preukazov, informácie o osobách a súčastiach, kde pôsobia, druh štúdia)		
Možnosti integrácie:	dátová – MS SQL Server aplikačná – J2EE EJB webové služby		
Poznámky:			

Názov:	*Anketa FMFI UK*		
Skratka:	Anketa	**Umiestnenie:**	ŠKAS FMFI UK
Popis:	Správa študijnej ankety		
Správca:		**Kontakt:**	
Typ:	webová aplikácia		
Dodávateľ:	študenti FMFI UK		
Architektúra:	dvojvrstvová		
Softvérové súčasti:	1. PHP		
	2. MySQL		
Priorita:	*Stredná* ()		
Prepojenia:	1. CDO – export čísel preukazov študentov vo formáte CSV		
	2.		
	3.		
Možnosti integrácie:	dátová – priamy prístup do databázy		
Poznámky:	Prístupné na anketa.uniba.sk V súčasnosti používa FMFI UK a FTVŠ UK		

Príloha B. Používateľské popisy systémov (scenáre)

Názov:	*Hodnotenie učiteľa*		
Úroveň:		**ID:**	Anketa 2/1
Systém:	Anketa 2		
Iniciátor:	študent		
Vstupná podmienka:	študent je prihlásený		
Hlavný scenár:	1. Študent si vyberie zo zoznamu učiteľa, ktorého chce hodnotiť. 2. Študent vyberie predmet, ktorý učiteľ vyučoval. 3. Študent môže ohodnotiť známkami A-Fx jeho celkový výkon, starostlivosť o študentov, aký im dal priestor a využívanie nových postupov. 4. Študent môže pridať komentár s voľným textom k učiteľovi.		
Rozšírenia:	1a: Učiteľ sa nenachádza v zozname učiteľov. 1. Študent môže pridať učiteľa vpísaním jeho mena. 2a: Učiteľ nemá priradený predmet, ktorý vyučoval. 1. Študent ručne priradí predmet zo zoznamu predmetov učiteľovi.		

Názov:	*Hodnotenie predmetu*		
Úroveň:		**ID:**	Anketa 2/2
Systém:	Anketa 2		
Iniciátor:	študent		
Vstupná podmienka:	študent je prihlásený		
Hlavný scenár:	1. Študent si vyberie predmet zo zoznamu predmetov, ktoré navštevoval. 2. Študent zahlasuje, či by predmet odporučil iným študentom (áno/nie). 3. Študent zhodnotí, ako často predmet počas semestra navštevoval (klasifikovaná stupnica).		
Rozšírenia:	1a: Predmet sa nenachádza v zozname predmetov, ktoré navštevoval. 1. Študent môže pridať predmet ručne zo zoznamu všetkých predmetov.		

Názov:	*Reakcia učiteľa na hodnotenie*		
Úroveň:		**ID:**	Anketa 2/9
Systém:	Anketa 2		
Iniciátor:	učiteľ		
Vstupná podmienka:	učiteľ je prihlásený		

Hlavný scenár:	1. Učiteľ si vyberie predmet zo zoznamu predmetov, ktoré učil (a tých, ktoré mu ručne priradili študenti) a na hodnotenie ktorého chce reagovať. 2. Učiteľ vloží do systému reakciu vo forme voľného textu.
Rozšírenia:	

Názov:	*Zobrazenie informácií o študijnom programe*		
Úroveň:		**ID:**	Web 1/5
Systém:	Web		
Iniciátor:	návštevník		
Vstupná podmienka:			
Hlavný scenár:	1. Návštevník si zo zoznamu odborov vyberie odbor, do ktorého študijný program patrí. Okrem odborov sa zobrazí špeciálna kategória celofakultných predmetov, ktoré nepatria v rámci hierarchie do žiadneho odboru. 2. Návštevník si vyberie konkrétny študijný program. 3. Systém zobrazí používateľovi informácie o programe, kde musia byť uvedené aspoň informácie o garantovi a predmetoch, ktoré sa doň spadajú roztriedených na povinné, povinne voliteľné a voliteľné.		
Rozšírenia:	2a. Ak si návštevník vybral v bode 1 celofakultné predmety, tento bod sa vynecháva.		

Názov:	*Zobrazenie informácií o predmete*		
Úroveň:		**ID:**	Web 1/6
Systém:	Web		
Iniciátor:	návštevník		
Vstupná podmienka:			
Hlavný scenár:	1. Návštevník si vyberie predmet, buď zo samostatného zoznamu predmetov alebo zoznamu predmetov spadajúcich po vybraný študijný program (pozri scenár Web 1/5). 2. Systém zobrazí informácie o predmete, a to aspoň jeho plný názov, podiel priebežného hodnotenia k záverečnému, cieľ a osnovu predmetu, vyučujúcich, jazyk, v ktorom sa vyučuje a odporúčanú literatúru.		
Rozšírenia:	2a. Ak si návštevník vybral v bode 1 celofakultné predmety, tento bod sa vynecháva.		

Príloha C. Podnikové objekty

Anketa

Názov	Zdroj	Konzument	Odchýlky	Poznámky
študent		Á		bez doktorandov
učiteľ		Á		
zapísaný predmet		Á	iba názov a formy výučbu s rozsahom	
študijný program		Á		
hodnotenie učiteľa	Á			
hodnotenie št. programu	Á			

Web

Názov	Zdroj	Konzument	Odchýlky	Poznámky
študijný odbor		Á		
študijný program		Á		
predmet		Á		
organizačná štruktúra		Á		
učiteľ		Á		

Príloha D. XML správy

```
<?xml version="1.0" encoding="utf-8"?>

<courses>

	<course name="informatika" branch1="9.2.2" branch2="9.4.5" degree="1"
standardLength="5" studyForm="1">

		<subjectGroups>

			<subjectGroup name="Povinné predmety" optionality="5">

				<subject id="43" rcmdYear="3" prereq="368*458"
active="1" term="1" examRatio="40" credits="6" language="sk">

				</subject>

				<subject>
					...
				</subject>

			</subjectGroup>

			<subjectGroup name="Výberové predmety" optionality="4">
				...
			</subjectGroup>

			<subjectGroup name="Povinne voliteľné bloky B1..B4"
optionality="1" optParm="1">

				<subjectGroup name="Súbor B1" optionality="5">
					...
				</subjectGroup>

				<subjectGroup name="Súbor B2" optionality="5">
					...
				</subjectGroup>

				<subjectGroup name="Súbor B3" optionality="5">
					...
				</subjectGroup>

				<subjectGroup name="Súbor B4" optionality="5">
					...
				</subjectGroup>


			</subjectGroup>

			<subjectGroup name="Bloky výberových predmetov C1..C4"
optionality="4">

				...


			</subjectGroup>
```

```
                    <subjectGroup name="Predmety štátnej skúšky"
optionality="99">
                                ...
                    </subjectGroup>

            </subjectGroups>

      </course>

      <course>

            ...

      </course>

      </courses>
```

Obrázok 7-1 Príklad tela správy SubjectsHierarchy

```
<?xml version="1.0" encoding="utf-8"?>

<subjects>

  <subject id="43" rcmdYear="3" prereq="368*458" active="1" term="1"
examRatio="40" credits="6" language="sk" departmentId="12" code="M-INAA-022"
name="Matematická analýza 1" englishName="Mathematical Analysis 1">
    <goal>Základné pojmy a nástroje diferenciálneho počtu reálnych funkcií
jednej premennej.</goal>
    <synopsis>
      Stručný historický prehľad. Limita funkcie a postupnosti a základné
vety o limitách. Cauchyho-Bolyanovo kritérium a existencia konvergentných
podpostupností. Spojitosť funkcie v bode a na množine, vlastnosti spojitých
funkcií na intervaloch a na kompaktných množinách. Derivácia funkcie,
základné vety o výpočte derivácií, derivácia inverznej a zloženej funkcie.
Vety o strednej hodnote diferenciálneho počtu, vyšetrovanie priebehu funkcií.
L´Hospitalovo pravidlo. Taylorove polynómy.
    </synopsis>

    <liteature>
      <literatureItem>
        Neubrunnn T., Vencko J.: Matematická analýza 1, skriptum UK,
Bratislava 1989
      </literatureItem>
      <literatureItem>
        Kubáček Z., Valášek J.: Cvičenia z matematickej analýzy 1, skriptum
UK, Bratislava 1989
      </literatureItem>
      <literatureItem>
        Brabec J., Martan F., Rozenský Z.: Matematická analýza 1, SNTL-Alfa,
Praha 1985
      </literatureItem>
    </liteature>

    <teachers>
      <teacherRef id="589" />
      <teacherRef id="6984" />
      <teacherRef id="145" />
```

```
    </teachers>

    <formExtents>
      <formExtent formId="1" extent="3" timeUnit="1"/>
      <formExtent formId="2" extent="1" timeUnit="1"/>
    </formExtents>

  </subject>

  <subject>
    ...
  </subject>

</subjects>
```

Obrázok 7-2 Príklad tela správy SubjectsInfo

```
<?xml version="1.0" encoding="utf-8"?>

<branches>

  <branch branchId="58" name="Informatika" deptCode="9.2.1">

    <courses>

      <course courseId="698" name="informatika" branch1="9.2.2"
branch2="9.4.5" degree="1" standardLength="5" studyForm="1" />

      <course ...="" />

    </courses>

    <branch ...="">
      ...
    </branch>

  </branch>


  </branches>
```

Obrázok 7-3 Príklad tela správy StudyCourses

Príloha E. Fyzická štruktúra ADM

Legenda popisných tabuliek:

- PK = primárny kľúč,
- FK = cudzí kľúč,
- C10 = dátový typ reťazec s dĺžkou 10,
- vC10 =dátový typ reťazec s variabilnou dĺžkou do 10,
- INT = dátový typ celé číslo (integer),
- sINT = dátový typ malé celé číslo (small integer)
- DATETIME = dátum a čas,
- D = číslo s pevným počtom desatinných miest (decimal).

Courses (Študijné programy)

Názov stĺpca	Popis	Typ/enumerované hodnoty
CourseId	PK	INT
Code	Mnemotechnická skratka programu (napr. INF pre informatiku)	C10
Name	Názov	vC100
Degree	Stupeň štúdia	sINT
StudyForm	Študijná forma	FK do CT_StudyForms
StandardLength	Štandardná dĺžka štúdia v rokoch	sINT
Title	Udeľovaný akademický titul	FK do CT_Titles
Branch1Id	Primárny študijný odbor	FK do Branches
Branch2Id	Sekundárny študijný odbor	FK do Branches
RootSubjGroupId	Koreňová (najvyššia) skupina predmetov	FK do SubjectGroups

Branches (Študijné odbory)

Názov stĺpca	Popis	Typ/enumerované hodnoty
BranchId	PK	INT
Name	Názov odboru	vC100
DeptCode	Rezortný kód odboru. Definovaný číselníkom odborov z MŠ SR.[26] Formát kódu je A.B.C, kde A je číslo skupiny	C8

[26] Sústava študijných odborov vysokoškolského vzdelávania Slovenskej republiky.

	študijných odborov, B je číslo podskupiny a C je číslo odboru v skupine.	

SubjectGroups (Skupiny predmetov)

Názov stĺpca	Popis	Typ/enumerované hodnoty
SubjGroupId	PK	INT
Name	Názov skupiny	vC100
ParentSubjGroupId	Rodičovská skupina predmetov	FK do SubjectGroups
Optionality	Podmienka na výber v skupine.	FK do CT_SubjectGroupOptionality
OptParam	Celočíselný parameter podmienky na výber v skupine.	INT

SubjectRealizations (Nasadenia predmetov)

Názov stĺpca	Popis	Typ/enumerované hodnoty
SubjRealId	PK	INT
VariantId	Variant, ktorého nasadením je táto entita	FK do SubjectVariants
RcmdYears	Zoznam čiarkou oddelených odporúčaných rokov štúdia bez bielych znakov vo vzostupnom poradí.	vC20 Napr.: 3,4,5
Prereq	Dobre uzátvorkovaný výraz ID predmetov s operátormi + pre logický súčet a * pre logický súčin. Po dosadení logických konštánt true/false za ID predmetov v prípadoch, keď študent predmet úspešne absolvoval, resp. neabsolvoval, sa získa logický výraz, vyhodnotením ktorého je hodnota, či spĺňa podmienky na zápis predmetu vzhľadom na absolvovanie podmieňujúcich predmetov.	vC50, elementárne operandy FK do Subjects
RcmdPrereq	Syntax a význam rovnaký ako Prereq, len s tým rozdielom, že výsledná	ako Prereq

	hodnota výrazu má iba odporúčací charakter.	
SubjGroupId	Skupina predmetov, v ktorej je predmet nasadený.	FK do SubjectGroups.

SubjectVariants (Varianty predmetov)

Názov stĺpca	Popis	Typ/enumerované hodnoty
VariantId	PK	INT
SubjectId	Predmet, ktorého variant je táto entita.	FK do Subjects
Active	1, ak je predmet aktívny (vyučuje sa), 0, ak je suspendovaný (dočasne sa neučí).	BIT
Term	Semester výučby.	FK do CT_Terms
ExamRatio	Pomerná časť hodnotenia na skúške k priebežnému hodnoteniu.	D Napr. 60 znamená, že pomer priebežného hodnotenia ku skúške je 40/60.
Credits	Počet kreditov, ktoré získa študent úspešným absolvovaním predmetu.	INT
Language	Jazyk výučby predmetu.	C2, FK do CT_Languages. Kód je z číselníka jazykov ISO 639-1.
DepartmentId	Pracovisko, ktoré je za predmet zodpovedné.	FK do Departments.

FormExtentsVariants (Rozsah a forma výučby)

Tabuľka reprezentujúca many-to-many vzťah medzi jednotlivými formami výučby a ich rozsahom a variantmi predmetov.

Názov stĺpca	Popis	Typ/enumerované hodnoty
FormId	PK, forma výučby.	FK do CT_TeachingForms
VariantId	PK, variant predmetu.	FK do SubjectVariants
Extent	Časový rozsah formy výučby.	D
TimeUnit	Časová jednotka rozsahu.	FK do CT_TimeUnits

TeachersVariants (Vyučujúci na predmete)

Tabuľka reprezentujúca many-to-many vzťah medzi variantom predmetu a vyučujúcimi.

Názov stĺpca	Popis	Typ/enumerované hodnoty
VariantId	ID variantu predmetu.	FK do SubjectVariants

TeacherId	Vyučujúci.	FK do Employees

Subjects (Predmety)

Názov stĺpca	Popis	Typ/enumerované hodnoty
SubjectId	PK	INT
Code	Fakultný kód predmetu (doteraz)	C10, formát A-BBBB-NNN
Name	názov	vC100
EnglishName	anglický názov	vC100
Goal	cieľ predmetu	vC1000
Synopsis	osnova predmetu (sylabus)	vC10000
Literature	odporúčaná literatúra k predmetu	vC10000
StudentId	kód predmetu v systéme Študent	C10
DepartmentId	katedra, ktorá predmet zabezpečuje (gestoruje)	FK do Departments
FacultyId	fakulta, na ktorej sa predmet vyučuje	FK do Faculties

CourseTerms (Etapy študijného programu)

Názov stĺpca	Popis	Typ/enumerované hodnoty
CourseTermId	PK	INT
CourseId	Študijný program, etapa ktorého je táto entita	FK do Courses
SeqNumber	Poradové číslo etapy	sINT
PrevTermId	Predchádzajúca etapa	FK do CourseTerms
MinCreditsTerm	Minimálny počet kreditov, ktoré musí študent získať počas prvej časti (zimného semestra) etapy na úspešné absolvovanie etapy.	sINT
MinCredits	Minimálny počet kreditov, ktoré musí študent získať počet celej etapy na jej úspešné absolvovanie.	sINT
DepartmentId	Pracovisko, na ktorom etapa prebieha.	FK do Departments

StudyDuties (Študijné povinnosti)

Názov stĺpca	Popis	Typ/enumerované hodnoty

StudyDutyId	PK	INT
StudentId	študent, ktorý vlastní túto povinnosť	FK do Students
EnrolmentTerm	semester zápisu povinnosti	
SubjectRealizationId	nasadenie predmetu, ktorý vytvára študijnú povinnosť	FK do SubjectRealizations
FinalGrade	výsledná známka, ktorú študent dostal za túto povinnosť	

Exams (Termíny skúšok/Hodnotenia študijných povinností)

Názov stĺpca	Popis	Typ/enumerované hodnoty
ExamId	PK	INT
StudyDutyId	študijná povinnosť, z ktorej sa robí skúška	FK do StudyDuties
Grade	výsledná známka na termíne	INT
DateTaken	dátum vykonania termínu	DATETIME
ExamType	druh termínu (riadny, opravný, mimoriadny)	číselník

CoursesFaculties (Študijné programy, ktoré sa učia na fakulte)

Názov stĺpca	Popis	Typ/enumerované hodnoty
CourseId	PK	FK do Courses
FacultyId	PK	FK do Faculties

Štandardné číselníky

Languages (Jazyky)

Použiť číselník jazykov ISO 639-1. Typ kódu jazyka je CHAR(2).

SubjectGroupOptionality (Voliteľnosť skupiny predmetov)

kód	slovenský text	anglický text
1	minimálny počet predmetov	minimum number of subjects
2	počet kreditov	number of credits
3	počet predmetov (presne)	number of subjects (exact)
4	žiadna	arbitrarily
5	všetky	all
99	neudaná	N/A

Terms (Časti akademického roka, semestre)

kód	slovenský text	anglický text
1	zimný semester	winter term (semester)
2	letný semester	summer term (semester)
3	oba semestre	both terms (semesters)

Titles (Tituly)

P = pred menom (prefix)

S = za menom (sufix)

slovenský text	anglický text	Pred menom/za menom
Bc.		P
Mgr.		P
RNDr.		P
PhDr.		P
MUDr.		P
MVDr.		P
JUDr.		P
Ing.		P
PhD.		S
Doc.		P
Prof.		P
CSc.		S
DrSc.		S
Dr.		P
Ing. arch.		P
PaedDr.		P
PharmDr.		P
PhLic.		P
ThDr.		P
ThLic.		P
B.A.		S
M.A.		S
akad. soch.		S
prom. mat.		S
M.D.		S
M.Sc.E.E.		S

StudyForms (Študijné formy)

slovenský text	anglický text
denná	present
externá	extramural

TeachingForms (Formy výučby)

slovenský text	anglický text
prednáška	lecture
cvičenie	practical session
seminár	seminar

kurz	course
iná	other form
prax	practice
laboratórne cvičenie	laboratory practice
diplomová práca	diploma thesis
odborné sústredenie	special intense course

TeachingStatus (Stav výučby)

slovenský text	anglický text
vyučuje sa	active
suspendovaný	suspended

TimeUnits (Časové jednotky)

slovenský text	anglický text
vyučovacia hodina	teaching hour
deň	day
mesiac	month
hodina	hour

Faculties (Fakulty)

Číselník fakúlt je prítomný kvôli identifikácii fakúlt, ktorým patria entity zo systému Študent. Každý výskyt identifikátora (primárne kľúča) zo Študenta predchádza kód z číselníka fakúlt. V súčasnosti si vystačíme iba s fakultami na UK, prvá číslica kódu reprezentuje univerzitu (v našom prípade iba 1 = UK). Poradové čísla (nasledujúce dve číslice) sú zhodné s poradovým číslom fakulty uvedenej v (Univerzita Komenského, 2006).

kód	slovenský text	anglický text
101	Lekárska fakulta	Faculty of Medicine
102	Právnická fakulta	Faculty of Law
103	Filozofická fakulta	Faculty of Philosophy
104	Prírodovedecká fakulta	Faculty of Natural Sciences
105	Pedagogická fakulta	Faculty of Education
106	Farmaceutická fakulta	Faculty of Pharmacy
107	Fakulta telesnej výchovy a športu	Faculty of Physical Education and Sports
108	Jesseniova lekárska fakulta	Jessenius Faculty of Medicine
109	Fakulta matematiky, fyziky a informatiky	Faculty of Mathematic, Physics and Informatics
110	Rímskokatolícka cyrilometodská bohoslovecká fakulta	Faculty of Roman Catholic Theology
111	Evanjelická bohoslovecká fakulta	Faculty of Evangelical Theology

112	Fakulta managementu	Faculty of Management
113	Fakulta sociálnych a ekonomických vied	Faculty of Social and Economic Sciences

Príloha F. Dátové zdroje pre ADM/STIF

Dátové zdroje zo Systému Študent

KTP – Číselník pracovísk

Pôvodne tento číselník slúžil ako evidencia katedier, ktoré zabezpečujú výučbu jednotlivých predmetov a zamestnávajú učiteľov. Keďže jednak si študenti môžu zapisovať predmety z cudzích fakúlt a jednak na fakultách často vypomáhajú učitelia z iných fakúlt (dokonca aj z iných vysokých škôl a inštitúcií), postupne sa číselník rozrástol i o tieto subjekty. Napr. v databáze FMFI UK sa nachádzajú takmer všetky katedry Prírodovedeckej fakulty UK, ale i položky „Elektrotechnická fakulta STU" či „Katedra geodézie STU". Z týchto príkladov vidno, že nie je ani jasná granularita, na ktorej sa cudzie pracoviská v našich databázach vedú.

Názov stĺpca	Popis	Typ/enumerované hodnoty
kodkp	Kód pracoviska	
nazov	Názov pracoviska	

UCP – Zoznam učiteľov

Primárnou úlohou tejto tabuľky je udržiavať zoznam ľudí, ktorých je možné priradiť k predmetu; inak povedané, všetci, ktorí majú pedagogický výkon na nejakom predmete, sa musia nachádzať v zozname. Z tohto prvoradého cieľa vychádza, že rozhodnutie, či sa niekto objaví v UCP je dané tým, či existuje alebo v minulosti existoval ľubovoľný predmet, ktorý učil. Treba si uvedomiť, že z toho vyplývajú niektoré podstatné závery. V prvom rade na fakultách neučia len vyučujúci s pevným pracovno-právnym vzťahom k fakulte, ale niekedy sa stáva, že vyučujúci pochádza z inej fakulty či dokonca z inej univerzity a vyrovnania sa robia rozličnými dohodami (napr. vzájomné pedagogické aktivity), následkom čoho sú v UCP objavujú mená, ktoré nikdy neboli vedené v personálnej databáze fakulty.

Ďalším problémom je rozličná údržba týchto dát, vzhľadom na to, že databáza sa prevádzkuje lokálne na každej fakulte zvlášť a je udržiavaná ručne. Keďže nové predmety prichádzajú do databázy priebežne a takisto priebežne sa upravuje plné meno ľudí (nový titul pred menom alebo za menom, vydaj a pod.), stáva sa, že tá istá osoba je fyzicky vedená viackrát v databáze (napr. pod rodným menom i menom súčasným, bez titulu i s ním).

Čo sa týka úplného mena, je s ním ďalší problém: malo by byť vo formáte RNDr. Ján Mrkva, PhD., ale takmer rovnako veľa je záznamov identických s príslušných krátkym menom (obe sú rádovo po stovkách). Zvyšných niekoľko desiatok záznamov je neúplných, zvyčajne je to iba priezvisko, príp. krátke meno s titulmi.

Názov stĺpca	Popis	Typ/enumerované hodnoty
kodup	Kód (ID) učiteľa	
imeno	Krátke meno (vo formáte Mrkva J. alebo Mrkva Ja.),	

	bez titulov	
umeno	Úplné meno	
typpv	Typ pracovného vzťahu	(?, D, E, Z) D = domáci, interný E = externý Z = zahraničný ? = tzv. fantóm[27]
przar	Pracovné zaradenie[28]	(A, L, P, S, U, V) A = doktorand („ašpirant“) L = lektor S = študent U = učiteľ V = vedecký pracovník I = iný
kodkp	Kód pracoviska, na ktoré je zaradený	FK do KTP

SPG – Zoznam študijných programov

Názov stĺpca	Popis	Typ/enumerované hodnoty
idspgr	Kód (ID) študijného programu	C8
nazov	Skrátený názov[29]	
rlspg	Realizácia študijného programu	(1, 2, 3) 1 = iba jeden odbor, 2 = dva rovnocenné študijné odbory, 3 = jeden primárny, druhý sekundárny
sodb1	Primárny študijný odbor	
sodb2	Sekundárny študijný odbor	
stups	Stupeň štúdia	(1, 2, 3, 4) 1 = prvý, 2 = druhý, 3 = spojený prvý a druhý, 4 = doktorandský
druhs	Druh štúdia	(B, M, S, D, P) B = bakalárske, M = magisterské,

[27] Už nie je celkom jasný význam tohto typu pracovného vzťahu (bolo to zavedené pre niekoľkými rokmi), pravdepodobne ide o učiteľov, ktorí odišli z fakulty, príp. zomreli a stále sú vedení v evidencii. V databáze FMFI UK je asi polovica záznamov označená týmto kódom.

[28] Tento atribút bol po prvýkrát zavedený na podnet FiF UK, ktorá má veľké množstvo rozličných zahraničných lektorov.

[29] Interný názov. Oficiálny názov (názov z akreditačného spisu), ktorý sa používa hlavne v oficiálnych tlačových výstupoch, sa nachádza v osobitnom číselníku.

		S = spojené, D = doktorské, I = inžinierske, P = doktorandské (PhD.)
stdls	Štandardná dĺžka štúdia v rokoch	INT, 1..6
forms	Forma štúdia	(e, d) d – denná, e – externá[30]
pkrus	Minimálny počet nazbieraných kreditov na úspešné ukončenie programu	INT typické hodnoty 0, 120, 180, 300; vždy ide o násobok 60
titul	Udeľovaný akademický titul	C typické hodnoty Bc., PhD., Mgr., MUDr., MVDr.
sysst	Systém štúdia	(sr, sb, ka, kp) sr = „statický ročníkový“ – jednotná ročníková organizácia štúdia, sb = „statický blokový“ – ročníková organizácia štúdia po blokoch predmetov, ka = „kreditový absolútny“ – možnosť individuálneho tempa štúdia a kontrola na báze počtu nazbieraných kreditov, podmienky sa ich počet sú špecifikované absolútne („na postup do ďalšieho ročníka treba 40 kreditov“), kr = ako ka, ale podmienky sú špecifikované relatívne („na postup treba získať aspoň 80% kreditov“)[31]
blokp1, blokp2, blokp3, blokp4, blokp5	Bloky predmetov	každý C5 FK do BPR, môže sa použiť zástupný znak ? pre práve jedno písmeno

[30] Do budúcnosti sa uvažuje i o dištančnej forme štúdia.
[31] Niekoľko rokov sa používal takýto systém na FaF UK.

ESP – Etapy študijného programu

Študijný program sa podľa platných noriem delí na akademické roky. Pod názvom etapy študijného programu sú evidované v tabuľke ESP. Akademický rok sa ďalej delí na dva semestre alebo tri semestre. Všetky fakulty UK delia rok iba na dva semestre.

Názov stĺpca	Popis	Typ/enumerované hodnoty
idesp	Kód (ID) etapy študijného programu	
idspg	ID študijného programu tejto etapy	FK do SPG
etapa	poradové číslo etapy	INT
lokal	lokalita, kde vyučovanie v danej etape prebieha[32]	FK do číselníka lokalít.
petap	ID predchádzajúcej etapy[33]	FK na idesp
pksem	minimálny počet kreditov v zimnom semestri na úspešné absolvovanie etapy	
pkros	minimálny počet kreditov na úspešné absolvovanie etapy	pksem ≤ pkros
pkmts	minimálne tempo štúdia; najmenší možný medzisúčet dosiahnutých kreditov do tejto etapy, aby študent nebol vylúčený pre neprospech	
idrso	ID predvolenej referentky študijného oddelenia[34]	FK do číselníka RSO
pstus	počet študijných skupín	
lstus	maximálny počet študentov v jednej študijnej skupine	

BPR – Bloky predmetov

Jednoduchý číselník blokov predmetov. Blok predmetov je logické združenie viacerých predmetov do množiny, ktoré slúži na triedenie a organizáciu predmetov. Bloky v tomto číselníku stoja v hierarchii predmetov najvyššie.

[32] Jednotlivé etapy môžu prebiehať na rozličných pracoviskách a v externých partnerských organizáciách (napr. SAV). Je možné aj to, že jedna etapa (jedného študijného programu s daným poradovým číslom) sa vyučuje vo viacerých lokalitách.

[33] Používa sa hlavne na výber ďalšej etapy na obrazovkách. V prípade, že etapa je prvá svojím poradovým číslom, môže tu byť uvedené ID prijímacieho konania, pod ktorým sa vedú študenti pred oficiálnym nástup do prvej etapy svojho študijného programu.

[34] Študijné referentky sú priraďované priamo študentom. Záznam v tabuľke ESP hovorí, ktorá referentka sa automaticky priradí študentovi pri zápise danej etapy, čo zjednodušuje prácu, keďže väčšina študentov v tej istej etape má patrí pod rovnaké referentky.

Názov stĺpca	Popis	Typ/enumerované hodnoty
blokp	Kód (ID) bloku predmetov	
nazov	Názov	

SBP – Subbloky predmetov

Zoznam blokov predmetov (pozri tabuľku BPR) stojacich v hierarchii predmetov na druhej a nižších úrovniach (tzv. subblokov alebo podblokov). Má zložený primárny kľúč a sémantika informácií uložených v ňom závisí od poradia riadkov.

Názov stĺpca	Popis	Typ/enumerované hodnoty
blokp	ID najvyššieho bloku predmetov, ktorému tento podblok prislúcha	FK do tabuľky BPR
idsbp	Poradové číslo podbloku.	Samo o sebe nemusí byť jedinečné v rámci všetkých podblokov. Primárny kľúč tejto tabuľky je treba brať ako zložený kľúč (blokp, idsbp).
atsbp	Atribúty podbloku	# = podblok bez ďalších podblokov % = podblok s vnorenými blokmi. Pri vzostupnom utriedení podľa stĺpca idsbp pri danej hodnote atribútu blokp dostaneme riadky reprezentujúce vnorené bloky. Ich počet je daný hodnotou v stĺpci psbk.
psbk	Počet vnorených blokov.	
metod	Metóda výberu (obligatórnosť)	P = povinný V0 = výberový BK = výberový na základe počtu kreditov Ax = alternatívny výber (práve jeden, exkluzívne alebo) S = štátnicové predmety
kvnat	Kvantitatívna podmienka na metódu výberu	Ak je hodnota v stĺpci metod rovná BK, hodnota reprezentuje počet kreditov, ktoré si musí študent zapísať na úspešné absolvovanie

		tohto bloku.

VEP – Nasadenia predmetov

Názov stĺpca	Popis	Typ/enumerované hodnoty
idsbp	ID študijného bloku, v ktorom je predmet nasadený	FK do tabuľky SBP.
uipmt	Kód variantu predmetu	Pozri tabuľku RLP, ako sa konštruuje.
orost	Odporúčaný rok štúdia	
idprq	Zoznam prerekvizít (podmieňujúcich predmetov)	Zoznam ID predmetov oddelený logickými operátormi (bez bielych znakov, žiadne zátvorky): . = logický súčin (AND), + = logický súčet (OR). V súčasnosti sa používa výlučne logický súčin, aj keď je žiaduci i súčet. Príklad: 37.0g.0x znamená, že na zapísanie tohto predmetu je potrebné úspešne absolvovať predmety s ID 37, 0g a 0x.
atprq	Atribúty zoznamu prerekvizít	Určuje, či prerekvizita je povinná alebo odporúčaná. Vyjadrený ako zoznam znakov ! (povinná) a medzier (odporúčaná), ktorých pozície korešpondujú s ID predmetov uvedenými v stĺpci idprq.

RLP – Variant (realizácia) predmetu

Názov stĺpca	Popis	Typ/enumerované hodnoty
uipmt	ID variantu predmetu	C3 Prvé dva znaky sú ID predmetu (FK do PRM), tretí znak je poradové číslo variantu.
stats	Status predmetu	(a, s) Príznak, či sa predmet v danom akademickom vyučuje.

		a = aktívny, s = suspendovaný (neučí sa)
semes	Semester výučby predmetu	(A, Z, L) Z = zimný, L = letný, A = zimný aj letný
poctv	Počet týždňov výučby	INT Zaviedlo sa kvôli výpočtu štatistiky pedagogického zaťaženia.
forov	Forma výučby.	Zoznam foriem výučby s časom a časovými jednotkami oddelenými medzerou. Každá jednotka („forma výučby“) je typu CIntC, kde prvý znak je typu (P, C, S, K, I, L, D, X) s významom: P = prednáška, C = cvičenie, S = seminár, K = kurz, I = iná, L = laboratórne cvičenia, O = odborné sústredenie, D = diplomová práca, X = prax. Číslo za týmto znakom udáva, koľko týchto častí výučby beží, implicitne sú to hodiny za týždeň. Posledný znak je časová jednotka, ktorá môže byť: NULL = hodiny za týždeň, d = počet dní, t = počet týždňov.
phod	Podiel priebežného hodnotenia k hodnoteniu na skúške.	Formát i/i, kde i je typu INT.
hkred	Počet ECTS kreditov	
gespr	ID gestorského pracoviska (zodpovedného za predmet)	FK do číselníka KTP.
ucitl	Vyučujúci	Zoznam položiek typu C2, ktoré sú FK do tabuľky UCP.

limit	Maximálny počet študentov, ktorí si môžu predmet zapísať	Väčšinou sa nepoužíva a má hodnotu 0 (s významom neobmedzene).
pozna	Kód poznámky, ktorá sa tlačí v ročenke fakulty.	Poznámka sa pridáva ručne v DTP programe pri príprave ročenky. Kódy sa vytvárajú tiež ručne.

PRM – Predmety

Názov stĺpca	Popis	Typ/enumerované hodnoty
idprdmt	ID predmetu	C2
nazov	Názov predmetu	C100
anglnaz	Anglický názov predmetu	C100
kod	Kód predmetu	C10

Dátové zdroje – Informačné listy predmetov

Informačné listy predmetov obsahujú pre nás niekoľko dôležitých informácií, ktoré sa nenachádzajú v žiadnom inom elektronickom zdroji informačného systému UK. Ide predovšetkým o:

- priebežné hodnotenie popísané voľným textom (napr. „hodnotené domáce úlohy, testy v priebehu semestra“),
- záverečné hodnotenie popísané voľným textom (napr. „záverečný test, skúška“),
- cieľ predmetu (napr. „Prednáška poskytne študentom základy matematickej analýzy nevyhnutné pre štúdium informatiky. Študenti si zároveň osvoja matematickú kultúru, notáciu, spôsob myslenia a vyjadrovania.“),
- stručná osnova predmetu,
- zoznam literatúry, z ktorej predmet čerpá a odporúčanej na štúdium.

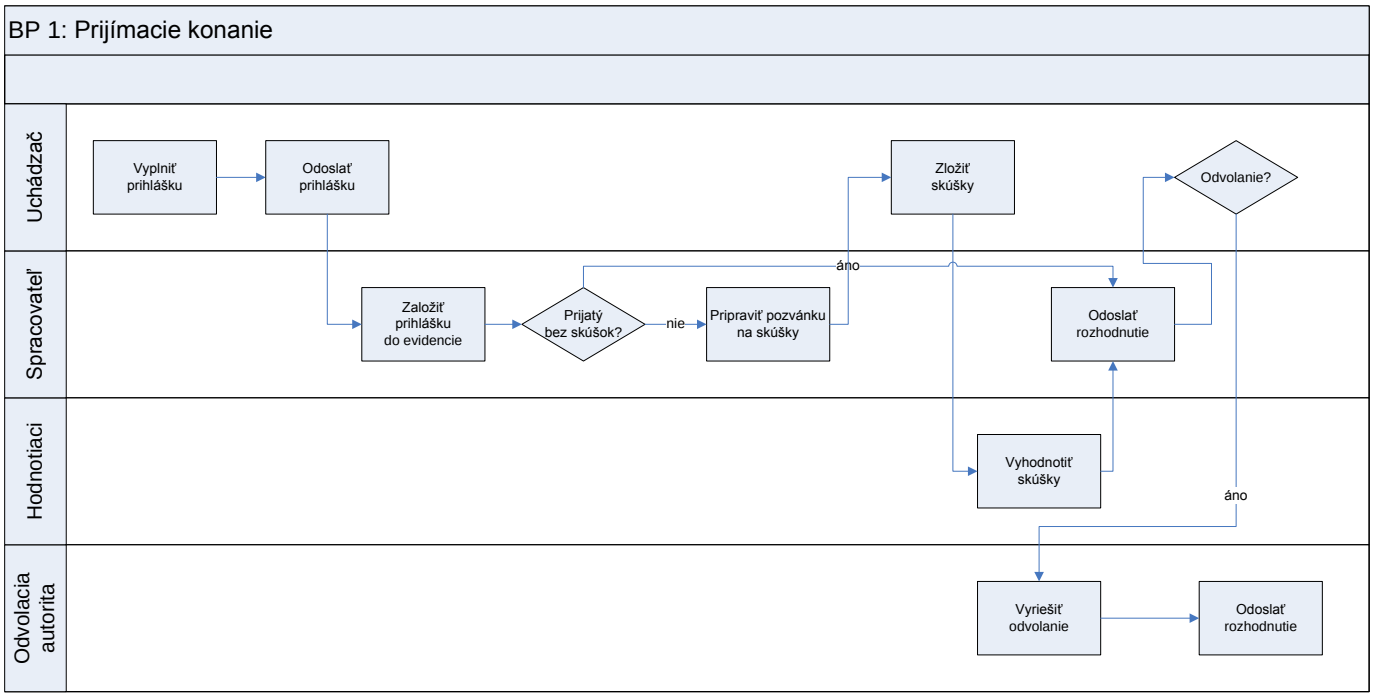
Informačné listy predmetov sú usporiadané v troch adresároch na najvyššej úrovni: *Kredit* pre informačné listy dobiehajúceho kreditového štúdia a *ILS* a *ILA*, v ktorých sa nachádzajú informačné listy nového štúdia v slovenskom, resp. anglickom jazyku. Ďalej sú štruktúrované podľa študijných programov, pričom každý program je reprezentovaný jedným adresárom, v ktorom sú už uložené jednotlivé informačné listy v súboroch, ktorých názov je *kód premetu* s príponou *rtf*.

RTF súbory sú súbory s formátovaným textom (Microsoft, 2007). Dátové položky sú označkované referenciami typu záložka (*bookmark* v anglických verziách aplikácie Microsoft Word), vďaka čomu k nim môžeme jednoducho a rýchlo pristupovať, napr. cez OLE Automation. V každom liste sú takto zadefinované záložky:

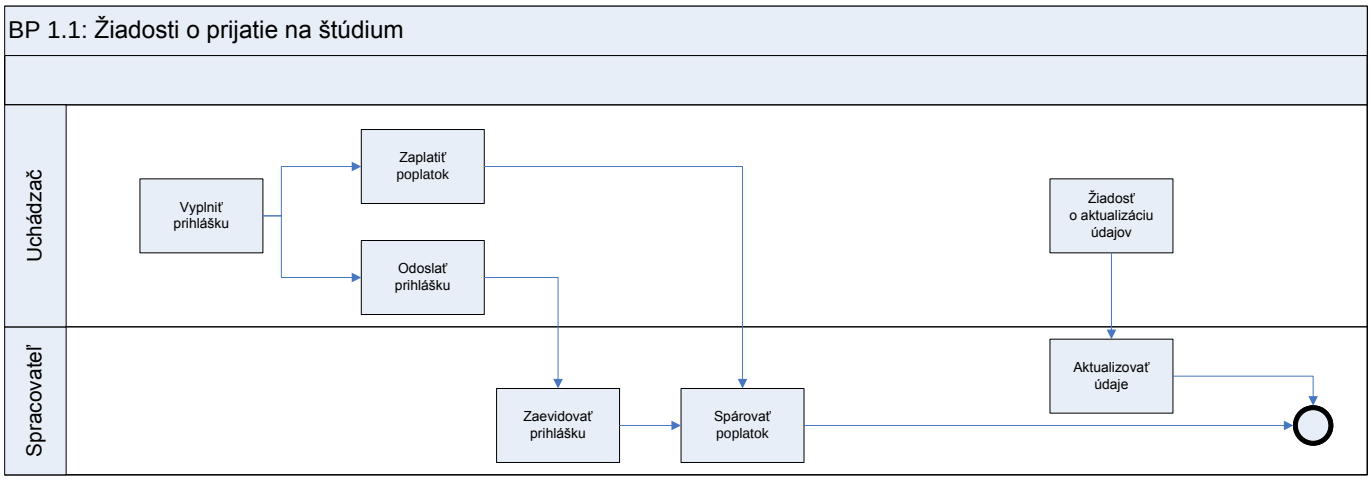
- *Ciel* pre cieľ predmetu,
- *Datum* pre dátum poslednej úpravy informačného listu,
- *Jazyk* pre jazyk výučby,
- *Kod* pre kód predmetu,
- *Literatura* pre literatúru,
- *Nazov* pre názov predmetu,
- *Osnova* pre stručnú osnovu predmetu,
- *Priebezne* pre text priebežného hodnotenia,
- *Zaverecne* pre text záverečného hodnotenia.

Niektoré dáta nebudeme využívať, keďže ich máme k dispozícii v systéme Študent ako v autoritatívnom zdroji (menovite názov a jazyk výučby), kód predmetu použijeme na prepájanie ako cudzí kľúč.

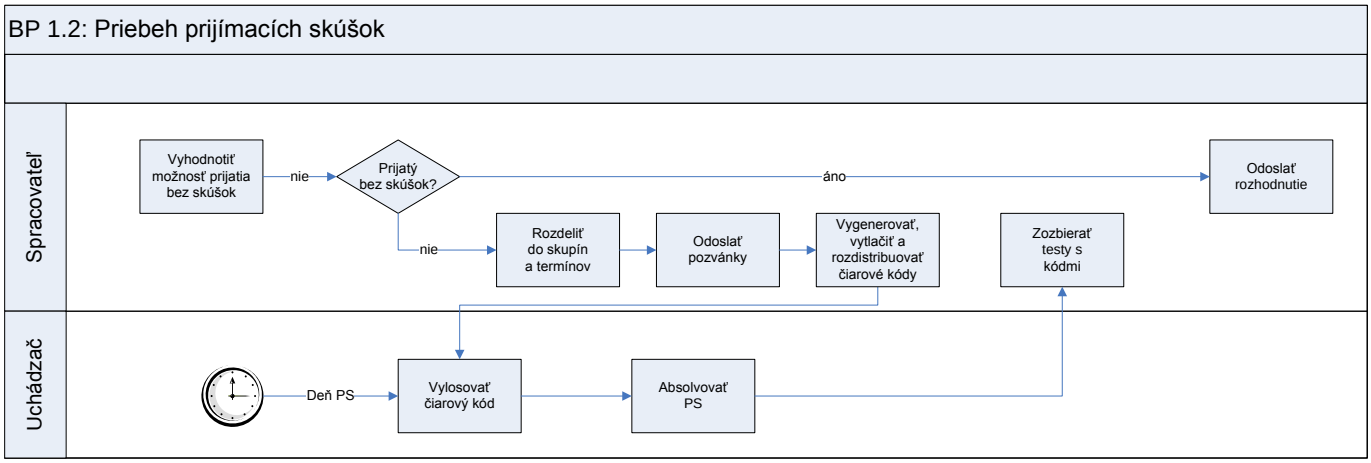
Príloha G. Diagramy



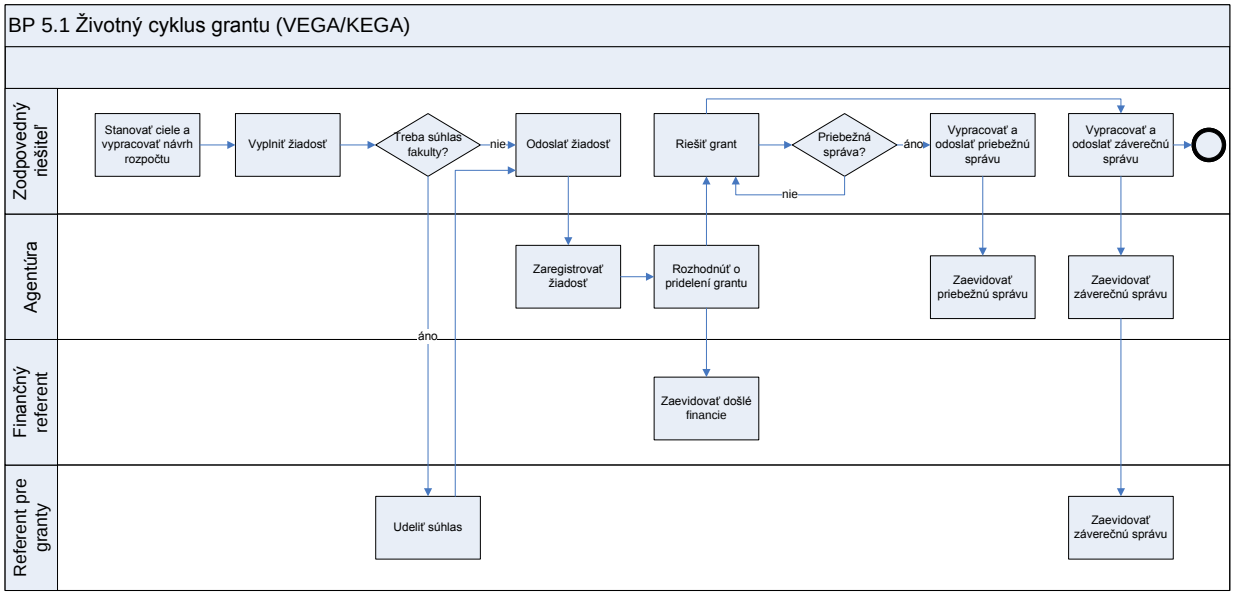
Obrázok 7-4 Model podnikového procesu prijímacieho konania (BP1)



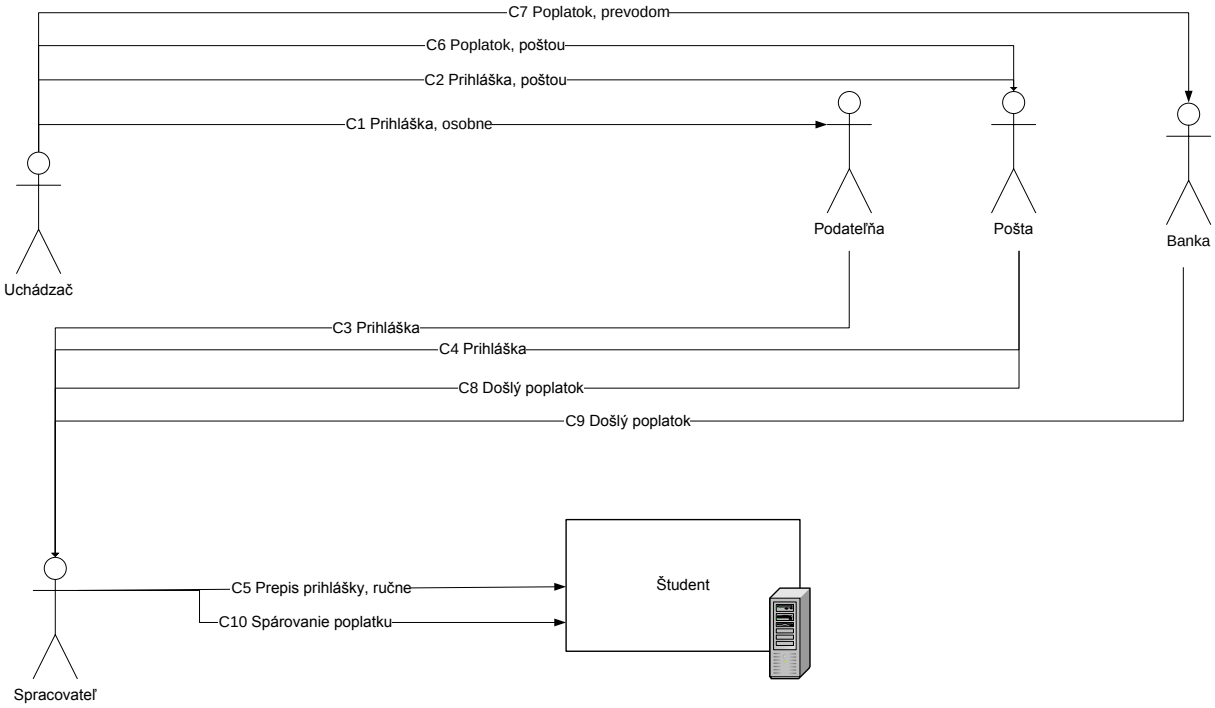
Obrázok 7-5 Dekompozicía podnikového procesu BP1



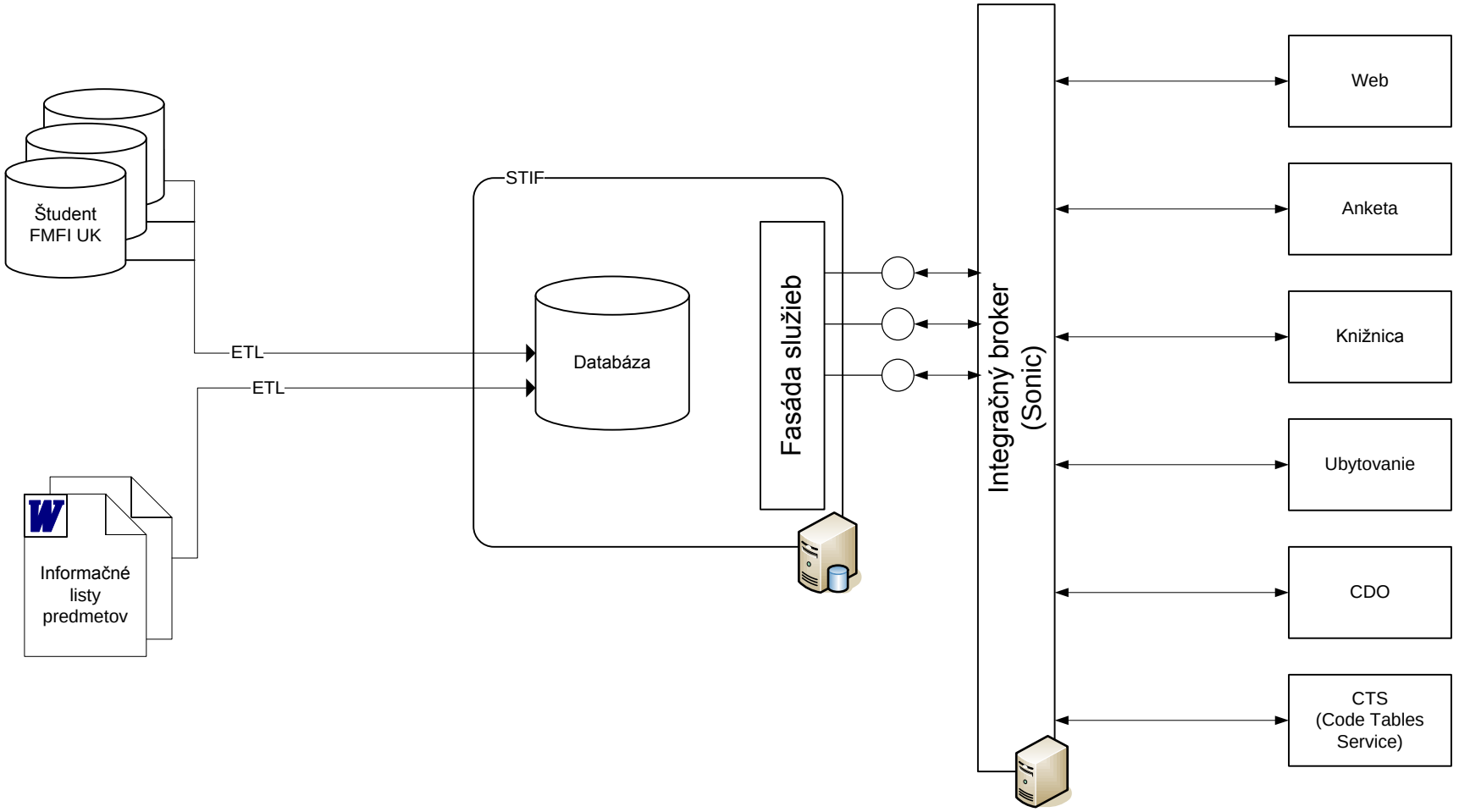
Obrázok 7-6 Dekompozícia podnikového procesu BP1



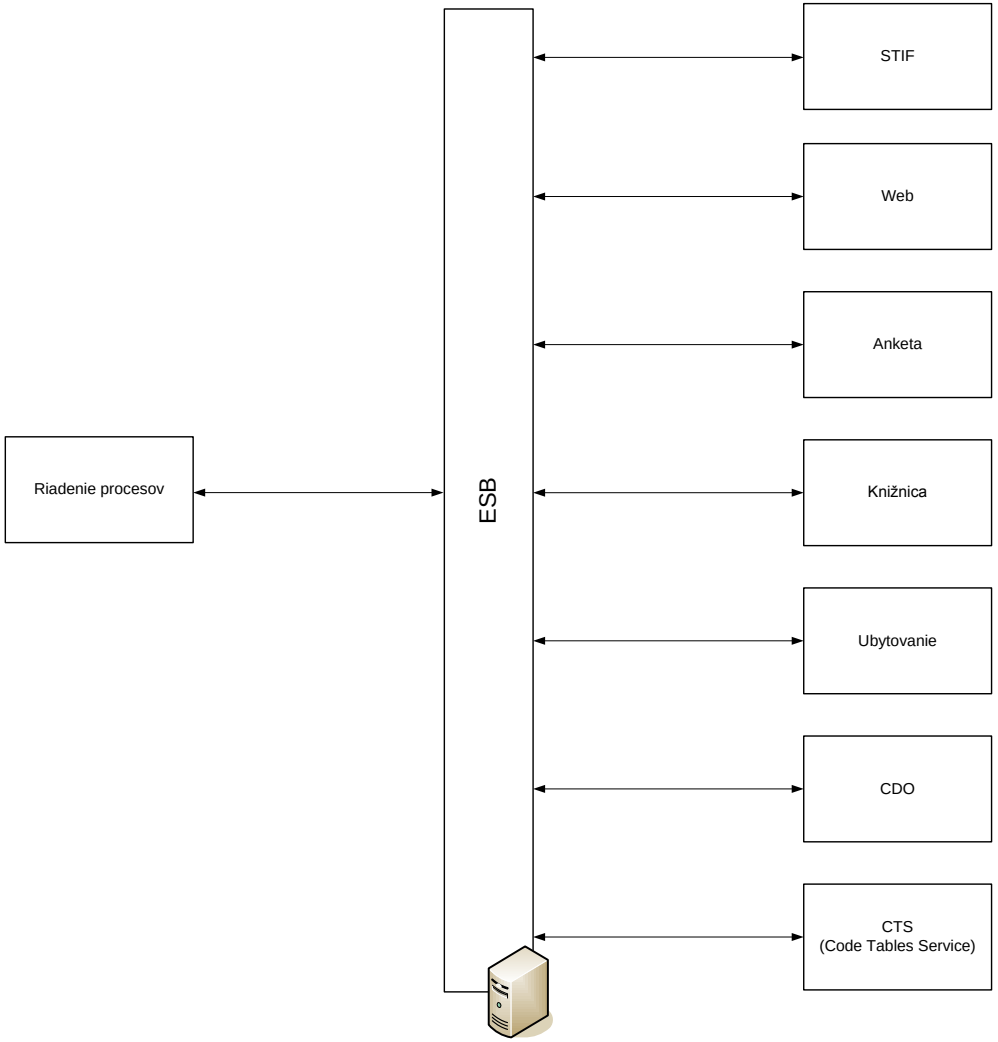
Obrázok 7-7 Podnikový proces prideľovania a riešenia grantov VEGA/KEGA



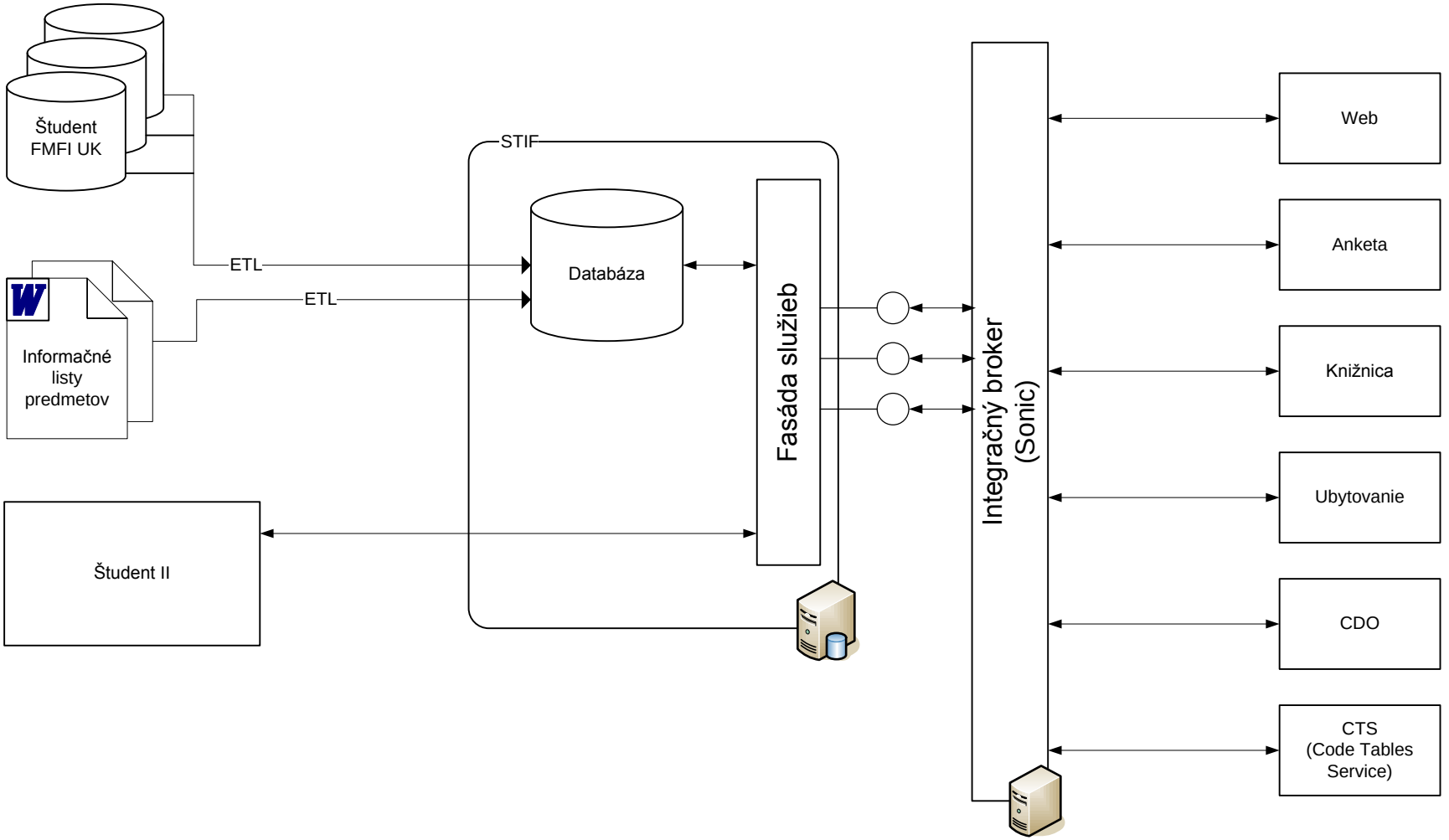
Obrázok 7-8 Procesno-systémový model pre BP1.1



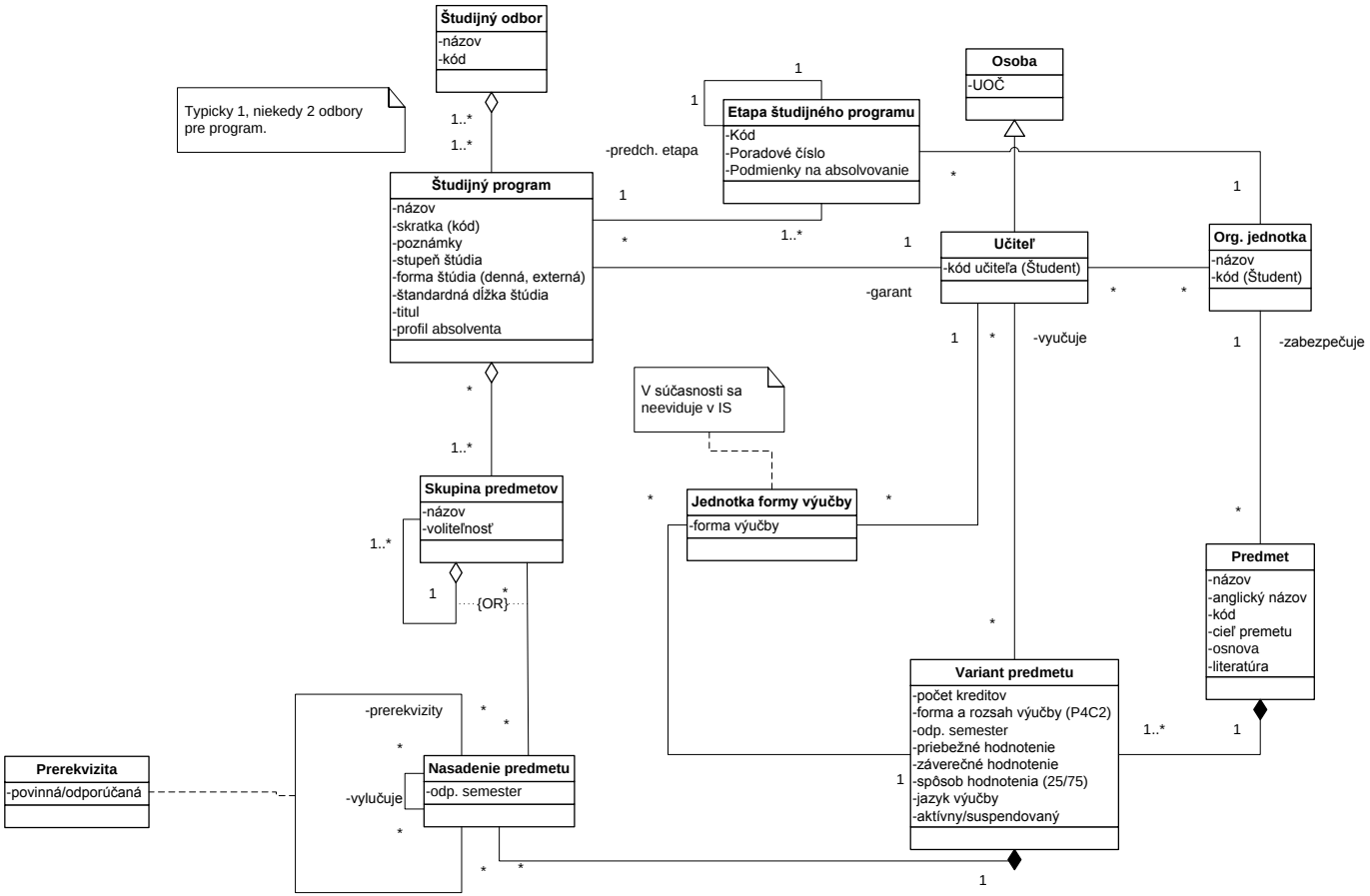
Obrázok 7-9 Navrhovaná integračná architektúra v strede so systémom STIF



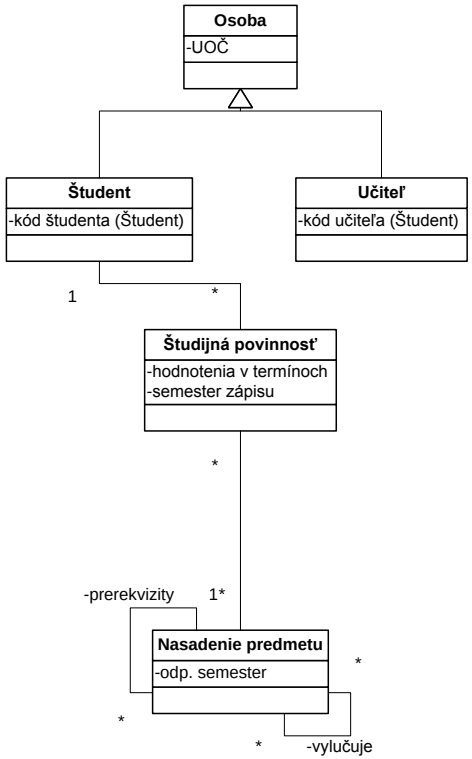
Obrázok 7-10 Vybrané systémy UK v kontexte podnikovej zbernice služieb



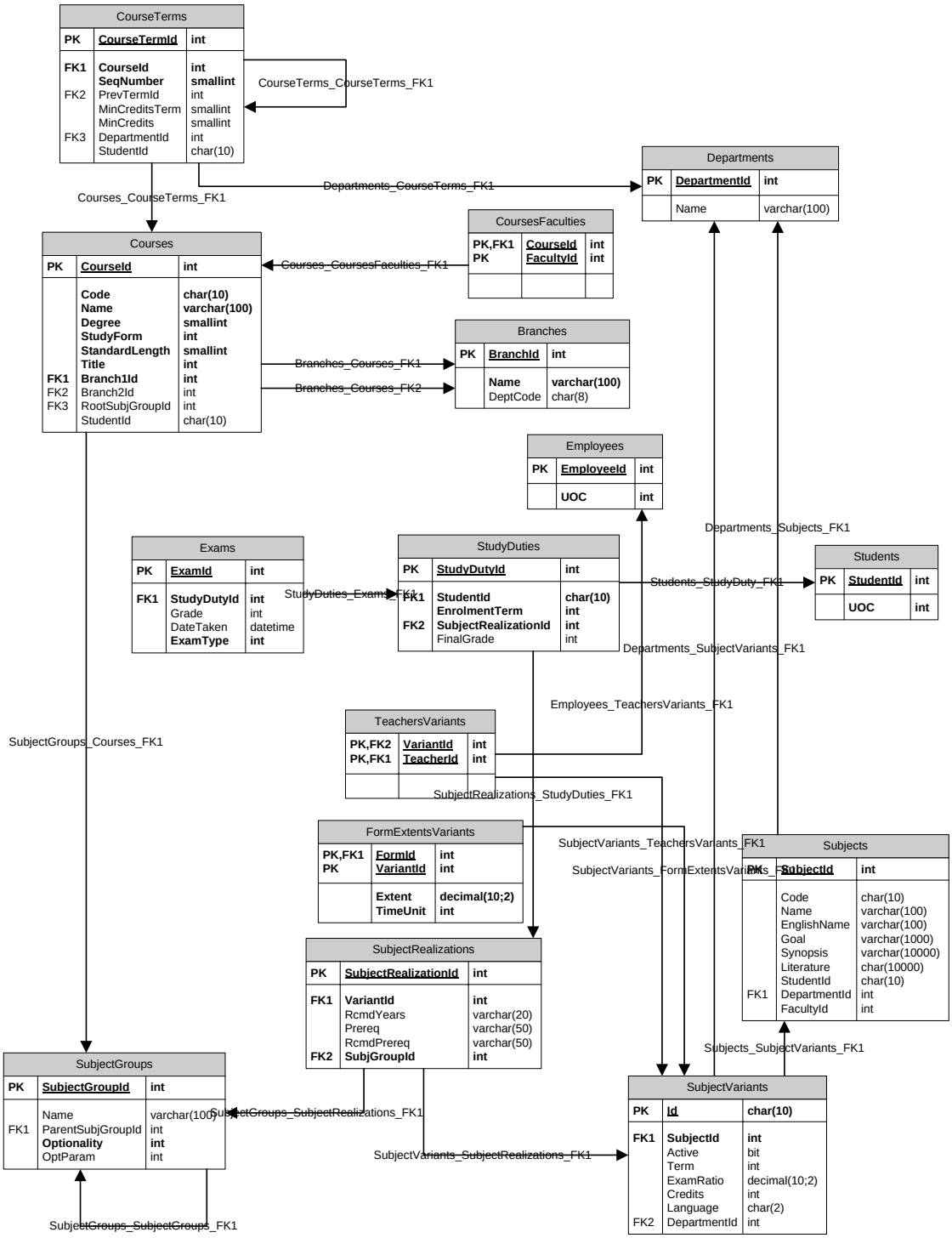
Obrázok 7-11 Integračná architektúra po nasadení novej verzie systému pre podporu študijnej agendy (Študent II) za predpokladu súbežnej prevádzky so systémom Študent I



Obrázok 7-12 Logický dátový model STIF (časť 1)



Obrázok 7-13 Logický dátový model STIF (časť 2)



Obrázok 7-14 Fyzický dátový model STIF