

Metodika vývoje informačního systému s pomocí nástroje Power Designer

Předmluva

Tato publikace popisuje metodiku vývoje informačního systému.
Na metodice se podílel autorský kolektiv, formovaný na půdě Katedry informačních
technologií Vysoké školy ekonomické v Praze, ve složení:

Užší autorské jádro:

Václav Řepa
Václav Synáček
Petr Hamerník
Ondřej Diviš
Ivo Klimeš

Spolupracovníci (v abecedním pořadí):

Jitka Bastlová
Marek Bednář
Jarek Farkaš
Jan Juříček
Roman Plášil
Hynek Škoda
Martin Vilímovský
Petr Žižka

Metodika byla vyvinuta za podpory společnosti Sybase Software a jejího produktu Power
Designer v.12 v době od podzimu 2005 do jara 2006.

Obsah

Metodika vývoje informačního systému s pomocí nástroje Power Designer	1
Předmluva	1
Obsah	2
Úvod	4
Díl A Základní postup vývoje IS	4
A.1 Základní východiska a principy	5
A.1.1 Princip abstrakce	8
A.1.2 Princip modelování	22
A.2 Životní cyklus vývoje IS	28
A.3 Řízení projektů vývoje IS	30
A.4 Konceptuální a procesní modelování	34
A.5 Funkčnost systému	38
A.6 Požadavky na IS a jejich role v procesu vývoje IS	41
A.7 Podrobnější postup etap analýzy a návrhu IS	43
A.7.1 Kroky globálního návrhu	43
A.7.2 Kroky detailní analýzy	46
A.7.3 Kroky designu	48
A.7.4 Závěrečné kroky	50
Díl B Použití jednotlivých modelů a jejich provázání	52
B.1 Analytické modely	53
B.1.1 Diagram tříd	53
B.1.2 Diagram procesů	55
B.1.3 Diagram stavů	62
B.1.4 Provázání procesů s třídami objektů pomocí jejich životních cyklů	64
B.1.5 Popis funkčnosti IS - Diagram datových toků	68
B.1.6 Ostatní – doplňkové diagramy UML a jejich použití	76
B.2 Konstrukční modely	79
B.2.1 Kritéria designu	81
B.2.2 Diagram komponent	83
B.2.3 Diagram procesů (rozmístění)	87
Díl C Realizace modelů v nástroji Power Designer	95
C.1 Odlišné používání pojmů „model“ a „diagram“	95
C.2 Modely PoweDesigneru nezbytné pro realizaci principů metodiky	95
C.2.1 Object Oriented Model (OOM)	95
C.2.2 Business Process Model (BPM)	96
C.2.3 Data Flow Diagram	96
C.2.4 Model požadavků (Requirements Model - RM)	96
C.2.5 Konceptuální datový model (Conceptual Data Model - CDM)	98
C.2.6 Fyzický datový model (Physical Data Model - PDM)	98
C.2.7 Information Liquidity Model (ILM)	98
C.2.8 XML model	98
C.2.9 Free Model (FM)	98
C.2.10 Výčet sady konzistenčních pravidel	99
C.3 Podrobný popis realizace vybraných modelů a vazeb	101
C.3.1 Diagram datových toků (DFD)	101
C.3.2 Vazba externích aspektů procesu na diagram tříd	102
C.3.3 Specifikace a kontrola povinných metod u každé třídy	104

C.3.4	Metody atributů	105
C.3.5	Metody asociovaných tříd	105
C.3.6	STD diagram pro každou neprimitivní třídu	106
C.3.7	Vazba elementárních Data storů do diagramu tříd	106
C.3.8	Vazba mezi procesem a požadavkem	107
C.3.9	Vazba mezi funkcí a procesem	108
C.3.10	Vazba mezi funkcí a požadavkem	109
C.3.11	Vazby mezi Use Case a funkčním požadavkem	109
Díl D	Použití metodiky	111
D.1	Metodika a standardy	111
D.2	Role metodiky v organizaci	112
D.3	Role uživatelů ve vztahu k metodice	112
D.3.1	Role vedení	112
D.3.2	Role analytiků	113
D.3.3	Role programátorů, designerů a jiných profesí podílejících se na vývoji IS	113
D.3.4	Role pracovníků helpdesku	113
D.4	Metodika a řízení projektů	113
D.5	Změna Business Process Modelu jako cesta ke zlepšení systému	115
D.5.1	Role požadavků na IS	117
Díl E	Příklad použití metodiky v prostředí Power Designer	120
E.1	Popis business systému – základního východiska návrhu IS	120
E.1.1	Předmět podnikání a základní koncept imaginární firmy	120
E.1.2	Podniková vize	120
E.1.3	Strategie k naplnění vize	121
E.1.4	Popis činnosti	121
E.1.5	Specifikace Informačního systému:	123
E.2	Globální analýza systému	125
E.2.1	Analýza BSP	125
E.2.2	Základní procesy	137
E.2.3	Funkční analýza a datové toky na úrovni GAN	142
E.2.4	Návrh základních subsystémů IS	146
E.2.5	Globální návrh designu systému	149
E.3	Detailní analýza vybraných částí systému	154
E.3.1	Hranice systému - část systému pro detailní analýzu a návrh.	155
E.3.2	Analýza událostí a činností	155
E.3.3	Diagramy datových toků	156
E.3.4	Procesní modely	160
E.3.5	Diagram tříd – Class diagram	171
E.3.6	Životní cykly klíčových tříd – stavové diagramy (State Chart)	173
E.4	Design systému – Hardwarová a softwarová architektura	179
E.4.1	Softwarová architektura	179
E.4.2	Odvození diagramu komponent z DFD	180
	Hardwarová architektura	181
E.5	Závěrečná poznámka k příkladu	182
	Slovník pojmů	183
	Literatura	186

Úvod

Tato publikace se věnuje metodice vývoje informačního systému se specifickým zaměřením na použití nástroje Power Designer. Vzhledem k tomuto svému zaměření se tak z celého životního cyklu vývoje informačního systému zaměřuje především na fáze analýzy a konstrukce systému, které jsou pro podporu nástrojem CASE klíčovými.

Publikace je rozdělena do pěti dílů:

- **Díl A** poskytuje *globální přehled o vývoji informačního systému (IS)*. Vysvětluje základní principy vývoje IS, popisuje životní cyklus IS a jeho vztah k problematice řízení projektů vývoje IS, vysvětluje jednotlivé klíčové druhy činností v procesu vývoje IS a specificky se věnuje významu jednotlivých analytických modelů.
- **Díl B** se zaměřuje *detailně na jednotlivé analytické a konstrukční modely*, vytvářené v procesu vývoje IS. Vysvětluje povahu soustavy analytických a konstrukčních modelů, smysl jejich rozlišování, jakož i přirozenou potřebu jejich vzájemného provázání.
- **Díl C** obsahuje *specifikaci možností a způsobů podpory jednotlivých modelů a jejich vzájemných souvislostí v nástroji Power Designer*. Mapuje přirozenou potřebu vnitřního provázání jednotlivých modelů, vysvětlovanou v předchozím dílu, na konkrétní možnosti nástroje Power Designer.
- **Díl D** se zabývá *možnostmi použití metodiky*. Vysvětluje vztah mezi obecnou metodikou a metodikou v roli firemního standardu, rozebírá nutnost přizpůsobení metodiky konkrétním podmínkám použití, vymezuje vztah řízení projektů vývoje IS a řízení informačního systému v organizaci apod.
- **Díl E** obsahuje *vzorový příklad použití metodiky v prostředí Power Designer*, demonstrující povahu jednotlivých modelů a jejich vzájemné provázání.

Díl A Základní postup vývoje IS

Tento díl popisuje metodický postup vývoje informačního systému, postavený na podrobném vysvětlení základních východisek - principů a přístupu k modelování vůbec. Metodika považuje tato základní východiska za svou nejdůležitější – klíčovou část. Ostatní aspekty metodiky (postup, techniky, způsoby tvorby jednotlivých modelů a detaily jejich propojení) jsou již odvozeny od těchto základních východisek. Jelikož žádnou metodiku nelze nikdy použít „tak jak je“, tedy jako „kuchařku“, podle níž by bylo možné postupovat bez jakékoliv výchozí znalosti, naopak – použití metodiky vždy vyžaduje značnou kvalifikaci a důkladné porozumění jejím principům, je právě tou nejdůležitější částí každé metodiky důkladná soustava hlavních východisek. Samotné použití metodiky pak vyžaduje její „implementaci“ v konkrétních „znalostních“ podmínkách dané organizace, dotvoření do konkrétní podoby vnitřního předpisu a stylu práce a také zorganizování jejího dalšího – permanentního vývoje.

V tomto smyslu jsou informace, obsažené v tomto dílu publikace, klíčovou částí celé metodiky.

A.1 Základní východiska a principy

Pozadí, které dalo podnět ke vzniku a určuje také směry dalšího vývoje systémů CASE, se běžně nazývá **metodikou tvorby informačních systémů**. Zahrnuje jednak celkový pohled na proces vzniku a existence informačního systému ve formě tzv. "životního cyklu systému", jednak rozpracování jednotlivých atributů (dimenzí) všech fází životního cyklu a konečně příslušné metody, techniky a nástroje, používané v jednotlivých činnostech vývoje a provozu informačního systému.

Nimal Jayaratna v [Jayaratna, 1994] definuje metodiku¹ jako „jasný způsob uspořádání myšlenek a činů“ a dále dodává, že „Metodiky obsahují modely a odrážejí určité náhledy na „realitu“ založené na souhrnu filosofických myšlenkových vzorů (paradigmat). Metodika nám říká „jaké“ kroky činit v jakém pořadí a „jak“ je provádět, avšak to nejdůležitější, co nám říká, je „proč“ to tak má být“.

Metodika (občas nevhodně zvaná též „metodologie“ - viz pozn.¹) tvorby IS je souhrn

- etap
- přístupů
- zásad
- postupů
- pravidel
- dokumentů
- řízení
- metod
- technik
- a nástrojů,

který pokrývá celý životní cyklus informačních systémů². Metodiky vývoje a provozu IS jsou určeny pro pracovníky podílející se na vývoji IS (dodavatele IS i zákazníky - zadavatele).

Určují kdo, kdy, co a proč má dělat během vývoje a provozu IS. Metodika by se měla vztahovat na všechny prvky IS:

- pracovníky
- organizační procedury a postupy
- data
- SW a HW
- organizační vlivy IS
- ekonomické otázky spojené s vývojem a provozem IS
- produkty a potřebné dokumenty v jednotlivých fázích životního cyklu IS
- použitelné metody, techniky a nástroje v jednotlivých fázích životního cyklu IS
- způsob řízení v jednotlivých fázích životního cyklu IS.

Základní pojmy

¹ v anglosaském světě, milujícími silná slova, se metodice v našem smyslu říká „metodologie“ (methodology)

² Pojem „životní cyklus informačního systému“ je pochopitelně relativní a je každou konkrétní metodikou chápán trochu jinak (viz následující popis jednotlivých známých metodik v dalších kapitolách). Zde je důležitá především snaha každé metodiky po celistvosti a to jak v obsahu životního cyklu, tak i v jeho možných aspektech (dimenzích).

Jak bylo zmíněno, metodiky jsou obsahově naplňovány jednotlivými

- **metodami**,
s nimi souvisejícími
- **technikami**
a k tomu potřebnými
- **nástroji**

Rozdíly mezi těmito základními pojmy z oblasti vývoje informačního systému nejsou vždy jednoduše rozeznatelné - v různých publikacích bývají tyto pojmy vzájemně zaměňovány nebo bývá různě posunován jejich význam. V této publikaci jsou tyto pojmy používány striktně vždy ve svém definovaném smyslu. Proto následuje stručné vymezení (definice) těchto základních pojmů:

Metodika tvorby IS

je doporučený souhrn etap, přístupů, zásad, postupů, pravidel, dokumentů, řízení, metod, technik a nástrojů pro tvůrce informačních systémů, který pokrývá celý životní cyklus informačních systémů. Určuje **kdo, kdy, co a proč** má dělat během vývoje a provozu IS.

Metodika by se měla vztahovat na všechny prvky IS (pracovníky, organizační procedury, data, SW a HW a další), organizační vlivy IS, ekonomické otázky spojené s vývojem a provozem IS a doporučené dokumenty a případně způsob řízení v jednotlivých fázích životního cyklu IS.

Metoda

určuje **co** je třeba dělat v určité fázi nebo činnosti vývoje či provozu IS.

Metoda je vždy spojena s určitým přístupem, jako je funkční, datový, nebo například objektový přístup. S přihlédnutím k této charakteristice řeší každá metoda postup činností v určité části (jedné nebo několika fázích) procesu vývoje systému, nebo pouze z některého úhlu pohledu na systém (data, funkce, SW, HW, atd.).

Příklady metod: informační analýza, funkční analýza, analýza konceptuálních tříd a procesů, aj.

Technika

určuje, **jak** se dobrat požadovaného výsledku. Zpravidla určuje přesný postup jednotlivých činností, způsob použití nástrojů, varianty rozhodnutí v určitých situacích a co z nich vyplývá, vymezuje obor své působnosti atd. Na rozdíl od metody je přesnější v závěrech a omezenější v okruhu použití.

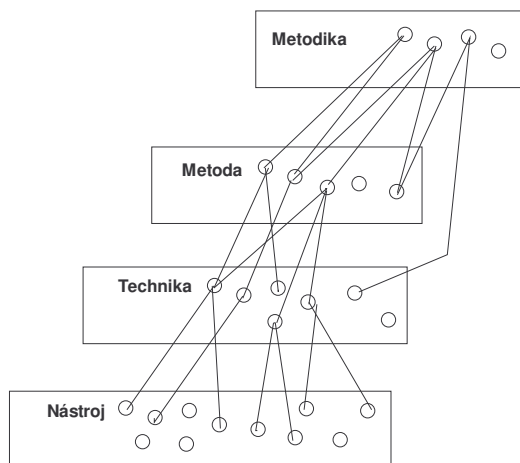
Příklady: transakční analýza, normalizace datového modelu, analýza událostí aj.

Nástroj

je prostředkem k uskutečnění určité činnosti v procesu vývoje a provozu IS a prostředkem k vyjádření výsledku této činnosti. Nástroj je často svázán s konkrétní technikou. Nástroje vždy formalizují vyjádření, proto je možné a žádoucí, aby byly v maximální míře automatizovány. Příklad: diagramy tříd, toku dat, stavový diagram, diagram procesů aj.

Není možné jednoduše prohlásit, že jednotlivé metody jednoznačně patří určitým metodikám, nebo že každá technika je tu výhradně ku podpoře nějaké své metody apod. Vztahy mezi metodami, technikami a nástroji, i jejich příslušností k metodikám, mohou být rozmanité, jak je ilustrováno níže (Obrázek A 1). Metodiky se odkazují na příslušné metody v relevantních místech životního cyklu IS, přičemž některé metody jsou více specifické (třeba

i jedinečné - vyskytují se toliko ve specifických metodikách), jiné mají univerzálnější charakter (například na metody datového modelování se odkazují prakticky všechny metodiky). Obdobně i technika může patřit k určité metodě, nebo je společnou pro řadu různých metod (například v dalším textu této publikace popisované techniky normalizace datových struktur, či analýza událostí). Technika buď vyžaduje specifický nástroj, který je s ní pevně svázán a bez ní nemá smysl, nebo používá obecněji použitelný nástroj (jako například zmiňovaný Diagram procesů, nebo ER diagram). Některé nástroje jsou natolik univerzální, že nemá smysl je vázat k nějakým technikám - jsou prostě nástroji, používanými různými metodami – takovým způsobem je například koncipován celý jazyk UML (viz jeho podrobnější popis v části Díl B této publikace).



Obrázek A 1 Vztahy mezi pojmy

V předchozím textu jsme vcelku podrobně diskutovali veškeré podstatné náležitosti metodiky vývoje IS. Na závěr je tedy namísto rekapitulace smyslu toho všeho, základního určení metodiky a jejích přínosů.

Metody analýzy a návrhu IS kladou značný důraz na své základní, *východí principy*. Tato kapitola se zaměřuje na popis těch základních principů metod analýzy, které mají obecnou platnost - platí jak pro metody „strukturované“ (např. Yourdonovu), tak i „nestrukturované“ (objektové a vývojově vyšší).

V metodách strukturované analýzy, stejně jako v metodách objektových, se tyto principy promítají do všech stránek návrhu informačního systému a tvoří tak jakési neměnné jádro těchto metod. Zatímco jednotlivé techniky a nástroje se mění a zdokonalují, stejně jako jednotlivé metody a jimi předepsané postupy návrhu informačního systému, přičemž tento proces změn a zdokonalování je časově neomezený, základní principy stále zůstávají a navždy zůstanou - definují totiž *základní obsah pojmu "Metody analýzy"* - ustoupení od těchto základních principů by vedlo k neadekvátnímu použití jejich nástrojů a pravidel a potažmo i k naprosto nesmyslnému výsledku.

Způsoby, jakými se základní principy do metod promítají, jsou rozličné:

- ⇒ Najdeme je v základních vlastnostech nástrojů (např. top-down charakter diagramu datových toků).
- ⇒ Najdeme je v návaznostech jednotlivých nástrojů, jak jsou definovány metodou (viz. nástroje konceptuálního versus technologického návrhu systému, nebo vztahy mezi nástroji konceptuálního popisu jako vyjádření principů abstrakce).
- ⇒ Najdeme je také v technikách a návodech k použití jednotlivých nástrojů (např. princip modelování v technikách návrhu funkčního a datového modelu).

⇒ A konečně již základní definice pojmů, používaných metodami, nejsou ničím jiným, než vyjádřením těchto principů.

Základní principy metod analýzy jsou:

- různé formy principu abstrakce:
 - Top-Down charakter funkční a procesní struktury
 - generalizace a specializace v datovém a objektovém modelu
 - princip tří architektur
 - princip různých pohledů na model systému
- princip modelování

V dalším textu se budeme jednotlivými principy zabývat podrobněji.

A.1.1 Princip abstrakce

Abstrakce znamená

1. *Myšlenkový proces odlučující odlišnosti a zvláštnosti a zjišťující obecné a podstatné vlastnosti předmětů a jevů okolní skutečnosti a vztahy mezi nimi.*
2. *Nepřihlížení k něčemu (tj. záměrná, vědomá nekonkrétnost).*

Hlavním důvodem existence principu abstrakce v metodách analýzy a návrhu IS je **snaha po rozdělení zkoumané problematiky na mentálně zvládnutelné části**.

Typickým rysem problematiky, zkoumané při návrhu informačního systému je totiž vždy její značná *rozsáhlost a složitost*.

- ◆ Již samotná předmětná oblast zkoumání je ve všech svých detailech, kterými je třeba se zabývat, značně složitá.
- ◆ Automatizace se týká vždy značného množství procedur a pracovních postupů, které mohou mít spoustu různých specifických variant ve specifických situacích a jsou ve složitých vzájemných vztazích.
- ◆ Data, zpracovávaná v informačním systému na jednu stranu vyjadřují celou řadu specifických informací, což je závislé na způsobu jejich zpracování a kombinaci, na druhou stranu jsou ve vzájemných logických, na způsobu jejich zpracování nezávislých, vztazích.

To vše vyvolává potřebu shlukovat procesy a data informačního systému

1. jednak podle jejich obecných logických souvislostí.

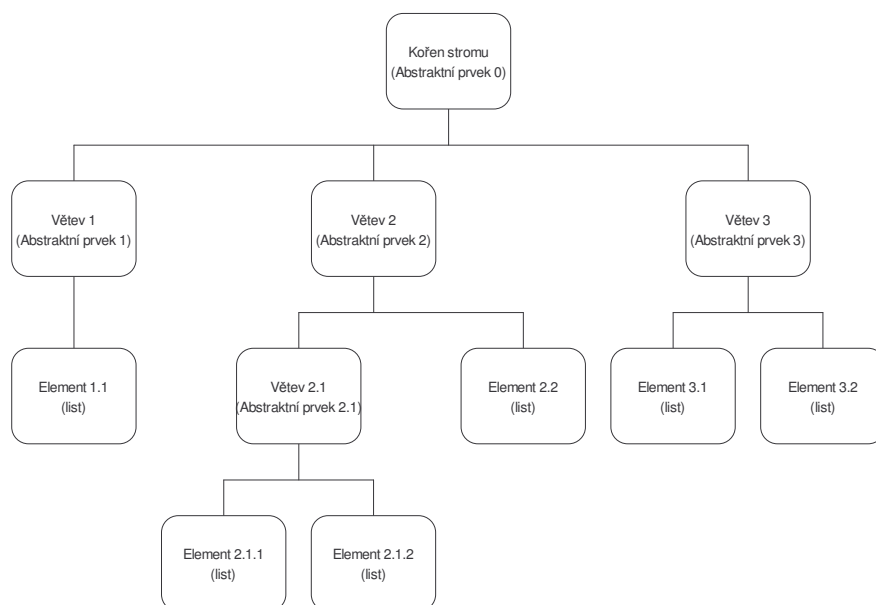
2. jednak podle jejich způsobu zpracování,

Kromě esence procesů a dat předmětné oblasti zájmu informačního systému má vyvíjený systém řadu dalších aspektů:

- *specifika technologie zpracování dat* (databázová versus souborová koncepce datové základny, centralizace, či naopak distribuce procesů a dat, typ použitého vývojového prostředí 3GL, 4GL, GUI atd.)
- *specifika použitého implementačního prostředí* (programovací jazyk, operační systém, databázový systém) a další
- A konečně: veškeré výše nastíněné oblasti, aspekty a specifika vyvíjeného systému je třeba vždy brát v úvahu v *jejich vzájemných souvislostech*, protože se konec konců jedná o jeden jediný celek-systém.

Abstrakce lze podle jejich způsobu rozdělit do základních typů:

- *víceúrovňové*, kde každý abstraktní prvek může být tvořen buď konkrétními (v případě nejnižší úrovně), nebo také abstraktními prvky (v případě některé z vyšších úrovní).
- *hierarchické (stromové)* abstrakce znamenají dělení celku na části (sdružování částí do celku). Definují stromovou strukturu prvků:



Obrázek A 2 Hierarchická stromová struktura

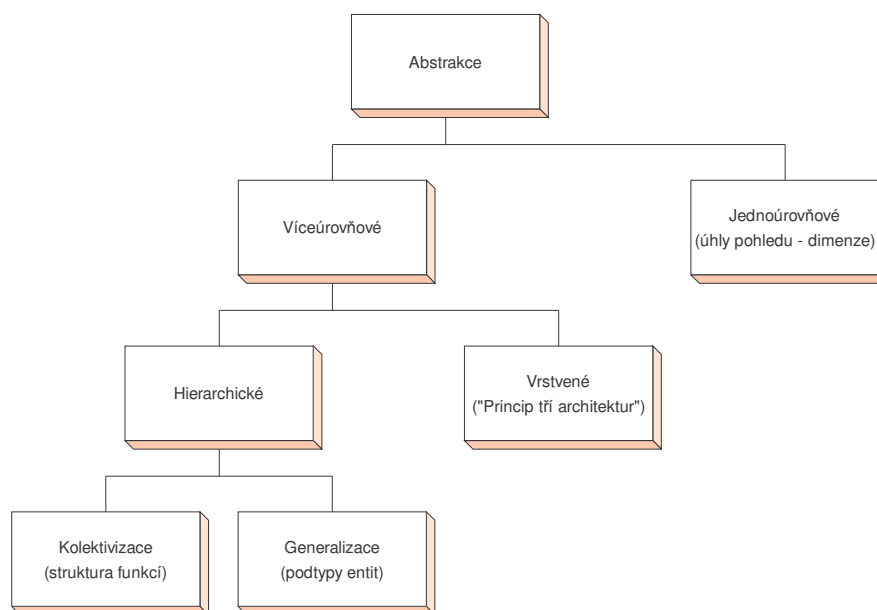
Každý prvek stromu má svůj jediný nadřazený abstraktní prvek (pojem) a je buď konkrétní (tvoří list stromu), nebo abstraktní (pak se skládá z podřazených prvků - tvoří další větev stromu). Podle způsobu tvorby abstraktních pojmů se rozlišují dva typy hierarchických abstrakcí:

- *agregace (kolektivizace)*, kde abstraktní pojem vyjadřuje pouze účelové sdružení svých prvků a nedefinuje jejich společné vlastnosti. Příkladem takové abstrakce může být pojem "zpracování účetních dokladů", který zahrnuje řadu procesů, jako "zpracování faktur", "zpracování příjmových dokladů" atd., aniž by přitom všem svým prvkům definoval povinné společné vlastnosti. Konkrétně se v metodách analýzy jedná o **Top-Down strukturu funkcí, nebo procesů**.
- *generalizace*, kde abstraktní pojem vyjadřuje sdružení prvků na základě jejich společných vlastností, které jsou všemi podřazenými prvky povinně děděny. Příkladem takové abstrakce může být pojem "zaměstnanec", zahrnující celou řadu podpojmů, jako "dělník", "manager", "úředník", u nichž nás zajímají jednak jejich specifické vlastnosti (u managera řídicí schopnosti, u dělníka řemeslné schopnosti, atd.), ale i takové vlastnosti, jako je věk, délka zaměstnaneckého poměru, plat atd., které nás zajímají u všech kategorií zaměstnanců. Konkrétně se v metodách analýzy jedná o princip **generalizace entit v datovém modelu, nebo tříd v modelu objektovém**.
- *vrstvené* abstrakce definují takovou hierarchickou strukturu prvků, kde každý prvek je detailním rozpracováním svého nadřazeného prvku z určitého hlediska s tím, že detaily hledisek, zohledněných u svých nadřazených prvků, přejímá. Každý nižší prvek tak tvoří další vrstvu detailního rozpracování ze svého hlediska, přidanou k vrstvám předchozím. Taková struktura hierarchie není stromová, ale lineární zohledňované hledisko se vždy týká celé předchozí (vyšší) vrstvy, nikoliv pouze její

části. Konkrétně se v metodách analýzy a návrhu IS jedná o **princip tří architektur informačního systému**

- *jednoúrovňové*, kde celá struktura konkrétních prvků je jednorázově plošně rozdělena na jednotlivé abstraktní oblasti substruktury (úhly pohledu). Každá oblast přitom akcentuje pouze některé charakteristiky, od ostatních abstrahuje. Konkrétně se v metodách analýzy a návrhu IS jedná o **princip různých pohledů (modelů)** na vytvářený systém datový versus funkční model atd.

Výše uvedenou klasifikaci abstrakcí v metodách analýzy a návrhu IS ilustruje následující obrázek:



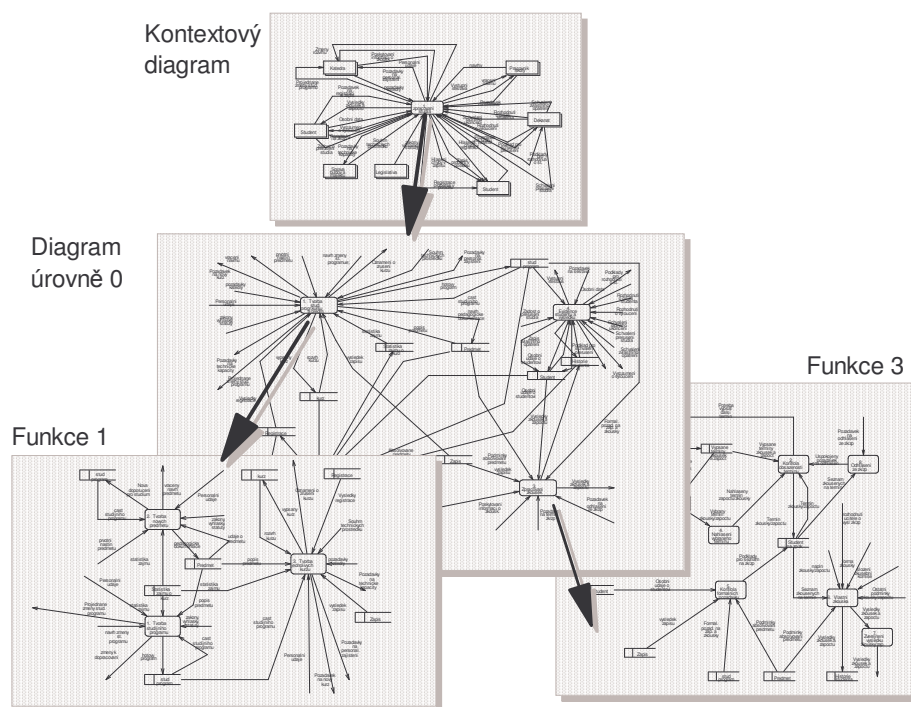
Obrázek A 3 Klasifikace abstrakcí

V následujícím textu jsou podrobněji popisovány jednotlivé druhy abstrakcí používané v metodách analýzy systému:

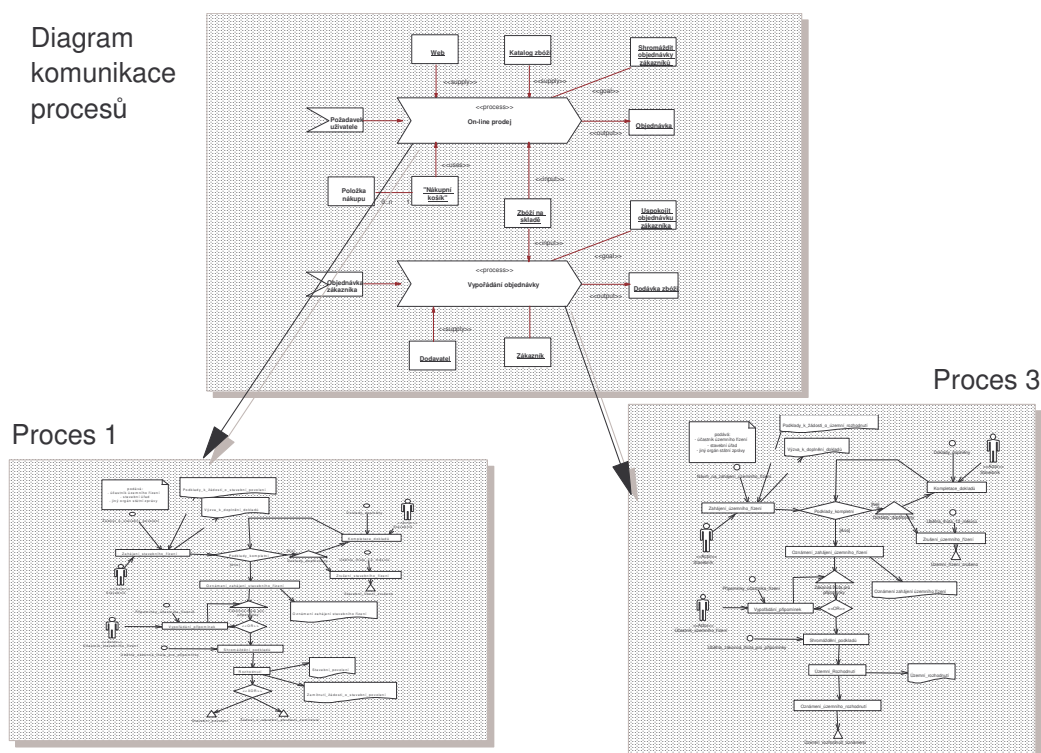
- Top-Down hierarchie funkcí a procesů
- generalizace / specializace v datovém modelu a modelu tříd objektů
- princip tří architektur a
- různé pohledy na vyvíjený systém

Top-Down hierarchie funkcí a procesů

Top-Down princip je tradičním principem, používaným ke zjednodušení pohledu na popisovaný systém. Tento princip byl použit v metodách strukturovaného programování, navrhování programových systémů (tzv. programování ve velkém) a také v metodách analýzy systému, založených na tzv. funkčním přístupu. Je také primárním (dominantním) principem strukturalizace procesů v současných metodách, ovlivněných procesní analýzou. Princip Top-Down spočívá v rozdělení pohledů na zkoumaný systém podle úrovně podrobnosti pohledu, jak ilustrují následující obrázky:



Obrázek A 4 Hierarchie funkcí v Diagramu datových toků



Obrázek A 5 Hierarchie podnikových procesů

Na nejvyšší úrovni je pohled nejméně podrobný, zato však úplný (je vidět celý systém). Na následujících nižších úrovních jde postupně vždy o pohled lokálnější (je vidět stále menší a menší část systému), avšak detailnější. Top-Down princip je tak formalizací *kompromisu mezi úplností a podrobností*. Bráno od detailů k vyšším celkům jde zde o uplatnění jednoho ze

dvou základních typů hierarchické abstrakce – agregace (kolektivizace). Systém je znázorněn ve *stromové struktuře*, kde hierarchicky nejvyšší prvek - *kořen stromu* se skládá z hierarchicky nižších prvků a každý takový prvek se může skládat z ještě nižších prvků atd., čímž vznikají jednotlivé *větvě stromu*. Na samém konci stromové struktury jsou prvky dále nerozložitelné - elementární, tak zvané *listy stromu*.

Smyslem principu top-down jako *postupu analýzy návrhu systému* je umožnit zkoumání a návrh systému po částech - po jednotlivých abstraktních úrovních tak, aby bylo možné zabývat se v určitém okamžiku pouze návrhem jedné úrovně jedné větve stromu. Tím získáme možnost rozdělit komplexní úvahy o navrhovaném systému do mentálně zvládnutelných celků. Kromě toho lze podle stromové struktury systému snadno dělit práci mezi jednotlivé vývojové týmy při zachování představy o celku. Aby byl takový postup rozumně uskutečnitelný, je třeba minimalizovat redundantní vazby mezi jednotlivými prvky struktury. K tomu slouží předepsané *vlastnosti stromové struktury*:

- každý prvek struktury, s výjimkou kořene stromu, má právě jeden prvek nadřazený
- každý prvek struktury může být buď prvkem
 - *abstraktním* (skládá se z podřazených prvků), nebo
 - *konkrétním*, tj. listem stromu.

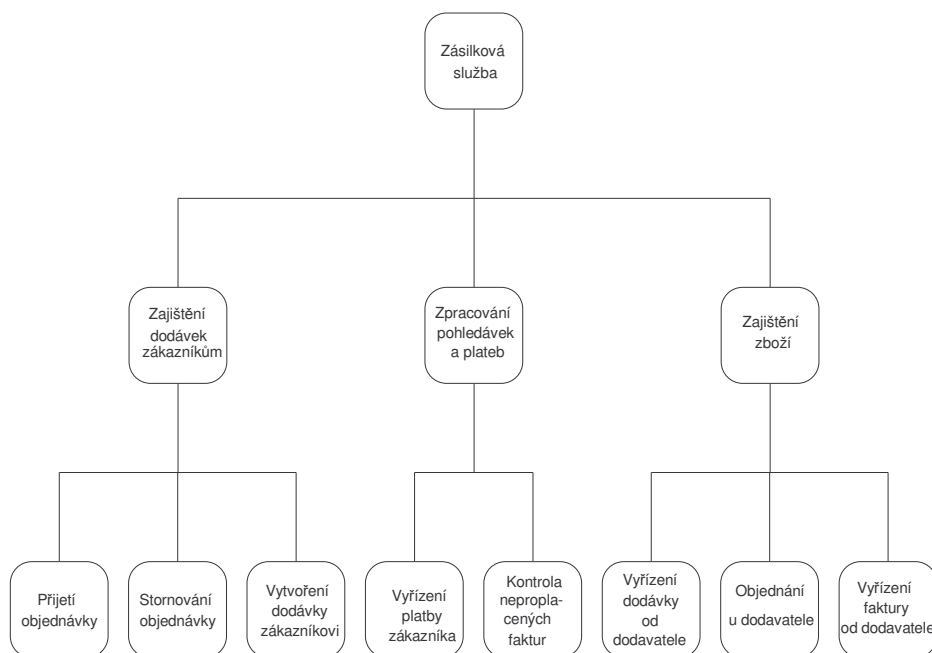
Jedině vlastnostmi listů stromu (konkrétních prvků) jsou definovány skutečné vlastnosti systému, všechny abstraktní (neelementární) prvky slouží pouze k popisu uspořádání systému (jejich vlastnosti jsou definovány vlastnostmi prvků, z nichž se skládají).

Z těchto dvou základních vlastností stromové struktury vyplývají následující odvozené vlastnosti vazeb mezi prvky struktury:

- vertikální vazby mezi prvky vyjadřují výhradně hierarchickou podřízenost typu "skládá se z"
- horizontální vazby mezi prvky vyjadřují interakci jednotlivých prvků a mohou být popisovány pouze mezi prvky téže hierarchické úrovně téže větve stromu. Interakce prvků z různých větví stromu totiž vždy vyplývá z interakce prvků jim nadřazených.

Dodržením těchto pravidel je dosaženo takového popisu vazeb mezi prvky, kde jsou redundance omezeny na minimum: buď se jedná o vazbu mezi prvky téže hierarchické úrovně téže větve stromu - potom je popsána přímo, nebo o vazbu mezi prvky z různých větví stromu - ta je popsána prostřednictvím vazby mezi prvky jim nadřazenými. Pro úplnost: vazba mezi prvky z různých hierarchických úrovní téže větve je nesmyslná, protože každý takový vyšší prvek je abstrakcí toho nižšího. Každá vazba tak může být popsána pouze jednou, zde jsou redundance vyloučeny - ty se vyskytují pouze u hraničních vazeb (které se musí objevit jak u nadřazeného prvku, tak i ve struktuře prvků podřazených - míra těchto „nutných“ redundancí ale již závisí především na konkrétní podobě uplatnění tohoto principu - na konkrétním diagramu a pravidlech jeho použití).

Výše popsané zákonitosti stromové struktury ilustruje následující obrázek:



Obrázek A 6 Příklad struktury funkcí

V tomto příkladu se Zásilková služba skládá z elementárních činností:

Přijetí objednávky, Stornování objednávky, Vytvoření dodávky zákazníkovi, Vyřízení dodávky od dodavatele, Objednání u dodavatele, Vyřízení faktury od dodavatele, Vyřízení platby zákazníka a Kontrola neproplacených faktur. To jsou konkrétní činnosti. K jejich popisu je použito abstraktních činností Zajištění dodávek zákazníkům, Zajištění zboží a Zpracování pohledávek a plateb, které vyjadřují účelové seskupení jednotlivých činností. Samotná Zásilková služba je také abstraktní činností. Vertikální vazby zde vyjadřují jaký pojem se z jakých podpojmů skládá, případné horizontální vazby (které tu vidět nejsou) by mohly vyjadřovat interakci činností - předávané údaje. Je zřejmé, že například existuje interakce mezi činnostmi Vytvoření dodávky zákazníkovi a Vyřízení platby zákazníka. Tento vztah je vyjádřen prostřednictvím vazby mezi jim nadřazenými abstraktními činnostmi Zajištění dodávek zákazníkům a Zpracování pohledávek a plateb, které si předávají údaje o objednávkách a fakturách zákazníků.

Ačkoliv původním podnětem ke vzniku principu Top-Down byla snaha usnadnit *postup* návrhu systému (systém má být navrhován po jednotlivých abstraktních úrovních shora dolů s používáním abstraktních, čím dál konkrétnějších pojmů až na samém konci dospějeme k návrhu uspořádání elementů systému), takové použití tohoto principu je velmi problematické a vpravdě neuskutečnitelné. Až teprve při zkoumání detailů jsme totiž schopni konkrétně rozhodovat o uspořádání celého systému (tedy i abstraktních - nadřazených vazeb mezi prvky). V praktickém postupu jsme tak nuceni (pokud nemáme již na začátku návrhu zcela jasno ohledně struktury systému) se velmi často vracet a opravovat chybné návrhy vazeb mezi prvky vyšších úrovní, případně celou dekompozici (tj. hierarchické rozdělení na podprvky) systému. Ve složitějších systémech to může mít za následek až neúnosné prodlužování návrhu. Význam principu Top-Down je tak třeba vidět především ve smyslu *dokumentačním* - jako přehledný *způsob popisu logického uspořádání* systému, nikoliv ve smyslu postupu návrhu. M.A.Jackson uvádí v [Jackson, 1983] analogii s učebnicemi matematiky, které popisují teorii v logickém uspořádání: jednotlivé teoremy jsou dokazovány pomocí důkazů teorémů podřízených. Avšak pořadí, v jakém jsou teoremy v matematice objevovány, je zcela

jiné - uspořádání popisu, sledující věcnou logiku teorie, nikdy neodpovídá pořadí jejího vývoje.

Princip Top-Down se v metodách analýzy informačních systémů projevuje především v základních vlastnostech nástrojů, které slouží k popisu funkční struktury systému a struktury procesů - diagramu datových toků a procesním diagramu.

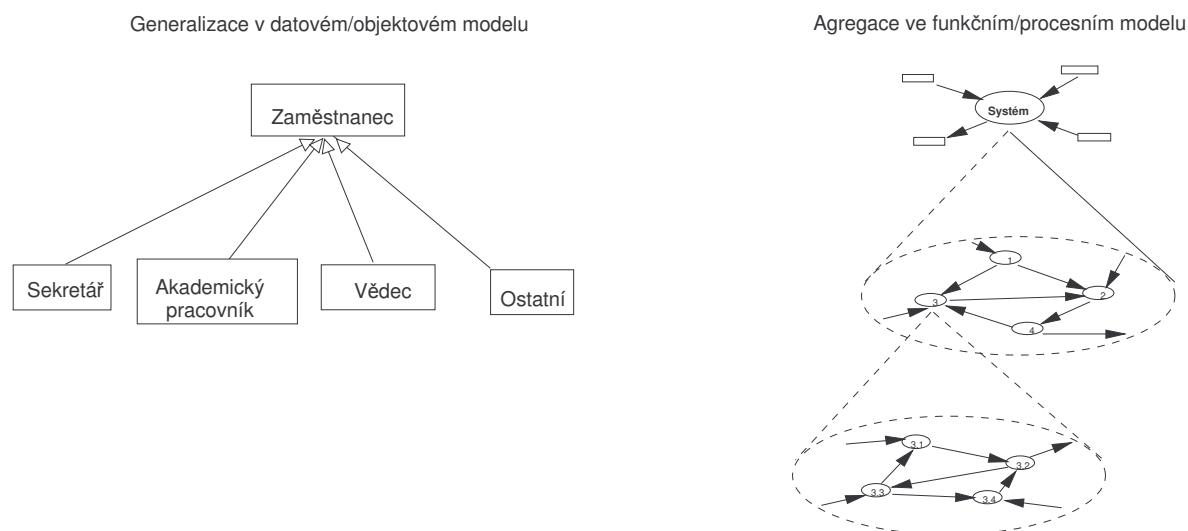
Generalizace - specializace v objektovém a datovém modelu

Jak již bylo naznačeno, v analytických metodách se používají dva základní typy hierarchické abstrakce:

- abstrakce **část - celek** (kolektivizace, agregace), která se běžně používá například ve funkčním modelu systému, kde se dělí systém na subsystémy, části subsystémů atd. (viz předchozí kapitoly), nebo v procesním modelu při dekompozici činností procesu do samostatných detailních procesů.
- abstrakce **specifický typ - generický typ** (generalizace), která je naopak typickou hierarchickou abstrakcí v datovém a objektovém modelu, kde umožňuje jednotlivé entity / objekty zobecňovat do vyšších abstraktních celků – generických typů.

Zatímco pro agregaci je typická principiální neomezenost dělení a vyšší celek je jí zcela definován jako souhrn svých částí (nemá žádný jiný význam), v případě generalizace není, na rozdíl od agregace, *nadřazený celek* definován jako souhrn podřazených částí, ale jako *nositel jejich společných vlastností (atributů)*.

Rozdíl mezi těmito dvěma základními typy hierarchických abstrakcí ilustruje následující obrázek:



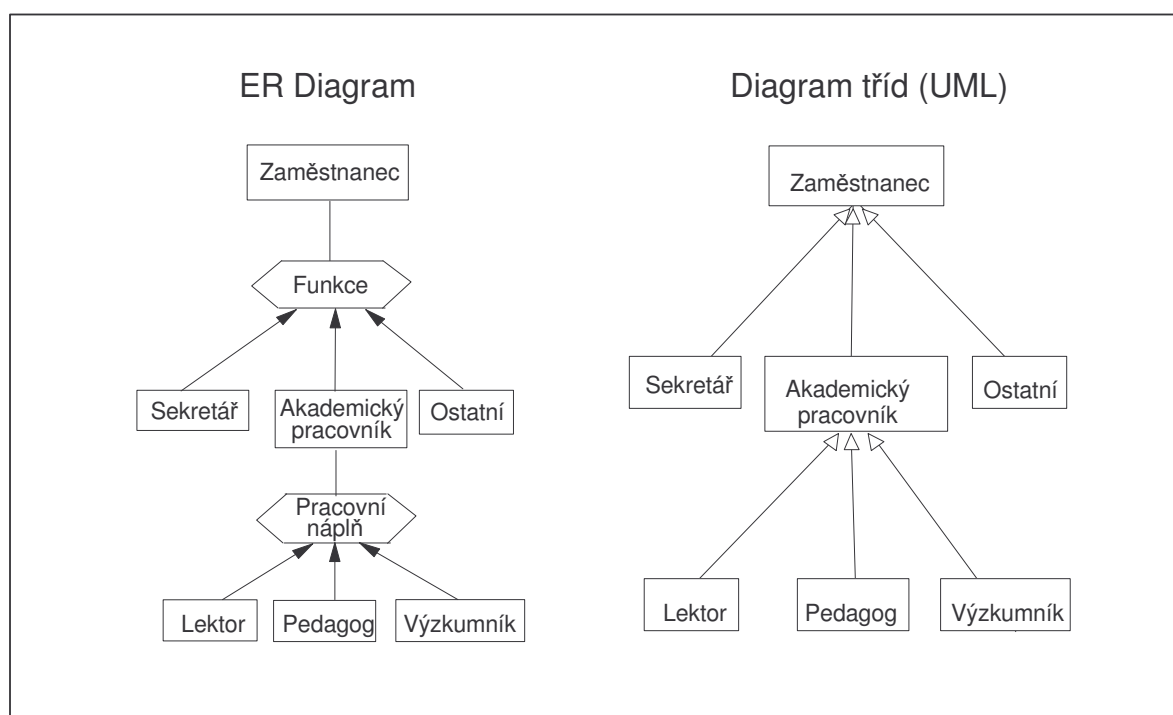
Obrázek A 7 Hierarchické abstrakce

Zatímco abstraktní funkce jsou ve funkčním modelu souhrnem svých subfunkcí, podobně jako každou činnost procesu můžeme vidět jako samostatný pod-proces, sestávající z podčinností (viz pravou stranu obrázku, kde si lze dosadit třeba konkrétní příklad Zásilkové služby z předchozí kapitoly), abstraktní entita v datovém modelu, stejně jako generická třída objektů, je *zobecněním* (nadtypem) svých podtypů (viz levou stranu obrázku - Sekretář, Akademický pracovník, Vědec a Ostatní zjevně nejsou jednotlivými složkami Zaměstnance, nýbrž jeho specifickými variantami). Vztah mezi celkem a jeho částí je v datovém /

objektovém modelu vztahem objektu (entity) ke svým elementárním charakteristikám (atributům, či „metodám“), není však vyjadřován zaváděním abstraktních objektů³.

Obecný nadtyp definuje společné vlastnosti všech svých podtypů, z nichž každý jeden může mít ještě specifické své vlastnosti, jimiž se od ostatních podtypů (variant nadtypu) liší. V datovém modelu to znamená, že existují atributy nadtypu (zde Zaměstnanec), které mají všechny jeho specifické varianty (zde například jméno, nebo plat apod.), každý podtyp potom může mít specifické atributy, které se u ostatních variant objektu nevyskytují (například míra pedagogické zkušenosti bude specifickým atributem Učitele, zatímco rychlost psaní na stroji nás bude zajímat spíše u typu Sekretář, než u kteréhokoliv jiného). Atributem, podle nějž se jednotlivé podtypy rozlišují, je zde pracovní Funkce.

Jednotlivé subtypy (varianty) jsou vzájemně disjunktní a - aby klasifikace byla úplná - všechny varianty téhož nadtypu dohromady dají úplný popis všech možností. Jedině za těchto podmínek lze hovořit o klasifikaci úplné a minimální a potažmo i garantovat, že to, co tato klasifikace popisuje, lze vždy vyložit jednoznačně. Je zřejmé, že za těchto podmínek lze v datovém modelu vidět pouze takové podtypy entity, jejichž existence je zcela universální, tj. nesporná a bez výjimek. Tak například by bylo lze připustit, že ti učitelé, kteří nejsou pouhými lektory, mají v rámci své práce i vědeckou činnost, tedy že podtypy Učitel a Vědec nejsou zcela disjunktní. V takovém případě bude nutné považovat výše uvedený popis za nesprávný a nahradit jej jiným - například (viz Obrázek A 8) :



Obrázek A 8 Vícestupňová generalizace v datovém a objektovém modelu

³ V tomto smyslu je třeba si uvědomit, že jakmile něco prohlásíme za objekt, uvažujeme jeho jednoznačnou identitu a představa, že se skládá z jakýchsi pod-objektů, je pak zcela irrelevantní. Tyto pod-objekty by totiž pak musely mít jakousi pod-identitu, což by dávalo smysl toliko jako vztah mezi abstraktním generickým pojmem, označujícím množinu obdobných objektů a jeho specifickými variantami – objekty. Identické by pak byly ony pod-objekty, nikoliv však generický pojem, pod nějž patří – ten žádné vlastní instance nemá, není sám objektem.. Chceme-li tedy ve strukturálním (objektovém, či datovém) modelu vyjádřit, že se tento objekt z něčeho „skládá“, je to možné jedině jako jeho vztah k ostatním objektům (vztah typu agregace), z nichž každý má svou vlastní identitu – je tedy zásadně *jiným* objektem. Primární hierarchií je zde tedy generalizace, zatímco agregaci lze vyjádřit pouze jako vztah mezi různými objekty, nikoliv však jako vlastnost objektu.

Zde se podtyp Akademický pracovník dělí podle Pracovní náplně ještě na specifické podtypy Lektor (jenom učí), Výzkumník (vědec, který neučí) a Pedagog, který kromě výuky pracuje i vědecky. Pokud bychom i nadále trvali na tom, že se jednotlivé varianty obsahově překrývají, nezbude než konstatovat, že tato klasifikace není universální a že je nutno příslušné skutečnosti vyjádřit nějakým jiným způsobem - v datovém modelu například jako vztah entity Zaměstnanec k příslušné pracovní náplni (vědecké, pedagogické, pomocné apod. činnosti), kde máme pomocí kardinality a parciality možnost specifikovat vztah vícenásobný a případně i částečný (viz podrobnější popis datového modelování).

Obrázek A 8 ilustruje ještě jeden podstatný rys generalizace. I když má generalizace více stupňů, neznamená to, že si jednotlivé varianty jsou podřízené (jak je tomu v případě agregace). Zde například podtyp Akademický pracovník, ač je podtypem, je stále ještě abstraktní entitou - tedy nemá své konkrétní výskyty/instance (jiné, nežli výskyty svých podtypů). Všechny varianty a sub-varianty jsou si tedy rovny - venkonce by bylo možné zrušit podtyp Akademický pracovník a uvažovat pět základních podtypů Pracovníka: Sekretář, Lektor, Pedagog, Výzkumník a Ostatní, rozlišované podle kombinace Funkce a Pracovní náplně (zvané například Pracovní funkce). Kvalita popisu reality by tím nijak neutrpěla.

Jak již bylo konstatováno, tato abstrakce se používá jako primární v datovém a objektovém modelu systému (je základní vlastností objektů v jejich přirozené hierarchii), zaměřujících se na strukturu reality, zatímco v procesním a funkčním modelu, které se zaměřují na chování reality a systému, je primární agregace (je základní vlastností procesů a funkcí v jejich přirozené hierarchii).

Je důležité, že tyto dva základní typy abstrakce, jakkoliv jsou si podobné, jsou vzájemně v obecné rovině neslučitelné a tím tvoří jádro základního rozporu mezi procesním (funkčním) a objektovým (datovým) modelem v analytických metodách. Tento rozpor se projevuje především v nutnosti striktně rozlišovat tyto dva analytické modely, a to právě pro jejich vzájemnou obecnou neslučitelnost – strukturální (objektové) a procesní náležitosti reality nelze modelovat současně, neboť oba modely mají odlišnou logiku – vyžadují jiné úvahy o jiných aspektech v jiném kontextu. Logika strukturálního (objektového) modelu je dána generalizací, jako primárním druhem hierarchické abstrakce v tomto modelu, zatímco u procesního modelu je jeho logika dána agregací, coby zde primárním druhem hierarchické abstrakce. To je také důvod k nutné existenci dvou dimenzí analytického modelu – procesní a strukturální (viz dále).

Princip tří architektur

Princip tří architektur (P3A) definuje způsob použití abstrakce pro vývoj informačního systému po jednotlivých vrstvách (vrstvená abstrakce). Jednotlivé vrstvy se zaměřují na 3 hlavní aspekty vyvíjeného systému: obsah, technologii a implementační/realizační specifika. Tyto hlavní aspekty vyvíjeného systému tvoří přirozenou posloupnost: ze specifikace obsahu systému vyplývají možnosti technologického řešení a konkrétní použitá technologie určuje implementační možnosti.

Návrh informačního systému potom podle P3A probíhá ve třech po sobě následujících architekturách:

- **obsahové**

Zde je vytvořen zcela obecný, čistě obsahový model systému, nezátížený ani technologickou koncepcí řešení, ani jeho implementačními specifiky. Je zde abstrahováno

od technologických a implementačních specifik řešení. Obsahový návrh určuje **co** je obsahem systému.

- **technologické**

Zde je vytvořen model systému, zohledňující technologickou koncepci řešení, tj. ve strukturovaném pojetí koncepci organizace dat (technologie souborová, stromově, síťově, či relačně databázová atd.) a technologickou koncepci jejich zpracování (jazyk 3., či 4. generace, technologické prostředky architektury client - server atd.). Technologický model stále nesmí být zatížen implementačními specifiky řešení. Je zde tedy abstrahováno od implementačních specifik řešení, obsahové náležitosti jsou dány obsahovým modelem a zde se neřeší. Technologický návrh určuje **jak** bude obsah systému v dané technologii realizován (zohledňuje „logiku“ použití zvolené technologie).

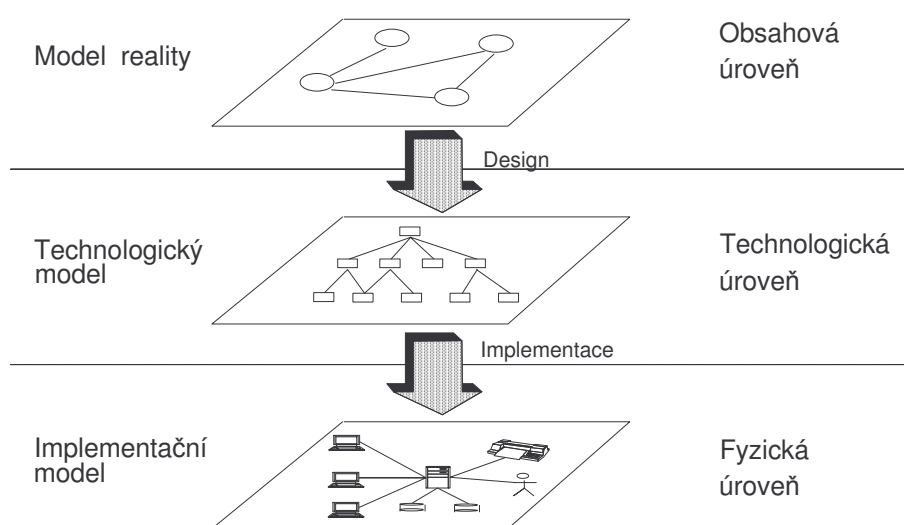
- **implementační**

Zde je vytvořen model systému, zohledňující implementační specifiky použitého vývojového prostředí (konkrétního databázového systému, programovacího jazyka a dalších prostředků, jako například vývojového prostředí GUI atd.). Není zde abstrahováno od žádných specifik řešení, obsahové náležitosti jsou dány obsahovým modelem, technologie je dána technologickým řešením, implementační návrh se tedy týká pouze implementačně specifických rysů systému. Implementační návrh určuje **čím** je technologické řešení realizováno.

Tento koncept tří úrovní modelu systému je rozpracováním použití abstrakce pro odstínění nepatřičných hledisek při tvorbě systému (viz myšlenky o smyslu modelování v následující kapitole) a současně je vidět i v obecně používaných třech základních etapách tvorby systému:

- analýza, čili stanovení obsahu,
- konstrukce (design), neboli technologické řešení a
- implementace.

Následující obrázek ilustruje různé úrovně návrhu informačního systému.



Obrázek A 9 - Princip tří architektur

Každá ze tří úrovní definuje specifickou architekturu. Každá architektura má svou specifickou logiku a specifický předmět zájmu (obsah, technologii a implementační specifiky). Pro metody to znamená:

- pro každou úroveň návrhu mít specifický **jazyk a techniky návrhu**,
- pro každý přechod z jedné úrovně do následující mít specifické **techniky přechodu** z jedné úrovně do druhé.

Jazykem návrhu je v metodách souhrn nástrojů, používaných pro příslušnou architekturu.

- Pro *modelování reality* poskytuje UML analytické diagramy, především Diagram tříd a Stavový diagram a sadu doplňkových diagramů (Use Case, Diagram komunikace objektů a Diagram posloupností), pro modelování funkčnosti systému definovaly již strukturované metody Diagram datových toků (DFD) a pro konceptuální datový model Diagram entit a jejich vztahů (ERD). Důležitou součástí soustavy analytických nástrojů je také nástroj pro popis datových struktur v systému (strukturovaný jazyk, nebo strukturní diagram).
- Pro *technologický návrh* (design systému) je to Diagram komponent a Diagram rozmístění z UML, nebo Structure Chart ze strukturovaných metod, relační datový model a další specifické nástroje pro to které technologické prostředí.
- Pro *implementační prostředí* jsou to konkrétní programovací jazyky, jazyk popisu a manipulace daty v databázi, jazyk generátoru aplikace, či integrační platforma na bázi aplikačního serveru atd.). Nástroje tím, jak jsou konstruovány a jaká jsou pravidla jejich použití, zohledňují specifickou logiku své architektury.

Techniky návrhu určují postup, který zaručí dodržení logiky příslušné architektury (normalizace datového modelu, kanonická procedura, technika analýzy událostí atd.).

Techniky přechodu mezi úrovněmi určují postup, který zaručí, že vytvářený model plně převezme nadřizenou architekturu řešení (například techniky návrhu logických datových struktur z konceptuálního datového modelu zajišťují technologicky vhodné struktury dat při zachování konceptuálního obsahu datové základny, technika transakční a transformační analýzy poskytuje základní hrubý návrh vhodné modulární programové struktury systému, vycházející z konceptuálního funkčního modelu apod.).

Informační systém, navržený v těchto třech architekturách má následující vlastnosti:

- specifikace obsahu systému je nezávislá jak na použitém implementačním prostředí, tak dokonce i na technologické struktuře systému,
- technologické řešení systému je nezávislé na použitém implementačním prostředí, určuje pouze jeho základní technologické vlastnosti (technologické prostředí),
- stejný konceptuální návrh lze realizovat v libovolném technologickém prostředí (v databázovém systému, objektové databázi, nebo souborech, s, nebo bez použití 4GL, v architektuře client-server, nebo host-terminal, v relační, či stromové, síťové apod. databázi, anebo v prostředí jazyků 3 generace, či v jazyku objektově orientovaném, případně s využitím internetové technologie (např. u jazyka Java) atd.),
- stejné technologické řešení lze implementovat v libovolných implementačních prostředcích tohoto technologického prostředí,
- jakékoliv změny implementačního prostředí se netýkají obsahu, ani technologie řešení (pokud se nejedná o implementační prostředky jiného technologického prostředí),

- jakékoliv změny technologického prostředí se netýkají konceptuálního obsahu systému.

Jak je z výše nastíněných vlastností patrné, potřeba takového rozlišení modelů není dána jenom

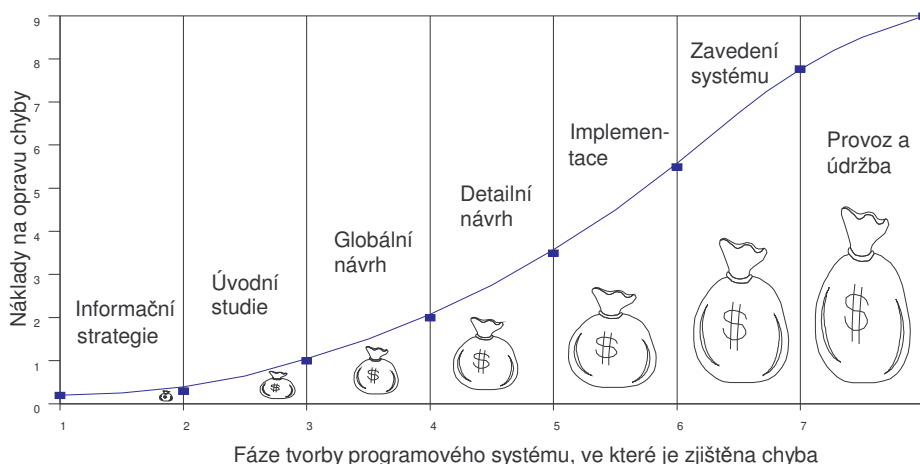
- potřebou rozdělit specifikaci systému na etapy, ale také a především
- "technologickou" potřebou nezávislosti specifikace systému na použité koncepci organizace dat a implementačním prostředí, která umožňuje klasifikovat činnosti údržby a rozvoje systému na úpravy vyplývající ze změn v reálném světě (okolí systému) a změn v realizačním prostředí.

Toto rozlišení původu změn má podstatný vliv na pružnost a rozvojeschopnost informačního systému: v případě potřeby jakékoliv změny systému je třeba nejprve identifikovat původ této potřeby - jedná-li se o změnu:

- *implementační* (například potřeba realizovat systém v nové verzi použité relační databáze, nebo optimalizovat jeho výkon programátorskými úpravami, odstranit chyby v programech apod.),
- *technologickou* (například potřeba realizovat systém, postavený na cobolovských procedurách a indexsekvencní organizaci dat v prostředí relační databáze, nebo potřeba realizovat systém v architektuře client / server s použitím vývojového prostředku GUI, aby byl umožněn jeho pružnější rozvoj v budoucnu, anebo záměr realizace tradičního intra-podnikového systému v prostředí internetu na bázi tomu příslušných technologií apod.),
- *obsahovou* (například potřeba rozšířit automatizaci na doposud neautomatizované činnosti, nebo potřeba zohlednit v systému některé změny předmětné oblasti, ke kterým v realitě došlo, anebo i potřeba odstranit chybu v systému, vyplývající z chybného pochopení reality při analýze).

Teprve po takové identifikaci původu změny, je možné přistoupit k její realizaci. Realizace čistě implementační změny nemůže mít vliv na technologické řešení, zatímco změna technologie se musí promítnout v celé implementační sféře. Změny na úrovni obsahové potom vyžadují kompletní promítnutí v technologickém řešení i s následnou implementací. Snad největší chybou, které se lze při změnách systému dopustit, je právě ignorování původu změny. To je také nejčastější příčinou příliš nízké udržitelnosti značného množství v současnosti fungujících informačních systémů, u nichž zpravidla dokonce chybí značná část dokumentace, natož aby respektovala různé architektonické úrovně návrhu systému (v lepších případech je k dispozici dokumentace na implementační úrovni). U takového systému, nechceme, nebo nemůžeme-li provést znovu jeho kompletní analýzu a návrh (to je zpravidla nemožné ať již z finančních, tak i z věcných důvodů), nezbyvá než změnu provést naprogramováním "záplaty", aniž bychom byli schopni dohlédnout veškeré její důsledky. Pokud se taková změna týká základní funkčnosti systému - jeho konceptuální podstaty, je i u nepříliš složitých systémů téměř jistota, že se dříve, nebo později projeví v chybném chování celého systému. Proto jsou takové systémy zpravidla považovány za prakticky neudržovatelné.

Tuto situaci ilustruje následující obrázek růstu nákladů na regulérní opravu chyby v různých fázích vývoje systému.



Obrázek A 10 Náklady na změny v projektu

Různé pohledy na vyvíjený systém

Princip plošné abstrakce - různých pohledů na strukturu systému, představuje základní přístup k modelování při analýze a konstrukci informačního systému.

Navrhovaný informační systém má (s přihlédnutím k nutnosti odlišovat strukturální a behaviorální aspekty systému, zmiňované u popisu principu abstrakce (viz výše)) následující čtyři podstatné dimenze:

- *strukturální - objektovou dimenzi (popis složení)*, informační systém odráží strukturu (uspořádání) reálného systému, jehož je modelem,
- *behaviorální - procesní dimenzi (popis chování)*, informační systém odráží chování (procesy) reálného systému, jehož je modelem,
- *funkční dimenzi*, určující funkcionalitu a chování jeho samotného, jako modelu reality.
- *technologickou dimenzi*, určující strukturu technologické realizace systémových funkcí, jejich časových návazností a datových struktur.

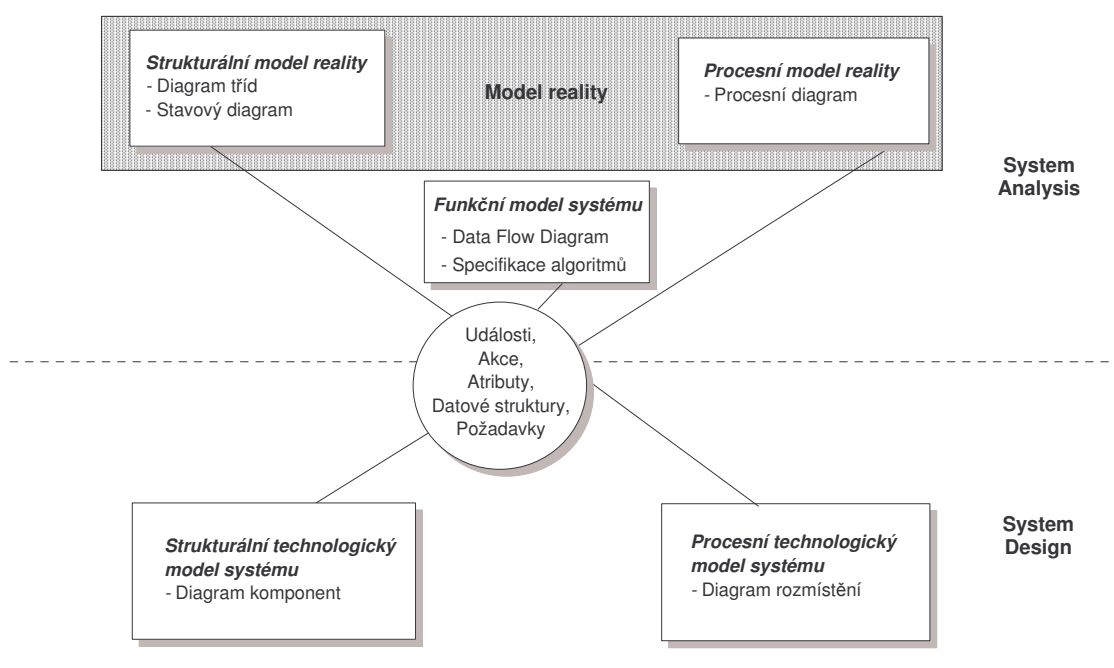
Jednotlivým dimenzím informačního systému odpovídají příslušné druhy modelů, které jsou při jeho specifikaci vytvářeny (viz Obrázek A 11):

První dvě dimenze pokrývá obsahový model reálného systému (reality), funkční model je doplňkem modelu reálného systému – modeluje funkčnost informačního systému jako modelu reality. Technologický model představuje realizační podobu informačního systému na úrovni použité technologie. Technologická dimenze se kryje s technologickou architekturou systému, popisovanou v předchozí kapitole.

Konkrétně popisuje Obrázek A 11 následující vytvářené modely:

- *Model tříd spolu se stavovým modelem klíčových tříd* popisuje realitu v objektové dimenzi – popisuje klíčové relevantní objekty (třídy objektů) reality a podstatné vztahy mezi nimi, které jsou důležité pro vytváření IS. Model je vytvářen prostřednictvím diagramů UML – *Diagram tříd* a *Stavový diagram*.
- *Procesní model* reality popisuje realitu v procesní dimenzi – popisuje klíčové procesy a jejich produkty, jako reakce na klíčové události reality, které musí vytvářený IS pokrývat. Model je vytvářen prostřednictvím *Diagramu procesů*.

- *Funkční model* popisuje informační systém z hlediska jeho funkcionality – popisuje klíčové funkce (jednotky chování) systému a podstatné vztahy mezi nimi v podobě datových toků. Model je vytvářen prostřednictvím *Diagramu datových toků* (v této metodice realizovaném jako specifický profil diagramu tříd).
- *Strukturální technologický model systému* je jedním ze dvou modelů technologické dimenze (konstrukce – designu systému) - popisuje uspořádání informačního systému jako struktury komponent a jejich vzájemných vazeb. Model je vytvářen prostřednictvím diagramu UML – *Diagram komponent*.
- *Procesní technologický model systému* je druhým ze dvou modelů technologické dimenze (konstrukce – designu systému) - popisuje rozmístění jednotlivých „logických procesů“ informačního systému na jeho „fyzických procesorech“. Model je vytvářen prostřednictvím diagramu UML – *Diagram rozmístění*.



Obrázek A 11 Různé úhly pohledu na IS

Procesní a strukturální model tvoří dvě základní složky modelu reality, funkční model je také obsahovým modelem (nezatíženým technologickými, ani implementačními specifiky řešení), ale již pouze informačního systému (nikoliv reality). Tyto tři modely jsou vytvářeny v rámci činností *analýzy systému* a představují *obsahovou úroveň modelování* v rámci principu tří architektur (viz Obrázek A 9).

Oba technologické modely (strukturální i procesní) jsou vytvářeny v rámci činností *konstrukce (designu) systému* a představují *technologickou úroveň modelování* v rámci principu tří architektur (viz Obrázek A 9).

A.1.2 Princip modelování

Model znamená

1. *Formální vyjádření zkoumaného jevu (systému) sloužící jako vyjádření skutečnosti.*
2. *Zjednodušené zobrazení určitého jevu (systému) pomocí vhodných zobrazovacích prostředků znázorňujících pouze ty rysy, jež jsou podstatné z hlediska cíle, který při konstrukci modelu sledujeme.*
3. *Reprodukce charakteristik určitého objektu na objektu jiném, zvláště vytvořeném pro jejich studium.*

Z předchozích kapitol a z výše uvedených definic pojmu model vyplývá, že základním principem návrhu informačního systému, který je společný jak strukturovaným, tak objektovým, a obecně jakýmkoliv jiným metodám návrhu IS je *princip modelování*. Jako základní princip byl historicky prvně předjímán především v teoriích datové analýzy a datového modelování, například v [Chen, 1976], nebo v [Martin, 1988] apod., brzy je však toto hledisko přijato i ve "funkční" větvi metod, především v pracích E.Yourdona. Tento princip je také dobře viditelný v základních východiscích standardního modelovacího jazyka UML.

Obecně vzato modelem vždy rozumíme *abstraktní obraz reality (reálného světa)*. Na této obecné úrovni se v charakteristice pojmu model zcela shodují všechny přístupy k tvorbě IS. Podstatné rozdíly v pojetí různých přístupů však nacházíme při bližším pohledu na to, *co* má být obsahem modelu a *co* ne (čili *co* a *od čeho* v reálném světě se vyskytujícího abstrahovat) a *proč*.

V [Yourdon, 1989] najdeme "funkční" charakteristiku principu modelování a také jeho odůvodnění. Funkční přístup chápe smysl modelu reálného světa především v tom, že obsahuje *souhrn stavů reálného světa a změn těchto stavů* - v reciprokém pohledu jde vlastně o *souhrn událostí* v reálném světě, vedoucích ke změnám stavů. Ke změnám stavů v modelu dochází prostřednictvím *operací* (to je abstraktní obraz událostí). Operace jsou podle různých hledisek a především s respektováním Top-Down principu sdružovány do vyšších celků - funkcí. Z Top-Down principu vyplývá, že funkce je pojem relativní, tedy funkce se může skládat z funkcí, a/nebo je součástí vyšší, hierarchicky nadřazené funkce. Funkčním modelem reálného světa je potom systém funkcí a jejich vzájemných vztahů, přičemž elementární podobou funkce je operace, coby abstrakce reálné události.

Smyslem modelování ve funkčním pojetí je podle Yourdona:

- použití abstrakce, která umožňuje:
 - odhlížet od nepodstatných rysů reality (implementačních charakteristik, organizačních celků a činností, které se netýkají přímo informačního systému atd.) a tím zjednodušit úlohu analytika systému,
- formalizací pojmů vytvořit prostředek dorozumění mezi odborníky rozdílných profesí analytikem systému a odborníkem v dané oblasti činností reálného světa,
- možnost provádět relativně lacino a bez následků změny v modelu, které provádět přímo v reálném světě by bylo příliš nákladné, nebo neuskutečnitelné. Potřeba neustálých změn modelu vyplývá jednak z neustálých změn reality a jednak ze samotného procesu

zkoumání reálného světa, jehož se účastní řada lidí, z nichž každý má jiný úhel pohledu a tím nutně vidí stejné věci různě a vždy mají problémy si vzájemně porozumět. Z hlediska "datového" přístupu je podstata modelu jiná. Zatímco funkční pohled zajímají především události, stavy atd., datová analýza se zaměřuje na *vlastnosti (atributy)* reálného světa, jejichž abstraktním obrazem v informačním systému jsou *data*. Při snaze o uspořádání atributů reálného světa si datová analýza již nevystačí s prostou představou hierarchické top-down struktury (jak je tomu u funkčního přístupu), ale nutně se dostává k představě *objektů (entit)* reálného světa, jimž vlastnosti/atributy přiřazuje. Datovým modelem reálného světa je potom systém entit, charakterizovaných jejich atributy, a jejich vzájemných vztahů. Smyslem modelování z hlediska datového přístupu je především formulovat ideální (konceptuální) podobu uspořádání dat v informačním systému, která je věrným obrazem reálného světa. Datové položky jsou sdružovány do skupin podle objektů, jimž atributy náleží, vztahy mezi jednotlivými objekty jsou rovněž vyjadřovány položkami (atributy vztahů). Základním axiomem datového modelování je, že takováto podoba datové základny informačního systému zaručuje:

- jednoznačnost datových položek (každá položka má jasný význam, vyjadřuje buď atribut entity, nebo vztahu),
- konsistenci dat v informačním systému, tím, že omezuje redundance dat na technologické minimum.

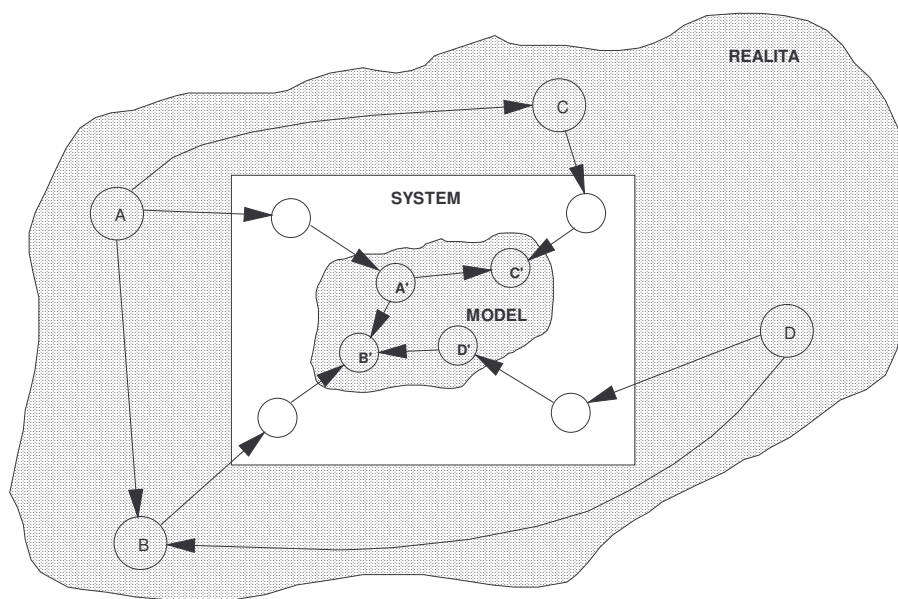
Ostatní argumenty, vyjadřující smysl modelování tak, jak byly popsány u funkčního přístupu, platí i pro datový přístup.

Obecná charakteristika modelu

Bez ohledu na použité hledisko při tvorbě modelu reálného světa (funkční/datové) lze vidět obecné charakteristiky každého vytvářeného modelu:

- model je formulován jako *systém*, tedy souhrn prvků a jejich vazeb. Konkrétní povaha prvků i vazeb je dána použitým hlediskem (data/operace),
- zvláštní význam v modelu zaujímají jeho *hraniční prvky*, tedy ty prvky, které mají vazby s okolím systému (modelu). Těmito prvky je definována *hranice systému*, tedy jeho *kontext*. U funkčního přístupu je tato hranice zajímavá především proto, že pomocí ní jsou definovány *vstupy a výstupy* systému, které jsou chápány jako *podněty a reakce* systému na ně.
- *obsah* modelu (souhrn jeho prvků a vazeb) je vždy *objektivní*, tedy každý prvek modelu musí odpovídat některému objektu reálného světa. Datová analýza (a všechny objektové přístupy) hledá v realitě přímo skutečné objekty (entity) [Chen, 1976], Yourdonova funkční analýza hovoří o požadavku *esenciality* konceptuálního modelu [Yourdon, 1989]. Každá metodika má vypracovány konkrétní techniky zajišťující objektivnost modelu, nejlepšími představiteli jsou techniky normalizace v oblasti datové analýzy, nebo technika analýzy událostí v Yourdonově strukturované analýze,
- vnitřní *struktura systému* (uspořádání prvků) je vždy poplatná struktuře reálného světa. Tato poplatnost je nejvíce viditelná na konceptuální úrovni, na ostatních úrovních je zkrácena zohledněnými specifiky (např. implementačními), vždy však struktura systému vychází ze struktury reality (to je zajištěno technikami odvozování logického modelu z konceptuálního a implementačního z logického).

Obecně princip modelování ilustruje následující obrázek.



Obrázek A 12 Princip modelování

Systém na obrázku (pochopitelně je jím míněn IS) je součástí reality se specifickým úkolem: poskytovat pravdivý a (svým způsobem) úplný její obraz. Je součástí reality v tom smyslu, že je na její jednotlivé prvky (A,B,C,D) informačně napojen (potřebuje o nich informace). Systém tak tyto prvky reality modeluje - každému jednomu reálnému prvku odpovídá jistá část obsahu systému (A',B',C',D'). Systém však modeluje také vztahy mezi nimi. A právě v tom spočívá jeho půvab: zákonitosti vztahů mezi reálnými prvky (občas se jim říká „business rules“), které jsou tvůrci systému schopni vložit do jeho obsahu, způsobují schopnost systému odvozovat informace o stavu reality z pouze základních informací o stavu jejích prvků.

Kdo by za těchto okolností ještě pochyboval o tom, že základním úkolem metod a technik pro návrh IS musí být v maximální míře umožnit tyto zákonitosti poznat, popsat a s pomocí dané (informační) technologie realizovat! Úkolem nástrojů je poskytnout způsob popisu těchto zákonitostí takový, aby na jedné straně zřetelně odpovídal reálné povaze prvků (aby v modelu byla realita co nejvíce vidět), na straně druhé umožňoval co nejjednodušší (nejpřímější) realizaci pomocí IT. Je zřejmé, že při takovéto syntéze ohně a vody se neobejdeme bez abstrakce. Ale to už jsme u tvorby modelů (viz dále).

Závěrem se k tomuto obrázku ještě sluší dodat, že ne všechny prvky systému jsou přímo modelem reality (jsou zde vidět i „pomocné“ prvky, zajišťující transformace z nechutně technologické podoby vstupních dat do podoby vhodné k použití těchto dat pro modelování dění venku). Rovněž je třeba počítat i s tím, že ve skutečnosti je systém vždy aktivním prvkem reality (nejen, že ji pozoruje, ale též ovlivňuje svými výstupy - byť se tak děje prostřednictvím chudáka uživatele). Naštěstí lze na konceptuální úrovni od tohoto faktu, alespoň místy, s úspěchem abstrahovat. V celkovém postupu vývoje IS je však třeba s tím počítat.

Způsob modelování prostřednictvím abstrakcí

Při modelování reality se mnohé metodiky uchylují k využití různých výrazových prostředků. Tyto modely se nazývají **sémantické modely**. Základním prostředkem modelování sémantiky (významu) je výše zmiňovaná **abstrakce**. Velmi dobrý přehled abstrakcí používaných v těchto modelech dává [DEL1]. V kap. 4 jsou zde diskutovány tyto modely: ER, FDM,

SDM, SAM*, IFO, RM/T, GEM , Format a LDM. Přehled abstrakcí používaných v těchto modelech shrnuje následující tabulka. Tabulka uvádí základní způsoby použití abstrakce při modelování reality bez ohledu na obecnou klasifikaci abstrakcí, uvedenou v této kapitole poněkud výše, s níž se tento výčet vzájemně proplétá. Je to tím, že zde není hlavním zájmem abstrahování samo o sobě, ale jeho použití k modelování reality.

Třídy a entity	Entity ⁴ sdílející společné charakteristiky jsou sdružovány do tříd. Existence entity je závislá na její příslušnosti k nějaké třídě. Bez této příslušnosti nemůže entita existovat.	
Agregace	Agregace je proces spojování více entit do entity vyšší úrovně.	
Asociace	Asociace nevytváří novou entitu, nýbrž jen spojuje více entit dohromady.	
Atributy	Formálně je atribut dvousměrný vztah mezi dvěma třídami entit. (jednohodnotový nebo více-hodnotový). Intuitivně se atributy využívají pro reprezentaci charakteristik tříd entit.	
Seskupení (Grouping)	Využívá se pro vytvoření entity z (konečné) množiny entit s danou strukturou. Na rozdíl od třídy není entitou na meta-úrovni.	
Specializace a generalizace	Specializace je zjemněním třídy do podtřídy. (Může být buď explicitní /Zaměstnanec/ nebo odvozená /Mladá osoba/). Generalizace je sjednocením více tříd do jedné /Hist. budova/. Generalizace je sjednocením více tříd do jedné /Hist. budova/	

⁴ Pojem „entita“ je zde použit ve zcela obecném smyslu, nikoliv jako datová entita, známá z oblasti datového modelování. Ve světle dneška by jí snad více slušel název „objekt“, tento je však obecně přesnější.

K uvedeným abstrakcím je potřeba poznamenat:

Koncept třídy

V [Delobel, 1995] se uvádí, že pro instanci třídy jsou podstatné dvě věci. Za prvé zmíněná skutečnost, že entita musí náležet nějaké třídě, aby vůbec mohla existovat a za druhé to, že každá entita má svoji identitu bez ohledu na hodnoty svých atributů. V relačním modelu splývají dvě entity, jejichž všechny atributy (atributy primárního klíče) mají stejnou hodnotu, v jednu. V konceptu tříd a instancí jsou dvě naprosto shodné instance stále dvěma instancemi.

Asociace

Vztah mezi dvěma entitami je jedním ze základních vyjadřovacích prostředků všech modelů. Je zajímavé, že tento fakt nebývá nijak významně reflektován v programovacích jazycích nebo databázových systémech. Vztahy mezi entitami se většinou na těchto nižších úrovních vyjadřují pomocí odkazů. Odkazy mají nejčastěji podobu atributu z oboru hodnot identifikátorů entit (pointery, primární klíče). Vztahy M:N je pak třeba vyjadřovat pomocí vazebních entit. O n-árních vztazích a vztazích s atributy ani nemluvě. Problém definice vztahů neřeší ani objektově orientované jazyky, ani v nich neexistuje žádný koncept explicitně vyjadřující vztah.

Seskupení

Rozdíl mezi třídou a skupinou je uveden v tabulce: „*třída je konstrukt na meta-úrovni*“, [DEL1]. Třída je něco víc než jen množina instancí této třídy. Seskupení (grouping) je právě jen množina entit, které spojuje nějaká charakteristika. Další co se v [Delobel, 1995] uvádí, je rozdíl mezi seskupením a vícehodnotovým atributem. Seskupení je určitá entita o níž lze hovořit, například personál určité budovy. Vícehodnotový atribut (z oboru identifikátorů) pouze definuje vztah mezi dvěma entitami, například mezi člověkem pracujícím v budově a touto budovou. Nevytváří se tedy takto žádná nová entita, např. uskupení lidí, o nichž by bylo lze hovořit jako o personálu budovy.

Generalizace a specializace

Generalizace a specializace je velmi široký pojem, pod nějž lze ukrýt nejrůznější vztahy tříd. Z [Delobel, 1995] je přebrána následující klasifikace typů generalizace a specializace: Třída se vyděluje (specializuje) z jiné třídy na základě hodnoty nějakého jejího atributu. Příkladem je třída *Mladá_osoba*, do níž náleží každá instance třídy *Osoba* mladší třiceti let automaticky.

Klasická specializace, kdy podtřída je speciálním případem rodičovské třídy, a je bohatší o nějaké vlastnosti. Například *Zaměstnanec* je *Osoba*, u níž se navíc sleduje, kde pracuje a jaký má plat. To, do které třídy bude instance patřit je specifikováno uživatelem při jejím vytváření. Uživatel se musí rozhodnout, zda vytváří pouze *Osobu* nebo *Zaměstnance*. Třída je definována z jiných tříd na základě množinových operátorů. Příkladem budiž třída *Historická_budova*, která sjednocuje třídy *Muzeum* a *Palác*. Systém nemůže obsahovat žádnou entitu třídy *Historická_budova*, vždy půjde buď o *Muzeum* nebo *Palác*.

Třída *Palác_muzeum* je třída, vzniklá takzvanou vícenásobnou dědičností a může sloužit například pro vyjádření faktu, že některé paláce slouží jako muzea. V tomto případě skutečně mohou existovat entity této třídy a systém musí řešit případný konflikt atributů rodičovských tříd.

V [Delobel, 1995] se rozlišují dva základní typy sémantických modelů podle toho, zda upřednostňují modelování pomocí atributů (jedno a více hodnotových) nebo pomocí seskupení a asociací. Oba dva (více méně ekvivalentní) druhy sémantických modelů ovšem směřují k objektově orientovanému přístupu.

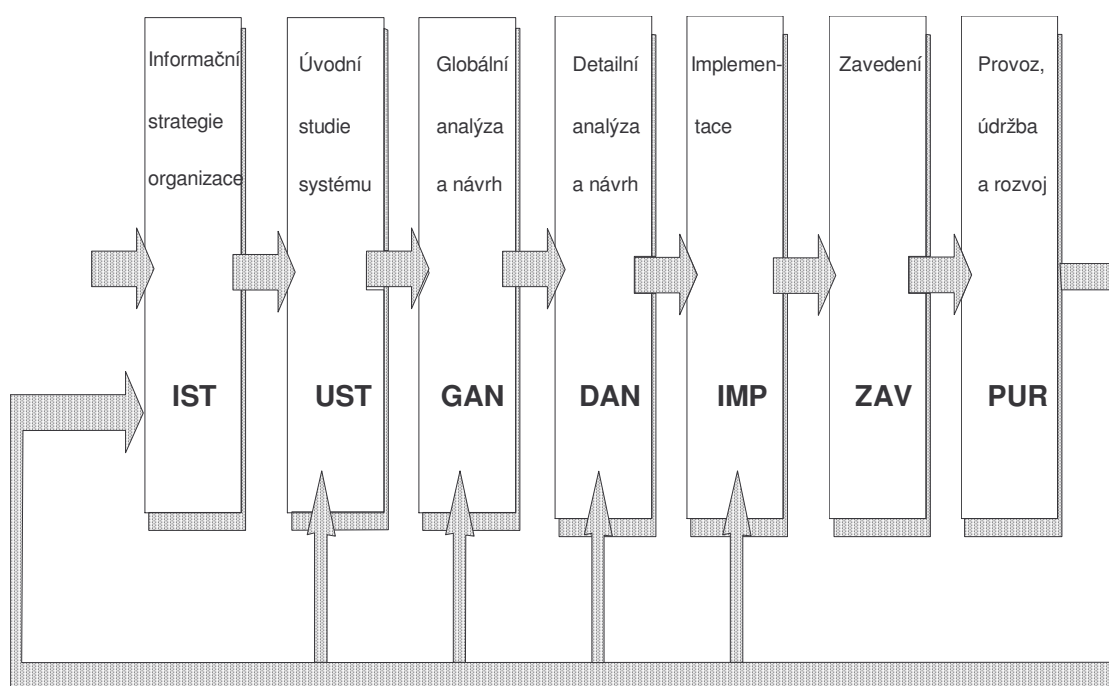
Pochopení a správná aplikace principů metod analýzy a návrhu IS je tím nejdůležitějším, co aktivní zvládnutí těchto metod vyžaduje. Vývoj jednotlivých podrobnějších součástí metod - nástrojů, postupů, technik atd. - není a nikdy nebude zcela uzavřený. Vývoj přináší další techniky, další rozšiřování základního aparátu nástrojů, nové pohledy na různé aspekty návrhu informačního systému. Veškeré tyto změny a doplňky vždy vycházejí ze základních principů, které jsou také měřítkem jejich správnosti a smysluplnosti. Kromě toho je pro aktivní zvládnutí metod jednotlivcem nezbytné jejich přizpůsobení konkrétním podmínkám použití - konkrétním charakteristikám navrhovaného systému, používaným normám, zvykům a zkušenostem. Aby toto přizpůsobení bylo v rámci metod regulérní, musí být ve shodě s jejich základním smyslem - základními principy. Totéž platí pro veškerá konkrétní rozhodování v procesu návrhu systému - rozhodování v tak specifických situacích, pro které již metody nenabízejí žádnou techniku, nebo vzor. Základním *měřítkem smysluplnosti a správnosti* každého takového rozhodnutí je právě *shoda se základními principy* metod.

V následujícím textu se často budeme k základním principům obracet tak, jak se budou projevovat v jednotlivých popisovaných podrobnostech metod.

A.2 Životní cyklus vývoje IS

Základním východiskem veškerých úvah v metodikách je představa o struktuře a obsahu tzv. **životního cyklu informačního systému** (viz Obrázek A 13 Životní cyklus informačního systému). Od této představy se potom odvíjí a k ní se váže veškerý další obsah metodiky.⁵ Jednotlivé potřebné dokumenty, cíle a specifika řízení, metody, techniky a nástroje jsou metodikou vázány k jednotlivým etapám, fázím a krokům životního cyklu. Smysl těchto základních jednotek životního cyklu IS je dán právě rozdíly v jejich obsahu z hlediska výše jmenovaných základních dimenzí vývoje IS (jejich obsah je pro každou etapu, fázi a krok jiný). Začátky a konce etap, fází a kroků vývoje IS jsou tak základními klíčovými body postupu (tzv. „milníky„), v nichž metodika určuje způsob řízení postupu vývoje IS.

Životní cyklus IS



Obrázek A 13 Životní cyklus informačního systému

Konkrétně by měla metodika **u každé etapy** stanovit:

- Cíl etapy (proč etapa má být provedena a co je jejím výsledkem)
- Účel a obsah etapy (popis role etapy v celém vývoji systému, shrnutí činností prováděných v etapě)
- Předpoklady zahájení etapy (kdy je možné začít s pracemi v rámci dané etapy)
- Kritéria ukončení etapy (kdy je možné prohlásit etapu za ukončenou)
- Klíčové dokumenty etapy (seznam dokumentů, vyprodukovaných nebo upravených v dané etapě, které musí být schváleny vedením nebo rozhodčí komisí)
- Kritické faktory etapy (na co je třeba v etapě dávat největší pozor, faktory, které mohou způsobit problémy při vývoji)

⁵ Pojem „životní cyklus informačního systému“ je pochopitelně relativní a je každou konkrétní metodikou chápán trochu jinak (viz následující popis jednotlivých známých metodik v dalších kapitolách). Zde je důležitá především snaha každé metodiky po celistvosti a to jak v obsahu životního cyklu, tak i v jeho možných aspektech (dimenzích).

- Činnosti etapy (seznam a popis činností, které se v etapě provádějí)
- Návaznosti činností v etapě (graficky vyjádřená návaznost a souběžnost provádění jednotlivých činností etapy)

Každá etapa je rozdělena na činnosti (v různých metodikách se nazývají různě: fáze, kroky apod.), které je třeba v dané etapě udělat). **Pro každou činnost** by měl být metodikou popsán:

- Cíl činnosti,
- Postup (jednotlivé kroky činnosti) (kroky určují, co je třeba udělat v rámci příslušné činnosti, mohou probíhat i opakovaně).
- Vstupy (podklady) (z jakých dokumentů a dalších materiálů se v dané činnosti čerpá, na jaké dokumenty se navazuje).
- Výstupy (produkty) (jaké produkty a dokumenty se v dané činnosti vytvářejí nebo upravují), případně v členění (pokud v činnosti vzniká, nebo je upravován určitý klíčový dokument):
- klíčové dokumenty,
- ostatní produkty.
- Zúčastněné profese a odpovědnost (profese, které se v účastní prací dané činnosti, a za co odpovídají).
- Doporučené techniky a nástroje (techniky, které se v činnosti používají, jejich možná podpora a ostatní nástroje vývoje IS)

Je zřejmé, že smysl metodiky nespočívá ani tak v tom, že by do nejmenšího detailu popisovala postup vývoje informačního systému a veškeré jeho možné varianty, metody a techniky, jako spíše v tom, že si všímá **veškerých podstatných aspektů** tohoto procesu a že postihuje proces vývoje IS **od samého začátku až do případného úplného konce** - nemusí být tedy zcela detailní, ale musí být **úplná**.²

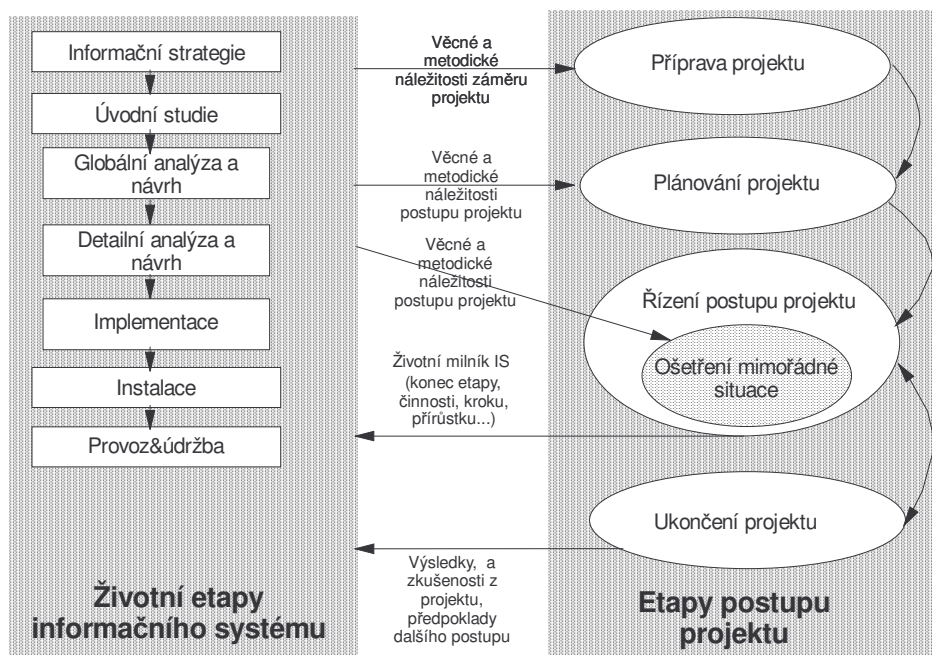
A.3 Řízení projektů vývoje IS

Úspěch procesu vývoje informačního systému je přímo závislý nejenom na porozumění obsahu prací v jednotlivých fázích a krocích, ale z celé své jedné poloviny i na způsobu řízení tohoto procesu. Vývoj informačního systému, či jeho části je vždy projektem - jedná se o akci se zjevně definovatelným cílem, časově omezenou a neopakovanou a se zjevně stanoveným a omezeným rozpočtem a se striktními požadavky na kvalitu (z hlediska uživatele je nepřípustné si představovat vývoj IS jako nějakou permanentní a nikdy neukončenou rutinní činnost s nejistým výsledkem - vždy musí být zřetelný okamžik, od kterého bude informační systém sloužit a musí být záruka, že skutečně sloužit bude). To jsou všechno aspekty, které je nutno v postupu projektu řídit.

Řízení projektu je do jisté míry samostatným oborem. U každého projektu lze vidět obecně tentýž základní životní cyklus:

- příprava a naplánování projektu
- zahájení a operativní řízení postupu projektu
- ukončení projektu

Životní cyklus IS versus životní cyklus projektu



Obrázek A 14 mezi metodikou řízení projektu a metodikou vývoje IS

Je zřejmé, že obecný životní cyklus vývoje IS, daný metodikou vývoje IS a obecné schema postupu, dané metodikou řízení projektu jsou věci, které na jednu stranu musí platit současně, ale přitom jsou na druhou stranu zcela různé (viz Obrázek A 14). Proto je vždy v procesu vývoje IS důležité přesně vědět co náleží projektu a co metodice postupu - je vždy třeba rozlišovat mezi

- (metodikou vývoje IS stanovenými) věcnými činnostmi projektu a
- (metodikou řízení projektů stanovenými) činnostmi jejich řízení.

Metodika vývoje IS stanoví veškeré podstatné obsahové náležitosti celého života informačního systému. Konkrétní způsob naplnění tohoto, metodikou předepsaného, života je potom záležitostí konkrétních projektů, které se pak řídí pravidly obecné metodiky řízení projektu. Jeden projekt zpravidla pokrývá jednu (smysluplně ucelenou) část životního cyklu projektu. Tak například má smysl uvažovat o projektu, pokrývajícím etapy analýzy, návrhu a implementace systému. Takovému projektu by měl předcházet jeden, nebo i více projektů, pokrývajících informační strategii organizace (ta platí pro veškeré informační systémy organizace) a úvodní studii IS (pro každý IS, identifikovaný informační strategií). Následovat by měl také nejspíše samostatný projekt instalace (zavedení do provozu) systému. Samotný provoz a údržba IS je pak záležitostí rutinní (je činností „permanentního charakteru“).

Permanentní činnost sama o sobě nemá běžné náležitosti projektu - jasně definovaný začátek, konec, cílový produkt a danou celkovou sumu finančních zdrojů. Jako samostatné projekty je však třeba uvažovat vývoj jednotlivých nových verzí systému (ty musí být vyvíjeny souběžně s provozem původní verze, neboť fungování systému nesmí být vývojem nové verze narušeno a omezeno). U větších systémů je též rozumné uvažovat vývoj po jednotlivých funkčních celcích (tzv. přírůstcích), které jsou uváděny do provozu postupně tak, jak vznikají, zatímco jiná funkční část systému ještě ani ve vývoji není. V takovém případě jsou i etapy analýzy a návrhu pokryty více, než jedním projektem (prakticky každý jeden přírůstek může obnášet samostatný projekt).

Obrázek A 14 je ilustrací vztahu mezi metodickou představou životního cyklu IS a metodickou představou průběhu jednoho projektu. Metodiku vývoje IS zajímá především smysl, obsah a vzájemné souvislosti jednotlivých věcných činností, které je nutno v rámci vývoje IS provádět. V této věci metodika vývoje IS s výhodou abstrahuje od problematiky detailního řízení tohoto vývoje, která sama od sebe není nijak zvlášť závislá na věcném obsahu prací. Z hlediska problematiky řízení projektu zase, na druhou stranu, není až tak důležitý konkrétní smysl a obsah prací, které se v projektu řídí, jako problematika řízení samotná. Existují tak vedle sebe dvě relativně uzavřené (a samostatné) metodiky, které je ve věci konkrétního projektu vždy třeba spojit v jedno. Aby vznikl konkrétní projekt, je třeba obecné metodické představě o průběhu (libovolného) projektu dodat konkrétní věcnou náplň prací a tím i potřebné parametry pro řízení.

- K prvnímu kontaktu obou metodik dojde na samém počátku přípravy projektu. Smyslem etapy Přípravy projektu je především formulovat věcný obsah projektu a posoudit jeho uskutečnitelnost v daných podmínkách. K tomu je třeba vědomí smyslu a vztahů mezi jednotlivými potřebnými činnostmi vývoje IS. Toto dodá metodika vývoje IS (Věcné a metodické náležitosti záměru projektu).
- V následující etapě Plánování projektu je třeba dodat záměru projektu konkrétní náplň jak věcnou, tak i časovou (rozvrhnout práce mezi jednotlivé dostupné zdroje a projekt rozplánovat tak, aby jej bylo možno řídit směrem k úspěšnému vytvoření definovaného výstupu v daném čase a v rámci daného rozpočtu). Zde metodika vývoje IS dodá vědomí potřebné následnosti činností a jejich požadavků na zdroje a jejich parametry (tj. potřebné profese a jejich kvalifikace a potřebné nástroje), včetně možností variant posloupnosti činností a vzájemné substituce zdrojů (Věcné a metodické náležitosti postupu projektu).
- Obdobná účast metodiky vývoje IS, jako při plánování projektu, je zapotřebí i v samotném postupu projektu, a to při všech změnách v plánu v důsledku ošetření mimořádné situace (zde je postup projektu chápán jako posloupnost mimořádných situací, jinými slovy: kdyby projekt běžel přesně tak, jak byl na počátku naplánován, nebude třeba se zabývat věcným obsahem

prací, pouze sledovat jejich formální náležitosti. Pochopitelně, tak to v praxi nikdy není, neboť prakticky všechny věcné parametry projektu se v jeho průběhu mění, ani naplánování projektu zpravidla nemůže být dostatečně dokonalé). Výsledkem ošetření mimořádné situace bude vždy požadavek na změny v plánu projektu, jejichž uskutečnění vyžaduje příslušné znalosti z oblasti metodiky vývoje IS (Věcné a metodické náležitosti postupu projektu).

- Ukončením projektu se vždy uzavře určitá část vývoje IS (etapa, skupina etap, přírůstek apod.) - z hlediska metodiky vývoje IS jde o dosažení jistého milníku v životním cyklu IS. Milníky, neboli místa, kde se završí jistá ucelená část vývoje IS, jsou metodikou vývoje IS věcně definovány. Jsou to všechny konce etap životního cyklu IS a dále ta místa uvnitř jednotlivých etap, která uzavírají ucelenou skupinu činností (fází, kroků atd.), která má nějaký viditelný a celistvý výstup (například ukončení analýzy uživatelských požadavků, nebo detailního návrhu programového systému apod.). Milníkům se metodika vývoje IS věnuje v tom smyslu, že jednak definuje jejich věcnou náplň (definuje příslušný výstup) a jednak potřebné řídicí náležitosti - kriteria kvality, kterou je třeba v rámci ukončení prací prověřit a potřebu účasti příslušných (nejen řešitelských, ale i uživatelských, managerských a dalších) rolí. V tomto bodě se stýká s metodikou řízení projektů, která se postupem schvalování výstupů a řízení kvality zabývá obecně.

- Hlavním smyslem etapy Ukončení projektu je závěrečné posouzení průběhu a výsledku projektu a stanovení (resp. upřesnění) předpokladů dalšího vývoje IS. Dalším výstupem, důležitým pro metodiku vývoje IS, jsou zde zkušenosti projektem získané, jejichž zobecněním vznikají příslušné úpravy metodiky vývoje IS (nově objevené souvislosti, specifikace organizace a v ní vyvíjených IS, nové možné varianty vývoje IS apod.). Metodika vývoje IS se tak každým ukončeným projektem zkvalitňuje získanou zkušeností a současně i tvoří rámec pro zachycení takové zkušenosti (dodává jí kontext - umísťuje ji do příslušných souvislostí - a formu - zkušenost je třeba formulovat „jazykem“ metodiky, pomocí jejích nástrojů a v rámci daných pravidel).

Výše diskutované vztahy mezi etapami života systému a problematikou řízení projektů jeho vývoje jsou velice důležitým aspektem problematiky vývoje IS, kterému musí být v každé metodice vývoje IS věnována dostatečná pozornost.

Z výše popisovaného vztahu mezi „věcnou“ metodikou vývoje IS a metodikou řízení projektů je zřejmé, že metodické náležitosti řízení projektů je třeba považovat za kvalitativně jiný postup, než je vývoj IS. Tak dnes pojímají metodiku řízení projektů i všeobecně uznávané standardy, jako je především PMBOK (Project Management Body of Knowledge) od společnosti PMI (Project Management Institute), nebo podobně významný PRINCE2, jako součást britských státních standardů.

Pro podporu řízení projektů existuje na trhu skupina relativně samostatných nástrojů – tzv. schedulerů – které podporují plánování, operativní řízení a mnohdy i koordinaci projektů, a to bez ohledu na to, co je jeho věcným obsahem (přesně ve smyslu výše popisované relativní samostatnosti obou problematik).

Co se týká podpory řízení projektů nástrojem vývoje IS, lze ji vidět principiálně ve dvou oblastech (co je možné, aby CASE udělal pro řízení projektů):

- podpora týmové práce,
- CASE jako nástroj řízení životního cyklu informačního systému.

V oblasti podpory týmové práce je Power Designer vybaven možností sdílet veškerá data návrhu systému v externí repository a využívat tak, kromě sdílení dat i další obecné databázové vlastnosti pro podporu týmové práce.

Role CASE jako nástroje řízení životního cyklu informačního systému se týká prakticky veškerá tato metodika. Významným rysem Power Designeru je v této oblasti především silná

podpora řízení (uživatelských) požadavků na IS, podrobněji popisovaná zejména v kapitole „Díl D Použití metodiky“ jako nástroj zajištění zpětné vazby mezi provozem IS a jeho postupným vývojem.

A.4 Konceptuální a procesní modelování

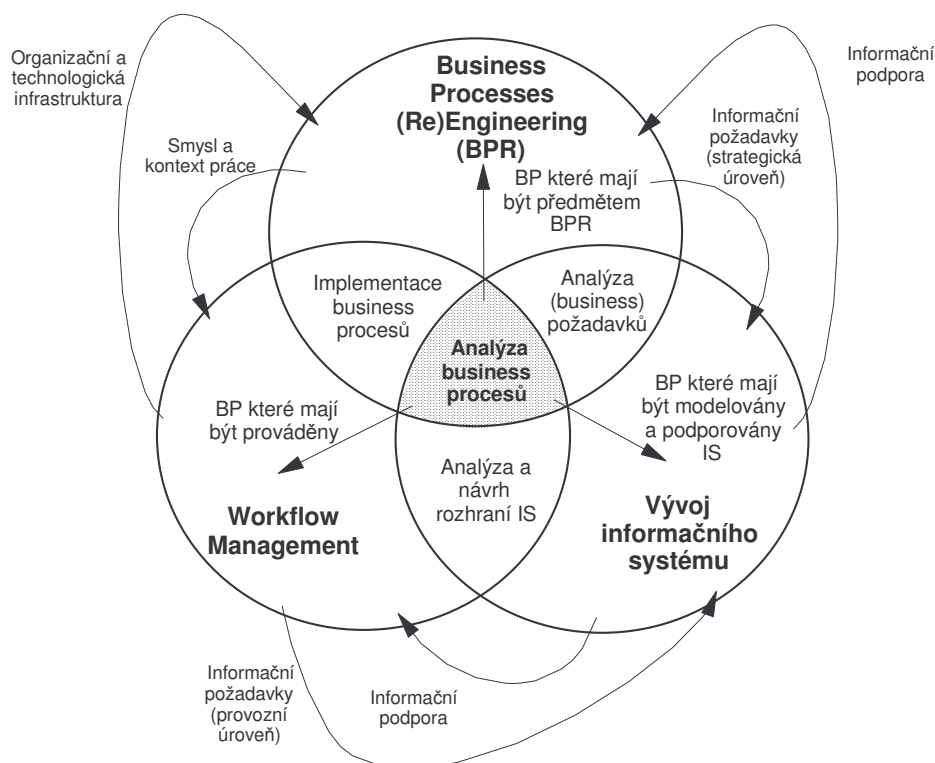
Jak je popisováno již v kapitole A.1.1 Princip abstrakce a jak je také ilustruje tam použitý Obrázek A 11, základem analytických modelů je **model reality**, sestávající ze dvou, vzájemně souvisejících modelů: **procesního** a **objektového**. Model procesů (procesní model), je tak vedle modelu objektů, jedním ze dvou klíčových východisek modelu informačního systému. Oba tyto modely ve skutečnosti tvoří základ obecného business modelu podniku (nezávislého na informačním systému a použitelného tak nejen k vývoji informačního systému).

Objektový model představuje statický model reality (businessu) - popisuje z čeho je realita složena a jaké jsou základní, *podstatné*, tedy *stálé* (*statické*) složky, čili objekty a vazby mezi nimi. Případné akce (metody) a algoritmy, vázané k objektům v jejich životních cyklech, zde mají význam též statický - jsou vnímány toliko z hlediska objektů a jejich dané struktury, resp. jsou podřízeny tomuto - statickému - pohledu.

Na úrovni modelu reality se takový model nazývá **modelem konceptuálním**. Konceptuální modelování má své kořeny v počátcích datového modelování a úzce souvisí i například s teorií relačních dat. Jedná se o modelování reality prostřednictvím základních pojmů (konceptů) a jejich vzájemných souvislostí. Konceptuální modelování je dnes široce rozvinutý samostatný obor s propracovanými nástroji a metodami, jejichž principy se odráží i v pravidlech modelování tříd objektů pomocí UML. Tato pravidla a principy jsou podrobněji vysvětlovány v kapitole B.1.1 Diagram tříd.

Oproti tomu **procesní model je modelem dynamickým** v tom smyslu, že, narozdíl od statického modelu objektů, popisuje změny - následnosti akcí, vedoucí od počátečních ke koncovým stavům procesů na základě obecného (předdefinovaného) schematu, vlivem nastávajících událostí a jejich vzájemných kombinací.

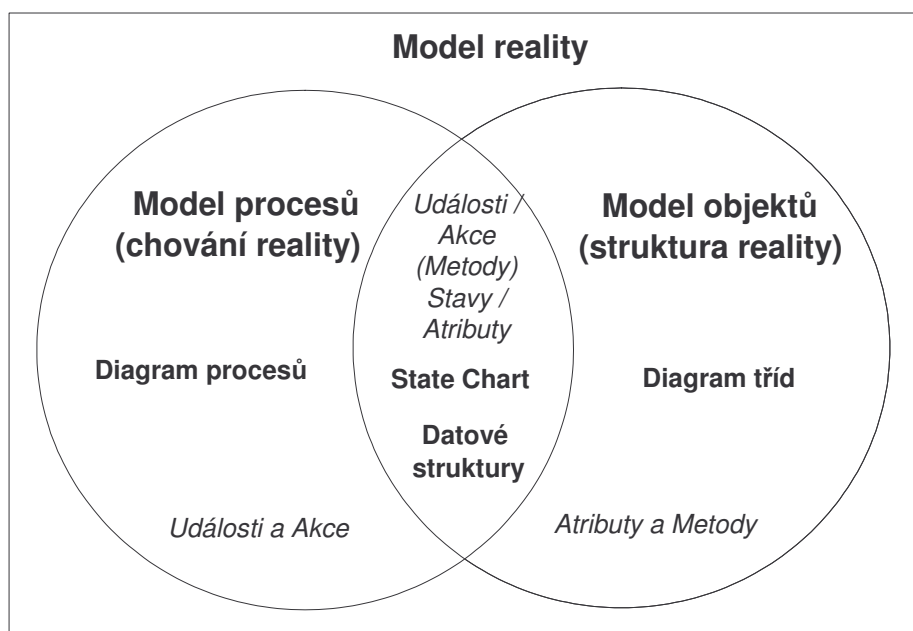
Modelování procesů a příklon k procesnímu řízení podniků je jedním z aktuálních trendů současnosti. Zatímco v teorii řízení je orientace na business procesy relativně novým (až módním) jevem, v metodikách analýzy a návrhu informačních systémů nejsou činnosti analýzy business procesů až tak nové. V těchto metodikách lze nalézt řadu různých přístupů k modelování dynamiky reality. Některé jsou zaměřeny přímo na business procesy (Lundeborg M., Goldkuhl G., Nilsson A., 1981, BSP, 1984, Turner, W.S., Langerhorst, R.P., Hice G.F., Eilers, H.B., Uijttenbroek, A.A., 1987), nebo alespoň na procesní - dynamické modelování (Yourdon, E., 1989). Obvykle jsou však v těchto metodikách činnosti modelování business procesů rozesety mezi ostatní činnosti budování informačního systému ve formě analýzy současného stavu, analýzy informačních potřeb, analýzy časových závislostí apod. Jako příklad metodiky asi nevíce orientované na business procesy viz ISAC (Lundeborg M., Goldkuhl G., Nilsson A., 1981). Skutečně novým pohledem na věc je zde potřeba v metodikách analýzy a návrhu informačního systému *oddělit činnosti modelování business procesů od ostatních modelovacích činností* (tedy modelování statické objektové struktury, jakož i modelování vnitřní dynamiky objektů). Analyzování a modelování business procesů by tak mělo být samostatnou a nezávislou činností, předcházející ostatní činnosti budování IS. Hlavním důvodem tohoto požadavku je skutečnost, že model business procesů je zcela univerzální. Je základem nejenom vývoje informačního systému, ale též implementace workflow, jakož i činností BPR (business process reengineering) - viz Obrázek A 15.



Obrázek A 15 BPR vs. vývoj IS vs. Workflow Management

Obrázek A 16 ukazuje model business procesů (BPM) a konceptuální objektový business model (BOM) jako základnu modelu informačního systému. BPM poskytuje informaci o potřebné struktuře funkcí rozhraní ve formě produktů, aktérů a zjištěných potřebných činností. BOM poskytuje informaci o struktuře reality ve formě zjištěných objektů, produktů a aktérů. Ve skutečnosti by měla být většina činností, tradičně považovaných za činnosti vývoje informačního systému, prováděna formou (a za účelem) analýzy business procesů. Konceptuální model informačního systému má dvě hlavní části. Jádrem modelu tvoří objektový model informačního systému, který představuje čistý model reality z hlediska systému. Popisuje ty objekty a vazby reality, které má informační systém podporovat (a tudíž modelovat). Dynamika reality, obsažená v tomto modelu, je zde viděna z pohledu objektů samých - jako jejich životní cykly, definující obecně platná pravidla jejich chování. Okrajová - vnější část konceptuálního modelu informačního systému představuje model chování reality z hlediska systému. Popisuje ty reálné procesy a jejich vazby, které mají být informačním systémem podporovány (a tudíž jím jsou modelovány). Řečeno jazykem informačního systému zde jde o strukturu funkcí rozhraní vstupů/výstupů.

Je zřejmé, že oba modely jsou různými pohledy z různých úhlů na totéž - na realitu jistě struktury a jistého chování. Událostmi motivované procesy změn v realitě se tak musí prolínat parametry "statických" vlastností reality (objektů, jejich atributů, s pomocí jejich akcí (metod) v rámci definovaných omezení životními cykly dotčených objektů).



Obrázek A 16 Dvě základní dimenze modelu reality

Obrázek A 16 ukazuje model (business) reality jako souhrn dvou, vzájemně propojených modelů.

Model objektů (tedy podstaty - struktury) reality sleduje základní stavební kameny, z nichž realita sestává – objekty a jejich vlastnosti (atributy). K takovému popisu existuje specifický diagram – Diagram tříd (Class Diagram), jenž je základním diagramem jazyka UML. Kromě existence objektů, jejich vlastností a vztahů mezi nimi, umožňuje diagram tříd modelovat i akce k objektům vázané (tzv. metody tříd objektů). Pomocí dalšího diagramu jazyka UML – Stavového diagramu (State Chart) lze model tříd doplnit o popis základních časových návazností mezi metodami objektů v návaznosti na události a specifikovat tak průběh a stavy tzv. „životního cyklu“ třídy objektů. Metody, vázané k objektům, jakož i procesy jejich životních cyklů, jenom upřesňují charakteristiky modelovaných objektů a vztahů mezi nimi – vymezují základní vývojové zákonitosti, jež musí chování reality respektovat (například to, že objekt Objednávka poté, co je potvrzen, již nesmí měnit některé své dané atributy apod.). Procesy životů jednotlivých objektů, narozdíl od podnikových procesů, nesledují žádný cíl, nesměřují k žádnému produktu (ledaže by cílem byl konec života objektu a cílovým produktem jeho nebytí). Model tak nepřestává sledovat svůj základní zájem – modelovat strukturu reality a její základní zákonitosti. Jedná se o model bytí, což souvisí i s pojmem ontologie, používaným pro modely tohoto typu ve znalostním inženýrství.

Model věcných procesů (tedy chování) reality sleduje specifické řazení akcí v realitě do věcných – podnikových (business) procesů. K takovému popisu je zapotřebí specifický diagram – Procesní diagram, popisovaný v kapitole B.1.2 Diagram procesů. Takový nástroj doposud není standardizován způsobem, podobným standardizaci jazyka UML. Narozdíl od procesu životního cyklu objektu je podnikový proces vždy projevem vůle - vždy směřuje k určitému cíli, produkuje měřitelný a říditelný výstup. Modelování takového procesu má tedy zcela jiné náležitosti, než modelování životního cyklu objektu (zde nás například zajímají vstupy a výstupy, jejich významové rozlišení na suroviny/produkty, nebo řídicí informace apod. – viz popis procesního diagramu v kapitole B.1.2). Cílem modelu podnikových procesů je postihnout chování reality v termínech akcí a jejich vzájemných časových návazností v závislosti na událostech a stavech procesů. Typické je, že ve svém běhu kombinují různé objekty, namísto aby se týkaly objektu jediného (ledaže by tímto objektem byl proces sám,

čímž se ovšem posouváme na vyšší meta-úroveň modelu a debata přestává mít původní smysl). Jedná se o model chování.

Každý z obou modelů popisuje realitu z hlediska určité své dimense – bytí, versus chování. V obou dimensích mají své místo procesy – buď ve významu životního cyklu, nebo věčného – cíleného procesu (business process). Z hlediska čistě „technického“ tedy jak věčné procesy, tak procesy životních cyklů podléhají stejným pravidlům – vyjadřují řazení akcí v závislosti na událostech a stavech. Rozdíl mezi nimi je toliko významový – životní cyklus je procesním vymezením určitých zákonitostí, jež musí být respektovány jakýmkoliv chováním, zatímco věčný – cílený proces je popisem tohoto chování - vyjádřením způsobu dosažení určitého cíle, vytvoření určitého produktu. Z výše uvedeného rozdílu jasně plyne, že k modelování podnikových procesů naprosto nestačí je pojímat z čistě technického hlediska. Právě jejich účelovost, cílenost, tvoří jejich podstatu.

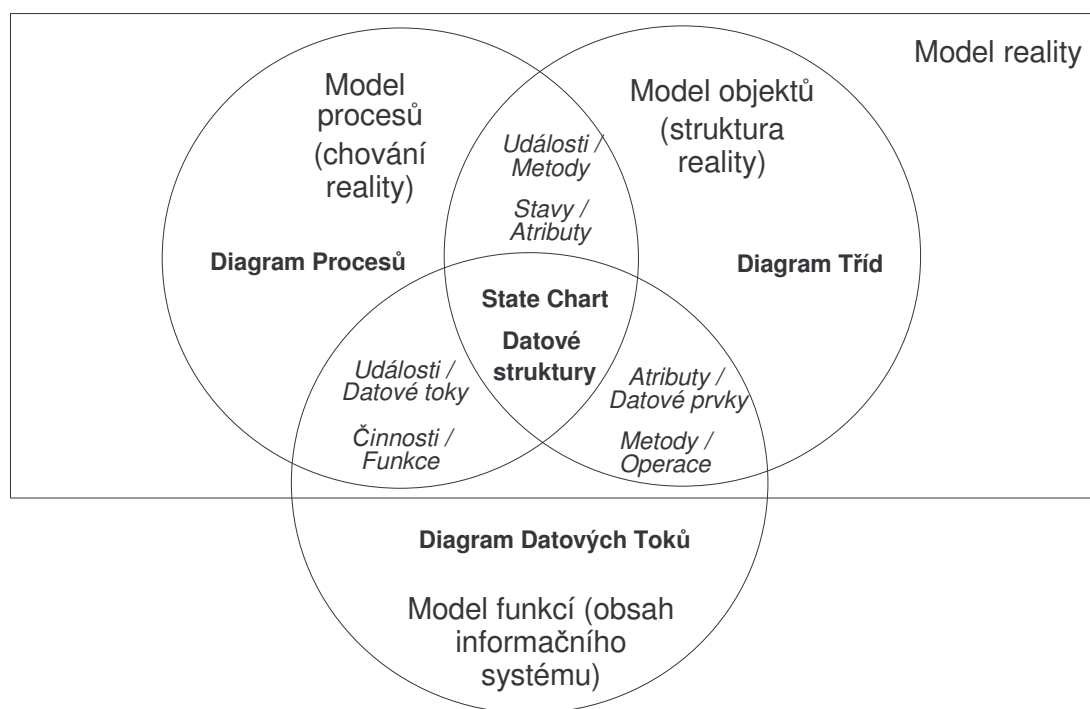
Dalším podstatným rozdílem mezi oběma dimensemi je typ hierarchické abstrakce, která je v nich primární – zatímco v modelu tříd je primární abstrakcí generalizace (třídy – objekty mohou být zobecňovány a konkretizovány, nikoliv však děleny na části), je v modelu procesů naopak primární abstrakcí agregace (procesy mohou být děleny na části – pod-procesy, nebo shlukovány do vyšších celků, jejich zobecňování však v procesním modelu nemá smysl). V každém z obou modelů je možné používat oba druhy hierarchické abstrakce, vždy však podřízeně tomu primárnímu (tak je možné například vyjadřovat jeden objekt jako shluk jiných, nic to však nemění na tom, že základním principem objektů je dědičnost. Tak je možné například popisovat různé variantní cesty procesu, nic to však nemění na tom, že základním principem procesů je jejich neomezená dělitelnost na části a z toho plynoucí relativnost rozdílu mezi pojmy proces a činnost)⁶.

Kromě rozdílů mezi dimensemi ukazuje Obrázek A 16 také, co mají společného (jejich průnik). Jsou to jednak základní procesní náležitosti (události, akce, stavy), jednak i základní strukturální náležitosti (atributy). Znamená to, že v obou modelech se pracuje jak s procesy, tak i s atributy (v procesním modelu je například důležité sledovat atributy stavu procesu a atributy jeho produktů, které se obsahově kryjí s atributy objektů, jichž se produkty týkají, nebo jimi přímo jsou). Tento průnik obou dimensí vyvolává potřebu sladění obou modelů dohromady, aby byla zajištěna jejich vzájemná konsistence (bezrozpornost). Specifikace základních souvislostí obou modelů a z nich plynoucích pravidel jejich konsistence, je dalším důležitým úkolem pro metodiku. Tyto souvislosti jsou podrobně popisovány v kapitole B.1.4 Provázání procesů s třídami objektů pomocí jejich životních cyklů.

⁶ Nerespektování těchto rozdílů je jádrem mnoha problémů slovních teorií, které tyto mají s podnikovými procesy. Je jakousi esencí chyb přliší „technokratického“ přístupu k podnikovým procesům i v rámci jejich exaktního popisu (modelování). Takto se například stále víceméně neúspěšně potýká s podnikovými procesy jazyk UML.

A.5 Funkčnost systému

Model reality, popisovaný v předchozí kapitole, není jediným obsahovým modelem informačního systému. Další nutnou složkou obsahového modelu informačního systému, která doplňuje Model reality, sestávající z procesního a objektového modelu, je *model funkčnosti informačního systému*, jak je také vidět na obrázku níže:



Obrázek A 17 Tři složky obsahového modelu informačního systému

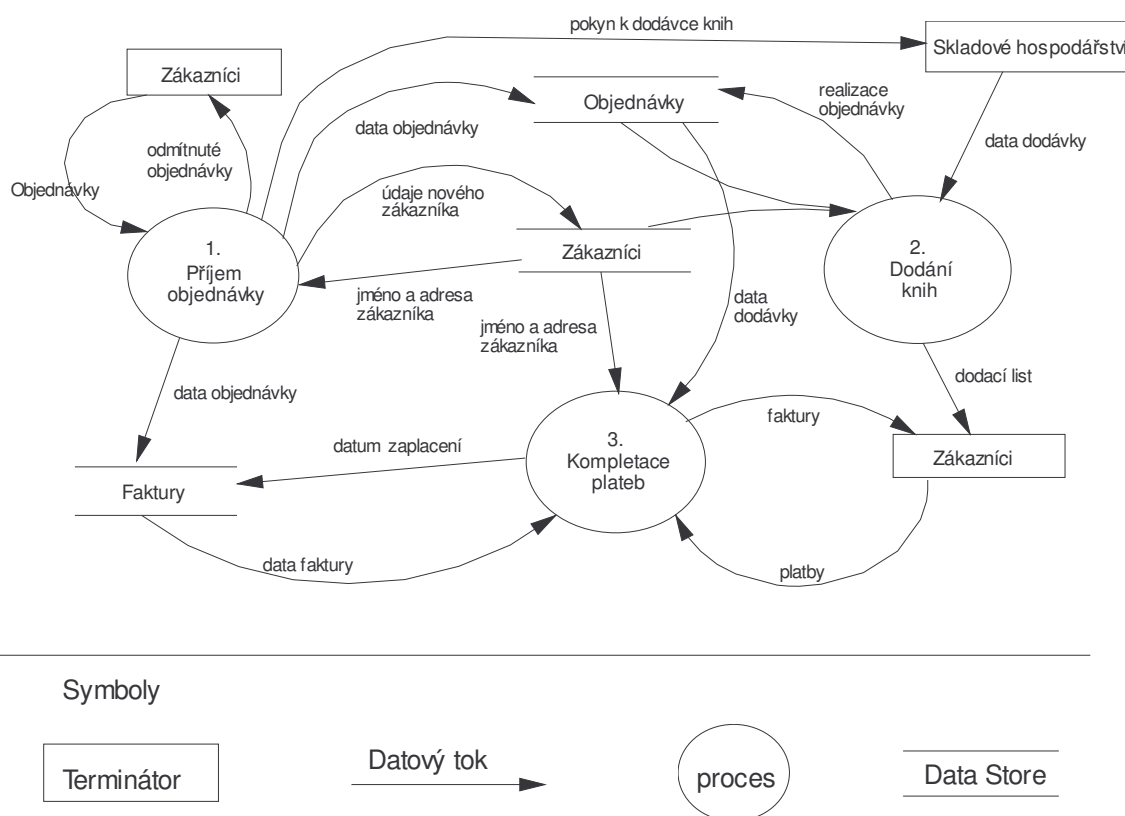
Funkčností informačního systému se rozumí obsahové vymezení jeho činnosti – tedy jaké činnosti podporuje a jak (jaká data / informace poskytuje). Je zřejmé, že – ve smyslu platnosti principu modelování – je funkčnost informačního systému také odrazem reálného systému – tedy reálných procesů, probíhajících pro potřebu, za účasti, z vůle, s cílem vytvoření, eliminace, či podpory apod. reálných objektů a jejich vzájemných podstatných vztahů. V tomto smyslu je funkční model informačního systému nutno odvodit z modelu reálných procesů a z modelu objektů. Nicméně v informačním systému samotném – jako součásti reálného systému – se procesní a strukturální náležitosti reality projevují jemu specifickým způsobem: jako data / informace o reálných jevech ve vzájemných souvislostech, vědomí jichž je vloženo ve vnitřních vlastnostech informačního systému. Informační systém přijímá ve formě svých vstupů (datových toků) informace o reálných jevech (událostech), aby je vzájemně kombinoval za účelem odvození informace z jejich vzájemného vztahu. Samotné kombinování informací o vzájemně časově nezávislých (asynchronních), avšak věcně souvisejících jevech, vyžaduje tyto informace ukládat (tedy zapamatovat si událost a čekat na budoucí další, věcně související, události)⁷. K tomu však musí informační systém realitu

⁷ Typovým příkladem může být událost „zákazník potvrdil objednávku“, na niž není vhodné reagovat dodáním zboží zákazníkovi dříve, než tento zaplatí. Je tedy třeba čekat na další navazující událost „zákazník zaplatil“ a teprve poté zboží expedovat. Informační systém musí tedy nejprve pojmout informaci o potvrzení objednávky, zapamatovat si ji a čekat na předpokládanou informaci o zaplacení, na niž pak reaguje pokynem k expedici zboží. Aby takto byl informační systém schopen fungovat, musí ovšem „znát“ celou tuto „business kauzalitu“ a mít ji v sobě „zakódovanou“.

„znát“, přesněji, musí umět předpokládat základní (esenciální) souvislosti základních (esenciálních) reálných jevů a mít tyto očekávané souvislosti ve své struktuře „zakódovány“. Obrázek A 17 popisuje prvky, v nichž se funkční model překrývá s modelem objektů (jsou to jednak atributy jednotlivých objektů, realizované v informačním systému ve formě datových prvků a jejich struktur (které tak odrážejí vzájemné vztahy objektů), jednak metody objektů, realizované ve formě operací informačního systému a jejich struktur (také odrážející vzájemné vztahy objektů)). Obrázek dále popisuje prvky, v nichž se funkční model překrývá s modelem procesů (zde jsou to jednak události, na něž jednotlivé procesy reagují, realizované v informačním systému ve formě datových toků a jejich struktur (ty odrážejí vzájemné vztahy procesů – jejich komunikaci), jednak činnosti procesů, realizované ve formě operací a funkcí informačního systému a jejich struktur (také odrážející vzájemnou komunikaci procesů)). Na obrázku je také vidět společný průnik všech tří modelů – společnými jmenovateli všech bilaterálně společných prvků jsou jednak *datové struktury* (odrážející atributy objektů a událostí a jejich esenciální vztahy) a jednak esenciální zákonitosti, odrážející esenciální návaznosti činností a akcí (dané *životními cykly objektů*).

Samotný funkční obsah IS je tedy specifickým modelem reálného systému, a to v obou jeho dimenzích. Pak je ovšem nutné *striktně odlišovat funkční obsah informačního systému od formy jeho realizace* (dané jak obecněji – technologickým prostředím, tak konkrétně prostředím implementačním). Tedy model funkčnosti IS, jakkoliv sám je specifickým modelem reálného systému, je, z hlediska informačního systému, stále modelem obsahovým – teprve z něj jsou odvozovány modely technologické a posléze implementační. Někteří autoři proto funkční model nazývají také *modelem esenciálním* (např. E. Yourdon ve svém stěžejním díle Yourdon, 1988).

Základním nástrojem modelování funkčnosti IS je Diagram datových toků (viz Obrázek A 18).



Obrázek A 18 Diagram datových toků

Diagram datových toků (resp. hierarchická soustava těchto diagramů pro celý systém) znázorňuje jak data protékají systémem funkcí informačního systému od svého vzniku (u „terminátoru“) až po finální spotřebu („terminátorem“). Tyto přirozené (rozuměj: povahou reálného systému dané) toky dat jsou na své cestě přerušovány ukládáním do „datastorů“, a to z důvodu potřeby čekat na další informace o dalších jevech, které pak bude třeba interpretovat v kontextu informací uložených.

Diagram datových toků je v této metodice realizován v rámci jazyka UML, ve formě specifického profilu tohoto jazyka na bázi modelu tříd, kde funkce, datastory a terminátory zastupují specifické stereotypy tříd, datové toky pak specifický stereotyp asociace. Přirozená hierarchická struktura diagramů je zde realizována vazbou mezi funkcí (třídou stereotypu <<function>>) a souvisejícím diagramem datových toků (diagramem tříd stereotypu <<data flow diagram>>). Detailní principy a pravidla použití diagramu datových toků, jakož i související technika návrhu funkční struktury systému zkoumáním událostí, jsou předmětem zájmu kapitoly B.1.5.

A.6 Požadavky na IS a jejich role v procesu vývoje IS

Požadavkem se obecně rozumí jednotka potřeby funkcionality nebo jiné vlastnosti systému.

Pojem požadavek může mít širší, nebo užší význam:

- uživatelský (užší) – znamená subjektivní požadavek, kladený uživatelem na systém,
- obsahový (širší) – znamená požadavek na obsah systému, obecně jakýkoliv důvod k potřebě jisté funkcionality či vlastnosti systému. Kromě fyzického požadavku uživatele může požadavek vzniknout na základě změny strategie společnosti nebo jako výsledek analýzy podle pravidel a technik metodiky, který odhalil potřebu modelovat systémem jisté reálné zákonitosti, nebo formální důvod kladený normou, či standardem apod.

V celé metodice se pojmem požadavek myslí širší obsahové pojetí požadavku, pokud není uvedeno jinak.

Obsahové požadavky, narozdíl od uživatelských, vznikají už před zavedením systému do provozu. Tyto požadavky můžeme hierarchicky členit. Nejvyšším stupněm jsou požadavky přímo vyplývající ze strategie organizace, nejnižším stupněm jsou požadavky detailní analýzy a návrhu. Mezi požadavky jednotlivých úrovní existuje příčinný vztah. Požadavek nižší úrovně by měl vždy být částečným řešením některého požadavku vyšší úrovně a naopak požadavek vyšší úrovně by měl být přesným sjednocením (agregací) požadavků nižší úrovně. Pokud tomu tak není, je velmi pravděpodobné, že některá z fází životního cyklu vývoje informačního systému proběhla neúplně nebo není v souladu s předchozí fází. Pomocí hierarchie požadavků je možné ke každé jednotlivé funkci systému vždy najít jaký má vztah ke strategii organizace - jak konkrétně určité části IS podporují strategické cíle.

Analýzou požadavků se rozumí systematický proces s cílem objevení podstaty – původu požadavku. Jedná se o analytickou činnost, probíhající od začátku analýzy systému až po ukončení provozu systému. Tato činnost spočívá v získávání, třídění (ve smyslu typů požadavků i hierarchie) a ověřování relevantnosti požadavků na systém a definování následných akcí pro uspokojení relevantních požadavků – od relativně jednoduchých změn v implementaci systému až po úpravy procesů a změnu strategie organizace.

Pro potřebu analýzy je třeba požadavky dělit na:

- **Funkční**

Funkční požadavky reprezentují požadavky na funkční obsah informačního systému. Vznikají už při tvorbě strategie společnosti a postupně jsou detailizovány až na úroveň požadavků na elementární funkce systému.

- **Technologické**

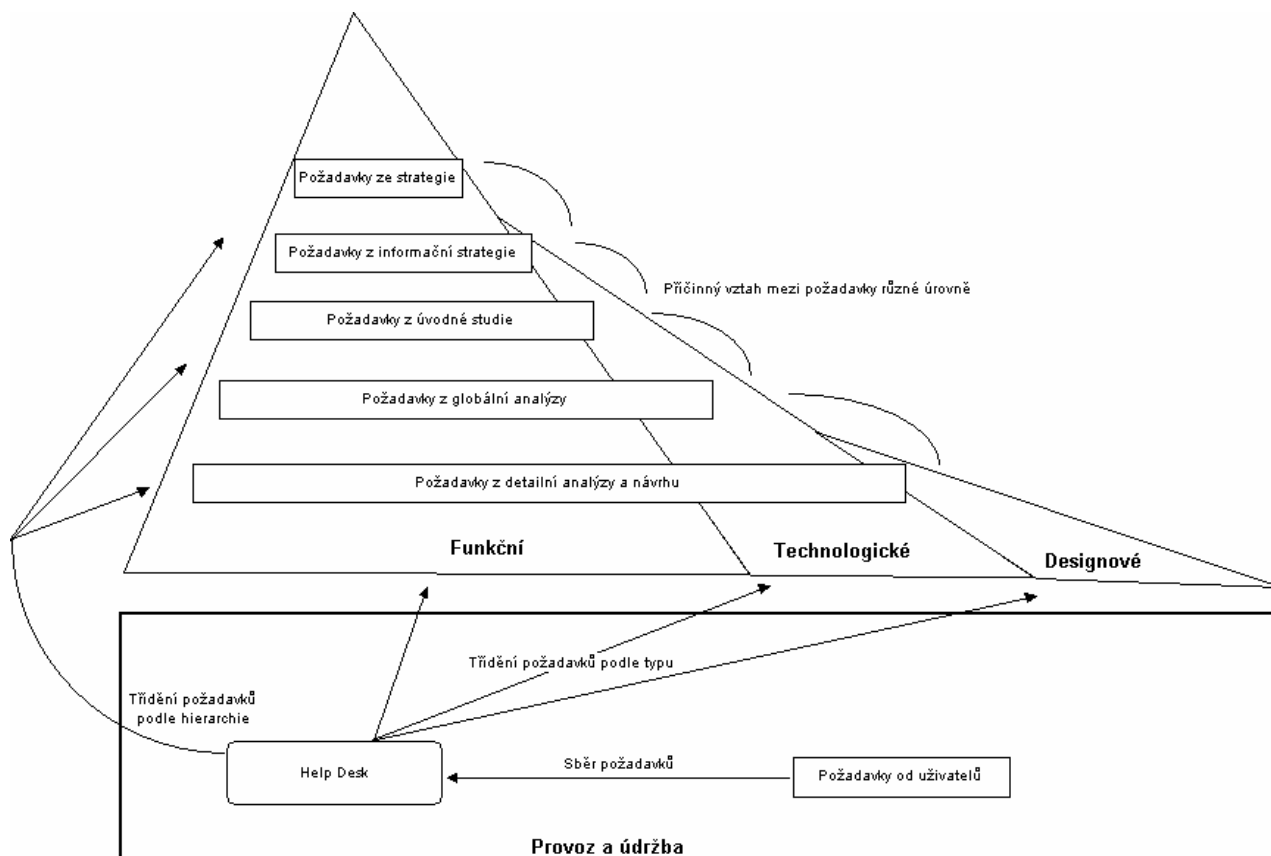
Technologické požadavky vznikají po určení technologií, které budou na realizaci informačního systému použity, např. určení operačního systému, typu databázového systému, atp.

- **Designové**

Designové požadavky se řeší v úrovni realizace systému. Sem patří například požadavky na vzhled uživatelského rozhraní, rychlost odezvy, apod

Výše uvedená klasifikace požadavků je základním nástrojem analýzy požadavků s cílem specifikace projektových záměrů pro budoucí vývoj systému. V tomto smyslu je analýza požadavků také důležitým prvkem *zpětné vazby mezi provozem informačního systému a jeho*

vývojem (vývojovými změnami). Objektivní potřeba změn a nových vývojových verzí informačního systému je signalizována právě ve formě požadavků na něj. Tyto požadavky přicházejí od vedení společnosti, poté od analytiků a vývojářů a po zavedení systému do provozu i od uživatelů. Proto je důležité ve všech fázích životního cyklu systému udržovat aktuální hierarchii požadavků a řídit jejich konzistenci.



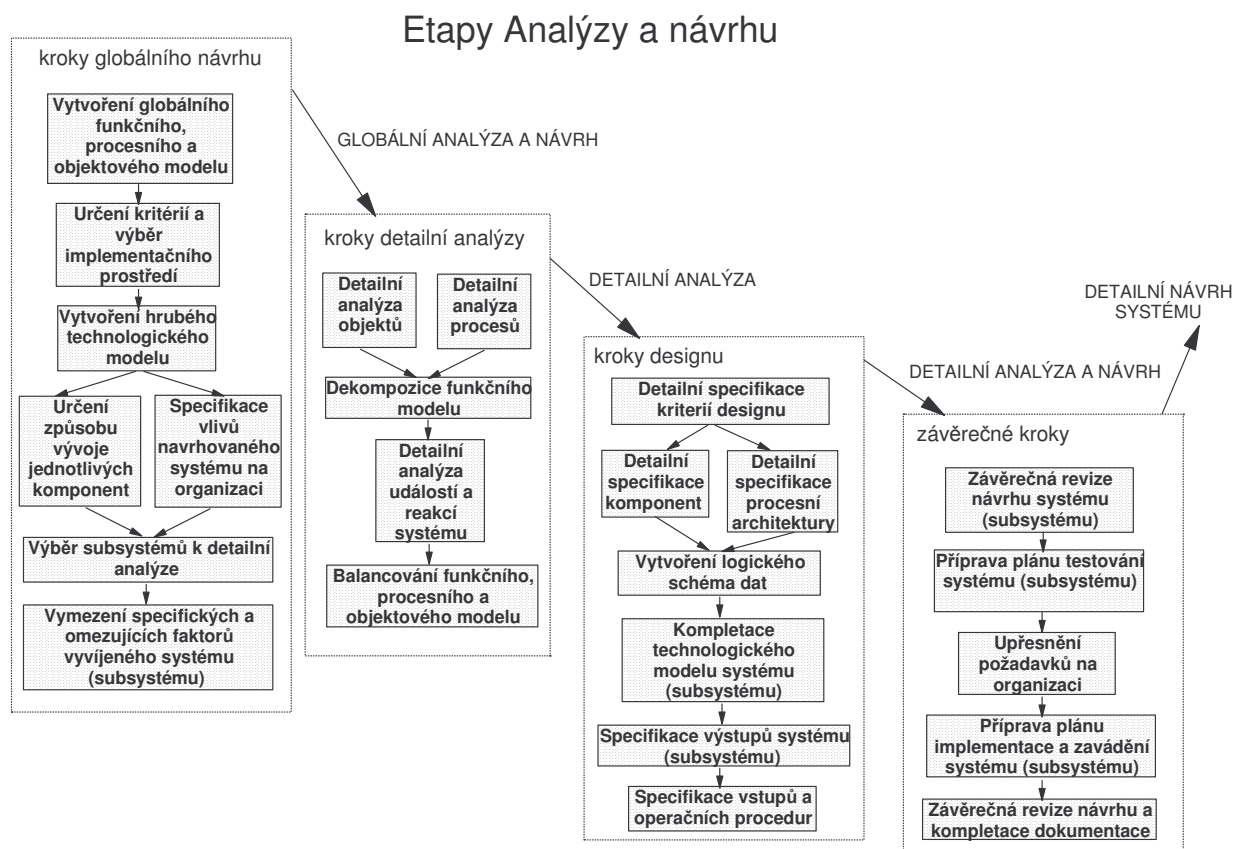
Obrázek A 19 Hierarchie a typy požadavků na IS

Povšimněme si, že takto pojatá klasifikace požadavků je v souladu se základními principy metodiky:

- hierarchie požadavků je v souladu s principem hierarchické abstrakce na bázi agregace
- dělení na tři typy požadavků je v souladu s principem tří architektur.

A.7 Podrobnější postup etap analýzy a návrhu IS

Podrobnější postup etap analýzy a návrhu, jejich rozdělení do jednotlivých skupin kroků a kroků, ukazuje následující obrázek:



Obrázek A 20 Postup etap analýzy a návrhu

V této kapitole jsou popisovány současně etapy globální a detailní analýzy a návrhu, a to zejména proto, že jejich vzájemné provázání je natolik úzké, že je zapotřebí řadu jejich společných aspektů vnímat společně.

Jak ukazuje i Obrázek A 20, v obou etapách se provádí jak analýza, tak návrh (design) systému. V etapě globální analýzy a návrhu

V následujících podkapitolách jsou stručně charakterizovány a vysvětleny jednotlivé uvedené kroky postupu.

A.7.1 Kroky globálního návrhu

Cílem globálního návrhu je především zmapování všech podstatných okolností, jimiž je dán obsah a struktura informačního systému, a to na globální úrovni, a především rozdělení systému na jednotlivé relativně samostatné funkční celky, které budou realizovány různými způsoby. Důležitým efektem takového rozdělení je právě možnost zaměřit se v detailní analýze pouze na relevantní části systému (ty, které detailní analýzu objektivně vyžadují).

Globální zaměření znamená orientaci na *celek*, a to na *úkor detailů*. Důležitým kritériem je zde tedy *úplnost* (tedy postižení *všech* podstatných faktorů, působících především na systém jako *celek*), narozdíl od podrobnosti, která je naopak klíčovým předmětem zájmu analýzy

detailní. Úroveň podrobnosti je třeba zvolit tak, aby byla co nejmenší (nejhrubší) při zachování úplnosti (aby nic podstatného nebylo vynecháno) – to je také hlavním úkolem všech použitých metod v této etapě.

Globální návrh zahrnuje jak modely analytické, tak i konstrukční (designové), obojí globálně zaměřené. Základními výchozími analytickými modely jsou:

- Globální konceptuální model business objektů,
- Globální model business procesů,
- Globální funkční model systému.

Z těchto modelů je, zohledněním specifických charakteristik technologické a implementační povahy, dále odvozen globální technologický model systému, který bude později rozpracován ve fázi detailního designu systému.

Důležitými finálními kroky globálního návrhu je jednak výběr subsystémů k detailní analýze, učiněný na základě globálního rozhodnutí o způsobu vývoje jednotlivých komponent (funkčních celků) systému, jednak vymezení specifických a omezujících faktorů pro další vývoj jednotlivých subsystémů.

A.7.1.1 Vytvoření globálního funkčního, procesního a objektového modelu

Cílem tohoto kroku je globálně zmapovat obsah informačního systému, a to ve vazbě na reálný - business systém a veškeré podstatné aspekty organizace, jako je strategie, organizační a technologická infrastruktura a další.

Základním výstupem tohoto kroku je:

- *Globální konceptuální model business objektů*, popisující existenci a hlavní atributy základních business objektů a jejich vzájemných vztahů. Existence objektů a jejich vztahů je formální definicí základních omezení (business pravidel), která bude muset informační systém respektovat – modelovat,
- *Globální model business procesů*, popisující hlavní – základní business procesy a jejich vzájemné vztahy (komunikaci). Model procesů a jejich komunikace je vymezením základních požadavků / záměrů na chování business systému, která bude muset informační systém podporovat – modelovat,
- *Globální funkční model systému*, popisující základní funkční celky systému. Globální funkční model poskytuje především dekompozici systému na funkčně vymezené subsystémy. Jednotlivé subsystémy musí být funkčně homogenní (pokrývat věcně související oblast činností business systému) a relativně uzavřené (vyžadovat minimální komunikaci s jinými subsystémy).

V tomto kroku je vhodné použít analytické metody, zaměřené na zmapování základních globálních souvislostí reálného systému, které mají kritický vliv na informační systém jako celek. Jedná se především o

- strategii organizace,
- její podnikové procesy,
- organizační strukturu
- a s tím související informace o
 - jevech (událostech) a
 - objektech v rámci organizace existujících a působících.

Jednou z nejvhodnějších takových metod je Business System Planning (BSP), podrobněji popisovaná například v Řepa, 1999, nebo Řepa, 2005.

A.7.1.2 Určení kritérií a výběr implementačního prostředí

Cílem tohoto kroku je globálně zmapovat základní známá omezení a kritické faktory, dané současným stavem, momentálními okolnostmi, použitou technologií, či technologickými záměry apod. Jde o omezení a kritické faktory, které budou mít podstatný vliv na možnosti realizace informačního systému.

V případě, kdy je v této etapě již detailně známo implementační prostředí, je základní sada kritérií zpravidla dána tímto prostředím. V případě, kdy implementační prostředí není určeno, je třeba, na základě globálních charakteristik a faktorů, vyplynuvších z výsledku předchozího kroku, určit základní charakteristiky, které implementační prostředí bude mít (například důraz na internetové technologie díky teritoriálnímu rozmístění, či heterogenitě uživatelského rozhraní, nebo uzavřený systém s důrazem na bezpečnost dat apod.).

Výstupy tohoto kroku budou důležitým podkladem pro Detailní specifikaci kritérií designu ve fázi detailního designu.

A.7.1.3 Vytvoření hrubého technologického modelu

Cílem tohoto kroku je vytvořit základní globální představu o technologickém uspořádání systému.

Podkladem k takové představě je:

- Základní rozdělení systému na subsystémy,
- Zvolené charakteristiky a kriteria implementačního prostředí.

V tomto kroku jsou určeny základní technologické celky systému – komponenty a je globálně zmapováno rozmístění komponent v jednotlivých fyzických místech systému.

Technologický model je globální – tedy jednotlivé komponenty a jejich fyzické umístění mapují základní funkční celky systému, dané globálním funkčním modelem.

U standardně řešitelných komponent je zpravidla jejich další design určen příslušným prostředím. Individuálně řešené komponenty pak budou muset být předmětem detailního designu tak, jak je popisován níže.

A.7.1.4 Určení způsobu vývoje jednotlivých komponent

Cílem tohoto kroku je, na základě výsledku předchozího kroku - globální představy o technologickém uspořádání systému – rozhodnout o způsobu vývoje jednotlivých technologických komponent.

V zásadě je hlavním, universálním kritériem takového rozhodnutí *míra standardnosti* příslušné komponenty. Dobře standardizovatelné oblasti by měly být pokryty pokud možno standardním řešením (např. modulem ERP, či jiným parametrizovatelným standardním technologickým celkem). Naopak, standardizovatelné oblasti budou muset být podrobeny Detailní analýze a designu a implementovány zpravidla specifickým způsobem.

Na výstup tohoto kroku přímo navazuje následující krok Výběr subsystémů k detailní analýze.

A.7.1.5 Specifikace vlivů navrhovaného systému na organizaci

Cílem tohoto kroku je posoudit vliv navrženého systému na organizaci.

Tento krok je důležitý zejména pro následný krok výběru subsystémů k detailní analýze, kde je vliv systému na organizaci důležitým hlediskem rozhodování o dalším postupu analýzy (zejména detailní analýzy).

Kromě následného kroku výběru subsystémů k detailní analýze je vliv systému na organizaci důležitým faktorem dalšího postupu.

A.7.1.6 Výběr subsystémů k detailní analýze

Cílem tohoto kroku je vybrat subsystémy, které bude nutno podrobit detailní analýze a současně stanovit základní rysy dalšího postupu vývoje systému, zohledňující jak samotnou povahu subsystému (určující nutný/možný způsob jeho realizace), tak i specifické vnější podmínky jeho realizace (například situační faktory, priority organizace apod.).

Výběr subsystémů k detailní analýze je dán ze dvou stran:

- Určeným způsobem realizace komponenty (použití standardního řešení versus specifický vývoj „na míru“),
- Faktory, plynoucími ze zjištěného vlivu na organizaci.

Na základě posouzení výše uvedených dvou dimenzí výběru subsystémů k detailní analýze pak může být rozhodnuto o detailech dalšího postupu. Výsledkem je posléze specifikace jednotlivých přírůstků jako základního východiska stanovení plánu navazujících vývojových projektů.

A.7.1.7 Vymezení specifických a omezujících faktorů vyvíjeného systému (subsystému)

Cílem tohoto kroku je vymežit specifické a omezující faktory pro další vývoj jednotlivých subsystémů.

Specifické a omezující faktory jsou vždy poplatné určenému způsobu vývoje daného funkčního celku – subsystému. Typicky u standardizovatelných subsystémů, řešených například moduly ERP, nebo jiným standardním řešením, plynou specifika a omezení ze zvoleného realizačního prostředí (aplikace). Naopak u subsystémů specifických, které bude nutno podrobit detailní analýze a posléze realizovat specifickým způsobem, platí spíše obecnější faktory a omezení, definované namnoze metodikou – podrobnosti jejich realizace a zvoleného prostředí zpravidla v této etapě nejsou známy, naopak je třeba předpokládat, že jejich potřeba bude upřesněna až teprve na základě detailní analýzy těchto subsystémů.

Vymezení specifických a omezujících faktorů vyvíjeného systému je, vedle zjištěného vlivu na IS na organizaci, dalším důležitým hlediskem, rozhodujícím o detailech dalšího postupu.

A.7.2 Kroky detailní analýzy

Cílem detailní analýzy systému je rozpracování vybraných funkčních oblast (subsystémů) do „absolutního detailu“, tedy do detailu, vhodného k následnému designu a implementaci. Absolutnost detailu je zde metodicky dána – je definována použitými principy a technikami, které jsou podrobněji popisovány v díle B této publikace.

Narozdíl od globálního návrhu zde již není hlavním cílem úplnost celého systému (detailní analýza se týká pouze vybraných subsystémů), ale *úplná podrobnost na úkor celistvosti*. K tomu musí být systém, jako celek, kvalifikovaně rozdělen do takových částí, které povedou k identifikaci těch funkčních oblastí (subsystémů), jejichž *detailní analýza je objektivně nutná*. To je také jeden z hlavních úkolů globálního návrhu, k jehož splnění disponuje již

zmiňovanými nástroji a technikami, které jsou ještě podrobněji popisovány v dílu B této publikace.

Detailní analýza systému zahrnuje následující analytické modely:

- Detailní konceptuální model business objektů,
- Detailní model business procesů,
- Detailní funkční model systému,
- Stavový model životních cyklů klíčových objektů.

Na základě detailní analýzy procesů a objektů jsou nejprve vytvořeny oba detailní modely (procesů a objektů), které jsou posléze vnitřně provázány prostřednictvím stavového diagramu, popisujícího životní cykly klíčových objektů (resp. tříd objektů). Z těchto modelů je odvozena detailní dekompozice funkčního modelu, a to až do úrovně, vhodné k následné analýze událostí a reakcí systému na ně, na jejímž základě je detailizován funkční model. Posledním krokem je balancování všech detailních modelů podle daných pravidel.

A.7.2.1 Detailní analýza objektů

Cílem tohoto kroku je detailně specifikovat soustavu reálných (tzv. business) objektů dané oblasti, a to ve formě diagramu tříd a jejich vztahů. Základním výchozím modelem je zde globální model objektů z předchozí etapy. Detailní analýza objektů se pak odehrává toliko na vybraných objektech globálního modelu – těch, které podstatným způsobem figuruji ve vybrané funkční oblasti (přírůstu).

Narozdíl od globálního modelu objektů, zde je u jednotlivých tříd třeba popisovat nejenom atributy, ale i akce k nim vázané – tzv. „metody“ objektů. Metody musí vyjadřovat konceptuální – tedy esenciální reálné akce, které objekt prování, nebo jsou na něm prováděny. U jednotlivých klíčových tříd objektů jsou popisovány jejich „životní cykly“ ve formě stavového diagramu. Klíčovými třídami jsou zde myšleny takové třídy objektů, které hrají podstatnou roli ve vztahu k business procesům. Z hlediska procesů jsou to zejména ty třídy, které představují jednak cílové produkty procesů, jednak jejich klíčové aktéry. Smyslem popisu životního cyklu objektu je pak především specifikovat základní objektivní omezení (business constraints, business rules), a to formou procesního popisu (životní cyklus je popisován jako proces – následnost jednotlivých stavů objektu, dosahovaných vlivem jeho akcí - metod).

Podrobnější popis analýzy objektů je v dílu B této publikace.

A.7.2.2 Detailní analýza procesů

Cílem tohoto kroku je detailně specifikovat soustavu reálných (tzv. business) procesů dané oblasti, a to na úrovni „absolutního detailu“. Tímto „absolutním detailem“ – tedy z hlediska obsahu elementární úrovně popisu procesu – se zde rozumí úroveň *elementárních událostí*. Business procesy, popisované detailně v tomto kroku, představují zcela jiné procesy, než jsou životní cykly objektů – zatímco proces života objektu slouží v analytických modelech jako specifikace základních omezení (business rules), neměnných vlastností - zákonitostí reálného systému, business proces vyjadřuje vůli, má cíl, specifikuje soustavu akcí, vedoucích k realizaci tohoto cíle. Proto také *pouze business procesy vnímá metodika jako popis chování reality*, zatímco popisy životů jednotlivých objektů vnímá jako specifikaci podmínek, jež musí každý business proces respektovat.

Podrobnější popis analýzy procesů je v dílu B této publikace.

A.7.2.3 Dekompozice funkčního modelu

Cílem tohoto kroku je dekomponovat (rozložit) funkcionalitu zvolené oblasti až do úrovně, vhodné k následné analýze událostí a reakcí systému na ně (viz následující krok), a to s cílem detailizace funkčního modelu až na úroveň „absolutního detailu“, k níž dojde v kroku následujícím.

Tento krok je nutný pouze v případě, kdy detailně analyzovaná funkční oblast je natolik rozsáhlá, že by nebylo možné použít techniku analýzy událostí (zmiňovanou v následném kroku) na celou tuto oblast – není tedy zvládnutelné detailně popsat celou funkční oblast najednou – jednou analýzou událostí.

A.7.2.4 Detailní analýza událostí a reakcí systému

Cílem tohoto kroku je detailizace funkčního modelu až na úroveň „absolutního detailu“. Detailizace probíhá ve struktuře, určené předchozí dekompozicí funkcí. „Absolutní detail“ – tedy z hlediska obsahu elementární úroveň popisu – je zde určen úrovní *elementárních událostí*. Toto kritérium se zde stejné, jako u analýzy procesů.

Základní technikou, používanou k vytvoření detailního funkčního modelu na úrovni jednotlivých esenciálních událostí, je *technika analýzy událostí* a reakcí na ně, jejíž základ je popisován originálně v Yourdon, 1988, podrobněji pak v Řepa, 1999. Tato technika představuje specifické vnímání chování (dynamiky) reality jako souhrnu *esenciálních událostí* a jejich vzájemných souvislostí. Funkčnost informačního systému je potom přímým odrazem souvislostí (kombinace) jednotlivých událostí – funkce představuje reakci systému na kombinaci událostí, potřeba ukládat data v systému je pak přímým odrazem nutnosti kombinovat různé události v systému.

Podstata detailní analýzy událostí, jako základu specifikace funkčnosti, byla vysvětlena již v kapitole A.5 a tato technika je také podrobněji popisována v kapitole B.1.5.

A.7.2.5 Balancování funkčního, procesního a objektového modelu

Cílem tohoto kroku je finální vybalancování všech analytických modelů s cílem dosažení jejich úplné konsistence. K tomu metodika poskytuje pravidla konsistence a nástroje a techniky jejího dosažení, již několikrát zmiňované v předchozím textu. Podrobněji jsou tato pravidla, postupy a nástroje, popisovány v dílu B této publikace.

A.7.3 Kroky designu

Cílem designu systému je navrhnout realizaci obsahu IS, jenž je dán analytickými modely, v daném technologickém prostředí. Technologickým prostředím se zde rozumí použitá technologie, jež s sebou vždy nese určité specifické principy a předurčené způsoby realizace (například prostředí relačních databází striktně předurčuje způsob realizace datových celků a vztahů mezi nimi, jenž, z hlediska obecných kritérií pružnosti a nezávislosti systému, neumožňuje jeho plnou optimalizaci v těchto vlastnostech. Zato nabízejí relační databáze technologické řešení zvládnutí z toho plynoucích potenciálních nebezpečí nekonsistence systému například v podobě mechanismu integritních omezení. Tím je také poměrně přesně určen způsob realizace systému s takovou technologií, jenž je radikálně odlišný od možných způsobů realizace v jiných prostředích – například v prostředí internetových technologií s použitím jazyka Java bez databázového systému).

Z hlediska systému jako celku je třeba vždy uvažovat souběžně různá technologická prostředí. Různé části obsahu systému tak musí být realizovány různými způsoby, které jsou v různé míře předurčeny zvoleným způsobem realizace dané části (funkční oblasti - subsystému), jak již bylo v předchozím textu několikrát zmiňováno. To je také důvodem k pečlivému zvážení způsobů realizace jednotlivých subsystémů již ve fázi globálního návrhu (viz výše). Každý informační systém je totiž ze své podstaty heterogenní – je složen z různých modulů, realizovaných různými způsoby v různých technologických prostředích. Proto je zejména důležité sledovat společné hodnoty systému, na jejichž základě je teprve možné rozvoj systému smysluplně řídit. Touto společnou hodnotou je především obsah systému – tedy jeho obsahový model, nebo ještě přesněji procesně-objektový model business systému⁸.

A.7.3.1 Detailní specifikace kritérií designu

Cílem tohoto kroku je určit detailní kritéria designu systému. Metodika obsahuje obecnou představu o základních kritériích designu. V konkrétním případě každého systému je pak třeba uvažovat specifickou soustavu kritérií (některá obecná kritéria v daném případě neplatí, některá platí modifikovaně, mají různé váhy, resp. se různě podmiňují apod.). Detailní design tedy vždy musí začít podrobnou specifikací kritérií.

Specifikovaná kritéria potom slouží v dalším postupu designu jako základní a průběžné měřítko kvality a konsistence jednotlivých rozhodnutí.

Podrobnější popis postupu designu je v kapitole B.2 této publikace.

A.7.3.2 Detailní specifikace komponent

Cílem tohoto kroku je stanovit vhodnou architekturu komponent (technologických částí) systému, splňující obecné principy designu, a to v rámci specifikovaných kritérií. K tomu metodika poskytuje obecnou představu jednotlivých typových architektur (architektonických vzorů). Komponentou systému se obecně rozumí souhrn programových částí, tvořících celek s definovanými odpovědnostmi.

Tento krok je třeba provádět souběžně s následujícím krokem specifikace procesní architektury a v jejich vzájemné interakci. Jednotlivá rozhodnutí o komponentách a „procesech“ se totiž vzájemně podmiňují a ovlivňují.

Podrobnější popis postupu designu je v kapitole B.2 této publikace.

A.7.3.3 Detailní specifikace procesní architektury

Cílem tohoto kroku je stanovit vhodnou architekturu procesů systému. Procesní architekturou se obecně rozumí struktura nezávislých procesů popisující běh systému. Obecným smyslem je pak navrhnout takovou strukturu systémových procesů, která nemá úzká místa a maximálně využívá koordinaci sdílení zdrojů s aktivními objekty.

Tento krok je třeba provádět souběžně s předchozím krokem specifikace komponent a v jejich vzájemné interakci. Jednotlivá rozhodnutí o komponentách a „procesech“ se totiž vzájemně podmiňují a ovlivňují.

Podrobnější popis postupu designu je v kapitole B.2 této publikace.

⁸ Tento fakt je též dobře vidět v nejnovějším trendu vývoje tzv. „integračních nástrojů“ (integračních platform, či middleware), které staví integraci na vědomí základního obsahového modelu business procesů organizace s tím, že právě business procesy určují jednotlivým složkám, z podstaty heterogenního informačního systému, potřebný kontext. Jedním z představitelů takových nástrojů je i Sybase Integration Orchestrator, zmiňovaný též v dílech B a C této publikace.

A.7.3.4 Vytvoření logického schéma dat

Cílem tohoto kroku je vytvořit logické schéma dat pro zvolené prostředí realizace datové základny systému. Tento krok je zejména důležitý v případě rozhodnutí o realizaci datové základny v prostředí databáze (například relační databáze), jelikož databázové technologie silně předurčují způsob realizace

Logickým schematem dat se rozumí rozvržení obsahu datové základny do jednotlivých logických celků, poplatných dané technologii (například do tabulek relační databáze, stromové struktury uzlů databáze hierarchické, nebo souborů s přímým, či sekvenčním přístupem k datům apod.). Jedná se stále o rozvržení logické, protože se ještě neuvažuje konkrétní realizace (fyzická realizace - implementace), pouze základní logika daného prostředí. Různá realizační prostředí datové základny (zejména databázové systémy) zpravidla poměrně přesně definují „logiku“ své technologie, a to na obecné úrovni – na úrovni standardů (viz např. relační standardy, standardy jazyka SQL apod.).

A.7.3.5 Kompletace technologického modelu systému (subsystému)

Cílem tohoto kroku je shrnutí technologického modelu systému ve vzájemných souvislostech, zejména ve vazbě jednotlivých komponent na jednotlivé části datové základny a jednotlivé fyzické uzly systému, a to na úrovni, vhodné k fyzické realizaci – implementaci⁹.

A.7.3.6 Specifikace výstupů systému (subsystému)

Cílem tohoto doplňkového kroku je, na základě kompletního technologického modelu systému, detailně specifikovat jednotlivé výstupy systému, a to včetně jejich formy a specifik, daných příslušným fyzickým umístěním komponenty.

A.7.3.7 Specifikace vstupů a operačních procedur

Cílem tohoto doplňkového kroku je, na základě kompletního technologického modelu systému, detailně specifikovat jednotlivé vstupy systému a operační procedury (tedy procedury, spojené s provozem systému), dané jednak obecněji příslušnou komponentou, jednak konkrétně příslušným fyzickým umístěním komponenty.

A.7.4 Závěrečné kroky

Následující kroky uzavírají a finalizují závěry jednotlivých dílčích kroků předchozího postupu.

A.7.4.1 Závěrečná revize návrhu systému (subsystému)

Cílem tohoto kroku je celkově posoudit návrh systému především s cílem souhrnného zhodnocení vzájemného provázání jednotlivých úhlů pohledu (analýza versus design, procesy versus objekty, versus funkčnost, apod.). V dosavadním postupu analýzy a návrhu byly veškeré pohledy na systém dílčí, v tomto kroku je příležitost znovu (jako na počátku – u globálního návrhu) zaujmout globální pohled na systém a finálně posoudit jeho celistvost.

⁹ úroveň, která je vhodná k implementaci, se přirozeně liší podle zvoleného způsobu realizace (bude například zcela jiná u komponenty vyvíjené na míru – tam bude muset být velmi detailní - a u komponenty, realizované standardním modulem ERP, který může veškerou realizaci redukovat toliko na nastavení základních parametrů modulu).

A.7.4.2 Příprava plánu testování systému (subsystému)

Cílem tohoto kroku je připravit plán testování systému jako celku.

Problematika testování není touto metodikou typicky pojímána jako jednorázová záležitost, ale jako *integrální součást postupu vývoje systému*, integrovaná do jednotlivých principů, metod a technik. Tak například analytické modely umožňují „testovat“ obsahovou správnost úvah jednak v mezích svého úhlu pohledu (pomocí principů a pravidel návrhu daného modelu – například pravidly popisu business procesu, pravidly modelování objektů, nebo pravidly návrhu funkční struktury zkoumáním událostí), jednak i z hlediska jejich překryvů a vzájemných souvislostí (v podobě tzv. konsistenčních pravidel).

Podobně i v oblasti designu systému například zvolená kritéria designu specifikují kvalitativní požadavky na jednotlivá učiněná rozhodnutí, která jsou pak jimi permanentně „testována“ z hlediska obsahové korektnosti a vztahu k celku.

Přesto, že základní testování je výše popsaným způsobem průběžně ošetřováno celým postupem, je v postupu počítáno i se závěrečným souhrnným testováním systému (resp. jeho přírůstkem), a to s cílem jednak tradičního ověření a prokázání funkčnosti jako celku, jednak upřesnění případného dalšího kola požadavků na změny na bázi „prototypového“ přístupu.

A.7.4.3 Upřesnění požadavků na organizaci

Cílem tohoto kroku je navázat na krok globálního návrhu „Specifikace vlivů navrhovaného systému na organizaci“ a na krok designu a rozvést jeho závěry ve světle celkového návrhu. Výstupem tohoto kroku je zejména návrh organizačních, personálních a kvalifikačních změn, celková specifikace operačních procedur a další.

A.7.4.4 Příprava plánu implementace a zavádění systému (subsystému)

Cílem tohoto kroku je připravit plán implementace a zavádění systému, jako následný krok metodického postupu (viz následující etapy Implementace a Zavedení systému).

Tento krok již není krokem obsahovým, ale představuje spíše *řídící dimenzi postupu vývoje IS*. Při plánování vývoje IS (resp. jeho části – tedy přírůstkem) jako projektu, byla naplánována základní následnost jednotlivých etap a hrubý obsah jejich činností výstupů. V tomto bodě postupu – po finalizaci všech obsahových kroků analýzy a návrhu systému – je možné existující hrubou představu postupu zpodrobnit na základě zjištěných skutečností (analýzou) a učiněných rozhodnutí (designem).

A.7.4.5 Závěrečná revize návrhu a kompletace dokumentace

Tento krok uzavírá celou analýzu a návrh systému finálním globálním posouzením výstupů ve vzájemných souvislostech a kompletací jeho dokumentace.

Podobně jako předchozí krok, i tento již není čistě obsahovým, ale představuje zčásti i *řídící dimenzi postupu vývoje IS* – výsledná dokumentace má povahu řídicího dokumentu, který slouží nejenom k dokumentaci, ale i jako nástroj řízení postupu – metodika určuje role a odpovědnosti k tomuto dokumentu vázané. Prostřednictvím schválení této dokumentace pak dochází k – z hlediska řízení projektu nezbytně nutné – aktivizaci příslušných odpovědností v příslušném kontrolním bodu – *milníku* – projektu.

Díl B Použití jednotlivých modelů a jejich provázání

Tento díl se zabývá konkrétní realizací podpory jednotlivých modelů, uvedených v předchozím dílu, prostřednictvím jednotlivých diagramů, a to se specifickým zaměřením na podporu nástrojem Power Designer.

Power Designer podporuje především jazyk UML, jenž je standardem objektově orientovaného modelování a tento jazyk doplňuje specifickým jazykem (diagramem) pro modelování podnikových procesů, jakož i dalšími doplňkovými možnostmi, z nichž nejvýznamnější (alespoň z hlediska analýzy a konstrukce systému) podpora analýzy požadavků (viz podrobnější popis níže).

UML (Unified Modeling Language) je grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. UML nabízí standardní způsob zápisu jak návrhů systému včetně konceptuálních prvků jako jsou business procesy a systémové funkce, tak konkrétních prvků jako jsou příkazy programovacího jazyka, databázová schémata a znovupoužitelné programové komponenty.

Podle oficiálního členění zahrnuje UML definici 8-mi typů diagramů, tj. 8-mi různých pohledů na model vyvíjeného systému:

- **diagramy tříd a objektů** (class diagrams, object diagrams) popisují statickou strukturu systému, znázorňují datový model systému od konceptuální úrovně až po implementaci,
- **modely jednání** (diagramy případů užití - use case diagrams) dokumentují možné případy použití systému - události, na které musí systém reagovat,
- **diagramy posloupností** (sequence diagrams) popisují scénář průběhu určité činnosti v systému,
- **diagramy spolupráce** (collaboration diagrams) zachycují komunikaci spolupracujících objektů,
- **stavové diagramy** (statechart diagrams) popisují dynamické chování objektu nebo systému, možné stavy a přechody mezi nimi,
- **diagramy činností** (activity diagrams) popisují průběh aktivit procesu či činnosti,
- **diagramy komponent** (component diagrams) popisují rozdělení výsledného systému na funkční celky (komponenty) a definují náplň jednotlivých komponent,
- **diagramy rozmístění** (deployment diagrams) popisují umístění funkčních celků (komponent) na fyzické uzly informačního systému.

Diagramy tříd, diagramy spolupráce, diagramy komponent a diagram nasazení vyjadřují **statickou strukturu systému**.

Model jednání, diagramy činností, diagramy posloupností a diagramy spolupráce. popisují **funkčnost systému**.

Stavové diagramy, diagramy posloupností, diagramy spolupráce a diagramy aktivit popisují **dynamiku systému**.

UML je koncipován tak, že podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů, nepředepisuje však způsob, jak se má používat, ani neobsahuje metodiku(y), jak analyzovat, specifikovat či navrhovat programové systémy. Použití diagramů v jednotlivých fázích vývoje je tak dáno konkrétní metodikou.

V dalším textu bude věnována specifická pozornost jednotlivým diagramům tak, jak jsou podporovány nástrojem Power Designer.

B.1 Analytické modely

V této kapitole jsou popisovány modely a nástroje pro jejich tvorbu, které mají význam z hlediska fáze analýzy. Popis je veden podle *jednotlivých nástrojů* – diagramů, používaných k tvorbě toho kterého modelu, nicméně obecnější náležitosti modelu – tedy bez ohledu na nástroj - diagram, k jeho popisu použitý - jsou diskutovány v textu uvnitř kapitol spolu s vazbou modelu na obecnější principy metodiky, vazbami mezi modely a pravidly jejich tvorby, tedy, jednoduše řečeno, ostatními *metodickými náležitostmi modelů*..

B.1.1 Diagram tříd

Zobrazuje statickou strukturu systému prostřednictvím tříd a vztahů mezi nimi. Diagram tříd tvoří základ všech prostředků objektové analýzy a designu. Jejich cílem je definovat formálnějším zápisem termíny a vztahy v studované oblasti. Jde o statickou stránku systému. Lze ho rozpracovat až do podoby zdrojového kódu.

Zvláště při tvorbě těchto diagramů se doporučuje dodržovat nepsané pravidlo 7 + 2, které říká, že v jednom diagramu je vhodné zobrazit 7, maximálně až 9 tříd. Při respektování tohoto doporučení se většinou výraznou měrou zpřehlední výsledné modely, nicméně na druhou stranu vyvstává problém vhodného rozdělení systému na dílčí oblasti.

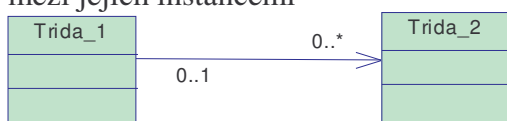
Třída	
-	Atribut_1 : int
-	Atribut_2 : int
-	Atribut_3 : int
+	Operace_1 () : int
+	Operace_2 () : int
+	Operace_3 () : int

Základní prvky Třídy:

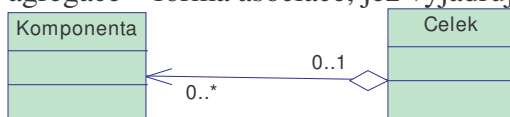
1. atribut – vlastnosti třídy
2. operace – akce třídy

Základní vazby:

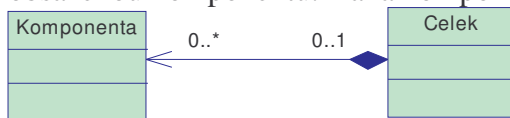
1. asociace – obecný sémantický vztah mezi prvky modelu, který specifikuje spojení mezi jejich instancemi



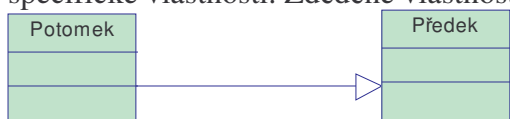
2. agregace – forma asociace, jež vyjadřuje vztah celek – část



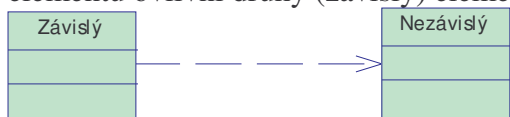
3. kompozice – silnější vazba než agregace - zrušením celku automaticky zrušíme i obsaženou komponentu. Daná komponenta může být součástí právě jednoho celku



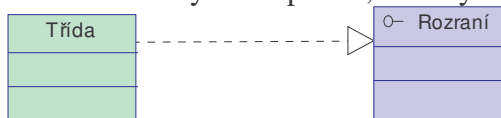
4. dědičnost – hierarchický vztah tříd, v němž třída - potomek dědí atributy a operace třídy – předka. Kromě zděděných vlastností je možné, aby měl potomek ještě své specifické vlastnosti. Zděděné vlastnosti mohou být v potomkovi modifikovány.



5. závislost – vztah mezi dvěma elementy modelu, v němž změna jednoho (nezávislého) elementu ovlivní druhý (závislý) element



6. realizace – souhrn všech veřejně přístupných metod dané třídy. Umožňuje vícenásobné využití operací, aniž bychom museli zavádět dědičnost mezi třídami



7. multiplicita




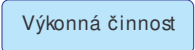

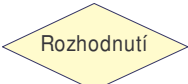
Zápis	Význam
0..1	0 nebo 1
0..*	0 až více
1..1	Právě jedna
1..*	1 až více
*	0 až více







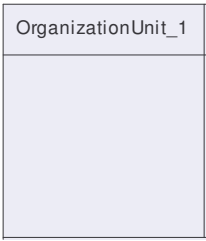
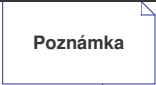

B.1.2 Diagram procesů

Diagram procesů modeluje business procesy v podniku. Jednotlivé procesy mohou být děleny do podprocesů, jednotlivé nezávislé procesy jsou vzájemně synchronizovány prostřednictvím spojení výstupních stavů a počátečních událostí (výstupní stav jednoho procesu může být vstupní událostí procesu jiného).

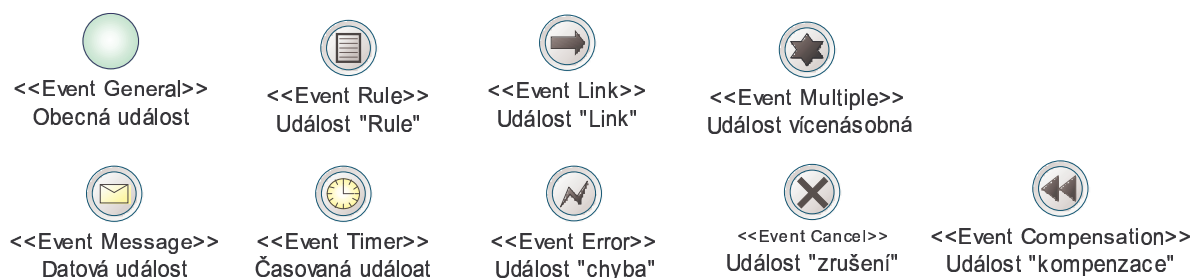
Základní požadavky metodiky na Diagram procesů vycházejí z představy o základních zákonitostech detailního modelování postupu procesu, jež je formálně zachycena v metamodelu podnikového procesu, publikovaného v [Řepa 2005] a na <http://opensoul.panrepa.org>. Metamodel popisuje základní potřebné prvky modelu procesu a jejich základní vazby a klasifikace. Popis a vysvětlení významu základních konstruktů modelu procesu, včetně jejich, v Power Designeru dostupné notace BPNM, obsahuje tabulka níže. Z použité notace jsou uvedeny pouze základní symboly, další rozšiřující symboly BPNM jsou diskutovány níže.

Základní konstrukce diagramu:

Konstrukt	Použitý symbol	Popis
Událost	 <<Event General>> Obecná událost	Vnější podnět činnosti. Informace o skutečnosti nastalé mimo proces (nezávisle na něm). <i>V nástroji Power Designer lze vyjádřit použitím symbolu „start“ doplněného názvem události. Start lze použít vícenásobně – pro každou událost. Pro popis formy vstupu, jímž je událost signalizována (pokud je s událostí spojen nějaký hmotný, či informační vstup, např. u událostí časovaných (periodických) lze použít bohatý repertoár symbolů BPMN, diskutovaný níže a vhodný i pro rozlišení událostí časovaných od běžných (business).</i>
Stav procesu	 <<Parallel(AND)>> Vnitřní stav procesu  <<End Terminate>> Koncový stav obecný	Vnitřní podnět činnosti. Výsledek činnosti logicky předcházející. Místo mezi činnostmi procesu. <i>V notaci Power Designeru, lze vyjádřit použitím „synchronizace“.</i> Koncový stav procesu. <i>V nástroji Power Designer lze použít symbol „End“. Pro vyjádření formy výstupu, s nímž je koncový stav případně spojen, obsahuje jazyk BPMN bohatou paletu symbolů, podobně jako u událostí (viz Událost).</i>
Činnost	 Výkonná činnost  Komplexní činnost	Základní element procesu – zpracování vstupů na výstupy. Činnost je z principu dekomponovatelná, čili může být nahlížena jako samostatný proces (komplexní činnost). <i>Dekompozice (nastavení volby „Change to Composite“) je graficky znázorněna smyčkou v boxu činnosti.</i>
Rozhodovací činnost	 Rozhodnutí	Elementární (dále nedekomponovatelná) činnost, jejímž výstupem je nic více, než rozhodnutí o dalším postupu procesu.

Konstrukt	Použitý symbol	Popis
	 <<Complex>> Rozhodnutí (BPMN)	
Logická spojka (primitivní rozhodnutí)	 <<Parallel(AND)>> AND  <<Inclusive(OR)>> OR  <<Data-XOR>> XOR - vylučnost dat  <<Event-XOR>> XOR - vylučnost událostí	<p>Primitivní rozhodovací činnost, která nepotřebuje žádné dodatečné (informační) vstupy.</p> <p><i>V nástroji Power Designer jsou z nabídky BPMN použitelné standardní stereotypy rozhodovací činnosti AND, OR a (nikoliv nezbytné) dva podtypy XOR (datový a událostní).</i></p>
Množina dat	 Vstup / výstup	<p>Množina údajů, či surovin, které slouží jako zdroj pro provedení činnosti procesu nebo je jejich výstupem (obecný zdroj). Příklady: výrobní plán, strategický plán investic, dodací list apod. Lze použít i jako množina materiálu v kombinaci s informací. Příklad: dodávka společně s dodacím listem.</p>
Smíšená množina		
Množina materiálu		
Aktér		Abstraktní účastník (osoba – její role, útvar, systém, orgán, objektivní entita) procesu.
Organizační jednotka		<p>Organizační část organizace, v níž proces probíhá.</p> <p><i>V notaci BPMN jsou organizační jednotky použitelné pouze ve formě tzv. „swim lanes“ (plaveckých drah), uzavírajících všechny činnosti náležející dané jednotce (roli). To, žel, poněkud redukuje možnosti popisu procesu, nezávislého na organizační struktuře.</i></p>
Problém	 	<p>Problém, spojený s procesem v jistém jeho místě (stavu).</p> <p><i>V nástroji PowerDesigner lze vyjádřit poznámkou („note“), nebo rovnou do popisu procesu.</i></p>

Notace BPMN nabízí pro události a koncové stavy bohatou paletu různorodých symbolů k rozlišení různých možných forem události a stavu.



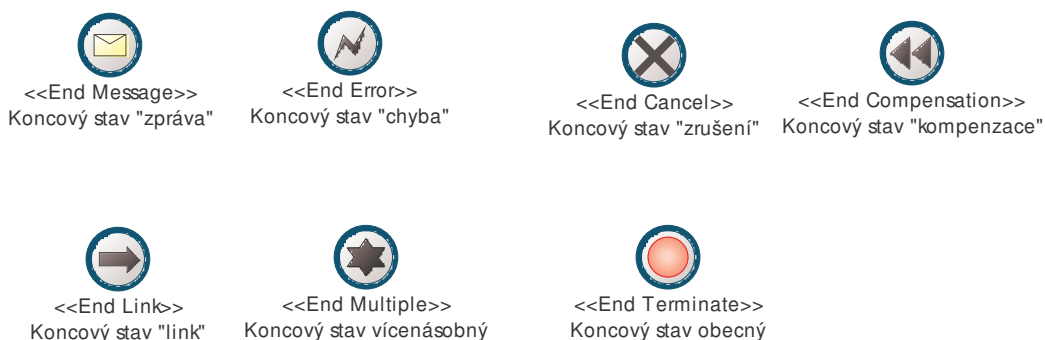
Obrázek B 1 Druhy událostí v BPMN

Jak ukazuje Obrázek B 1, kromě základního rozlišení událostí na časované a business (zde tzv. „datové“), jsou rozlišovány jako specifické události též chyba, zrušení, dané pravidlo, propojení na jiný proces, či tzv. „kompenzace“. Tato bohatá paleta různých druhů události je do značné míry důsledkem poněkud techničtějšího (přesněji IT) pojetí business procesu v zaměření organizace BPMI a z hlediska analýzy tedy nepřiliš podstatná, ne-li přímo nevhodná (z hlediska potřeby odlišit obsah od jeho technické realizace).



Obrázek B 2 Druhy startovacích událostí procesu v BPMN

Kromě obecného rozlišení odlišuje BPMN od obecných událostí také události „startovací“ (Obrázek B 2). Vzhledem k definici události v této metodice, je takové rozlišení irelevantní (každá událost je z hlediska informačního systému „startovací“), nicméně může být významné pro „business“ analýzu, jejímž cílem není jen vývoj informačního systému. Například v globálním rozlišení základních procesů a jejich globálním popisu je důležité najít klíčovou, prvotní událost, která charakterizuje základní smysl procesu – tzv. „startovací událost“.



Obrázek B 3 Druhy koncových stavů procesu v BPMN

Jako u událostí, rozlišuje BPMN obdobné druhy koncových stavů procesu (Obrázek B 3). Při modelování procesů je třeba se řídit základními metodickými pravidly, které vyplývají z podstaty procesního modelu, popisované v části A. Tato pravidla jsou zpracována také do Techniky modelování procesů, popisované například v [Řepa, 2005] a jsou rovněž formálně vyjádřena v metamodelu.

Metamodel vymezuje jednak základní pojmy z oblasti modelování procesu, jednak jejich základní nutné vztahy. Vyjadřuje jakési minimální požadavky na model procesu – co vše by měl zahrnovat a v jakých souvislostech. Navíc, právě popisem oněch souvislostí, vyjadřuje metamodel i základní pravidla, která objektivně a obecně pro všechny procesy platí a která tedy musí být každým modelem procesu respektována. Metamodel, jakožto konceptuální model, je proveden v jazyku UML (UML, 2003), jakožto všeobecně přijatém standardu pro konceptuální modelování (což platí nepochybně zejména v oblasti metamodelování). Pro důkladné porozumění metamodelu je tak, ne-li nutné, tedy přinejmenším velmi vhodné rozumět jazyku UML, nejlépe jeho použití k metamodelování (viz např. MOF, 2003).

Každý prvek modelu podnikového procesu je **elementem modelu**. Existují dva základní druhy elementu modelu podnikového procesu:

- pojem
- externí aspekt

Termín **pojem** označuje všechny vnitřní (ne externí) elementy modelu podnikového procesu. Tyto se dělí do dvou skupin:

- hlavní pojem
- vstupně-výstupní množina

Hlavní pojem označuje všechny aspekty vnitřního (ne vstupně-výstupního) chování procesu.

Hlavní pojmy jsou tři:

- stimul (podnět)
- stav
- činnost

Stimulem se rozumí přípustný podnět činnosti. Tím může být buď **událost**, nebo **řídící činnost**¹⁰.

Stavem procesu se rozumí každá objektivně nutná přestávka mezi dvěma činnostmi procesu. Objektivní je proto, že je nutná z důvodů, ležících mimo proces – je to čekání na vnější (zevnitř procesu neovlivnitelnou) událost.

Stavy a stimuly spolu podstatným způsobem souvisí. Jak je z meta-modelu vidět, každý stimul přichází do procesu v určitém jeho stavu (přesněji musí mít alespoň jeden vstupní stav). Jedinou výjimkou je úplně první stimul – **počáteční událost**, jež se neváže k žádnému stavu (jak je vidět ze specifické kardinality 0:0, jež zde „přebíjí“ zděděnou obecnou asociaci mezi stimulem a (nekoncovým) stavem). Podobně i mezi stavy jsou výjimkou stavy koncové, tedy takové, na něž neváže, z hlediska procesu, už žádný další stimul – tato výjimka je v meta-modelu vyjádřena pomocí specifického pojmu **nekoncový stav**¹¹.

¹⁰ Řídící činnost, jakožto stimul, dědí schopnost stimulovat jiné činnosti. Z této skutečnosti, společně s kardinalitou 1 (tedy monopolitou) asociace, vyjadřující stimulaci, vyplývá, že výkonná činnost může být stimulována buď jen událostí, anebo řídící činností, nikoliv oběma najednou.

¹¹ Pojmy počáteční událost a koncový stav jsou relativní konkrétnímu specifikovanému modelu. Použití modelu jako části jiného modelu změní všechny jeho počáteční události na běžné a všechny jeho koncové stavy na interní (z hlediska nadřazeného procesu).

Činnost může být buď komplexní, nebo elementární.

Komplexní činnost je pojata jako ne-elementární část podnikového procesu, tedy jako uspořádaný souhrn aspektů vnitřního chování procesu – stimulů, stavů a činností. Zvláštním případem komplexní činnosti je potom **podnikový proces**. Je to taková komplexní činnost, která má samostatný cíl, vlastníka, omezení a případné další atributy podnikového procesu. Tato konstrukce vyjadřuje relativnost pojmů proces a činnost (jakákoliv komplexní činnost může být popsána jako samostatný proces, má-li to smysl – tedy má-li samostatný cíl, vlastníka atd.) a je projevem principu rozkladu top-down, který je procesnímu modelu vlastní.

Elementární činnost pak může být buď řídicí, nebo výkonná. Rozdíl mezi těmito dvěma druhy činností je v jejich určení – zatímco **řídicí činnost** vyjadřuje rozhodování v procesu (a v tomto smyslu může být i stimulem jiné činnosti, jak je zmiňováno výše), nikoliv výkon, **výkonná činnost** je určena k výkonu, tedy ke zpracování vstupu a „výrobě“ výstupu, jak plyne z asociací této třídy se vstupně/výstupní množinou. Z modelu je rovněž vidět, že i řídicí činnost může mít něco společného se vstupně/výstupní množinou, ovšem jen ve smyslu vstupu (vstupní informace pro rozhodnutí). Zvláštním případem je potom takové rozhodnutí, které nemá žádný vstup. Jedná se o rozhodnutí předem definovaného výsledku, v němž, narozdíl od normálního rozhodnutí, není žádná nejistota – tzv. **logický konektor** (konjunkce / disjunkce).

Pojem **vstupně-výstupní množina** zahrnuje veškeré vstupy a výstupy procesu. Tyto se rozlišují především z hlediska jejich účelu v procesu na:

- **materiálové**
- **informační**
- **smíšené.**

Materiálem se zde rozumí abstrakce jakékoliv formy cíleného produktu. Podle toho, zda jde o vstup, nebo výstup, se jí tedy rozumí buď „surovina“ v procesu dále zpracovávaná, nebo cílený „výrobek“ procesu. Z hlediska formy jím může být skutečná hmota, anebo třeba i informace. Důležitý je zde účel, nikoliv fyzikální povaha: jde o předmět zpracování¹². Má-li tedy „materiál“ povahu informace, jde o informaci procesem buď produkovanou, nebo rozpracovávanou, nikoliv pouze použitou k jeho řízení.

Informací se rozumí abstrakce jakékoliv formy řídicí informace. Jedná se o informace, nezbytné pro řízení procesu. Rozumí se jimi dodatečné informace, upřesňující platné okolnosti, například parametry příslušné události, atributy stavu(ů) jiného(ých) procesu(ů), na něž se navazuje apod.¹³

Smysl **smíšené** množiny spočívá ve faktu, že produktové a řídicí toky se často vyskytují souběžně. Rozumí se tím například tok suroviny současně s informací o kontextu jejího výskytu, nezbytně důležitou pro její další zpracování.

Vstupně-výstupní množiny vstupují (resp. musí vstupovat) do jistých vztahů s příslušnými činnostmi. Každá taková množina je buď výstupem výkonné činnosti, nebo vstupem, a to buď

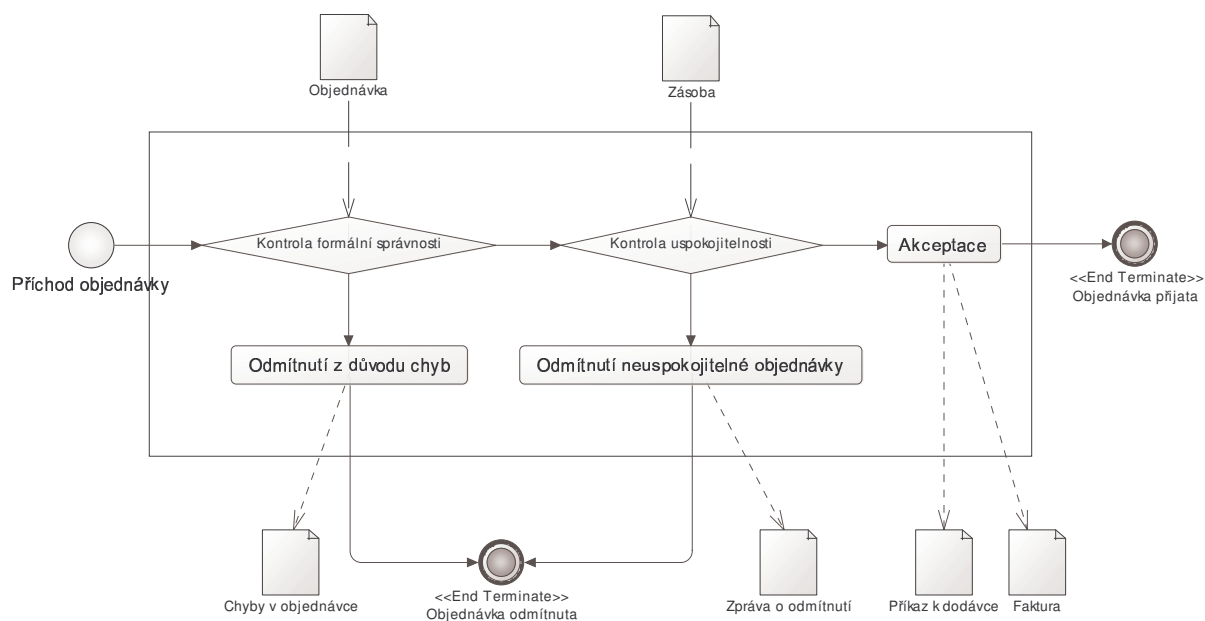
¹² Pro upřesnění smyslu tohoto rozlišení je dobré vzít v úvahu, že jde o *cílený* produkt. Tedy výstup procesu, daný jeho cílem (nebo vstup určený ke zpracování ve smyslu jeho cíle). Je totiž pravda, že i řídicí informace jsou „produktem“ procesu, nicméně jejich potřeba je dána okolnostmi procesu (například návaznostmi na jiné procesy), nikoliv jeho cílem.

¹³ Metoda ISAC, kterou je rozlišení základních druhů vstupně-výstupní množiny inspirováno, dokládá ještě další důležité pravidlo, a to že každá hmota nese současně i jistou informaci, v každém materiálovém vstupu je tedy třeba vidět současně i informaci potřebnou k řízení procesu, informaci o tom, že nastala nějaká událost, jejímž projevem je onen materiál. Technika modelování procesů registruje události jiným způsobem (jako samostatný element modelu), nicméně toto pravidlo říká, že existuje silná objektivní asociace mezi materiálovým vstupem do procesu a nějakou – příslušnou událostí. Tato verze metamodelu sice tuto asociaci explicitně nemodeluje, nepřímou však tento vztah eviduje v povinných vztazích mezi stimulem, výkonnou činností a materiálovým vstupem (sice ne úplně přesně, příslušná přesnost je však již pod úrovní záběru zde uvedené základní verze metamodelu).

do výkonné činnosti, nebo do rozhodnutí (což je zvláštní druh řídicí činnosti – viz výše). Jedna konkrétní vstupně/výstupní množina může vstupovat buď jen do rozhodnutí, nebo do výkonné činnosti, nikoliv do obou současně. To je v metamodelu vyjádřeno tak, že výkonná činnost a rozhodnutí jsou obě kompozitními agregacemi vstupně/výstupních množin (konkrétní vstupně-výstupní množina je tedy „majetkem“ jedné konkrétní činnosti).

Externím aspektem se rozumí jakákoliv entita (v obecném smyslu toho slova) z okolí procesu, která z jakéhokoliv důvodu s procesem souvisí. V tomto obecném metamodelu jsou naznačeny základní tři varianty externího aspektu: aktér, organizační jednotka a problém (všechny jsou diskutovány v této kapitole výše). Nemodeluje ani žádné specifické vazby těchto variant na jiné pojmy, pouze zcela obecnou asociaci mezi externím aspektem a jakýmkoliv jiným pojmem modelu, vyjadřující jejich obecnou souvislost. Metamodel netvrdí, že toto jsou jediné možné druhy externího aspektu, naopak počítá s rozšiřováním specifických variant externích aspektů, jakož i jejich případných specifických asociací, ve více specializovaných (méně obecných) verzích metamodelu¹⁴.

Jedním z nejdůležitějších faktů, jenž plyne z metamodelu, je tvrzení, že **významnou roli v podnikovém procesu hrají události a stavy**. Tyto dva konstrukty navíc spolu úzce souvisí. Události, z hlediska procesu uvažované jako vnější (představují objektivní, zevnitř procesu neovlivnitelný jev, jemuž se proces musí přizpůsobit), dělí proces na jakési objektivní elementární jednotky. V tomto smyslu je vhodné rozlišovat procesy primitivní a procesy komplexní.



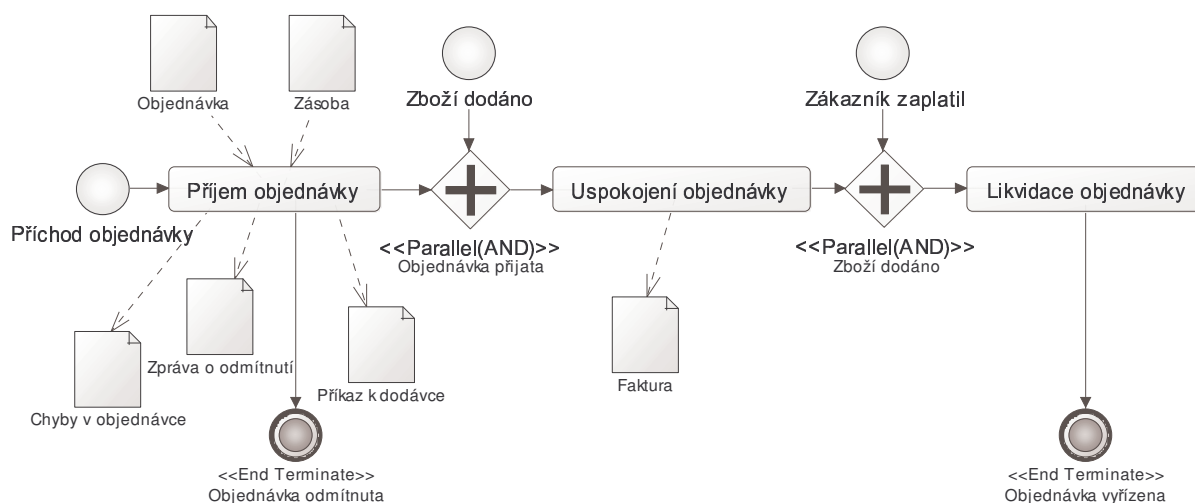
Obrázek B 4 Příklad primitivního procesu (Příjem objednávky)

Primitivní proces je takový, u něž není obecně objektivní (rozuměj: z hlediska procesu vnější) důvod jej popisovat podrobněji, než jako elementární činnost. Tímto obecně objektivním důvodem totiž může být jen vnější událost – fakt, jehož nastání nejsme schopni ovlivnit a musíme se mu v procesu přizpůsobit – tedy na něj čekat. Primitivní proces, jakkoliv

¹⁴ V příkladech rozvinutí tohoto pojmu se lze inspirovat květnatější pojetými nástroji procesního modelování, například tím bezkonkurenčně nejkvětnatějším, jímž je Aris Toolset, jenž ke standardním prvkům procesního modelu umožňuje asociovat mnoho nejrůznějších entit, například znalost, modul informačního systému, či část jeho datové základny apod.

může mít složitou vnitřní strukturu, neobsahuje žádný vnitřní stav – není u něj objektivní potřeba na nic čekat. Obrázek B 4 je schematickou ukázkou takového procesu. Jak je z popisu vidět, v celém procesu se vyskytuje jen jediná událost Příchod objednávky. Proces nemá žádný vnitřní stav, tedy nepopisuje nic víc, nežli jednorázové rozhodnutí (i když není úplně triviální, nebo předem definované) a s ním spojené bezprostřední akce a možné koncové stavy.

Komplexní procesy jsou takové, které mají (musí mít popsány) vnitřní stavy. Jejich nutnost je objektivní – vnitřní stavy znamenají vždy čekání procesu na nějakou událost. Obrázek B 5 je schematickou ukázkou komplexního procesu, jehož součástí – jednoduchou činností – je i výše zmiňovaný primitivní proces Příjem objednávky. Z popisu procesu je zřetelná role stavů v něm – představují vždy nutné čekání na akci vnějšího subjektu (dodávku materiálu skladem a platbu zákazníka, v tomto případě).



Obrázek B 5 Příklad komplexního procesu (Obchodní případ)

Je třeba zdůraznit, že obecný rozdíl mezi komplexním a primitivním procesem je vždy objektivní. Může existovat řada „subjektivních“ důvodů k podrobnějšímu popisu vnitřní struktury primitivních procesů, například různost zde sdružených činností, to, že mohou vyžadovat různou kvalifikaci, znalost, zastávají je různí aktéři, či organizace, mají významně odlišné trvání apod. Nicméně členění procesu na části vnějšími událostmi zůstane vždy jeho jediným obecně objektivním¹⁵ členěním.

Zde popisovaná technika modelování průběhu procesu neusiluje o standardizaci způsobu modelování procesů, ani jeho notace, jak tomu je u standardů, popisovaných podrobněji v kapitole D1. Je vyjádřením základních zákonitostí, jimiž se musí modelování podnikového procesu vždy řídit. Usiluje o to stát se jakousi esencí poznání v oblasti modelování podnikových procesů. Vzhledem k tomu, že jde o oblast, která je stále ve vývoji a lze

¹⁵ Obecná objektivnost dělení procesu je klíčově důležitá například při jeho podpoře technologií, která vždy znamená posun k obecnosti (jak je dáno již podstatou technologie, zejména je to viditelné u té „informační“). Takto zjištěné stavy procesu je například nutno uvažovat jako místa, kde musí docházet k ukládání dat, neboť jsou to místa synchronizace vzájemně asynchronních akcí v procesu (tedy přesněji řízených akcí procesu a neřízeného výskytu vnější události, prostě řečeno – čekání na událost, jež není v naší moci^{33a}).

^{33a} Což je ovšem také relativní (je například otázkou, zda fakt, že různé činnosti dělají různé organizace, není třeba považovat za vnější vliv – v takovém případě musí vždy rozhodovat, zda jsou všechny činnosti procesu řízeny procesem, nebo nikoliv (jsou-li všechny organizace zcela ovládnuty procesem, není možné uvažovat jejich činnosti jako objektivně vnější), u každé události je tedy třeba zkoumat její podstatu, zda je „objektivní“).

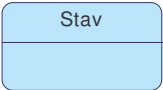

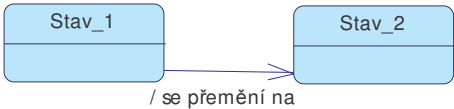
usuzovat, že také stále bude, je vývoj této techniky součástí procesu vývoje tzv. Meta-modelu podnikového systému, jenž je hlavním předmětem zájmu projektu OpenSoul. Pro bližší informace o tomto projektu viz <http://opensoul.panrepa.com/>.

B.1.3 Diagram stavů

Diagram stavů - zachycuje stavy jednoho objektu z diagramu tříd, kterými prochází (nebo lépe řečeno může projít) v průběhu svého životního cyklu v systému. Ke změně stavu může dojít konkrétní událostí nebo i pouhým plynutím času. Každý objekt má svůj počáteční stav a musí mít i stav koncový (nebo i více alternativních koncových stavů, narozdíl od počátečního, jenž může být vždy pouze jeden).

Diagram popisuje stavy objektu a možné přechody mezi těmito stavy. Přechody mezi stavy, které nejsou diagramem popsány, jsou nepřípustné. Každý přechod mezi stavy je popsán dvojicí údajů, z nichž první vyjadřuje důvod přechodu (událost, či podmínku) a druhý způsob jeho provedení (akci, metodu).

Konstrukce diagramu:

Konstrukt	význam
	Stav objektu.
	Počáteční „startovní“ událost.
	Koncový stav objektu.
	Přechod z jednoho stavu k druhému.

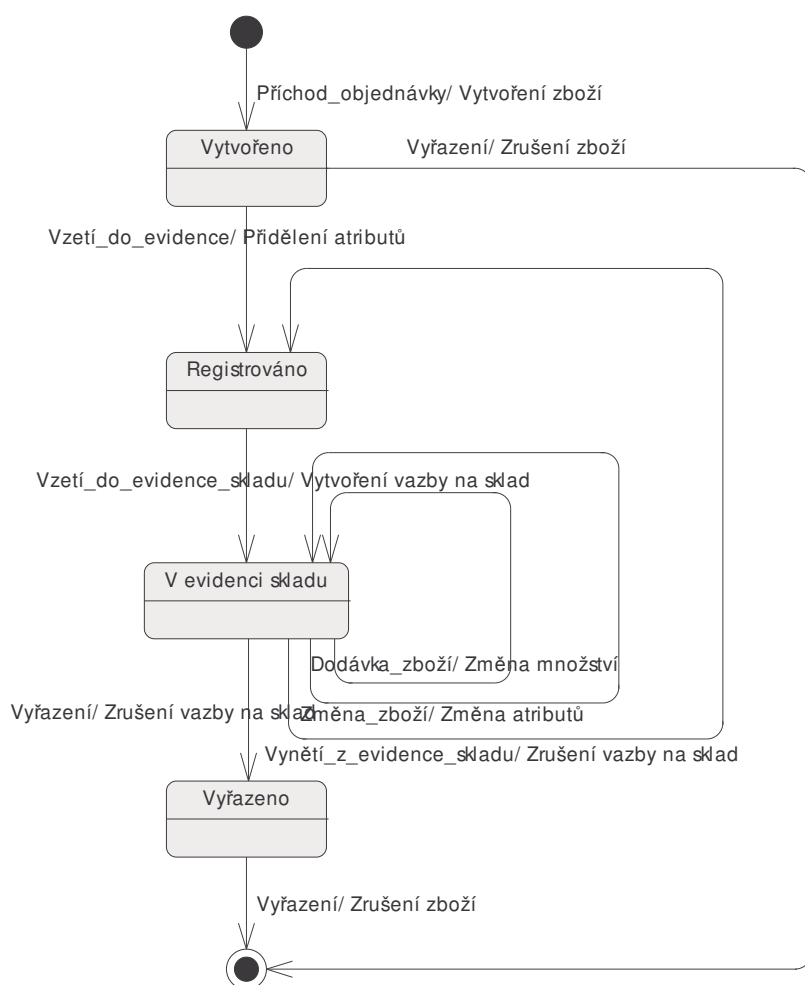
Diagramy stavů jsou obecně určeny k popisu přechodů mezi stavy třeba i více souvisejících objektů, mohou tak sloužit (a často jsou tak používány) například jako ještě další pohled na „komunikaci“ objektů (vedle diagramů komunikace a posloupností). V této metodice slouží diagramy stavů výhradně popisu životních cyklů vybraných (klíčových) objektů. Znamená to, že jedním diagramem stavů je popisována jedna třída objektů z diagramu tříd. Ne všechny třídy objektů jsou svou vnitřní dynamikou natolik analyticky zajímavé, že je nutné popisovat stavovým diagramem jejich životní cyklus.

Každý diagram stavů je nutné navázat na objekt, jehož životní cyklus popisuje. K tomu slouží v nástroji Power Designer v „Properties“ diagramu položka „Default classifier“.

Doporučený postup tvorby diagramu je následující:

- v OO modelu vytvořit nový StateChart

- b) v Properties nového diagramu nastavit jako Default Classifier příslušnou třídu z diagramu tříd, jejíž životní cyklus bude popisován,
- c) specifikovat seznam událostí (Events), které budou použity jako důvody přechodů mezi stavy objektu (seznam je možno tvořit i průběžně při popisu přechodů),
- d) specifikovat možné stavy objektu a možné přechody mezi nimi,
- e) při popisu každého přechodu mezi stavy (s výjimkou startovního přechodu) specifikovat na záložce Trigger některou z událostí jako Trigger Event a vybrat některou z metod třídy jako Operation tohoto přechodu. Při specifikaci metody (Operation na záložce Trigger) se nabízejí pouze metody té třídy, která je nastavena jako Default Classifier diagramu (podobně jako u událostí, ani u metod třídy není nutné, aby byly specifikovány dopředu. Metody je do objektu, nastaveného jako Default Classifier, možno doplňovat i průběžně při popisu přechodů).



Obrázek B 6 Příklad životního cyklu objektu Zboží

Obrázek B 6 popisuje ukázkou životního cyklu třídy objektů Zboží. Určuje, jakými stavy může objekt třídy Zboží ve svém životě procházet a jakým způsobem. Z diagramu je například patrné, že Zboží může být vyřazeno buď ze stavu „Vyřazeno“, nebo ze stavu „Vytvořeno“, z ostatních stavů není přímé vyřazení možné, pouze přes stav „Vyřazeno“. Je také vidět, že stavy „Registrováno“ a „V evidenci skladu“ se mohou v životě objektu opakovat, narozdíl od

ostatních dvou stavů, které jsou jednorázové. U každého přechodu je popsán a příslušná událost, které přechod způsobí a příslušná metoda třídy Zboží, kterou je přechod proveden. Popis života objektu musí být vždy úplný – musí tak být specifikovány veškeré možné události, které jsou pro daný objekt významné a jejich důsledky v životě objektu. Každému stavu objektu také odpovídá specifická kombinace hodnot atributů tohoto objektu.

B.1.4 Provázání procesů s třídami objektů pomocí jejich životních cyklů

Model procesů a model tříd objektů musí být mezi sebou vzájemně provázány. Toto provázání je nutné, jelikož odráží existující logické vazby mezi oběma těmito pohledy na reálný systém:

- pohledem na reálný systém jako na strukturu objektů a jejich podstatných (stálých, relativně neměnných) vztahů – tento pohled představuje model tříd (business) objektů,
- pohledem na reálný systém jako na strukturu vzájemně navazujících činností, zpracovávajících vstupy (suroviny) na výstupy (produkty), a to v zájmu definovaného cíle – tento pohled představuje model (business) procesů.

Jedná se o dva pohledy na jeden jediný reálný systém, kde prvky tohoto systému, viděné jedním pohledem, musí obsahově korespondovat s prvky tohoto systému, viděnými pohledem druhým. Konkrétně například business objekty, popisované v diagramu tříd, se musí vyskytovat v procesech jako aktéři, nebo vstupy, či výstupy, anebo alespoň jako další související aspekty, jako například organizační jednotky, pracovní funkce nebo role, technologické systémy a jejich prvky atd.

Smysl tohoto vidění systému ze dvou stran spočívá v tom, že oba pohledy jsou různé – různé pohledy na totéž umožňují „prostorový vjem“, jenž synergicky přináší novou informaci, která se nevyskytuje ani v jednom z obou pohledů, nýbrž jen v celku jimi tvořeném.

Protože, ač hledí na totéž, jsou oba pohledy různé, je nutno mít především jasno v tom, co společného se za nimi skrývá - jak prvky jednoho vidění systému obsahově korespondují s prvky druhého vidění.

A protože oba pohledy jsou různé, nelze ani očekávat, že zmiňovaná korespondence bude příliš jednoduchá – jeden objekt, identifikovaný v modelu tříd, se typicky může vyskytovat v několika různých podobách v jednom procesu (například současně jako aktér i jako informační vstup). A naopak, prvek procesního diagramu lze vidět v diagramu tříd spíše jako účelovou kombinaci tříd (z nichž je navíc zpravidla relevantní jen část jejich atributů a metod) a jejich asociací, než jako jedinou třídu v plné šíři jejího popisu.

Následující sada pravidel o provázání modelu tříd s modelem procesů shrnuje výše popisované skutečnosti:

- A) Každá třída objektů z modelu tříd musí být zastoupena v modelu procesů v alespoň jednom z jeho vstupů, či výstupů a/nebo aktérů, či jiných externích aspektů.**
- B) Každý vstup, či výstup procesu, jakož i každý externí aspekt procesu, musí být zastoupen v modelu tříd jako třída, nebo asociace mezi třídami, či jako kombinace obojího.**

Výše popsaná pravidla se týkají jedné stránky reality, popisované oběma modely – stránky existenční. Tato stránka reality zahrnuje objekty a artefakty a jejich obecné (podstatné, nadčasové) vztahy. V obou modelech (pohledech) je však zastoupena ještě druhá podstatná

stránka reality, a tou je stránka procesní. Tato stránka reality zahrnuje akce a stavy a jejich (časové) návaznosti a je rovněž zastoupena v obou pohledech – jak v procesním, tak v objektovém. Zatímco v procesním modelu je tato stránka reality přirozenou a nikdo o ní nepochybuje, v objektové pohledu je zpravidla třeba si její projev uvědomit. K tomu slouží popis „životního cyklu třídy objektů“.

B.1.4.1 Životní cyklus třídy objektů

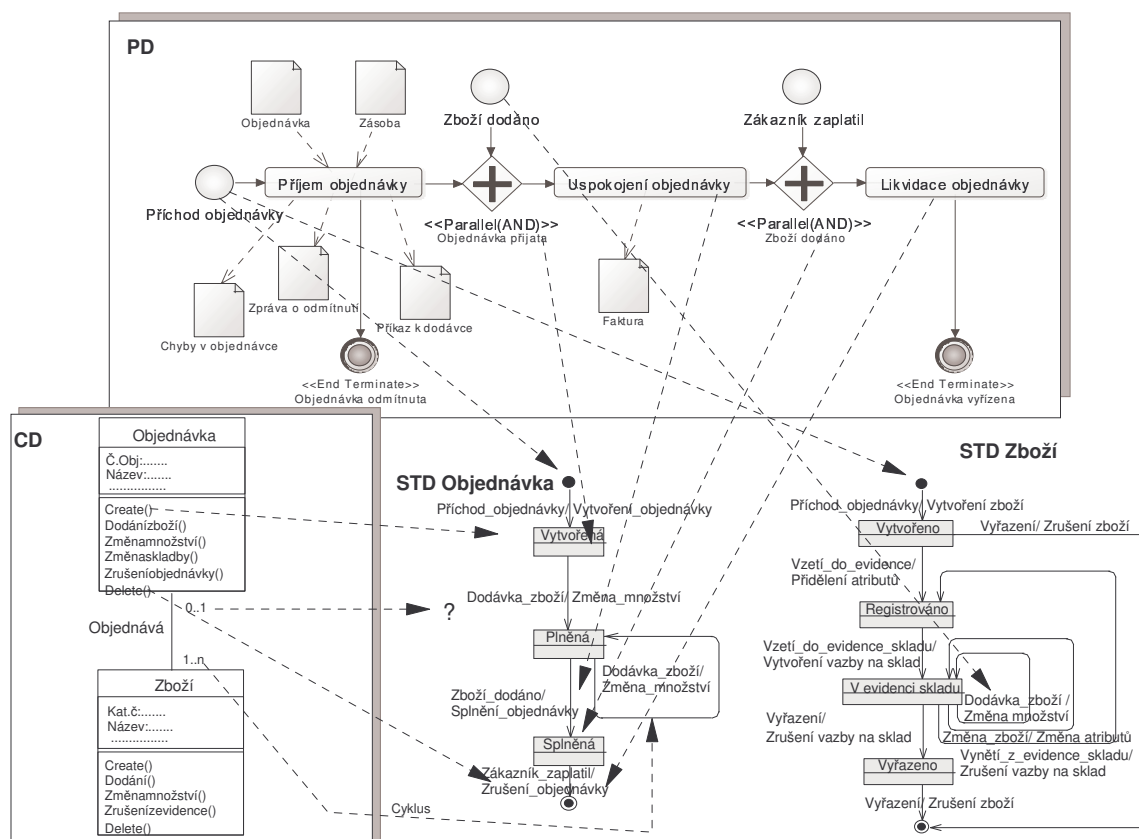
Životní cyklus třídy objektů vyjadřuje dynamiku objektu dané třídy – popisuje jak se objekt obecně vyvíjí v čase, jakými obecnými stadii svého vývoje prochází od vzniku až po zánik. Jedná se o procesní popis – život objektu je procesem přechodů z jednoho stavu objektu do jiného s tím, že jeden ze stavů objektu je stavem počátečním a jeden stavem konečným. K takovému popisu životního cyklu třídy se používá stavový diagram, podrobněji popisovaný v předchozí kapitole. Pro každou relevantní třídu (ne pro všechny třídy je nutné popisovat jejich životní cykly) z diagramu tříd je vytvořen stavový diagram, popisující její životní cyklus.

Životní cyklus třídy objektů je tedy procesním popisem objektů příslušné třídy. Je však diametrální rozdíl mezi procesním popisem ve smyslu business procesu a procesním popisem života třídy objektů. Každý z obou těchto modelů totiž popisuje realitu z hlediska určité své dimenze – bytí versus chování. V obou dimensích mají své místo procesy – buď ve významu životního cyklu, nebo věcného – cíleného procesu (business process). Z hlediska čistě „technického“ tedy jak věcné procesy, tak procesy životních cyklů podléhají stejným pravidlům – vyjadřují řazení akcí v závislosti na událostech a stavech. Rozdíl mezi nimi je toliko významový – životní cyklus je procesním vymezením určitých zákonitostí, jež musí být respektovány jakýmkoliv chováním, zatímco věcný – cílený proces je popisem tohoto chování - vyjádřením způsobu dosažení určitého cíle, vytvoření určitého produktu¹⁶. Nestačí tedy, aby metodika paušálně požadovala procesní popis, ale musí přesně rozlišovat čeho se tento popis týká. Narozdíl od business procesu je tedy životní cyklus třídy procesním vymezením zákonitostí, které budou muset být všemi business procesy respektovány. Pro takové zákonitosti se také používá pojem „business rules“.

Životní cyklus třídy objektů stavovým diagramem popisuje možné stavy, jimiž objekt za své existence může procházet a možné přechody mezi nimi. Přechody, které nejsou v diagramu popsány, jsou neregulérní – objektivně nepřipustné, či nemožné. Žádný business proces nemůže směřovat k takové nepřipustné kombinaci stavů. Všechny stavy pak vymezují objektivní kompletnost života objektu. Systém business procesů musí nějakým způsobem pokrýt všechny stavy všech relevantních objektů a všechny možné přechody mezi nimi, jinak nemůže být považován za kompletní (tj. nezaručuje, že je schopen adekvátně reagovat na všechny možné kombinace událostí, které může život přinést).

Základní směry provázání procesů s třídami objektů naznačuje Obrázek B 7.

¹⁶ Z uvedeného rozdílu mimo jiné jasně plyne, že k modelování podnikových procesů naprosto nestačí je pojímat z čistě technického hlediska. Právě jejich účelovost, členost, tvoří podstatu jejich významu.



Obrázek B 7 Ilustrace vztahů mezi procesním a objektovým modelem prostřednictvím životního cyklu objektu

Na obrázku je patrné jak spolu korespondují oba zmiňované procesní pohledy na realitu. Oba popisují reakce na stejné události, ale ze dvou různých hledisek – jako zákonitosti vývoje objektu v čase a jako následnosti akcí business procesu. Tato dvourozměrnost umožní vidět souvislosti, neviditelné ani z jednoho samotného pohledu. Oba pohledy se tak jednak doplňují, jednak vzájemně korigují – modely business procesů, popisující kroky k dosažení cíle, určují smysluplnost kombinací jednotlivých událostí a reakcí na ně, zatímco modely životních cyklů objektů, popisující zákonitosti vývoje objektu, definují, které kombinace akcí jsou objektivně správné a vymezují též, co vše musí být systémem business procesů vzato v úvahu, aby byl objektivně kompletní. Diagram stavů tak především slouží detailnímu mapování akcí a stavů procesu na akce a stavy objektů. Toto mapování vysvětluje detaily složitých existenčních vztahů mezi procesy a objekty (v jakém smyslu se každý objekt částečně obrací v různých prvcích a místech procesů a naopak).

Konkrétně je z příkladu na obrázku vidět, jak oba objekty (Objednávka a Zboží) spolu souvisí nejenom existenčně (pomocí asociace „Objednává“, jak je vidět v diagramu tříd), ale i akčně: událost „Příchod objednávky“ je významnou událostí - konstruktorem jak pro „Objednávku“, tak i pro „Zboží“, rovněžtak událost „Zboží dodáno“ znamená jak plnění „Objednávky“, tak současně změnu stavu „Zboží“ na skladě. Obě události jsou podrobnějším způsobem naplnění existujícího vztahu mezi oběma objekty, popsáno v diagramu tříd. Současně jsou tyto události základním pojítkem mezi objektovým a procesním pohledem na reálný svět, vysvětlujícím základní detailní souvislosti těchto pohledů. Přitom ani jeden z těchto pohledů není zárukou úplnosti popisu reality, zatímco současné respektování obou pohledů je tak nástrojem dosažení takové úplnosti (ovšemže relativního – vždy nakonec nejvíce záleží na tom, co vše je konkrétní analytik schopen vidět).

Následující sada pravidel doplňuje předchozí pravidla o náležitosti popisu životních cyklů, jak se projeví ve vzájemných vztazích mezi modelem tříd, modelem procesů a modelem životního cyklu objektu:

- C) Každá třída objektů z modelu tříd musí mít specifikovány alespoň tři metody:*
 - *metodu, již objekt (instance třídy) vzniká (konstruktor),*
 - *metodu, již objekt (instance třídy) zaniká (destruktor),*
 - *alespoň jednu metodu, již se mění atributy objektu (transformer).*
- D) Pro každý atribut třídy objektů z modelu tříd musí být u této třídy specifikována metoda, již je tomuto atributu přidělena počáteční hodnota a metoda, již je hodnota atributu změněna.*
- E) Pro každou asociaci mezi třídami musí být u každé takto asociované třídy specifikována metoda, odpovídající této asociaci.*
- F) Ke každé třídě objektů, která není považována za primitivní, je přiřazen stavový diagram, popisující její životní cyklus.*
- G) Stavový diagram životního cyklu třídy musí mít popsány všechny přechody mezi všemi svými stavy. Popis každého přechodu specifikuje jednak událost, na základě které se přechod uskuteční (spoušť), jednak metodu, již se tento přechod uskuteční.*
- H) Stavový diagram životního cyklu třídy musí obsahovat v popisech přechodů mezi stavy všechny metody této třídy. Popisy přechodů mezi stavy nesmí obsahovat metody, které nejsou specifikovány u této třídy.*
- I) Každá událost, specifikovaná v popisech přechodů ve stavovém diagramu životního cyklu třídy, musí korespondovat s událostí, specifikovanou v popisu nějakého (nějakých) business procesu (procesů).*

B.1.5 Popis funkčnosti IS - Diagram datových toků

zobrazení funkčního modelu systému. Je jedním ze základních nástrojů analytického modelu IS. Funkční model popisuje funkce a jejich návaznosti. Funkce určují, jaké procesy mají probíhat v informačním systému, který má být věrným modelem podporované reality. Tyto procesy informačního systému jsou odrazem procesů, z nichž se realita skládá. Funkční model však není procesním modelem (ačkoliv popisuje procesy), u jednotlivých funkcí je popisována jejich existence a podstatné vazby mezi nimi prostřednictvím toků dat – jedná se o popis statický. Důležité je, že popisované procesy jsou procesy informačního systému, tedy procesy modelující, nikoliv informačním systémem podporované (modelované) business procesy, které se ve funkcích pouze odrážejí. Na úrovni analýzy systému tak postačuje statický popis funkcí (neprocesní), neboť veškeré procesní aspekty *chování reality* jsou na této úrovni plně modelovány popisem business procesu, zatímco procesní aspekty *chování informačního systému* nejsou již předmětem analýzy reality. Procesní popis specifických procesů informačního systému se tedy ve fázi analýzy nevyskytuje a je záležitostí až designu systému (viz kap. B.2.3 Diagram procesů (rozmístění)). **Funkční model** systému je tak **základním analytickým zadáním** a východiskem designu systému.

DFD se vyvinul z tzv. Activity Diagrams, používaných v metodice SADT. SADT (Structured Analysis and Design Technique) je metodika, používaná cca od poloviny sedmdesátých let. Nejlépe je DFD metodicky uchopen v díle Eda Yourdona (viz www.yourdon.com).

V DFD se používají základní prvky:

- funkce
- datový tok (DATA FLOW)
- datový sklad (DATA STORE)
- terminátor (externí entita)

B.1.5.1 Funkce



Značí se kolečkem.

Jde o informační procesy (zpracování dat), jimiž je modelováno reálné dění. Funkce znázorňuje transformaci dat, která vede k vyprodukování výstupu (transformace vstupu na výstup).

Tradičně se rozlišují dva druhy procesů, popisovaných v DFD datové (funkce) a řídicí:

Datový proces - funkce vyjadřuje fyzickou transformaci dat, tj. změnu reprezentace dat, nebo změnu stavu určité části dat, tj. změnu hodnot údajů, vznik nových údajů. Hlavním úkolem funkce (datového procesu) je tedy zpracovávat, transformovat data.

Řídící proces vyjadřuje algoritmus řízení (vzájemných časových návazností) funkcí v určité části systému. Používá se k zachycení real-time charakteristik aplikace. Narozdíl od funkce není úkolem řídicí funkce zpracovávat data. V systémech zpracování hromadných dat tak řídicí procesy ztrácejí smysl, resp. neměly by být uvažovány ve fázi analýzy, ale až ve fázi designu (viz kap. B.2.3 Diagram procesů (rozmístění)). Lze ukázat, že i z hlediska principů objektové orientace je obecně přítomnost procesů informačního systému v analytických modelech nesmyslná. Z těchto důvodů nebude tomuto prvku DFD v dalším textu věnována pozornost.

Každá funkce v DFD musí mít název a jednoznačnou identifikaci v hierarchii funkcí. Tento identifikátor se skládá z identifikace bezprostředně nadřazené funkce, jejíž částí označovaná funkce je (např. 3.1.2), a identifikace v rámci úrovně svého diagramu (např. 5, celé označení funkce : 3.1.2.5.). Je-li k identifikaci použito číslování funkcí, pak v rámci úrovně má význam pouze identifikační a v žádné případě nevyjadřuje pořadí provádění (neboť funkční model není modelem procesním, pouze staticky specifikuje funkce a jejich datové vztahy).

B.1.5.2 Datový tok



Značí se orientovanou šipkou.

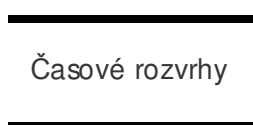
Jde o abstrakci jakékoliv formy přesunu dat.

Datový tok vyjadřuje přesun dat/informací z jedné části systému do jiné, nebo z okolí systému do systému, anebo ze systému do okolí. Znázorňuje se šipkou (data tekou naznačeným směrem, je možné použít i dialog flow - stejná data tekou oběma směry).

Datový tok musí mít známý obsah a musí být pojmenovaný (s výjimkou datových toků do datastoru a z něj - viz dále). Název datového toku musí daná data reprezentovat a jasně vyjadřovat jejich obsah. (např. "objednávka"), nikoliv pouze formu (např. "data", "formulář" apod.).

Datové toky obsahují ta data, která jsou systémem zpracovávána a ukládána. I když původně byl Diagram datových toků vyvinut i k popisování toku dokumentů, materiálu, zboží, surovin apod. (např. ve smyslu workflow), není vhodné k tomuto účelu tento diagram používat. Pro takové použití bychom se museli vzdát celé řady, pro vývoj informačních systémů metodicky poměrně cenných pravidel použití DFD. Kromě toho, při takovém použití nabývají jednotlivé konstrukce DFD poněkud jiného, mírně posunutého významu, který lze mnohdy od původního významu jen těžko rozlišit, a tak se takovéto použití jednoho nástroje ke dvěma odlišným účelům zbytečně stává semeništem analytických chyb. A konečně - k tomuto účelu vznikly jiné, vhodnější nástroje, například A-grafy z metodiky ISAC (viz), nebo nástroje modelování business procesů.

B.1.5.3 Data store



Značí se dvěma vodorovnými čarami – technickým symbolem přerušení.

Jde o abstrakci jakékoliv formy uložení dat.

Data Store (skladiště dat) vyjadřuje "depozitář" dat (data uchovaná pro jejich pozdější použití). Znázorňuje se pomocí dvou rovnoběžek, mezi nimiž je umístěn název - tento symbol, používaný v technické symbolice jako symbol pro přerušení, připomíná, že uložení dat znamená přerušení toku dat v čase - tento fakt má při modelování procesů dalekosáhlý význam (viz později popisovanou techniku analýzy událostí).

Význam data store je tedy "místo dočasného uchování dat". Může být implementován různě (jako pole, soubor - obyčejný nebo databázový, ale i jako cokoli jiného, například šanon, kniha apod.). Používá se všude tam, kde mezi procesy existuje časově zpožděné předávání dat (asynchronní). Asynchronnost procesů musí vyplývat vždy z jejich podstaty, nikoliv z formy implementace (to je záležitostí technologického, nebo implementačního modelu). Na konceptuální úrovni platí pro Data Store totéž, co již bylo řečeno o datovém toku - je abstrakcí jakékoliv formy uložení dat, vyjadřuje pouze fakt, že data jsou uschována (čili jejich tok je v čase přerušen) a neříká nic o konkrétní formě tohoto uložení.

Název Data Store by měl být v množném čísle (např. "dodavatelé").

Pro každý Data Store musí existovat alespoň jeden datový tok dovnitř (uložení dat) a jeden ven (použití dat). Tento tok může vyjadřovat jakoukoliv substrukturu struktury Data Store:

- skupinu (výskyt) dat (např. zaměstnanec),
- víc skupin dat vybíraných ze storu,
- část jednoho výskytu (např. jen telefon),
- část z více výskytů.

Data Store je pasivním prvkem diagramu - data do a z něj musí vždy téci přes transformaci dat (funkci).

Datastore je, vedle datového toku, další formou propojení (komunikace) funkcí.

B.1.5.4 Terminátor



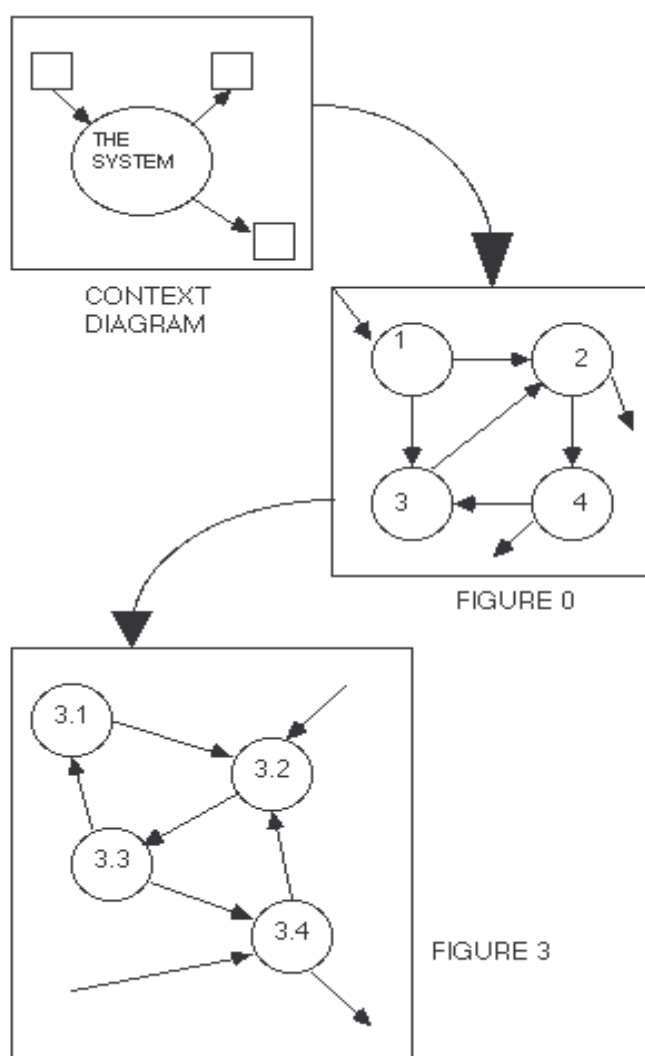
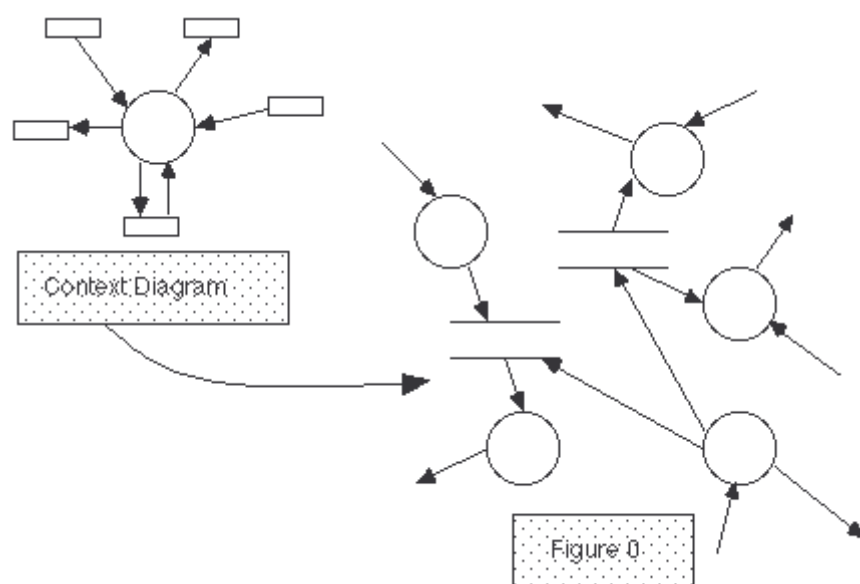
Finanční instituce

Značí se čtvercem/obdélníkem.

Představuje objekty, které nepatří do popisovaného systému, nýbrž do jeho podstatného okolí. Terminátor (počátek, případně konec datového toku, zdroj dat, případně místo a účel spotřeby dat) znázorňuje externí zdroj, nebo místo určení dat (někdy se též nazývá externí entita - objekt). Vyjadřuje tedy okolí systému, s nímž systém komunikuje.

Fyzicky to může být člověk, skupina lidí (oddělení), skupina oddělení ve stejné organizaci / podniku, ale vždy je to objekt vně modelovaného systému. Může to být i jiný systém, který není přímo v centru zájmu našeho modelu, ale má na něj vliv.

I terminátor by měl mít výstižný název vyjadřující typ externího zdroje/místa určení (např. "zákazník", a ne "pan Dlouhý"). Při jisté míře abstrakce lze tedy i terminátor považovat za další formu komunikace procesů (prostřednictvím okolí systému).



Obrázek B 8 Hierarchická struktura DFD (zdroj: Yourdon.com)

Jak ilustruje Obrázek B8, popis funkčnosti systému předpokládá nikoliv jediný DFD, ale celou hierarchickou soustavu těchto diagramů. Každá funkce může být podrobněji popsána samostatným diagramem na nižší úrovni (tj. vyšší podrobnosti). V takové soustavě diagramů se přirozeně opakují prvky na rozhraní funkcí, což vyvolává obecné nebezpečí nekonsistence modelů obou úrovní (tedy situace, kdy podrobnější model popisuje stejný prvek rozhraní jinak, než model vyšší úrovně). To se nazývá hierarchickou konsistencí diagramů.

B.1.5.5 Metoda zkoumání událostí (Event Partitioning Approach (EPA))

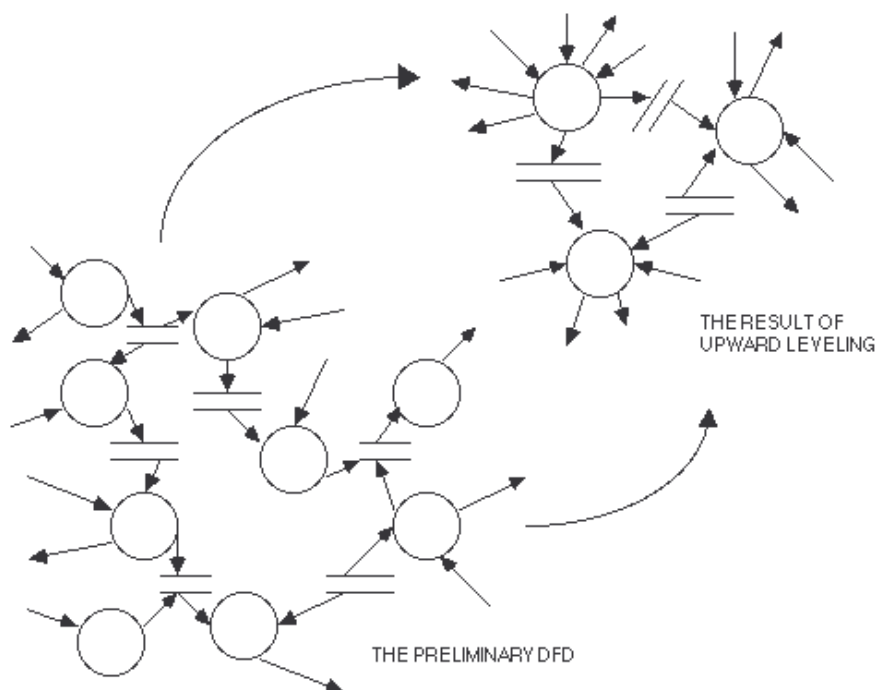
Vhodnou metodou k vytváření DFD je metoda zkoumání událostí. Jedná se o metodu k odvození funkcí a data storů na základě procházení seznamu událostí na něž by měl vyvíjený systém umět reagovat.

Postup, který metoda EPA navrhuje je následovný:

1. Sepíše se seznam událostí, na které by měl systém reagovat a ke každé se nakreslí symbol pro funkci.
2. Funkce je pojmenována tak, že se popíše reakce systému, která je touto událostí vyvolána.
3. K funkci jsou přiřazeny odpovídající vstupy a výstupy, které jsou v případě potřeby asynchronní komunikace mezi funkcemi doplněny o data story.
4. Výsledný DFD je porovnán s kontextovým diagramem a seznamem událostí kvůli úplnosti a konsistenci.

Výstupem tohoto postupu je DFD diagram nejnižší úrovně. Z něj je sdružováním příbuzných prvků potřeba vytvořit hierarchicky se rozpadající množinu diagramů tak, aby každý takto vytvořený diagram byl dostatečně malý a přehledný (viz Obrázek B 9).

Více o metodě zkoumání událostí a o práci se získaným DFD nejnižší úrovně je možno se dozvědět na www.yourdon.com, přesněji <http://www.yourdon.com/strucanalysis/index.html>.



Obrázek B 9 Tvorba DFD vyšší úrovně sdružováním funkcí (zdroj: Yourdon.com)

B.1.5.6 Data Flow Diagram (DFD) a udržování konzistence s ostatními diagramy

Tato kapitola popisuje potřebu udržování konzistence mezi modely z pohledu DFD. Diagramy, se kterými je potřeba počítat jsou Class Diagram (CD, diagram tříd), Procesní diagram (PD) a případně též State-Transition Diagram (STD).

Konzistence DFD a objektového modelu (Class Diagramu – CD)

Class Diagram (CD) představuje z hlediska DFD *statický pohled na datovou základnu* modelovaného systému. Jsou v něm zachyceny prvky pro uložení dat – třídy a jejich atributy – a množina operací, které lze nad těmito daty provádět ve formě metod objektu. DFD je na druhé straně model, který zachycuje interakci systému s okolím, jeho reakce na externí události, a zaznamenává to, jakých datových struktur se pochody v systému, v závislosti na spouštěcí události, dotknou.

Styčnými body obou diagramů jsou tedy především data, a také, do jisté míry, operace které jsou nad daty prováděny (i když ne ve vztahu 1:1).

Při udržování konzistence dat mezi DFD a CD je nutné tedy především udržovat vazbu mezi každým Data Store a nějakou danou strukturou tříd a jejich vazeb v CD. Dále je vhodné udržovat nějakou vazbu mezi metodami třídy a toky dat v DFD. Tato konzistenční vazba je však již mnohem méně těsná a proto je možné ji při technických obtížích s jejím zaznamenáváním do modelovacího nástroje oželeť.

Následující sada pravidel doplňuje pravidla předchozí kapitoly B.1.4 o náležitosti vztahů mezi modelem tříd (CD) a diagramem datových toků (DFD):

- J) Každý elementární Datastore¹⁷ v DFD musí být v CD zastoupen jako třída, nebo asociace, anebo kombinace obojího.***
- K) Atributy každého elementárního Datastore z DFD musí být datovou strukturou atributů tříd, jimiž je tento Datastore v CD zastoupen.***
- L) Metody každé elementární funkce¹⁸ z DFD musí být algoritmickou strukturou metod tříd, jimiž jsou v CD zastoupeny Datastorey, spojené datovými toky s touto funkcí.***

Konzistence DFD a procesního modelu

Procesní diagram (PD) je modelem reálných (business) procesů. Viděno očima informačního systému - tyto procesy jím procházejí napříč, informační systém jim poskytuje příslušnou informační podporu. Jako model reálného (business) procesu se procesní diagram nezabývá strukturou informačního systému, ale zaznamenává postup transformace vstupů do systému na výstupy. Zároveň sleduje zdroje, které jsou v této transformaci spotřebovávány.

Jednotlivé činnosti, které jsou zobrazeny v PD, mají v jisté formě svůj obraz v procesech DFD, i když se nejedná o vztah 1:1 ale spíše 0..n:0..n. Důvodem tohoto zamlženého vztahu je především různé zaměření i charakter obou diagramů – jeden popisuje proces a druhý strukturu funkcí systému. Nicméně přesto by činnosti business procesu měly mít svůj obraz v DFD.

Vzhledem k tomu, že procesní model i DFD pracují s událostmi, měly by se stejné události objevovat v obou diagramech, nebo by se mezi událostmi z DFD a PD měla objevovat vazba.

¹⁷ Elementární Datastore je DataStore u nějž není objektivní důvod k jeho rozkladu do podrobnější struktury DataStoreů (tedy neskrývá v sobě rozporné struktury, jak jsou definovány M.A.Jacksonem).

¹⁸ Elementární funkce je funkce, u níž není objektivní důvod k rozkladu do DFD nižší úrovně (tedy neskrývá v sobě rozporné struktury, jak jsou definovány M.A.Jacksonem).

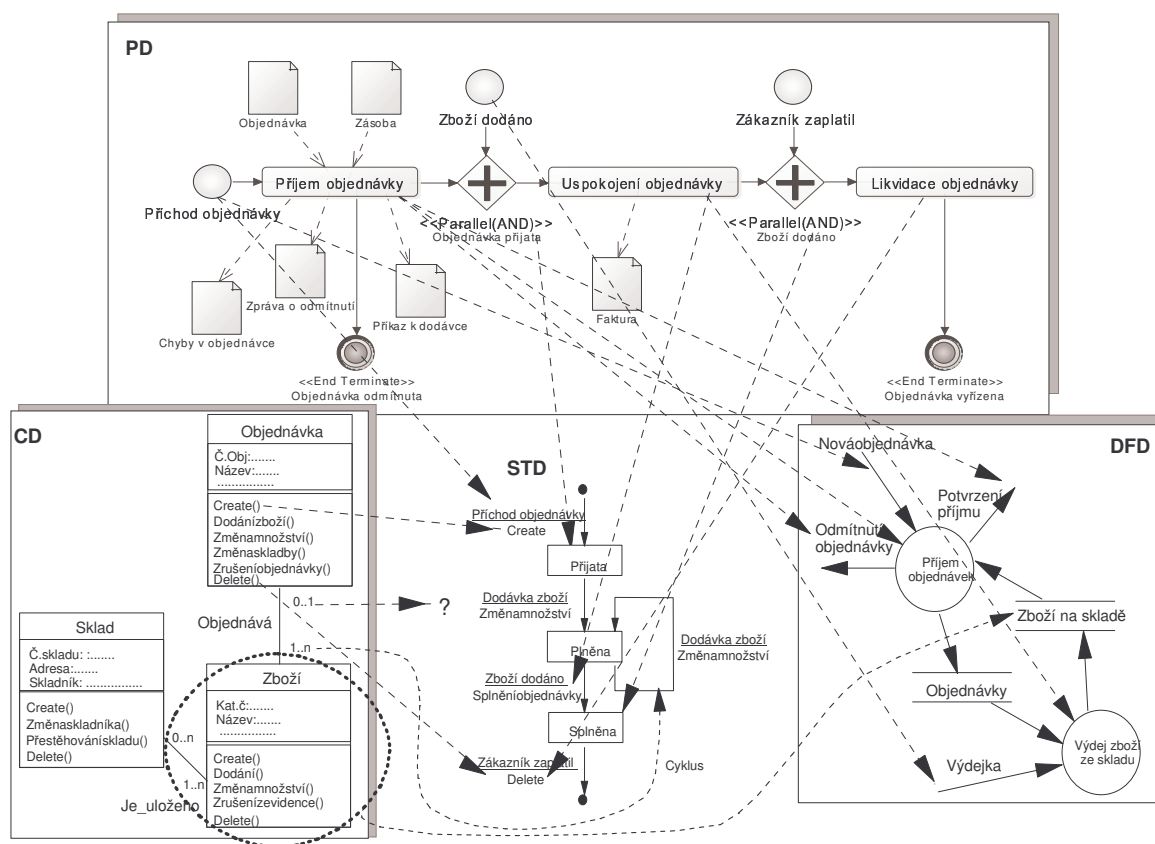
Následující sada pravidel doplňuje pravidla předchozí předchozí kapitoly B.1.4 o náležitosti vztahů mezi modelem procesů (PD) a diagramem datových toků (DFD):

- M) Každý elementární vstupní datový tok v DFD od terminátoru (tj. zvnějšku systému) musí odpovídat nějaké události, specifikované v popisu nějakého (nějakých) business procesu (procesů) v PD.**
- N) Každý stav každého procesu v PD musí korespondovat s některým(i) elementárním(i) Datastorem(y) v DFD a naopak každý elementární Datastore v DFD musí korespondovat s některým(i) stav(y) procesů(ů) v PD. Jde o korespondenci M:N.**

Konzistence DFD a STD

State-Transition diagram popisuje životní cyklus objektů nějaké třídy. Obsahuje stavy, ve kterých se objekt dané třídy může nacházet a přechody mezi těmito stavy. Přechody jsou realizovány nějakou akcí, která nastane na základě události z vnějšího světa. V tomto případě je opět přechod vyvolán nějakou událostí, jež může mít obraz v DFD i v PD. STD, jako základní nástroj popisu souvislostí mezi objektovým a procesním modelem reality (viz předchozí kapitoly včetně tam specifikovaných pravidel konsistence), jsou tak z hlediska DFD skryty za těmito modely a nemusí být vůbec z hlediska funkčnosti systému brány v úvahu.

Obrázek B 10 ilustruje některé základní vztahy mezi modely, popisované výše.



Obrázek B 10 Konsistence modelu procesů, objektů a DFD

B.1.5.7 Implementace Diagramu datových toků v nástroji PowerDesigner

V předchozím textu byly diskutovány potřeby udržování konzistence diagramu datových toků s ostatními diagramy. Protože nástroj PowerDesigner standardně neobsahuje model, odpovídající DFD, je potřeba využít širokých možností rozšiřitelnosti tohoto nástroje, a model v něm vytvořit.

V rámci této metodiky byl DFD implementován pomocí úpravy - sdpecializace diagramu tříd. DFD implementovaný na bázi modelu tříd pak může být snadno propojen s STD i CD a dále i s designovými modely (viz kapitolu B2)., což je žádoucí zejména z důvodu možností zajištění konzistence modelů

Specializace diagramu tříd na DFD je realizována zavedením 4 standardních stereotypů:

- DataStore,
- Funkce,
- Terminátor,
- DataFlow.

První tři jsou stereotypy objektu třída, DataFlow je stereotypem asociace.

Pro takto stereotypizovaný model tříd pak platí pravidla konzistenčních vztahů v DFD:

- DataStore musí mít alespoň jeden vstupní DataFlow a jeden výstupní DataFlow.
- DataFlow smí spojovat pouze Funkci a Funkci, Funkci a DataStore nebo Terminátor a Funkci.
- DataFlow Terminátor -> Funkce musí mít přiřazenu událost
- Funkce musí mít alespoň jeden DataFlow

B.1.6 Ostatní – doplňkové diagramy UML a jejich použití

Jazyk UML obsahuje některé další, doposud nezmiňované nástroje, které jsou z hlediska této metodiky nástroji doplňkovými. Jejich existence není nutná (nepřinášejí žádnou informaci, kterou by nebyly schopny přinést diagramy základní), mohou však být vhodným způsobem pohledu na problematiku z jiného úhlu.

B.1.6.1 Use Case Diagram

Use case diagram zachycuje vazby mezi aktéry a funkcemi systému a mezi funkcemi navzájem. Aktéry mohou být uživatelé systému, uživatelské role, organizační jednotky ale i jiné informační systémy. Tito aktéři přistupují k funkcím informačního systému.

Tento diagram je diagramem doplňkovým, protože zobrazuje modelovaný systém z velmi podobného pohledu jako diagram funkcí. Funkce zachycené v use case diagramu by by měly odpovídat svým významem i názvem funkcím z diagramu funkcí.

Use case diagram můžeme být na rozdíl od funkčního diagramu použít k:

- modelování scénářů užití
- modelování přístupových práv
- modelování vzorů uživatelského rozhraní

Modelování scénářů pomocí use case diagramů je zvažováno i ve specifikaci UML. Vlastní UML ovšem nemá jiné prostředky pro modelování posloupnosti interakce různých uživatelů s funkcemi systém. Tato metodika naopak obsahuje modelování procesů pomocí procesního diagramu, který tuto oblast pokrývá daleko detailněji a tudíž není důvod scénáře modelovat i pomocí use case diagramů.

Modelování přístupových práv pomocí use case diagramů může být pro některé skupiny uživatelů těchto diagramů mnohem přehlednější než popis přístupových práv jiným způsobem (např. slovním popisem nebo poznámkami ve funkčním diagramu). Z tohoto důvodu má smysl tento diagram využít přinese-li to zlepšení komunikace a pomůže-li to k přesnější definici přístupových práv.

Modelování vzorů uživatelského rozhraní je zřejmě nejsilnějším argumentem pro používání use case diagramů. Toto modelování má však smysl pouze u některých architektur systému. Proto lze tento diagram začít používat na rozhraní fáze a analýzy a designu, kdy je určena cílová architektura systému.

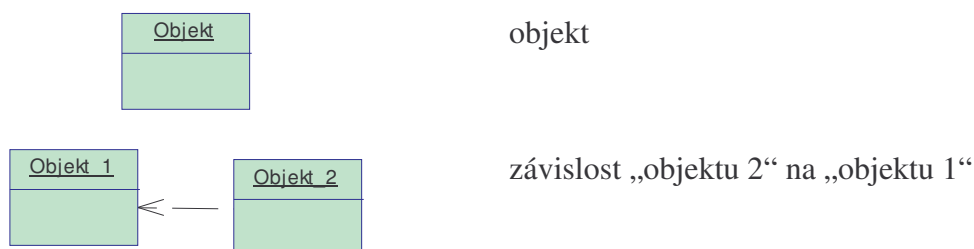
Modelováním vzorů uživatelského rozhraní rozumíme především určení, které funkce jsou speciálním případem funkcí jiných (vazba stereotypu <<extends>>) a které funkce se skládají z jiných funkcí systému (vazba stereotypu <<includes>>).

Při použití některých architektur lze use case diagramy vhodně využít i ve fázi designu k popisu komponent systému. Například při použití architektury model-view-controller pro tvorbu uživatelského rozhraní odpovídají jednotlivé use case komponentám typu <<controller>>, což může designérům pomoci ve vytvoření komponentového diagramu, návrhu jednotlivých komponent a určení jejich vazeb.

B.1.6.2 Diagram objektů (instancí)

Objektové diagramy jsou částí notace pro objektový design. Zachycuje objekty – instance tříd - a vztahy mezi nimi. Zachycuje tedy vlastně stav systému v určitém okamžiku. Zjednodušuje

pohled na diagram tříd a umožňuje také modelovat konkrétní případy kombinace instancí, což napomáhá plnému pochopení informace, vyjadřované modelem tříd.



B.1.6.3 Diagramy posloupností a spolupráce objektů

Diagramy posloupností a diagramy spolupráce objektů slouží k popisu vzájemných vazeb a součinnosti jednotlivých objektů systému. Tyto diagramy jsou v podstatě jakýmsi doplňkem diagramu tříd a diagramu stavů a slouží pro lepší pochopení vztahů business procesů na straně jedné a životních cyklů objektů na straně druhé.

Oba typy modelů jsou součástí objektově orientovaného modelu – OOM (Object-Oriented Model). Oba modely v podstatě zachycují stejné vazby, ale ze dvou různých pohledů. Hlavním rozdílem tedy je, že diagramy spolupráce objektů se soustředí na popis vzájemných vazeb spolupracujících objektů, na zobrazení struktury vzájemných vazeb a předávaných zpráv. Naproti tomu diagram posloupností zdůrazňuje hledisko časové posloupnosti v jaké dochází k interakci objektů.

Diagramy spolupráce objektů (Collaboration Diagrams)

Pomocí diagramů spolupráce objektů se zobrazují vzájemné vazby (instance link) aktérů (actor), objektů (object) – instancí jednotlivých tříd – a informace/zprávy/data (message) které si vzájemně vyměňují. Tímto diagramem popisujeme tyto elementy v konkrétní části resp. funkcionalitě systému, může například popisovat provedení určité operace.

Diagram spolupráce popisuje chování z pohledu interakce jednotlivých elementů, specifikuje chování tříd, rozhraní a možné využití operací. Doplňuje tím class diagram, kterýžto je víceméně statickým popisem struktury systému.

Při tvorbě tohoto diagramu dochází k ověřování jednak statického modelu – diagramu tříd – tak i procesního modelu. Analýza pomocí těchto diagramů pak může vést k dodatečnému doplnění jednotlivých modelů.

Tento typ diagramů je užitečný pro ujasnění významu a rolí jednotlivých objektů.

Diagramy posloupností (Sequence Diagrams)

Diagram posloupností (Sequence Diagram), jak již bylo řečeno výše, je svou podstatou příbuzný diagramu spolupráce a slouží k popisu vzájemných interakcí objektů a činností mezi nimi v chronologickém sledu. Klíčovým rozdílem od zmiňovaného diagramu spolupráce je právě časové hledisko, tj. diagram posloupností se snaží zachytit

Základní diagram posloupností lze automatizovaně vygenerovat z diagramu spolupráce objektů pomocí volby

Tools/Create Default Sequence Diagram

Takto ovšem dojde k vygenerování modelu neúplného, je nutné jej upravit a to zejména co se týká časové posloupnosti.

Obdobným způsobem lze vytvářet i základní diagram splupráce na základě diagramu posloupností, a sice pomocí volby

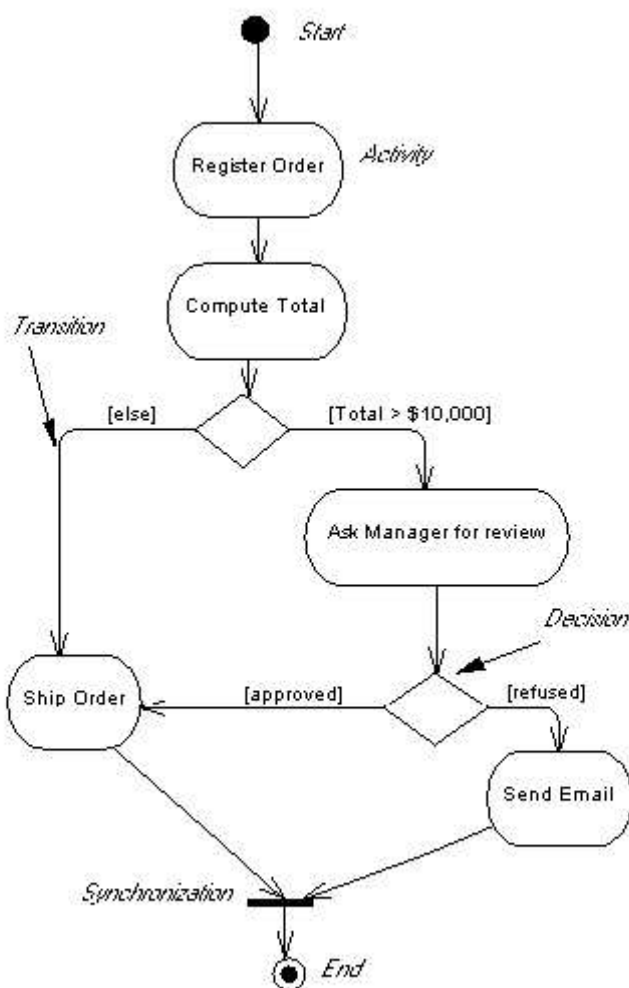
Tools/Create Default Collaboration Diagram

B.1.6.4 Diagram činností

Diagram činností (Activity diagram) je jedním z modelů UML a slouží k popisu chování řídicích toků (transition) mezi akcemi prováděnými v systému (activity).

Diagram činností je využíván pro popis jednotlivých procesů, tak jak jsou (či by měly být) odráženy systémem. Tedy v zásadě popisuje chování jednotlivých metod.

Zachycuje jednotlivé řídicí toky od jedné činnosti k druhé.



Obrázek B 11 - Příklad použití Activity Diagramu v nástroji Power Designer

B.2 Konstrukční modely

Architektonický design je klíčovou fází vývoje informačního systému. Jeho kvalita rozhoduje o úspěchu celého projektu. Při tvorbě designu systému vycházíme z analytických modelů, předem definovaných v globální a detailní analýze IS. Výsledkem architektonického designu je jasně specifikovaná architektura IS.

Architektura informačního systému je obvykle velice komplexní a můžeme se na ni dívat z různých hledisek (statické vs. dynamické aspekty systému; abstraktní vs. fyzická úroveň, popis systému vs. procesy v něm probíhající). Z této komplexnosti vyplývají dva základní, částečně se překrývající pohledy. Jejich kombinace nám umožní porozumět architektonickému designu. Těmito pohledy jsou komponentová a procesní architektura (viz Obrázek B 12).

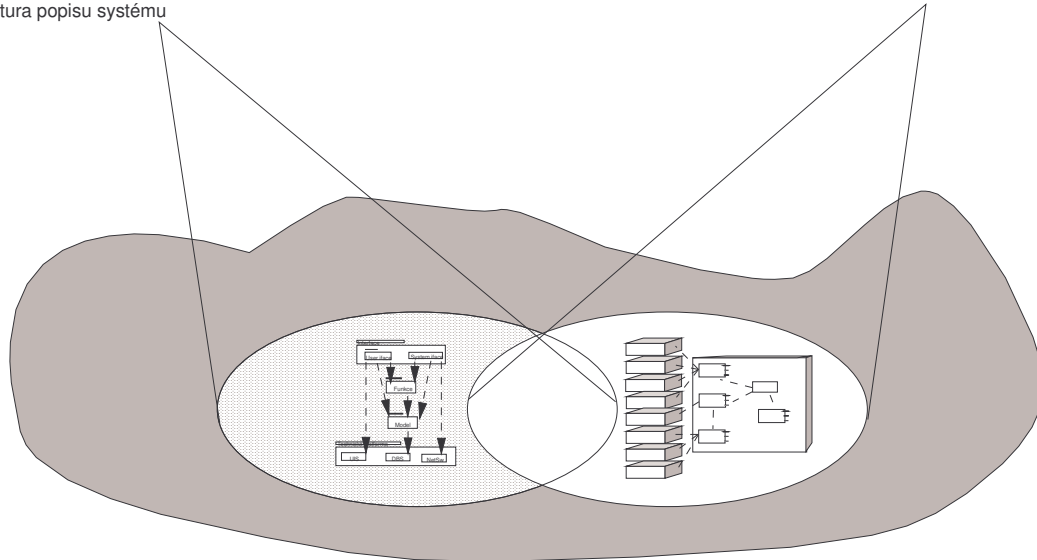
Komponentová architektura se zaměřuje na třídy, tedy statické aspekty systému. Rozděluje systém na jasně identifikovatelné, vzájemně propojené komponenty, řeší jejich vnitřní stavy a přechody mezi nimi. **Architektura procesů** se naproti tomu věnuje instancím tříd (dynamické aspekty systému). Dělí systém na procesy, řeší jejich vzájemné interakce a koordinaci.

Architektura komponent:

- třídy
- statické aspekty
- propojení komponent
- logický pohled
- struktura popisu systému

Architektura procesů:

- objekty
- dynamické aspekty
- koordinace procesů
- fyzický pohled
- struktura chování systému



Obrázek B 12 Komponentová versus procesní architektura

Oddělením komponentové a procesní architektury snižujeme komplexnost a zlepšujeme porozumění architektuře jako celku. Avšak hranice mezi oběma pohledy není ostrá, obě architektury se částečně prolínají. Zabývá-li se například komponentová architektura vazbou mezi objekty komunikujícími pomocí zpráv, zaměřuje se procesní architektura na interakci jako takovou, její skutečné vykonávání na fyzické úrovni systému.

Architektonický design je založen na třech základních principech. Prvním z nich je:

Princip 1: Definování všech potencionálních kritérií na architekturu systému

Výchozím bodem pro architektonický design je stanovení kritérií vycházejících z analýzy. Kritéria mohou a většinou jsou ve vzájemném rozporu (např. požadavek na rychlost odezvy oproti požadavku centrálního zpracování dat na vzdáleném serveru). Je proto potřeba stanovení priorit pro jednotlivá kritéria.

Druhý princip vyjadřuje obecný cíl architektonického designu:

Princip 2: Propojení kritérií s technickým prostředím

Architektonický design musí na jedné straně rozumět systémovému kontextu vyjádřenému stanovenými kritérii a na straně druhé prostředkům technické platformy.

A protože během architektonického designu musíme udělat velké množství zásadních rozhodnutí, která výsledný IS výrazně ovlivní, přidáme ještě třetí důležitý princip:

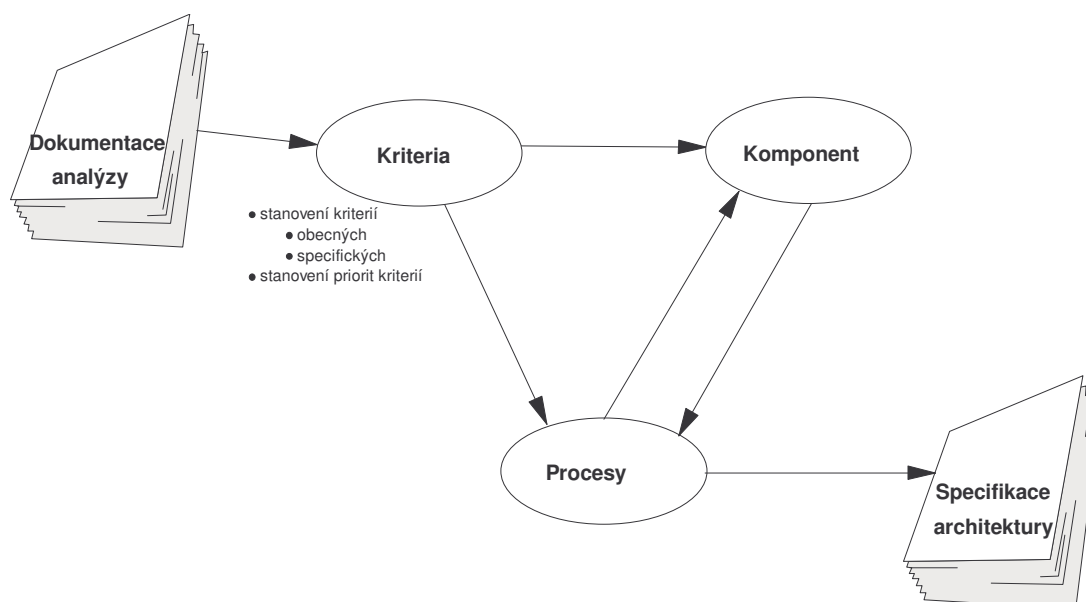
Princip 3: Brzy přehodnotit

Je potřeba včas zjistit, zda daný design splňuje stanovená kritéria. Pokud tomu tak není, je potřeba jej změnit. Jakákoliv architektonická přehodnocení učiněná pozdě znamenají obrovské dodatečné náklady a mohou celý projekt znehodnotit.

Během designu tedy:

- definujeme kritéria designu, stanovíme jejich priority,
- popíšeme všechny programové části a jejich vzájemné vazby
- definujeme chování IS a specifikujeme vzájemné vazby procesů a komponent.

Obrázek B 13 ilustruje jednotlivé kroky architektonického designu.



Obrázek B 13 Postup architektonického designu

V následující částech se budeme podrobněji věnovat jednotlivým fázím.

B.2.1 Kritéria designu

Kritéria designu, tedy seznam požadovaných vlastností architektury systému, jsou výchozím bodem každého designu. Neexistuje žádný obecný recept, jak navrhnout dobrý design libovolného informačního systému. Vždy je potřeba individuálně posuzovat konkrétní podmínky každého projektu. Stanovení kritérií nám v tom pomůže.

B.2.1.1 Obecná kritéria designu

Kvalitní architektonický design je hodnocen podle toho, jak splňuje požadavky výchozí analýzy. Systém by měl správně modelovat zadanou problémovou doménu a implementovat všechny požadavky na funkcionalitu. Podstatné však je, aby neobsahoval žádnou významnou závadu. Nedodržení jediného důležitého kritéria znamená neúspěch. Platí tedy:

Princip: *Dobry design nemá žádné kritické slabiny*

Během let se objevilo mnoho různých výzkumů zabývajících se obecnými kritérii architektonického designu. Následující tabulka nám představuje klasický seznam kritérií pro kvalitu softwaru. Vyjdeme-li z něho, může nám to pomoci při stanovení seznamu pro náš projekt.

Kritérium	Je měřítkem...
Použitelnost	přizpůsobitelnosti systému organizačnímu, provoznímu a technickému kontextu
Bezpečnost	zabezpečení vůči neautorizovanému přístupu k datům a zařízením
Efektivnost	schopnosti ekonomicky využít technickou platformu
Správnost	naplnění uživatelských požadavků
Spolehlivost	naplnění požadované přesnosti výkonu funkcí
Udržitelnost	nákladů na nalezení a opravu chyby
Testovatelnost	nákladů na ověření, že systém správně provádí své určené funkce
Flexibilita	nákladů na modifikaci
Srozumitelnost	úsilí potřebného k porozumění systému
Znovupoužitelnost	možnosti použití částí systému v jiných systémech
Přenositelnost	nákladů na přenos systému na jinou technickou platformu
Interoperabilita	nákladů na propojení systému s jinými systémy

Je zřejmé, že některá z těchto obecných kritérií jsou vůči sobě v rozporu. Záleží na konkrétním návrhu, která z nich považujeme za podstatná a která méně. Z toho vyplývá další důležitý princip:

Princip: *Dobry design vyvažuje více kritérií*

Objektově orientovaná analýza a design vyzdvihuje tři klíčová kritéria s shrnuje je do následujícího principu:

Princip: *Dobry design je použitelný, flexibilní a srozumitelný*

Dá se říci, že tato kritéria mají obecnou platnost. Podívejme se na ně podrobněji.

Použitelnost

Při hodnocení použitelnosti se díváme na design jako celek, bez ohledu na jeho vnitřní členění a sledujeme tyto dva pohledy:

- jak design uspokojuje uživatelské požadavky
- jak využívá technickou platformu – špatný design některé prvky platformy přetěžuje, zatímco jiné nevyužívá efektivně

Flexibilita

Při návrhu IS je velice obtížné spatřit jej v celé jeho komplexnosti současně s pochopením všech podstatných souvislostí. Je pravděpodobné, že některé požadavky byly opomenuty, jiné budou změněny, některá architektonická rozhodnutí je zase během designu potřeba odložit například z různých organizačních důvodů. Je tedy jasné, že při návrhu vždy disponujeme pouze nedostatečnými znalostmi potřebnými pro design.

Kvalita designu závisí na ceně pozdějších změn. Flexibilita je podstatně ovlivněna modularitou systému, možností vyměnit celé komponenty za jiné se stejným nebo podobným rozhraním.

Srozumitelnost

Jak již bylo vícekrát zmíněno, je obtížné během návrhu designu znát a rozumět současně všem částem jednotlivě i designu jako celku. Systém je potřeba proto budovat způsobem, který bude snadno pochopitelný a tím udržitelný. Při designu systému nám v tom mohou pomoci některé nástroje.

První z nich je *abstrakce*. Porozumíme-li některému konceptu, budeme snadno chápat všechny jeho použití v konkrétních případech v různých částech designu. Za další můžeme považovat užití *návrhových vzorů* (patterns) a to na různých úrovních systému od celých komponent až po jednotlivé třídy.

Srozumitelnost může být také zvýšena vyčleněním a spojováním podobné funkcionality do specializovaných komponent. Bude-li například mnoho částí systému přistupovat k nějakému zdroji technické platformy, můžeme tuto funkcionalitu vyčlenit do komponenty manažeru tohoto zdroje. Tím se systém stává srozumitelnější (a samozřejmě také flexibilnější).

B.2.1.2 Specifické podmínky designu

Kromě diskutovaných obecných kritérií je potřeba brát v úvahu také projektově specifické podmínky. Ty můžeme v zásadě rozdělit do třech kategorií:

- *Technické*: existující hardware, základní software i další existující systémy, použití existujících standardních komponent, zakoupení nových (např. novější verze pravděpodobně má větší životnost, ale je méně vyzkoušená), využití existujících návrhových vzorů, atd.
- *Organizační*: uzavřené kontrakty (např. rozdělení úloh mezi více nezávislých dodavatelů, aby žádný z nich neměl moc nad celým IS), plány dalšího rozvoje IS, atd.
- *Personální*: kompetence, zkušenosti z podobných projektů, znalosti technické platformy

B.2.1.3 Stanovení priorit

Na závěr úvodní fáze designu je potřeba přidělit jednotlivým kritériím jejich priority. Do celkového seznamu použijeme obecná kritéria stejně jako kritéria vzešlá z projektově specifických podmínek. Pro další vývoj IS je důležité kontrolovat, zda design uvedená kritéria splňuje.

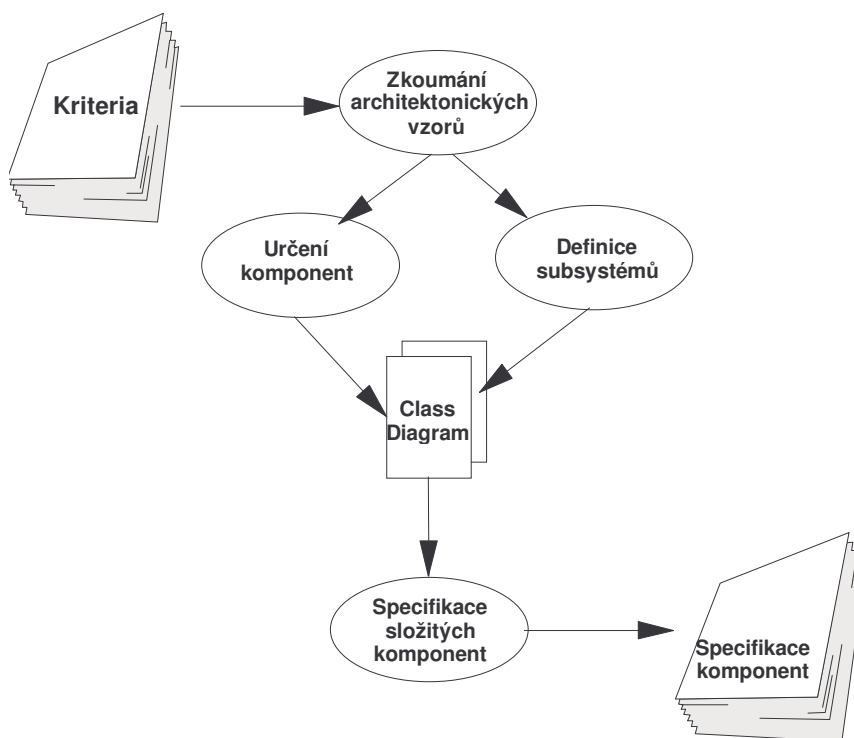
B.2.2 Diagram komponent

Komponentová architektura vychází z definovaných kritérií designu a modeluje programovou strukturu systému. Výsledkem fáze komponentové architektury je diagram tříd spolu se specifikací komplexních komponent.

Komponentou rozumíme souhrn programových částí, tvořících celek s definovanými odpovědnostmi.

Takto záměrně široce pojatá definice zahrnuje vše od jednotlivých tříd přes podsystémy až po systém jako celek (který pak chápeme také jako jednu komponentu). V této fázi se ale zaměřujeme zejména na střední úroveň, tedy na části celkového systému, kdy každá z nich zahrnuje obvykle více tříd, které spolu souvisí.

Postup specifikace komponent ilustruje Obrázek B 14.



Obrázek B 14 Postup specifikace komponent

Dříve než se budeme věnovat podrobněji jednotlivým krokům specifikace komponent, podívejme se na několik obecně platných principů. Hlavními přednostmi správně navržené komponentové architektury jsou flexibilita a srozumitelnost. I komplexní systém je tedy možné navrhnout, pokud budeme dodržovat:

Princip 1: *Redukce složitosti rozdělením na komponenty podle oblastí zájmu*

Návrh komponentové architektury je interaktivní proces, kterým se snažíme překlenout požadavky a kritéria s technickými možnostmi. Komponentová architektura musí poskytovat

flexibilitu, ale samozřejmě neumožní jakékoliv architektonické změny. Proto je důležitý následující princip:

Princip 2: *Uvažování stabilních kontextových struktur (stabilní aspekty reality a podmínek práce systému)*

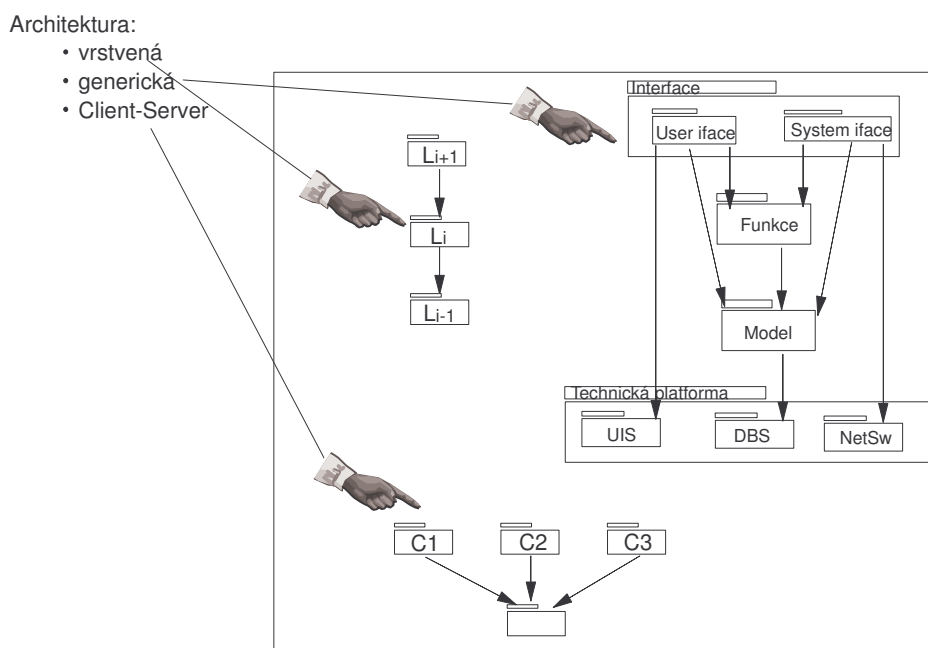
Problémová doména IS se nemění například tak často jako aplikační prostředí. Proto je vhodné rozdělit systém například na tři komponenty – model, funkce, uživatelské rozhraní (klasický návrhový vzor *MVC – model-view-controller*). Je pravděpodobné, že uživatelské rozhraní se bude měnit častěji, může jich například vzniknout více pro různé platformy (web, mobilní telefony, atd.) při zachování stabilních komponent model a funkce.

Při návrhu komponentové architektury nám mohou pomoci existující vzory, jejich využívání shrnuje:

Princip 3: *Použití stávajících komponent*

B.2.2.1 Zkoumání architektonických vzorů

Cílem této fáze designu je výběr architektury systému nejlépe odpovídající kritériím a následné určení komponent. K výběru vhodné architektury nabízí skandinávská metodika tři typové architektonické vzory (viz Obrázek B 15):



Obrázek B 15 Typové architektonické vzory

Vrstvená architektura je v softwaru jedním z nejpoužívanějších vzorů. Jednotlivé vrstvy představují komponenty, které jsou vzájemně propojeny ve směru nahoru a dolů. Klasickým příkladem bývají síťové architektury (ISO-OSI nebo TCP/IP). Každá vrstva má jasně definované rozhraní, které mohou ostatní vrstvy využívat.

Každou vrstvu je samozřejmě dále možné dekomponovat na menší části, čímž nám v jednotlivých vrstvách vzniknou sub-komponenty. Vazby pak nemusí být specifikované mezi celými vrstvami ale pouze mezi těmito sub-komponentami.

Architektonický vzor vrstvené architektury můžeme vidět v několika variacích, podle toho, jaké vazby mezi sebou jednotlivé vrstvy mají. Rozlišujeme tak vrstvené architektury:

- *Uzavřené* – kdy každá vrstva (i) používá pouze vrstvy bezprostředně sousedící ($i+1$ a $i-1$)
- *Otevřené* – každá vrstva může využívat všech vrstev, nejen těch, které přímo sousedí

Vrstvené architektury můžeme dále dělit na

- *Striktní* – vrstva smí využívat a volat pouze vrstvy položené níže
- *Volné* – vrstva může využívat jiné vrstvy v obou směrech – nahoru i dolů

Zkombinováním uvedených dělení mohou vzniknout čtyři typy vrstvených архитектур – od nejvíce omezené uzavřené-striktní až po druhý extrém otevřené-volné, kdy prvek vrstev již v podstatě mizí.

Klient-server architektura

Klient-server architektura je založena na principu jednoho poskytovatele služby a jednoho nebo více uživatelů této služby. Tato architektura je asymetrická v tom smyslu, že server obecně nemá žádné informace o klientech, zatímco klienti musí znát server a služby, které poskytuje. Klienti o sobě rovněž vzájemně neví a přistupují k serveru nezávisle na ostatních.

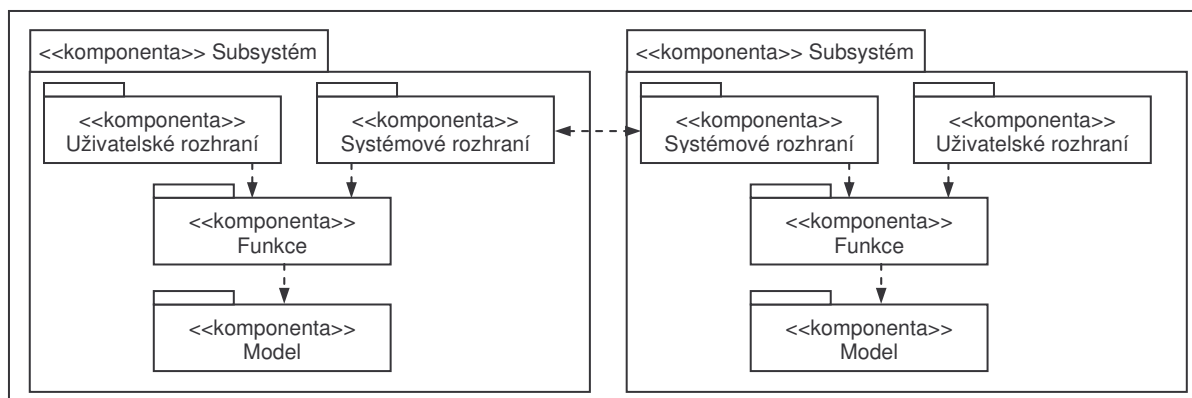
Generická architektura

Na předchozím obrázku (Obrázek B 15) vidíme také příklad generické architektury. Je poměrně vhodné vyčlenit části systému, které jsou specifické pro technickou platformu a vytvořit pokud možno obecnější rozhraní, které budou moci další komponenty využívat. Na příkladu je vidět takové vyčlenění komponenty (knihovny) UIS pro uživatelská rozhraní, komponenty DBS pro databázově orientované části systému a komponentu NetSW pro části spolupracující se sítí. Zbývající část architektury navržené v tomto příkladu je již klasický model-view-controller.

B.2.2.2 Definice subsystémů

U jednodušších systémů není problém využít pouze standardní architekturu MVC a celý systém se tak bude skládat pouze ze třech komponent: model, funkce a uživatelské rozhraní. Designovat tímto způsobem komplexnější systémy by bylo nevhodné. Ty je potřeba rozdělit na více částí (subsystémů), které spolu budou komunikovat pomocí jasně definovaných rozhraní. Každý subsystém se tak bude skládat dále z vlastních sub-komponent.

Jak je vidět na následujícím obrázku, každý subsystém obsahuje navíc komponentu Systémové rozhraní (system interface). Na rozdíl od komponenty uživatelského rozhraní, která umožňuje interakci s uživatelem, stará se systémové rozhraní o zpřístupnění funkcí jiným subsystémům.



Obrázek B 16 Příklad architektury subsystémů

Architektura uvnitř každého subsystému na předchozím obrázku je uzavřená a striktní. Každá vrstva využívá pouze vrstvu ležící bezprostředně pod ní. Subsystémy naopak mají mezi sebou symetrické závislosti prostřednictvím komponent systémových rozhraní. Systémové rozhraní je zodpovědné za monitorování sub-komponenty pod sebou a pokud je potřeba, vyřizuje požadavky u jiných subsystémů. Bývá často výhodné změnit závislost systémového rozhraní na sub-komponentě funkce na závislost symetrickou. Sub-komponenta funkce pak může volat přímo systémové rozhraní a jeho prostřednictvím využívat jiných subsystémů. Není příliš vhodné, aby funkce využívaly přímo systémová rozhraní jiných subsystémů. To by přidávalo další závislosti a systém by se stal příliš provázaný a těžko udržitelný.

V případě distribuované aplikace uvažujeme obvykle klient-server architekturu. Na klientskou a serverovou stranu se můžeme dívat jako na dva nezávislé subsystémy, každý s vlastními sub-komponentami (model, funkce, uživatelské rozhraní). Druhou možností je rozložit tyto sub-komponenty mezi klientskou a serverovou stranu. V tomto případě se nám nabízí několik variant a je potřeba rozhodnout, která z nich je pro naše účely nejvhodnější:

Klient	Server	Architektura
U	U + F + M	Distribuovaná prezentace
U	F + M	Lokální prezentace
U + F	F + M	Distribuovaná funkcionalita
U + F	M	Centralizovaná data
U + F + M	M	Distribuovaná data

B.2.2.3 Určení komponent

Při definici systému nebo subsystémů obvykle začínáme u standardního rozdělení na komponenty model, funkce a rozhraní. V případě komplexních návrhů je potřeba ale dále postupovat s dekompozicí jednotlivých komponent.

Dekompozice modelu je obvykle nejsložitější, protože bývá obtížné nalézt jasné hranice v modelované problémové doméně. Pokud jsou některé části méně provázané a nebo jsou jejich vazby jednodušší, oddělíme tyto části do samostatné komponenty. Často je také vhodné rozdělit modelovou komponentu na její databázovou část a část skutečného modelu.

Rozdělení komponenty funkcí je možné například s využitím vrstvené architektury.

Nejspodnější komponenta funkcí poskytuje pouze primitivní operace pracující přímo s modelem. Vrstvy položené výše mohou tyto operace využívat a sestavovat nich komplexnější funkce. Nejvyšší vrstva pak bude poskytovat funkce, které jsou vyžadovány komponentami rozhraní (uživatelského případně systémového).

V neposlední řadě je potřeba dekomponovat uživatelské rozhraní. Zejména v poslední době roste potřeba podpory různých zařízení – systém musí poskytovat rozhraní pro uživatele prostřednictvím PC (webový prohlížeč nebo aplikační klient), mobilních telefonů, PDA a podobně. Poměrně vhodné je rozdělení na obecnější vrstvu uživatelského rozhraní, která poskytuje popisy obrazovek, ale nestará se o vlastní zobrazování. Tato vrstva musí směrem nahoru poskytovat jednoduché rozhraní (protokol), které je pak možné interpretovat nejvyšší vrstvou aplikace na různých platformách.

B.2.3 Diagram procesů (rozmístění)

Návrh architektury procesů je po specifikaci kritérií a návrhu komponentové architektury poslední fází designu. Jejím cílem je rozmístění komponent navržených v komponentové architektuře na jednotlivé fyzické části systému (procesory). Zatímco v komponentové architektuře pracujeme s třídami a komponentami, procesní architektura se věnuje instancím a jejich interakcím.

Procesní architektura je tedy struktura nezávislých vzájemně spolupracujících procesů, která popisuje běh systému.

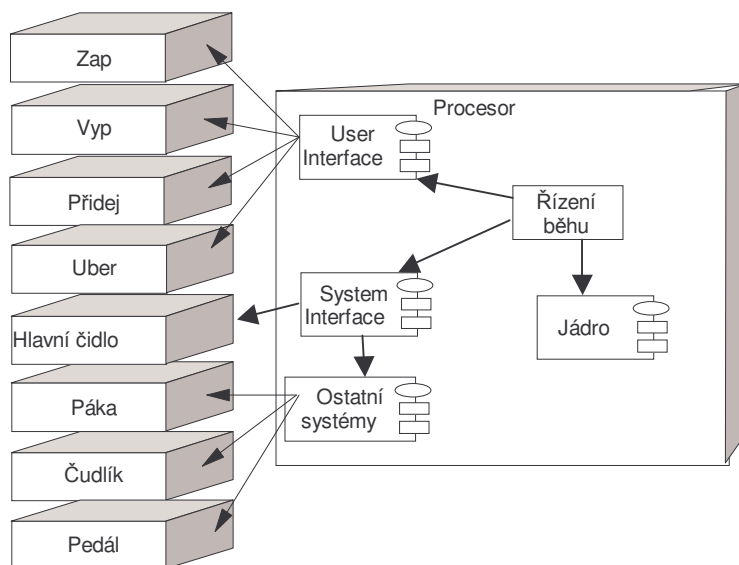
Procesor je zařízení schopné vykonávat program.

Aktivní objekt je objekt, který byl přiřazen procesu.

Programová komponenta je fyzický modul programového kódu.

Na procesní architekturu se díváme ve dvou rovinách abstrakce: Za prvé na globální úrovni, kdy rozmisťujeme jednotlivé komponenty na dostupné procesory systému. A za druhé na úrovni procesů, kdy sledujeme kooperaci komponent a řešíme synchronizaci procesů. V případě, kdy designujeme samostatnou aplikaci, která nemá příliš interakcí se svým okolím, bývá procesní architektura poměrně snadná. Situace se ale komplikuje, pokud tvoříme např. komplexnější monitorovací systém, systémy často komunikujícími s okolím nebo když náš systém pracuje s časově náročnými úkoly a mnoho z nich se řeší na pozadí hlavního běhu aplikace (tedy v jiných vláknech či procesech) a je potřebná jejich synchronizace.

Příklad výstupu procesní architektury vidíme na následujícím obrázku.



Obrázek B 17 Diagram rozmístění (Deployment Diagram)

Procesní architektura vychází ze třech základních principů:

Princip 1: Rozmístění komponent na procesory

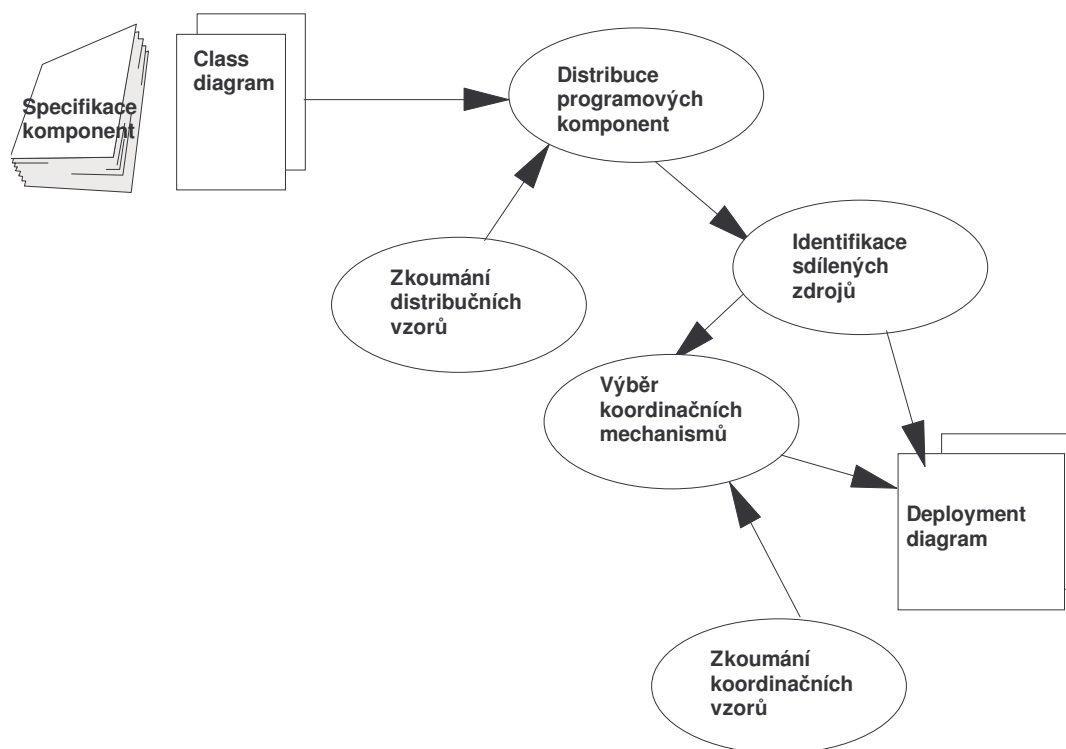
Procesní architektura musí navrhnout systém tak, aby vytěžoval rovnoměrně všechny dostupné procesory. To je vyjádřeno dalším principem:

Princip 2: Vytvoření architektury bez úzkých míst

Procesy vyžadují během svého vykonávání přístup k různým zdrojům a objektům (respektive různé objekty jsou k nim přiřazovány jako *aktivní*). Často samozřejmě dochází ke kolizi požadavků (např. pobočkové systémy banky potřebují přistupovat k některému sdílenému systému). Procesy je proto potřeba koordinovat, což vyjadřuje:

Princip 3: Koordinace sdílení zdrojů s aktivními objekty

Během tvorby procesní architektury postupujeme tak, jak je vyjádřeno na následujícím obrázku.



Obrázek B 18 Postup specifikace procesní architektury systému

Podívejme se na jednotlivé kroky specifikace procesní architektury podrobně.

B.2.3.1 *Distribuce programových komponent*

Procesní architektura obvykle následuje poté, co jsou již poměrně přesně definované komponenty a třídy z fáze komponentové architektury. Kromě toho, je možný i jiný postup, jak k designu celkově přistupovat - tvořit procesní i komponentovou architekturu víceméně souběžně. Tento přístup umožňuje, aby procesní architektura zpětně ovlivňovala tvorbu komponent. Celý design pak probíhá ve více iteracích, kterými se postupně přibližujeme celkovému řešení. Je na každém tvůrci systému, jaký z přístupů zvolí. V každém případě je ale důležité udržet obě architektury – komponentovou a procesní – v konzistenci.

Prvním krokem při distribuci komponent na fyzické procesory je separace programových komponent, které neobsahují žádné aktivní objekty, a samotných aktivních objektů. Bude-li například komponenta obsahovat dva aktivní objekty – čtení ze sítě a zápis do souboru, je potřeba oba tyto objekty vyčlenit a z hlediska procesní architektury s nimi pracovat jako se dvěma oddělenými procesy.

V dalším kroku je potřeba identifikovat dostupné procesory, které využijeme k vykonávání systému.

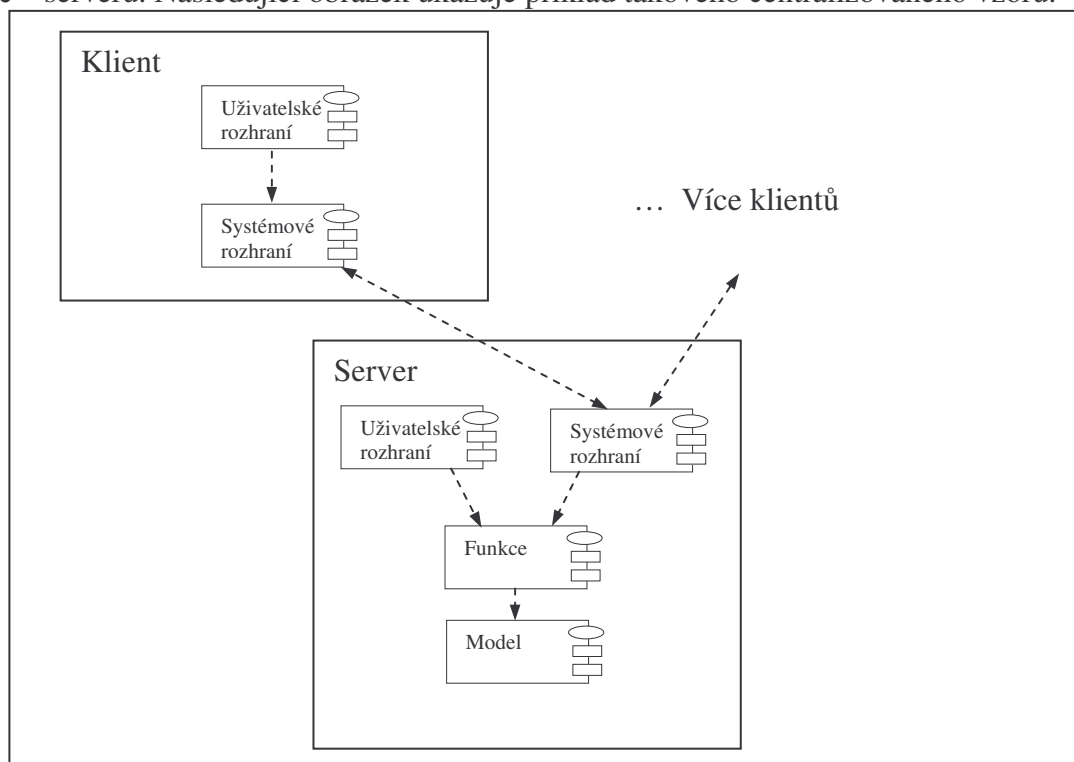
Třetím krokem je vlastní rozmístění programových komponent obsahujících pouze pasivní objekty a rozmístění aktivních objektů na dané procesory. V závislosti na architektuře (vrstvená, klient-server) umísťujeme programové komponenty na jeden nebo více procesorů. Pro systém navržený podle vrstvené architektury není problém umístit vše na jeden procesor. V případě umístění architektury klient-server na jediný procesor je potřeb mít na zřeteli případný požadavek paralelního běhu více klientů a řešit tento požadavek prostředky, které poskytuje operační systém na daném procesoru. Obvyklejší situace ale je varianta oddělení serverových a klientských programových komponent a jejich umístění na různé procesory.

Při rozmístění se nám hodí využít jeden ze tří distribučních vzorů – centralizovaný, distribuovaný a decentralizovaný. Podívejme se na ně podrobně.

B.2.3.2 Zkoumání distribučních vzorů

Centralizovaný vzor

První variantou je distribuovat programové komponenty na procesory co nejméně je to možné. Jinými slovy udržet (v ideálním případě) veškerá data a funkcionalitu na jednom místě – serveru. Následující obrázek ukazuje příklad takového centralizovaného vzoru.

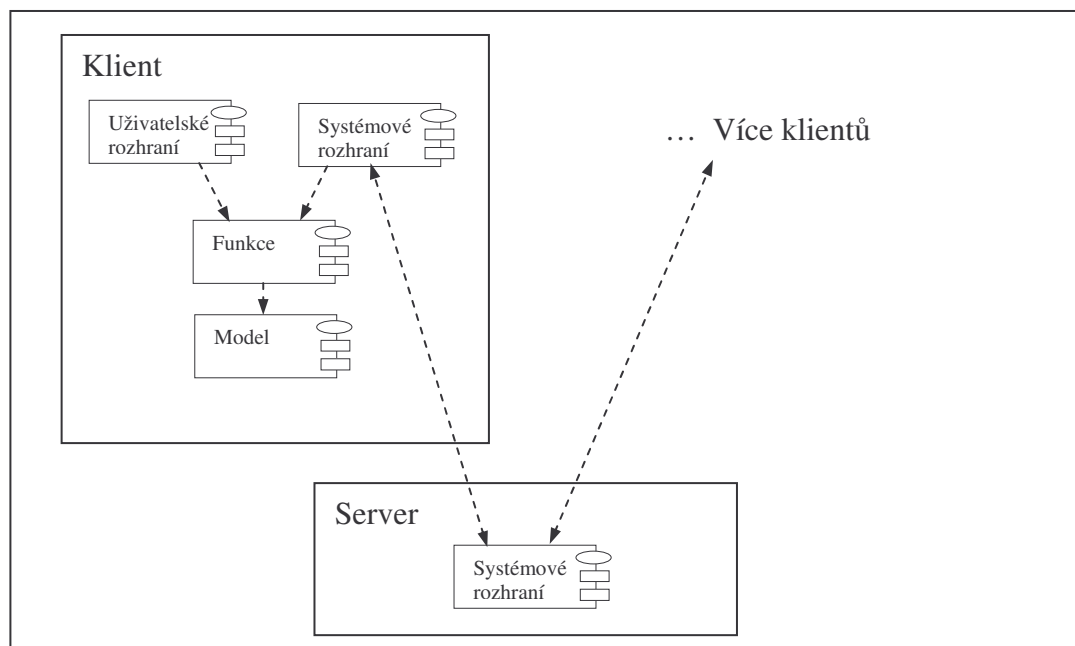


Obrázek B 19 Centralizovaný vzor

Na klientské procesory umístíme pouze uživatelské rozhraní, takže se jedná v rámci systému podstatě o jednoduché terminály k serveru.

Distribovaný vzor

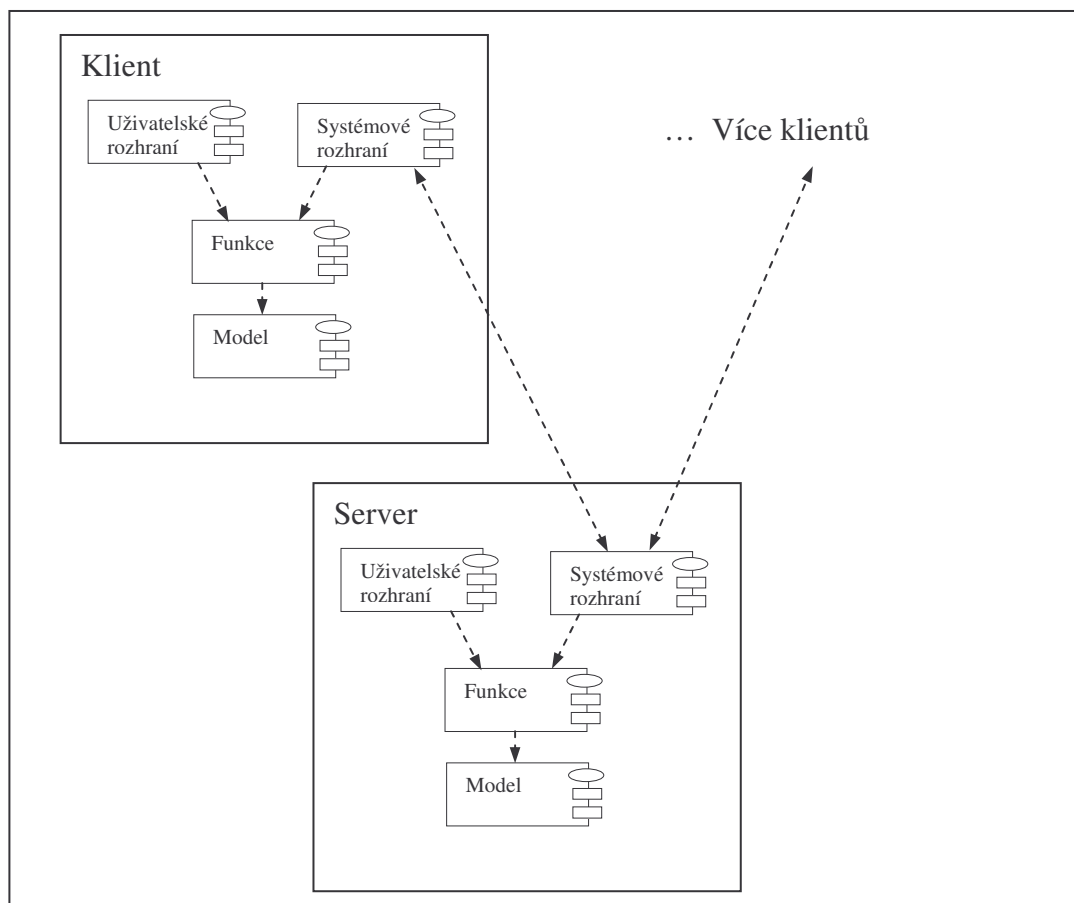
Distribuívaný vzor predstavuje presný opak predchádzajúceho vzoru. Model (tedy všetky dáta) sú umiestnené na každom klientskom procesore. Server zaisťuje iba komunikáciu medzi klientmi (správnu replikáciu dát a jej udržiavanie v konzistentnom stave). Nasledujúci obrázok ukazuje tento vzor.



Obrázek B 20 Distribuovaný vzor

Decentralizovaný vzor

Určitou kombinací předchozích dvou je vzor třetí – decentralizovaný. Vzor je založený na rozdělení dat a tím i funkcí do dvou částí – část vlastní každému jednotlivému klientu a část sdílená, která je spravovaná serverem. Tento vzor (viz následující obrázek) je využitelný zejména v případech, kdy problémová doména je snadno rozdělitelná na několik podobných sub-domén, kdy každá z nich se stává problémovou doménou pro klientskou stranu.



Obrázek B 21 Decentralizovaný vzor

Výhody a nevýhody třech uvedených vzorů shrnuje následující tabulka:

<i>Vzor</i>	<i>Výhody</i>	<i>Nevýhody</i>
Centralizovaný	<p>Klientské komponenty jsou nenáročné na hardware</p> <p>Veškerá data konzistentní, protože jsou na jednom místě</p> <p>Systém je relativně snadný na pochopení a implementaci</p> <p>Síťový provoz je přiměřený</p>	<p>Nízká robustnost celkového systému. V případě výpadku serveru nebo sítě nefunguje ani žádný klient</p> <p>Data nejsou zreprodukována a tudíž je větší riziko jejich ztráty (tento problém se ale samozřejmě dá vyřešit snadno zálohováním)</p>
Distribuovaný	<p>Rychlé odezvy (veškerá data jsou na klientské straně)</p> <p>Klienti mohou pracovat i při výpadku sítě nebo serveru</p> <p>Data jsou zreprodukována, takže riziko jejich ztráty</p>	<p>Množství redundantních dat</p> <p>Riziko nekonzistence dat</p> <p>Vysoký provoz na síti (replikace a synchronizace)</p> <p>Vysoké technické požadavky na klientský hardware</p> <p>Komplexní architektura, složitá na porozumění a</p>

	je menší	implementaci
Decentralizovaný	Data jsou konzistentní, protože klientská data není potřeba replikovat a sdílená data jsou umístěna pouze jednou – na serveru Síťový provoz by měl být relativně malý Rychlé odezvy v případě lokálních dat	Náročnost na hardware klientských procesorů Klientská data nejsou zreprodukována, větší riziko ztráty, komplikovanější zálohování klientských dat

B.2.3.3 Identifikace sdílených zdrojů

Jakmile jsou komponenty systému rozmístěny na dané procesory, stojí před námi poslední velký úkol – nalezení sdílených zdrojů a vyřešení synchronizace.

Předmětem našeho zájmu budou v této fázi tři druhy zdrojů, jejichž využití musíme koordinovat a vyřešit:

- Procesory (současný běh více procesů na jednom procesoru)
- Programové komponenty (využití jedné komponenty více procesy současně)
- Externí zařízení (např. současné sdílení tiskárny nebo různých vstupních měřících zařízení)

K identifikaci sdílených zdrojů potřebujeme poměrně dobře znát komponenty a operace nad nimi. Je těžké projít touto fází designu procesů před tím, než budou komponenty přesně definované. Nicméně na druhou stranu je nevhodné čekat s návrhem procesní architektury tak dlouho. Vhodným způsobem se zdá postupovat současně a ve více iteracích se přibližovat celkovému designu.

B.2.3.4 Výběr koordinačních mechanismů, jejich vzory

Výběr mechanismů pro zajištění správné funkce systému při sdílených zdrojích závisí velmi na programovacím jazyku zvoleném pro implementaci a na cílové platformě. Díky tomu je těžké říci nějaký všeobecně platný návod, jak v této fázi postupovat.

Obecně můžeme koordinovat procesy sdílející nějaký zdroj buď pomocí synchronizace procesů (pozastavení, jeden proces čeká na druhý) nebo pomocí výměny dat (kdy data jsou předávána z jednoho procesu do druhého bez potřeby nějaké synchronizace).

V závislosti na platformě a jazyku bývají k dispozici některé z následujících vzorů:

Vyhrazený (dedikovaný) monitor zabezpečuje vstup do sdíleného zdroje pouze jednomu volajícímu procesu. Pokud jiný proces má zájem použít zdroj ve stejné chvíli, je zastaven až do okamžiku, kdy první proces zdroj neuvolní. Pro pozastavený proces je toto volání zcela transparentní a nemusí proto situaci sám nijak řešit. Určitým rizikem tohoto návrhu je to, že při nevhodném návrhu dochází k velmi častým přerušením běhu procesů.

Centralizovaný dispečer úloh je založen na principu vykonávání všech úloh v (pod-)systému jedním aktivním objektem. Dispečer úloh je oblíbeným vzorem v několika současných platformách.

Registrace na změnu stavů je dalším z možných mechanismů. Místo aby objekt aktivně testoval např. změny externích čidel v nějakých pravidelných intervalech, postačí zaregistrovat se na změnu a komponenta je vyvolána v případě, že nastane daná událost (změna hodnoty na čidle).

Asynchronní výměna dat předpokládá existenci datových struktur typu zásobník (fronta) s atomickými operacemi na dané technické platformě. Na předchozím příkladu by například jeden proces do fronty přidával nové a nové naměřené údaje. Druhý proces by čas od času přečetl poslední hodnotu, ale současně vyčistil všechna zastaralá měření, která už nejsou v tu chvíli užitečná.

Díl C Realizace modelů v nástroji Power Designer

Tento díl publikace se zabývá možnostmi realizace soustavy modelů metodiky, popisované v díle B za dodržování principů popsanych v díle A v prostředí nástroje PowerDesigner.

C.1 Odlišné používání pojmů „model“ a „diagram“

V tomto dílu jsou pojmy model a diagram chápány jinak než v dílech A, D a především B. V díle B se modelem rozumí zjednodušený popis reality, který je graficky reprezentován příslušným grafickým diagramem. Textové tebulky požadavků nejsou v díle B chápány jako diagram. V některých případech mohou být pojmy model a diagram chápány jako synonyma. V tomto díle zaměřeném na popis realizace metodiky v nástroji PowerDesigner je význam těchto pojmů jiný - stejný jako v uživatelské dokumentaci tohoto nástroje. V nástroji Sybase PowerDesigner verze 12 je možné vytvářet 6 typů modelů. Každý z těchto modelů může obsahovat jeden i více typů diagramů. Diagramem se rozumí buď klasický grafický diagram různých notací nebo i různé textové tabulky a tabulky provázání (především u modelu požadavků - Requirements Model). Model je tudíž chápán jako agregace několika diagramů stejného či různého typu. Ukládání do souborů se provádí na úrovni modelů; na úrovni modelů jsou organizovány i metamodely a jejich rozšiřování. Různé prodejné verze a instalace nástroje se také mohou lišit počtem typů podporovaných modelů. Z pohledu PowerDesigneru je tedy pojem model používán spíše z technologického hlediska k sdružování různých diagramů do samostatných celků. Příkladem takového celku je Objektově orientovaný model (Object Oriented Model - OOM), který se skládá z 10 typů diagramů tak jak je definuje standard UML.

C.2 Modely PowerDesigneru nezbytné pro realizaci principů metodiky

C.2.1 Object Oriented Model (OOM)

Podpora objektového modelování v PowerDesigneru je založena na jazyku Unified Modeling Language (UML). Jsou podporovány všechny standardní diagramy UML:

název v metodice	název v PowerDesigneru	sekce popisující metodické využití
Use case diagram	Use Case Diagram	B.1.6.1
Diagram tříd	Class Diagram	A.4, B.1.1
	Composite Structure Diagram	
Diagram instancí	Object Diagram	B.1.6.2
Diagram spolupráce objektů	Collaboration Diagram	B.1.6.3
Diagram posloupností	Sequence Diagram	B.1.6.3
Diagram stavů	Statechart Diagram	B.1.3
Diagram činností	Activity Diagram	B.1.6.4
Diagram komponent	Component Diagram	B.2.1
Diagram rozmístění	Deployment Diagram	B.2.2

Všechny tyto diagramy jsou sdruženy v Objektově orientovaném modelu (Object Oriented Diagram - OOM). Objektový model je díky množství v něm obsažených diagramů a díky vazbě na generátory diagramem, který se používá jak v analýze tak v designu a implementaci systému a z tohoto pohledu je důležitým provázáním těchto fází. PowerDesigner umožňuje používat v diagramech různé verze notace pro analýzu (zjednodušené modely) i pro design ve specifických programovacích jazycích se stereotypy typickými pro dané jazyky. Je tudíž možné používat jeden analytický OOM pro analýzu a ve fázi designu z něj vygenerovat OOM

s notací pro konkrétní programovací jazyk, který je po upřesnění možné použít ke generování kódu aplikace. Standardně je podporováno několik programovacích jazyků: C#, C++, Java, PowerBuilder a další, viz dokumentace PowerDesigneru.

Dále je zde možnost synchronizace OOM s datovými modely (konceptuálními i fyzickými) a tak udržovat konzistenci mezi datovou a aplikační stránkou vyvíjeného systému.

C.2.2 Business Process Model (BPM)

Business procesy jsou v prostředí PowerDesigneru modelovány pomocí Modelu business procesů (Business Process Model - BPM). Tento model může obsahovat diagramy dvou typů: procesní diagram (Business Process Diagram) a diagram hierarchie procesů (Process Hierarchy Diagram). V sekcích A.4 a B.1.2 jsou popsána metodická východiska pro popis procesů pomocí diagramu procesů. Diagram hierarchie procesů není v metodice zmiňován, ale jeho využití může zlepšit orientaci v organizaci procesů pomocí základního principu metodiky - hierarchickou abstrakcí.

BPM může být podobně jako OOM použit v kombinaci s některou z doplňujících notací. Na výběr je zjednodušená analytická notace, notace podle standardu BPMN 1.0, notace BPEL4WS a další. Pokud má být součástí implementace výsledného systému Business Process Management System, lze použít analytický popis procesů ke generování modelů pro fázi designu a implementace v notaci BPEL4WS a následnou generaci BPEL implementace.

C.2.3 Data Flow Diagram

Data Flow Diagram (DFD), tak jak je popsán v sekci B.1.5, není standardní součástí PowerDesigneru. Pro potřeby této metodiky byl vytvořen upravením diagramu tříd. Diagram tříd byl zvolen jako základ tohoto modelu, protože modeluje existenci reality a DFD modeluje existenci funkcí a datových úložišť informačního systému. Notace diagramu je však velmi odlišná od diagramu tříd. Některé prvky diagramu mají nový význam (třída -> funkce, data store, terminátor; asociace -> data flow) jiné nebyly vůbec použity.

Definice tohoto modelu se nachází na přiloženém CD a lze ji doinstalovat do PowerDesigneru jako Extended Model Definition.

C.2.4 Model požadavků (Requirements Model - RM)

PowerDesigner nabízí podporu pro práci s požadavky, tak jak je vyžadováno sekcí A.6, v podobě **Requirements Modelu (RM)**. Tento model umožňuje evidenci požadavků v hierarchické struktuře a jejich export do formátu Microsoft Word. Ke každému požadavku je možné zadat jeho jméno, kód, typ (standardně: funkční, technický, designový), stereotyp a vazby na libovolný objekt v jiném modelu.

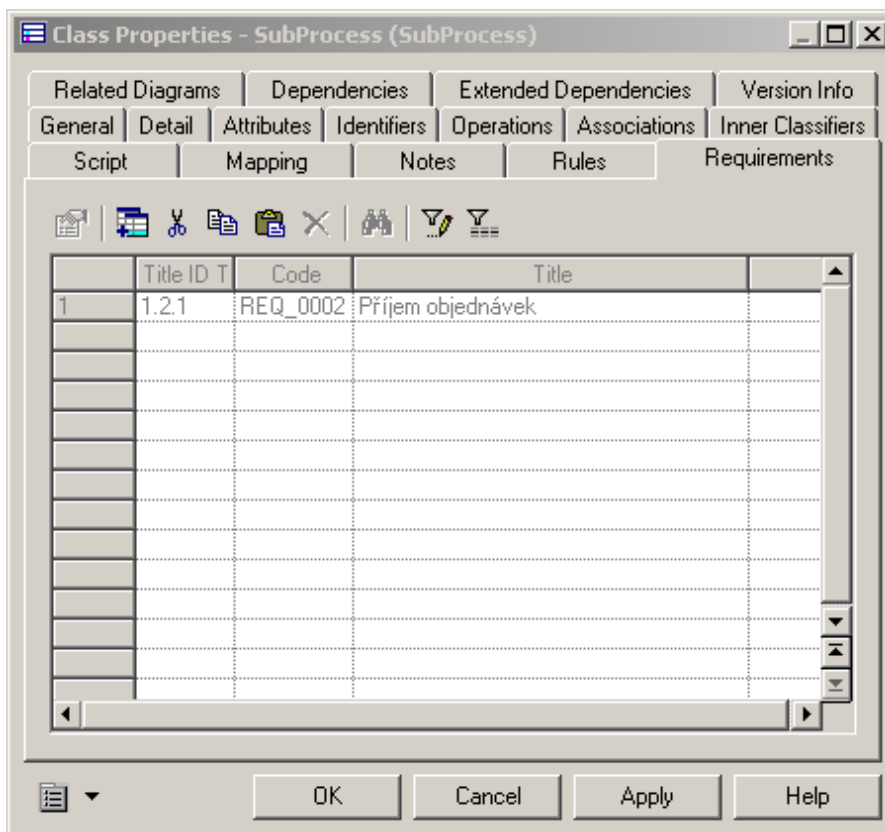
RM obsahuje tři typy diagramů:

- a/ **Dokument správy požadavků (Requirements document view)** – k zobrazení všech požadavků, které jsou zaznamenány během vývoje projektu
- b/ **Trace ability matrix view** – k propojení požadavků s dalšími typy modelů, externími soubory nebo dalšími požadavky.
- c/ **User allocations matrix views** – k propojení požadavků s uživateli a skupinami, ke kterým se tyto požadavky vztahují

	Title ID	Full Description	Code	Stereotype	Type
1	1.	Funkční požadavky	REQ_0001		Functional
2	1.1	Evidence zboží	REQ_0006		Functional
3	1.1.1	Výdej zboží Systém musí sledovat všechny výdeje zboží ze skladu	REQ_0003		Functional
4	1.2	Evidence objednávek	REQ_0005		Functional
5	1.2.1	Příjem objednávek Systém musí evidovat všechny příchozí objednávky	REQ_0002		Functional
6	1.2.2	Likvidace Objednávek Splněné objednávky systém vyřadí z evidence.	REQ_0004		Functional
7	2.	Technické požadavky	REQ_0007		Technical
8	2.1	Tenký klient Přístup k aplikaci musí být možný z internetového prohlížeče	REQ_0008		Technical
9	2.1.1	Firefox Přístup k aplikaci musí být možný z internetového prohlížeče Firefox verze 1.0	REQ_0009		Technical
10	3.	Designové požadavky	REQ_0010		Design
	3.1	Přístupnost Aplikace musí umožňovat přístup i barvoslepým zaměstnancům	REQ_0011		Design

Obrázek C 1 Příklad Requirements Modelu

Pro jednodušší vkládání vazeb objektů jiných modelů na požadavky je dobré v každém modelu v menu Tools>>Model Options zaškrtnout 'Enable links to requirements'. Poté je součástí properties každého objektu i záložka Requirements, kde lze nastavit vazby na požadavky.



Obrázek C 2 Požadavky jako vlastnosti objektu repository

Vazby požadavků je možné sledovat přes Traceability Matrix View – tabulku, která přehledně zobrazuje vazby mezi požadavky a objekty:

	Přijem Objednávek	Výdej zboží ze skladu
1. Funkční požadavky		
1.1 Evidence zboží		
1.1.1 Výdej zboží		✓
1.2 Evidence objednávek		
1.2.1 Přijem objednávek	✓	
1.2.2 Likvidace Objednávek		

Obrázek C 3 Traceability Matrix View

C.2.5 Konceptuální datový model (Conceptual Data Model - CDM)

Tento model popisuje logické vztahy mezi datovými entitami aplikace bez určení technologického prostředí pro jejich uložení a správu. Jeho význam je především ve fázi designu IS, kdy je třeba podrobněji popsat jakým způsobem budou objekty reality reprezentovány daty. PowerDesigner umožňuje tento model vygenerovat z objektového modelu - z diagramů tříd.

C.2.6 Fyzický datový model (Physical Data Model - PDM)

Tento model slouží k návrhu fyzické struktury databáze po zvolení konkrétního databázového systému. Lze ho vygenerovat z fyzického datového modelu. Naopak z tohoto modelu lze generovat implementaci v jazyce SQL v dialektu zvoleného databázového systému. PowerDesigner podporuje většinu významných databázových systémů.

C.2.7 Information Liquidity Model (ILM)

Tento model je nejspecifičtější z datových modelů. Umožňuje modelovat replikaci dat v relačních databázových systémech. Lze ho využít při provozu IS, který je třeba replikovat.

C.2.8 XML model

XML model je nástroj pro vizualizaci struktury XML dokumentů. Pokud implementace IS obsahuje ukládání dat nebo výměnu zpráv ve formátu XML, lze ve fázi designu použít XML model k definici struktury XML. Tento model lze vygenerovat z objektového modelu a lze z něj dále generovat implementaci v jazyce DFD nebo XML Schema (XSD). Multi-Model Report (MMR)

Multi-Model Report je speciální typ modelu, který umožňuje definovat obsah a formu dokumentace k jednomu a více modelům vytvořeným v PowerDesigneru. Výstupy lze tisknout, nebo generovat ve formátech RTF nebo HTML. Tento model lze s výhodou použít k dokumentačním účelům ve všech fázích životního cyklu IS, kdy se používá PowerDesigner.

C.2.9 Free Model (FM)

Free model lze využít k modelování čehokoliv naprosto volnou notací. Správnost takového modelu však není nijak kontrolována a je tudíž vhodný jen k jednodušším dokumentačním účelům. Vazby mezi modely pro zajištění konzistence Vazby v PowerDesigneru

V rámci jednoho modelu lze využívat různého provázání objektů, které je specifické pro daný model. Například vazba dědičnosti mezi třídami v diagramu tříd apod. Mezi modely však takové vazby tvořit nelze. Mezi modely lze tvořit pouze 'Extended dependencies' – rozšiřující vazby.

Rozšiřující vazby jsou orientované vazby mezi libovolnými objekty nebo celými diagramy. Každá taková vazba se zobrazuje v záložce 'Extended dependencies' v properties zdrojového objektu a v záložce 'Dependencies', podzáložce 'Extended dependencies' u cílového objektu. Každá vazba může mít stereotyp. Standardně nejsou žádné stereotypy definovány.

Rozšiřující vazby byly původně určeny pouze k dokumentačním účelům. Pro potřeby této metodiky jsou využívány i k zajištění konzistence modelů ve smyslu dílů A a B. Součástí příloženého CD je definice kontrol rozšiřujících vazeb, která po doinstalování do PowerDesigneru kontroluje dodržování konzistenčních pravidel definovaných v dílech A a B a dále pro přehlednost vyjmenovaných. Zkontrolování konzistence se v prostředí PowerDesigneru provádí volbou v menu Tools>>Check Model.

C.2.10 Výčet sady konzistenčních pravidel

Následující text vyjmenovává sadu konsistenčních pravidel, která by měla být dodržována při tvorbě modelů, a to v zájmu udržení vnitřní kvality analýzy, jíž metodika nabízí. Tato pravidla jsou do různé míry a různými způsoby implementována v nástroji Power Designer v.12, jak je podrobněji popsáno v následující kapitole. Bez ohledu na to zda a jak je v nástroji dané pravidlo implementováno, jeho platnost je obecná a mělo by být při vývoji systému dodržováno.

C.2.10.1 Pravidla pro konsistenci vztahů mezi modelem tříd, jejich životními cykly (State Chart) a modelem procesů

- A) Každá třída objektů z modelu tříd musí být zastoupena v modelu procesů v alespoň jednom z jeho vstupů, či výstupů a/nebo aktérů, či jiných externích aspektů.
- B) Každý vstup, či výstup procesu, jakož i každý externí aspekt procesu, musí být zastoupen v modelu tříd jako třída, nebo asociace mezi třídami, či jako kombinace obojího.
- C) Každá třída objektů z modelu tříd musí mít specifikovány alespoň tři metody:
 - a. metodu, jíž objekt (instance třídy) vzniká (konstruktor),
 - b. metodu, jíž objekt (instance třídy) zaniká (destruktor),
 - c. alespoň jednu metodu, jíž se mění atributy objektu (transformer).
- D) Pro každý atribut třídy objektů z modelu tříd musí být u této třídy specifikována metoda, jíž je tomuto atributu přidělena počáteční hodnota a metoda, jíž je hodnota atributu změněna.
- E) Pro každou asociaci mezi třídami musí být u každé takto asociované třídy specifikována metoda, odpovídající této asociaci.
- F) Ke každé třídě objektů, která není považována za primitivní, je přiřazen stavový diagram, popisující její životní cyklus.
- G) Stavový diagram životního cyklu třídy musí mít popsány všechny přechody mezi všemi svými stavy. Popis každého přechodu specifikuje jednak událost, na základě které se přechod uskuteční (spoušť), jednak metodu, jíž se tento přechod uskuteční.

- H) Stavový diagram životního cyklu třídy musí obsahovat v popisech přechodů mezi stavy všechny metody této třídy. Popisy přechodů mezi stavy nesmí obsahovat metody, které nejsou specifikovány u této třídy.
- I) Každá událost, specifikovaná v popisech přechodů ve stavovém diagramu životního cyklu třídy, musí korespondovat s událostí, specifikovanou v popisu nějakého (nějakých) business procesu (procesů).

C.2.10.2 Pravidla pro konsistenci vztahů mezi modelem tříd (CD) a diagramem datových toků (DFD).

- J) Každý elementární Datastore v DFD musí být v CD zastoupen jako třída, nebo asociace, anebo kombinace obojího.
- K) Atributy každého elementárního Datastore z DFD musí být datovou strukturou atributů tříd, jimiž je tento Datastore v CD zastoupen.
- L) Metody každé elementární funkce z DFD musí být algoritmickou strukturou metod tříd, jimiž jsou v CD zastoupeny Datastorey, spojené datovými toky s touto funkcí

C.2.10.3 Pravidla pro konsistenci vztahů mezi modelem procesů (PD) a diagramem datových toků (DFD)

- M) Každý proces má vazbu alespoň na 1 funkci
- N) Každá funkce má vazbu alespoň na 1 proces
- O) Každá událost v procesním modelu má vazbu na vstupní tok (T-F) v DFD (protějšek k bodu M)
- P) Každý elementární vstupní datový tok v DFD od terminátoru (tj. zvnějšku systému) musí odpovídat nějaké události, specifikované v popisu nějakého (nějakých) business procesu (procesů) v PD.
- Q) Každý stav každého procesu v PD musí korespondovat s některým(i) elementárním(i) Datastorem(y) v DFD a naopak každý elementární Datastore v DFD musí korespondovat s některým(i) stav(y) procesů(ů) v PD. Jde o korespondenci M:N.

C.2.10.4 Ostatní doplňková pravidla

- R) Každý proces má vazbu alespoň na 1 požadavek
- S) Každý požadavek má vazbu alespoň na 1 proces
- T) Každá funkce má vazbu alespoň na 1 požadavek
- U) Každý požadavek má vazbu alespoň na 1 funkci
- V) Každá událost v STD by měla mít vazbu na vstupní tok v DFD
- W) Pokud je volitelný use case diagram použit, tak každý funkční požadavek má vazbu na alespoň jeden use case, který má vazbu na aktéra
- X) Každý use case, který má vazbu na aktéra, má vazbu i na alespoň jeden funkční požadavek
- Y) Use case má alespoň jednu vazbu na aktéra nebo je cílem alespoň jedné dependency nebo obojí

C.3 Podrobný popis realizace vybraných modelů a vazeb

Tato kapitola popisuje jak jsou realizovány některé specifické modely v nástroji Power Designer, nebo úpravy standardních modelů a vazeb mezi nimi ve smyslu podpory metodiky. Popisuje také způsob realizace podpory metodických (konsistenčních) pravidel v tomto nástroji.

Následující vazby mohou být uplatňovány za těchto předpokladů:

- Procesní model bude založen na jazyku BPMN 1.0
- Objektově orientovaný model bude založen na jazyku Analysis 2 a zároveň bude použito rozšíření (Extended Model Definition) pro diagram datových toků – DFD
- Pro model požadavků bude použito rozšíření RQM Language.

C.3.1 Diagram datových toků (DFD)

Použití

Diagram datových toků (DFD) byl implementován jako rozšíření (Extended Model Definition – EMD) objektově orientovaného modelu. Pro jeho použití je potřeba vytvořit nový diagram tříd s rozšířením DFD nebo toto rozšíření do již existujícího diagramu tříd dodatečně naimportovat.

- Vytvoření nového diagramu tříd s rozšířením DFD

File → New... → Object Oriented Model → záložka General: Class Diagram (language Analysis 2), záložka Extended Model Definitions → DFD

- Dodatečné naimportování rozšíření DFD

Model → Extended Model Definition → Import an Extended Model Definition → DFD

Po úspěšném zavedení DFD rozšíření bude v každém diagramu tříd nově přístupná paleta nástrojů s prvky pro tvorby diagramu datových toků.



Paletu s DFD nástroji lze v případě potřeby obvyklým způsobem zavřít, její opětovné otevření probíhá přes menu:

Tools → Customize Toolbars... → DFD

Kontrola formální správnosti diagramu DFD

Kontrola formální správnosti diagramu je zajišťována ihned při jeho vytváření. Pokud by došlo vložením některého objektu ke vzniku chybového stavu, je uživatel upozorněn na to, jaká chyba nastala a chybně vložená vazba je ihned odstraněna. Kontrolovány jsou následující situace:

- Rekurzivní Datový tok – nemůže být vytvořen takový Datový tok, který vchází do toho samého objektu, ze kterého i vychází (Terminator, Datový sklad, Funkce). Po potvrzení MessageBoxu s chybovou hláškou je neplatná vazba ihned odstraněna.
- Nepovolené vazby těchto typů:
 - Terminator – Terminator
 - Data Store – Data Store
 - Terminator – Data Store
 - Data Store – Terminator

Tyto vazby jsou povolené pouze prostřednictvím Funkce Při pokusu vytvořit vazbu přímo bude zobrazen MessageBox s oznámením chyby a následně bude neplatná vazba odstraněna.

Úpravy pro další verzi

Kontrola Datového skladu – do a z každého datového skladu musí vést alespoň jeden Datový tok

C.3.2 Vazba externích aspektů procesu na diagram tříd

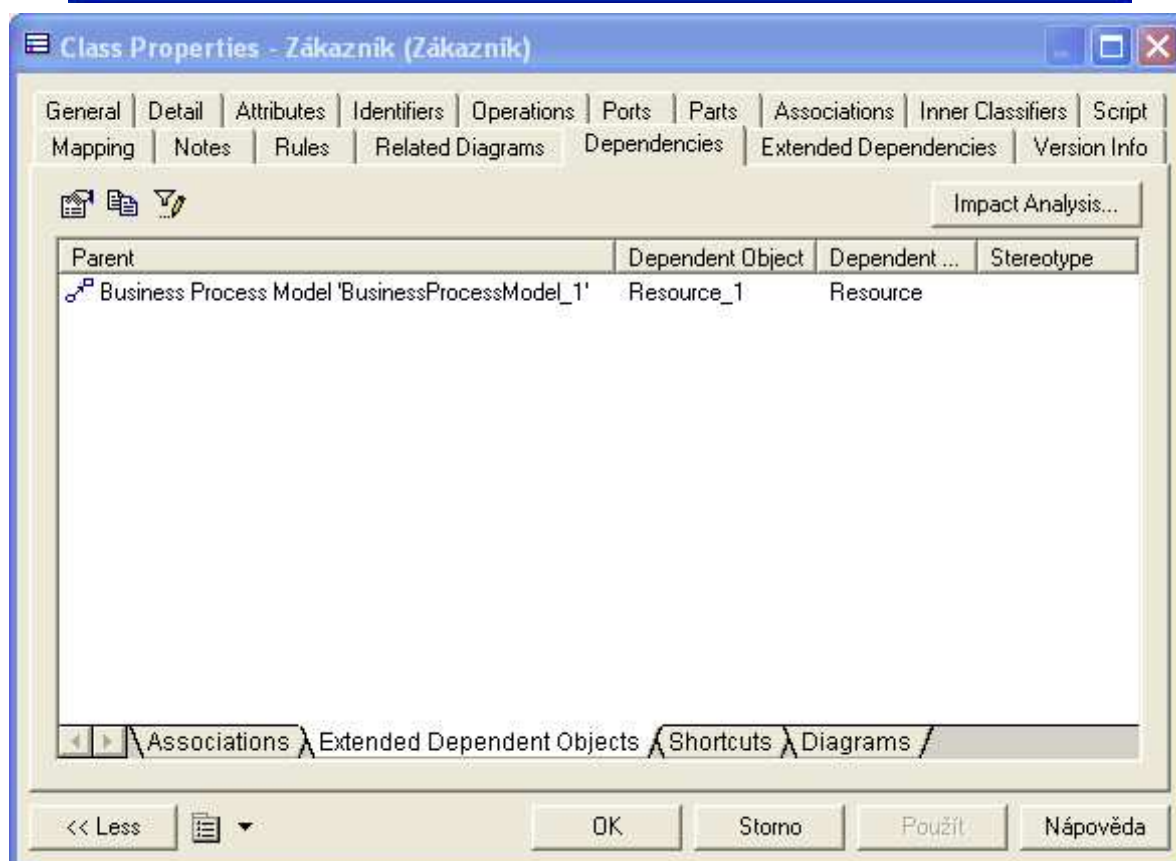
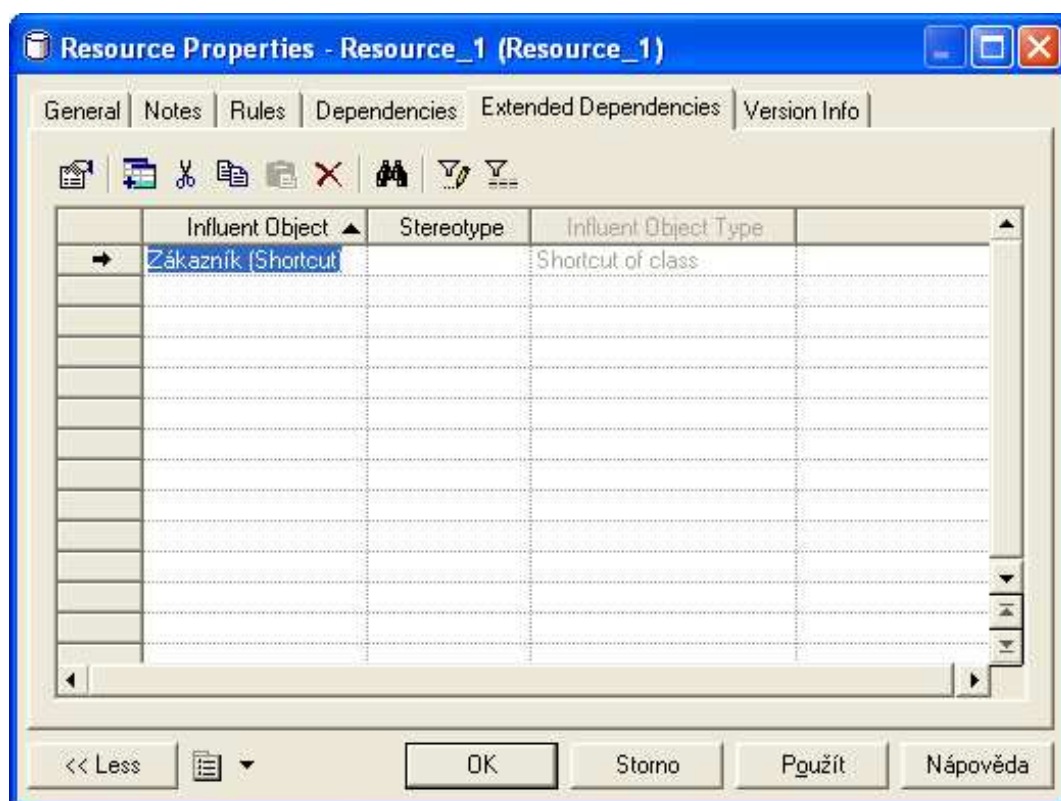
Je potřeba vytvořit a kontrolovat vazbu mezi externími atributy procesu v procesním modelu a třídami, asociacemi nebo kombinacemi tříd a asociací v diagramu tříd. Vazba není v diagramech vidět, ale je možné ji nalézt ve vlastnostech obou svázaných elementů.

Postup tvorby

Tuto vazbu je možné vytvořit ve vlastnostech externího atributu (Resource, Organization Unit) procesu na záložce Extended dependencies - Add Object. Po vybrání odpovídajícího diagramu tříd je možné si vybrat konkrétní třídu, se kterou by měl být svázán. U této třídy potom bude vazba vidět na kartě Dependencies.

Kde je vazba k nalezení:

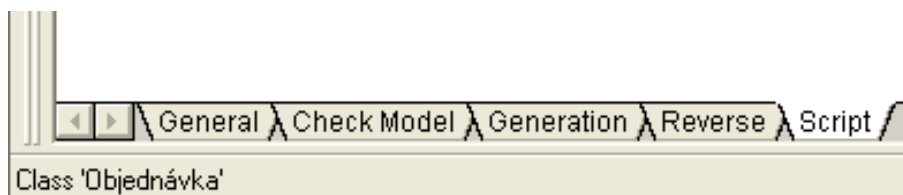
- externí atribut v procesním modelu: Vlastnosti - záložka Extended Dependencies
- třída v objektovém modelu: Vlastnosti - záložka Dependencies - karta Extended Dependent Objects (EDO), v dialogu dole



Kontrola existence vazby

Kontrola, zda všechny externí atributy procesního modelu obsahují tyto vazby, je prováděna po spuštění Check Modelu, konkrétně to jsou položky „Organization unit to class“ a

„Resource to class.“ Detailní rozpis všech nalezených chyb je k dispozici na záložce Script po proběhnutí Check Modelu.



Jediné, na co je třeba dát pozor při opakovaném spouštění Check modelu je fakt, že detailní výpisy chyb se při spuštění kontroly nemažou, proto se případné nově nalezené chyby přidávají za chyby z minulých kontrol. Výpis je tedy nutné mazat příkazem Clear v kontextovém menu.

Úpravy pro další verzi

Bude přidána kontrola i ze strany diagramu tříd, tzn. bude kontrolováno také to, jestli každá třída má vazbu na externí atribut(y) v procesním modelu.

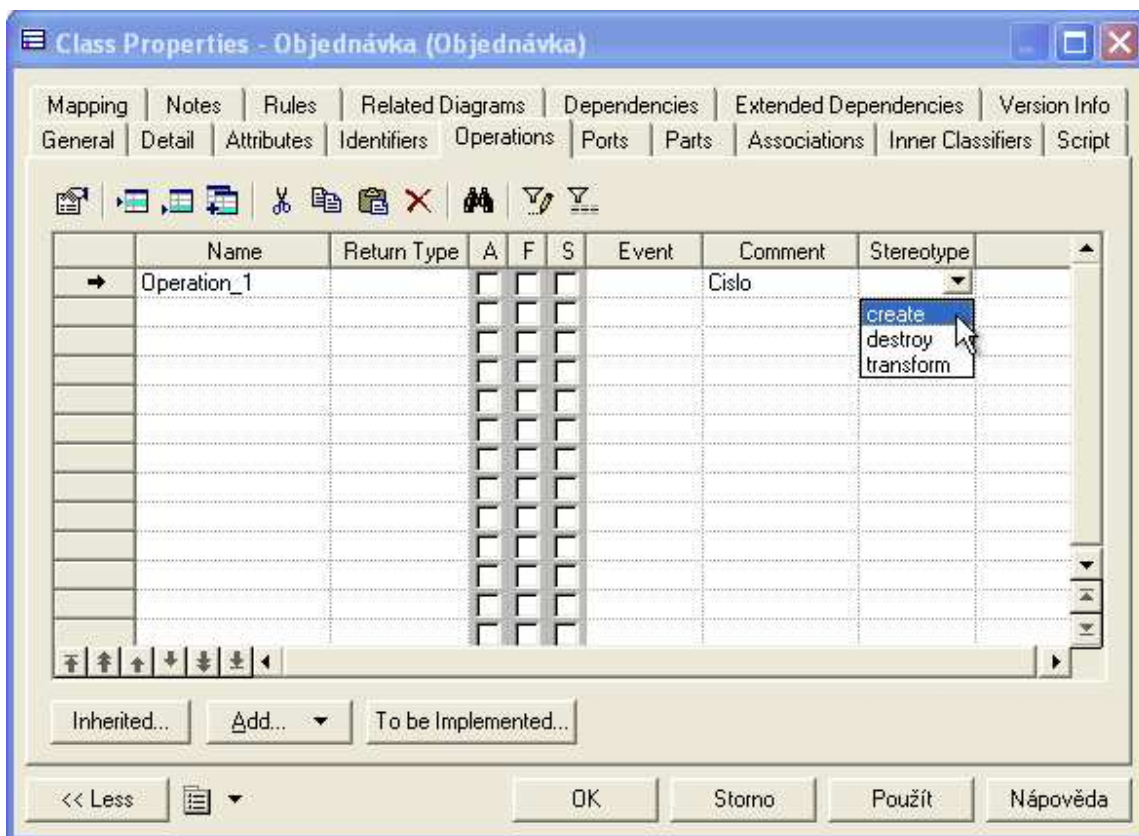
C.3.3 Specifikace a kontrola povinných metod u každé třídy

Každá třída objektů v diagramu tříd musí mít specifikovány alespoň tři následující metody:

- metodu, jíž objekt (instance třídy) vzniká (konstruktor)
- metodu, jíž objekt (instance třídy) zaniká (destruktor)
- alespoň jednu metodu, jíž se mění atributy objektu (transformer)

Postup tvorby

Tato „specifikace“ metody se provádí ve vlastnostech dané třídy v poli Stereotype. Pro každou třídu je potřeba definovat alespoň tři metody, každou s jedním ze stereotypů create/transform/destroy. Ostatní metody mohou být i bez definovaného stereotypu.



Kontrola existence povinných metod

Splnění těchto podmínky kontroluje Check „Basic methods check.“ Funguje tak, že je testována přítomnost alespoň jedné metody každého povinného stereotypu u každé třídy.

C.3.4 Metody atributů

Pro každý atribut třídy z diagramu tříd musí být u této třídy specifikována metoda, kterou bude danému atributu přidělena počáteční hodnota. Dále by měla být u této třídy specifikována metoda, jíž může být hodnota atributu změněna.

Postup tvorby

Propojení mezi jednotlivými metodami a příslušnými atributy je realizováno vkládáním názvu atributu do komentáře metody. Na obrázku u předchozí kapitoly je uveden v komentáři (Comment) název atributu `Cislo`. Jméno každého atributu by tedy mělo být v komentáři alespoň dvou metod této třídy.


Kontrola existence nastavených stereotypů

Splnění této podmínky testuje Check „Methods of attributes,“ který testuje, zda je název atributu obsažen v poli komentář alespoň u dvou metod. Pokud není uveden vůbec, nebo pouze jednou, následuje detailní výpis na záložce Script, který upřesní zjištěné nedostatky.

C.3.5 Metody asociovaných tříd

Pro každou asociaci mezi třídami musí být u obou takto asociovaných tříd specifikována metoda odpovídající této asociaci (resp. třídě na druhém konci asociace).

Postup tvorby

Vazba na asociovanou třídu se tvoří specifikováním pole Return Type u dané metody třídy. Tato vazba není oboustranná a je tedy nutné definovat odpovídající metodu u obou asociovaných tříd. Vazba se určuje v seznamu metod třídy v poli Return Type, kde je po kliknutí na tlačítko  (Select Classifier) vybrána odpovídající třída.

Kontrola metod asociovaných tříd

Check „Methods of associated classes“ kontroluje vždy vazbu mezi dvěma, asociací spojenými třídami a to tak, že každá třída musí obsahovat metodu, jejíž Return Type odpovídá názvu asociované třídy. Mohou nastat pouze dva typy chyb:


- Obě třídy na sebe vzájemně nemají vytvořeny metody
- Pouze jedna třída nemá vytvořenu metodu na protější třídu

Obě varianty jsou v případě výskytu opět detailně rozepisovány na záložce Script.

C.3.6 STD diagram pro každou neprimitivní třídu

Ke každé třídě objektů, která není považována za primitivní, je přiřazen stavový diagram, popisující její životní cyklus.

Postup tvorby

Ve vlastnostech každé neprimitivní třídy je na kartě Related Diagrams pomocí tlačítka Add Object  potřeba přiřadit odpovídající Statechart diagram.

Kontrola existence nejvýše jednoho STD u každé třídy

Check pro tuto vazbu je nazván „Related Statechart diagram“ a kontroluje existenci maximálně jedné vazby na Statechart diagram u každé třídy. Třída tedy může mít žádný nebo jeden navázaný Statechart diagram, nemůže jich ale mít více.

C.3.7 Vazba elementárních Data storů do diagramu tříd

Každý elementární Datastore v DFD byl v diagramu tříd zastoupen jako třída, asociace anebo kombinace obojího.

Postup tvorby

Každý Data Store je do diagramu vkládán implicitně jako elementární (stereotyp Data Store). V případě potřeby vytvořit komplexní Data Store stačí změnit stereotyp vloženého elementárního Datastoru na „Complex Datastore.“ V tom okamžiku pro něj přestává být nezbytné obsahovat alespoň jednu vazbu (Extended dependency) do diagramu tříd (nicméně stále je to možné), ale je nutné, aby obsahoval alespoň jeden vnořený objekt. I u vnořených elementárních datastorů je nutné, aby měly alespoň jednu vazbu do diagramu tříd.

Kontrola formální správnosti vazeb a struktury datastorů

Check kontrolující tyto vlastnosti je nazván „Elementary and Complex Data Store test“. U elementárních datastorů (stereotyp Data Store) je kontrolována přítomnost alespoň jedné vazby na třídu, asociaci nebo kombinaci obojího v diagramu tříd. U komplexních datastorů (stereotyp Complex Datastore) je kontrolována existence vnitřních objektů, kontrolou tedy neprojde prázdný komplexní datastore.

Úpravy pro další verzi

- a) zajistit, aby Data Store nemohl mít inner classes/classifiers,
- b) nebo při vložení inner class dojde automaticky ke změně stereotypu na Complex Datastore.

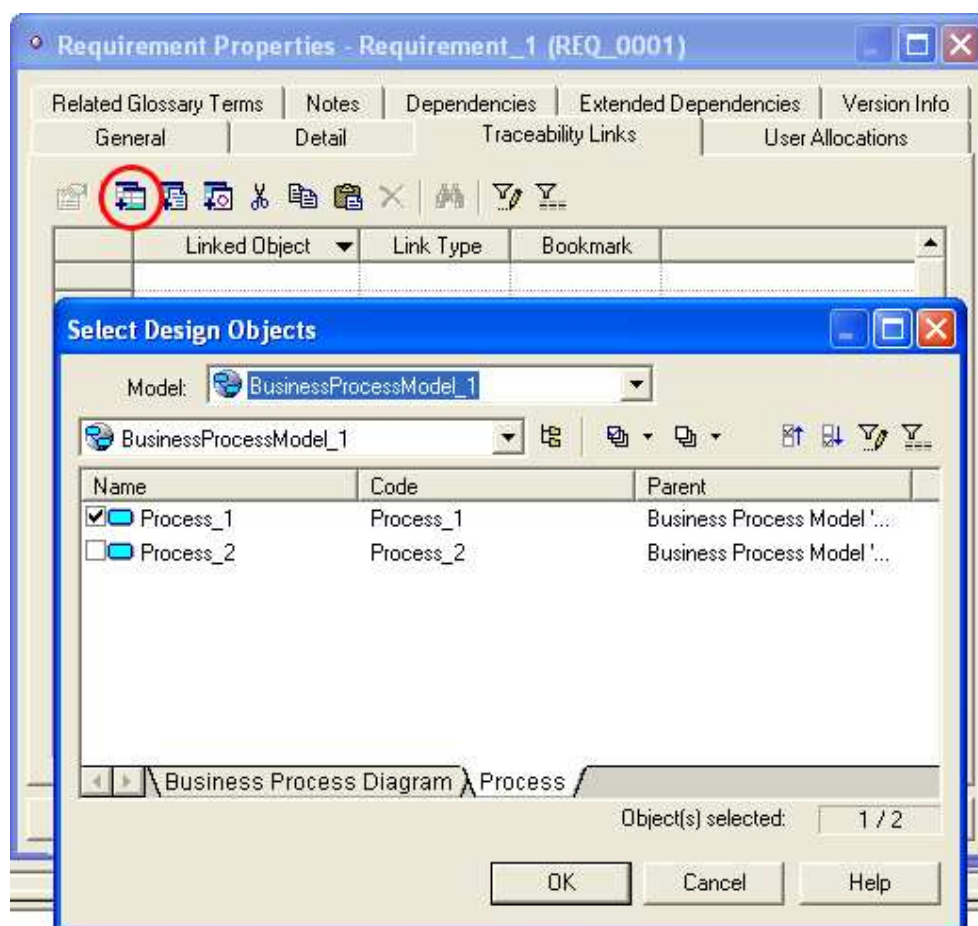
C.3.8 Vazba mezi procesem a požadavkem

Účelem je vytvořit a kontrolovat vazby mezi procesním modelem a modelem požadavků tak, aby měl každý proces vazbu na alespoň jeden požadavek a každý požadavek vazbu na alespoň jeden proces.

Postup tvorby

Vzhledem k tomu, že ve vlastnostech procesního (a obecně jakéhokoliv jiného modelu, jehož prvky mohou obsahovat vazby na požadavky) modelu není implicitně přístupná karta Requirements, sloužící k propojování procesního modelu (BPM) s modelem požadavků, je zapotřebí nejdříve procesní model pro vazby na požadavky nakonfigurovat. To je možné provést dvěma způsoby:

- z procesního modelu:
Tools → Model Options → Model Settings → Check Box „Enable links to requirements“
- z modelu požadavků:
To je možné provést tak, že prvotní vazba procesu s požadavkem se vytvoří v modelu požadavků ve vlastnostech konkrétního požadavku na kartě Traceability Links, ke které se odstaneme následovně:
Properties → Traceability Links → Add Links to Design Objects → označit proces pro vazbu



Po potvrzení vazby na konkrétní proces je ještě nutné odsouhlasit dotaz na doplnění procesního modelu o tento druh propojení a od tohoto momentu je možné vytvářet vazby z obou modelů bez nutnosti jakékoliv další konfigurace.

Kde je vlastnost k nalezení:

- procesní model: Properties → záložka Requirements → Add Objects
- model požadavků: Properties → Traceability Links → Add Links to Design Objects

Kontrola existence vazeb

Je potřeba kontrolovat, jestli každý proces v procesním modelu má vazbu alespoň na jeden požadavek a naopak. První část kontroly byla implementována jako Custom Check „Process to requirement“ v procesním modelu, druhá část jako Custom Check „Requirement to process“ v modelu požadavků.

C.3.9 Vazba mezi funkcí a procesem

Každá funkce v diagramu datových toků musí mít vazbu na alespoň jeden proces.

Postup tvorby

Tuto vazbu je možné vytvořit ve vlastnostech funkce na záložce Extended Dependencies. Properties → Extended Dependencies → Add Objects

Kontrola existence vazeb

Vazbu funkce na proces kontroluje Custom check „Function to process.“

Úpravy pro další verzi

Bude implementována kontrola vazby i pro druhou stranu, tedy vazba procesu na alespoň jednu funkci.

C.3.10 Vazba mezi funkcí a požadavkem

Každá funkce v diagramu datových toků musí mít vazbu na alespoň jeden požadavek.

Postup tvorby

Diagram datových toků je stejně jako v případě procesního modelu potřeba nejdříve na vazby na požadavky nakonfigurovat. I tady je to možné provést dvěma způsoby, stejně jako u procesního modelu.

Kde je vazba k nalezení:

- DFD: Vlastnosti Funkce - záložka Requirements (nutno mít povolené/povolit requirement links v model options)
- RQM: Vlastnosti požadavku - záložka Traceability Links nebo záložka Dependencies - karta Design Objects

Kontrola existence vazeb

Vazbu funkce na alespoň jeden požadavek kontroluje Custom check „Function to requirement.“

Úpravy pro další verzi

Bude implementována kontrola vazby i pro druhou stranu, tedy vazba požadavku na alespoň jednu funkci.

C.3.11 Vazby mezi Use Case a funkčním požadavkem

Každý use case, který má vazbu(association) na aktéra, má vazbu i na alespoň jeden funkční požadavek. Platí to i naopak, tedy každý funkční požadavek má vazbu na alespoň jeden use case, který má vazbu na aktéra. Zároveň platí, že každý Use Case má vazbu buď na aktéra nebo je cílem alespoň jedné závislosti (dependency).

Postup tvorby

Vazba Use Case na požadavek je tvořena stejným způsobem jako u ostatních diagramů objektového modelu, tedy ve vlastnostech na záložce Requirements. Vazba z opačné strany, tedy vazba požadavku na Use Case, se tvoří ve vlastnostech požadavku na záložce Traceability Links.

Kontrola existence vazeb

Tuto vazbu kontrolují celkem dva checky:

- model požadavků: Check „Requirement to Use Case“ kontroluje přítomnost vazby na alespoň jeden Use Case u každého funkčního požadavku

- Use Case diagram: Check „Use Case to requirement“ kontroluje u Use Casů, které mají vazbu na aktéra, vazbu na požadavek a u těch, které vazbu na aktéra nemají, kontroluje, jestli jsou cílem alespoň jedné závislosti.

Díl D Použití metodiky

D.1 Metodika a standardy

Metodika je vystavěna na existujících metodách, technikách a nástrojích. V případě potřeby si z použitých metod, technik či nástrojů vybírá vhodné části a jiné nechává k volitelnému použití podle úvahy a specifických potřeb uživatelů. Pro modelování struktury reality využívá některé diagramy jazyka UML, pro popis chování reality využívá procesního modelování podle standardu BPMN a pro popis funkcionality systému používá standardní Data Flow Diagram ze strukturované analýzy.

Základními zdroji standardů v této metodice tedy jsou:

- jazyk UML ([UML, 2006])
- standard Business Process Modelling Notation ([BPMN, 2005])
- Yourdonova strukturovaná analýza ([Yourdon, 1989])

Metodika je vystavěna tak, aby ji standardy v principu nijak neomezovaly. Již tím, že je metodika postavena na základních principech, jež mají nadčasovou platnost (viz kapitolu A1), je částečně zajištěna její nezávislost na standardech, které v budoucnu vzniknou¹⁹. To je dobře vidět například na vztahu metodiky k nejsilnějšímu oborovému standardu – jazyku UML.

Z tohoto jazyka využívá metodika pouze klíčové analytické nástroje – diagram tříd (Class Diagram) a diagram stavů (State Chart) a oba klíčové nástroje designové – diagram komponent (Component Diagram) a diagram rozmístění (Deployment Diagram). Jmenované nástroje jsou přímým odrazem základních principů a lze tedy očekávat, že se jejich podstata měnit nebude. Ostatní nástroje, včetně oblíbeného diagramu případů užití systému (Use Case Diagram), které již nejsou přímým odrazem základních principů a jejich význam často bývá předmětem sporů mezi různými metodickými přístupy, jsou touto metodikou považovány za doplňkové²⁰ a metodika je tak na nich i na jejich případném dalším pohnutém osudu, tím, že se jí netýká, dokonale nezávislá.

Diagram datových toků je tradičním de-facto standardem pro funkční modelování ještě z dob strukturovaných přístupů k vývoji IS. Jeho absence v systému standardních nástrojů objektového přístupu v UML je asi nejdiskutovanějším předmětem sporů různých metodických přístupů a potažmo brzdou dalšího vývoje metodik v oblasti (srovn. např. jeho klíčovou úlohu v původní metodice OMT v [Rumbaugh, 1992]). Nejsystematičtěji jej, včetně klíčových vazeb na ostatní standardní modelovací nástroje, popisuje Ed Yourdon ve své Strukturované analýze ([Yourdon, 1989]). V naší metodice je diagram datových toků implementován jako specializace modelu tříd, čímž je v základních rysech a priori dokonale konsistentní s klíčovými nástroji objektového modelování, přičemž smysluplnost pojetí DFD jako specializace diagramu tříd je metodicky odůvodněna (mimo jiné i odkazy na teorie kolem typové specializace objektů, existence tzv. funkčních a datových objektů apod.). Jeho další vývoj je tak plně říditelný podle vývoje diagramu tříd.

Nejméně usazená oblast standardů jsou standardy procesního modelování. Zde metodika vychází z vlastního metamodelu podnikového procesu²¹, jenž je základním obsahovým vymezením potřebných vlastností nástroje procesního modelování (Diagramu procesů). Tento metamodel zde slouží jako základní šablona, podle níž jsou srovnávány příslušné existující

¹⁹ pokud tyto standardy budou vycházet ze stejných principů, ovšem, což by se mělo dát předpokládat

²⁰ jak je vysvětleno v příslušných kapitolách dílů B a D této metodiky

²¹ protože standardizace v oblasti modelování procesů není doposud na takové úrovni zralosti, aby bylo zvykem standardy stavět na systematicky rozvíjeném metamodelu, jako je tomu např. v případě UML

standardy diagramu procesů. V současnosti patří k nejsilnějším standardům Business Process Modelling Notation (BPMN), podporovaná i nástrojem Power Designer v.12 a tak se i vyskytující v příkladech v této publikaci. Vazba BPMN na základní konstrukty metamodelu je podrobněji popisována v kapitole B1.2.

D.2 Role metodiky v organizaci

Je zřejmé, že žádná metodika není a ani nemůže být v praxi striktně vynucována a dodržována ve všech svých detailech. Metodika je určitou kostrou poskytující základní rámec pro vývoj systému, poskytuje návod jak postupovat v jednotlivých fázích životního cyklu informačního systému. V některých částech sama nabízí variantní použití doplňujících technik a nástrojů. Všechny části metodiky mohou být a pravděpodobně budou uživateli upravovány podle specifických potřeb organizací. Takové změny jsou možné pokud přinášejí dostatečné zlepšení a pokud jsou v souladu se základními principy metodiky a neporušují konzistenci použitých nástrojů a technik. Z tohoto pohledu je třeba tuto metodiku chápat jako šablonu pro konkrétní metodiku organizace, ikdyž se tato šablona snaží být kompletní, konzistentní a univerzálně použitelná.

Role konkrétní upravené metodiky v organizaci je především ve strategickém uchopení business modelu organizace a jeho provázání na informační systém. I když je metodika vytvořena s cílem sladit vývoj IS s business modelem, právě zvolená cesta přes business model a strategii organizace dává metodice daleko širší význam než jen vývoj IS. Pomocí modelování business systému je možné odhalovat zákonitosti a zlepšovat fungování organizace jako takové. V tomto smyslu je metodika provázána i s principy obecného procesního řízení organizace a metodami procesního reengineeringu a tím pádem relevantní i pro širší vedení společnosti - ne jen specialistů na IT.

Druhou důležitou rolí metodiky je kontinuální zlepšování fungování organizace v měnícím se prostředí. Tohoto cíle metodika dosahuje pomocí sběru požadavků uživatelů systému a jejich analýzou - jedná se tedy o zlepšování fungování organizace odspoda (bottom-up). Analýza těchto požadavků může odhalit nutnost změny IS ale i business systému organizace - struktury strategických cílů i fungování procesů. S rostoucí mírou turbulence podnikatelského prostředí rostou i požadavky na rychlé přizpůsobení organizace na všech úrovních od strategických cílů, přes procesy až k IS. Některé cíle, procesy a části IS mohou v průběhu delšího času dokonce zanikat nebo být nahrazovány novými. Má-li zůstat organizace životaschopná, musí nutně dojít ke konzistentní změně ve všech oblastech. Pakliže je analýze požadavků věnována patřičná péče, zpětnovazebné působení bude udržovat business procesní model v „živém“, tj. aktuálním stavu odpovídajícím aktuálním potřebám organizace. Aktuální stav strategie společnosti, odpovídajících procesů a jejich podpory informačním systémem je předpokladem dlouhodobého úspěchu organizace.

D.3 Role uživatelů ve vztahu k metodice

D.3.1 Role vedení

Role vedení organizace ve vztahu k metodice je dobře viditelná v pojetí, jež představuje Obrázek D 1 Projekty IS/ICT v kontextu procesů a řízení organizace. Vedení tým, že rozhoduje o změnách strategického zaměření, rozhoduje současně i o potřebách informační podpory činnosti organizace. V současné situaci přechodu organizací na procesní řízení je

nezbytné jednak už ve strategickém rozhodování proaktivně uvažovat roli informačních technologií a systémů přímo při koncipování business procesů, jednak si uvědomovat strategický význam této podpory. Pružná informační podpora procesů, korespondující s jejich principiální proměnlivostí, se stává nezbytností. Na druhou stranu však pružný informační systém nezbytně vyžaduje přímé zapojení již konceptorů systému business procesů (a tato koncepce začíná na úrovni strategického vedení organizace) do procesu jeho vývoje. Znamená to, že vrcholové vedení musí umět přesně specifikovat své záměry v termínech business procesů a jejich realizačních specifik – a to včetně využití technologie. Takový vpravdě intimní vztah mezi strategickým vedením organizace a vývojem jejího informačního systému se neobejde bez precizního jazyka pro komunikaci konceptorů businessu a realizátorů jeho informační podpory (IS). A právě takovým jazykem se snaží být popisovaná metodika.

D.3.2 Role analytiků

Analytici organizace jsou rolí, pro kterou je metodika určena především. Především oni by měli dbát na celkovou konzistenci reality, modelů reality a modelů IS, odhalené problémy analyzovat a v součinnosti s ostatními rolemi také navrhovat řešení. Z toho vyplývají značné nároky na schopnosti analytiků i na jejich osvojenou znalost celé metodiky a pravidel konzistence.

D.3.3 Role programátorů, designerů a jiných profesí podílejících se na vývoji IS

Design, implementace a zavádění nových systémů, stejně jako výběr, nákup a integrace dodaných systémů musí být v organizaci ve vzájemném souladu s ostatními částmi IS. Zajišťování této konzistence je smyslem metodiky, proto i všichni pracovníci podílející se na rozvoji IS by měli metodiku znát a řídit se jejím doporučeními. Jejich role spočívá především v přejímání závěrů analytiků a v detailizování analytických modelů vytvořených za použití metodiky a jejich aplikaci na konkrétní vyvíjené či nakupované části IS. Při své práci by designéři, programátoři a systémoví integrátoři měli dbát na hierarchickou konzistenci s modely reality a informačního systému, čímž bude zajištěna efektivní informační podpora strategických cílů organizace.

D.3.4 Role pracovníků helpdesku

Při provozu informačního systému je kladen značný důraz na realizaci uživatelského střediska (helpdesk / hotline), které je v tomto smyslu základním bodem signalizace objektivní potřeby změny IS (viz Obrázek D 1). Uživatelské středisko tak má důležitou roli prvotního analyzátorů požadavků na IS. Na něm leží hlavní odpovědnost za to, že žádný takový signál nezůstane nevyslyšen i za to, že u požadavků, signalizujících stejnou vlastnost reálného systému, bude identifikován stejný původ a nebudou řešeny jako různé (což poskytuje příležitost k nekonzistencím ve vývoji IS). Mimo jiné plynou z výše uvedeného mimořádné požadavky na kvalifikaci personálního obsazení uživatelského střediska: pracovník uživatelského střediska musí mít především analytické schopnosti, musí být schopen odpovědně odlišit subjektivní požadavky, či omyly a nepochopení uživatelů od signálů o objektivní potřebě změny v informačním systému, nebo dokonce v reálném systému (požadavek na IS může také signalizovat nedostatky v řízení organizace, či její strategii).

D.4 Metodika a řízení projektů

Problematika vztahu mezi metodickým postupem prací na vývoji IS a zákonitostmi vedení projektu je již diskutována v obecné rovině v kapitole A3. Tato kapitola doplňuje obecně

pojednání v kapitole A3 o jednoduchou ilustraci na příkladu možného promítnutí metodických návodů jak z oblasti tvorby IS, tak z oblasti vedení projektu do jednoho konkrétního postupu konkrétního projektu.

V rámci vedení projektu dělíme činnosti a práci, kterou je třeba vykonat na libovolné menší celky až do takové úrovně detailu, která umožňuje rozdělení dílčích úkolů na jednotlivé členy týmu popř. pracovní skupiny a úspěšné zvládnutí projektu. Zde se nabízí několik variantních postupů, z nichž uvedme dva:

- a) jednotlivé pracovní skupiny dostávají přesně ohraničené celky, o jejichž vývoj se starají v průběhu celého metodického cyklu popř. více cyklech
- b) jednotlivé pracovní skupiny pracují způsobem „manufaktury“, v rámci metodického cyklu navazují jedna na druhou

Poznámka: Metodickým cyklem rozumíme průchod všemi fázemi metodiky od prvního kroku po poslední.

Stanovíme milníky projektu tak, aby byly umístěny vždy

- a) na konci fáze – tato varianta je vhodná v případě velmi rozsáhlých projektů a v počátku projektu
- b) na konci celého cyklu – výhodné u menších projektů, kde si můžeme dovolit postupovat přes celý cyklus až po nasazení a testování a odvození nových požadavků pro další cyklus. Tuto variantu je vhodné použít v závěru projektu, kdy je již IS v testování a máme tak dostatečnou zpětnou vazbu v podobě proudu požadavků od testerů.

Ukažme si nyní příklad průchodu cyklem metodiky na nejobecnější rovině.

1. Analýza
 - 1.1. Analýza business strategie
 - 1.2. Tvorba analytických modelů
 - 1.2.1. Tvorba BPM (Business Process Modelu)
 - 1.2.2. Tvorba modelu požadavků (Requirements Model)
 - 1.2.3. ERD
 - 1.2.4. DFD (Data Flow Diagram)
 - 1.2.5. Diagram tříd (Class Diagram)
 - 1.2.6. Diagram užití (Use Case)
2. Design
 - 2.1. Tvorba designových modelů
 - 2.1.1. Model komponent (Component Model)
 - 2.1.2. Model nasazení (Deployment Model)
3. Nasazení
4. Testování v provozu
5. Odchytávání požadavků

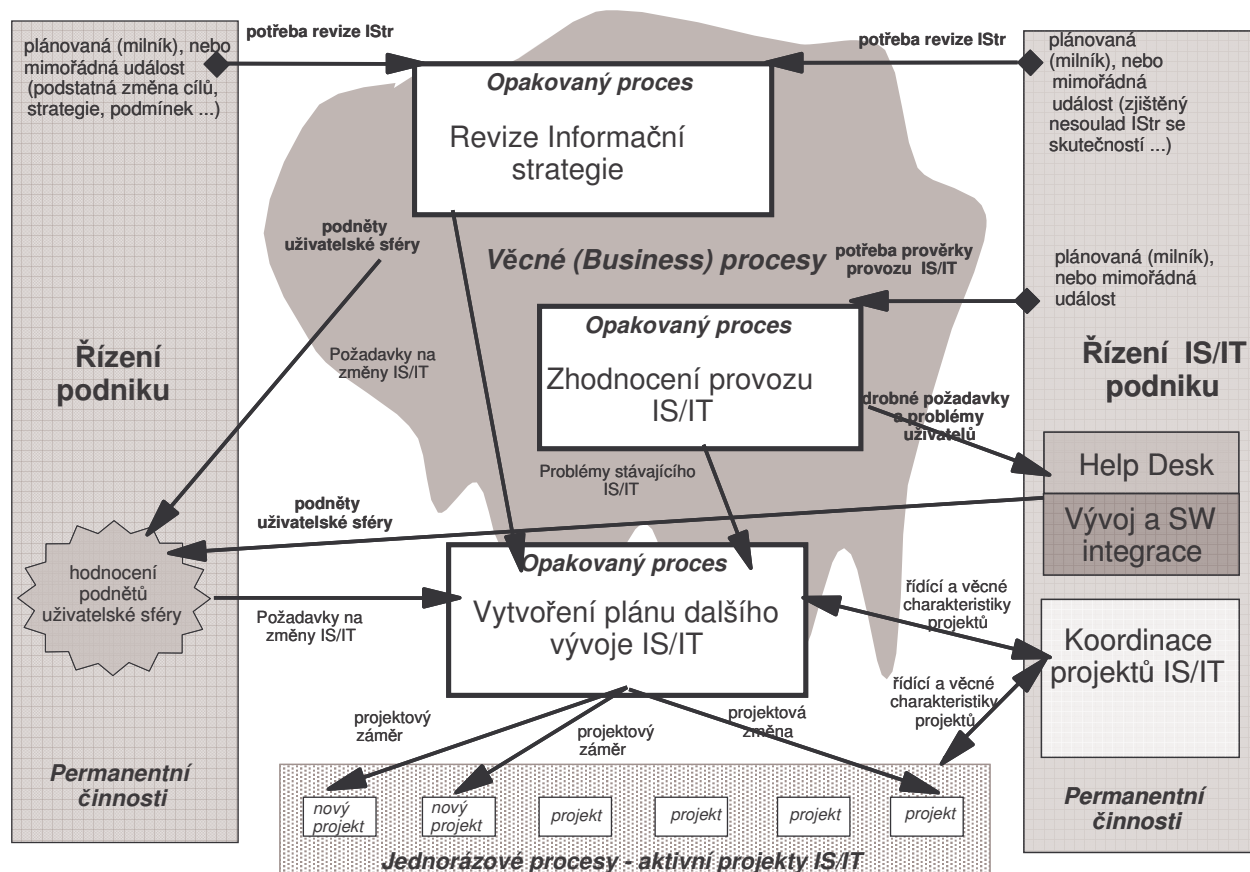
Na fázi 5 navazuje znovu fáze 1, další cyklus znovu začíná analýzou požadavků na IS, tentokrát přírůstku (nebo změně) požadavků oproti požadavkům předchozímu cyklu.

Fáze 1.2.3 až 1.2.6 mohou být řazeny také alternativně. Tedy není nutné dopracovávat se k modelu tříd přes mezistupeň zjednodušeného statického vidění business systému v

objektech datového modelu, zvláště pokud je analytikům přirozený pohled na realitu prostřednictvím modelu tříd. Obdobně není nutné specificky mapovat uživatelské rozhraní informačního systému, jelikož jeho obsah je již úplně definován v diagramu datových toků (DFD). Nicméně může to být velmi vhodné jako doplňkový model, zdůrazňující vidění systému uživatelskýma očima, stejně jako častý nástroj užitečných semi-formálních metod analýzy uživatelských požadavků, jakou je například oblíbený prototyping.

D.5 Změna Business Process Modelu jako cesta ke zlepšení systému

Z hlediska vztahu metodiky a řízení organizace je podstatná zpětnovazební smyčka, které umožňuje kontinuální zlepšování business modelu. IS podléhá neustálé změně, a to jak ve fázi vývoje, tak také po jeho nasazení. Vzniká potřeba reakce na nově přichodící požadavky a z toho plynoucí úpravy IS a přidávání nových funkcionalit. S rostoucí mírou turbulence podnikatelského prostředí dochází ke stále častějším změnám business strategie, které se projevují změnou business procesů. Procesy mohou v průběhu delšího času dokonce zanikat nebo vznikat. Má-li zůstat organizace životaschopná, musí nutně dojít k promítnutí těchto změn do IS. Veškeré známé požadavky na IS musí být pečlivě analyzovány a v případě, že se prokáže jejich opodstatněnost, implementovány do IS. Tuto zpětnou vazbu ilustruje Obrázek D 1.



Obrázek D 1 Projekty IS/ICT v kontextu procesů a řízení organizace

Obrázek ukazuje základní vztahy mezi řízením organizace a řízením jejího IS/IT. Vývoj IS/IT probíhá prostřednictvím jednotlivých projektů, zatímco jeho provoz je záležitostí rutinních

opakovaných procesů. Rozhraní mezi provozem IS/IT a jeho vývojem – tedy jak přechází rutina do vzniku něčeho nového – je v podstatě esencí problematiky řízení IS/IT. A k ujasnění tohoto rozhraní je nezbytné si ujasnit také rozhraní mezi reálným systémem organizace – jejími business procesy – a systémem jeho informační podpory – IS/IT. Esence problematiky řízení IS/IT je tak současně definicí základního smyslu a základních východisek existence IS/IT.

Na obrázku jsou vidět procesy – činnosti čtyř základních druhů.

- **věcné (business) procesy** tvoří základ fungování celé organizace. Jsou to procesy, tvořící běžný život organizace, procesy, do nichž je koncentrován celý smysl její existence. Z nich také pocházejí základní podněty jak pro vývoj informačního systému, tak pro řízení podniku jako celku. Často ani nejsou jednotliví původci těchto podnětů schopni rozlišit skutečnou podstatu svého problému, své potřeby. Tak se jim mohou typicky jevit problémy / potřeby změny v řízení, či strategickém zaměření organizace, jako problémy nedostatečné podpory jejich práce informačním systémem.
- činnosti řízení organizace a řízení jejího IS/IT jsou **činnosti permanentní**. Probíhají prakticky stále, průběžně, jsou spíše nestrukturované, ad-hoc reaktivní, nemají charakter typického procesu, spíše permanentně probíhající činnosti. V rámci těchto činností se odehrávají důležité procesy hodnocení podnětů uživatelské sféry, integrace vývoje SW a koordinace projektů IS/IT. V kompetenci těchto činností je též identifikace důležitých podnětů řídicího charakteru – podněty k revizi strategie, prověrce IS/IT apod.
- procesy Revize informační strategie, Zhodnocení provozu IS/IT a Vytvoření plánu vývoje IS/IT na další období jsou **procesy opakované**. Se slovníkem procesního analytika bychom je mohli nazvat procesy, řízenými **časovanými událostmi**. Primárně mají tedy tyto procesy plánovitý charakter. Kromě plánovaného běhu tyto procesy ovšem slouží i ad-hoc situačnímu použití (v případě mimořádné události, problému, na základě výsledku auditu IS/IT apod.). Jedná se o hlavní řídicí procesy z oblasti řízení IS/IT organizace.
- **projekty vývoje IS/IT** jsou zvláštním druhem procesů – **procesy jednorázovými**. Jejich formou probíhá veškerý vývoj IS/IT. Oblast projektů a jejich vedení je samostatnou metodickou oblastí s vlastními, na účelu projektu (v tomto případě vývoji IS/IT) do značné míry nezávislými, pravidly, jak je diskutováno již v předchozích kapitolách. Nicméně tyto projekty neprobíhají bez vzájemných souvislostí a bez širšího kontextu. Zatímco řízení každého **jednoho projektu** je relativně samostatná záležitost, každý projekt má svůj plán, tým, řídicí strukturu a komunikační procedury, vlastní mechanismy řízení změn apod., nad všemi projekty je současně nutno uvažovat procesy jejich vzájemné koordinace a řízení jako **systému projektů**. A tyto procesy jsou již součástí rutinních procesů, narozdíl od projektů je jejich charakter permanentní a opakovaný²². Patří sem zejména procesy plánování a koordinace projektů (konkrétně Vytvoření plánu vývoje IS/I na další období a Koordinace projektů IS/IT). Základním nástrojem ošetření rozhraní mezi rutinními procesy řízení IS/IT a projekty jeho vývoje, ležícím přesně mezi těmito dvěma obecně neslučitelnými problémovými oblastmi, je základní klíčový dokument řízení projektu – **Projektový záměr**, u běžících projektů pak jeho derivát - Projektová změna.

²² zatímco k základním obecným charakteristikám projektu patří jeho neopakovatelnost

Sledující vazby mezi jednotlivými naznačenými druhy procesů na obrázku, můžeme pozorovat základní zpětnou vazbu od informačního systému organizace k jejím business procesům. Pokud informační systém organizace dokonale podporuje potřeby procesů, projevují se všechny potřeby změn v business procesech (lhostejno zda ve významu problémů a nedostatků, nebo záměrů, či podnětů ke zlepšení) i nějakým způsobem v jejím informačním systému. Buď jsou tyto potřeby vnímány přímo jako problémy s informačním systémem (například neposkytuje všechny potřebné informace pro podporu business procesu), anebo je jejich vazba na informační systém zřejmá (například potřeba vnímat data v jiné – doposud nepodporované struktuře a souvislostech). Tak již v samotných business procesech organizace vznikají prvotní podněty ke změnám informačního systému. Tyto podněty se vztahují buď jenom k informačnímu systému (ty pak mají zpravidla charakter hlášení problémů, nedostatků IS/IT), anebo, a to velmi často, se za nimi ve skutečnosti skrývá mnohem hlubší problém / potřeba změny ve věcném systému business procesů a/nebo řízení organizace. Proto jak v oblasti procesů řízení organizace, tak v oblasti procesů řízení jejího IS/IT, musí existovat procesy, zajišťující základní vyhodnocení podstaty každého takového problému a jeho adekvátního řešení (tedy postoupení na správné místo organizační struktury, odpovídající jeho podstatě). Na straně procesů řízení organizace je to proces Hodnocení podnětů uživatelské sféry, na straně procesů řízení jejího IS/IT je to proces, v obrázku nazvaný Help Desk. Help Desk tak, po vyhodnocení podstaty problému, buď zpracuje příslušný podnět v rámci řízení IS/IT, nebo jej předá k řešení do oblasti procesů řízení organizace, podobně ze strany procesů řízení organizace může přirozeně vzniknout požadavek na změnu IS/IT s cílem podpory změny v řízení organizace, nebo jejích business procesech. Pro řízení provozu a kontinuálního vývoje IS/IT je pak zapotřebí veškeré podněty ke změnám IS/IT analyzovat z hlediska jejich vzájemných souvislostí, aby mohly být příslušně dávkovány a promítnuty do plánu vývoje IS/IT na další období v podobě naplánovaných projektů (viz proces Vytvoření plánu dalšího vývoje IS/IT).

D.5.1 Role požadavků na IS

V souvislosti s procesem Help Desk, jmenovaným v předchozím odstavci, je třeba upozornit na souvislost s fenoménem tzv. „uživatelských požadavků na IS“. Tato problematika je samostatně rozebírána v kapitole A6. Tam uvedená klasifikace požadavků je používána jako základní nástroj *analýzy požadavků* s cílem specifikace projektových záměrů pro budoucí vývoj systému. V tomto smyslu je analýza požadavků také důležitým prvkem *zpětné vazby mezi provozem informačního systému a jeho vývojem* (vývojovými změnami). Objektivní potřeba změn a nových vývojových verzí informačního systému je signalizována právě ve formě požadavků na něj, a to zejména od jeho uživatelů. Proto je také v provozu informačního systému kladen značný důraz na realizaci uživatelského střediska (Help Desk), které je v tomto smyslu základním bodem signalizace objektivní potřeby změny IS. Uživatelské středisko tak má důležitou roli prvotního analyzátora požadavků na IS. Na něm leží hlavní odpovědnost za to, že žádný takový signál nezůstane nevyslyšen i za to, že u požadavků, signalizujících stejnou vlastnost reálného systému, bude identifikován stejný původ a nebudou řešeny jako různé (což poskytuje příležitost k nekonzistencím ve vývoji IS). Mimo jiné plynou z výše uvedeného mimořádné požadavky na kvalifikaci personálního obsazení uživatelského střediska: pracovník uživatelského střediska musí mít v tomto smyslu takové analytické schopnosti, aby byl schopen odpovědně odlišit subjektivní požadavky, či omyly a nepochopení uživatelů od signálů o objektivní potřebě změny v informačním systému, nebo dokonce v reálném systému (požadavek na IS může také signalizovat nedostatky v řízení organizace, či její strategii).

Z výše uvedeného je zřejmá *klíčová úloha procesů hodnocení podnětů uživatelů* nejenom pro vývoj IS/IT, ale pro řízení vývoje organizace jako celku (resp. vývoje jeho business

procesů). Mimo jiné to dokazuje kritickou důležitost procesů Help Desk a potažmo potřebu špičkové kvalifikace jeho personálního obsazení. Jak už bylo zmíněno výše, pracovníci v těchto procesech musí dokonale rozumět podstatě businessu a současně být i zdatní v technických aspektech jeho informatické podpory, musí být schopni řešit technické detaily a současně tvořit rozhodnutí o změnách procesů a vidět jejich důsledky v oblasti strategického řízení organizace. V procesně řízené organizaci se pak Help Desk přímo stává centrem realizace vývoje systému business procesů prostřednictvím jeho IS/IT²³.

Pakliže tato vazba funguje, zpětnovazebné působení bude udržovat business procesní model v „živém“, tj. aktuálním stavu. Aktuální stav procesního modelu je logickým předpokladem aktuálního IS, tj. IS, jehož funkce odpovídají požadavkům na IS a pozitivně působí na dosahování strategických cílů organizace.

²³ K základním principům procesního reengineeringu organizace (jako cestě k přechodu organizace na procesní řízení) totiž patří i klíčová úloha IS/IT. Informační systém je jednak nástrojem realizace procesní změny, jednak samotný vývoj IT (a jejich využití v IS) poskytuje důležité podněty ke změnám v procesech. Kromě toho je procesně řízená organizace charakteristická nutnou permanentní proměnlivostí svých procesů a ta je nemožná bez delegování řídicí pravomoci z tradiční úrovně strategického řízení do koncepčního řízení jednotlivých procesů, kde IS/IT zaujímá, již výše zmiňovanou, klíčovou úlohu. Podtrženo a sečteno to znamená, že značná část řízení vývoje procesů je obsažena v práci s podněty uživatelů IS/IT.

Poznámka k užití pojmů „požadavek“ a „potřeba“

V tomto textu používáme tohoto vymezení pojmů, které odpovídají potřebě odlišit potřeby a požadavky IS. Potřeby a požadavky jsou zřetelně vymezeny a v metodice se s nimi pracuje zvlášť. **Termíny požadavek a potřeba tedy není možné libovolně zaměňovat.**

Potřeba (*Need*) = objektivní potřeba určité funkcionality IS, která je analytiky odvozena výhradně z modelů reality.

Požadavek (*Requirement*) = požadavek uživatele na funkcionalitu navrhovaného IS, získaný v průběhu analýzy (např. pomocí metody dotazníku, interview, etc.) nebo po zavedení IS do provozu (helpdesk). Jeho vznik může být determinován subjektivním přáním uživatele, snahou usnadnit si práci, bez vztahu k ostatním částem systému, což může vést ke zkreslení. Při zjišťování potřeb systému je nutné odlišit expertní požadavky od těch, které jsou determinovány pouze individuálním přínosem. Expertní požadavky mohou mít dopad na úpravu modelů reality a tím pozitivně ovlivnit celý systém, zatímco ostatní požadavky mohou mít pozitivní dopad pouze na funkce, užívané individuem.

Díl E Příklad použití metodiky v prostředí Power Designer

Tento díl popisuje použití metodiky na konkrétním příkladu. Většina použitých metod a technik je realizována v prostředí Power Designer. Pro úplnost je třeba dodat, že ne všechny modely a techniky, použité v příkladu, jsou v tomto prostředí podporovány. Jedná se konkrétně o metodu Business System Planning (BSP), použitou ve fázi globální analýzy, u které, vzhledem k její povaze, jakož i k odlišným povahám a zaměření globální a detailní analýzy, ani nemá příliš smysl požadovat její podporu v tomtéž prostředí, které slouží detailní analýze, designu a dalším navazujícím aktivitám v postupu vývoje informačního systému.

Příklad tedy není zaměřen na demonstraci možností nástroje, ale na kompletní ukázkou metodikou daného postupu od základních výchozích rozhodnutí strategické povahy přes jejich rozpracování v podnikových procesech a dalších náležitostech firemního systému, integrovanou koncepci informační podpory procesů a aplikace informačních technologií v nich, až po přechod k jejich realizaci v daném prostředí konkrétní zvolené technologie.

Podrobněji je pak rozepsána zejména část detailní analýzy a návrhu systému, kde je nástrojová podpora jednotlivých analytických metod a technik obecně nejsilnější a nejefektivnější.

E.1 Popis business systému – základního východiska návrhu IS

E.1.1 Předmět podnikání a základní koncept imaginární firmy

Nově vzniklá společnost na trhu estetické chirurgie „Plastická chirurgie, s.r.o.“ se chce ve své podnikatelské činnosti věnovat poskytování služeb estetické chirurgie zahraničním klientům, kteří budou za tímto účelem dopraveni do ČR. Tato koncepce vychází z toho, že ceny těchto zákroků jsou v České republice několikanásobně nižší než např. ve Spojených státech nebo Velké Británii. Oproti ostatním službám, které lze tímto způsobem samozřejmě nabízet i v mnoha jiných oborech, má plastická chirurgie jednu velkou výhodu – ve světě a mezi odbornou má velmi dobrou pověst díky zručnosti českých lékařů a dlouhé tradici. Díky tomu lze vybudovat firmu s image nejen levného, ale především kvalitního poskytovatele služeb.

Předpokládaným segmentem klientely budou občané západních států (zejména jde o oblast USA, VB a západní Evropy) a občasné bývalého Sovětského svazu. V těchto zemích je poptávka po estetické chirurgii velmi vysoká a zároveň jsou zde velmi vysoké poplatky za prováděné zákroky v této oblasti.

V České republice existuje několik podobných zařízení, která se specializují na tuto oblast chirurgie. I tato zařízení jsou z části zaměřena na zahraniční klientelu. Žádné z těchto zařízení není založeno na přímém získávání klientů ze zahraničí prostřednictvím partnerských lékařů a žádné z těchto zařízení neposkytuje služby formou „dovolené“. Kromě těchto center estetické chirurgie existují také jednotky plastické chirurgie, při velkých nemocnicích. Tyto jednotky nejsou zaměřeny na výdělečnou činnost poskytováním služeb estetické chirurgie.

Všeobecně existuje přetlak poptávky nad nabídkou v této oblasti chirurgie na území ČR.

E.1.2 Podniková vize

„Uspokojovat zákazníky toužící po změně svého vzhledu, bez ohledu na vzdálenost mezi státem kde žije a Českou republikou. Přes tyto spokojené „zákazníky“ se bude snažit

dále pronikat do povědomí dalších a dalších potenciálních klientů s cílem zlepšit a zvětšit rozsah svých poskytovaných služeb.“

E.1.3 Strategie k naplnění vize

K naplnění vize, chce společnost využívat moderního IS, pomocí kterého bude schopen přijímat klienty z celého světa bez ohledu na vzdálenost mezi zemí kde klient žije a Českou republikou.

Dosažení své vize je plánováno ve dvou krocích v závislosti, jak se bude vyvíjet poptávka po službách estetické chirurgie ve společnosti Plastická chirurgie, s.r.o.

Fáze jsou následující:

V **první** fázi bude celý projekt provozován ve spolupráci se zvolenou nemocnicí, která za úplatu poskytne prostory (operační sály + možnost rekonvalescence). V případě úspěšného chodu společnosti postoupí projekt v časovém horizontu cca 2 let do druhé fáze, kterou bude vybudování samostatného sanatoria za Prahou. Toto uspořádání má několik výhod.

- v případě neúspěšného chodu projektu minimální finanční ztráty
- získání zkušeností a zaškolení vlastního personálu
- možnost vytvoření osobních kontaktů a případné „přetáhnutí“ vytipovaných jedinců do vlastních řad
- lepší podmínky pro poskytnutí úvěru (nebo vstup investora) na výstavbu sanatoria po prokazatelné 2 roky trvající úspěšné činnosti

V **druhé** fázi přijde na řadu výstavba sanatoria a postupné stěhování provozu do jeho prostor. To bude obsahovat kromě operačních sálů a zázemí pro lékařský a zdravotnický personál také ubytovací kapacity pro pacienty jak před zákrokem, tak pro následnou rekonvalescenci (pokud bude potřeba). I v této fázi ovšem bude pokračovat spolupráce s partnerskou klinikou – pro potřeby plastické chirurgie totiž není potřeba žádné speciální diagnostické a monitorovací vybavení. V některých výjimečných případech (např. při mimořádných komplikacích) však může být jeho přítomnost nezbytná. Proto je například potřeba zajistit spolupráci s ARO (anesteziologicko – resuscitační oddělení) nebo JIP (jednotka intenzivní péče), které používají mimořádně nákladné přístroje.

Kromě této spolupráce bude ještě nutné zajistit souhru se zahraničními subjekty. Ty budou kliniku zastupovat ve vybraných zemích a kromě propagace a zajištění klientely také zajistí základní konzultace a předoperační vyšetření (základní úkony jako krevní tlak, EKG, zjištění alergie atd.). Z toho vyplývá, že se bud jednat o zdravotnická zařízení, která sama tyto služby neposkytují a za zákazníky budou mít zajištěnou provizi (typicky půjde o privátní praktické lékaře).

E.1.4 Popis činnosti

Core business společnosti je, jak již bylo uvedeno ve vizi a strategii, poskytování služeb estetické chirurgie. Zákazníci budou přicházet do kontaktu se společností přes kontaktní lékaře. Tito lékaři budou provádět všechna potřebná vyšetření a ta budou zasílána do centra v České republice. Je tedy nutný robustní informační systém, protože po několika letech půjde o stovky až tisíce klientů. S každým klientem je spojené obrovské množství dat v elektronické podobě, jako je například EKG, rentgeny, sono atd. Pro společnost je podstatný bezpečný, spolehlivý a rychlý přenos dat od partnerských lékařů do centra v ČR. Systém bude vyžadovat velkou flexibilitu jednak co do rozsahu, tak také do možnosti rozšíření o další předměty podnikání. Také je třeba vzít v úvahu, že musí být koncipován s možností fungovat mnoho let.

Celý proces začíná vyjádřením zájmu zákazníka na pobočce našeho partnerského zařízení (partnerského lékaře). Tomuto předchází úkony, které závisí na konkrétní situaci a řešení těchto úkonů není součástí požadavku investora. Proces Za počátek procesu tak považujeme až příchod klienta na pobočku s konkrétním přáním.

V partnerském zařízení kdekoliv ve světě je klientovi vytvořena předběžná nabídka a to na základě představ, které na této pobočce vyjádří. Takto vytvořená nabídka se předkládá klientovi a současně se ukládá do databáze. Do databáze se ukládají všechny nabídky a to i v případě, pokud nejsou klienty přijaty. Na této úrovni není řešena situace, kdy bude zákazník do partnerského zařízení volat. Způsob reakce na tuto situaci bude záviset na každém partnerském zařízení. Pokud bude ochotno vytvořit nabídku pouze po telefonu, je toto možné.

Na nabídku může klient reagovat dvěma způsoby. Nabídku vytvořenou na základě jím předložených požadavků může odmítnout, v tomto případě je proces ukončen. Může také nabídku přijmout a v tomto případě je s klientem vytvořena smlouva o provedení předoperačních vyšetření. Tato smlouva je opět ukládána do databáze. Smlouvu může zákazník nepodepsat, v tomto případě je proces ukončen. V případě podpisu smlouvy jsou zahájena předoperační vyšetření. Výsledky předoperačních vyšetření se ukládají do databáze stejně, jako tomu je u všech výše uvedených případů.

Výsledek předoperačního vyšetření může být:

- nevyhovující – s klientem je ukončena spolupráce v daném požadavku. V případě jiného typu požadavku může podmínky splnit.
- vyhovující – s klientem je upřesněna nabídka.

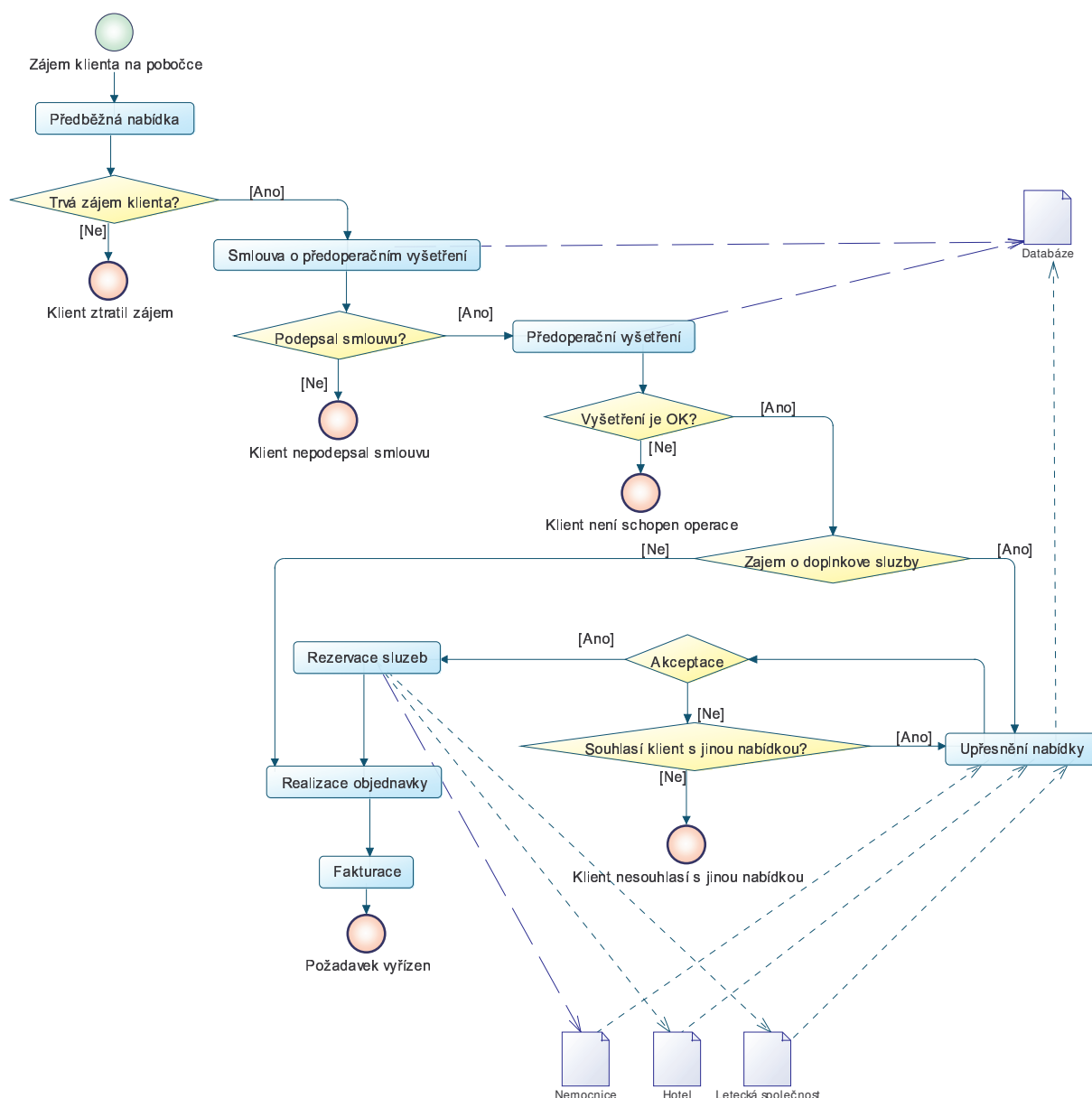
Po vytvoření nové (respektive upravení původní) nabídky dochází k její akceptaci. Pokud klient nabídku neakceptuje, může mu být vytvořena jiná nabídka. Pokud nesouhlasí s další nabídkou (respektive jejím opakováním v různých podobách) je proces ukončen. Pokud s nabídkou souhlasí (akceptace je pozitivní) – platí jak v případě upravované nabídky tak i v případě první nabídky, je kontaktována příslušná nemocnice, kde bude zákrok proveden. U upřesnění nabídky je podstatné, že jsou zahrnuty do okolností i kontextové faktory, kterými jsou letecké společnosti a hotelová zařízení, kde bude klient ubytován. Tyto faktory mohou způsobit, že je nabídka odmítnuta a je požadováno její upravení. Nemusí například vyhovovat termín provedení operace, kvalita cílového hotelu nebo způsob dopravy (moc přestupů, pouze druhá třída atd.). Společnost Plastická chirurgie, s.r.o. si tato potřebná data sama stahuje z provozních systému spolupracujících společností (hotely, letecké společnosti, nemocnice).

Po akceptaci nabídky je informována nemocnice, hotel a letecká společnost a jsou zablokovány (objednány) příslušné termíny dohodnuté v objednávce. Tímto je proces ukončen a dochází k samotnému příletu zákazníka, jeho ubytování, provedení operace a rekonvalescence. Toto již není cílem tohoto dokumentu. Přílet a ubytování zákazníka bylo vyřešeno zablokováním příslušných termínů. Operace je zajištěna zablokováním příslušného termínu v partnerském lékařském zařízení. Rekonvalescence a kulturní vyžití závisí na volbě příslušného hotelu a místa, kde se bude operace provádět.

Procesy spojené s kulturním vyžitím zde nejsou řešeny. Toto je spojeno s konkrétním hotelem a nemocničním zařízením.

Zde uvedený procesní model je jen velice hrubým nástinem celého procesu, téměř každý proces může být rozepsán na nižší úrovni. Slouží spíše jako formální zobrazení procesu vyřízení požadavku klienta.

Následující obrázek je grafickým vyjádřením procesu:



Obrázek E 1 Model business procesů (činností)

Důležité pro datový model IS bude, aby umožňoval (resp. obsahoval data), které bude možné transportovat do DWH. Součástí projektu může tedy být i multidimenzionální analýza (vytížení doktorů (zejména v druhé fázi realizace – vlastní lékaři), rentabilita jednotlivých činností, zákazníků, analýza které činnosti jsou nejvíce žádány, jaké zboží se nejvíce prodává apod.). V datovém modelu musí být informace ve struktuře, která bude umožňovat ETL do DWH.

E.1.5 Specifikace Informačního systému:

Zde uvedená specifikace informačního systému je specifikací hrubou a bude doladěována v oboustranném dialogu mezi vedoucím projektu a sponzorem projektu.

E.1.5.1 Aplikační architektura

Funkcionalitu budoucího IS by měly zabezpečit následující aplikace.

- SW pro řízení kontaktního centra – software pro řízení vztahů mezi jednotlivými partnerskými organizacemi a okamžité rozpoznání klientů a poskytování aktuálních informací o daném klientovi,
- systém na zakázku (jádro projektu) – evidence klientů, zdravotního stavu, partnerských organizací, (všechny činnosti jsou uvedeny v popisu businessu)

E.1.5.2 Funkční architektura

IS by měl zabezpečit následující funkce:

- uchování veškerých informací o klientech, včetně zdravotního stavu, počtu plastických operací konaných v našem zařízení i v jiných zařízeních, požadovaný standard na kvalitu související péče, atd.,
- řízení lidských a ostatních zdrojů – optimalizace počtu lékařů (zejména v druhé fázi) na základě údajů z DB klientů,
- funkcionalita contact centra – identifikace klienta, zobrazení údajů o klientovy, přístup ke znalostní databázi
- EIS – hlavně nástroje pro efektivní řízení zdrojů (opt. počet zdravotnického personálu ve správné struktuře), dále také analýza ekonomických ukazatelů podle různých dimenzí, analýza jednotlivých druhů požadovaných operací atd.

E.1.5.3 Procesní architektura

- viz popis business činností a z toho pramenící procesy (může být postupně konzultovány s týmem)

E.1.5.4 Datová architektura

- musí obsahovat požadavky stanovené v popisu business činností. Požadováno je RDBMS.

E.1.5.5 Organizační architektura

- Důležitou součástí cílů informačního systému je redukce nákladů na lidské zdroje. Tzn. je nutná optimalizace lidských zdrojů k počtu prováděných operací. Z toho důvodu je nutné držet pro každého lékaře individuální kalendář, pomocí kterého bude docházet k optimalizaci. Z důvodu efektivního řízení lidských zdrojů je také vybudování DWH. Efektivní řízení lidských zdrojů pak bude prováděno na základě OLAP analýz.
- organizační architektura je velice relativně plochá. Každý lékařský tým má vedoucího. Nad jednotlivými lékaři stojí primář, který zastřešuje veškeré lékařské činnosti. Nad sestrami stojí vrchní sestra, která v rozhodovací pravomoci podléhá primáři.
- Součástí organizační architektury jsou ředitel společnosti Plastická chirurgie, s.r.o., jeho poradci a samozřejmě vedoucí a členové jednotlivých činností, které jsou s provozem společnosti spojeny (účetnictví, marketing, atd.) jak bylo specifikováno ve speciálním dokumentu (Globální podniková analýza a požadavek investora).

E.2 Globální analýza systému

Globální analýza a návrh IS představuje úvodní nástin budoucího řešení IS. Zabývá se analýzou předmětné oblasti z obecného hlediska. Jejím cílem v žádném případě není provádět výběr finálně nasazených produktů. Jejím cílem je naopak analyzovat všechny možné aspekty konkrétního projektu a to především: vnitřní požadavky organizace, požadavky na vnější subjekty, požadavky vnějších subjektů, možné architektury řešení IS, apod.

E.2.1 Analýza BSP

BSP analýza slouží k ověření souladu podnikových strategií, definovaných procesů a navržené organizační struktury. Jejími základními vstupy byly zadavatelem dodané dokumenty Podnikatelský záměr a SWOT analýza. Dalšími dokumenty, z kterých analýza vychází, jsou Globální strategie, Informační strategie a zadavatelský projekt. Některé detaily byly konzultovány přímo se sponzorem. Kromě zajištění souladu strategií, procesů a organizační struktury byly s pomocí techniky informačního kříže specifikovány základní komponenty navrhovaného informačního systému.

Cílem je popis procesů probíhajících ve firmě s důrazem na vymezení subsystémů s ohledem na splnění strategických cílů firmy.

Metoda BSP je využita pouze částečně a to jako podklad pro další vývoj IS. Jsou použity ty části a do takové hloubky, aby postačovaly ke splnění cílů, které tato část analýzy má poskytnout pro další práce na vývoji systému.

E.2.1.1 Dimenze matic

E.2.1.1.1 Strategie

V souladu s globální strategií firmy byly stanoveny tyto krátkodobé i dlouhodobé strategické cíle organizace s vlivem na IS organizace.

- levný poskytovatel služeb pro zahraniční klientelu,
- kvalitní poskytovatel služeb,
- kvalitní smluvní vztahy s partnery (nemocnice, pojišťovny, klienti, cestovní kanceláře), které zajistí dosažení cílů,
- řízení zkušeností (znalostí) zaměstnanců,
- spolehlivost a bezpečnost systému.

Strategie kvalitní a levný poskytovatel služeb pro zahraniční klientelu vychází přímo ze zadání sponzora, který na těchto dvou aspektech staví svůj podnikatelský záměr a svoji konkurenční výhodu.

Strategie smluvní vztahy je pojata velmi komplexně a široce. Spadají pod ni veškeré vztahy firmy s okolními subjekty. Partnerské nemocnice v zahraničí, nemocnice v Čechách pronajímající svoje kapacity za úplaty, pojišťovny, cestovní kanceláře, letecké společnosti, hotely atd.

Návrhem řešitele projektu bylo zahrnutí do strategií rovněž minimalizace ztrát, jež by představovala „zadní vrátka“ v případě neúspěchu celého projektu, tedy o směřování veškerého podnikání k minimalizaci jakýchkoli ztrát. Je třeba si uvědomit, že tato strategie, i když to může svádět k opačnému názoru, je odlišná od strategie levný poskytovatel služeb. Strategie číslo 1 znamená levný produkt z hlediska trhu (tedy pokud budeme nabízet na zahraničním trhu – levný produkt pro daný zahraniční trh). Kdežto minimalizace ztrát

znamená ošetření, vyvarování se a řízení veškerých případných krizových situací, možných finančních nesouladů apod. Tento námi navrhovaný strategický cíl byl odmítnut, a proto není do analýzy BSP zahrnut.

Strategie řízení zkušeností (znaností) zaměstnanců může být brána analogicky k tradičním vzdělávacím programům, které probíhají prakticky v každé společnosti. Zde je míněno jak vzdělávání a zvyšování kvalifikace našich pracovníků (především lékařů – specialistů), tak monitoring trhu (opět především trhu lékařů – specialistů) a případné přetažení perspektivních odborníků do naší firmy.

Velké množství dat procházejících vytvářeným systémem jsou data velmi citlivého charakteru, a z tohoto důvodu je stoprocentní spolehlivost, bezpečnost a stabilita systému nezbytnou podmínkou.

.E.2.1.1.2 Procesy

Za základní procesy, které výrazně ovlivňují chod firmy považujeme tyto:

- získávání a uzavírání kontraktů,
- registrace nového zákazníka,
- rezervace operačního zařízení,
- diář specialistů (řízení HR),
- rezervace doprovodných služeb,
- fakturace,
- evidence a management hmotných zdrojů.

Proces získávání a uzavírání kontraktů obsahuje, jak je již z jeho názvu patrné, několik dílčích podprocesů. Proces získávání kontraktů je procesem při němž jsou vyhledávání a získávání jak klienti/zákazníci tak dodavatelé externích služeb, nebo partneři pro outsourcing. Tyto procesy zahrnují také například účast na kongresech týkajících se plastické a estetické chirurgie.

Druhý proces představuje přijetí poptávky od klienta, zaslání nabídky klientovi, registraci zákazníka a objednávku plastické operace.

Další proces představuje komunikaci a zajištění vlastní rezervace operačního sálu, případně dalšího lékařského zázemí potřebného pro výkon vlastní plastické operace u externího partnera se kterým již bylo dopředu nasmlouván pronájem operačních sálů a zařízení.

Diář specialistů úzce souvisí s funkcí řízení lidských zdrojů. Na základě přijaté objednávky na plastickou operaci je zanesen záznam do diáře specialistů. Ten obsahuje rozvrhy všech pracovníků nutných pro plynulé provedení objednávky – tzn. lékařů, sester apod.

Již v procesu registrace nového zákazníka je klientovi nabídnuta řada doprovodných služeb (návštěva kulturních akcí, výlety, sportovní vyžití atd.). Podle objednávky zadané klientem jsou tyto doprovodné služby objednány u externích partnerů.

Proces fakturace zahrnuje nejen vyfakturování provedené plastické operace, ale i doprovodných služeb. Tento proces dále zahrnuje přijetí platby a její kontrolu od klienta a případné předání lékařské dokumentace týkající se provedeného lékařského zákroku

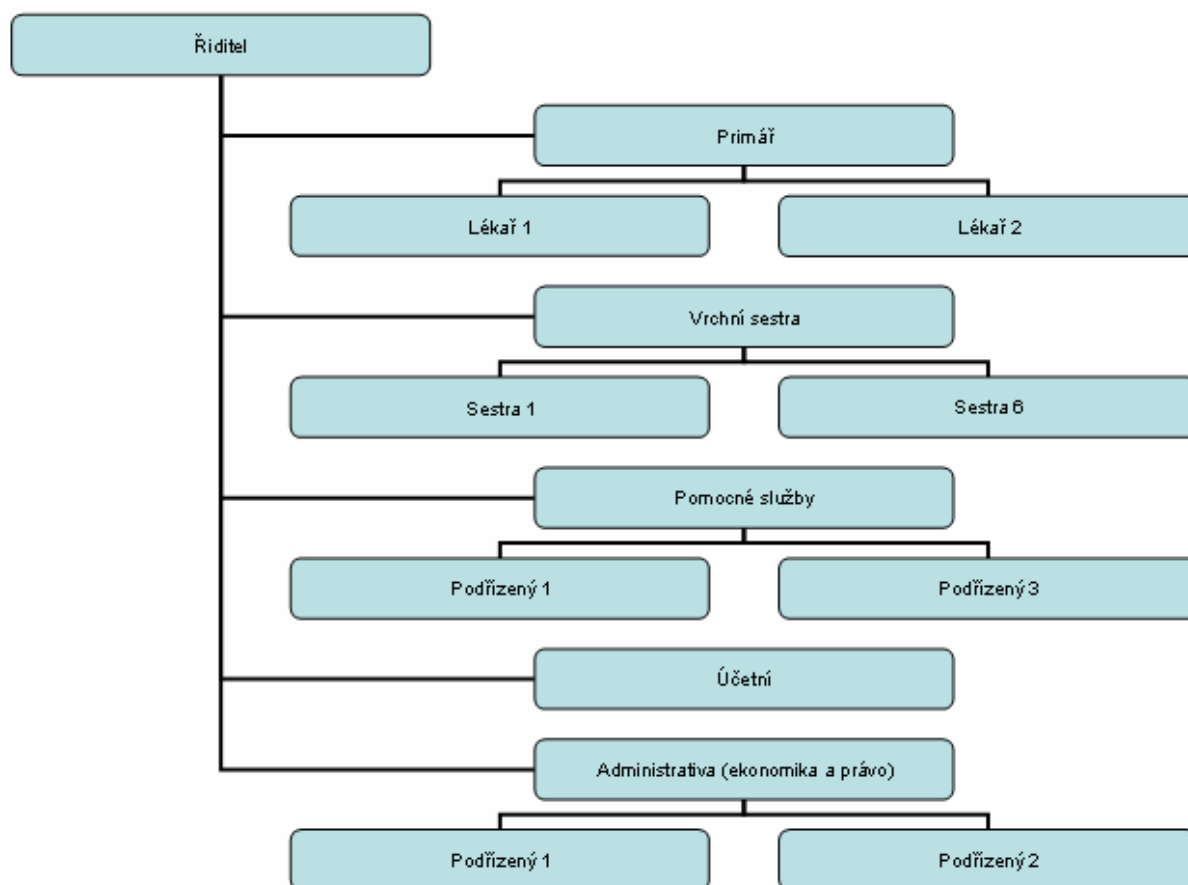
Evidence a management hmotných zdrojů, představují řízení a evidenci veškerého majetku, který společnost vlastní nebo který využívá na základě pronájmů. Součástí je i rozhodování o možných pronájmech zdrojů v případě, že kapacity nejsou plně využity nebo vyjádření doporučení k nákupu (pronájmu) nových kapacit, pokud stávající kapacity jsou nedostatečné.

.E.2.1.1.3 Organizace

Z obchodní strategie (globální strategie) společnosti Plastická chirurgie, s.r.o. vyplynula v první fázi návrhu následující relativně plochá organizační struktura. Z grafu č. 1 je vidět, že řediteli je podřízeno pět vedoucích a každý má přiřazen určitý počet podřízených. Počet podřízených pro každého z vedoucích bude odvozen podle aktuální velikosti kliniky. Předpokládané počty zaměstnanců pro jednotlivé velikosti jsou následující:

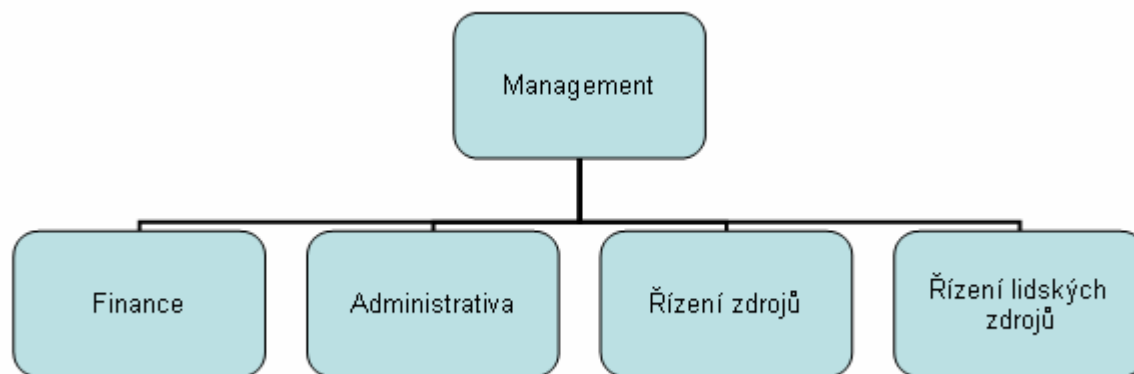
- malá klinika = 2 lékaři, 6 sester, 3 zaměstnanců pomocného personálu, účetní, 2 administrativní zaměstnanci (právní a ekonomické záležitosti),
- střední klinika = 5 lékařů, 10 sester, 4 zaměstnanci pomocného personálu, účetní, 3 administrativní zaměstnanci (právní a ekonomické záležitosti),
- velká klinika = 10 lékařů, 15 sester, 6 zaměstnanců pomocného personálu, 2 účetní, 5 administrativních pracovníků (právní a ekonomické záležitosti).

Společnost začíná jako malá klinika, která by se za optimálních podmínek měla rozvinout do podoby velké kliniky a s tím souvisejícím počtem zaměstnanců. Změny v grafu spočívají pouze v přidávání podřízených jednotlivým vedoucím.



Obrázek E 2 Organizační struktura společnosti Plastická chirurgie, s.r.o.

V následujícím grafu je vytvořená organizační struktura tak, jak byla dohodnuta s investorem projektu. Cílem této úpravy bylo rozpracování původního podnikatelského záměru, který je pouze obecným cílem, nikoliv však cílem finálním. Z tohoto důvodu byla provedena tato úprava, která lépe vyjadřuje potřeby řízení společnosti Plastická chirurgie, s.r.o.



Obrázek E 3 Organizační struktura společnosti Plastická chirurgie, s.r.o upravená po dohodě se sponzorem pro potřeby BSP analýzy

Z upravené organizační struktury firmy uvedené v obrázku č.2 vyplývají tyto organizační jednotky:

Management – úkolem Managementu je řídit lidi, optimalizovat procesy pro zákazníka a komunikovat s dodavateli.

Finance – oddělení má na starosti účetnictví, čímž se dostává do styku s daty zákazníků a fakturuje jim údaje ze zakázky. Dále přijímá faktury jiných dodavatelů.

Řízení zdrojů – oddělení Řízení zdrojů komunikuje s dodavateli materiálu a jiných potřeb, součástí je samotná péče o majetek ve vlastnictví společnosti, zajištění konzistence se zákony, vyhledávání partnerských organizací, atd.

Řízení lidských zdrojů – Řízení lidských zdrojů se stará o vlastní zaměstnance.

Administrativa – oddělení Administrativa zajišťuje běžné administrativní úkony, součástí administrativy jsou záležitosti právní povahy a samozřejmě také záležitosti ekonomického charakteru.

.E.2.1.1.4 Charakteristiky rolí

Na úrovni GAN jde pouze o vyjmenování základních činností (funkcí), které budou jednotlivé role vykonávat. Nejde o úplný a přesný seznam činností jimi vykonávaných.

Management:

- definuje cíle a strategie společnosti,
- uzavírá smlouvy s hotely, cestovními kancelářemi,
- stanovuje konečné ceny,
- vypracovává analýzy vytížení pronajatých kapacit a využití lékařského personálu,
- kontroluje hospodaření společnosti.

Finance:

- zajišťuje vyúčtování s externími subjekty (platební styk),
- eviduje faktury zákazníkům,
- eviduje pohledávky a závazky,
- zajišťuje vedení účetnictví a mzdy.

Administrativa

- zajišťuje běžné administrativní úkony,
- součástí administrativy jsou záležitosti právní povahy (právník),
- součástí jsou záležitosti ekonomického charakteru (ekonom).

Řízení zdrojů

- v první fázi minoritní – firma využívá zejména pronajaté prostory,
- v dalších fázích standardní evidence veškerého majetku (movitého, nemovitého),
- zajišťuje soulad softwarových systémů s příslušnými předpisy (autorská práva u SW),
- dává návrhy na možnosti případného pronajmutí přebytkového majetku (budovy, vybavení atd.),
- zajišťuje evidenci zdrojů u partnerských organizací a partnerských lékařů,
- zajišťuje vyhledávání potenciálních nových partnerských organizací či lékařů,
- vytváří reporty o vytížení zdrojů pro management,
- definuje pravidla pro komunikaci partnerských organizací s klienty a realizují požadované činnosti tak, jak je specifikováno v dalších kapitolách.

Řízení lidských zdrojů

- minoritní část,
- zahrnuje řízení lékařského personálu,
- realizuje vyhledávání a nábor lékařského personálu potřebné kvalifikace,
- zajišťuje péči o zaměstnance,
- řeší případné problémy se zaměstnanci,
- vytváří podklady pro management pro tvorbu reportů.

.E.2.1.1.5 Třídy dat

Základní množiny dat opět vyplývají z globální strategie firmy a informační strategie a dále ze způsobu, kterým bude činnost firmy vykonávána.

Operace data vztahující se k dané operaci (místo konání, čas konání, předpokládání operatéri, atd.)

Faktura daňový doklad odesílaný jednak dodavatelům (s příslušnými specifickými údaji pro ni potřebnými) a odesílaný klientům (rovněž s údaji specifickými pro klienty). Souhrnně jsou obsažena data z jednotlivých faktur (platebních dokladů).

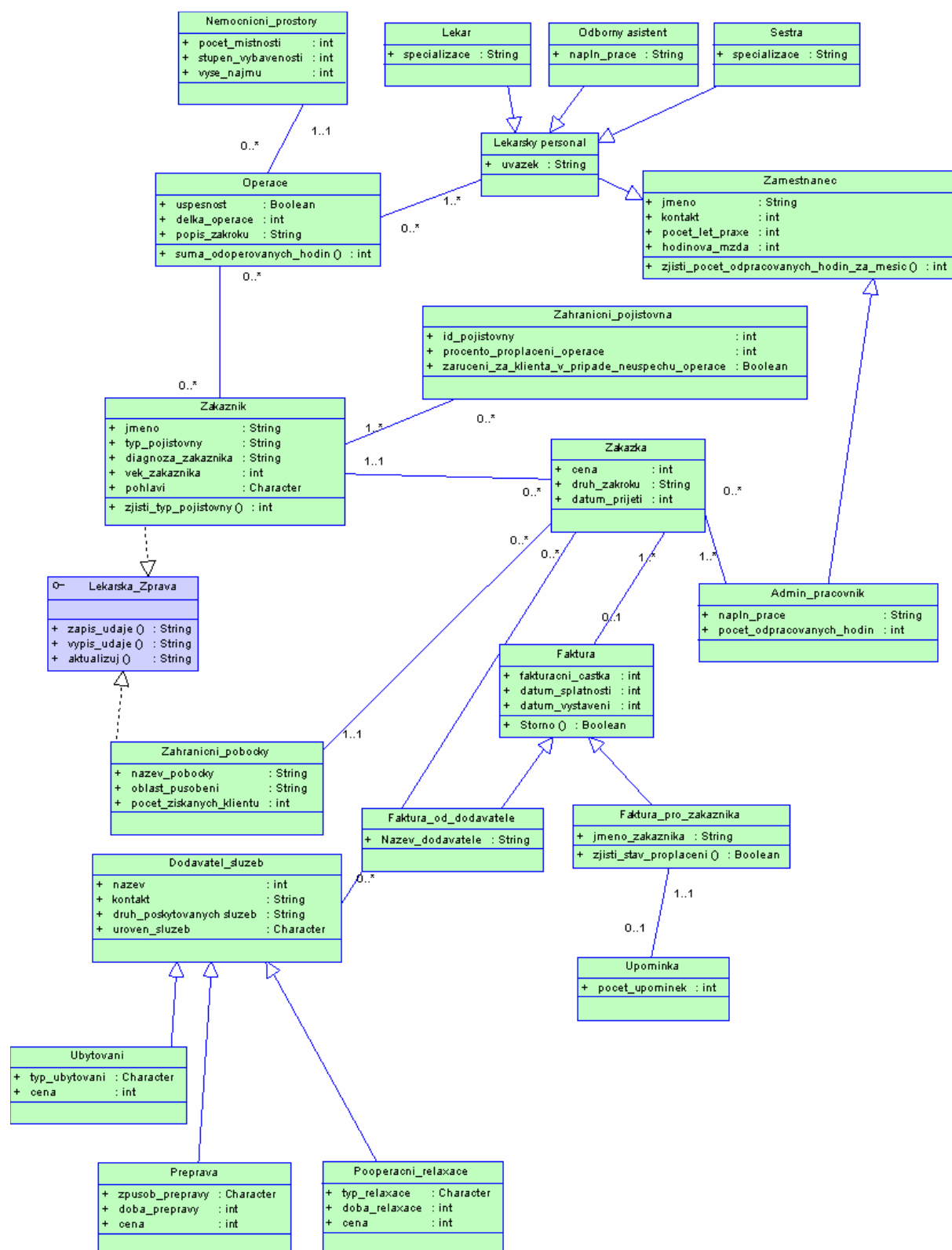
Zákazník data o zákaznících (jméno, adresa, kontaktní údaje...),

Zakázka požadovaný druh estetického chirurgického zákroku,

Pracovník veškeré údaje o zaměstnancích (osobní, kontakty, plat...),

Dodavatel informace o dodavatelích (jméno, adresa, zboží...),

Grafické vyjádření základních tříd dat obsahuje následující class-diagram.



Obrázek E 4 Diagram tříd

E.2.1.2 Pomocné matice

V následující podkapitole jsou uvedeny jednotlivé pomocné matice z nichž je v následující kapitole „E.2.1.3 Informační kříž“ vytvořen informační kříž.

Smyslem každé z matic je vyjádřit, v kterých faktorech se matice protínají.

.E.2.1.2.1 Matice STRATEGIE / PROCESY

Procesy/Strategie	Levný poskytovatel služeb pro zahraniční klientelu	Kvalitní poskytovatel služeb	Kvalitní smluvní vztahy s partnery	Řízení zkušeností (znalostí) zaměstnanců	Spolehlivost a bezpečnost systému
Získávání, uzavírání kontraktů a propagace	X		X		
Registrace nového zákazníka		X			X
Rezervace operačního zařízení			X		
Rezervace doprovodných služeb			X		
Evidence a management hmotných zdrojů			X		
Diář specialistů (řízení HR)				X	
Fakturace					X

Tabulka E 1 Pomocná matice Procesy/strategie

.E.2.1.2.2 Matice PROCESY / TŘÍDY DAT

Procesy/Třídy dat	Pracovník	Zákazník	Zakázka	Operace	Faktura	Dodavatel
Diář specialistů (řízení HR)	X			X		
Rezervace doprovodných služeb		X				X
Získávání, uzavírání kontraktů a propagace		X				
Registrace nového zákazníka		X	X	X		
Rezervace operačního zařízení			X	X		
Fakturace					X	
Evidence a management hmotných zdrojů						X

Tabulka E 2 Pomocná matice Procesy/Třídy dat

.E.2.1.2.3 Matice STRATEGIE / TŘÍDY DAT

Strategie/Třídy dat	Pracovník	Operace	Zákazník	Zákazník	Dodavatel	Faktura
Rizici zkušenosti (znalosti) zaměstnanců	X					
Kvalitní poskytovatel služeb	X	X			X	
Levný poskytovatel služeb pro zahraniční klientelu			X			
Spolehlivost a bezpečnost systému				X		
Kvalitní smluvní vztahy s partnery					X	

Tabulka E 3 Pomocná matice Strategie/Třídy dat

.E.2.1.2.4 Matice PROCESY / ORGANIZAČNÍ JEDNOTKY

Procesy/Organizační jednotky	Finance	Řízení lidských zdrojů	Oddělení správy zdrojů	Administrativa
Fakturace	X			
Diář specialistů (řízení HR)		X		
Rezervace operačního zařízení			X	
Evidence a management hmotných zdrojů			X	
Rezervace doprovodných služeb			X	
Získávání, uzavírání kontraktů a propagace				X
Registrace nového zákazníka				X

Tabulka E 4 Pomocná matice Procesy/Organizace

.E.2.1.2.5 Matice TŘÍDY DAT / ORGANIZAČNÍ JEDNOTKY

Třídy dat/Organizační jednotky		Finance	Řízení lidských zdrojů	Oddělení správy zdrojů	Administrativa
Faktura		X			
Pracovník			X		
Dodavatel				X	X
Operace			X	X	
Zakázka			X	X	X
Zákazník					X

Tabulka E 5 matice Třídy dat/Organizační jednotky

.E.2.1.2.6 Matice STRATEGIE / ORGANIZAČNÍ JEDNOTKY

Strategie/Organizační jednotky		Finance	Řízení lidských zdrojů	Oddělení správy zdrojů	Administrativa
Levný poskytovatel služeb pro zahraniční klientelu		X			
Kvalitní poskytovatel služeb			X		
Řízení zkušeností (znalostí) zaměstnanců			X		
Spolehlivost a bezpečnost systému				X	X
Kvalitní smluvní vztahy s partnery					X

Tabulka E 6 Pomocná matice Strategie/Organizační jednotky

E.2.1.3 Informační kříž

Výše popsaným srovnáním a uspořádáním jednotlivých matic a následnou další analýzou se ustavily čtyři dimenze informačního kříže. Poslední dimenzí je rozpad plánovaného informačního systému do jednotlivých funkčních modulů či subsystémů. Ten bylo třeba provést tak, aby tyto byly co nejlépe v souladu s podnikovými procesy, podnikovou organizační strukturou společnosti a jejími partnery. Předchozí analýza by potom měla zajistit, že tyto čtyři zmíněné dimenze jsou v souladu s dimenzí nejdůležitější a prvotní – totiž s podnikovými strategiemi.

Následující odstavce popisují výsledný rozpad IS na jednotlivé subsystémy a jejich vazby na ostatní dimenze.

X					Získávání, uzavírání kontraktů a propagace					X
X					Registrace nového zákazníka					X
X	X				Rezervace operačního zařízení				X	
	X				Evidence a management hmotných zdrojů				X	
	X				Rezervace doprovodných služeb				X	
		X			Diář specialistů (řízení HR)			X		
			X		Fakturace		X			
Zákaznický IS (koordin. případů) IS pro komunikaci s ext. subjekty Personální IS Finanční IS					<div><div>Činnosti</div><div>IS</div><div>Funkce</div><div>Entity</div></div>					
			X		Faktura		X			
		X			Pracovník			X		
	X				Dodavatel				X	X
	X	X			Operace			X	X	
X	X	X			Zakázka			X	X	X
X					Zákazník					X

Tabulka E 7 Informační kříž

E.2.1.4 Subsystémy IS:

Z předchozí analýzy vyplynuly tyto subsystémy:

- IS pro komunikaci se subjekty a správu vnitřních zdrojů (Business modul),
- Finanční modul,
- Personální modul (Řízení HR),
- Zákaznický IS (Koordinace případů).

E.2.1.5 *Procesy + IS*

Moduly IS, které podle našeho názoru budou nejlépe podporovat procesy organizace. Jsou následující:

Business modul (IS pro komunikaci se subjekty a správu vnitřních zdrojů) – podporuje nejvíce procesů a je také považován za nejrobustnější modul IS. Skrývá v sobě rovněž mohutnou datovou základnu. Tento modul zprostředkovává komunikaci s partnerskými organizacemi, zajišťuje zprostředkování přenosu dat mezi partnerskými organizacemi a centrálou v České republice. Primárním cílem modulu je zpracování dat o zákaznících. Charakteristické pro komunikaci v Business modulu je, že tato komunikace spočívá v komunikaci s externími subjekty (hotely, letecké společnosti, cestovní kanceláře atd.). Součástí modulu je rovněž správa vnitřních zdrojů, které má společnost k dispozici. Podobná komunikace je uskutečňována ve finančním modulu, kde se jedná o komunikaci z finančního pohledu v podobě realizace plateb externím subjektů. Ve finančním modulu je součástí také příjem plateb od externích subjektů, kterými jsou naši klienti.

Modul **Finanční** se podílí na primárních procesech pouze sekundárně a to zprostředkováním plateb za poskytované služby jednak lékařského charakteru, ale také služby za navazující a samozřejmě také předcházející péči. Prostřednictvím finančního modulu je zajištěna včasná realizace plateb dodavatelům tak, abychom se nedostali do případných sporů z důvodů neplacení svých závazků. Cílem je, aby veškeré tyto činnosti probíhaly v co největší míře elektronickou formou, což má tento a také business modul zajistit.

Dalším významným modulem, který rozhodujícím způsobem ovlivňuje chod organizace je modul **Koordinace operačních případů** (zákaznický IS). Jeho hlavním úkolem je zajištění správného a včasného operačního požadavku tak, jak bylo ve smlouvě dohodnuto s klientem. Součástí této činnosti je i následné hodnocení činností, které jsou s tímto v souladu – hodnocení úspěšnosti, správnosti provedených vyšetření atd. Koordinace spočívá v nastavení správné situace mezi partnery, zablokování vhodných termínů pro realizaci požadavků (ať už termínů operace, termínů letů a potřebných lékařských kapacit), které byly dohodnuty s klientem. Tento modul úzce souvisí s Business modulem a vzájemně se doplňují.

Poslední **Personální modul** zpracovává veškeré informace o zaměstnancích a podporuje tak hlavní procesy.

Funkcionalita „kontaktního centra“ tak, jak bylo zmiňované v předcházejících dokumentacích, byla po provedení BSP a zhodnocení všech okolností zahrnuta do modulu „Business modul“ a z tohoto důvodu zde není popisována.

E.2.1.6 *Data + IS*

Data a IS spolu velice úzce souvisí. Každému modulu IS náleží data, se kterými modul pracuje. Data musejí být poskytována včas a v požadované kvalitě tak, aby systém mohl řádně vykonávat své funkce.

U **Business modulu** (IS pro komunikaci se subjekty a správu vnitřních zdrojů) jsou nejdůležitější následující data:

- údaje o klientech,
- požadované operační zákroky klientů,
- poskytované zákroky z naší strany,
- údaje o partnerských organizacích v zahraničí,
- údaje o partnerských lékařích,
- údaje o smluvních zařízeních v nichž je poskytována lékařská péče,

- údaje o dodavatelích (hotely, cestovní kanceláře,...)

Výše bylo zmíněno, že funkcionalita „**Podpora kontaktního centra**“ je součástí modulu „Business modul“. Smyslem funkcionality, kterou by za jistých podmínek mohlo být možné vyčlenit jako samostatný modul, je zabezpečit přijetí, zaevidování požadavku klienta, poskytnutí možností volných termínů pro operace, volných letenek, atd. Tato funkcionalita je zahrnuta do „Business modulu“ a bude umožňovat vkládání nových klientů do databáze, zobrazení dat o zákazníkovi, zaevidování nového požadavku. Hlavními daty budou právě tyto požadavky (detailní informace o každém zákaznickém požadavku).

Sekundárně jsou požadavky zákazníků vedeny také v modulu **Koordinace případů** (zákaznický IS), kde se jednotlivé požadavky hodnotí s hlediska spokojenosti zákazníka a řadí se tak, aby byla dosažena maximální efektivnost.

Finanční modul zpracovává data spojené s platbou za zboží nebo služby.

Finanční modul zabezpečuje především účetnictví a jeho propojení s dokumenty vzniklými mimo samotný modul. Jde hlavně o dodavatelské faktury, odběratelské faktury, bankovní výpisy atd. V matici jsou mu přiřazena data Faktury – tato data jsou však spíše souhrnem všech platebních možností. Tzn., že platba na fakturu není jediným způsobem úhrady zboží či služeb.

Modul **Personální oddělení** je zodpovědný za veškerou zaměstnaneckou agendu + evidenci poboček. Personální modul zabezpečuje veškeré informace pro řízení lidských zdrojů. Obsahuje především databázi zaměstnanců. Jelikož firma potřebuje udržovat i údaje o spokojenosti klientů s jednotlivými zaměstnanci (zejména zdravotními sestrami, s kterými jsou v nejčastějším kontaktu), jsou data o zaměstnancích sekundárně poskytována i modulu **Koordinace případů** (zákaznický IS). Tento modul se tak stává poměrně univerzálním nástrojem pro řízení chodu společnosti v podobě správného a efektivního pořadí operací, evidenci a hodnocení služby a hodnocení zaměstnance, který tuto „službu“ poskytl. Tím se stává velmi vhodným pro to, aby byl centrem znalostního řízení a místem kde jsou spravována odpovídající data.

E.2.1.7 Procesy + Organizace

Součástí původního návrhu struktury organizace byly i **externí pracovníci**. Tito externí zaměstnanci jsou plánováni zejména pro první fázi životního cyklu. Ve fázi druhé je již plánován stabilní personál s ohledem na počet operačních zákroků, které jsou společností Plastická chirurgie, s.r.o. poskytovány. Po dodatečné konzultaci se sponzorem jsme se rozhodli převést činnost těchto subjektů na stálé zaměstnance pod hlavičkou **personální oddělení**. Externí pracovníci měli být využíváni zejména v první fázi životního cyklu, jak bylo uvedeno výše. Externí pracovníci mohou být využíváni také v případech velké jednorázové poptávky po operačních zákrocích, kterou nebude společnost Plastická chirurgie, s.r.o. schopna uspokojit z vlastních zdrojů. Cílem však je dosáhnout stavu, kdy budeme využívat pouze vlastní zkušené a kvalifikované pracovníky, kteří budou plně vytíženi, než najímat externisty u nichž nemusí být zaručena požadovaná kvalita pro námi poskytované služby.

Součástí modulu „**Personální oddělení**“ je samozřejmě hodnocení zaměstnanců tak, jak bylo uvedeno výše, ale toto také navazuje na modul „Koordinace operačních případů“. Hodnocením zde rozumíme hodnocení nejen z pohledu kvality odvedené práce, ale také z pohledů přístupu lékařského personálu ke klientům atd.

Do řídicích procesů jsme ve výsledku přidali Řízení lidských zdrojů (řízení HR), které jsme dali do primární souvislosti s personálním oddělením. Tento proces by měl zajistit dostatek kvalifikovaných pracovníků na příslušné posty.

E.2.1.8 Organizace a Data

V postupu tvorby BSP analýzy jsme dospěli k závěru, že z hlediska firmy není až tak podstatné, kdo danou operaci provede, ale že bude provedena co nejlépe a v souladu s firemní strategií. S touto myšlenou vlastně padly veškeré původní *sekundární* vazby. Konkrétně, kromě již zmíněných zaměstnanců, i *klienti a požadavky*.

O tom, že by měl být primárně vztah mezi zaměstnanci a personálním oddělením není snad pochyb.

Proti původní úvaze jsme zrušili sekundární vztah mezi personálním oddělením a požadavky. Původní idea byla taková, že personální oddělení sleduje na základě požadavků vytíženost zaměstnanců, což ale vyplývá z báze zásahů.

U finančního oddělení jsme vazby omezili primárně na faktury a účetnictví (což bylo i v původní úvaze) a přidali jsme vztah na klienty. Ostatní vazby jsou navázány prostřednictvím jiných částí organizace.

Kontaktní centrum by mělo zajišťovat prvotní styk se zákazníky (prostřednictvím partnerských organizací). Proto jsme se nakonec rozhodli soustředit jeho sílu zejména na zákazníky. Sekundárně pak na sledování požadavků klientů. Fakturace příslušných provedených operací řeší finanční oddělení.

E.2.1.9 Shrnutí BSP

Ve fázi BSP analýzy jsme identifikovali základní podnikové strategické cíle a procesy, s pomocí kterých hodlá zadavatel těchto cílů dosáhnout. Tyto dvě dimenze jsme dali do souvislosti s navrhovanou organizační strukturou podniku. Výsledné dimenzí byly zkombinovány do informačního kříže (viz dále) doplněny o nejvhodnější rozpad IS na jednotlivé subsystémy.

V následujících částech tohoto dokumentu budou výše zmíněné základní procesy detailněji popsány.

Výsledkem další analýzy potom bude návrh konkrétního řešení výše definovaných modulů IS a rovněž návrh technické a komunikační infrastruktury potřebné pro jejich provozování.

E.2.2 Základní procesy

Plný proces před rozdělením na dílčí procesy

Core business společnosti je, jak již bylo uvedeno ve vizi a strategii poskytování služeb estetické chirurgie. Zákazníci budou přicházet do kontaktu se společností přes kontaktní lékaře. Tito lékaři budou provádět všechna potřebná vyšetření a ta budou zasílána do centra v České republice. Je tedy nutný robustní informační systém, protože po několika letech půjde o stovky až tisíce klientů. S každým klientem je spojené obrovské množství dat v elektronické podobě, jako je například EKG, rentgeny, sono atd. Pro společnost je podstatný bezpečný, spolehlivý a rychlý přenos dat od partnerských lékařů do centra v ČR. Systém bude vyžadovat velkou flexibilitu jednak co do rozsahu, tak také do možnosti rozšíření o další

předměty podnikání. Také je třeba vzít v potaz, že musí být koncipován s možností fungovat mnoha let.

V následujícím diagramu je vyjádřen průběh realizace toho procesu. Jednotlivé části procesu budou dále zpodrobněny v dalších částech této kapitoly.

Porovnáme-li první fáze následujícího diagramu zjistíme, že jsou odlišné od plánovaného procesu, které je uveden v dokumentu „Zadavatelsky_projekt_v1.04.doc“. Toto je způsobeno postupným ujasňováním požadavků s investorem.

Aktivujícím článkem procesu je zájem klienta, respektive jeho příchod na pobočku (příchod k partnerské organizaci). Tímto je míněno vyjádřením zájmu zákazníka na pobočce našeho partnerského zařízení (partnerského lékaře). Tomuto předchází úkony, které závisí na konkrétní situaci a řešení těchto úkonů není součástí požadavku investora. Za počátek procesu tak považujeme až příchod klienta na pobočku s konkrétním přáním.

V partnerském zařízení kdekoliv ve světě je klientovi vytvořena předběžná nabídka a to na základě představ, které na této pobočce vyjádří. Takto vytvořená nabídka se předkládá klientovi. Na této úrovni není řešena situace, kdy bude zákazník do partnerského zařízení volat. Způsob reakce na tuto situaci bude záviset na každém partnerském zařízení. Pokud bude ochotno vytvořit nabídku pouze po telefonu, je toto možné.

Na nabídku může klient reagovat dvěma způsoby. Nabídku vytvořenou na základě jím předložených požadavků může odmítnout, v tomto případě je proces ukončen. Může také nabídku přijmout a v tomto případě je s klientem vytvořena smlouva o provedení předoperačních vyšetření. Smlouvu může zákazník nepodepsat, v tomto případě je proces ukončen. V případě podpisu smlouvy jsou zahájena předoperační vyšetření.

Výsledek předoperačního vyšetření může být:

- nevyhovující – s klientem je ukončena spolupráce v daném požadavku.
V případě jiného typu požadavku může podmínky splnit.
- vyhovující – s klientem je upřesněna nabídka.

V případě vyhovujícího výsledku zákazník formuluje přesně svůj požadavek (požadavky). Následně je nutné zjistit, zda je již zákazník zaregistrován (tedy byli mu již námi poskytnuty služby) či nikoliv. Pokud zákazník není zaregistrován proběhne jeho registrace, s kterou souvisí uložení jeho údajů a souvisejících materiálů do databáze evidence zákazníků. Výsledek samozřejmě může být také, že již je zaregistrován. Pokud je tomu tak, proběhne kontrola, zda není zákazník dlužníkem z předcházejících zásahů. Pokud je zákazník dlužný za provedené úkony, tento požadavek není akceptován a je s ním rozvázán vztah (toto je samozřejmě možné ošetřit způsobem, že zákazník okamžitě uhradí své závazky a následuje vyhodnocení, že zákazník není dlužník).

Pokud zákazník není dlužník nebo je nově registrován, následuje uzavření finální smlouvy.

Po uzavření smlouvy již nastává registrace souvisejících služeb, které budou spojeny s plánovanou plastickou operací (doprava, ubytování, kulturní vyžití atd.).

Klient se může rozhodnout realizovat svůj požadavek na plastickou operaci samostatně, aniž by využil nabídku našich souvisejících služeb (hotel, letecká doprava atd.). V tomto případě je vynechán proces upřesňování nabídky a je přistoupeno přímo k realizaci objednávky. Toto je zde z důvodů, že klient nemusí mít zájem tyto služby využít nebo může

jít o klienta z České republiky, který tyto služby nebude potřeba. Toto je řešeno v rámci níže uvedených subprocesů.

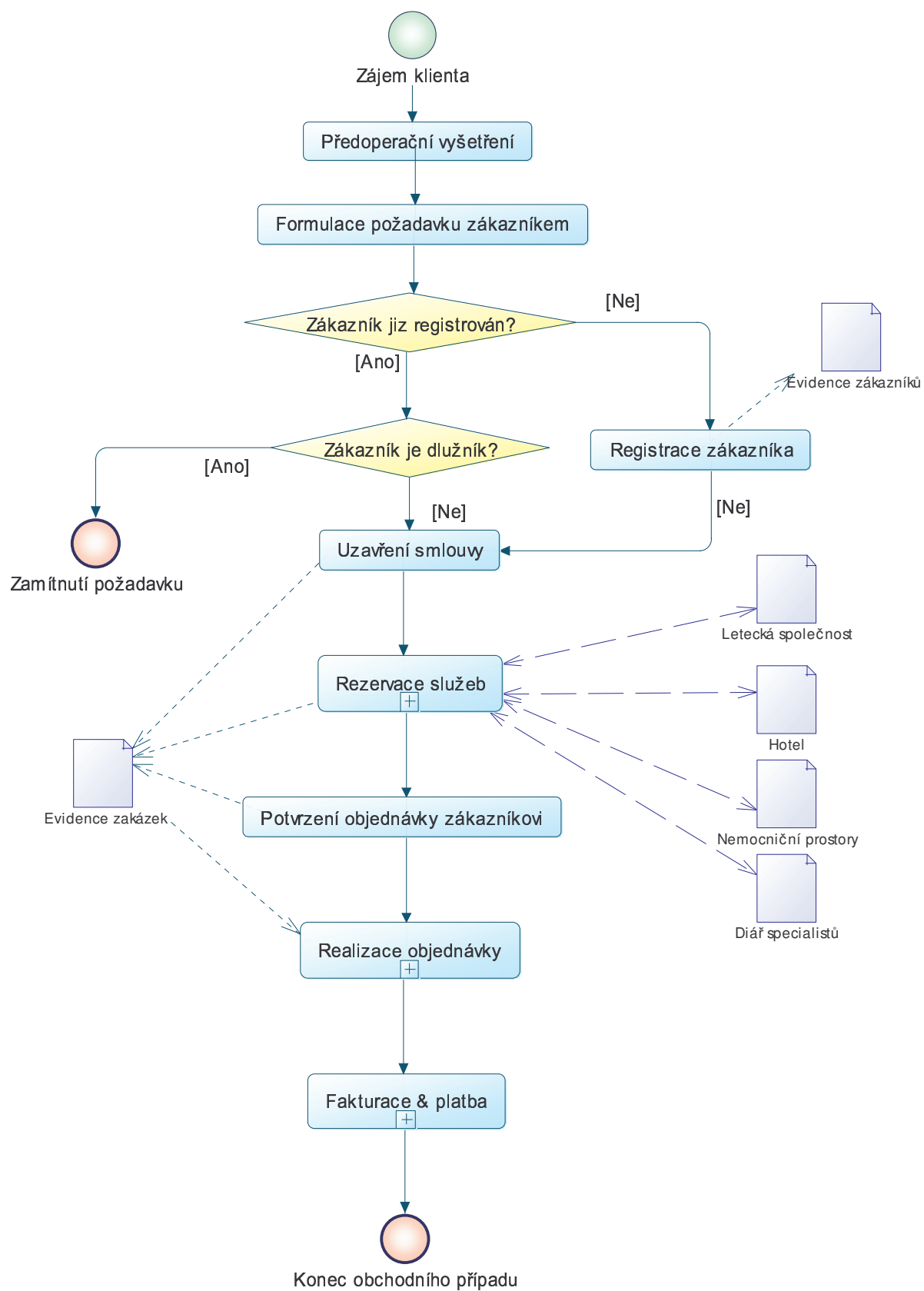
Po akceptaci nabídky (v rámci subprocesu) je informována nemocnice, hotel a letecká společnost a jsou zablokovány (objednány) příslušné termíny dohodnuté v objednávce. Tímto je proces ukončen a dochází k samotnému přiletu zákazníka, jeho ubytování, provedení operace a rekonvalescence. Toto již není cílem tohoto dokumentu. Přilet a ubytování zákazníka bylo vyřešeno zablokováním příslušných termínů. Operace je zajištěna zablokováním příslušného termínu v partnerském lékařském zařízení. Rekonvalescence a kulturní vyžití závisí na volbě příslušného hotelu a místa, kde se bude operace provádět – řešeno v následujících podkapitolách o subprocesech.

Procesy spojené s kulturním vyžitím zde nejsou řešeny. Toto je spojeno s konkrétním hotelem a nemocničním zařízením.

Po provedení realizace objednávky již následuje pouze fakturace a ukončení vztahu se zákazníkem.

Zde uvedený procesní model je jen velice hrubým nástinem celého procesu, téměř každý proces může být rozepsán na nižší úrovni. Slouží spíše jako formální zobrazení procesu vyřízení požadavku klienta.

Grafické vyjádření popisuje následující obrázek č. 5



Obrázek E 5 Core process

Obrázek č.5:

Proces rezervace služeb

Proces rezervace služeb je zpodrobněním (vyjmutím) jedné části core-procesu. Účelem je zpřehlednit obě grafická provedení.

Rezervace služeb začíná uzavřením smlouvy. Následně je nutné zjistit termíny volných míst a nabídnout je klientovi. Toto je velmi podobné části akceptace v předcházejícím grafu. Zde je to již vybíráno pod konkrétní smlouvou, kdežto ve výše uvedeném procesu byla rozepsána část, která je ještě nezatížená uzavřenou smlouvou. Je zde snazší odstoupení od vzájemných závazků.

Po vytvoření nové (respektive upravení původní) nabídky dochází k její akceptaci. Pokud klient nabídku neakceptuje, může mu být vytvořena jiná nabídka. Pokud nesouhlasí s další nabídkou (respektive jejím opakováním v různých podobách) je proces ukončen. Pokud s nabídkou souhlasí (akceptace je pozitivní) – platí jak v případě upravované nabídky tak i v případě první nabídky, je kontaktována příslušná nemocnice, kde bude zákrok proveden. U upřesnění nabídky je podstatné, že jsou zahrnuty do okolností i kontextové faktory, kterými jsou letecké společnosti a hotelová zařízení, kde bude klient ubytován. Tyto faktory mohou způsobit, že je nabídka odmítnuta a je požadováno její upravení. Nemusí například vyhovovat termín provedení operace, kvalita cílového hotelu nebo způsob dopravy (moc přestupů, pouze druhá třída atd.). Společnost Plastická chirurgie, s.r.o. si tato potřebná data sama stahuje z provozních systému spolupracujících společností (hotely, letecké společnosti, nemocnice). Je-li vybrán vhodný termín, je nabídnut zákazníkovi, který jej může či nemusí schválit. Pokud je vše schváleno, dochází k rezervaci služeb.

Další zpodrobnění části rezervace služeb bude součástí detailní analýzy a návrhu.

Proces realizace objednávky

Proces realizace objednávky následuje po procesu rezervace. Tento proces začíná termínem nástupu na „prázdniny“. Proces postupuje postupně od příletu zákazníka, kdy je kontaktován naším hotelem, který zajišťuje jeho ubytování a odvoz do hotelu z letiště. Po ubytování má příslušné odpočinkové aktivity v závislosti na dohodnuté smlouvě a pak následuje příjem do nemocnice. Odvoz do nemocnice zajišťuje buď hotel nebo společnost Plastická chirurgie, s.r.o. opět v závislosti na uzavřené smlouvě.

Po příjmu do nemocnice je provedena operace a následuje rekonvalescence. Pokud má klient objednány další doprovodné služby, pak jsou realizovány a na jejich konci je odlet klienta domů. Pokud tyto služby objednány a zaplacený nemá následuje rovnou po nezbytné rekonvalescenci odlet klienta domů.

Řízení lidských zdrojů

Nábor zaměstnanců – při poklesu počtu zaměstnanců anebo rozšiřování firmy dochází k nutnosti přijmout nové zaměstnance. Model zachycuje proces přijímání nových zaměstnanců. Je z něj vidět:

Prvá fáze je příprava přijímací řízení, po který přichází na rad pohovor s kandidáti na volné pozice. Následuje rozhodující fáze, kde se zhodnotí kvalifikační a jiné předpoklady kandidáta, na jejich základe je rozhodnuto o jeho (ne)přijetí. V případě přijetí je zapsán do databáze zaměstnanců.

Výplata mezd – model zachytává proces kalkulace a výplaty mezd zaměstnanců. Je z něj vidět:

Ve výplatné období proběhne kalkulace mezd každého zaměstnance, jako vstup slouží databáze zaměstnanců, odkud se čerpají údaje o odpracovaných hodinách, přesčasech,

prémii, atd. Po této fázi dochází k převodu výplaty na účet každého zaměstnance (číslo účtu se zjišťuje z databáze zaměstnanců) a platby jsou zaevidovány pro finanční řízení.

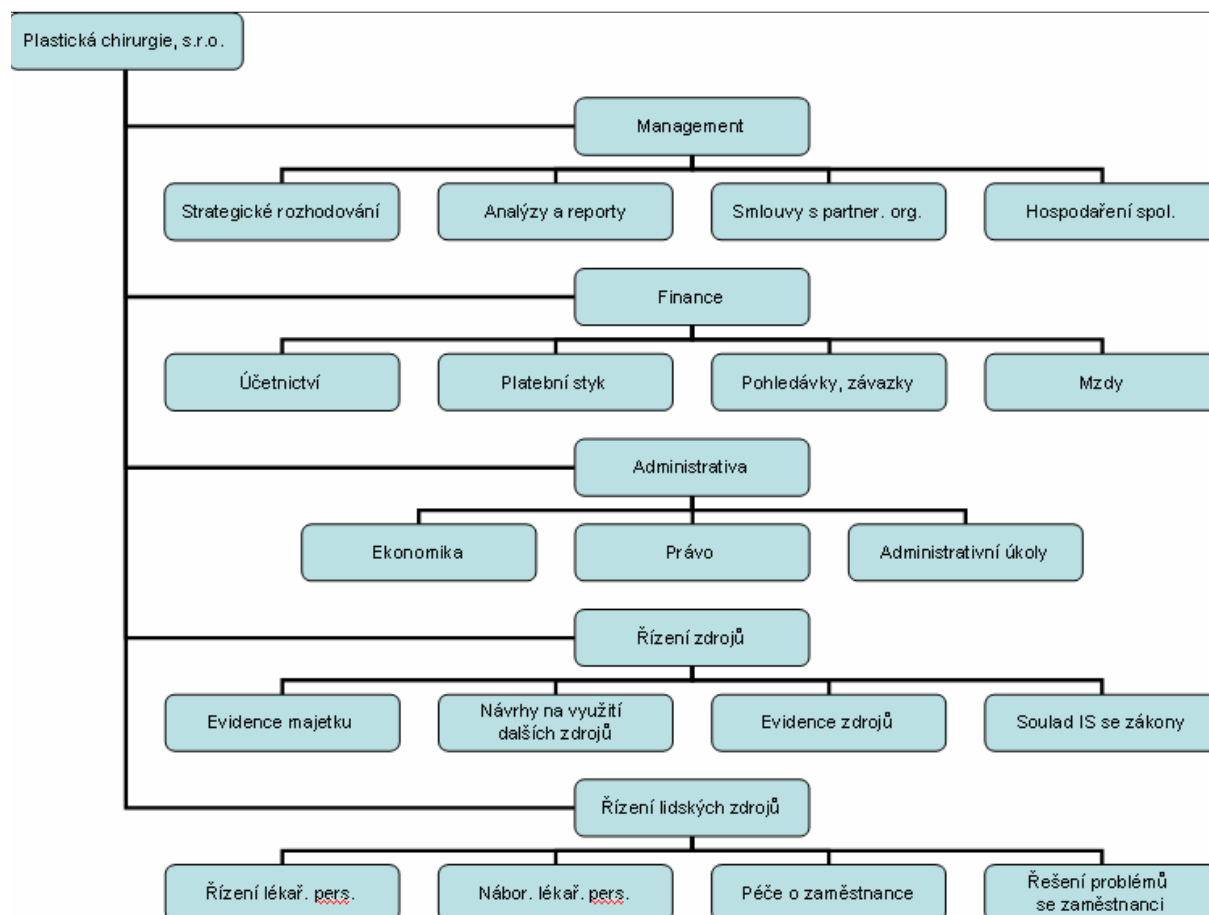
Propuštění zaměstnanců – model zachytává proces propuštění zaměstnance, v případě nesplňování podmínek.

Nejdříve dochází k výstraze zaměstnance pohovorem. Jestli zaměstnanec napraví nedostatky, může pokračovat v práci ve firmě, jestli je zaměstnanec nesplní, je mu dána výpověď, je zaznamenána do databáze zaměstnanců a později převeden výmaz ex-zaměstnance.

E.2.3 Funkční analýza a datové toky na úrovni GAN

E.2.3.1 Hrubý rozpad funkcí

Následující schéma vyjadřuje na globální úrovni funkce, které mají jednotlivé části firmy vykonávat, a které budou podporovány IS. Ačkoliv to v grafu není vyjádřeno exaktně, management je nadřazen ostatním útvarům (finance, administrativa, řízení zdrojů, řízení lidských zdrojů). V grafu nejsou vyjádřeny všechny funkce, jak bylo uvedeno výše. Funkce jsme pro účely tohoto hrubého rozpadu slučovali a to podle příbuznosti.



Obrázek E 6 Rozpad funkcí

E.2.3.2 Popis jednotlivých funkcí

Název funkce	Popis funkce
1 Management	
1.1 Strategické rozhodování	Funkce umožňuje IT podporu strategického rozhodování ve vztahu ke stanoveným cílům
1.2 Reporty	Zajištění podávání reportů ze strategických odvětví (např. vývoj trhu), jejich zpracování a analýza
1.3 Analýzy	Funkce umožňuje nejhlubší analýzy podniku
1.4 Hospodaření společnosti	Podobně, jako analýzy podniku, umožňuje hlubší analýzu, která je však cíleně zaměřena na kontrolu hospodaření společnosti a s tím souvisejících ukazatelů
2 Personální oddělení	
2.1 Nábor zaměstnanců	Evidence volných pozic ve firmě a zajištění nových pracovních sil
2.2 Evidence zaměstnanců	Evidence pracovníku firmy
2.3 Řízení lékař. pers.	Funkce umožňující kontrolu vytíženosti zaměstnanců. Poskytuje údaje pro vyšší stupně řízení.
2.4 Péče o zaměstnance	Funkce pro motivační program pro zaměstnance. Evidence spokojenosti zaměstnanců s pracovním prostředím.
2.4 Řešení problémů se zaměstnanci	Evidence vzniklých problémů a způsobů jejich řešení.
3 Finanční řízení	
3.1 Účetní kniha	Vedení účetnictví a hlavní přehled o financování podniku
3.2 Platební styk	Zprostředkování rozličných plateb, evidence přijatých a vyplacených plateb
3.3 Pohledávky	Sledování pohledávek
3.4 Závazky	Sledování závazků
3.5 Mzdy	Zajišťování výplat
4 Administrativa	
4.1 Ekonomika	Funkce pro podporu ekonomických analýz, které jsou dále poskytovány vyšší úrovni řízení.
4.2 Právo	Funkce vytvářející prostor pro tvorbu právního zajištění činnosti společnosti
4.3 Administrativní úkony	Podpora širokého spektra administrativních úkonů
5 Řízení zdrojů	
5.1 Evidence majetku	Funkce zajišťující evidenci majetku, který patří společnosti nebo je jí využíván
5.2 Návrhy na využití dalších zdrojů	Funkce umožňující analýzy aktuálního stavu a skutečných potřeb zdrojů, návrhy na jejich doplnění (omezení), předávané vyšším úrovním podniku
5.3 Evidence zdrojů	Funkce pro podporu širokého spektra zdrojů, které jsou využívány k dosažení hlavních cílů společnosti
5.4 Zajištění souladu IS se zákony	Funkce zajišťující přístup k normám, které ovlivňují

	činnosti IS
5.5 Řízení partnerských organizací	Metodicky řídí a hodnotí partnerské organizace. Zajišťuje rozsáhlou činnost s partnerskými organizacemi spojenou.

Tabulka E 8 Rozpad funkcí

E.2.3.3 Hierarchie funkcí IS

Obrázek E 7 Hierarchie funkcí IS

E.2.3.4 Popis vlastností v hierarchii funkcí IS

Č.	Název funkce	Popis funkce
1	Správa zakázek	Umožňuje vést evidenci zakázek.
2	Správa objednávek	Umožňuje vést evidenci objednávek.
3	Správa zákazníků	Umožňuje vést evidenci zákazníků.
4	Správa zahraničních partnerů	Umožňuje vést evidenci zahraničních partnerů.
5	Správa tuzemských zdravotnických zařízení	Umožňuje vést evidenci tuzemských zdravotnických zařízení a informací o jejich volné kapacitě.
6	Správa chirurgů	Umožňuje vést evidenci chirurgů a informací o jejich volné kapacitě.
7	Správa doprovodných služeb	Umožňuje vést evidenci poskytovatelů doprovodných služeb a informací o jejich volné kapacitě.
8	Správa zaměstnanců	Umožňuje vést evidenci zaměstnanců.
9	Účetnictví	Umožňuje vést účetnictví organizace.

Tabulka E 9 Popis funkcí IS k alternativnímu pohledu

E.2.3.5 Kontextový diagram a diagram datových toků

Kontextový diagram vyjadřuje ve zjednodušené podobě toky informací, stejně jako je tomu u DFD. DFD je však podrobnější. Kontextový diagram vyjadřuje situaci budoucího systému v kontextu daného prostředí.

Model datových toků (DFD diagram) je informací o jednotlivých tocích dat uvnitř a vně organizace. Diagram vyjadřující tyto toky informací je zobrazen na následujícím diagramu. Diagram správně patří až na úroveň DAN, ale z důvodů upřesnění si požadavků investora byl vyhotoven již ve fázi GAN, a proto je i zde uveden.

Mohli bychom se věnovat podrobnému popisu tohoto diagramu, jeho obsah je však natolik jednoznačný, že není třeba žádný detailní popis provádět.

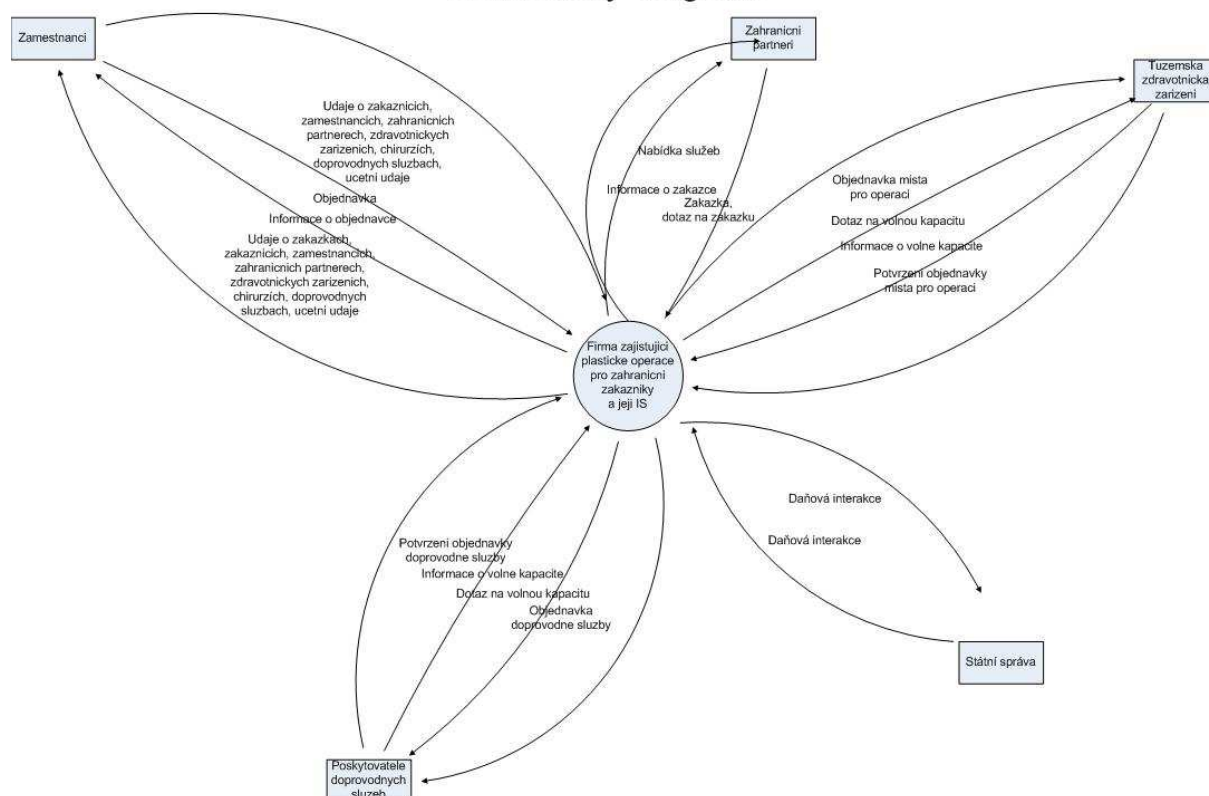
Zdůraznit je nutno skutečnost, že „Zaměstnanci“ jsou zde uvedeni dvakrát. Toto je z důvodu větší přehlednosti diagramu.

Základními toky dat, které v podniku probíhají jsou následující:

- tok údajů o partnerech,
- tok údajů o zaměstnancích,
- tok údajů o účetních informacích,
- tok údajů při komunikaci s partnerskými zařízeními a lékaři,
- tok informací o doprovodných službách (v různých podobách),
- tok informací o objednávkách

a velmi důležitý tok informací o platbách.

Kontextový diagram





Nejprve si je třeba uvědomit, že se v našem případě jedná o subjekt spadající do kategorie malých a středních podniků, který si může dovolit investovat do IS/ICT pouze omezené množství prostředků. Přitom klade na IS systém široké spektrum požadavků.

- podpora běžných administrativních funkcí
- elektronická forma komunikace, včetně výměny dokumentů v elektronické podobě
- propojení s informačními systémy vnějších subjektů (nemocnice, cestovní agentury, ubytovací zařízení, atd.)
- podpora partnerských nemocničních zařízení (PNZ), sledování historie klienta
- řízení volných kapacit nemocnic a specialistů

Na druhou stranu předmětná oblast podnikání představuje na trhu dosud ojedinělé požadavky kladené na IS a její integrace do současných SW systémů bude představovat

vysokou přidanou hodnotu a může vést ke značné míře konkurenceschopnosti na trhu poskytování služeb plastické chirurgie.

Jedním ze zásadních požadavků je bezesporu vedení veškeré dokumentace v elektronické podobě a to hlavně při komunikaci s partnerskými nemocničními zařízeními (PNZ) v zahraničí. Tato zařízení nepodléhají české legislativě, je na místě předpokládat velkou roztržičnost legislativních požadavků na elektronickou komunikaci. Každý subjekt nacházející se v jiné zemi bude podléhat jiné legislativě týkající se ochrany osobních údajů a jejich poskytování elektronickou cestou, včetně požadavků na zabezpečení této formy komunikace. V některých zemích tato problematika nemusí být řešena vůbec.

Při propojování IS bude nutné analyzovat konkrétní informační systém u vnějších subjektů a jeho možnosti propojení s navrhovaným IS. Největší přínos propojení pak lze vidět u propojení s IS nemocnic a tudíž v efektivním řízení volných kapacit nemocnic a specialistů.

E.2.4.1 Analýza požadavků na IS

Elektronická komunikace s sebou přináší nové spektrum problémů, ať již se jedná o legislativní či technické problémy. Legislativu v tomto směru nelze opomíjet, představuje velkou část požadavků kladených na IS. Proto velký důraz na technickou stránku celého řešení může vést v závěru de facto k ilegaltě IS a k promarnění investovaných prostředků. Zákon též může pomoci při vymáhání pohledávek soudní cestou, což je při přechodu k „bezpapírové“ kanceláři dalším důležitým faktorem pro úspěch IS.

.E.2.4.1.1 Elektronická komunikace

Pro komunikaci se subjekty v České republice je rozhodující stávající legislativa. Ta již konečně začíná smysluplným způsobem umožňovat elektronickou komunikaci a staví ji na stejnou úroveň jako klasickou listinnou formu. Hlavním prostředkem, který z pohledu zákona zajišťuje rovnost takovéto komunikace, je elektronický podpis. Stávající úprava je však teprve v začátcích, bude nutné analyzovat a vyhodnotit i budoucí novely.

Dalším a možná ještě důležitějším aspektem je ochrana osobních údajů (dle zák. 101/2000 Sb. v současném znění). Mnoho uživatelů IS bude pracovat s citlivými údaji o klientech a tudíž je nutné IS navrhnout takovým způsobem, aby byly plně splněny legislativní nároky a nedošlo ke kolizi se zákonem. Dále pak bude nutné identifikovat pracovníky, kteří s citlivými údaji přicházejí do styku a provést alespoň základní školení v oblasti bezpečnosti dat. Jedná se především o administrativní pracovníky a lékařský personál.

Při komunikaci se zahraničními subjekty je důležité pro každou zemi analyzovat legislativní požadavky na ochranu osobních údajů, uznání elektronické komunikace z pohledu zákona (hlavně pro řešení možných konfliktů soudní cestou) a požadavky legislativy na zabezpečení elektronické komunikace. Klienti také budou muset souhlasit s poskytnutím osobních údajů pro potřeby IS, tento fakt je nutné prosadit do smluvních podmínek s klientem!

S ohledem na orientaci firmy spíše na západní trhy – tzn. hlavně USA a země EU – odkud bude pocházet majoritní část klientů, lze však očekávat vzájemně velmi podobnou legislativu. Komunikaci se subjekty z prostoru EU bude velice významným způsobem ulehčovat i členství ČR v EU a z toho plynoucí požadavky na sjednocování a harmonizaci legislativy.

.E.2.4.1.2 Propojení s informačními systémy vnějších subjektů

Propojování s IS všech vnějších subjektů klade na IS systém požadavky jak legislativního rázu, tak především technického rázu.

Legislativní požadavky jsou víceméně stejné jako u předchozího bodu, proto se dále budeme zabývat technickou stránkou.

Na první pohled je jasné, že co subjekt, to jiný informační systém, to jiná míra možností propojení. Není možné tedy očekávat jednotné rozhraní mezi jednotlivými IS, a to ani mezi IS stejné kategorie subjektů. Naopak, předpokladem je, že každá organizace bude mít jiný informační systém a tudíž bude nutné řešit každé jednotlivé propojení. To si může vyžádat dodatečné nákladů nezanedbatelné výše. Také není možné cílový subjekt žádným způsobem donutit k propojení, což vede k nutné stimulaci, nejlépe finančního charakteru, např. v podobě určité provize. Tento bod však již bude muset být řešen v rámci každé smlouvy o zprostředkování služeb PNZ pro Plastickou chirurgii s.r.o. Pokud se nelze se subjektem dohodnout na jakékoli aktivní účasti, nelze s ním ani spolupracovat!

Hlavní faktory bránící propojení:

- smluvní
 - nutnost zavázat se k propojení
 - informační a bezpečnostní politika
- legislativa
 - uznání elektronické formy komunikace
 - požadavky na zabezpečení
 - ochrana osobních dat
- technické
- finanční

.E.2.4.1.3 Podpora PNZ, sledování historie klienta

Informační systém musí poskytovat zázemí pro partnerská nemocniční zařízení, hlavně pak sledování historie komunikace s klientem. Tento základní požadavek je součástí Zadavatelského projektu, uzavřeného s Plastickou chirurgií s.r.o., včetně podrobného nástinu způsobu podpory. Jde především o možnost sledovat předchozí nabídky klientovy, evidovat jeho požadavky, změny a výsledky.

Proces podpory PNZ lze rozdělit do dvou fází:

- 1) Anonymní nabídka služeb
- 2) Poskytování konkrétních služeb konkrétnímu klientovi

V první fázi je klient přijat PNZ, jsou vylechnuty jeho požadavky a učiněna nabídka služeb. Klient vystupuje vůči IS jako anonymní klient, jemu učiněná nabídka je uložena do IS, např. pro pozdější analýzu poskytovaných služeb. Důležité je, aby byl klient vůči systému anonymní, ale přesto nějakým způsobem identifikován pro případ přijetí nabídky.

Pokud klient nabídku přijme, vstupuje do systému jako konkrétní klient a partnerská zařízení mají přístup k jeho historii v IS a následně mohou, v souladu s procesními schématy, postupovat dále k předoperačním vyšetřením.

Při takovémto nastavení procesů je však nutné si uvědomit, že druhá strana (PNZ) pracuje de facto se dvěma IS najednou a provádí takřka duplicitní evidenci údajů. Jednak pracuje se svým IS, jelikož vykonává běžnou lékařskou praxi a tu musí nějakým způsobem vykazovat, jednak s IS Plastické chirurgie s.r.o, neboť se ve smluvních podmínkách zavázal

ke spolupráci. Za této situace by tedy bylo vhodné, propojit IS PNZ, aby se tak zamezilo dvojímu zadávání dat. Bohužel, takovéto propojení by si dle bodu .E.2.4.1.2 mohlo vyžádat příliš velké náklady, konkrétní dohody o vzájemném poskytování údajů proto bude vhodné uzavřít pouze s několika hlavními PNZ.

.E.2.4.1.4 Řízení volných kapacit nemocnic a specialistů

Pro efektivní fungování podniku je nutné efektivním způsobem řídit volné kapacity nemocnic a specialistů. Z pohledu IS je toto možné dosáhnout propojením s IS smluvní nemocnice, která bude v první fázi poskytovat veškeré lékařské zázemí. Dle bodu .E.2.4.1.2 bude nutné právě s touto nemocnicí vybudovat efektivní propojení IS. Propojení si nevyžádá příliš velké náklady (pokud budeme uvažovat to, že se jedná o jediný subjekt a ne o mnoho různorodých subjektů). Personální požadavky pak budou zajišťovány v souladu s časovými možnostmi nemocnice.

.E.2.4.1.5 Uživatelé

Byli identifikovány následující skupiny uživatelů IS:

- Zaměstnanci
- Klienti – stávající i potencionální
- PNZ
- Dodavatelé služeb

Pro každou z těchto skupin musí IS co nejlépe zajišťovat pokrytí a uspokojení jejich potřeb.

E.2.5 Globální návrh designu systému

Informační systém Plastické chirurgie s.r.o. představuje unikátní směs požadavků a z nich vyplívajících řešení. V zásadě je možné provést rozčlenění celého informačního systému na dvě části:

- Standardní část
- Unikátní část

Standardní oblast představuje běžnou firemní agendu. Zde je nejlepším řešením zakoupení na trhu dostupných standardizovaných TASW. Ty představují časem prověřené řešení, poskytují širokou podporu i co se týče souladu s legislativou a nejsou příliš nákladné. Také jsou dostatečně modulární a s novými potřebami stačí pouze zakoupit nový modul. Výběr konkrétního ERP systému bude cílem Detailní fáze analýzy a návrhu (DAN). Do této kategorie produktů lze počítat i SW pro Kontaktní Centrum.

Unikátní část zrcadlí samotnou jedinečnost business projektu a tudíž bude nutné ji řešit jiným způsobem, viz. následující odstavce.

E.2.6 Jádro informačního systému (Business modul)

Jádro IS Plastické chirurgie s.r.o. a v něm uložená funkcionalita by měla plně pokrýt jedinečné požadavky celého projektu. Business modul podporuje především procesy komunikace s PNZ, smluvní nemocnicí a dodavateli služeb (letecké společnosti, hotely, cestovní kanceláře). Součástí business modulu je také databáze klientů, PNZ a dodavatelů služeb.

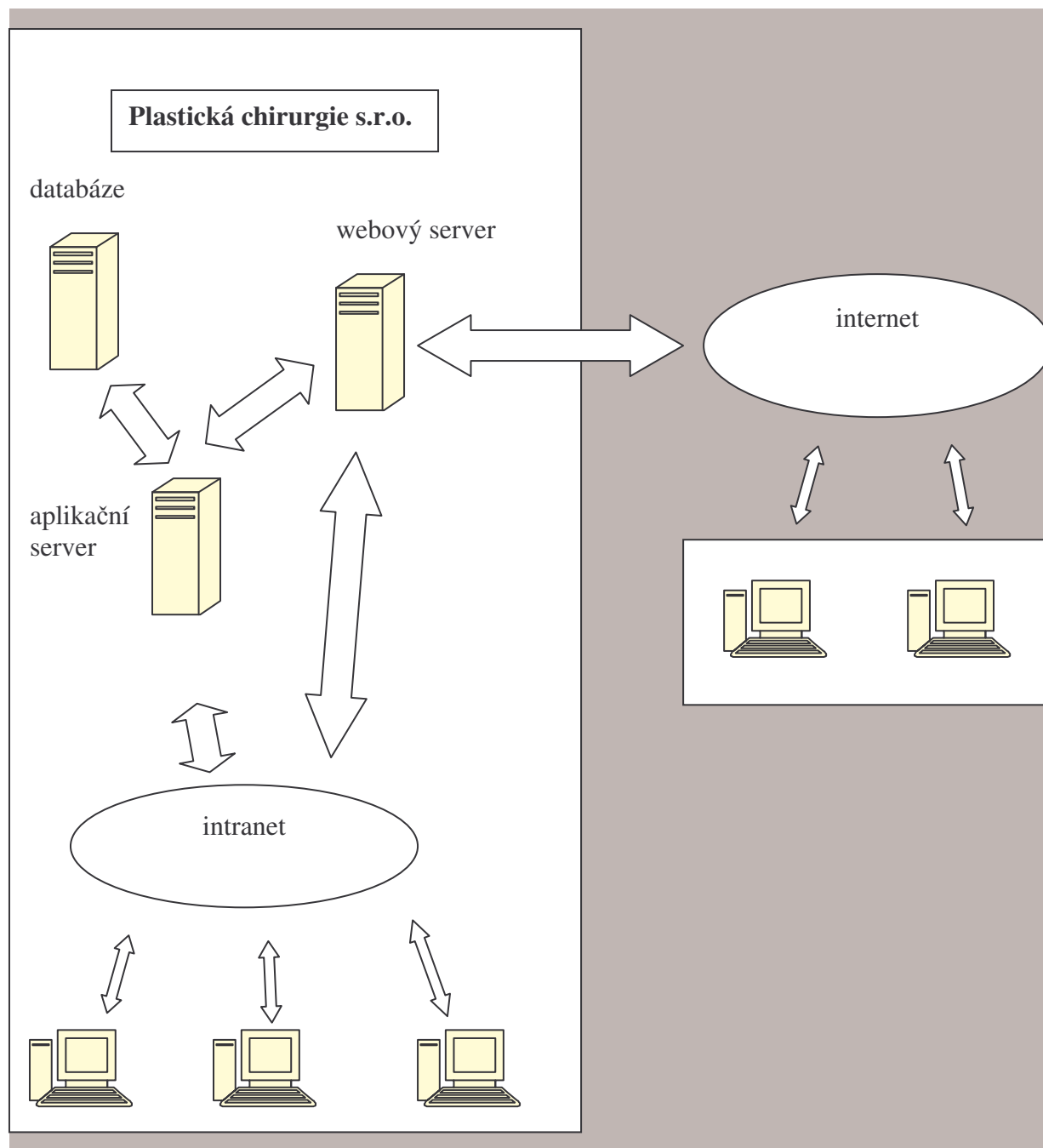
Námi navrhovaná architektura IS bude vycházet z třívrstvého modelu. Jeho základy budou spočívat na robustním databázovém serveru. Ten bude propojen s aplikacemi vnitřní sítě podniku a bude poskytovat data i aplikačnímu serveru. Ten bude zprostředkovávat komunikaci s vnějšími subjekty. Řešení komunikace s vnějšími subjekty bude založeno na standardizovaných internetových službách a technologiích.

Proto jádrem našeho řešení je aplikační server, zastřešující veškeré služby správy toku dokumentů mezi jednotlivými subjekty, poskytující služby související kompletní správou klientů, který je integrován s ostatními podnikovými aplikacemi. Pro vnější subjekty budou služby aplikačního serveru poskytovány prostřednictvím webového serveru. Navržený webový portálový systém bude hlavním centrem pro komunikaci s potencionálními – ale i stávajícími – klienty, partnerskými nemocničními zařízeními, stejně tak jako s ostatními subjekty, poskytujícím služby klientům při jejich pobytu v ČR. Pro konkrétní skupiny uživatelů budou v rámci definovaných rolí poskytovány patřičné služby, pro každou skupinu bude dostupná jiná kategorie služeb.

Jako obchodní partnery se bude snažit společnost Plastická chirurgie s.r.o získat nemocnice, jež budou ochotné se propojit s jejím informačním systémem a budou ochotny poskytovat informace o nabízených službách.. Nabídka konkrétních služeb, bude řešena prostřednictvím webové aplikace, která poběží na firemním webserveru a dále pak prostřednictvím partnerských organizací, které budou zajišťovat kromě vlastní prezentace zároveň i další úkony (lékařská vyšetření, atd.). Na tomto portále si budou moci zákazníci sami ověřit, zda jejich požadavek může být naší společností uspokojen. Pokud je daná operace v nabídce, je na klientovy, aby navštívil příslušnou partnerskou organizaci a nechal si provést potřebná vyšetření.

Součástí business modulu je také databáze klientů. Zde je třeba použít dostatečně spolehlivé a robustní řešení, které bude možno použít i v ostatních databázích dalších modulů. Jako uživatelský interface bude opět použita webová aplikace, tak aby byla databáze v rámci firemního intranetu operativně přístupná. Je pochopitelně zapotřebí důkladně vyřešit zabezpečení přístupu.

Tato webová aplikace umožní komunikaci s partnerskými nemocnicemi v zahraničí (zajišťující klienty), které jejím prostřednictvím budou znát naše kapacitní možnosti (neobsazenost termínů) a budou moci zasílat objednávky (tj. přihlášky na termíny) i s kompletní předoperační dokumentací. Volné termíny bude vkládat naše partnerská nemocnice (popřípadě jí pověřená osoba), u které se budou úkony provádět. Velice důležitá je správná definice přístupových práv a práv pro úpravy již zadaných dat a vysoký stupeň zabezpečení (šifrování).



Obrázek E 8 Architektura IS Plastické chirurgie

E.2.7 Shrnutí

S důrazem na efektivní vynakládání finančních prostředků navrhujeme řešit běžnou firemní agendu v IS pomocí standardizovaných komerčních aplikací dostupných na trhu. Výběr konkrétního balíku bude naplní další fáze analýzy a návrhu (DAN).

Hlavní část budoucího IS bude tvořena aplikačním serverem, poskytujícím požadované služby. Pro řešení jednotlivých subsystemů informačního systému navrhujeme použít především vlastní webové aplikace, ke kterým je možno přistupovat z jakéhokoli počítače či handheldu vybaveného webovým prohlížečem (po ověření příslušných přístupových oprávnění). Většina aplikací poběží v prostředí firemního intranetu a aplikace, které budou využívány pro komunikaci mezi partnerskými organizacemi budou využívat

prostředí internetu s využitím dostatečně zabezpečené šifrované komunikace pro zajištění potřebné bezpečnosti.

Konkrétní řešení bude prezentováno v dokumentu Detailní analýza a návrh.

E.2.8 Komunikační architektura

Komunikační architektura bude založená na sedmi vrstvém ISO OSI modelu. Pro vnitřní komunikaci se bude používat ethernet, eventuelně šifrovaného bezdrátového připojení WiFi. Pro datové přenosy mimo firmu se využije internetového připojení. Šifrování dat se bude provádět buďto na úrovni aplikace nebo bude vytvořena virtuální privátní síť pro každé spojení mezi firmou a protistranou. To vše bude záležet na dohodě mezi účastníky komunikace. V nejzazším případě se bude moci využít soukromého šifrovaného datového kanálu, což přináší značné náklady, ale na druhou stranu je to jeden z nejbezpečnějších způsobů komunikace.

E.2.9 Standard ISO OSI

ISO OSI je sedmi vrstvý model síťové architektury. První vrstva (fyzická) bude postavena na specifikaci 100BASE-TX (100Mbit/s). Co se týče páteční sítě, ta bude využívat buďto 1000BASE-T nebo 1000BASE-TX, bude záležet na protistraně, který z výše zmíněných protokolů bude používat.

Druhá vrstva (linková) bude používat dnes nejrozšířenější protokol „ethernet“. Třetí vrstva (síťová) bude IP a čtvrtá (transportní) bude TCP. Ostatní vrstvy 5-7 (relační, prezentační, aplikační) budou řízeny jak operačním systémem tak vlastními aplikacemi.

E.2.10 Důvody pro zvolení daného standardu

Výhodou použití ethernetu v ISO OSI je v tom, že je dnes nejběžnějším komunikačním protokolem a poskytuje uživateli flexibilitu v řešení datových sítí. Protože se jedná o nejrozšířenější protokol, tak bude bez problémů zajištěna kompatibilita mezi jednotlivými prvky určené pro komunikaci a zároveň při pohledu z finanční stránky se bude jednat o méně nákladné řešení než u konkurenčních proprietárních řešeních. Tento protokol zřejmě nebude použit pro přístup do vnějších sítí, kde jsou jiné podmínky fyzické vrstvy.

E.2.11 Hardwarová architektura

Celkové zpracování dat ve firmě bude centralizované. Z toho vyplývá, že jednotliví zaměstnanci budou používat vlastní osobní počítač (popř. přenosný počítač). Ty budou zapojeny přímo do datové sítě (kabelem nebo bezdrátově), a tím se propojí s ostatními stanicemi, především však s hlavním serverem a tiskárnami.

Veškeré počítačové stanice, servery, tiskárny a aktivní síťové prvky budou dodávány specializovanou firmou, a zároveň touto firmou budou zařízení spravovány. Tímto se má namysli správa pouze hardwarové části, nikoli softwarové. S danou firmou bude dohodnuto:

- Servisní zásah do 24h, a to včetně víkendů a svátků
- Profesionální přístup servisních techniků

Při použití outsourcingu se společnost bude moci naplno věnovat své primární činnosti, nebude se muset přímo starat o hardwarové komponenty. S tím budou spjaté zřejmě větší finanční výdaje.

E.2.12 Počítačové stanice

Nároky na počítačové stanice budou záležet na zatížení klientské části informačního systému a dále na jednotlivých aplikacích. Není však předpoklad, že by bylo nutné nakupovat technologicky nejnovější a nejrychlejší zařízení, podstatná bude především funkčnost a spolehlivost.

E.2.13 Servery

Počítačových serverů bude několik:

- Storage
- Server pro datovou základnu
- Mail/Internet/Proxy Server
- Firewall/NAT Server
- Aplikační server

Tato struktura serverů se v průběhu života firmy bude přizpůsobovat vnitřním i vnějším nárokům. Bude nutné již předem celou strukturu připravit na růst počtu zaměstnanců, tedy růst počtů počítačových stanic, a z toho vyplývající vzrůstání zatížení serverů.

E.2.14 Síť

Jednotlivé partnerské organizace budou s centrálou, resp. s centrálním serverem, komunikovat po internetu, ke kterému budou připojeny pevnou linkou (využíváno bude šifrované VPN). V rámci jednotlivých poboček bude vybudována drátová síť na architektuře Fast Ethernetu. Plánovaná rychlost v rámci poboček je 100 Mbit/s.

E.2.15 Důvody pro zvolení technologií

Celá hardwarová struktura bude tímto připravena na vnitřní i vnější změny. Bude možné přidávat, ubírat a měnit jednotlivé hardwarové prvky podle potřeby. Struktura je tedy maximálně flexibilní.

Počítačové stanice a servery budou již z počátku nastaveny tak, aby v budoucnu při větším počtu uživatelů nedocházelo ke zpomalení práce. V průběhu technologického vývoje bude možné jednotlivé komponenty měnit za nové, popřípadě se budou moci měnit komplexní hardwarové jednotky. Výkonnost celého systému se bude muset sledovat a na základě interpretací výsledných dat se budou provádět změny.

V oblasti drátových síťových prvků se v budoucnu nebudou konat velké změny, neboť celá struktura bude nastavena na velký počet uživatelů. Kapacity linek vnitřní sítě budou dostatečné, vnější linky bude nutné opět sledovat a zvyšovat podle potřeby rychlosti komunikace.

Struktura bezdrátové sítě je nyní navržena podle současně dostupných technologických možností. Vývoj v této oblasti je razantní, v budoucnu se jistě budou implementovat nová bezpečnější a rychlejší zařízení.

E.3 Detailní analýza vybraných částí systému

Detailní analýza a návrh IS představuje zpodrobnění úvodního nástinu budoucího řešení IS, který byl proveden v etapě Globální analýza a návrh systému. Zabývá se analýzou předmětné oblasti z konkrétních pohledů jednotlivých částí systému. Jejím cílem v žádném případě není provádět výběr finálně nasazených produktů. Jejím cílem je naopak podrobně analyzovat všechny možné aspekty konkrétního projektu a to především: vnitřní požadavky organizace, požadavky na vnější subjekty, požadavky vnějších subjektů, možné architektury řešení IS, apod. S tímto je spojená tvorba detailního diagramu datových toků, který vyjadřuje toky dat mezi jednotlivými částmi systému v probíhajících procesech. Dále to jsou detailní procesní diagramy a samozřejmě také datové modely a diagramy tříd.

V detailní analýze neprovádíme analýzu všech modulů, které jsme navrhli v etapě Globální analýza a návrh. Z těchto modulů jsme si vybrali pouze některé (například business modul – komunikace se subjekty), ty, které vyžadují analyzovat podrobněji. Jsou to ty části systému, které nemají standardní charakter a které tedy nelze realizovat akvizicí již hotového řešení. Takových částí nebývá v informačním systému mnoho, jsou však zpravidla jeho kriticky důležitou částí (i s přihlédnutím ke své specifičnosti) – adresují specifické, hluboce intimní, potažmo i nezřídka neveřejné a vysoce ceněné, oblasti působení a praktiky organizace. V každé organizaci lze očekávat jistý podíl takovýchto specifických částí informačního systému, dá se i říci, že přímo úměrný dynamičnosti dané organizace ve srovnání s jejím okolím.

Podkladem pro následující detailní analýzu je Globální analýza a návrh, uvedená v předchozích kapitolách tohoto příkladu.

Při analýze jednotlivých částí IS jsme určili, které části budou pro daný systém a tedy i pro zákazníka klíčové (viz BSP v GAN). Z identifikovaných modulů jsme si vybrali ty, které jsme označili za klíčové, a které zároveň podporují core-business organizace. Takto označené moduly považujeme za nutný cíl analýzy na detailní úrovni.

Jako klíčový jsme identifikovali modul s názvem Core business. Modul je složen z několika procesů, které se dále rozkládají na subprocessy. Jedním z těchto procesů je například proces Rezervace obsahující další subprocess, a také například proces Realizace. Procesem Rezervace máme na mysli sled subprocessů a událostí, který je iniciován přijetím zakázky (podpisem smlouvy) na operaci a je zakončen výběrem vhodných vhodného termínu nejen operace, ale i dalších souvisejících služeb, pokud jsou požadovány. Proces končí rezervací (objednáním) všech těchto dohodnutých skutečností. Stejným způsobem by šlo popisovat i další z námi zkoumaných procesů. Toto však necháme do samotné části tohoto dokumentu, kde se této tématice budeme věnovat podrobně.

To, jakým způsobem budou procesy navrženy a jak budou podporovány ze strany IS, je naprosto zásadní moment, který bude ovlivňovat budoucí fungování a efektivitu celé firmy.

Tento dokument dále rozpracovává jednotlivé poznatky uvedené v dokumentu Globální analýza a návrh. Dále konkretizuje model do větších detailů až do fáze, kdy je možné začít s implementací. Přechází od logického návrhu k fyzickému. Obsahuje dokončení datového modelu, funkční a procesní analýzy, data flow diagramu a stavového diagramu.

E.3.1 Hranice systému - část systému pro detailní analýzu a návrh.

V této části analýzy se zabýváme částí informačního systému. Musíme si uvědomit, že cílem této etapy není programování a ani implementace systému. Tato část má však vytvořit podklady pro výše zmíněná dvě části realizace nového IS.

Uvědomujeme si, že kvalita a využitelnost navrhovaného systému je velmi silně závislá na kvalitě vstupních dat a řízení jejich pořizování. Na druhou stranu se domníváme, že systém musí být schopen plnit alespoň částečně požadavky i v případě, pokud data nebudou dostupná ve stoprocentní kvalitě.

Přístupy k využívání systému se mohou lišit například podle kvalifikace pracovníků, úrovně zralosti organizace a podobně. Celý systém je tedy navržen tak, aby byl po této stránce pokud možno co nejvíce flexibilní a umožňoval i vývoj v čase.

E.3.2 Analýza událostí a činností

Tato kapitola obsahuje výběr nejdůležitějších událostí, na základě kterých bude provedena analýza toku dat uvnitř procesů (společnosti) a následná funkční analýza s pomocí DFD (Data Flow Diagram – diagram datových toků). Uvedeny jsou ty toky dat, jejichž název je příliš dlouhý pro zapsání do diagramu DFD a ty toky dat, které jsou použity v konzistenční tabulce. nejde tedy o seznam všech toků. Tabulka nejsou seřazena podle čísel svých řádků, a to z důvodu zachování konzistence s níže uvedenými modely DFD, které se na tuto tabulku odkazují. Uvedená podoba tabulky je výsledkem průběžných úprav spolu s modelováním toků dat, čímž docházelo k průběžným změnám až do uvedené výsledné podoby. Tabulka tak ilustruje i předchozí pracovní postup analýzy.

Všechny podstatné kroky jsou uvedeny v DFD a následné konzistenční tabulce.

Události, které jsou přímo použity v konzistenční tabulce, jsou očíslovány přímo v názvech a zobrazeny podtrženě.

Seznam událostí		
Číslo	Název události	Typ události
8	Zaměstnanec zjistil volné termíny jednotlivých služeb a vybral vhodný termín pro zákazníka	DF
10	Zaměstnanec zarezervoval specialisty	DF
11	Zaměstnanec zarezervoval nemocniční prostory a sdělil informace o rezervaci do IS nemocnice	DF
13	Požadavek rezervace hotelu	DF
14	Požadavek rezervace letenek	DF
15	Požadavek objednání doplňkových služeb	DF
17	Potvrzení termínu zákazníkem	DF
6	<u>1 Registrace zákazníka zahraničním partnerem</u>	DF
9	<u>3 Akceptace termínu zákazníkem</u>	DF
18	<u>4 Příjezd zákazníka</u>	DF
19	<u>7 Zákazník opustil nemocnici</u>	DF
20	<u>6 Platba zákazníka</u>	DF
21	<u>2 Podpis smlouvy zákazníkem</u>	DF
22	<u>5 Zákazník se dostavil k operaci</u>	DF
23	<u>8 Lékařská zpráva od PNZ</u>	DF
24	<u>9 Termín ubytování</u>	Č
25	<u>10 Zákazník nezaplatil v daném termínu</u>	Č
26	<u>11 Storno zakázky</u>	DF
27	<u>12 Termín plnění služeb</u>	Č

Tabulka E 10 Seznam událostí

E.3.3 Diagramy datových toků

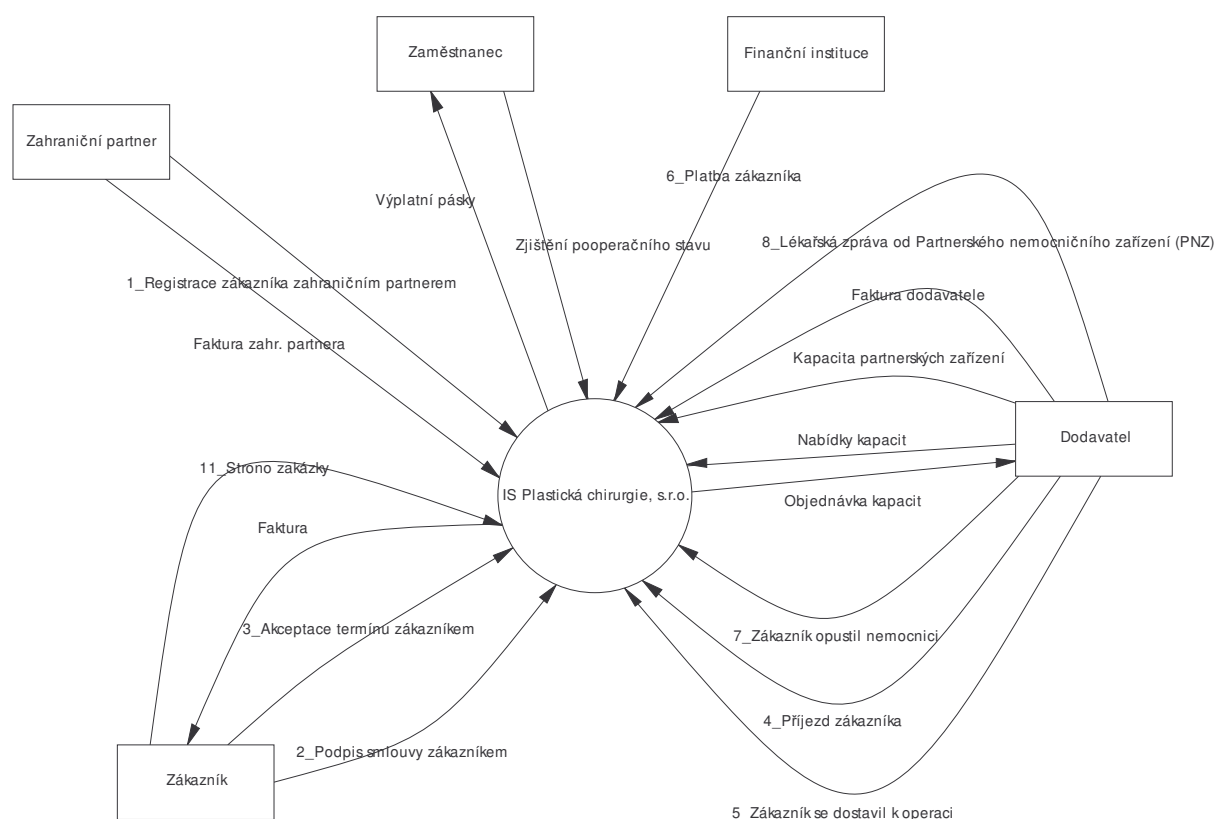
Kontextový diagram vyjadřuje ve zjednodušené podobě toky informací mezi systémem a jeho okolím. Vyjadřuje tak situaci budoucího systému v kontextu daného prostředí.

Následující podrobnější modely datových toků nižších úrovní dekompozice popisují jednotlivé toky dat uvnitř a vně organizace.

Základními toky dat v organizaci jsou:

- tok údajů o partnerech,
- tok údajů o zaměstnancích,
- tok údajů o účetních informacích,
- tok údajů při komunikaci s partnerskými zařízeními a lékaři,
- tok informací o doprovodných službách (v různých podobách),
- tok informací o objednávkách, atd.

E.3.3.1 Kontextový diagram



Obrázek E 9 Kontextový DFD

The diagram illustrates the business processes of a hospital, centered around the 'Realizace' (Realization) process. Key components and flows include:

- External Partners:**
 - Zahraníční partner** (Foreign partner) interacts with **Finance** (Faktura zahr. partnera) and **Registrace** (1 Registrace zákazníka zahraničním partnerem).
 - Dodavatel : 1** (Supplier 1) provides **Faktura dodavatele** to **Finance**.
 - Dodavatel : 2** (Supplier 2) provides **Nabídka kapacit** to **Rezervace kapacit**.
 - Dodavatel : 3** (Supplier 3) provides **5 Zákazník se dostavil k operaci** to **Realizace**.
- Internal Processes and Data Flows:**
 - Registrace** (Registration) receives **Data zákazníků** from **Zahraníční partner** and **0:10:11** from **5 Lékařské zpráva od Partnerského nemocničního zařízení (PNZ)**. It outputs **Data pacientů** to **Rezervace**.
 - Rezervace kapacit** (Capacity Reservation) receives **Rezervace kapacit** from **Dodavatel : 2** and **Termin plnění služeb** from **Realizace**. It outputs **Rezervace kapacit** to **Nemocnice** and **Volné kapacity** to **Realizace**.
 - Realizace** (Realization) receives **Data zakázek** from **Zakázky** and **5 Zákazník se dostavil k operaci** from **Dodavatel : 3**. It outputs **Realizace** to **HR** and **7 Zákazník opustil nemocnici** to **Dodavatel : 3**.
 - HR** (Human Resources) receives **Realizace** from **Realizace** and outputs **Výplata** to **Výplatní pásy** and **Záznamy z operací** to **Hodnocení záznamy z operací**.
 - Finance** (Finance) receives **Faktura dodavatele** from **Dodavatel : 1** and **Faktura zahr. partnera** from **Zahraníční partner**. It outputs **Podklady pro výplatu** to **Výplatní pásy** and **Podklady pro fakturaci** to **Dodavatelé**.
 - Dodavatelé** (Suppliers) receive **Podklady pro fakturaci** from **Finance** and output **Údaje o dodavatelích** to **Rezervace**.
 - Rezervace** (Reservation) receives **13,14,15** from **Dodavatelé** and **2 Podpis smlouvy zákazníkem** from **Zákazník : 2**. It outputs **Data rezervace** to **Zákazníci** and **Data zakázek** to **Zakázky**.
 - Zákazníci** (Customers) receive **Data rezervace** from **Rezervace** and output **Data zákazníků** to **Registrace**.
 - Zakázky** (Orders) receive **Data zakázek** from **Rezervace** and output **Zakázky** to **Realizace**.
 - Nemocnice** (Hospital) receives **Rezervace kapacit** from **Rezervace kapacit** and outputs **Nemocnice** to **Finance**.
- Documents and Data:**
 - Výplatní pásy** (Payroll) receives **Podklady pro výplatu** from **Finance** and **Výplata** from **HR**.
 - Hodnocení záznamy z operací** (Operational record evaluation) receives **Záznamy z operací** from **HR** and **Účetní záznamy** from **Realizace**.
 - Časové rozvrhy** (Timetables) receive **Kapacity** from **HR** and output **Časové kapacity** to **Rezervace kapacit**.
 - Časové kapacity** (Time capacities) receive **Časové rozvrhy** from **Časové rozvrhy** and output **Časové kapacity** to **Rezervace kapacit**.
 - Účetní záznamy** (Accounting records) receive **Účetní záznamy** from **Realizace** and output **Účetní záznamy** to **Hodnocení záznamy z operací**.
 - Údaje o dodavatelích** (Supplier data) receive **Údaje o dodavatelích** from **Dodavatelé** and output **Údaje o dodavatelích** to **Rezervace**.
 - 13,14,15** (Data) receive **Údaje o dodavatelích** from **Dodavatelé** and output **Údaje o dodavatelích** to **Rezervace**.
 - 0:10:11** (Data) receive **0:10:11** from **5 Lékařské zpráva od Partnerského nemocničního zařízení (PNZ)** and output **0:10:11** to **Registrace**.

Obrázek E 10 DFD úrovně 0

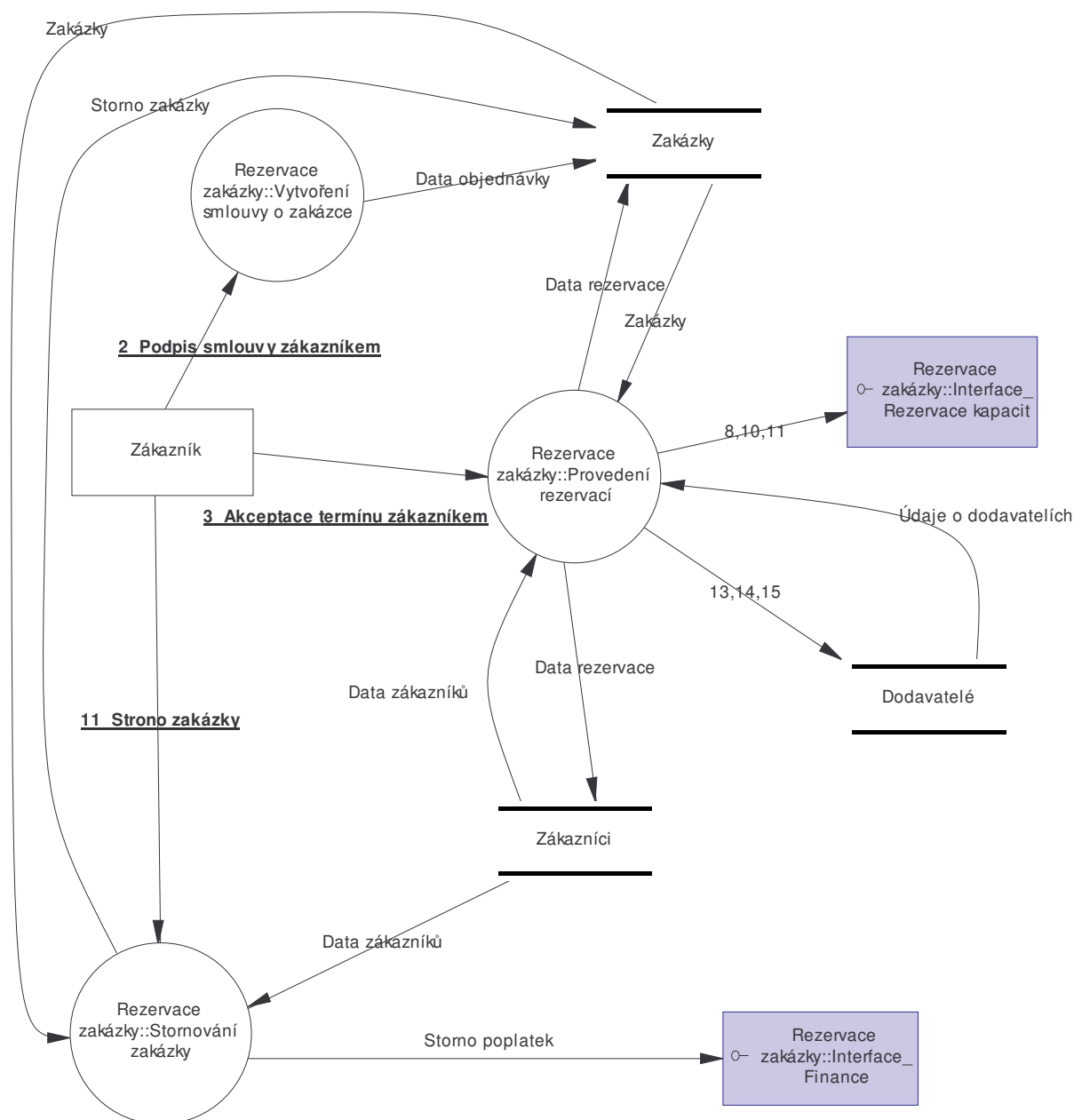
Výše zmiňovaná potřeba podrobnějšího popisu vzniká, z technického hlediska vzato, u všech funkcí s výjimkou jediné – funkce HR (ta je zjevně reakcí na jedinou časovanou událost, tedy v provozu systému bude realizována pravidelně a plánovitě). Nicméně přesto složitost funkce Registrace zákazníka lze považovat za triviální, jelikož kombinuje pouze dvě externí události (tedy postačí ujasnit si vztah těchto dvou událostí, jenž je zde víceméně zjevný). Funkce Rezervace kapacit kombinuje dvě externí události se třemi událostmi, zprostředkovanými funkcí Rezervace zakázky – je vhodné ji tedy odložit k vyjasnění až po vyjasnění funkce Rezervace zakázky. Netriviální se jeví také funkce Finance, jež má však standardní povahu a tudíž patrně nebude muset být podrobena detailní analýze.

- Rezervace zakázky a
- Realizace,

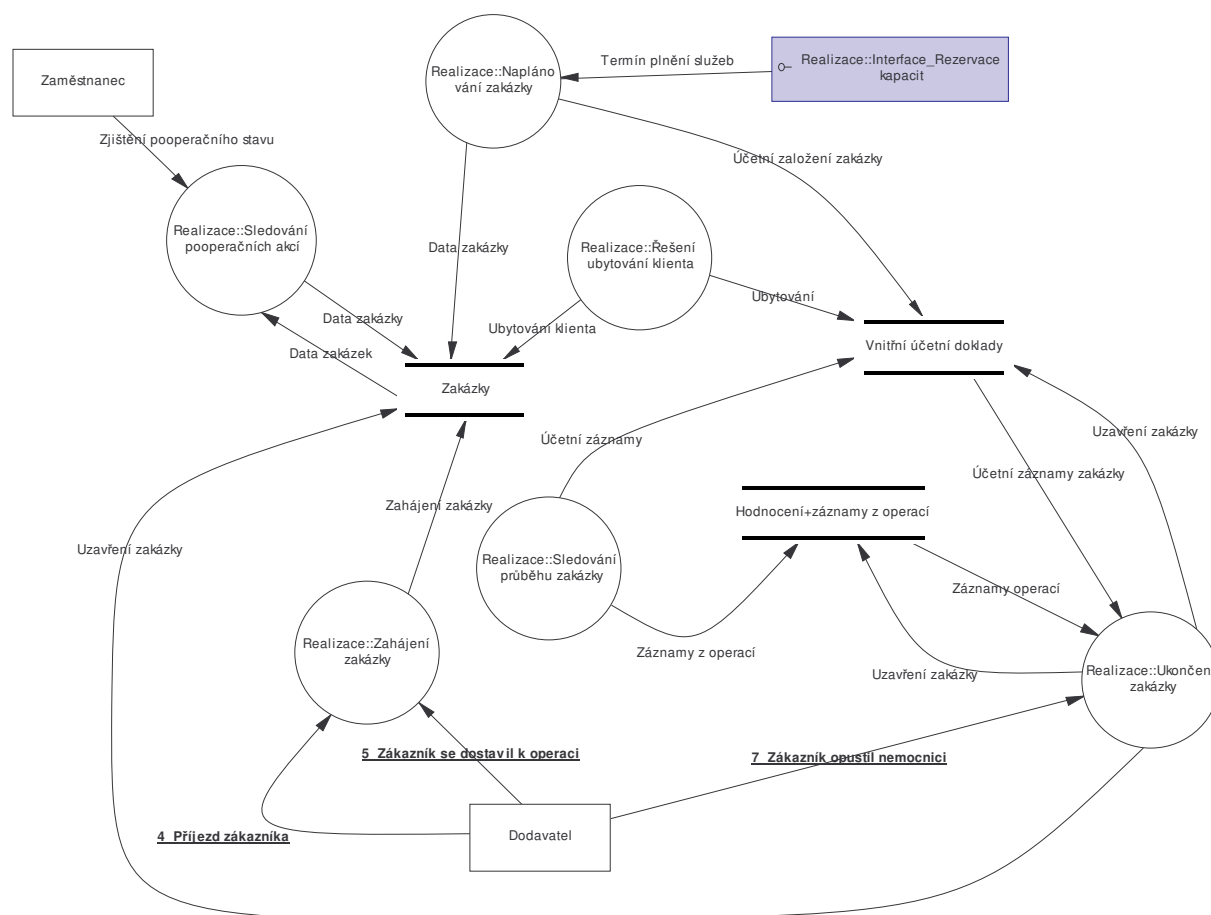
jako funkce, jež bude primárně zapotřebí detailně dekomponovat. Tyto dvě funkce jsou podrobněji popsány níže.

E.3.3.3 Diagramy datových toků úrovně 1

E.3.3.3.1 Funkce Rezervace zakázky



Obrázek E 11 Funkce Rezervace zakázky

.E.3.3.3.2 Funkce Realizace**Obrázek E 12 Funkce Realizace**

E.3.4 Procesní modely

E.3.4.1 Core proces

Core business společnosti je, jak již bylo uvedeno ve vizi a strategii, poskytování služeb estetické chirurgie. Zákazníci budou přicházet do kontaktu se společností přes kontaktní lékaře. Tito lékaři budou provádět všechna potřebná vyšetření a ta budou zasílána do centra v České republice. Je tedy nutný robustní informační systém, protože po několika letech půjde o stovky až tisíce klientů. S každým klientem je spojené obrovské množství dat v elektronické podobě, jako je například EKG, rentgeny, sono atd. Pro společnost je podstatný bezpečný, spolehlivý a rychlý přenos dat od partnerských lékařů do centra v ČR. Systém bude vyžadovat velkou flexibilitu jednak co do rozsahu, tak také do možnosti rozšíření o další předměty podnikání. Také je třeba vzít v potaz, že musí být koncipován s možností fungovat mnoha let.

V následujícím diagramu je vyjádřen průběh realizace toho procesu. Jednotlivé části procesu budou dále zpodrobněny v dalších částech této kapitoly.

Aktivujícím článkem procesu je zájem klienta, respektive jeho příchod na pobočku (příchod k partnerské organizaci). Tímto je míněno vyjádřením zájmu zákazníka na pobočce našeho partnerského zařízení (partnerského lékaře). Tomuto předchází úkony, které závisí na konkrétní situaci a řešení těchto úkonů není součástí požadavku investora. Za počátek procesu tak považujeme až příchod klienta na pobočku s konkrétním přáním.

V partnerském zařízení kdekoliv ve světě je klientovi vytvořena předběžná nabídka a to na základě představ, které na této pobočce vyjádří. Takto vytvořená nabídka se předkládá klientovi. Na této úrovni není řešena situace, kdy bude zákazník do partnerského zařízení volat. Způsob reakce na tuto situaci bude záviset na každém partnerském zařízení. Pokud bude ochotno vytvořit nabídku pouze po telefonu, je toto možné.

Na nabídku může klient reagovat dvěma způsoby. Nabídku vytvořenou na základě jím předložených požadavků může odmítnout, v tomto případě je proces ukončen. Může také nabídku přijmout a v tomto případě je s klientem vytvořena smlouva o provedení předoperačních vyšetření. Smlouvu může zákazník nepodepsat, v tomto případě je proces ukončen. V případě podpisu smlouvy jsou zahájena předoperační vyšetření.

Výsledek předoperačního vyšetření může být:

- nevyhovující – s klientem je ukončena spolupráce v daném požadavku. V případě jiného typu požadavku může podmínky splnit.
- vyhovující – s klientem je upřesněna nabídka.

V případě vyhovujícího výsledku zákazník formuluje přesně svůj požadavek (požadavky). Následně je nutné zjistit, zda je již zákazník zaregistrován (tedy byli mu již námi poskytnuty služby) či nikoliv. Pokud zákazník není zaregistrován proběhne jeho registrace, s kterou souvisí uložení jeho údajů a souvisejících materiálů do databáze evidence zákazníků. Výsledek samozřejmě může být také, že již je zaregistrován. Pokud je tomu tak, proběhne kontrola, zda není zákazník dlužníkem z předcházejících zásahů. Pokud je zákazník dlužný za provedené úkony, tento požadavek není akceptován a je s ním rozvázán vztah (toto je samozřejmě možné ošetřit způsobem, že zákazník okamžitě uhradí své závazky a následuje vyhodnocení, že zákazník není dlužník).

Pokud zákazník není dlužník nebo je nově registrován, následuje uzavření finální smlouvy.

Po uzavření smlouvy již nastává registrace souvisejících služeb, které budou spojeny s plánovanou plastickou operací (doprava, ubytování, kulturní vyžití atd.).

Klient se může rozhodnout realizovat svůj požadavek na plastickou operaci samostatně, aniž by využil nabídku našich souvisejících služeb (hotel, letecká doprava atd.). V tomto případě je vynechán proces upřesňování nabídky a je přistoupeno přímo k realizaci objednávky. Toto je zde z důvodů, že klient nemusí mít zájem tyto služby využít nebo může jít o klienta z České republiky, který tyto služby nebude potřeba. Toto je řešeno v rámci níže uvedených subprocesů.

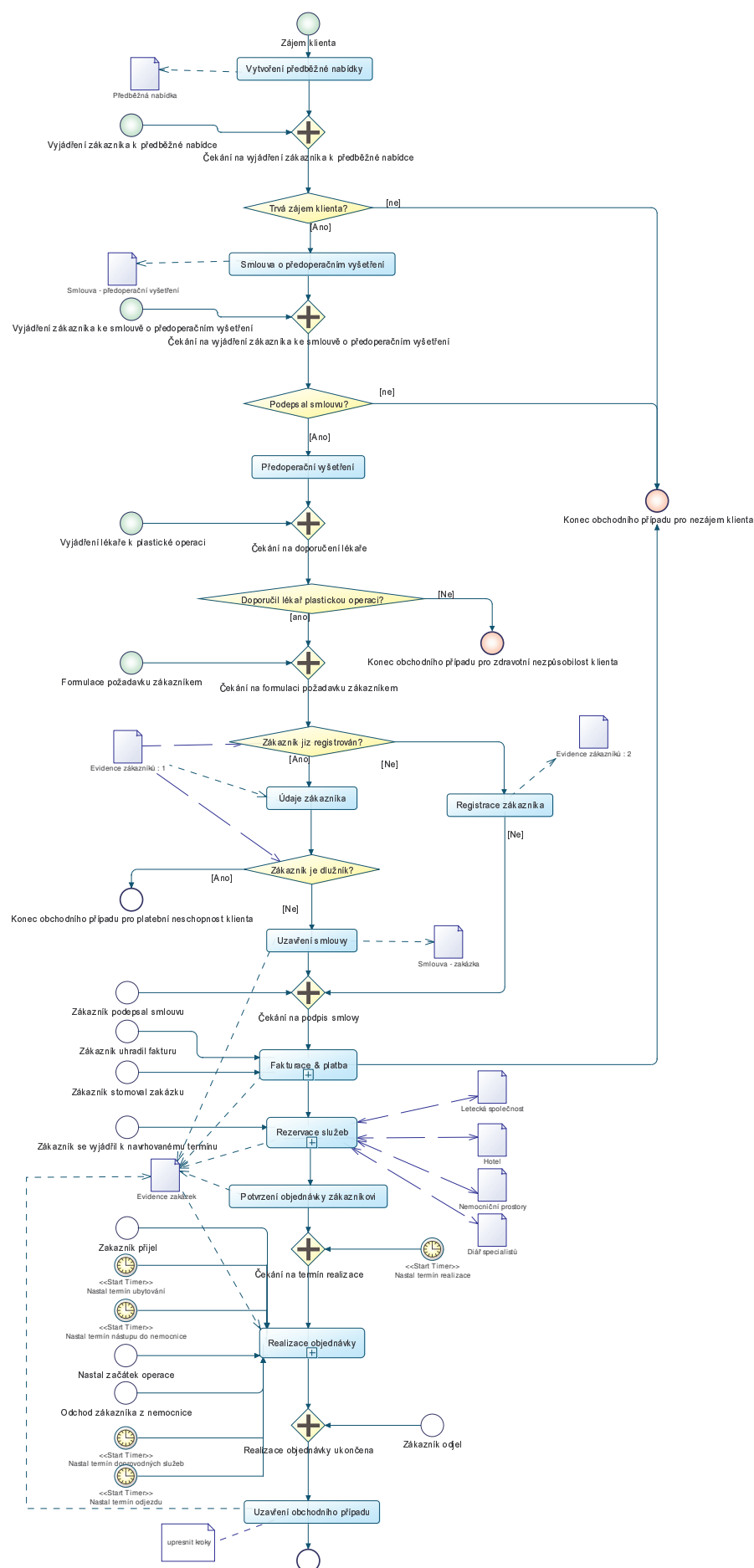
Po akceptaci nabídky (v rámci subprocesu) je informována nemocnice, hotel a letecká společnost a jsou zablokovány (objednány) příslušné termíny dohodnuté v objednávce. Tímto je proces ukončen a dochází k samotnému přiletu zákazníka, jeho ubytování, provedení operace a rekonvalescence. Toto již není cílem tohoto dokumentu. Přilet a ubytování zákazníka bylo vyřešeno zablokováním příslušných termínů. Operace je zajištěna zablokováním příslušného termínu v partnerském lékařském zařízení. Rekonvalescence a kulturní vyžití závisí na volbě příslušného hotelu a místa, kde se bude operace provádět – řešeno v následujících podkapitolách o subprocesech.

Procesy spojené s kulturním vyžitím zde nejsou řešeny. Toto je spojeno s konkrétním hotelem a nemocničním zařízením.

Po provedení realizace objednávky již následuje pouze fakturace a ukončení vztahu se zákazníkem.

Zde uvedený procesní model je jen velice hrubým nástinem celého procesu, téměř každý proces může být rozepsán na nižší úrovni. Slouží spíše jako formální zobrazení procesu vyřízení požadavku klienta.

Grafické vyjádření procesu je na následujícím obrázku:



E.3.4.2 Podproces Rezervace služeb

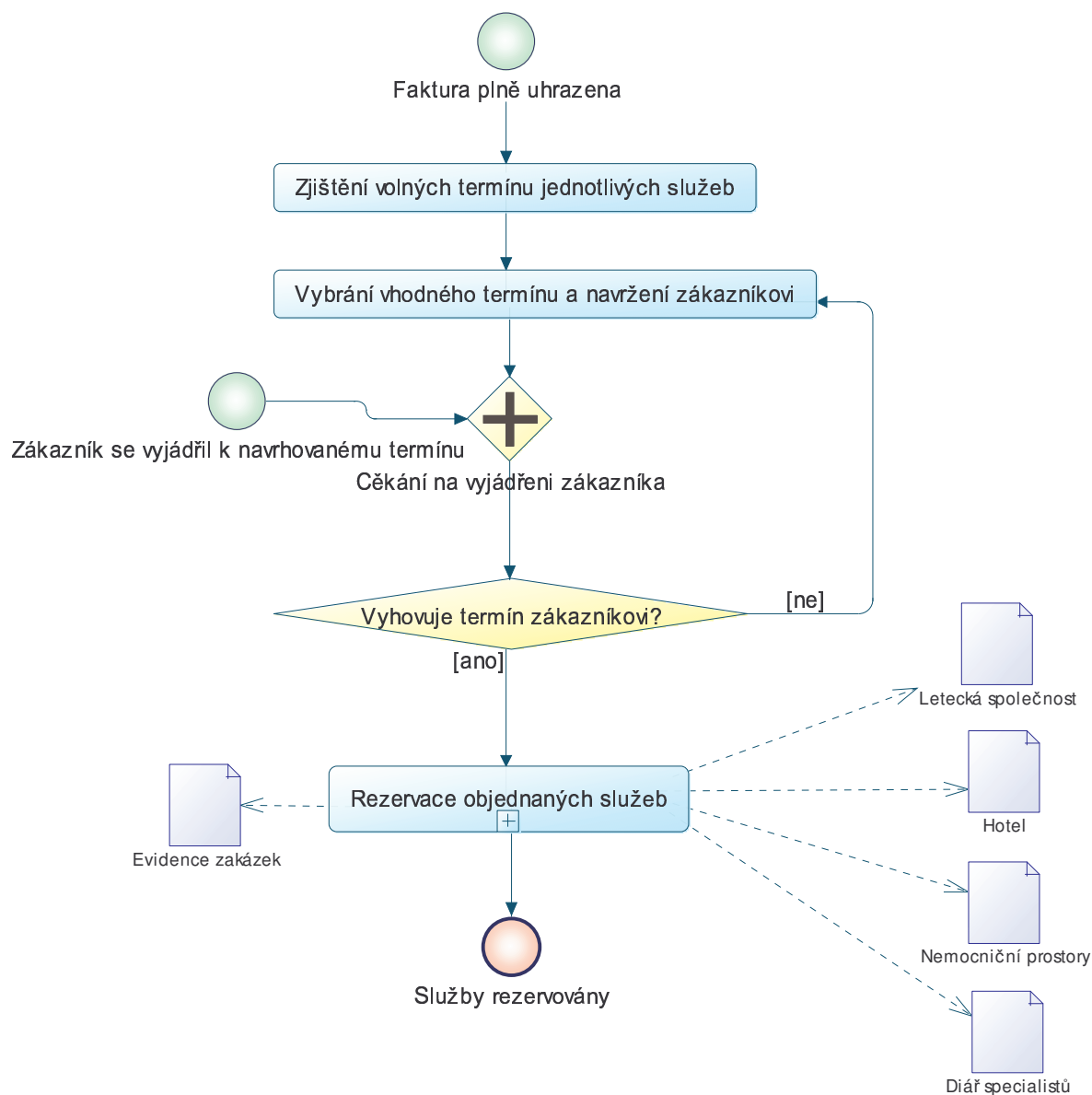
Proces rezervace služeb je zpodrobněním (vyjmutím) jedné části core-procesu. Účelem je zpřehlednit obě grafická provedení.

Rezervace služeb začíná uzavřením smlouvy. Následně je nutné zjistit termíny volných míst a nabídnout je klientovi. Toto je velmi podobné části akceptace v předcházejícím grafu. Zde je to již vybíráno pod konkrétní smlouvou, kdežto ve výše uvedeném procesu byla rozepsána část, která je ještě nezatížená uzavřenou smlouvou. Je zde snazší odstoupení od vzájemných závazků.

Po vytvoření nové (respektive upravení původní) nabídky dochází k její akceptaci. Pokud klient nabídku neakceptuje, může mu být vytvořena jiná nabídka. Pokud nesouhlasí s další nabídkou (respektive jejím opakováním v různých podobách) je proces ukončen. Pokud s nabídkou souhlasí (akceptace je pozitivní) – platí jak v případě upravované nabídky tak i v případě první nabídky, je kontaktována příslušná nemocnice, kde bude zákrok proveden. U upřesnění nabídky je podstatné, že jsou zahrnuty do okolností i kontextové faktory, kterými jsou letecké společnosti a hotelová zařízení, kde bude klient ubytován. Tyto faktory mohou způsobit, že je nabídka odmítnuta a je požadováno její upravení. Nemusí například vyhovovat termín provedení operace, kvalita cílového hotelu nebo způsob dopravy (moc přestupů, pouze druhá třída atd.). Společnost Plastická chirurgie, s.r.o. si tato potřebná data sama stahuje z provozních systému spolupracujících společností (hotely, letecké společnosti, nemocnice). Je-li vybrán vhodný termín, je nabídnut zákazníkovi, který jej může či nemusí schválit. Pokud je vše schváleno, dochází k rezervaci služeb.

Další zpodrobnění části rezervace služeb bude součástí detailní analýzy a návrhu.

Rezervace služeb



E.3.4.3 Podproces Rezervace objednaných služeb

Podproces Rezervace objednaných služeb je zpodrobněním procesu Rezervace služeb uvedeném v předcházející podkapitole. Aktivujícím prvkem procesu je požadavek na službu. Po tomto procesu dochází k činnostem rozpadajícím se do dvou větví. První větev představuje rezervaci specialisty (lékařský personál) a druhá větev představuje rezervaci nemocničních prostorů potřebných pro operaci v partnerském nemocničním zařízení. Z obou větví se ukládají výsledky do příslušných databází. V případě první větve (rezervace lékařského personálu) se ukládají do databáze interních specialistů. Zde ztotožním pojem specialista s pojmem lékařský personál potřebný pro daný lékařský zákrok (plastickou operaci). Výsledky rezervace nemocničního zařízení se promítají do interní databáze konkrétní nemocnice (partnerského zařízení).

V druhé větvi (rezervace nemocničních zařízení) je nutné „protlačení“ informací do všech potřebných informačních systémů v daném informačním zařízení tak, aby byla zajištěna oboustranná spokojenost a bezproblémový chod procesů rezervace v dalších fázích.

Po provedení obou větví následuje synchronizace, která představuje jistou podmínku, že proces nebude pokračovat dále, dokud nebudou splněny obě větve. Toto je nutné z následujících důvodů:

- není možné mít rezervované specialisty (lékařský personál) a nemít rezervované lékařské prostory, kde bude daný operační zákrok proveden,
- není možné, aby výše popsaná situace byla v opačném pořadí – nemáme lékařský personál, ale máme rezervované prostory,
- následná synchronizace vyjadřuje, že proces nepůjde dále, dokud nebudou splněny obě výše uvedené podmínky.

Poté jsou rozepsány jednotlivé operace a oznámeny specialistům. Tímto se rozumí vytvoření určitého reportu, dle investora nejméně čtrnáct dní předem, o rozdělení operačních zákroků, časů jejich konání a složení operačního týmu. Takovýto report bude k dispozici tak, aby si každý z oprávněných zaměstnanců mohl ověřit, kdy a jaké má pracovní povinnosti. Z tohoto vychází, že bude vytvářen jednak v papírové podobě administrativním personálem a současně bude k dispozici na intranetové síti tak, aby k němu měli příslušní pracovníci přístup i v případech, kdy nebudou přítomni v prostorách společnosti Plastická chirurgie, s.r.o.

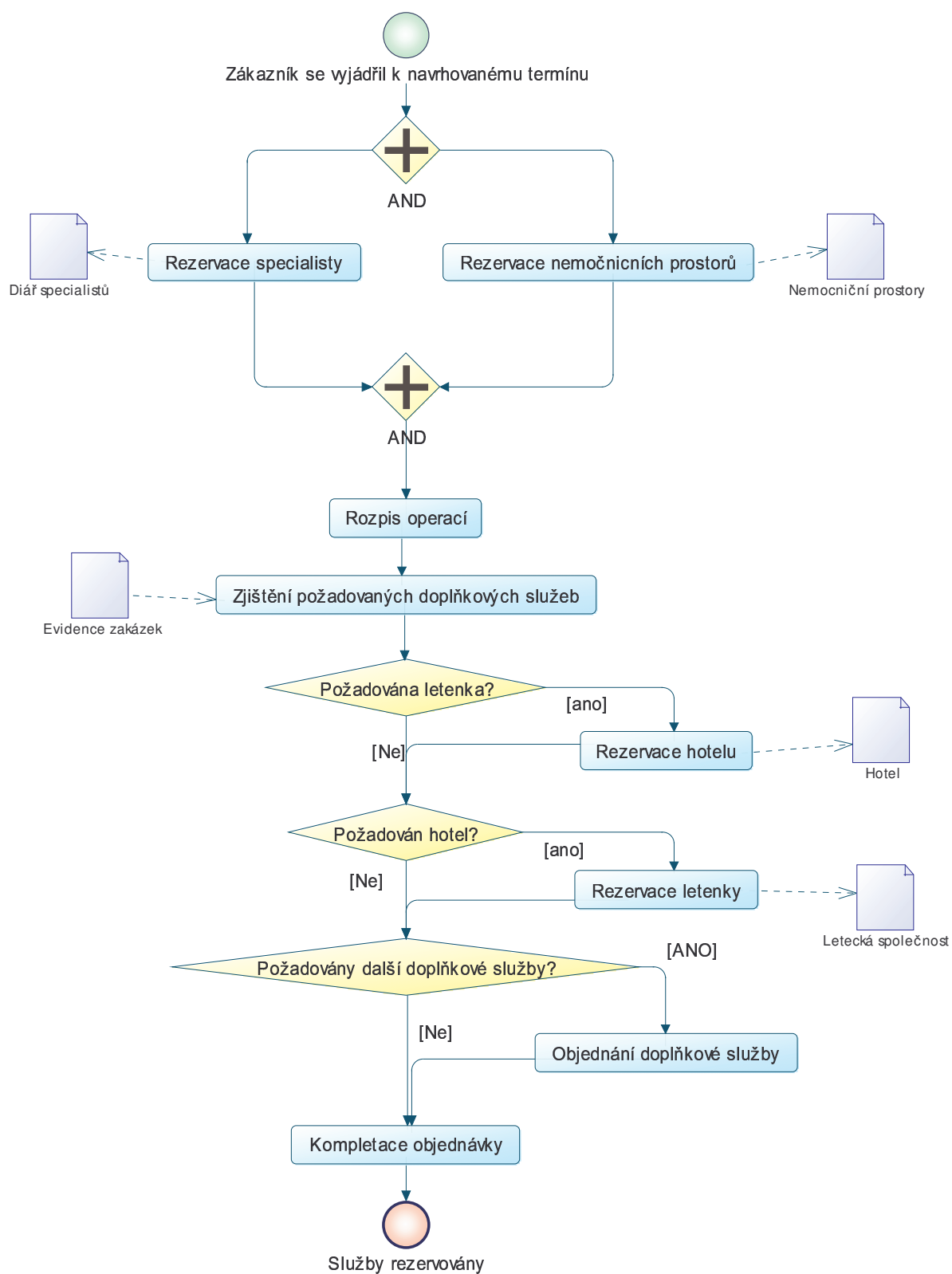
Následuje finální rezervace hotelu a letenek, kdy se příslušné termíny ukládají do databází. Podstata rezervace a objednávání obou služeb je založena na napojení do informačních systémů leteckých společností a hotelových služeb, kterými jsou například systém Amadeus pro letecké společnosti a Accordhotels pro oblast hotelových služeb.

Samozřejmostí je zde možnost nevyužívat námi nabízených doplňkových služeb. To je zejména v případech, kdy bude klientem občan České republiky, který pravděpodobně nebude potřebovat leteckou dopravu. Pro tyto případy je v procesu možnost odmítnutí poskytovaných služeb.

Pokud zákazník služby požaduje, jsou objednány a pak následuje proces kompletace objednávky. V případě, že poskytované služby nejsou požadovány, následuje přímo po výše zmíněné podmínce na požadavek vedlejších služeb kompletace objednávky.

Po kompletaci objednávky tento subprocess končí a vracíme se zpět na vyšší úroveň, kde pokračujeme v dalším průběhu procesu.

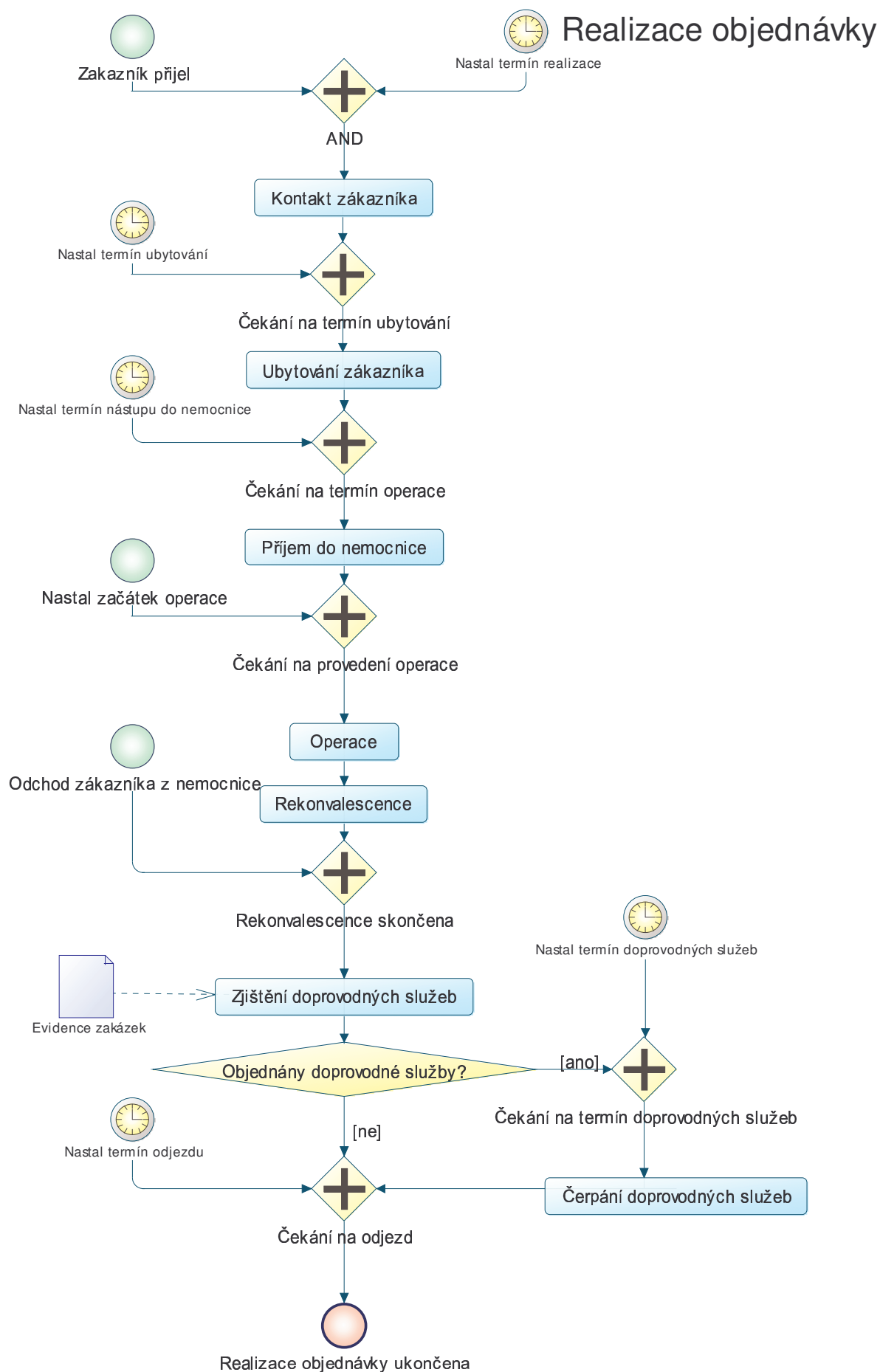
Rezervace objednaných služeb



E.3.4.4 Podproces Realizace objednávky

Proces realizace objednávky následuje po procesu rezervace. Tento proces začíná termínem nástupu na „prázdniny“. Proces postupuje postupně od příletu zákazníka, kdy je kontaktován naším hotelem, který zajišťuje jeho ubytování a odvoz do hotelu z letiště. Po ubytování má příslušné odpočinkové aktivity v závislosti na dohodnuté smlouvě a pak následuje příjem do nemocnice. Odvoz do nemocnice zajišťuje buď hotel nebo společnost Plastická chirurgie, s.r.o. opět v závislosti na uzavřené smlouvě.

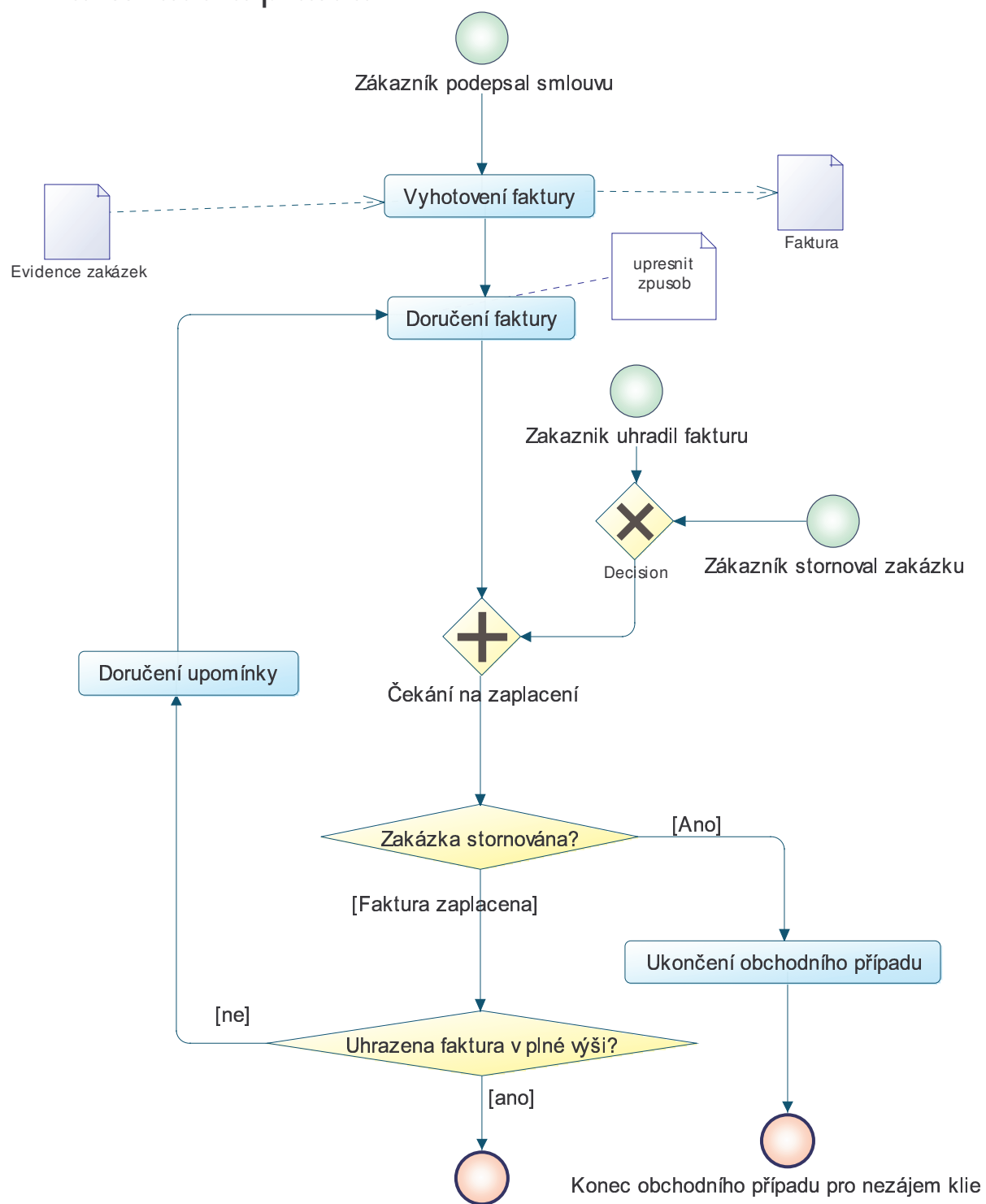
Po příjmu do nemocnice je provedena operace a následuje rekonvalescence. Pokud má klient objednaný další doprovodné služby, pak jsou realizovány a na jejich konci je odlet klienta domů. Pokud tyto služby objednaný a zaplacený nemá následuje rovnou po nezbytné rekonvalescenci odlet klienta domů.



E.3.4.5 Podproces Fakturace a platba

Proces vyjadřuje postup, který bude realizován při placení zakázky zákazníky. Podstatné v tomto procesu je část týkající se doručení faktury, který bude moci být realizován několika způsoby (pošta, elektronicky, kurýr atd.). Další důležitou částí je rozhodování, zda je faktura uhrazena. Pokud je faktura řádně uhrazena, je fakturace ukončena a je možné pokračovat dalšími částmi, které jsou uvedeny v core-procesu. Pokud nastane situace, že faktura řádně uhrazena není, dojde k aktivaci procesu doručení upomínky. Cílem této funkcionality je upozornit zákazníka, že jím objednané služby je třeba zaplatit. Na doručení upomínky může zákazník opět reagovat dvěma způsoby. V lepším případě zákazník pouze zapomněl fakturu uhradit a uhradí ji nyní a v důsledku toho se dostáváme zpět do větve rozhodování, zda byla faktura řádně uhrazena. Pokud dojde k situaci, že zákazník fakturu stornuje, dochází automaticky k ukončení obchodního případu z pozice klienta, který o něj nemá zájem.

Fakturace a platba



E.3.5 Diagram tříd – Class diagram

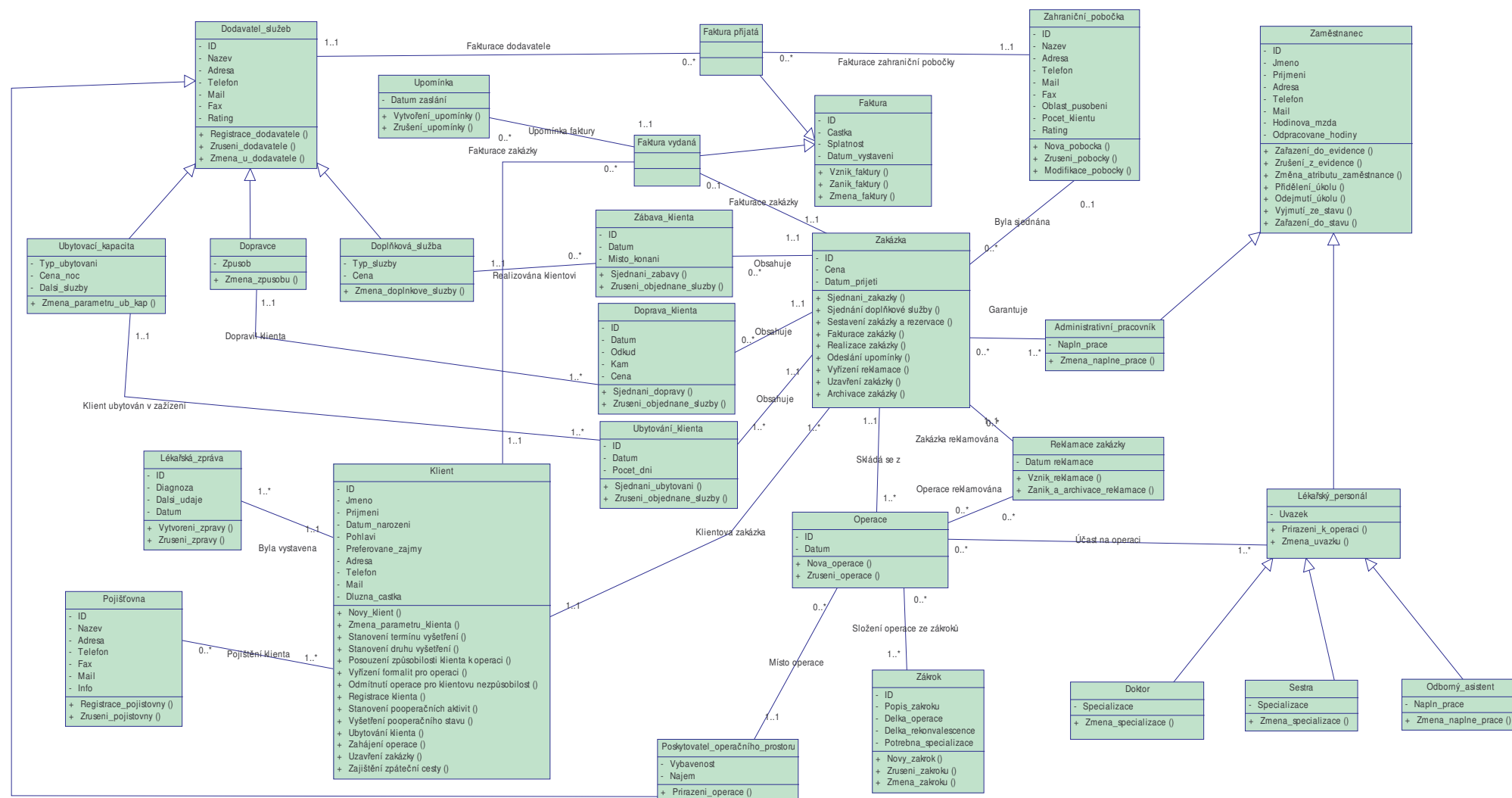
Model tříd vznikl detailnějším rozpracováním vybrané části globálního objektového (datového) modelu z globální analýzy. Z hlediska informačního systému zobrazuje tento model statický pohled na data, užívaná ve zvolené části informačního systému včetně vzájemných vazeb a hlavních metod pro práci s nimi.

Model je však především detailním konceptuálním modelem reálného systému organizace. To znamená, že jednotlivé třídy představují zobecněné objekty z reálného života, jejich základní klasifikace (generalizace) a popis základních principiálních vazeb (asociací) mezi nimi. Jednotlivé popsané atributy tříd tak představují reálné vlastnosti objektů, které jsou pro podporu činnosti organizace informačním systémem podstatné. Jednotlivé popsané metody tříd pak představují činnosti (akce), které jsou reálnými objekty prováděny, nebo jsou na nich páhány, a to takové, které jsou pro podporu činnosti organizace informačním systémem podstatné – tedy ty, které korespondují s událostmi, vedoucími k jednotlivým činnostem ve výše popsaných business procesech.

Existence objektů, jejich principiálních vazeb, důležitost jejich atributů a k nim vázaných činností (metod), jsou esenciálními důvody k realizaci datové základny informačního systému a definují její nezbytný obsah. Jedná se o definici toliko obsahu, plynoucího ze samotné podstaty problému, a to bez ohledu na různé možné další důvody, pocházející ze specifik situace (použité technologie, charakteristik okolí systému, organizačních a dalších neobecných rysů organizace²⁴). Tyto další faktory technologického a implementačního charakteru, jakkoliv důležité, mohou být, a budou, zohledněny až v následných fázích designu systému a jeho implementace.

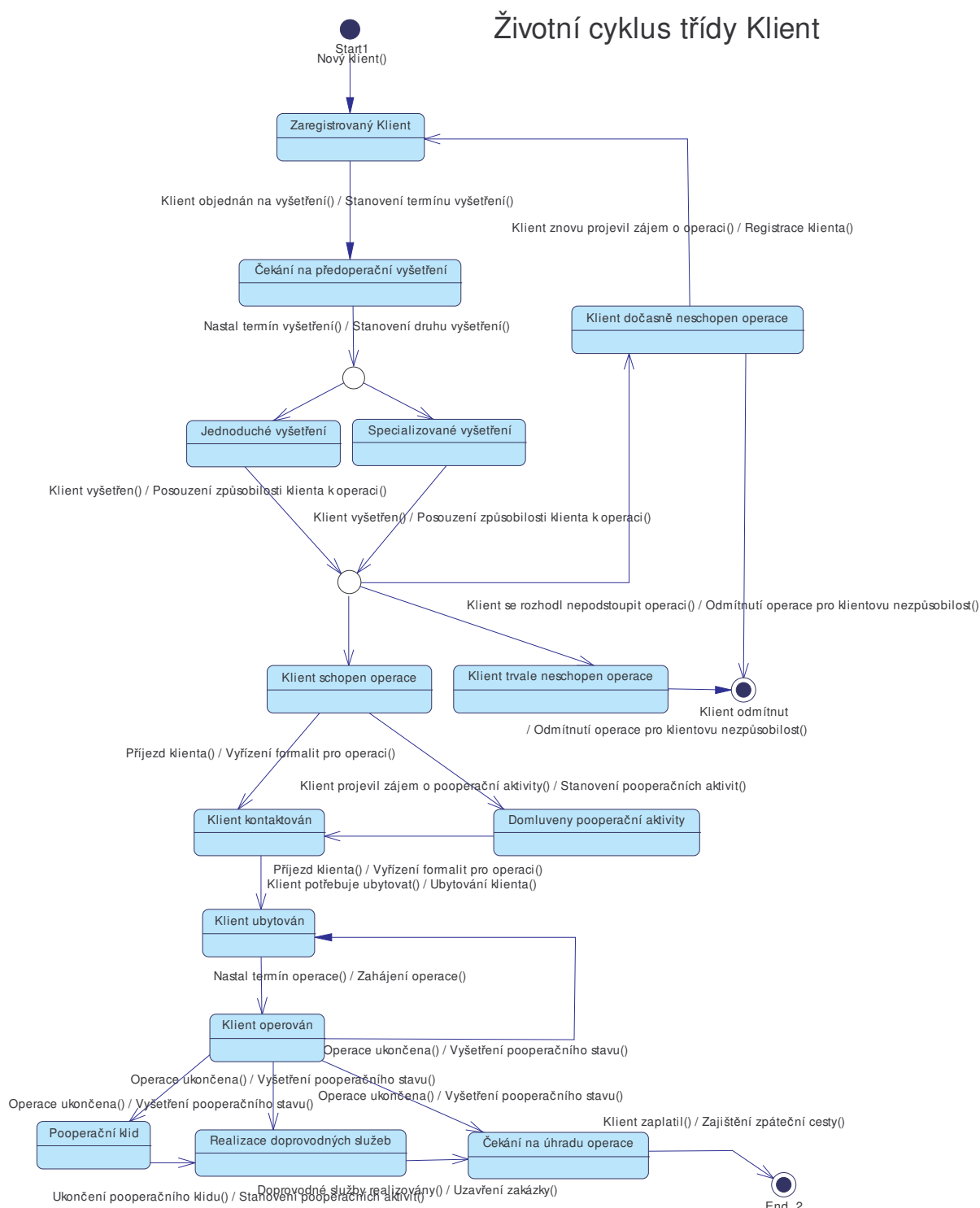
Na diagram tříd navazují popisy životních cyklů klíčových objektů modelu v podobě stavových diagramů. Tyto podrobněji zasazují jednotlivé metody do postupu vnitřní logiky vývoje objektu v čase.

²⁴ například nedostatku kvalifikovaných pracovníků, zabraňujícímú využití vhodné technologie apod.



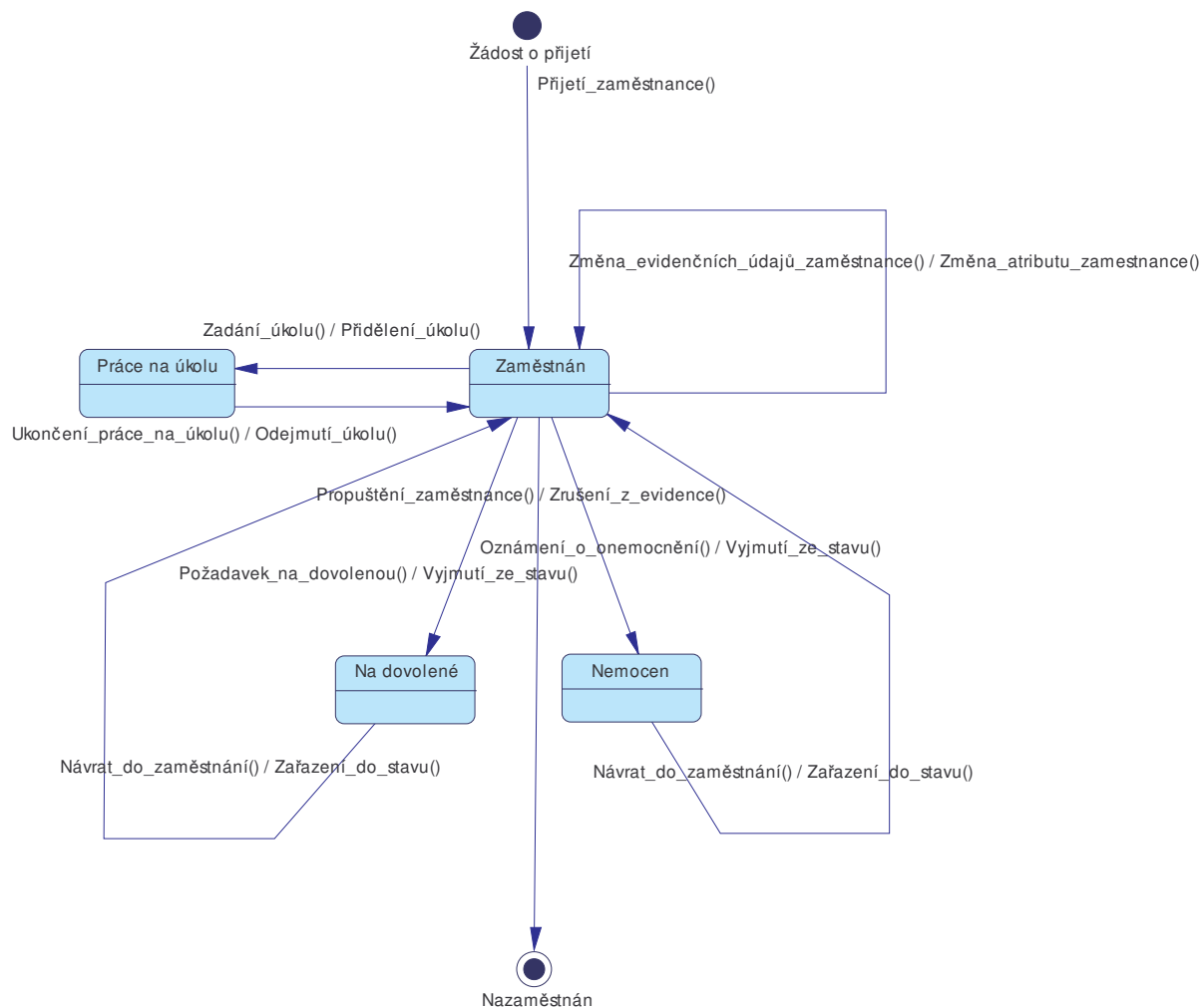
E.3.6 Životní cykly klíčových tříd – stavové diagramy (State Chart)

V této kapitole uvádíme popis životních cyklů klíčových tříd v podobě stavových diagramů. Stavový diagram (State Chart) je nástrojem specifického popisu procesu – v podobě stavů a přechodů mezi nimi. Jedná se o procesní popis, nicméně popisuje proces zcela jiného charakteru, než jsou výše uvedené business procesy. Smyslem popisu životního cyklu objektu je především vymezení základních zákonitostí (omezení), jimiž se veškeré jednání tohoto objektu (či jednání na něm, manipulace jím) musí řídit.



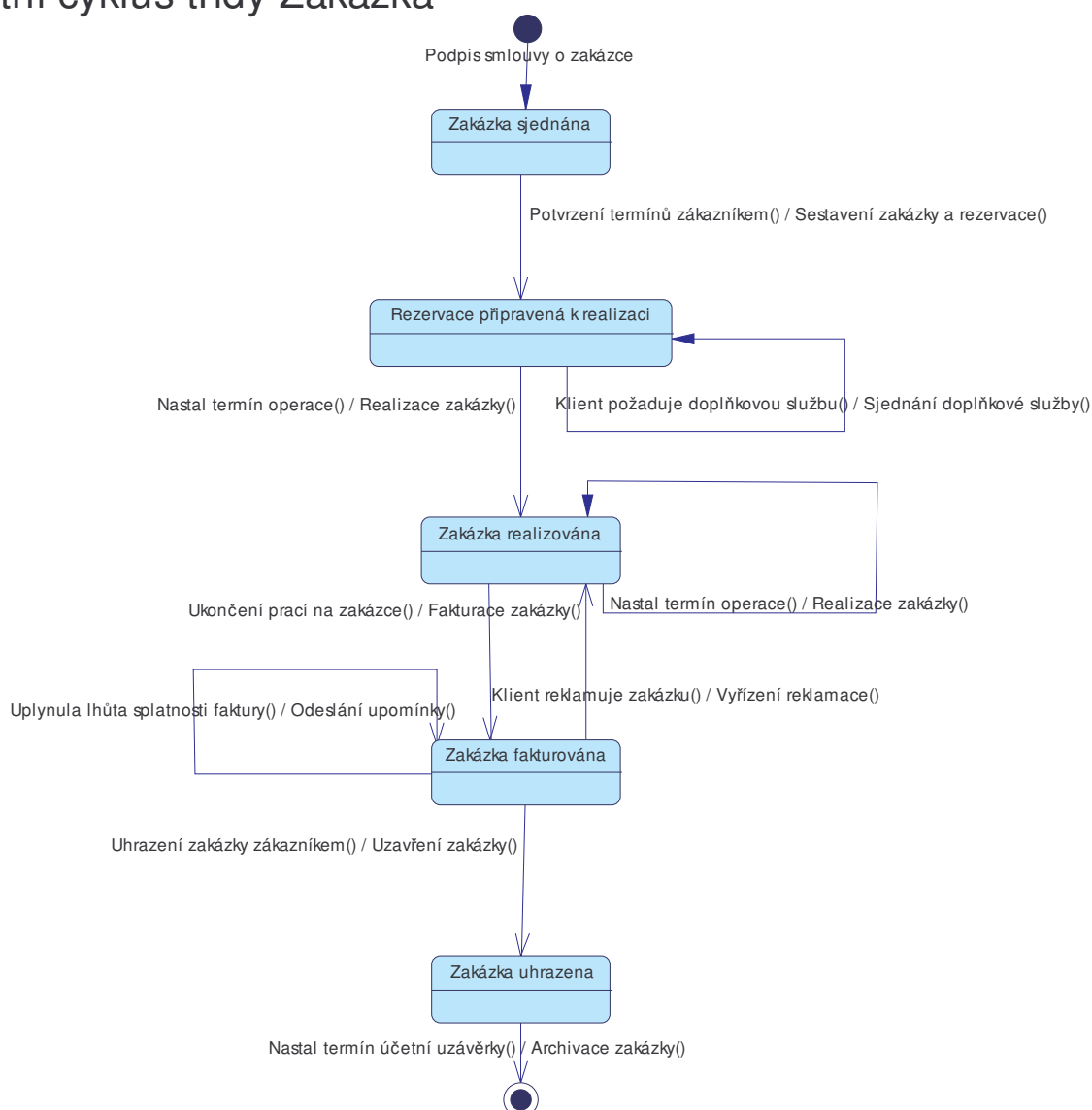
Životní cyklus třídy Klient procesně popisuje základní zákonitosti vývoje tohoto objektu v čase. Vymezuje podstatné stavy a možné přechody mezi nimi.

Životní cyklus třídy Zaměstnanec



Životní cyklus třídy Zaměstnanec procesně popisuje základní zákonitosti vývoje tohoto objektu v čase. Vymezuje podstatné stavy a možné přechody mezi nimi.

Životní cyklus třídy Zakázka



Výše uvedený model tříd v podobě diagramu tříd a popisu životních cyklů některých, procesně významných, tříd ukazuje smysl takové soustavy diagramů pro specifikaci konceptuálního modelu. Popisy životních cyklů vznikají především na základě znalosti reality a jejích zákonitostí – jsou procesním vymezením těchto zákonitostí. Přitom forma životního cyklu popisu přirozeně vede k promyšlení všech relevantních událostí, hrajících v životě daného objektu, a to včetně událostí okrajových, často zapomínaných, spojených se vznikem, či zánikem objektu (např. všechny možnosti, jak získat nového klienta, všechny možnosti ukončení zaměstnaneckého poměru, akce vázané k ukončení zakázky (archivace) apod.).

Životní cykly objektů a jejich kontext, daný vztahy k ostatním objektům, se musí vzájemně doplňovat. Stavy jednotlivých objektů docházejí změn vždy na základě nějaké události a často (resp. většinou) v souvislosti s jinými objekty – tedy daná událost tak má význam pro několik různých objektů. Tyto, více objektům společné, události pak musí odpovídat popsáním vazbám (asociacím) mezi těmito objekty. Tak například sjednání nové služby v rámci zakázky znamená jednak vznik nové sjednané služby (instance objektu příslušné ho typu služby), jednak v životě zakázky přechod ze stavu Rezervace připravená k realizaci do stavu téhož. Podobně reklamace zakázky, kromě vzniku instance objektu Reklamace, znamená u zakázky přechod ze stavu Zakázka fakturována zpět do stavu Zakázka

realizována (reklamace vrací zakázku zpět do hry, což je fakt, na nějž lze při tvorbě modelu tříd snadno zapomenout, ovšem při uvažování o všech možnostech konce života zakázky je nepřehlédnutelný - nezávislé popisování života třídy nás zde upozorňuje na důležité skutečnosti, které v diagramu tříd ještě nejsou popsány). Takto se různé pohledy na stejné věci (zde například nahlížení jedné události v kontextu života různých objektů) vzájemně doplňují a podmiňují a podněcují tak k dosažení úplného a ze všech podstatných hledisek korektního pohledu na modelovanou realitu.

U přechodů mezi stavy objektů, vyvolaných společnou událostí, je namístě také věnovat pozornost podrobnější charakteristice vztahu obou dotčených objektů. Ta by měla odpovídat adekvátnímu pojetí této události v životě třídy. Obecně by např. mělo platit, že asociace o kardinalitě 1 musí odpovídat jednorázovému výskytu příslušné akce v životě třídy na této straně asociace, kardinalitě n pak odpovídá vícenásobný výskyt příslušné akce. Podobně parcialitě vztahu (kardinalitě 0) musí odpovídat jistá podmíněnost příslušné akce v životě třídy. Tak například fakt, že tentýž klient může mít více zakázek a více lékařských zpráv, se projevuje v jeho životním cyklu existencí různých možností opakování akcí jako je vyšetření, stanovení termínu operace apod. Parcialita asociace mezi Klientem a Fakturou pak nutně vyžaduje možnost různých alternativních cest v životě klienta mezi jednotlivými jeho stavy (zde konkrétně například možnost klienta odmítnout a rozjednanou zakázku tak vůbec nerealizovat).

Něco podobného platí i pro generické objekty, u nichž popisujeme jejich životní cyklus. Pokud jsou jednotlivé specializace generického objektu (zde například Zaměstnanec, vyskytující se ve specializacích Administrativní pracovník a Lékařský personál) procesně odlišné – tedy odlišné ve svých životních cyklech, potom musí být tyto životní cykly popsány u každé specializace zvlášť a životní cyklus generického objektu pak popisuje pouze dynamiku, společnou všem specializovaným pod-objektům. V životním cyklu takového generického objektu pak musí existovat specifický stav pro každou jednotlivou variantu (specializaci) objektu, za nímž se skrývá samostatně popsáný život specializovaného pod-objektu. Životní cyklus generického objektu pak popisuje možné cesty jak se stává objekt specializovaným. To může být významné pro, často potřebný, popis změny specializace (zde by bylo například možné popsat, jakým mechanismem se odborný asistent může stát doktorem, aniž bychom jej museli přestat považovat za jednoho a téhož zaměstnance)²⁵.

Principiální různost a současně úzká provázanost těchto dvou pohledů na existenci objektů tak slouží jako universální nástroj dosažení potřebné úplnosti, komplexnosti a současně přesnosti modelu, jež jsou nezbytné pro důkladnou specifikaci všech podstatných východisek dalšího vývoje informačního systému – následného provázání s procesními modely, odvození funkčnosti a posléze designu programového systému.

²⁵ V tomto příkladu není popisovaný vztah životního cyklu generického objektu k životům jeho specifických variant ilustrován, protože z hlediska této úlohy není nutné jednotlivé specializace procesně rozlišovat. Nicméně tato potřeba bývá velmi aktuální, jak ukazuje například celá teorie CRM (Customer Relationship Management), kde klíčovým pojmem je životní cyklus zákazníka, popisujícím jak metamorfuje ze stavu potenciálního (hýčkaný) do stavu aktivního (dojený) a zpět, přičemž každému tomuto jeho stavu odpovídá specifický životní pod-cyklus (pod-životní cyklus), určující specifické akce a styl chování k němu, odpovídající danému stavu.

Konzistence modelů

Uvedená konzistenční tabulka demonstruje konzistenci modelů detailní analýzy: procesního modelu, stavového diagramu a diagramu datových toků. Jelikož se všechny modely, uvedené v jedné řádce tabulky, týkají téže činnosti (resp. jsou vyvolány toutéž událostí), je nutné, aby všechny události, akce a stavy měly svůj ekvivalent ve všech modelech. Proto je v konzistenční tabulce pro každou událost, akci i stav uveden její název použitý v každém z popisovaných modelů. Tabulka je užitečná jak pro zjištění názvů událostí, akcí a stavů v jednotlivých modelech, tak i pro kontrolu konzistence, tedy toho, že všechny modely popisují danou činnost stejným způsobem.

Č.	→						→	
	Událost			Akce			Stav	
	V procesu	V životním cyklu třídy (STD)	Ve vstupu IS (DFD)	V procesu	V životním cyklu třídy (STD)	Ve funkci IS (DFD)	V procesu	V životním cyklu třídy (STD)
1.	Formulace požadavku zákazníkem	Nový klient	1_Registrace zákazníka zahraničním partnerem	Registrace zákazníka	Přidání klienta	Registrace zákazníka	1_Čekání na uzavření smlouvy	Zaregistrovaný klient
2.	Formulace požadavku zákazníkem	Stávající klient	8_Lékařská zpráva od PNZ	Údaje zákazníka	Vyhledání klienta	Registrace zákazníka	2_Čekání na uzavření smlouvy	Vyhledaný klient
3.	Zákazník podepsal smlouvu	Podpis smlouvy o zakázce	2_Zákazník	Uzavření smlouvy	Nová zakázka (konstruktor)	Rezervace zakázky	Čekání na termín realizace	Zakázka sjednána
4.	Zákazník se vyjádřil k návrhovanému termínu	Akceptování data operace zákazníkem	3_Akceptace termínů zákazníkem	Rezervace objednaných služeb	Potvrzení termínů zákazníkem	Rezervace zakázky	Čekání na vyjádření zákazníka	Rezervace připravená k realizaci
5.	Zákazník přijel + nastal termín nástupu na pobyt	Příjezd klienta	4_Příjezd zákazníka	Kontakt zákazníka	Převzetí	Realizace	Čekání na termín operace	Kontaktovaný klient
6.	Nastal termín ubytování	Potřeba bytování	9_Termín ubytování zákazníka	Ubytování zákazníka	Ubytování	Realizace	Čekání na termín operace	Ubytovaný klient
7.	Nastal termín nástupu do nemocnice	Naplánovaná operace	5_Dostavení zákazníka k operaci	Příjem do nemocnice	Provedení operace	Realizace	Čekání na provedení operace	Provedená operace
8.	Zákazník uhradil fakturu	Uhrazení zakázky zákazníkem	6_Platba zákazníkem	Doručení upomínky	Uhrazení zakázky	Finance	Čekání na platbu	Zakázka uhrazena
9.	Zákazník uhradil fakturu	Uhrazení zakázky zákazníkem	6_Platba zákazníkem	Zjištění volných termínů	Uhrazení zakázky	Finance	Konec fakturace a Čekání na vyjádření zákazníka	Zakázka uhrazena
10.	Zákazník stornuje	Stornování zakázky	11_Storno zakázky	Ukončení obchodních	Stornování zakázky	Rezervace zakázky	Čekání na ukončení	Zakázka stornována

	zakázku	klientem		o případu			obchodního případu	
11.	Odchod zákazníka z nemocnice	Odeslání pacienta na rekonvales cenci	7_Zákazník opustil nemocnici	Zjištění doprovod ných služeb	Kontrola	Realizace	Čekání na termín doprovodných služeb	Propuštění pacienta
12.	Odchod zákazníka z nemocnice	Odeslání pacienta na rekonvales cenci	7_Zákazník opustil nemocnici	Zjištění doprovod ných služeb	Kontrola	Realizace	Čekání na termín odjezdu	Propuštění pacienta
13.	Nastal termín doprovodných služeb	Zjištění doprovodných služeb	12_Termín plnění služeb	Cerpaní doprovod ných služeb	Cerpaní	Realizace	Čekání na termín odjezdu	Zjištěny doprovodné služby

Tabulka E 11 Konsistence modelů

K zajištění konsistence modelů je velmi důležitá podpora nástroje CASE. Power Designer tuto podporu poskytuje jednak ve formě standardních možností kontroly konsistence (consistency checks) jak uvnitř modelu, tak mezi nimi (na základě vytváření vazeb mezi modely pomocí tzv. rozšířených závislostí (extended dependencies)). Vzhledem k tomu, že zajištění konsistence je kriticky důležitým aspektem metodického postupu a de facto tvoří jeho hlavní hodnotu, je velmi důležité v Power Designeru používat jak standardní kontroly konsistence, tak i kontroly specifické této metodice, doplněné do nástroje v podobě tzv. „custom checks“. K tomu je ovšem zapotřebí především vytvářet vazby mezi modely pomocí rozšířených závislostí, jak je popisováno na příslušných místech této metodiky.

E.4 Design systému – Hardwarová a softwarová architektura

Návrh informačního systému pro Plastickou chirurgii, s.r.o. s sebou přináší řadu specifických problémů a z nich vyplívajících požadavků a omezení.

Nejprve si je třeba uvědomit, že se v našem případě jedná o subjekt spadající do kategorie malých a středních podniků, který si může dovolit investovat do IS/ICT pouze omezené množství prostředků. Přitom klade na IS systém široké spektrum požadavků.

Shrnutí základních požadavků na IS:

- podpora běžných administrativních funkcí
- elektronická forma komunikace, včetně výměny dokumentů v elektronické podobě
- propojení s informačními systémy vnějších subjektů (nemocnice, cestovní agentury, ubytovací zařízení, atd.)
- podpora partnerských nemocničních zařízení (PNZ), sledování historie klienta
- řízení volných kapacit nemocnic a specialistů
- flexibilita pro další rozšiřování funkčnosti i výkonu

Od informačního systému se očekává, že bude podporovat běžné administrativní činnosti (účetnictví, vedení majetku apod.). Tato oblast je v IT odvětví již značně zaběhnutá, aplikace dostupné na trhu mají za sebou dlouhé období existence a jsou prověřeny časem. Nelze tedy očekávat dosažení žádné konkurenční výhody.

Na druhou stranu předmětná oblast podnikání představuje na trhu dosud ojedinělé požadavky kladené na IS a její integrace do současných SW systémů bude představovat vysokou přidanou hodnotu a může vést ke značné míře konkurenceschopnosti na trhu poskytování služeb plastické chirurgie.

Jedním ze zásadních požadavků je bezesporu vedení veškeré dokumentace v elektronické podobě a to hlavně při komunikaci s partnerskými nemocničními zařízeními (PNZ) v zahraničí. Tato zařízení nepodléhají české legislativě, je na místě předpokládat velkou roztržičnost legislativních požadavků na elektronickou komunikaci. Každý subjekt nacházející se v jiné zemi bude podléhat jiné legislativě týkající se ochrany osobních údajů a jejich poskytování elektronickou cestou, včetně požadavků na zabezpečení této formy komunikace. V některých zemích tato problematika nemusí být řešena vůbec.

Při propojování IS bude nutné analyzovat konkrétní informační systém u vnějších subjektů a jeho možnosti propojení s navrhovaným IS. Největší přínos propojení pak lze vidět u propojení s IS nemocnic a tudíž v efektivním řízení volných kapacit nemocnic a specialistů.

Analýza požadavků byla provedena v GAN, a proto zde na ní jen odkazujeme.

E.4.1 Softwarová architektura

Architektura IS bude vrstvená - třívrstvá (resp. čtyřvrstvá pro zahraniční uživatele):

- DB server jako centrální úložiště dat sloužící zároveň jako komponenta, která umožňuje integrovat nezávislé komponenty aplikační logiky
- Aplikační server zprostředkující funkcionalitu IS
- Různí klienti pro různé skupiny uživatelů:
 - tlustí intranetoví klienti

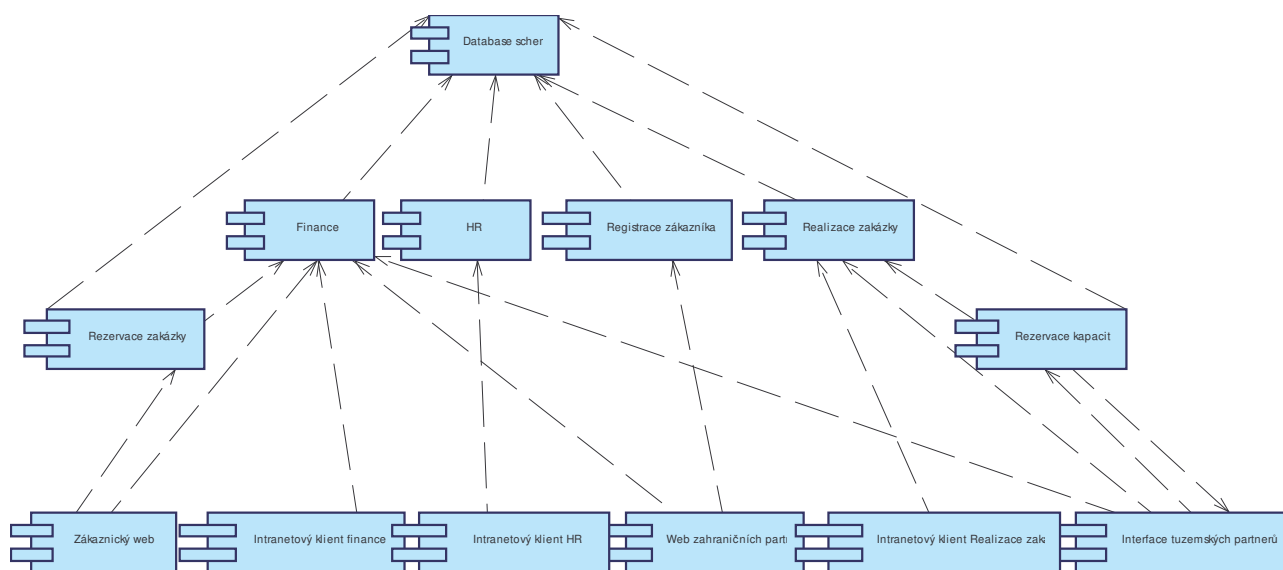
- webový server
- komponenty pro integraci s tuzemskými partnery
- tencí klienti - webové prohlížeče zákazníků a zahraničních partnerů

Takto zvolená architektura je dostatečně flexibilní a umožňuje vyměnit některé své části v případě, že jejich výkon nebude dostačující. Zvyšující se nároky na objem dat a výkon lze očekávat především u databázového serveru, protože na tomto serveru jsou všechny ostatní komponenty přímo či nepřímo závislé. Také aplikační server lze v budoucnu rozšiřovat o další komponenty informačního systému, upgradovat jeho hardware nebo distribuovat aplikační komponenty na více fyzických strojů pro větší výkon.

Tlustí klienti poskytují vyšší produktivitu práce než webové uživatelské rozhraní. Proto jsou použity pro intranetové uživatele - různé skupiny zaměstnanců.

Naopak externím uživatelům IS - zákazníkům a zahraničním partnerům - je funkcionality IS zpřístupňována pomocí webového rozhraní. Důvodem je především to, že tyto externí uživatele lze jen stěží nutit k instalaci speciálního softwaru a vzhledem ke frekvenci jejich práce s IS není rozdíl v produktivitě práce u těchto skupin uživatelů dostatečným argumentem.

- *Komponenty pro integraci s tuzemskými partnery budou pravděpodobně jiné pro každou partnerskou organizaci, jelikož nelze předpokládat, že partnerské organizace budou mít stejné informační systémy.*



Obrázek E 13 Diagram komponent

E.4.2 Odvození diagramu komponent z DFD

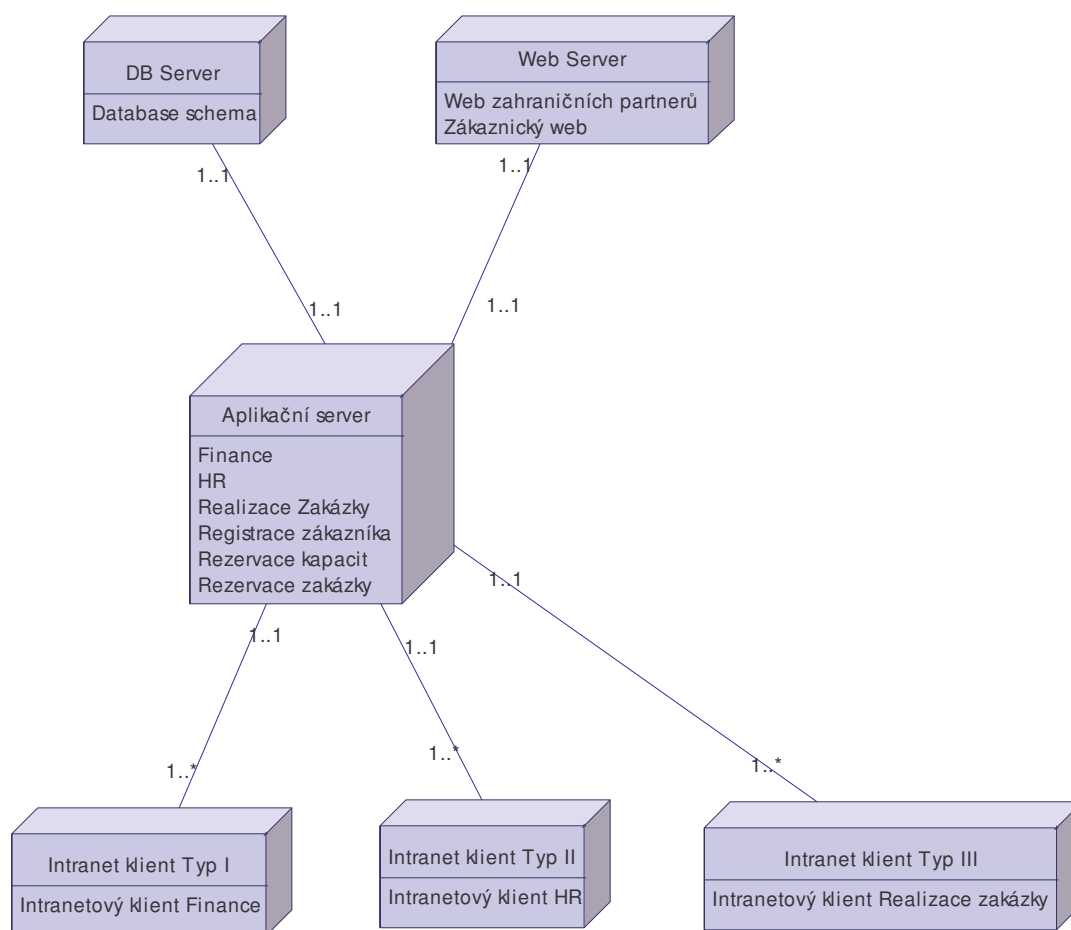
Diagram komponent znázorňuje hlavní komponenty IS a jejich závislosti. Byl odvozen z DFD úrovně 0 tak, že funkce úrovně 0 představující nějakou funkcionalitu systému jsou převedeny na komponenty v diagramu komponent. Tyto komponenty tuto funkcionalitu implementují. Data story z DFD jsou v diagramu komponent representovány databázovým schématem informačního systému, na němž jsou závislé všechny komponenty, které zapisovaly nebo četly z datastorů. Třetí vrstva klientských komponent byla odvozena od terminátorů v DFD tak, že každý terminátor - skupina uživatelů - bude používat jednu

klientskou komponentu ke komunikaci s IS. Výjimkou jsou tlustí klienti pro zaměstnance, kterých bude několik pro různé skupiny zaměstnanců, i když v DFD byly všechny skupiny zaměstnanců reprezentovány jediným terminátorem.

Závislosti mezi komponentami střední vrstvy odpovídají přímým dataflow z DFD. Nepřímé dataflow jsou reprezentovány závislostí na databázi. Závislosti mezi klientskými komponentami a střední vrstvou odpovídají dataflow mezi funkcemi a terminátory.

Hardwarová architektura

Výše uvedené komponenty softwarové architektury budou umístěny na fyzické stroje podle následujícího diagramu:



Obrázek E 14 Diagram rozmištění

Hardwarová architektura se skládá ze tří serverů: databázového, aplikačního a webového. Jejich rozdělení na tři fyzické stroje umožňuje snazší upgrade jednotlivých serverů a u webového serveru zjednodušuje správu sítě, jelikož k tomuto serveru jako jedinému bude možné přistupovat z vnějšku organizace. Dále diagram znázorňuje různé počítače zaměstnanců, které budou přes intranet komunikovat přímo s aplikačním serverem.

E.5 Závěrečná poznámka k příkladu

Příklad postupu vývoje informačního systému, uvedený v této kapitole, poměrně rozsáhle popisuje použití jednotlivých metod a technik, předepsaných metodikou, a to způsobem, předepsaným metodikou. Přes svůj rozsah ilustruje pouze některé fáze postupu vývoje – je zaměřen na analýzu a částečně na navazující design. Z analýzy pokrývá jak fázi globální (zaměřenou na celek), tak detailní (zaměřenou na dílčí detail).

Nepokrývá jednak fáze předcházející globální analýze – nezabývá se specificky strategickými úvahami na úrovni informačního systému, jednak fáze následné – implementaci, instalaci, ani zavedení systému do provozu a vůbec už ne provoz systému. Tyto, příkladem nepokryté fáze postupu vývoje IS, jsou velmi odlišné od fází analýzy a designu jak v metodách a nástrojích, tak i ve způsobu jejich řízení, plynoucího především z jejich charakteru.

Slovník pojmů

Pojem	Anglicky	Význam
Analýza požadavků	Requirements Analysis	Analýza <u>požadavků</u> je analytická činnost probíhající od začátku analýzy systému až po ukončení provozu systému. Tato činnost spočívá v získávání, třídění a ověřování relevantnosti požadavků na systém a definování následných akcí pro uspokojení relevantních požadavků – především změn systému anebo procesů. Systematický proces s cílem objevení podstaty – původu požadavku.
Asociace	Association	Obecná vazba mezi třídami v modelu tříd.
Data Store	Data Store	Abstrakce jakéhokoliv způsobu uložení dat v systému. Třída typu (stereotypu) <<datastore>> ve <u>funkčním modelu</u> .
Datový model	Data model	Statický strukturální model. Popisuje existenci objektů a jejich vzájemných vazeb, včetně jejich atributů.
Datový tok	Data Flow	Abstrakce jakéhokoliv způsobu přesunu dat v systému. Asociace typu (stereotypu) <<dataflow>> ve <u>funkčním modelu</u> .
elementární Datastore	Elementary Datastore	Datastore, u kterého není objektivní důvod k dekompozici do podrobnější struktury sub-datastorů. Na úrovni elementárních datastorů jsou formulována základní konsistenční pravidla, týkající se tohoto elementu modelů.
EMD	Extended Model Definition	Mechanismus (způsob) rozšíření modelu v Power Designer.
Funkce	Function	Jednotka chování systému. <ul style="list-style-type: none"> V širším pojetí zahrnuje jak abstraktní obsahovou jednotku chování, tak i její technologická, implementační, uživatelská a jiná specifika. V užším pojetí jde o základní prvek <u>funkčního modelu</u> (viz definici) - třída typu (stereotypu) <<function>> ve <u>funkčním modelu</u>.
Funkční model	Function model	Model funkcí informačního systému a jejich vazeb v podobě datových toků.
IS	Information System	Informační systém Systém podpory řízení organizace informacemi.
IT	Information Technology	Informační technologie Technologie, související se zpracováním informací.
IS/IT	Information System / Information Technology	Informační systém / Informační technologie Synonymum pro informační systém, zdůrazňující jeho propojení s informačními technologiemi. Jednota informačních technologií a jejich významu, daná jejich použitím v realizaci informačního systému.
Komponenta	Component	Souhrn programových částí, tvořících celek s definovanými odpovědnostmi
Link	Link	Vazba mezi instancemi tříd v modelu objektů. Instance asociace z modelu tříd.
Model objektů	Object model	Model instancí tříd objektů. Pomocný model k modelu tříd.

Pojem	Anglicky	Význam
Model tříd	Class model	Model zahrnující objekty, vystupující v business procesech systému v podobě třídy.
Objekt	Object	Součást systému, která má vztah k ostatním.
Požadavek	Requirement	<p>Jednotka potřeby funkcionality systému.</p> <p>Pojem může mít význam:</p> <ol style="list-style-type: none"> 1. užší - uživatelského požadavku / problému (požadavek, kladený uživatelem na systém), 2. širší - obsahového požadavku (obecnější, širší pojetí, jakýkoliv důvod k potřebě jisté funkcionality. Kromě fyzického požadavku uživatele může být tímto důvodem např. i výsledek analýzy podle pravidel a technik metodiky, nebo formální důvod kladený normou, či standardem apod.). <p>Obsahové požadavky, na rozdíl od uživatelských, vznikají už ve fázi analýzy systému a odrážejí potřebu podporovat firemní procesy informačním systémem.</p> <p>Požadavky se dělí na:</p> <ul style="list-style-type: none"> • Funkční (požadavky na výkon a chování systému) • Ostatní (non-functional) <ul style="list-style-type: none"> ○ technologické (řeší se na úrovni použité technologie) např. určení operačního systému, dodavatele databázového systému, apod, ○ designové (řeší se v úrovni realizace systému) např. vzhled uživatelského rozhraní, rychlost odezvy, apod. Některé tyto požadavky se začínají řešit už ve fázi designu. Toto označení také odpovídá předdefinovaným typům v nástroji Power Designer, ○ ostatní <p>Výše uvedená klasifikace požadavků je základním nástrojem analýzy požadavků (viz definici) s cílem specifikace projektových záměrů pro budoucí vývoj systému.</p>
Proces	Process	<p>Obecně: časová struktura činností.</p> <p>Specificky:</p> <ol style="list-style-type: none"> a) postup realizace business záměru (business process), b) posloupnost činností v systému (technology process), c) postup vývoje software (software proces), apod.
Procesní model	Process model	Model zachycující procesy v systému, vazby mezi nimi a jejich strukturu..
Procesní architektura		<p><i>Design:</i> Struktura nezávislých procesů zachycující chování informačního systému.</p> <p><i>Analýza:</i> Struktura nezávislých procesů zachycující chování reálného systému.</p>
Přechod	Transition	Vazba mezi dvěma stavy v modelu stavů a přechodů.
Přírůstkový model vývoje IS	Incremental IS Development Model	Model vývoje IS, postavený na principiálním rozdílu mezi obsahovým celkem IS a projektem jeho vývoje. Rozhraním mezi obsahovou jednotkou systému (subsystémem) a jednotkou jeho realizace (projektem) je tzv. přírůstek.
Stav	State	Statická veličina fáze systému.

Pojem	Anglicky	Význam
Stavový diagram	State Chart	Diagram obsahující životní fáze objektů v systému.
Terminátor	Terminator	Abstrakce všech možných zdrojů a míst spotřeby dat systému. Třída typu (stereotypu) <<actor>> ve <u>funkčním modelu</u> .
Třída	Class	Model objektu. Každá třída obsahuje atributy a operace.
Událost	Event	Podnět nebo činnost, která ovlivňuje prvky systému.
UML	Unified Modelling Language	Jednotný modelovací jazyk – standard objektově-orientovaného modelování (OO), rozvíjený společností OMG (Object Management Group). Ve formě meta-modelu definuje základní pojmy a principy OO modelování, soustavu nástrojů (diagramů) jazyka a jejich vzájemné vztahy, plynoucí z definovaných principů.
Vodopádový model vývoje IS	Waterfall IS Development Model	Teoretický model vývoje IS, postavený na myšlence jednopřechodové realizace celého systému po jednotlivých fázích jeho životního cyklu. Tento model je prakticky nereálný a je používán jen pro vysvětlení smyslu „přírůstkového přístupu“ k vývoji IS.

Literatura

- [BPMN, 2005] Business Process Modeling Notation, Working Draft, Business Process Management Initiative, <http://www.bpmi.org>
- [Chen, 1976] Chen, P.P.: "The Entity Relationship Model - Toward A Unified View of Data", ACM Transactions on Database Systems 1, No.1, March 1976
- [Delobel, 1995] Delobel C., Lécluse Ch., Richard P.: Databases: From Relational to Object-Oriented Systems; International Thomson Publishing, London, 1995
- [Jackson, 1983] Jackson, M.A.: "System Development", Prentice-Hall Inc., 1983
- [Jayaratna, 1994] Jayaratna, N.: "Understanding and Evaluating Methodologies", McGraw Hill Europe, 1994
- [Martin, 1988] Martin, J.: "Information Engineering Methodology", Prentice-Hall, 1988
- [Řepa, 1999] Řepa, V. a kol: Analýza a návrh informačního systému, Ekopress, Praha 1999
- [Řepa, 2005] Řepa, V.: Podnikové procesy, Grada Publishing, Praha, 2005
- [Rumbaugh, 1992] Rumbaugh, J. et al.: "Object - Oriented Modeling and Design, Prentice-Hall, 1992.
- [UML, 2006] Unified Modeling Language™ (UML®) Specification, version 2.0 Object Management Group (OMG) (downloadable from <http://www.omg.org>)
- [Yourdon, 1989] Yourdon, E.: "Modern Structured Analysis", Prentice-Hall, Englewood Cliffs, N.J., 1989