evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vysoká škola báňská – Technická univerzita Ostrava

Hornicko-geologická fakulta

ANALÝZA A PROJEKTOVÁNÍ SYSTÉMŮ

(E-learningová podpora)

Roman Danel

Ostrava, 2014

Obsah

Seznam zkratek

BPR	Business Process Reengineering
CEO	Chief Executive Officer
CIO	Chief Information Officer
COBIT	Control OBjectives for Information and related Technology
COCOMO	Constructive Cost Model
CPM	Critical Path Method
CRAMM	CCTA Risk Analysis and Management Method
DD	Data Dictionary
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
EVM	Earned Value Management
IT/ICT	Information Technology / Information Communication Technology
ITIL	IT Infrastructure Library
ITSM	Information Technology Service Management
JAD	Joint Application Design
MBTI	Myers-Briggs Type Indicator
MCA	Multicriterial Analysis
OMG	Object Management Group
PERT	Program Evaluation and Review Techniques
RUP	Rational Unified Process
SSADM	Structured System Analysis and Design Method
STD	State Diagram
SWOT	Strength, Weaknesses, Opportunities, Threats
TDD	Test Driven Development
UML	Unified Modeling Language
WBS	Work Breakdown Structure

Seznam obrázků

1 Úvod

Studijní materiál „Analýza a projektování systémů“ je určen pro studenty navazujícího studia oborů „Systémové inženýrství v průmyslu“ a dalších oborů, jejichž obsah souvisí s informačními systémy, analýzou a návrhem systémů nebo softwarovým inženýrstvím. Předpokládá se základní znalost pojmů a technologií z informačních systémů, databází a systémové integrace.

V první části e-learningové opory se seznámíte s metodami analýzy, podrobněji se budeme věnovat analýze informačních systémů, analýze požadavků a analýze rizik. Další část je věnována strukturované analýze a návrhu systému se zaměřením na Yourdanův strukturální návrh, a objektově orientovanému návrhu, se zaměřením na metodiku UML.

V dalších třech kapitolách se opora zabývá vývojem a implementací informačních systémů. Nejdříve jsou shrnuty poznatky o řízení projektů v IT, včetně řízení týmu pro vývoj softwaru. Následuje přehled metodik pro vývoj softwaru, tradičních i agilních, které odpovídají úkolocentrickému a hodnotocentrickému přístupu. Samostatná kapitola je pak věnována problematice testování software se stručným přehledem používaných testů.

V poslední části e-learningové opory se dozvíte o metodikách řízení IT/ICT (ITIL a COBIT) a závěr je věnován softwarovému právu a softwarovým licencím.

Předkládaný studijní materiál je koncipován jako příprava k úspěšnému vykonání zkoušky z předmětu „Analýza a projektování systémů“ a také jako materiál k přípravě na státní závěrečnou zkoušku.

2 Cíle a metody analýzy

V této kapitole se seznámíte se základními principy a metodami analýzy. Při návrhu a tvorbě informačních systémů je analýza požadavků na systém klíčovým prvkem celého návrhu. Chyby na úrovni analýzy mohou mít fatální důsledek a mohou vést k celkovému neúspěchu projektu.

Stručný obsah kapitoly:

- Cíle a metody analýzy
- Empirické metody analýzy
- Exaktní metody analýzy
- SWOT
- Multikriteriální analýza (MCA)

Tato kapitola nevyžaduje žádné vstupní znalosti.

Získáte:

- Znalosti o metodách a cílech analýzy.
- Informace, k čemu lze využít analýzu SWOT a multikriteriální analýzu

Budete umět:

- Definovat cíl analýzy.
- Popsat vybrané metody analýzy
- Vysvětlit, co to je multikriteriální analýza a k čemu se používá
- Objasnit, k čemu se používá SWOT analýza

Specifikace času potřebného pro nastudování kapitoly: 30 minut.

2.1 Definice analýzy

Analýza je vědecká metoda založená na dekompozici celku na elementární části, je to metoda zkoumání složitějších skutečností rozkladem (dissolution) na jednodušší, základní celky.

Cílem analýzy je identifikovat podstatné a nutné vlastnosti elementárních částí celku, poznat jejich podstatu a zákonitosti. Analýza je také způsob výkladu, jestliže oddělujeme jednotlivé jevy a zkoumáme je izolovaně.

Opačný postup k analýze se nazývá **syntéza**. Výsledkem syntézy je vytvoření nových celků.

2.1.1 Cíl analýzy

Výsledkem analýzy u informačních systémů není pouze zdokumentování stávajícího stavu, ale zejména pochopení logiky fungování systému a tím pádem získání výchozích předpokladů pro možnou optimalizaci.

Cílem analýzy tedy je:

- Získat znalosti o systému
- Zjistit nedostatky a slabá místa
- Uvědomit si potřebné změny

Analýza by měla postupovat od globálního pohledu k potřebným detailům.

V oblasti programování se jedná o rozložení problému, který chce někdo vyřešit. Měla by obsahovat přesné zadání a být srozumitelná pro obě strany (analytika i programátora). Častá chyba analýz, kdy zadavatel nedomyslí, co může způsobit prázdná nebo nulová hodnota, přechod na nový rok, změna legislativy, ztráta hesla uživatele, atd.

2.2 Metody analýzy

2.2.1 Empirické metody

- Analogie (porovnání s obdobnými systémy)
- Pozorování
- Rozhovor s uživateli
- Dotazník
- Test
- Experiment
- Měření
- Studium dokumentací
- Workshop

Empirické metody analýzy jsou založeny na rozhovorech, pozorování a zkušenostech.

2.2.2 Obecné exaktní metody

- Klasifikační analýza – třídění jevů do skupin
- Funkční (vztahová) analýza – hledáme závislosti mezi prvky, zejména funkční závislosti, tento typ analýzy je zaměřen na strukturu
- Kauzální (příčinná) analýza
- Systémová analýza
- Srovnávací analýza
- Hodnotová analýza
- Rozhodovací analýza
- Organizační analýza
- Informační analýza

2.3 Druhy analýz

- Analýza požadavků
- Analýza trhu
- Analýza technologií
- Analýza vnitřních funkcí
- Analýza dat

2.4 Příklady nástrojů a metod pro analýzu

- SWOT
- PEST (Political, Economic, Social and Technological Analysis) neboli analýzu politických, ekonomických, sociálních a technologických faktorů – pro strategické řízení
- SMART – metodika stanovení cílů - specific (konkrétní), measurable (měřitelný), agreed (odsouhlasený), realistic (realistický) a timely (definovaný v čase)
- BPR – Business Process Reingeneering – analýza a optimalizace podnikových procesů
- Porterova analýza pěti sil (v ekonomice)
- Analýza požadavků (Requirements Engineering)
- Objektově orientovaná analýza – např. Booch
- UML (Unified Modeling Langugae)
- Strukturální analýza – Yourdon
- Principal Component Analysis (v ekonomice)
- Lexikální analýza (využití u překladačů)
- Syntaktická analýza
- Petriho sítě – modelování paralelních systémů
- WCET (Worst-case Execution Time) – pro návrh systémů „hard real-time“

Obrázek 1 Jeden z cílů analýzy - specifikace alternativ
[Zdroj: http://www.dilbert.com, © Adams]

2.4.1 SWOT analýza

Při SWOT analýze nějakého problému nebo situace nejdříve se soustředíme na pohled na analyzovaný systém ve čtyřech následujících oblastech:

S (Strengths) – Silné stránky

W (Weaknesses) – Slabé stránky

O (Opportunities) – Příležitosti

T (Threats) - Hrozby

Takto vytvořenou SWOT analýzu je poté třeba vyhodnotit. Jednou z možností je setřídění faktorů podle priority (podstatné, méně podstatné), k negativním stránkám analyzujeme příčinu a můžeme navrhnout protiopatření.

Výsledkem SWOT tedy může být například návrh další strategie. Cílem je najít možnosti růstu nebo identifikovat problémy. Autorem SWOT analýzy je A. Humphrey (Stanfordova univerzita).

Základ metody spočívá v klasifikaci a ohodnocení jednotlivých faktorů, které jsou rozděleny do čtyř výše uvedených základních skupin. Vzájemnou interakcí faktorů silných a slabých stránek na jedné straně vůči příležitostem a nebezpečím na straně druhé lze získat nové kvalitativní informace, které charakterizují a hodnotí úroveň jejich vzájemného střetu.

Tabulka 1 Možné strategie ze SWOT analýzy

SWOT-analýza		Interní analýza	
		S: Silné stránky	**W: Slabé stránky**
Externí analýza	**O: Příležitosti**	*S-O-Strategie*: Vývoj nových metod, které jsou vhodné pro rozvoj silných stránek společnosti (projektu).	*W-O-Strategie*: Odstranění slabin pro vznik nových příležitostí.
	T: Hrozby	*S-T-Strategie*: Použití silných stránek pro zamezení hrozeb.	*W-T-Strategie*: Vývoj strategií, díky nimž je možné omezit hrozby, ohrožující naše slabé stránky.

2.4.2 Multikriteriální analýza

Multikriteriální analýza (Multicriteria Decisional Analysis, MCA) se zabývá hodnocením možných alternativ podle několika kritérií. Patří mezi rozhodovací analýzy.

Hodnocení podle jednoho kritéria nemusí odpovídat hodnocení podle jiného, metody multikriteriální analýzy pak řeší konflikty mezi vzájemně protikladnými kritérii.

Cílem MCA je **utřídit a shrnout informace o variantních řešeních**.

Kritérium (faktor) je vlastnost, kterou posuzujeme u dané **alternativy**.

Postup:

- Identifikace alternativ (variantních řešení)
- Stanovení kritérií, určujících při rozhodování
- Podrobné hodnocení dopadu každé alternativy na každé kritérium
- Každému z kritérií se určí jeho relativní váha (významnost)
- Zhodnocení zpracovaných alternativ

Kritéria mohou být **maximalizační** (žádoucí je vyšší hodnota kritéria), **minimalizační** (žádoucí je nižší hodnota) nebo **smíšená**.

Metody stanovení váhových koeficientů:

- Metoda pořadí
- Fullerova metoda
- Bodovací metoda
- Saatyho metoda

Příklad výpočtu vah z Fullerova trojúhelníku

Přístupový a docházkový systém

Kritéria	**n**	**v_i'**
Cena PS – A	3	0,107142857
Cena DS – B	4	0,142857143
Návaznost na IS – C	6	0,214285714
Propojitelnost DS a PS – D	3	0,107142857
Servis – E	3	0,107142857
Záruku – F	2	0,071428571
Reference – G	6	0,214285714
Původ - H	1	0,035714286
∑	**28**	1

Výpočet vah - Fullerův trojuhelník

A	A	A	A	A	A	A
B	C	D	E	F	G	H
	B	B	B	B	B	B
	C	D	E	F	G	H
		C	C	C	C	C
		D	E	F	G	H
			D	D	D	D
			E	F	G	H
				E	E	E
				F	G	H
					F	F
					G	H
						G
						H

Tato metoda spočívá ve vzájemném porovnání všech kriterií (A až H) za použití trojúhelníkového schématu, ve kterém se jednotlivé dvojice kritérií vyskytují pouze jednou.

Dostaneme: A – 3; B – 4; C – 6; D – 3; E – 3; F – 2; G – 6; H – 1

MCA

Kritéria	Cena DS		Cena PS		Návaznost na IS		Slučitelnost DS a PS	
firma		0,107		0,143		0,214		0,107
KOMPET	8	0,856	9	1,287	4,5	0,963	8	0,856
GACC	5	0,535	4	0,572	4,5	0,963	8	0,856
Z-WARE	6	0,642	7,5	1,073	4,5	0,963	8	0,856
TETRONIC	7	0,749	7,5	1,073	4,5	0,963	8	0,856
IVAR	1	0,107	2	0,286	4,5	0,963	8	0,856
DUHASYS	3	0,321	6	0,858	4,5	0,963	8	0,856
ID KARTA	2	0,214	3	0,429	4,5	0,963	8	0,856
GRANUS	4	0,428	1	0,143	0	0	8	0,856
EFG CZ	9	0,963	5	0,715	4,5	0,963	8	0,856

Příklad: Vyhodnocení firem podle kritérií [Zdroj: A. Makarová]

Kritéria	Servis		Záruka	Reference			Původ		Celkem	Pořadí
firma		0,107		0,071		0,214		0,036		
KOMPET	4,5	0,482	3	0,213	2	0,428	5,5	0,198	5,2825	4
GACC	3,5	0,375	3	0,213	3	0,642	5,5	0,198	4,3535	6
Z-WARE	5,5	0,589	3	0,213	4	0,856	5,5	0,198	5,389	3
TETRONIC	5,5	0,589	7	0,497	7	1,498	5,5	0,198	6,422	2
IVAR	3	0,321	3	0,213	8	1,712	1	0,036	4,494	5
DUHASYS	3	0,321	3	0,213	1	0,214	1	0,036	3,782	8
ID KARTA	3,5	0,375	3	0,213	5	1,07	5,5	0,198	4,3175	7
GRANUS	3	0,321	3	0,213	6	1,284	1	0,036	3,281	9
EFG CZ	6,5	0,696	8	0,568	9	1,926	5	[redacted]	**45**	1

2.4.3 Heuristická analýza

Používá se v situaci, kdy algoritmus nebo přesné řešení nejsou známy, využívá zkušeností, intuici. Příklad použití v praxi je heuristická analýza počítačových virů v antivirových programech (snaha odhalit pravděpodobné virové řetězce).

Shrnutí hlavních bodů kapitoly:

- Cílem analýzy je pochopení logiky fungování systému, zjistit slabá místa, podchytit požadavky na změny a připravit podklady pro návrh systému
- Metody analýzy dělíme na empirické a exaktní
- SWOT analýza se používá pro odhalení slabých míst, návrh strategií, posílení silných stránek
- Multikriteriální analýza je příkladem rozhodovací analýzy, kdy máme za úkol vybrat variantní řešení

Kontrolní otázky:

1) Co je cílem analýzy?
2) Jaké jsou empirické a exaktní metody analýzy?
3) K čemu je SWOT analýza?
4) K čemu se používá multikriteriální analýza?

Použitá literatura:

1) HOFFER, J., GEORGE, J., VALACICH, J. Modern systems analysis and design. 3rd ed. Překlad David Krásenský. Upper Saddle River, N.J.: Prentice Hall, c2002, xxxii, 733 p. ISBN 01-303-3990-3.
2) CHLAPEK, D., ŘEPA, V., STANOVSKÁ, I. Vývoj informačních systémů (pracovní sešit ke cvičením). Praha: VŠE, 2005. ISBN: 80-245-0977-6.
3) KŘUPKA, J., KAŠPAROVÁ, M., MÁCHOVÁ, R. Rozhodovací procesy. Metody stanovení vah kritérií. 2011 [online]. Dostupné na www: < http://www.rozhodovaciprocesy.cz/vicekriterialni-rozhodovani/2-1-metody-stanoveni-vah-kriterii.html>

3 Analýza a návrh informačních systémů

V této kapitole se seznámíte s analýzou prováděnou za účelem návrhu a implementace informačních systémů. Prvotní analýzou je studie proveditelnosti, která ověřuje, zda požadavek na informační systém je realizovatelný. Postupujeme od hrubé analýzy, jejímž výstupem je návrh globální architektury a koncepce systému. Následuje analýza požadavků, které se detailněji budeme věnovat v následující kapitole. Poslední fází analýzy je analýza detailní, která rozpracovává dílčí architektury systému a může vyústit ve fyzický model systému nebo podrobnou funkční specifikaci. V závěru kapitoly si vysvětlíme, co to je prototyp, k čemu se používá a jaké druhy prototypu známe.

Stručný obsah kapitoly:

- Hrubá analýza
- Analýza požadavků
- Detailní analýza
- Návrh řešení systému
- Mock-up
- Wireframe modely
- Prototypy

Tato kapitola předpokládá základní znalosti pojmů z informačních systémů.

Získáte:

- Znalosti o postupu analýzy při návrhu informačního systému.
- Přehled o postupu od hrubé analýzy, přes analýzu požadavků až k detailní analýze
- Představu, co je to mock-up, drátěný model aplikace, k čemu je prototyp a jaké typy prototypu můžeme vytvářet

Budete umět:

- Definovat cíl hrubé analýzy
- Vysvětlit, co je cílem analýzy požadavků
- Popsat, co je obsahem a výstupem detailní analýzy
- Objasnit, k čemu se používá prototypování

Budete schopni:

- Při zadání na návrh informačního systému nebo aplikačního vybavení budete schopni postupovat metodicky od koncepčního pojetí po popis detailů

Specifikace času potřebného pro nastudování kapitoly: 40 minut.

3.1 Studie proveditelnosti

Cílem studie proveditelnosti je posoudit, zda za prostředky, které jsou k dispozici a v daném čase, je vůbec možné požadovanou funkcionalitu zajistit. Tedy, zda se projekt vůbec vyplatí a zda je technicky realizovatelný. Studii proveditelnosti může vypracovat dodavatel (aby posoudil, zda projekt bude v zisku) nebo pro interní potřeby i zadavatel (ověření ROI – návratnosti investic).

3.2 Hrubá (globální) analýza

Cílem globální analýzy je:

- Zpracovat hrubý návrh řešení
- Posoudit proveditelnost řešení
- Odhadnout ekonomickou efektivnost řešení
- Určit možná rizika

V této fázi se plánuje, jak bude systém vypadat, zda a jak bude vytvořen prototyp (pilotní řešení)

Jedním z výstupů globální analýzy mohou být modely systému. Ty lze vytvořit buď strukturovaným přístupem (ERD, DFD…) nebo objektově orientovaným přístupem (UML,…).

Cílem je postihnout fungování systému, jeho strukturu, prvky a vazby uvnitř systému a vztah systému k okolí.

3.3 Analýza požadavků

Požadavek musí být:

- proveditelný
- měřitelný
- testovatelný

3.3.1 Sběr požadavků

Ve fázi **analýzy požadavků** se snažíme o identifikaci:

- nejasných požadavků,
- nekompletních
- protikladných

Požadavky by měly mít určenu svou prioritu a měly by mít přiřazeny akceptační kritéria.

3.3.2 Záznam požadavků

- Text
- USE-CASE diagramy (případy použití, UML)
- Specifikace procesů

Nedostatečně zpracované požadavky na systém nebo nedostatečná zkušenost analytika může vést k závažným koncepčním chybám při návrhu systému.

Stakeholder (zúčastněná strana) – vlastní organizace, kde se analýza provádí, ale také vrcholový management, spolupracující organizace, atd.

3.4 Detailní analýza

Cílem detailní analýzy je rozpracování modelu systému do dostatečných detailů tak, aby se stal podkladem pro implementaci systému.

Model systému pak musí obsahovat:

- Popis požadovaných funkcí
- Popis datových struktur
- Uživatelské rozhraní
- Návrh softwarové struktury systému (hierarchie modulů, knihovny…)
- Hardwarová a síťová architektura
- Způsob testování systému
- Akceptační testy
- Návrh zavádění IS do provozu
- Řešení bezpečnosti a zálohování

Funkční pohled na systém lze modelovat pomocí DFD diagramů v případě aplikace strukturovaného přístupu nebo Use Case diagramy v případě využití objektově orientovaného přístupu.

Datové struktury lze popsat pomocí ERD nebo v UML pomocí Diagramu tříd. Součástí návrhu by měla být normalizace, optimalizace z hlediska odezvy (návrh indexů) a řešení integrity a její kontroly.

Součástí návrhu by měla být také kvantifikace systému – přibližný odhad velikosti datových toků, uživatelských přístupů apod. Tento odhad je podkladem pro návrh hardwarové koncepce systému.

Výsledkem detailní analýzy často bývá tzv. **funkční specifikace**, která se pak někdy stává přílohou smlouvy o dílo na vyhotovení informačního systému. Je dobrou praktikou, aby obě strany – dodavatel i odběratel – funkční specifikaci vzájemnou dohodou potvrdili. Odběratel tak získá potvrzení rozsahu díla, které pak může požadovat u předávacího řízení. Dodavatel na druhé straně potvrzením funkční specifikace ze strany odběratele získává garanci konečného objemu prací na dodávce (pokud se objeví další požadavky po potvrzení funkční specifikace, je to většinou dodavatel vnímáno jako „vícepráce“, které nejsou součástí smlouvy o dílo a budou realizovány za dodatečné náklady).

3.5 Návrh řešení

Návrh řešení by měl obsahovat:

- Popis cíle systému
- Identifikaci uživatelů
- Vymezení hranic systému
- Závěry z analýzy požadavků na systém
- Návrh hlavních funkčních celků (subsystémů)
- Události, na které systém musí reagovat
- Odhad a návrh datové základny
- Technické řešení
- Řešení prototypu

Plán vývojových prací:

- Kdo bude projekt řídit
- Složení vývojového týmu
- Seznam úkolů a jejich priority
- Harmonogram
- Nároky na zdroje a finance

3.6 Prototypování

Prototypy slouží k řešení problémů analýzy požadavků.

Důležité: Prototyp je model aplikace, část aplikace nebo její vizualizace za účelem předvedení nebo otestování. Je to mezikrok mezi specifikací a fungujícím systémem**.**

.

Prototypem mohou být **reálné modely** aplikací, ukázky jak bude aplikace ve finále vypadat. Uživatelům pomohou získat představu, jak bude systém vypadat a jak se s ním bude pracovat. Měl by být uživateli předložen čím jak nejdříve, aby uživatel viděl, jak systém vypadá, jak reaguje a jaké je uživatelské rozhraní a mohl se k systému vyjádřit.

Nevýhody:

- Přílišná pozornost na uživatelské rozhraní místo na systém
- Nucení k používání reálného kódu v prototypu, pokud není dostatek času
- Manažeři mají pocit, že produkt už je hotový

Prototyp tedy slouží ke komunikaci se zákazníkem (uživatelem), cílem je pochopení, jak systém funguje a jak jsou realizovány požadavky:

- Uživatel vidí, jak systém pracuje
- Odhalení chyb nebo chybějících částí specifikace

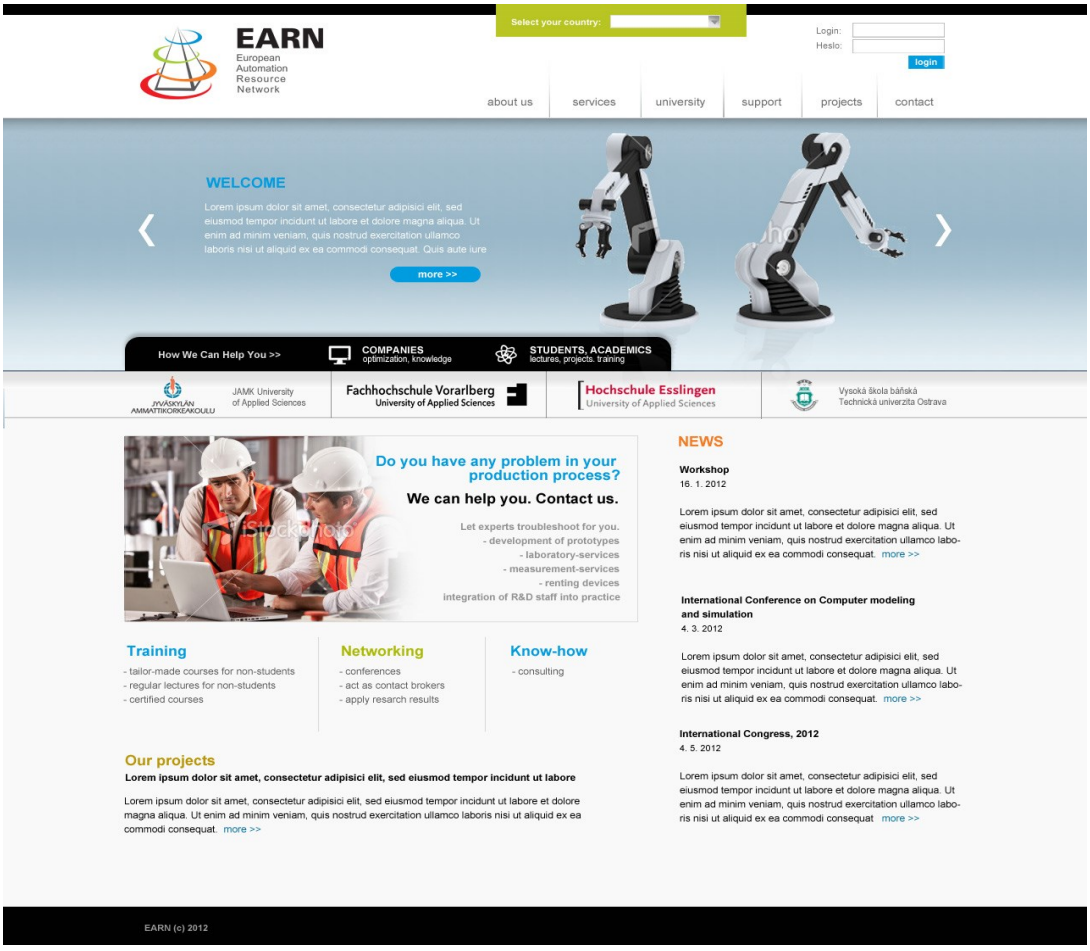
3.6.1 Přístupy k prototypování

- **Evolutionary prototyping** – cílem je dodat fungující aplikaci, prototyp hned na začátku a pak je předefinováván až k finálnímu produktu

- **Throw-away prototyping** - prototyp se týká požadavků, které nejsou dostatečně jasně definovány, slouží k pochopení a definici požadavků; vývoj systému s prototypem nesouvisí

Evoluční prototyping je vhodný tam, kde ve specifikaci nelze vše předem definovat. Má blízko k „agilnímu“ vývoji. Silnou stránkou je řešení chybějících, zmatených či špatně definovaných požadavků. Uživatelé tím, že od počátku pracují s aplikací, jsou současně proškoleni na užívání. Nevýhodou je složitější řízení vývoje.

Throw-away prototyping slouží k předvedení a experimentům, není vytvářen s tím, že budu fungovat jako finální systém. Výhodou je rychlá tvorba.



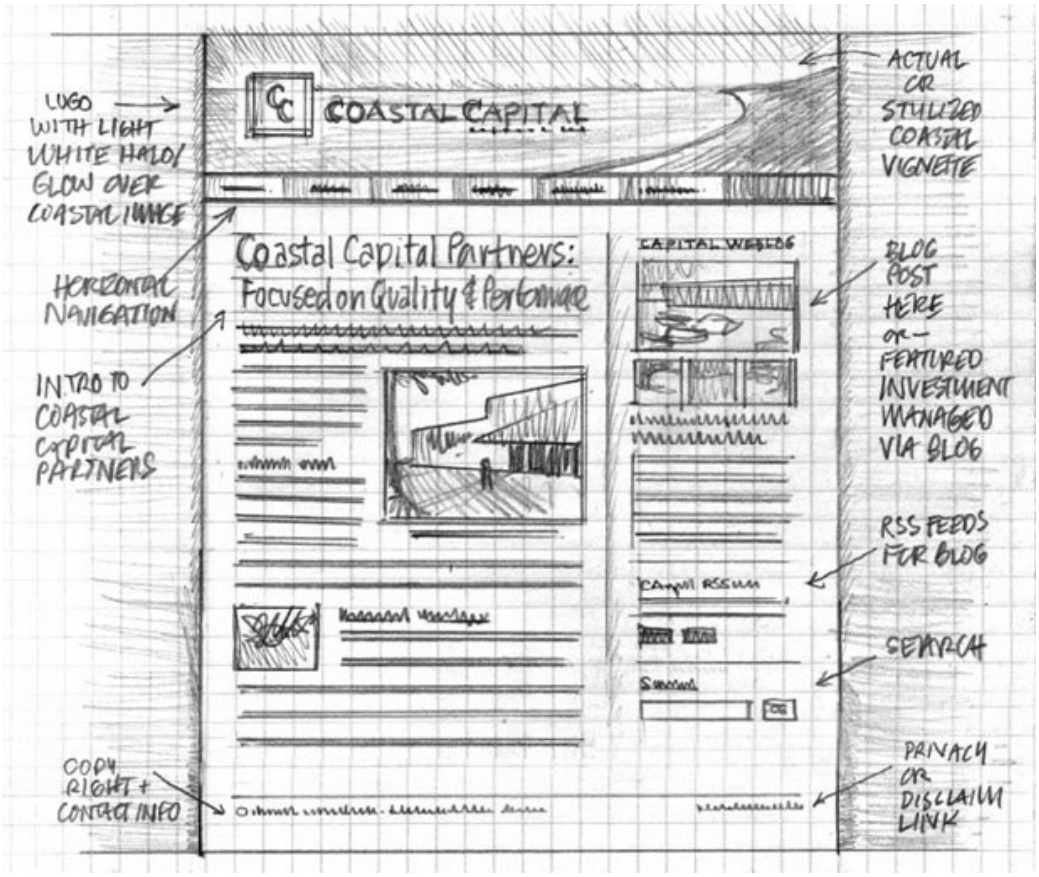
Obrázek 2 Ukázka mock-up webové aplikace
[Zdroj: projekt EARN, © Řepka – Danel]

3.6.2 Mock-up

Mock-up je nakreslený vzhled obrazovek (papír, elektronicky) s omezenou nebo nulovou funkcionalitou. Na obrázku 2 vidíme ukázku mock-up pro webovou aplikaci, nakreslený v CorelDraw.

3.6.3 Wireframe

Wireframe neboli **drátěný model** aplikace ukazuje její strukturu a vizuální řešení – rozkreslené obrazovky a umístění jednotlivých elementů. Časté je užití drátěných modelů u webových aplikací. Wireframe může mít různé podoby, od skic na papíře až po použití speciálních softwarových nástrojů pro vytváření drátěných modelů (např. Visual Paradigm).



Obrázek 3 Ukázka wireframe webové aplikace
[Zdroj: http://www.webdesignerdepot.com/2009/07/using-wireframes-to-streamline-your-development-process/]

3.6.4 Typy wireframů

- **Textový wireframe**. Jedná se o např. odrážkami vytvořený seznam položek a funkcí, které by měl web obsahovat. Je to nejjednodušší možný wireframe.
- **Blokový wireframe**. Tento již využívá bloky (čáry, obdélníky, čtverce), aby definoval rozmístění a velikost jednotlivých prvků. Ty pak mohou obsahovat text, který popisuje jejich obsah nebo funkci.

- **Podrobný wireframe**. Podrobný wireframe dopodrobna popisuje každý v něm použitý element, je vytvářen v přesných proporcích, je popsána funkčnost, obsah, prostě vše.
- **Proklikávací wireframe**. Tento může být nástavbou blokového nebo podrobného wireframu a jeho obrovskou výhodou je, že jednotlivé stránky (např. návrh titulky, přehledu a detailu zboží, košíku a objednávky) jsou propojeny, takže lze mezi nimi klikat jako ve skutečném webu. S jeho pomocí lze např. již předběžně provést malé uživatelské testování.

3.6.5 Prototype fidelity

Termínem fidelity označujeme přesnost prototypu.

Low-fidelity prototyp – nekompletní, ukazuje jen část, hrubá skica, obrázek
High-fidelity prototyp – prototyp je blízko finálnímu produktu.

3.6.6 Horizontální a vertikální prototypy

- **Horizontální prototypy** – zaměření na funkcionalitu a souvislosti na úkor implementace
- **Vertikální prototypy** – jen část systému s důrazem, aby prototyp směroval k finálnímu produktu

Shrnutí hlavních bodů kapitoly:

- Studie proveditelnosti – posuzuje, zda je požadovaný systém realizovatelný
- Hrubá analýza – definuje koncepci a globální architekturu systému
- Analýza požadavků – identifikuje uživatelské požadavky na systém a jejich realizovatelnost
- Detailní analýza definuje dílčí architektury systému a ústí v logické nebo fyzický model systému
- Funkční specifikace – možný výstup detailní analýzy, detailní popis funkcí systému, může obsahovat až popisy uživatelského rozhraní a obrazovek aplikace
- Prototyp – mezistupeň mezi návrhem a fungující aplikací – od drátěných modelů ve firmě skic jak bude aplikace vizuálně vypadat, až po částečně fungující aplikaci – cílem je ukázat uživateli jak bude systém vypadat

Kontrolní otázky:

1) Co je cílem studie proveditelnosti?
2) Jaký je cíl a výstup hrubé analýzy?
3) Jak na hrubou analýzu navazuje detailní analýza?
4) Co je to funkční specifikace?
5) Co je to prototyp?
6) Jaké jsou základní dva směry prototypování?
7) Co je to mock-up?
8) Co je to drátěný model (wireframe)?

Použitá literatura:

1) HOFFER, J., GEORGE, J., VALACICH, J. Modern systems analysis and design. 3rd ed. Překlad David Krásenský. Upper Saddle River, N.J.: Prentice Hall, c2002, xxxii, 733 p. ISBN 01-303-3990-3.
2) CHLAPEK, D., ŘEPA, V., STANOVSKÁ, I. Vývoj informačních systémů (pracovní sešit ke cvičením). Praha: VŠE, 2005. ISBN: 80-245-0977-6.
3) ŘEPA, V. Analýza a návrh informačních systémů. 1.vyd. Praha: Ekopress, 1999, 403 s. ISBN 80-861-1913-0.
4) GUCKENHEIMER, S., PEREZ, J.. Efektivní softwarové projekty: metodiky efektivního vývoje softwaru. Vyd. 1. Brno: Zoner Press, 2007, xi, 255 s. ISBN 978-80-86815-62-6.

4 Analýza požadavků

V této kapitole se seznámíte s analýzou požadavků. Popíšeme si typy požadavků, jaké požadavky řadíme mezi funkční a non-funkční. Vysvětlíme si základní metody analýzy požadavků, jejich výhody a nevýhody.

Stručný obsah kapitoly:

- Cíl analýzy požadavků
- Typy požadavků
- Metody analýzy požadavků
- Metoda analýzy JAD

Při studiu této kapitoly je doporučeno prostudování předchozích dvou kapitol tohoto učebního textu. Předpokládá se rovněž základní orientace v pojmech z informačních systémů.

Získáte:

- Znalosti o cílech a formě analýzy požadavků
- Přehled o metodách analýzy rizik, jejich výhodách a nevýhodách
- Přehled o typech požadavků

Budete umět:

- Definovat cíl analýzy požadavků
- Vysvětlit rozdíl mezi funkčním a nefunkčním požadavkem
- Popsat výhody a nevýhody základních metod analýzy požadavků

Budete schopni:

- Na základě zadání použít základní metody analýzy požadavků za účelem jejich podchycení a vyhodnocení

Specifikace času potřebného pro nastudování kapitoly: 60 minut.

4.1 Cíl analýzy požadavků

Cílem analýzy požadavků (Requirement Analysis) je:

a) Vymezení funkčnosti systému
b) Odhad množství práce
c) Vyjasnění zadání
d) Zachycení omezení

4.2 Problémy při analýze požadavků

e) Uživatelé nemají představu, co jim aplikace/systém může poskytnout
f) Nepřesný popis postupů a cílů
g) Uživatelé nemají nadhled (popisují pouze svou část bez znalosti celku)
h) Uživatelé se zaměřují na detaily a popisy, jak danou činnost vykonávají nyní (nemusí to být optimální postup)
i) Požadavky funkcí, které nejsou reálné
j) Požadavky funkcí, které jsou zbytečné
k) Zatížení provozní rutinou – nesdělí některé zásadní informace
l) Rozdílný pohled vedení firmy a zaměstnanců (snaha, aby jim inovace systémů nepřinášela další práci)

4.3 Uživatelské požadavky

Požadavky můžeme rozdělit do tří kategorií:

- **NORMAL** – minimální nutná funkcionalita, „samozřejmé“ části
- **EXPECTED** – důležité požadavky (do této kategorie patří např. „snadnost používání“)
- **EXCITING** – v dané chvíli „něco navíc“ (v budoucnu může spadnout do kategorie normal…)

Use Case – scénář, jak bude produkt používání v určitých situacích.

4.4 Výsledek analýzy požadavků

a) Jaké informace lidé potřebují pro svou práci
b) Business cíle
c) Data – definice a rozsah
d) Kde, jak, kdy a kým jsou data zpracovávány a využívány
e) Jaké jsou vzájemné souvislosti
f) Pravidla řídící zpracování informací
g) Události ovlivňující data (a kdy a jak tyto události vznikají)

Validace požadavků:
Odpovídá produkt reálným požadavkům?

Verifikace požadavků:
Odpovídá produkt výchozí specifikaci?

4.5 Metody analýzy požadavků

- Interview s uživatelem
- Dotazník (Questionnaire)
- Pozorování (analytik je přítomen na pracovišti)
- Studium dokumentace a písemných zdrojů
- Studium stávajícího informačního systému
- JAD techniky (Joint Application Design)

Každá metoda má své výhody a nevýhody. Pro důkladně provedenou analýzu požadavků je proto nutné zkombinovat více metod.

4.5.1 Interview

- Nutná příprava otázek předem – měl by být promyšlen scénář
- Hlavní metody výhoda je vhled na základě neverbálních informací – lze posoudit, co je pro uživatele důležité, otázky přizpůsobit odpovědím, zisk dalších informací z toho, jak uživatel reaguje, jak se tváří, pochopení, co je pro něho důležité
- Je důležité umět dělat poznámky nebo si hovor nahrávat (v tom případě je vhodné požádat o svolení k nahrávání)
- Na konci shrnout získané informace (potvrzení, zda jsme si něco nepoznamenali chybně) a poděkování za poskytnutí rozhovoru
- Získané informace je vhodné později projít (doporučuje se do 48 hodin po rozhovoru, později už tazatel ztrácí výhodu neverbálních informací) a pokud je nalezena nějaká nejasnost nebo rozpor, kontaktovat uživatele za účelem upřesnění
- Nevýhoda – nutnost domlouvat schůzku, časová náročnost
- Pokud to je možné, ještě přínosnější může být skupinové interview. Tato forma je však náročnější na vedení.

4.5.2 Dotazník

- Výhodou je podstatně nižší časová náročnost než rozhovor, není nutné domlouvat schůzku a
- Další výhodou je možnost oslovit velké množství respondentů a odpovědi statisticky vyhodnotit
- Nevýhodou je právě absence neverbálních informací; respondenti odpovídají na všechny otázky bez rozlišení, co je důležité a co ne; podstatné věci mohou uniknout (chybí otázka)
- Druhy otázek:
    - OPEN-END – respondent odpovídá svými slovy

    - CLOSE-END – odpověď je omezena (výběr z možností, škála, hodnocení 1 až 5,…)
- Close-end otázky se lépe statisticky vyhodnocují. V dotaznících obvykle převažují, je však vhodné respondentům ponechat možnost zvolit (a upřesnit) variantu, která není uvedena v předdefinovaných odpovědích. Pokud tato možnost není obsažena, může nám uniknout něco podstatného
- Zásadní je volba respondentů (ideální je oslovit různé skupiny uživatelů)
    - Convenient group
    - Random group
    - Purposeful group
    - Stratified sample
- Důležité je také znění otázek – otázka by neměla navádět k odpovědi
- Je třeba se vyhnout nejasnostem (někdy, občas, …)

4.5.3 Pozorování

Spočívá v přítomnosti analytika na pracovišti, analytik sleduje a vyhodnocuje činnost uživatelů při práci. Výhodou je zisk informací, které uživatelé nesdělí při rozhovoru/dotazníku (např. z důvodu, že si danou věc z důvodu provozní rutiny neuvědomí). Může být zásadní k porozumění, co je důležité. Eliminace subjektivních informací získaných jinými metodami („často dělám…“)

Při této metodě si analytik musí dát pozor na dvě věci:

a) Lidé, když vědí, že jsou sledováni, mohou být nervózní a dělat chyby
b) Lidé, když vědí, že jsou sledováni, začnou pracovat precizně a podle předpisů, ač to za běžných okolností nedělají

Do této kategorie patří i účast analytika na firemních poradách (pokud je to možné).

4.5.4 Studium dokumentů

Studium psané dokumentace může být užitečným zdrojem informací. Zde se můžeme setkat s problémem, že dokumentace není udržována a průběžně aktualizována a tedy již nemusí odpovídat reálnému stavu (Formal and Informal system).

Cenným zdrojem informací také může být studium stávajících aplikací, formulářů a reportů.

4.5.5 JAD (Joint Application Design)

Metoda analýza JAD je založena na následujících principech:

- Je definován požadavek na produkt
- Organizace společné schůzky analytiků, různých skupin uživatelů a vedení
- Dodání podkladů účastníkům před schůzkou
- Každý účastník prezentuje svoje požadavky na systém
- Požadavky se sepíší
- Diskuze k požadavkům
- Odsouhlasení požadavků, u kterých je konsenzus

Tato metoda je velmi efektivní. Různé skupiny uživatelů získají vhled díky seznámení se s požadavky jiných skupin. Mohou objevit informace a požadavky, které je nenapadnou a mohou být pro ně užitečné nebo naopak mohou ovlivnit či zamezit požadavky jiných skupin.

Nevýhodou je náročnost na organizaci (najít společný čas pro zúčastněné strany) a náročnost na vedení (zejména je nutné zabránit „žvanění“ a chaotické diskuzi). Je vhodné, aby workshop byl veden minimálně dvěma lidmi, z nichž jeden řídí diskuzi a druhý si vede poznámky.

4.6 Funkční a nefunkční požadavky

Požadavky na systém se dělí do kategorií:

a) Funkční – postihují chování a vlastní funkcionalitu systému
b) Nefunkční – definují vlastnosti systému jako celku a omezení
c) Odvozené (Derrived)
d) Výkonnostní a designové

Termín „nefunkční“ není příliš šťastně zvolen, nicméně v česko-jazyčné literatuře se běžně používá, pochází z anglického non-functional.

Do kategorie nefunkčních požadavků patří například požadavky na kvalitu, udržovatelnost, použitelnost, efektivitu, spolehlivost, přenositelnost, dodržování standardů, způsob dodání, součinnost s ostatním software, zálohování, řešení bezpečnosti, legislativní požadavky, reakční doba či etické požadavky.

4.7 Příklady metodik pro vyhodnocení kvality software

4.7.1 FURPS (Hewlett-Packard)

Dle této metodiky máme dva největší problémy:

a) Průběžné změny požadavků po zahájení vývoje
b) Nové požadavky po zafixování ceny a harmonogramu

4.7.2 ISO 9126

Norma pro vyhodnocování kvality softwaru, posuzuje šest kategorií:

1. Functionality
2. Reliability (spolehlivost)
3. Usability (použitelnost)
4. Efficiency (Výkonnost, efektivnost)
5. Maintainbility (Udržovatelnost)
6. Portability

Připravte písemně scénář pro rozhovor s uživatelem s cílem provést analýzu požadavků na vytvoření aplikace pro zpracování a prezentaci informací z řídicího systému technologického procesu.

Výchozí podklady pro interview

Máme technologický proces, který se skládá z několika zařízení, zásobníků a systému pásových dopravníků. Proces je vybaven binárními snímači, které zaznamenávají spuštění a zastavení jednotlivých strojů a dopravníků. Informace jsou zapisovány do databáze.

Zadání pro analytika

„Zpracujte analýzu požadavků na vytvoření aplikace, která bude zpracovávat a prezentovat informace o využití zařízení a dopravníků (chody – prostoje). Formou interview s uživatelem zjistěte, jaké informace potřebuje, jak je bude využívat, jaká má být forma prezentací informací, v jakých časových jednotkách mají být informace o chodu/prostojích sumarizovány, zda je potřeba tiskových výstupů nebo exportu dat do jiných systémů. Uživatelem aplikace bude pracovník na pozici ‚technolog – vedoucí výroby', nepředpokládají se hlubší IT znalosti. Uživatel, se kterým bude vedeno interview, není zadavatel ani investor informačního systému."

Doporučená struktura interview

1. Úvod, představení sebe
2. Stručné představení projektu a jeho cílů
3. Otázky (s důrazem na pořadí)
4. Shrnutí hlavních bodů
5. Prostor pro otázky a nápady uživatele
6. Poděkování a ukončení rozhovoru

Shrnutí hlavních bodů kapitoly:

- Analýza požadavků (Requirement Analysis) slouží k podchycení, zpracování a ověření požadavků uživatelů na systém; jedná se o kritickou fázi návrhu systému – chyba na úrovni analýzy požadavků může mít velmi negativní vliv na výsledný systém
- Požadavky – funkční a nefunkční
- Metody analýzy požadavků: interview, dotazník, pozorování, studium dokumentace
- Metoda JAD
- Metodiky pro vyhodnocení kvality (ISO 9126)

Kontrolní otázky:

1) Co je cílem analýzy požadavků?
2) Jaké znáte metody analýzy požadavků? Shrňte jejich výhody a nevýhody.
3) Jaký je rozdíl mezi open-end a close-end otázkami?
4) Jaká je nevýhoda metody „pozorování“?
5) Jaký je rozdíl mezi funkčním a nefunkčním požadavkem?
6) Čím se zabývá ISO 9126?

Použitá literatura:

1) HOFFER, J., GEORGE, J., VALACICH, J. Modern systems analysis and design. 3rd ed. Překlad David Krásenský. Upper Saddle River, N.J.: Prentice Hall, c2002, xxxii, 733 p. ISBN 01-303-3990-3
2) CHLAPEK, D., ŘEPA, V., STANOVSKÁ, I. Vývoj informačních systémů (pracovní sešit ke cvičením). Praha: VŠE, 2005. ISBN: 80-245-0977-6.
3) SOMMERVILLE, I.: Softwarové inženýrství. Brno: ComputerPress, 2013. ISBN: 978-80-251-3826-7

5 Analýza rizik

V této kapitole se seznámíte s analýzou rizik, která souvisí se zajištěním bezpečnosti IT/ICT. Komponenty IT/ICT a informační systémy nejsou statické, ale procházejí změnami a tak i řešení bezpečnosti a bezpečnostní politiky je kontinuální a nikdy nekončící proces. Popíšeme si, co je předmětem analýzy rizik, jaké jsou hlavní druhy hrozeb. Seznámíte se s některými metodikami používanými pro analýzu rizik. Vysvětlíme si, co je to bezpečnostní politika a co by mělo být obsaženo v havarijním plánu.

Stručný obsah kapitoly:

- Základní pojmy z analýzy rizik
- Kategorizace hrozeb
- Metodiky analýzy rizik
- Bezpečnostní politika
- Havarijní plán

Pro studium kapitoly se předpokládá znalost základních pojmů z počítačové bezpečnosti (např. termíny virus, trojský kůň, hoax, phishing apod.) a základní orientace v bezpečnostních mechanismech a opatřeních (co je to antivirus, firewall apod.).

Získáte:

- Znalosti o postupu analýzy rizik.
- Přehled o kategoriích hrozeb pro informační systém a IT/ICT
- Informace o metodikách pro analýzu rizik

Budete umět:

- Definovat cíl analýzy rizik
- Vysvětlit, co je to aktivum
- Kategorizovat hrozby v IT/ICT
- Popsat, co je obsahem bezpečnostní politiky
- Objasnit, co by mělo být obsaženo v havarijním plánu

Budete schopni:

- Základní orientace v problematice provádění analýzy rizik a zajištění bezpečnosti informačních systémů
- Sestavit havarijní plán pro koncového uživatele informačního systému
- Vysvětlit, jaké kroky je třeba udělat pro vytvoření bezpečnostní politiky firmy

Specifikace času potřebného pro nastudování kapitoly: 40 minut.

5.1 Základní pojmy z analýzy rizik

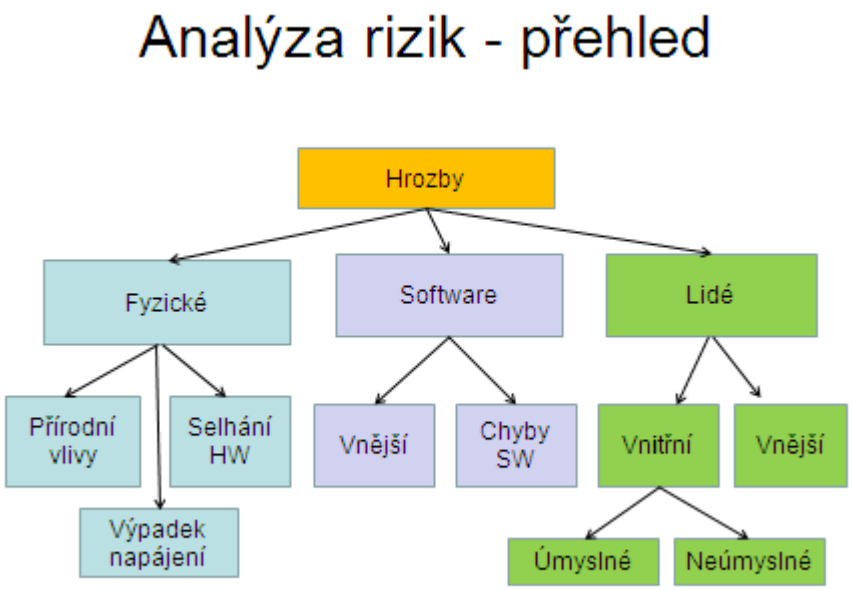
- **aktivum** (Asset) – vše co má pro společnost nějakou hodnotu a mělo by být odpovídajícím způsobem chráněno,
- **hrozba** (Threat) – jakákoliv událost, která může způsobit narušení důvěrnosti, integrity a dostupnosti aktiva
- **zranitelnost** (Vulnerability) – vlastnost aktiva nebo slabina na úrovni fyzické, logické nebo administrativní bezpečnosti, která může být zneužita hrozbou.
- **riziko** – pravděpodobnost, že hrozba zneužije zranitelnost a způsobí narušení důvěrnosti, integrity nebo dostupnosti.
- **opatření** (Countermeasure) – opatření na úrovni fyzické logické nebo administrativní bezpečnosti, které snižuje zranitelnost a chrání aktivum před danou hrozbou
- **ohrožení** (Exposure) – skutečnost, že existuje zranitelnost, která může být zneužita hrozbou
- **narušení** (Breach) – situace, kdy došlo k narušení důvěrnosti, integrity nebo dostupnosti v důsledku překonání bezpečnostních opatření.

Aktivem je cokoliv, co má pro organizaci **hodnotu**:

- fyzická aktiva (např. počítačový hardware, komunikační prostředky, budovy
- informace (dokumenty, databáze,...)
- software
- schopnost vytvářet určité produkty nebo poskytovat služby – know-how
- pracovní sílu, školení pracovníků, znalosti zaměstnanců, zapracování apod.
- nehmotné hodnoty (např. abstraktní hodnota firmy, image, dobré vztahy atd..)

5.2 Postup při analýze rizik

1. Stanovení hranice
2. Identifikace aktiv
3. Stanovení hodnoty aktiv (+seskupení)
4. Identifikace hrozeb
5. Analýza hrozeb
6. Stanovení pravděpodobnosti jevu
7. Měření rizika



Obrázek 4 Analýza rizik – přehled hrozeb pro IT/ICT

Analýza rizik by měla odpovědět na tyto otázky:

- Co se stane, když informace nebudou chráněny?
- Jak může být porušena bezpečnost informací?
- S jakou pravděpodobností se to stane?

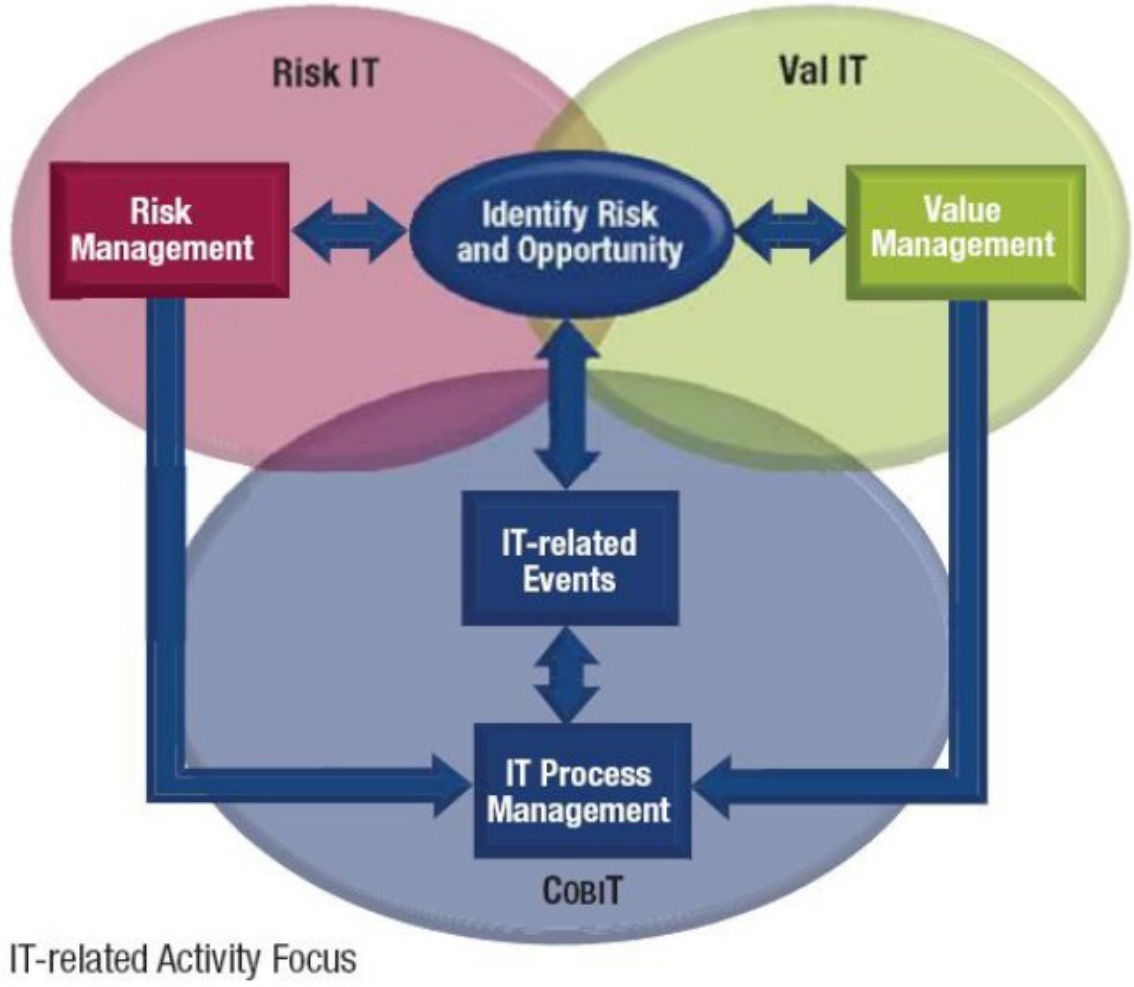
Co je cílem?

- Zajištění důvěrnosti dat
- Zajištění integrity dat
- Zajištění dostupnosti

5.3 Příklady metodik pro analýzu rizik

- **CRAMM** (CCTA Risk Analysis and Management Method)
  - Vychází z norem BS7799, ISO/IEC 27001:2005
  - Vyžadováno např. u informačních systémů pro NATO
- **OCTAVE-S** (Operationally Critical Threat, Asset and Vulnerability Evaluation) – pro menší firmy
- **RISK IT –** kompatibilní s metodikou Cobit

5.3.1 RISK IT



Obrázek 5 Analýza rizik – přehled hrozeb pro IT/ICT
[Zdroj: FISCHER, U. Risk IT [prezentace]. Rolling Meadows, USA : ISACA, 2009. Dostupné z: http://www.isaca.org/Knowledge-Center/Standards/Documents/Risk-IT-Overview.ppt]

Výhodou metodiky RiskIT je její kompatibilita s metodikou řízení IT/ICT COBIT (metoda je popsána v kapitole o řízení IT/ICT); obě metodiky pocházejí do stejných autorů.

5.3.2 Metodika Octave-S

OCTAVE-S

Fáze	Proces	Aktivita
Fáze 1: Návrh profilů hrozeb pro aktiva	Proces S1: Určení informací o organizaci	S1.1 Stanovení kritérií pro posuzování dopadů
		S1.2 Určení aktiv organizace
		S1.3 Zhodnocení bezpečnostních postupů organizace
	Proces S2: Vytvoření profilů hrozeb	S2.1 Zvolení kritických aktiv
		S2.2 Určení bezpečnostních požadavků na kritická aktiva
		S2.3 Určení hrozeb pro kritická aktiva
Fáze 2: Určení zranitelností infrastruktury	Proces S3: Šetření počítačové infrastruktury ve vztahu ke kritickým aktivům	S3.1 Zjištění přístupových cest
		S3.2 Analyzování procesů spojených s technologiemi
Fáze 3: Vytvoření bezpečnostní strategie a plánů	Proces S4: Identifikování a analýza rizik	S4.1 Ohodnocení dopadů hrozeb
		S4.2 Stanovení pravděpodobností pro hodnotící kritéria
		S4.3 Stanovení pravděpodobností výskytu hrozeb
	Proces S5: Vytvoření strategie ochrany a plánů pro zmírnění dopadů	S5.1 Zhodnocení současné bezpečnostní strategie
		S5.2 Zvolení přístupů pro snížení dopadů hrozeb
		S5.3 Vytvoření plánů pro snížení rizika dopadů hrozeb
		S5.4 Určení změn v bezpečnostní strategii
		S5.5 Definování dalších kroků

5.4 Postup analýzy rizik

- Stanovení hranice
- Identifikace aktiv
- Stanovení hodnoty aktiv (a jejich seskupení)
- Identifikace hrozeb
- Analýza hrozeb a pravděpodobnost jevu
- Měření rizika

5.5 Bezpečnostní studie

- Odhad hrozeb
- Škoda způsobená incidentem – dočasná nebo trvalá?
- Analýza zranitelnosti
- Odhad dopadů incidentu:
  - Stanovením finančních nákladů
  - Stupnice 1-10
  - Ohodnocení – nízký, střední, vysoký
- Analýza rizik: riziko je potenciální možnost, že daná hrozba využije zranitelnosti

5.6 Bezpečnostní politika

Bezpečnostní politika je soubor pravidel, který má za úkol zajistit bezpečnost IS s přihlédnutím k nákladové efektivitě a musí odpovídat na tyto otázky:

- Kdo nese zodpovědnost?
- Kdy to bude efektivní?
- Jak to bude vynuceno?
- Kdy a jak to bude uvedeno do praxe?

.

Standardní kroky pro zajištění bezpečnosti:

- studie informační bezpečnosti – aktuální stav,
- riziková analýza,
- tvorba bezpečnostní politiky - vytýčení cílů,
- bezpečnostní standardy – pro naplnění cílů bezpečnostní politiky,
- bezpečnostní projekt – technická opatření,
- implementace bezpečnosti – nasazení výše uvedeného,
- monitoring a audit – prověřování, zda vytvořené bezpečnostní mechanismy odpovídají dané situaci.

Bezpečnost je *kompromis* mezi **úrovní zabezpečení** a **náklady**.

5.7 Business Continuity Management

Business Continuity Management je manažerská disciplína, která se zaměřuje na identifikaci potencionálních dopadů, jež organizaci hrozí po havárii. Vytváří rámec pro zajištění určité míry odolnosti a schopnosti reagovat na neočekávané události

5.8 Havarijní plán

- Postup a reakce v případě havárie, poruchy nebo nefunkčnosti IS
- Součást „Bezpečnostní politiky“
- Uživatel IT/ICT by měl vědět KOHO a JAK informovat v případě poruchy

Je tedy rozdíl, pro koho je havarijní plán určen. Havarijní plán pro koncového uživatele nebude obsahovat postupy řešení problémů, ale většinou pouze kontakty na osoby, schopné a oprávněné problém řešit. V případě nepřetržitých provozů bývá v havarijním plánu telefonní číslo na hot-line. Havarijní plán pro administrátory a správce IT/ICT budou obsahovat konkrétní postupy pro Disaster Recovery (obnovu po výpadku).

Shrnutí hlavních bodů kapitoly:

- Analýza rizik – posuzuje aktiva (co má pro firmu hodnotu), hrozby a pravděpodobnost jejich výskytu
- Pro analýzu rizik existují metodiky – doporučené postupy na základě dobrých praktik – jak co nejlépe provádět analýzu rizik a stanovit bezpečnostní politiku
- Bezpečnostní politika – soubor opatření pro zamezení hrozeb a minimalizaci dopadů v případě jejich výskytu. Stanovuje rozsah opatření, náklady, zodpovědnost za řešení
- Havarijní plán – postup řešení v případě výskytu hrozby (výpadku systému, havárie). V případě koncového uživatele obsahuje informace, jak kontaktovat osoby, které zajistí zprovoznění systému

Kontrolní otázky:

1) Co je cílem analýzy rizik?
2) Jak lze kategorizovat hrozby v IT/ICT?
3) Která z uvedených hrozeb je nejhůře řešitelná a odstranitelná?
4) Proč vznikly metodiky pro analýzu rizik?
5) Jaké metodiky analýzy rizik znáte?
6) Jaký je postup při sestavení bezpečnostní politiky?
7) Co by měla obsahovat bezpečnostní politika?
8) Co bude obsaženo v havarijním plánu pro administrátora systému?
9) Co bude obsaženo v havarijním plánu pro koncového uživatele systému?

Použitá literatura:

1) FISCHER, U. Risk IT [prezentace]. Rolling Meadows, USA: ISACA, 2009. Dostupné z WWW: <http://www.isaca.org/Knowledge-Center/Standards/Documents/Risk-IT-Overview.ppt>.
2) HUBNER, M. Pohled nejen CIO na informační bezpečnost : příručka manažera. Praha: TATE International, 2012. ISBN: 978-80-86813-25-7
3) DOUCEK, P. Řízení bezpečnosti informací. Praha: ProfessionalPublishing 2011, 240 stran. ISBN: 978-80-7431-050-8

6 Strukturovaný přístup k analýze a návrhu IS

V této kapitole se seznámíte se strukturovaným přístupem k analýze a návrhu informačních systémů. Strukturovaný přístup oproti objektově orientovanému přístupu odděluje modelování datových struktur a modelování funkcí systému. Pro model datové struktury se používají Entitně-relační diagramy (ERD), pro modelování funkcí a datových toků Diagramy datových toků (DFD). Stavy systému a události vedoucí k změně stavu lze zachytit pomocí Stavových diagramů (STD). Celý model pak lze doplnit datovým slovníkem, který slouží ke kontrole metadat nebo FlowChart, pomocí kterého můžeme zaznamenat algoritmus řešení.

Stručný obsah kapitoly:

- Úvod do strukturované analýzy
- ERD – Entitně relační diagramy
- DFD – Diagramy datových toků
- STD – Stavové diagramy
- DD – Datový slovník

Tato kapitola předpokládá základní znalosti pojmů z oblasti relačních databázových systémů.

Získáte:

- Znalosti k čemu je strukturovaná analýza.
- Přehled, jaké typy digramů se v strukturované analýze používají
- Znalosti, k čemu jsou konkrétní digramy určeny a co lze pomocí nich modelovat
- Představu, k čemu je Yourdanova strukturovaná analýza

Budete umět:

- Definovat účel strukturované analýzy
- Vysvětlit, k čemu se používají ERD, DFD a STD diagramy
- Popsat, jaký je rozdíl strukturované a objektově orientované analýzy

Budete schopni:

- Na základě získaných znalostí vytvořit jednoduchý návrh datového a funkčního modelu informačního systému

Specifikace času potřebného pro nastudování kapitoly: 60 minut.

6.1 Úvod do strukturované analýzy

Klasickou metodou analýzy a návrhu informačních systémů je **strukturovaný přístup**, navržený v 70. letech (Tom DeMarco, Ken Orr, Larry Constantine, Vaughn Frick, Ed Yourdon, Steven Ward, Peter Chen…).

Základy strukturovaného přístupu položil Tom deMarco v roce 1979 v práci „Strukturovaná analýza a specifikace systémů“. Mezi jeho základní doporučení patří:

- rozdělit systém na subsystémy;
- používat grafické znázornění (grafické modely) systému;
- před implementací vytvořit logický model systému.

Strukturovaný přístup modeluje systém pomocí následujících prostředků:

1. ERD – Entity Relationship Diagram
2. DFD – Data Flow Diagram
3. FSD – Function Structure Diagram
4. STD – State Diagram
5. DD – Data Dictionary
6. Structure Chart
7. Flow Diagram

Oproti objektově orientovanému přístupu k analýze a návrhu strukturovaná analýza odděluje datový model od modelu chování a funkcí. Oproti UML v objektově orientovaném modelování je tedy obtížnější udržet konzistenci návrhu. Na druhé straně ale datový model vyjádřený pomocí ERD dává rychlejší orientaci pro návrh tabulek v relační databázi.

6.2 Filozofie strukturovaného návrhu

- Produkty analýzy musí být udržovatelné
- Velké problémy rozděleny na menší
- Použití grafického vyjádření
- Odlišení logické a fyzické úrovně
- Logický model má za cíl seznámení uživatele se systémem před jeho vytvořením a implementací
- Člení projekt na malé dobře definované aktivity
- Určuje posloupnost těchto aktivit a vzájemnou interakci
- Snaha vytvořit specifikaci, které rozumí uživatelé i návrháři

6.3 Úrovně modelu systému z hlediska abstrakce

- **Konceptuální model** – rozpoznání základních datových objektů a jejich vztahů – návrh – co je obsahem systému, návrh je nezávislý na technologickém prostředí
- **Logický model** – relační schéma (včetně integritních omezení) – určuje jakou mají data strukturu, není zatížen konkrétní implementací. Může být vyjádřen formou ERD a DFD diagramů.
- **Fyzický datový model** – implementace v konkrétním databázovém produktu

Úrovně modelů ze strukturované analýzy odpovídají těmto oblastem:

ANALÝZA – DESIGN – IMPLEMENTACE

6.4 Strukturovaný přístup – data a funkce

Strukturovaný přístup k analýze a návrhu pohlíží na informační systém ze dvou různých úhlů pohledu:

– z pohledu **dat**

– z pohledu **funkcí**

Každý má svoji specifickou logiku (ERD a DFD).

Abstrakce **část - celek (agregace).** Tato abstrakce se typicky používá ve funkčním modelu, kde se dělí systém na subsystémy, části subsystémů atd. Pro agregaci je typická principiální neomezenost dělení.

Abstrakce **specifický podtyp - obecný typ (generalizace)**. Tato abstrakce se typicky používá v datovém modelu, kde je možné uvažovat o jednotlivých specifických variantách nadřízeného pojmu (entity, objektu). Na rozdíl od agregace není nadřízený celek definován jako souhrn podřízených částí, ale jako nositel jejich společných vlastností (atributů).

Je důležité, že tyto dva základní typy abstrakce jsou vzájemně neslučitelné a tím tvoří jádro základního rozporu mezi funkčním a datových modelem.

Objektově orientované metody se pokoušejí tento rozpor překonat zapouzdřením obou - dat i funkcí - do jediného objektu.

6.5 ERD (Entity Relationship Diagram)

Peter Chen, 1976, http://bit.csc.lsu.edu/~chen/chen.html

ER diagramy slouží k modelování struktury informačního systému a databáze v grafické podobě.

Cílem ERD je zmapovat data ukládaná do databáze a jejich vzájemné vztahy.

Výstupem ERD je popis logické struktury databáze:

- **Entita** – objekt, který je předmětem zájmu
- **Atribut** – elementární datový prvek, který entitu blíže charakterizuje
- **Relace** – vztah mezi dvěma entitami
- **Kardinalita vztahu** – mocnost vztahu mezi entitami: 1:1, 1:N, N:1, M:N
- **Parcialita vztahu** – povinnost nebo volitelnost vztahu

Vztah je informace, kterou si systém musí pamatovat, nelze ji odvodit.

*Je rozdíl mezi ERD (Chen) a relačním datovým modelem (Codd), tj. je rozdíl mezi pojmy **Relationship** (vztah) a **Relation** (relace, která odpovídá v databázi tabulce) – viz dále vysvětlení modelů.*

ER model popisuje **statický pohled na systém**.

6.6 DFD (Data Flow Diagram)

Cílem DFD je modelování datových nebo řídicích toků v systému (v grafické podobě).

DFD diagramy popisují funkce systému.

U DFD diagramů existuje několik forem záznamu, nejčastěji se používá notace dle DeMarca nebo Yourdana. Vycházejí z teorie grafů.

DFD se skládá z prvků „**proces**", „**datový sklad**", „**terminátor**". Datovým skladem v tomto pojetí rozumíme jakékoli úložiště dat.

DFD diagramy mají hierarchickou úroveň, procesy se dají postupně zjemňovat. Na nejvyšší úrovni stojí **kontextový diagram** – jedná se o speciální případ DFD diagramu, kdy je systém zobrazen jako jediný proces a naznačuje vztah systému s okolím. Na dalších úrovnícj jsou popisovány jednotlivé procesy a datové nebo řídicí toky.

Dekompozice DFD na nižší úrovně až na úroveň základních elementárních funkcí:

- Nesmí existovat proces, který nemá žádné vstupy a přesto produkuje datové toky
- Nesmí existovat proces, který pouze spotřebovává data a nemá výstupy

6.7 FSD (Fuction State Diagram)

Účelem FSD je zobrazit dekompozici systému na funkční celky (subsystémy), zdokumentovat funkční hierarchii systému a poskytnout pohled na vyvíjený systém se zaměřením na jeho hierarchickou strukturu.

FSD diagram se skládá z

- funkce (proces, systém)
- vazba

Funkce dělíme na:

- procesní
- dialogové
- řídicí

Rozdíl mezi DFD a FSD je následující:

- FSD je zaměřen na hierarchickou strukturu subsystémů, jde o statický pohled
- DFD je zaměřen na datové toky a procesy transformující vstup na výstup, jde o dynamický pohled na systém

6.8 STD (State Transition Diagram)

Stavový diagram (STD) obsahuje sled stavů, v jakém se systém (nebo jeho část) může nacházet a za jakých podmínek může dojít ke změně stavu. Modeluje časově závislé chování systému.

- Důležitý z hlediska pochopení logiky systému
- Stavy jsou statické, změna stavu je většinou důsledek nějaké události.
- Vyjádření – např. vývojové diagramy
- Může být hierarchický.

STD diagram musí mít jeden počáteční bod a jeden nebo více koncových bodů. Samotný stav je vnímán jako statický. Diagram eviduje podmínku, při které systém přejde z jednoho stavu do druhého.

6.9 Data Dictionary (datový slovník)

Datový slovník slouží k formalizovanému popisu dat systému z pohledu uživatele.

Metadata (data o datech):

- Spravuje databázový stroj
- Přístup jen ke čtení

Vytvoříme-li datový slovník, který obsahuje všechny položky databáze, dostaneme informaci, zda se některé položky nevyskytují v databázi vícenásobně. Tím pádem můžeme upravit datový model a snížit redundanci.

6.10 Structure chart

Slouží pro znázornění hierarchie programových modulů systému, ve tvaru grafu (stromu). Kořenem je hlavní programový modul, uzly znázorňují dílčí volané moduly.

6.11 Flow chart (vývojový diagram)

Zachycuje algoritmus. Alternativní vyjádření algoritmu, vhodnější pro strukturované programování je strukturogram. Pro modelování systému má Flow Chart nejmenší důležitost, lze použít pro popisy funkcí.

6.12 Návrh informačního systému

Výše uvedené diagramy slouží pro návrh informačního systému. Návrh může být dále doplněn **případy užití**, modelem spolupráce nebo funkčním modelem. Důležitá je také volba architektury, dnes převažuje architektura **třívrstvá** (prezentační, funkční a datová vrstva).

6.13 Přehled modelů a metodik strukturované analýzy a návrhu

6.13.1 Logické modelování Gane – Sarson

Krátce po zveřejnění DeMarcovy "Strukturované analýzy a specifikace systému" (Structured System analysis: Tools and Techniques) publikovali Chris Gane a Trish Sarson svoji práci "Strukturovaná analýza systému". Metoda Gane/Sarson byla opět založená na **použití DFD jakožto výchozího modelu IS**, nově však zde bylo začleněno i datové modelování pomocí ERD.

- Co je příčinou pohybu dat?
- Kdy k pohybu dat dojde?
- Jak velká oblast systému se podílí na zpracování vstupu a na vytvoření výstupu?

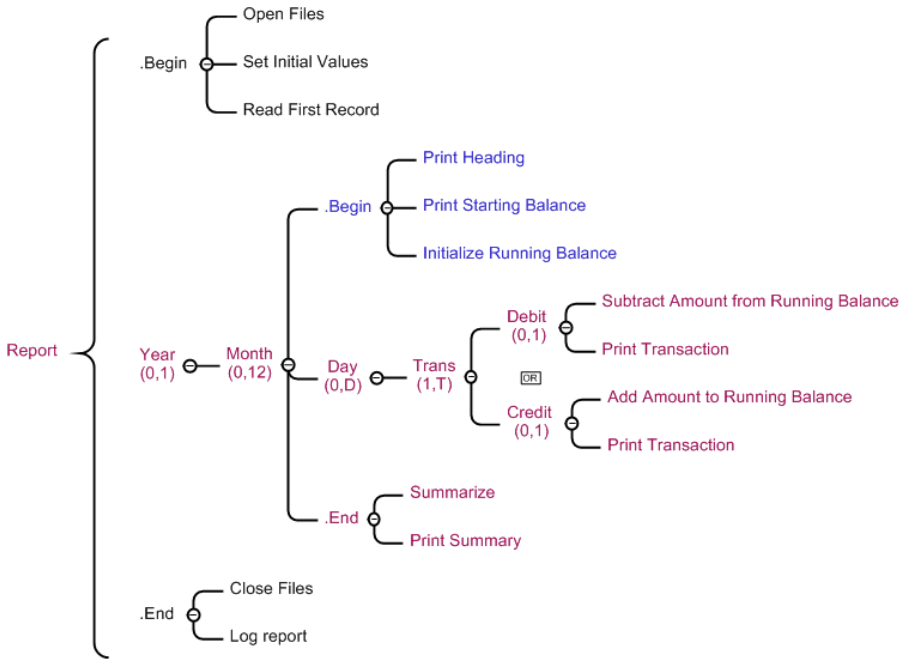
Kroky pro vytvoření modelu:

1. Systémový DFD
2. Hrubý ERD
3. Analýza entit a vztahů mezi nimi
4. Detailní ERD

5. Normalizace datového modelu
6. Úprava DFD podle ERD

6.13.2 Datově orientovaný přístup – Warnier/Orr

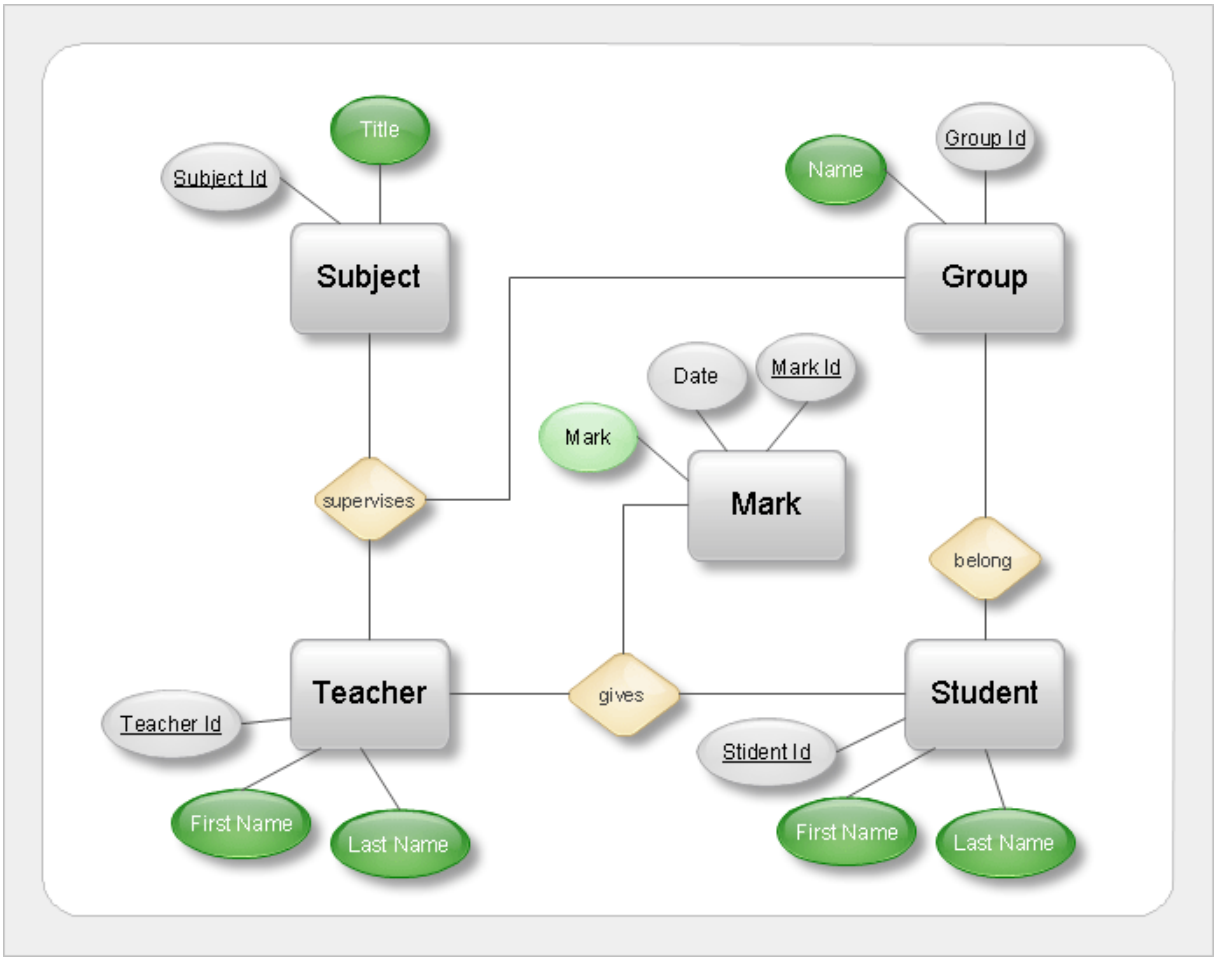
Vývoj této metody se datuje již od roku 1972. Jean Warnier (Francie) a Kenneth Orr (USA) svoji metodu založili na odvození logického datového modelu na základě **analýzy požadovaných výstupů systému**. Specifikují se také procesy výběru dat z databáze, transformace dat a výstupy. Model je rozšířen o stavové diagramy.



Obrázek 6 Ukázka Warnierr-Orr diagramu – specifikace požadavků na report [Zdroj: http://www.mindapp.com/mind-map-examples/warnier-orr/report/]

6.13.3 Entitně-relační model - Chen

Entitně vztahový model. Pracuje s pojmy **entita, atribut, vztah (relationship, association).** Entita je množina objektů stejného typu. V objektovém modelování odpovídá pojmu „entita“ pojem „třída“. Atribut je datová složka entity. ERM nemusí vždy odpovídat relační implementaci. Při popisu vztahu v ERD diagramech by se vztah neměl zaměňovat za popis funkce.



Obrázek 7 Příklad ERD diagramu v notaci Chen [Zdroj: http://ereyanitrikandi.blogspot.cz/]

6.13.4 Datový model

Zachycuje vztahy mezi daty, oproti konceptuálnímu modelu (Conceptual Model), který je odrazem reálného světa (a používá pro vyjádření ERD nebo prvky z objektového modelování…).

6.13.5 Relační datový model – Codd

Jedná se o vysoce abstraktní model určený pro modelování relačních databází. Pracuje s **pojmy relace (relation), atribut, funkční závislost**. V relačním datovém modelu nepracujeme s pojmy tabulka a řádek; pojmu „tabulka“ z fyzické realizace RDM v databázi zde odpovídá pojem „relace“. Mezi relacemi existuje funkční závislost. RDM by měl projít procesem **normalizace**, která vychází z koncepce funkčních vztahů. Cílem normalizace je navrhnout správné relace bez nadbytečných atributů (tj. bez redundance). Existuje celkem pět normálních forem (pátá je nejstriktnější), přičemž za správně navrženou databázi se považuje třetí normální forma. Na základě RDM lze pak vytvořit jeho fyzickou realizaci v konkrétním databázovém prostředí. Při fyzické realizaci pak podle skutečných potřeb aplikací může dojít k určité míře **denormalizace**, za účelem zrychlení reportovací vrstvy.

6.13.6 PERM – Physical ERM

Entitně relační model vztažený na prvky, které se skutečně vyskytují v relační databázi. Zde pracujeme s pojmy tabulka a řádek tabulky, který reprezentuje výskyt entity.

6.13.7 Yourdonova moderní strukturovaná analýza (YMSA)

Edward Yourdon vytvořil konceptuální popis systému, skládající se ze tří částí:

a) Datový model
b) Model chování
c) Model řízení

Yourdon doporučuje pro modelování nástroj následujících vlastností:

a) Musí být grafický
b) Podpora TOP-DOWN přístupu
c) Minimalizace redundance – řešení pomocí Data Dictionary
d) Snadná čitelnost
e) Předvídat chování systému

Jádro metodiky (1989) spočívá v nalezení **esenciálního modelu**, který vyjadřuje podstatu systému, je dlouhodobě stabilní a je nezávislý na použité technologii a implementaci. Z esenciálního modelu je následně odvozen implementační model.

Esenciální model se skládá ze dvou částí:

a) Model okolí
b) Model chování
c) Implementační model

Model okolí se skládá z:

a) Dokument o účelu systému
b) Kontextový diagram
c) Seznam událostí

V kontextovém diagramu je celý systém znázorněn jako jeden proces. Cílem je zachytit výměnu informací s okolím, tedy hlavně s uživateli systému.

Model okolí a zejména dokument o účelu systému je určen pro zákazníka a management. Model chování naproti tomu je vytvářen pro návrháře a obsahuje popis chování uvnitř systému. Pro model chování se používají DFD, ERD a DD, vytváří se hierarchická struktura DDF, která se následně vyvažuje. V posledním kroku se doplní o STD diagramy a **minispecifikaci** a dokončení datového slovníku.

Po dokončení modelu chování může být vytvořen implementační model. Zde se určí, které procesy budou automatizovány a které budou manuální (v modelu vystupují jako terminátory a v reálu reprezentují uživatele, který procesy manuálně provádí).

6.13.8 SSADM – Structured System Analysis and Design Method

Vyvinuta firmou LBMS, velmi rozšířená ve Velké Británii, kde se stala standardem. Vývoj software je rozdělen do šesti následujících etap:

- analýza stávajícího systému,
- specifikace požadavků,
- výběr technických možností,
- návrh logických dat,
- návrh logických procesů,
- fyzický návrh.

Hlavními nástroji pro modelování systému jsou diagramy datových toků, logické datové struktury (LDS) a životní cykly entit (ELH). Pro DFD používá odlišnou notaci.

6.13.9 Data Structured System Development – DSSD

Datově orientovaný přístup k analýze. Nejlepších výsledků je dosaženo, když struktura programu odpovídá hierarchické struktuře datového modelu. K datovému modelování používá diagramy entit (obdoba ERD).

6.14 Příklady aplikací pro podporu strukturální analýzy

- Microsoft Visio
- Dia - http://live.gnome.org/Dia
- CASE Studio Toad Modeler - http://www.casestudio.com/enu/default.aspx
- Visual Paradigm - http://www.visual-paradigm.com/
- Edge Diagrammer - http://www.pacestar.com/

Shrnutí hlavních bodů kapitoly:

- Strukturální analýza a návrh je ucelená metodika pro vytváření modelu informačních systémů či aplikací
- Návrh datového modelu se provádí pomocí ERD diagramů, model datových toků a funkcionality je popsán pomocí DFD diagramů, oba modely jsou vzájemně odděleny
- Další součástí modelu tvořeného strukturovaným přístupem mohou být stavové diagramy (STD) pro popis stavů a událostí, které mohou způsobit změnu stavu, datový slovník pro kontrolu metadat nebo další prvky jako je třeba FlowChart

Kontrolní otázky:

1) Co je cílem strukturované analýzy?
2) Co je to ERD diagram, z čeho se skládá a k čemu se používá?
3) Co je to kontextový diagram?
4) Jak mohu modelovat funkční a datové toky?
5) Jaký je vztah mezi ERD a DFD?
6) K čemu se používá diagram STD?
7) Jaká je funkce datového slovníku?
8) Jaký je rozdíl mezi strukturovaným přístupem k návrhu IS a přístupem objektově-orientovaným?

Použitá literatura:

1) TIETZE, P. Strukturální analýza: úvod do projektu řízení. Vyd. 1. Praha: Grada, 1993, 228 s. ISBN 80-854-2445-2.
2) STRAKA, M. Vývoj databázových aplikací. Vyd. 1. Praha: Grada, 1992, 162 s. ISBN 80-854-2443-6.
3) MIKULÁŠEK, A.: Modely strukturované analýzy pro nevidomé. Bakalářská práce, Masarykova univerzita, Fakulta informatiky, Brno 2008. Dostupné z http://is.muni.cz/th/134645/fi_b/Bakalarska_prace.txt
4) ŘEPA, Václav. Analýza a návrh informačních systémů. 1.vyd. Praha: Ekopress, 1999, 403 s. ISBN 80-861-1913-0.
5) ŘEPA, Václav. Podnikové procesy: procesní řízení a modelování. 2., aktualiz. a rozš. vyd. Praha: Grada, 2007, 281 s. ISBN 978-80-247-2252-8.

7 Objektový přístup k analýze a návrhu IS

V této kapitole se seznámíte s objektově orientovaným přístupem k analýze a návrhu informačních systémů. Tento přístup odstraňuje základní nedostatek strukturovaného přístupu – data a funkce jsou modelovány společně. To dává více komplexní pohled na modelovaný systém. V kapitole jsou uvedeny základní pojmy z teorie objektového přístupu a seznámíte se s nejznámější objektovou metodikou návrhu – UML. Objektově orientovaný návrh systému je vhodný především tam, kde samotná fyzická realizace software bude vytvářena s použitím objektově orientovaných programovacích jazyků.

Stručný obsah kapitoly:

- Objekt a třída
- Základní myšlenky objektového přístupu
- Objektově orientovaný návrh
- UML
- Přehled diagramů UML

Tato kapitola předpokládá znalosti základní orientace v analýze informačních systémů.

Získáte:

- Znalosti o postupu objektově orientovaného návrhu informačního systému.
- Přehled o pojmech z oblasti objektového návrhu
- Přehled o metodice UML

Budete umět:

- Vysvětlit základní pojmy a principy objektového návrhu
- Popsat diagramy metodiky UML a vysvětlit, k čemu slouží
- Vysvětlit rozdíl mezi strukturovaným a objektovým přístupem k návrhu systémů

Budete schopni:

- Díky základní orientaci v metodice UML a jejích diagramech má posluchač základy k jejich reálnému použití pro řešení praktických úloh v oblasti návrhu aplikací.

Specifikace času potřebného pro nastudování kapitoly: 60 minut.

7.1 Objekt a objektový přístup

Objektově orientovaný přístup je založen na objektech.

Objekt je struktura, která má definované

- **Vlastnosti** (tomu odpovídají atributy objektu)
- **Chování** (tomu odpovídají funkce, pro které se používá termín „metody“)

Vlastnosti a chování je zapouzdřené v jednotlivých objektech. Každý objekt je schopen reagovat na události.

Informační systém z pohledu objektově orientovaného přístupu je chápán jako množina spolupracujících objektů.

Objektový přístup lépe odpovídá chování reálného světa.

V oblasti programování je hlavní ideou objektového přístupu **znovupoužitelnost.**

7.2 Základní myšlenky objektového přístupu

- **Zapouzdření (encapsulation) -** objekt je pro nás „černou skříňkou – zajímá nás, co dělá a ne z čeho se skládá.
- **Dědičnost** - možnost vytvářet nové instance objektů s možností přidat nové prvky. Opakem dědičnosti je **generalizace** – z konkrétních dílčích částí sestavujeme obecnou společnou část
- **Polymorfismus (vícetvarost)** – různé chování objektů na stejný podnět – metoda stejného jména může mít trochu jinou funkcionalitu, přestože je tato funkcionalita pojmově blízká. Analogie je pojem „otevřít“ a rozdíl mezi funkcí „otevřít dveře“ a „otevřít láhev“. Příkladem polymorfismu je například funkce Read()/Write() v operačním systému – podle situace lze uplatnit na soubor nebo na nějaké zařízení.
- **Genericita** – možnost vytvářet parametrizovatelné programové moduly

7.3 Srovnání relačního a objektového modelu

Relační model – prvky reálného světa se snažíme zobrazit do pevných předem připravených struktur

Objektově orientovaný model – pro prvky reálného světa vytváříme objekty, které se jim podobají

7.4 Základní prvky objektového přístupu

7.4.1 Objekt
Konkrétní prvek vyskytující se v systému, který má definované vlastnosti a chování.

7.4.2 Třída (Class)
Kategorie (skupina) objektů, které mají podobné vlastnosti a stejné nebo podobné chování.

7.4.3 Atributy (Properties)
Slouží k popisu vlastností třídy (objektu).

7.4.4 Metody (Methods)
Algoritmy definující reakce objektu na vzniklé situace.

K objektu můžeme přistupovat pouze využitím metod.

7.4.5 Instance třídy - objekt
Daný objekt je instancí určité třídy, tedy jejím konkrétním výskytem. Objekt ze strany třídy dědí atributy a metody. Třídu tedy můžeme chápat jako šablonu, která je použita k vytvoření objektu. Z jedné třídy lze vytvořit více instancí objektu, každá instance má alokovanou vlastní paměť. Třídu si můžeme představit například jako technický výkres, zatímco objekty jakožto instance třídy jsou pak výrobky vytvořené podle technického výkresu.

7.4.6 Hierarchie tříd
Třídy mají svou hierarchii, založené na dědičnosti.

Hierarchii tříd je možné budovat dvěma způsoby:

- **Směr specializace** – návrh obecných tříd a postupnému odvozování speciálních podtříd
- **Směr generalizace** – návrh objektů, jejich tříd a postupnému zobecňování

7.4.7 Abstraktní třída
Z abstraktních tříd se neodvozují konkrétní objekty; slouží k odvozování speciálnějších podtříd (potomků). Neexistuje konkrétní instance této třídy.

7.4.8 Rozhraní (interface)
Pojmenovaná množina operací, které třída používá ve vztahu k jiným třídám. Vztah mezi třídou a jejím rozhraním říkáme **realizace.**

7.5 UML

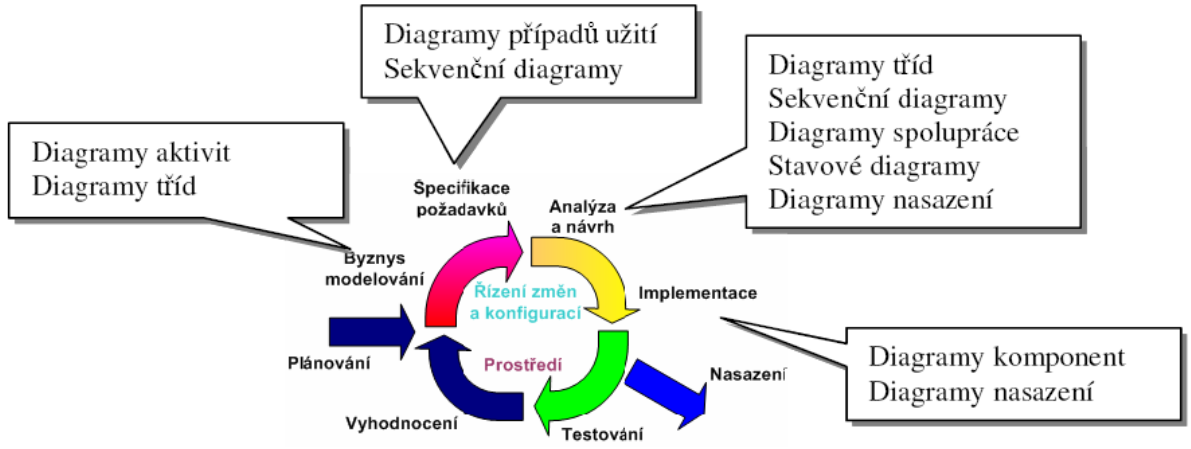
UML neboli **Unified Modeling Language** je nástroj pro modelování informačních systémů založený na objektově orientovaném přístupu.

Je přijat sdružením OMG (Object Management Group) jako standard pro tvorbu informačních systémů.

Jedná se o nejrozšířenější objektovou notaci, která je podporována všemi CASE nástroji.

Jednou z důležitých charakteristik UML je jeho nezávislost na metodologiích. Možná i z toho pramení jeho široké rozšíření jakožto implementačního jazyka.

Použití UML je široké – od prostředku pro obecný popis systému po detailní návrh, který lze využit pro generování kódu v objektově orientovaném programovacím jazyce.



Obrázek 8 Použití jednotlivých diagramů UML

7.6 Diagramy UML

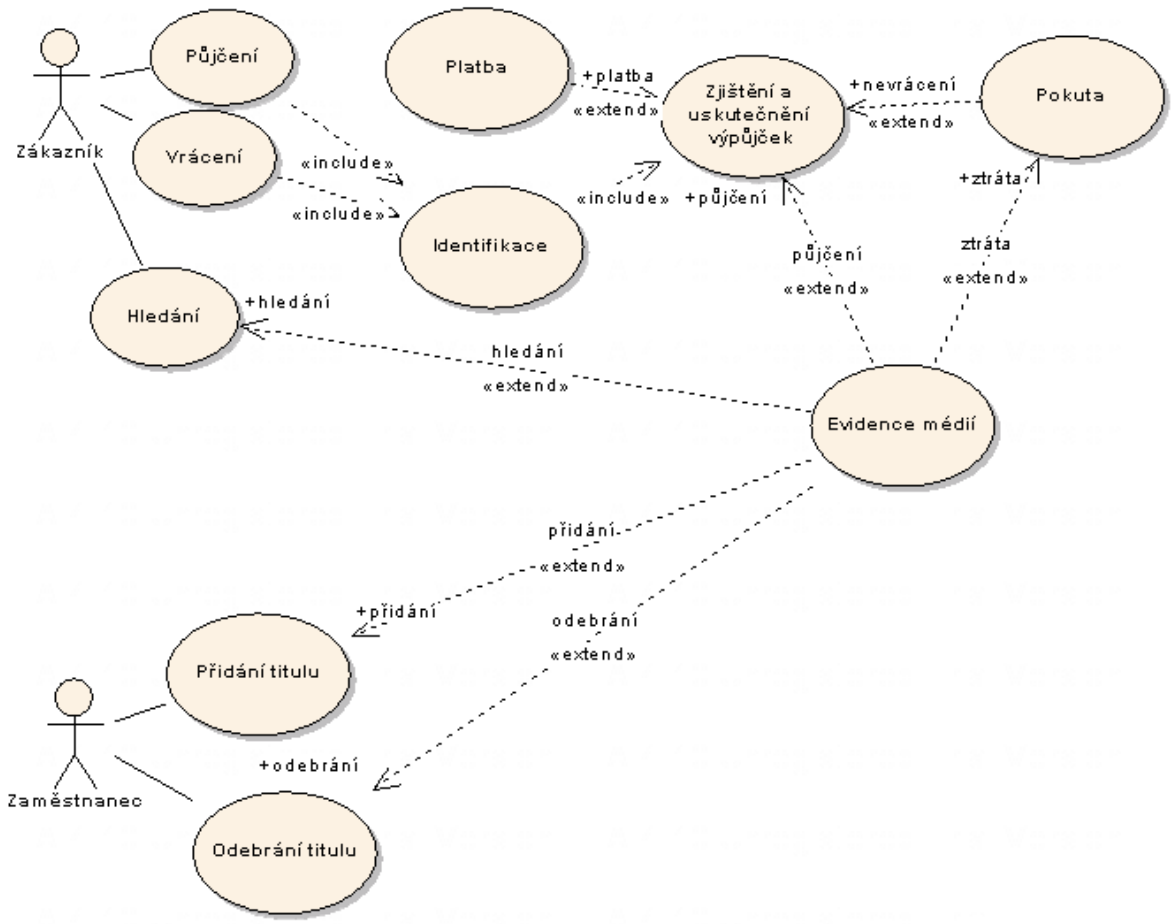
UML nabízí k popisu mnoho typů diagramů, rozdělených do tří skupin. První skupina diagramů popisuje statickou strukturu aplikace. Druhá skupina popisuje různé aspekty dynamického chování. Třetí slouží k organizaci a správě aplikačních modulů.

Statická struktura	**Dynamické chování**	**Správa modulů**
diagram tříd (Class Diagram)	Use case diagram	balíčky (Packages)
objektový diagram (Object Diagram)	sekvenční diagram (Sequence Digram)	subsystémy (Subsystems)
komponentový diagram (Component Diagram)	diagram činností (aktivit) (Activity Diagram)	modely (Models)
diagram nasazení (Deployment Diagram)	diagram spolupráce (Collaboration Diagram)	
	stavový diagram (Statechart Diagram)	

Diagram komponent a diagram nasazení reprezentují **implementační model**.

*UML **nezahrnuje DFD diagramy** (Data Flow Diagramy), které slouží v strukturované analýze k popisu chování systému. Datové toky a jiné typy diagramů, které nebyly do UML zahrnuty, nezapadají čistě do konsistentního objektově orientovaného paradigmatu. Diagramy aktivit a diagramy spolupráce splňují mnoho z toho, co lidé chtějí od DFD. Diagramy aktivit jsou zároveň vhodné pro modelování workflow.*

7.7 Use Case



Obrázek 9 Příklad diagramu Use case

Jedním ze základních diagramů je Use Case:

- Užití systému, jeho funkce
- Definice **požadavků na funkcionalitu!**
- Vysvětlení činnosti **aktorů** (účastníků systému – osoba, jiný IS, HW komponenta…)
- Vymezení hranic systému
- Dynamický pohled na vyvíjený systém z pohledu zákazníka (uživatele)

Pro další studium diagramů UML lze využít např. http://uml.czweb.org/index.html

Řešený příklad návrhu aplikace v UML najdete na http://interval.cz/clanky/navrh-aplikaci-v-jazyce-uml-textova-specifikace-pripadu-uziti/

.

7.8 Nedostatky UML

Za nedostatek UML můžeme považovat neschopnost UML poskytovat prostředky pro návrh uživatelského rozhraní a datových modelů. Například Activity diagram v UML neumožňuje zachytit místo pro uložení informací, podobně jako to umí diagram DFD ve strukturovaném návrhu.

Nedostatečností UML v oblasti datového modelování je to, že UML neposkytuje vhodné prostředky pro datové modelování, a proto se navrhuje rozšíření notace UML pomocí vhodných *stereotypů*, což jsou prostředky jazyka UML pro jeho vlastní rozšiřování. Soubor určitých *stereotypů*, které rozšiřují UML pro určitou oblast užití, se nazývá *UML extensions*.

7.9 RUP

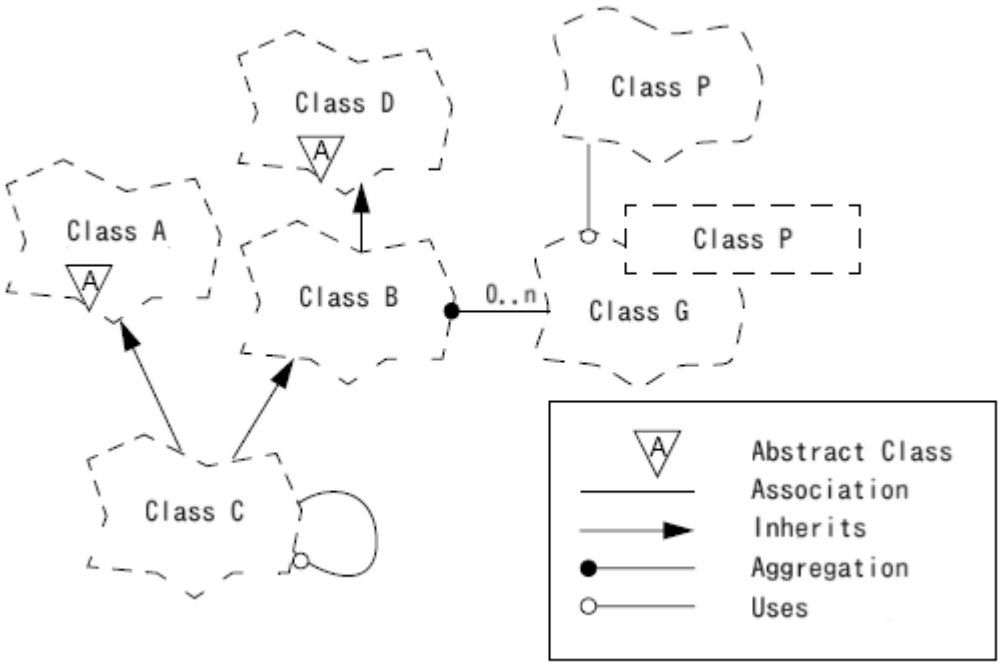
Na metodice UML je založena metodika vývoje softwaru RUP:

- Sběr požadavků
  - Kvantifikátory „nikdo“, „všichni“, „vždy“, „nikdy“
  - Dokument o představě
  - Model požadavků – aktoři, UseCase, vazby, subjekty
  - Odhad pracnosti řešení
- Analýza – CO se má udělat (ne JAK!)
  - Architektonická, analýza případů užití, analýza tříd, analýza balíčků (třídy do tématických celků)
- *Fáze:*
  - Zahájení – Rozpracování – Konstrukce – Zavedení
- *Postupy:*
  - Požadavky – analýza – návrh – implementace - testing

7.10 Ostatní objektově orientované metodiky

- Booch
- Object Modelling Technique
- Objectory/Object-Oriented Software Engineering

- Object-oriented Process, Environment and Notation
- Coad-Yourdon
- Shlaer/Mellor
- Class, Responsibility, Collaborators
- Business Object Notation
- Business Object Relation Modeling



Obrázek 10 Ukázka diagramu z metodiky Booch [Zdroj: Wikipedia]

7.11 Shrnutí objektového přístupu

Chceme-li vyvinout návrh systému od koncepce k podrobnému objektově orientovanému návrhu, musíme provést tyto kroky:

1. Pochopit a definovat kontext a externí interakce se systémem
2. Navrhnout systémovou architekturu
3. Identifikovat základní objekty v systému
4. Vyvinout modely návrhu
5. Specifikovat rozhraní

Shrnutí hlavních bodů kapitoly:

- Objekt, třída
- Zapouzdření, dědičnost, polymorfismus
- UML
- Diagramy UML
- Use Case – případy užití
- Ostatní objektové metodiky návrhu

Kontrolní otázky:

1) Co je podstatou objektově orientované analýzy a návrhu systému?
2) Co je to objekt a jaké má vlastnosti?
3) Jaký je rozdíl mezi objektem a třídou?
4) Vysvětlete pojmy dědičnost, zapouzdření a polymorfismus.
5) Co je to UML?
6) Jaké diagramy používá UML?
7) K čemu slouží Use Case diagram?

Použitá literatura:

1) KANISOVÁ, H., MULLER, M. UML srozumitelně. Brno: Computer Press, 2007. 176 s. ISBN: 8025110834.
2) ARLOW, J., NEUSTADT, I. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. Brno: Computer Press, 2007, 567 s. ISBN 978-80-251-1503-9.
3) HOFFER, Jeffrey A, Joey F GEORGE a Joseph S VALACICH. Modern systems analysis and design. 3rd ed. Překlad David Krásenský. Upper Saddle River, N. J.: Prentice Hall, c2002, xxxii, 733 p. ISBN 01-303-3990-3
4) SOMMERVILLE, I.: Softwarové inženýrství. Brno: ComputerPress, 2013. ISBN: 978-80-251-3826-7

8 Řízení projektů v IT

V této kapitole se seznámíte se základními principy projektového řízení při projektech v IT. Principem každého projektového řízení je řízení času, nákladů a funkcionality (a kvality). V závěru kapitoly se seznámíte také s řízením týmů a identifikace týmových rolí. V kapitole je zmíněna metodika MBTI pro určení typologie osobnosti jakožto jeden z možných prostředků pro řešení sestavení týmu.

Stručný obsah kapitoly:

- Základní princip projektového řízení
- Řízení času
- Řízení nákladů
- Řízení požadavků, funkcí a kvality
- Řízení týmu, rýmové role, MBTI

Tato kapitola vyžaduje orientaci v pojmech z informačních systémů. Znalosti z projektového řízení a řízení týmů jsou výhodou.

Získáte:

- Znalosti o principech projektového řízení v IT
- Přehled o postupech pro řízení času, nákladů a kvality funkcionality v IT projektech
- Znalosti o problematice týmových rolí a sestavování týmů

Budete umět:

- Vysvětlit, co je podstatou projektového řízení v IT/ICT projektech
- Popsat specifika při řízení času, funkcí a nákladů v IT projektech
- Použít dotazník MBTI k určení typologie osobnosti a její vhodnosti pro určité týmové role

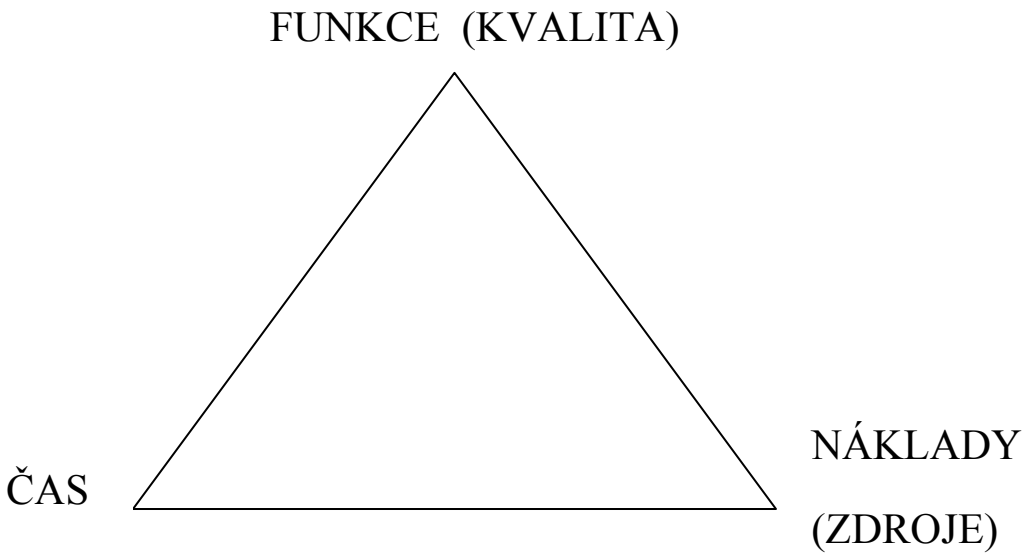
Budete schopni:

- Aplikovat základní principy projektového řízení na jednoduché projekty v IT
- Posluchač bude schopen vyhodnotit test MBTI a porozumět výsledkům ve vztahu k tvorbě týmu

Specifikace času potřebného pro nastudování kapitoly: 50 minut.

8.1 Řízení projektu

Řízení projektu v jakémkoli oboru je vždy o řízení tří oblastí, které spolu úzce souvisí:



Předmětem projektové řízení je tedy dodání produktu (nebo služby) mající požadovanou funkcionalitu (a kvalitu) v termínu dodávky a při dodržení plánovaných nákladů.

8.2 Řízení času

Řízení času spočívá v naplánování dílčích činností tak, aby byl dodržen finální termín projektu. Na začátku projektu je sestaven harmonogram prací.

Pro sestavování harmonogramu můžeme použít např.

a) Ganttovy diagramy
b) CPM (Critical Path Method)
c) Plánování kritického řetězu (aplikace teorie omezení, Theory of Constraints)
d) PERT

a další.

Pro plánování činností existuje celá řada softwarových nástrojů jako je Microsoft Project, Open Project atd.

Ganttův diagram se označuje také jako pruhový diagram a slouží pro vytvoření časového plánu projektu, aktivity jsou zaznamenány v kalendářovém formátu. Pracuje s prvky jako je: milník, aktivita, souhrnný úkol, závislost mezi úkoly atd.. Definování závislostí mezi úkoly

pak umožňuje přeplánování. Ganttův diagram může také porovnávat plán a skutečnost (tzv. sledovací Ganttovy diagramy).

CPM (Critical Path Method; metoda kritické cesty) *– kritickou cestou je nejdříve možný termín dokončení projektu, což je nejdelší „cesta" v diagramu aktivit, kde nejsou časové rezervy (není to nejkratší cesta – nejkratší je doba, po kterou je nutné zvládnout všechny úkoly). CPM tedy umožňuje zjistit nejdříve možný termín dokončení projektu.*
Může být více kritických cest v jednom projektu?
Může se kritická cesta v průběhu řešení projektu změnit?

PERT (Program Evaluation and Review Techniques) *vychází z metody CPM. Základem je předpoklad, že trvání dílčí činnosti má stochastický charakter – náhodná hodnota s určitým rozdělením pravděpodobnosti. Jedná se o aplikaci CPM na vážený průměr odhadů dob trvání.*

Plánování kritického řetězu je obdobné CPM, ale na rozdíl od této metody bere v úvahu omezení zdrojů. Pracuje se souběžným zpracováním (jeden zdroj pracuje souběžně na více aktivitách) nebo s faktem, že určitou aktivitu může provést pouze specializovaný pracovník.

Všechny zde uvedené metody se používají k odhadu celkové doby projektu.

8.3 Řízení nákladů

Řízení nákladů projektu spočívá v řízení tří oblastí:

a) Odhad nákladů
b) Vytvoření rozpočtu
c) Řízení nákladů s cílem dodržet rozpočet

8.3.1 Odhad nákladů

Náklady IT projektu se skládají z těchto položek:

1. Náklady na řízení projektu
2. Náklady na administrativní činnosti
3. HW
4. SW
    a. Licencovaný (OS, databáze, kancelářský SW…)
    b. Vlastní vývoj
5. Testování
6. Školení a podpora
7. Rezerva

8.3.2 Typy odhadu nákladů

a) Řádový (hrubý) odhad – Rough Order of Magnitude (ROM) – rozhodnutí o výběru projektu
b) Rozpočtový odhad
c) Konečný definitivní odhad – pro nákup prostředků

8.3.3 Techniky odhadu

a) Odhad podle analogie
b) Odhad zdola nahoru (od ceny jednotlivých položek)
c) Parametrické modelování (z kvantifikovaných parametrů)
d) Metody jako je např. COCOMO (Constructive Cost Model) – parametrický model odhadu nákladů - vychází z počtu funkčních bloků, počtu knihoven atd.

Měření efektivity – EVM (Earned Value Management) – nakolik je projekt v souladu se stanovenými cíli (baseline, schválený plán projektu)

Overrun – překročení nákladů.

8.4 Řízení rozsahu (funkcionality)

Řízení rozsahu se týká funkcionality – co všechno se musí udělat, aby byla dosažena.

Odhad rozsahu prací:

a) Sestavení podle zásad (podle existujících pravidel)
b) Sestavení podle analogie (s jiným obdobným projektem, WBS – Work Breakdown Structure)
c) Postup shora dolů (funkční bloky rozdělujeme na dílčí položky)
d) Postup zdola nahoru (detailní položky skládáme a seskupujeme do kategorií až do nejvyšší úrovně)
e) Mapa myšlení (mind mapping) – centrální hlavní myšlenka a z ní větve

Problém:

- Potlačení neúplných požadavků
- Potlačení měnících se požadavků
- Nafukování projektu

Techniky pro řízení požadavků:

- Zavedení procesu řízení požadavků
- Modelování – use-case
- Prototypování
- JAD (Joint Application Desing) – společné týmové definování požadavků
- Řízení testování
- Stanovení priority

- Stanovená pevného data – deadline pro definování požadavků – po tomto datu buď finanční prostředky navíc, nebo se musí vzdát jiného požadavku.

Součástí projektového trojúhelníku čas-náklady-funkce je také **řízení kvality**. S kvalitou souvisí aplikace norem ISO 9000. Kvalita (jakost) je definována jako shoda výrobku (služby) s požadavky zákazníka. ISO 9000 předepisuje minimální požadavky, které musí organizace splnit pro zajištění kontroly kvality, aby získala certifikaci.

8.5 Řízení týmu

8.5.1 Maslowowa hierarchie potřeb

Potřeba růstu a seberealizace se dostaví, až budou uspokojeny potřeby nižší úrovně (fyziologické – bezpečí – sociální).

8.5.2 Herbergova teorie motivační hygieny

- faktory vedoucí k uspokojení - motivátory
- faktory vedoucí k neuspokojení (např. prostředí – je-li příjemné, nevede k zvýšení produktivity, ale je-li nepříjemné, vede k nespokojenosti…)

Zdroj motivace dle Herzberga je pocit osobního úspěchu a uznání.

Benefity jako zvýšení platu zvyšují produktivitu pouze krátkodobě. Hlavní motivátor je pocit seberealizace.

8.5.3 Mc Clellandova teorie získaných potřeb

Potřeby člověk získává, formují se životními zkušenostmi.

Hlavní kategorie potřeb:

a) **dosažení** (Achievment) – potřeba něčeho dosáhnout – *náročné projekty s dosažitelnými cíli*
b) **zařazení** (Affiliation) – potřeba uznání daná být členem úspěšné skupiny - *tým*
c) **moc** (Power) – potřeba osobní nebo institucionální moci – *vedení týmu*

U každého člověka převládá jedna, maximálně dvě kategorie.

8.5.4 Mc Gregorova teorie X- Y

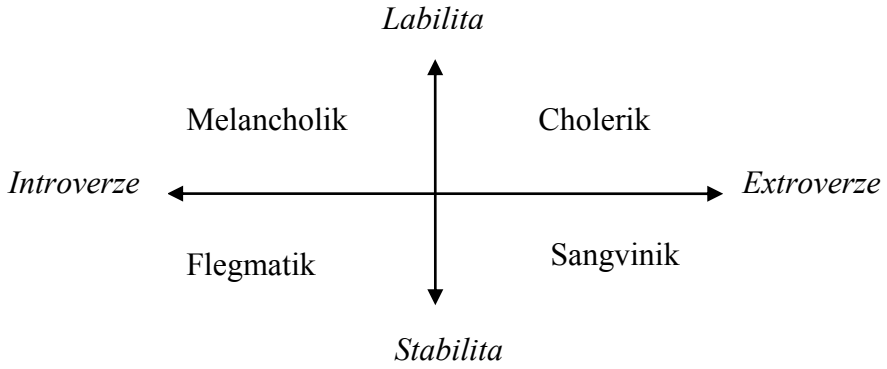
„The Human Side of Enterprises“

Teorie X – zaměstnanci jsou z principu líní a mají snahu se práci vyhýbat – je nutné používat nátlakové prostředky a kontrolní mechanismy. Pracovník chce být řízen, vyhýbá se přímé odpovědnosti a má nízké ambice; vyhledává jistotu.

Teorie Y – lidé mají práci jako přirozenou činnost, nejlepší odměnou je uspokojení z pocitu seberealizace; zaměstnancům lze důvěřovat, že splní úkoly, pokud zaměstnanci mohou důvěřovat, že je nadřízení podpoří

8.5.5 MBTI

Typologie osobnosti MBTI (Myers-Briggs Type Indicator) - dle kognitivních funkcí; vychází z **Jungovy teorie temperamentu**.



MBTI zařadí člověka do jedné z 16 skupin; umožňuje odhad, zda daný člověk je vhodný na určitý typ práce.

Posuzuje následující kombinace:

- Extroverze – introverze
- Myšlení (Thinking) – cítění (Feeling)
- Intuice (iNtuition) – smyslové vnímání (Sensing)
- Hodnocení, plánování (Judgment) – vnímání (Perceiving)

DeMarco v knize „Peopleware“ (1987) uvádí, že úkolem manažera není přinutit lidi pracovat, ale odstranit politické překážky a nechat lidi pracovat.

DeMarco & Lister statistickým šetřením zjistili, že u vývoje software není žádná korelace mezi produktivitou a použitými programovacími jazyky, počtem roků praxe a platem. Naopak klíčovým faktorem pro růst produktivity je vytvoření pracoviště s klidným pracovním prostředím.

Shrnutí hlavních bodů kapitoly:

- Projektové řízení: ČAS – ZDROJE – FUNKCE
- Řízení času – Microsoft Project
- CPM, PERT
- Odhady funkcionality
- Řízení nákladů – techniky odhadu nákladů
- Řízení týmu – Maslowova hierarchie potřeb, Herbergova teorie motivační hygieny, McGregorova teorie X-Y
- Typologie osobnosti – Jungova teorie temperamentu, MBTI

Kontrolní otázky:

1) Co je základem řízení projektů?
2) Jaké nástroje mohu použít pro řízení času?
3) Jak se provádí odhady pracnosti?
4) Jaké metody se používají pro řízení nákladů?
5) Jaké znáte teorie v souvislosti s řízením týmu?
6) Co je to MBTI?

Úlohy pro samostudium:

1) Najděte na Internetu on-line test pro typologii osobnosti MBTI
2) Proveďte a vyhodnoťte test MBTI
3) Prostudujte 16 osobnostních typů dle MBTI
4) Jaké jsou k dispozici prostředky pro řízení kvality?

Použitá literatura:

1) ČAKRT, M. Typologie osobnosti pro manažery: manažerské styly, rozhodování, komunikace, konflikty, týmová práce, time management a změny. 2., rozš. a přeprac. vyd. Praha: Management Press, 2009, 306 s. ISBN 978-80-7261-201-7.
2) GUCKENHEIMER, S., PEREZ J. Efektivní softwarové projekty: metodiky efektivního vývoje softwaru. Vyd. 1. Překlad Jan Kuklínek, Jan Pokorný. Brno: Zoner Press, 2007, xi, 255 s. ISBN 978-80-86815-62-6.
3) KOMZÁK, T. Řízení IT projektů pro úplné začátečníky. 1. vyd. Brno: Computer Press, 2013, 213 s. ISBN 978-80-251-3791-8.
4) KOSZLAJDA, A. Zarządzanie projektami IT: przewodnik po metodykach. Gliwice: Helion, 2010. ISBN 978-832-4618-040
5) SCHWALBE, K. Řízení projektů v IT. Vyd. 1. Překlad David Krásenský. Brno: Computer Press, 2007, 720 s. ISBN 978-80-251-1526-8.

9 Metodiky vývoje SW

V této kapitole se budeme věnovat problémům spojeným s vývojem informačních systémů. Seznámíte se s disciplínou nazývanou softwarové inženýrství, jejímž předmětem jsou metodiky pro tvorbu software. Vysvětlíme si rozdíl mezi úkolocentrickým a hodnotocentrickým přístupem k tvorbě software a seznámíme se s nejčastěji používanými metodikami vývoje software. Vysvětlíme si rozdíl mezi klasickým a agilním přístupem k řízení projektu vývoje.

Stručný obsah kapitoly:

- Klasifikace metodik
- Hodnotocentrický a úkolocentrický přístup k vývoji softwaru
- Tradiční metodiky návrhu
- Agilní metodiky návrhu
- SCRUM

Tato kapitola vyžaduje základní znalosti pojmů z informačních systémů.

Získáte:

- Po prostudování kapitoly budete znát základní nejčastěji používané metodiky vývoje software; a to skupinu metodik klasických jako je například vodopád, spirála či RUP a skupinu metodik agilních, jako např. SCRUM nebo Test-driven Development.

Budete umět:

- Vysvětlit rozdíl mezi úkolocentrickým a hodnotocentrickým přístupem k vývoji softwaru
- Popsat základní principy tradičních metodik vývoje
- Vysvětlit rozdíl mezi tradičními a agilními metodikami
- Popsat výhody a nevýhody agilního přístupu k vývoji
- Popsat princip metodiky SCRUM

Specifikace času potřebného pro nastudování kapitoly: 60 minut.

9.1 Problémy a specifika vývoje software

Vývoj prvních programů byl prováděn nadšenci, programy byly šité na míru. Žádná metodika vývoje SW v té době neexistuje. Vývoj SW byl vnímán jako výzkum. Cíl, co bude software dělat, si určoval programátor sám na základě velmi vágního zadání (ostatní často neměli představu, co by od počítačů a software mohli chtít). Vývoj většinou nebyl omezen žádnými termíny (až byl software hotov, došlo k jeho nasazení).

V důsledku stále většího rozmachu informatiky a počtu projektů, které neskončily úspěšně (ve smyslu – nebyly dokončeny včas nebo byly příliš nákladné nebo nebyly dokončeny vůbec) se na přelomu 60. a 70. let 20. století se začalo mluvit o tzv. „softwarové krizi“.

9.2 Softwarová krize

- Neúnosné prodražování projektů
- Neúnosné prodlužování projektů
- Nízká kvalita programů
- Nízká produktivita programátorů
- Neefektivita vývoje
- Nejistota výsledku

„Softwarová krize“ vyvolala potřebu používání metodik pro řízení vývoje software.

9.3 Nejčastější problémy vývoje SW

- Zpoždění
- Vysoká chybovost
- Neplnění požadované funkčnosti
- Nedostatečná výkonnost
- Složité uživatelské rozhraní
- Obtížná udržovatelnost programu

9.4 Základní příčiny problémů

- Podcenění projektu a špatný odhad (čas, náklady)
- Špatné zadání
- Nedostatečná analýza
- Přílišná složitost projektu
- Přehnaný důraz na technologii (použití novinek bez zkušeností)
- Špatná kvalita programového kódu (chybový, nesrozumitelný, pomalý, nedostatečně komentovaný)
- Nevhodné metodiky, postupy, technologie
- Nedostatečné testování
- Špatné projektové řízení

9.5 Softwarové inženýrství

Předmětem softwarového inženýrství jsou metodiky pro řízení vývoje softwaru.

- Proč potřebujeme tyto metodiky?
- Čím je vývoje softwaru specifický oproti jiným odvětvím?

Softwarové inženýrství je **zavedení a používání inženýrských principů tak, abychom dosáhli ekonomické tvorby softwaru**.

.

Takto vytvořený software je spolehlivý a pracuje na dostupných výpočetních prostředcích.

9.6 Přístup k vývoji IS

Z pohledu tvůrců informačních systémů můžeme přístup k vývoji IS rozdělit do dvou oblastí:

a) Úkolocentrický
b) Hodnotocentrický

9.6.1 Úkolocentrický přístup k vývoji IS

Úkolocentrický (work-down) přístup k vývoji IS je založen na základech **projektového řízení**. Vývoj a implementace IS postupuje podle daného plánu (harmonogramu) stanoveného na začátku řešení projektu, který se rozpadá do řady plnění dílčích konkrétních úkolů (termínů). Úkoly jsou postupně realizovány, plnění harmonogramu je průběžně kontrolováno.

Úkolocentrický přístup je vhodný u projektů s dobře známým návrhem a nízkými riziky - cíl je jasně definován, řešení má stanovenou koncepci.

Hodnota, kterou IS zákazníkovi přinese, je jednoznačně stanovena na začátku projektu.

Velkou výhodou tohoto přístupu je kontrola a přehled nad postupem prací, projekt je řiditelný podle jasně stanovených kritérií.

Nevýhody

- Změny vzniklé v průběhu řešení projektu se obtížně integrují
- Nápady a postupy, které by pro zákazníka mohly znamenat přínos, které však nebyly v původním návrhu, nemohou být uplatněny (cílem je dodržet harmonogram a rozpočet)
- Nedostatečná analýza a s tím vzniklá neurčitost řešení může způsobit vážné komplikace

- Zákazník má v průběhu projektu minimální vliv na řešení projektu
- Kvalita je definována splněním specifikace (dodržením harmonogramu)

9.6.2 Hodnotocentrický přístup k vývoji IS

U hodnotocentrického (value-up) přístupu vývoj probíhá v iteracích se zákazníkem. Je kladen důraz na hodnotu IS pro zákazníka, která se může průběžně zvyšovat. I při hodnotocentrickém přístupu je vytvářen plán, ale postup podle harmonogramu není hlavním cílem; objeví-li se během vývoje skutečnosti a postupy, které vedou k tomu, že zákazník dostane více, má tato inovace přednost před dodržením harmonogramu za každou cenu. Části SW, které se v průběhu řešení ukážou jako nepotřebné nebo zastaralé, mohou být z projektu zrušeny. Snahou je dodat zákazníkovi čím jak nejdříve fungující části tak, aby zákazník mohl řešení ověřit a připomínkovat a jeho podněty mohly být do vývoje zapracovány.

Hodnotocentrický přístup má:

- Zaměření na průběžné vytváření hodnoty
- Zapojení zákazníka do řešení projektu
- Očekáváme nejistotu a jsme na ni připraveni
- Zdrojem hodnoty jsou jednotlivci – uznáním zvyšujeme kreativitu
- Výkon povzbuzujeme skupinovou zodpovědností za výsledku a efektivitu týmu
- Kvalita je definována přínosem pro zákazníka (termín může být posunut, pokud to zákazníkovi něco přinese)
- Změny a odchylky jsou brány jako přirozená součást řešení
- Lze využít kreativity řešitelů

Samozřejmě, i tento přístup má své nevýhody – obtížnější řiditelnost projektu, včetně vyhodnocení postupu prací, nebo hrozba, že projekt nebude uspokojivě ukončen apod. Tento postup je také nepoužitelný u velkých investičních akcí, kde investor a uživatel systému jsou různé subjekty.

9.7 Doporučení k vývoji IS

Jedním z důležitých kritérií úspěchu IS jsou:

- snadnost ovládání
- rychlost uvedení na trh

Je výhodnější napsat aplikaci, která bude rozšiřitelná v budoucnu než monolitický systém, který bude obsahovat všechny možné funkce hned na začátku (včetně funkcí, které nikdy nikdo nepoužije).

Aplikace psaná na míru konkrétním požadavkům, postrádající jakékoli rysy obecnosti, může být v budoucnu obtížně modifikovatelná.

Tendence psát příliš obecné systémy často vede ke dvěma výsledkům – nesrozumitelné a neprůhledné architektuře nebo k systémům, které umožňují díky množství nastavitelných parametrů vyvolat chování, jenž ani tvůrce systémů není schopen dokumentovat.

Je tedy třeba najít rozumný kompromis mezi obecností a konkrétními požadavky.

Přístup k vývoji je odlišný, pokud vytváříme jednorázovou aplikaci, která bude fungovat po omezenou dobu, nebo se snažíme o vývoj „krabicového software“, který bude fungovat v mnoha firmách. V druhém případě určitě stojí za to navrhnout celou koncepci systému tak, aby byl maximálně obecný a parametrizovatelný. V případě jednorázového určení může být snaha po přílišné obecnosti ztráta času a prostředků.

9.8 Klasifikace metodik vývoje SW

Významné tradiční metodiky návrhu a vývoje IS:

- Model „napiš – oprav“ (Build and Fix)
- Striktní posloupnost fází (Stagewise)
- Vodopádový model (Waterfall)
- Spirálový model
- Další metodiky: RUP, UP, USDP, …

Některé agilní metodiky vývoje IS:

- Extrémní programování
- Crystal
- SCRUM
- Aspect Oriented Programming
- Test Driven Development
- Lean Development

9.9 Tradiční metodiky

9.9.1 Model „Napiš – oprav“ (Build and Fix)

Implementace -> Dodání -> Opravy chyb

Tato metodika je nejstarší známou metodikou z oblasti vývoje SW, zde je uvedena jen pro úplnost a s výjimkou malých jednorázových projektů je její použití nevhodné.

9.9.2 Stagewise model

Tento model vývoje software byl definován v roce 1957.

Založen na striktní posloupnosti fází:

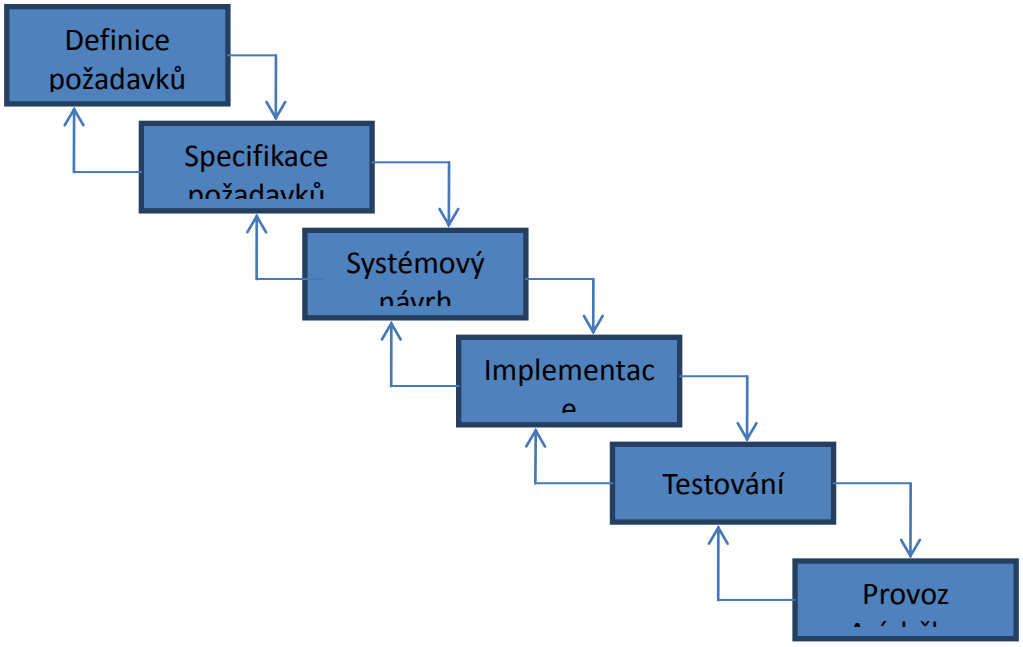
- Definice problému
- Analýza
- Specifikace požadavků
- Návrh
- Architektura
- Implementace (+testování)
- Provoz

Metodika Stagewise je založena na následujících skutečnostech:

Absence zpětné vazby, neprovádí se revize žádné fáze, nerevidují se požadavky ani se nehledají rizika.

9.9.3 Model vodopád (Waterstage)

- 1970 Winston Royce
- Každá etapa má stanovený přesný cíl a dokumenty, které musí v jejím průběhu vzniknout
- Na konci každé etapy dochází k jejímu vyhodnocení a případně přepracování nebo opravení
- Možnost vrátit se zpět do předchozí etapy
- Pokračuje se teprve tehdy, je-li etapa zcela dokončena a schválena (pak již návrat není možný)



Obrázek 11 Fáze v metodice vodopád

Výhody:

- Jednoduchý
- Ideální pro řízení
- Vnáší disciplínu do vývoje

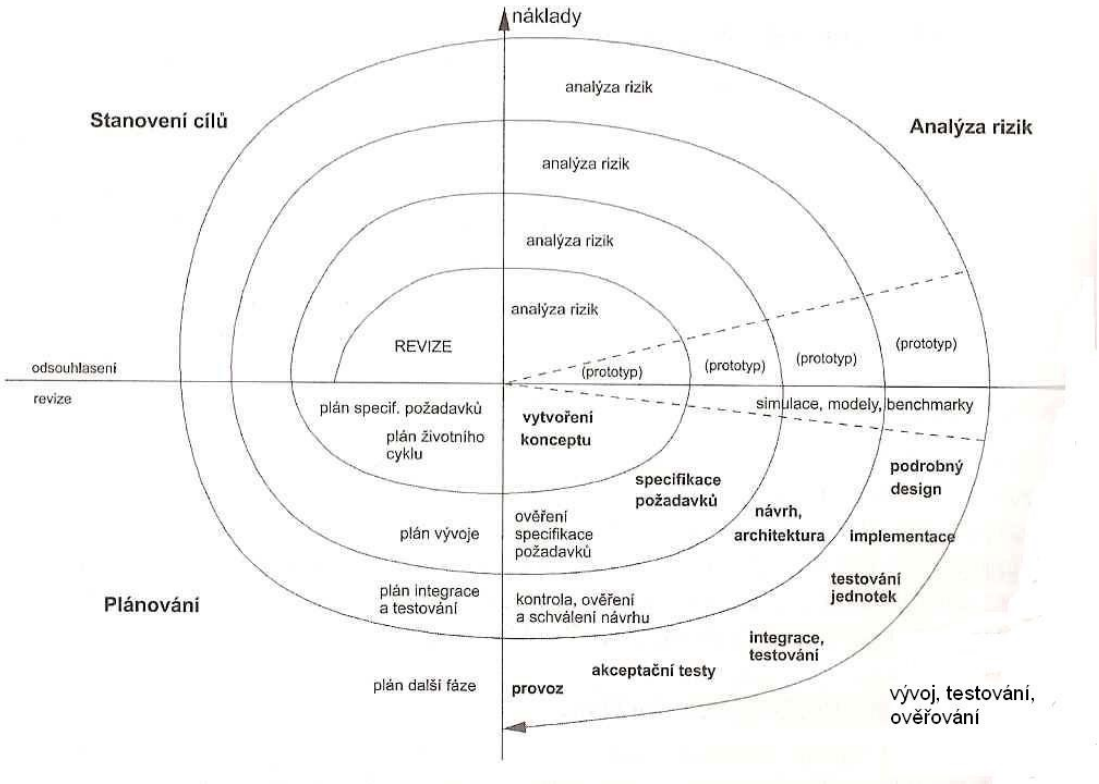
Nevýhody:

- Dodávka formou „velkého třesku“
- Určitá nepružnost
- V době mezi analýzou a nasazením se mohou změnit požadavky

„Začnu-li padat, nezastavím se dříve, než se rozbiji o kámen zvaný předvedení.“

9.9.4 Spirálový model

- 1985, Barry Boehm
- Zavádí iterativní přístup a opakovanou (důslednou) analýzu rizik



Obrázek 12 Model „spirála“
[Zdroj: http://ari.wikidot.com/zivotni-cyklus-is]

Rizika – situace nebo události, které mohou způsobit nesplnění cílů projektu.

Vychází se z předpokladu, že na začátku je obtížné nebo až nemožné přesně specifikovat všechny funkce.

Ve spirálovém modelu se tedy stanoví jen obecný rámec. Vyvine se část aplikace, následuje předvedení a konzultace se zákazníkem a po zpracování jeho námětů a připomínek pokračuje se dalším cyklem analýza-implementace-prototyp. Oproti agilním metodikám popsaným v další části se však uživatel neúčastní procesu vývoje a vidí pouze mezivýsledky z jendotlivých cyklů.

Výhody:

- Vytváří prostředí pro vývoj znovupoužitelných komponent
- Je komplexní a vhodný i pro složité projekty (díky důrazu na plánování)
- Včasné vyloučení nevhodných řešení

Nevýhody:

- Celková komplikovanost
- Software není uvolněn před dokončením posledního cyklu
- Změna požadavků je možná pouze po dokončení cyklu
- Pro nové druhy aplikací (např. internetové) je nepružný

Je vhodný pro rozsáhlé projekty!

9.9.5 RUP – Rational Unified Process

- Vývoj probíhá v iteracích; dělí se na 4 fáze: zahájení, projektování, realizace, předání (zákazníkovi nebo do další fáze vývoje)
- Robustní, propracovaná metodika, vhodná pro větší projekty a rozsáhlejší týmy
- Komerční produkt (dříve firma Rational, dnes IBM)
- Je založen na využití UML (grafický modelovací jazyk používaný pro návrh systémů, vychází z objektového přístupu, pomocí sady diagramů modeluje statický i dynamický pohled na systém)

Klíčové principy:

- Iterativní vývoj softwaru (vychází ze spirálového modelu, průběžná detekce rizik)
- Správa a řízení požadavků (požadavky se v čase mění)
- Použití komponentové architektury
- Vizuální modelování softwaru (za účelem porozumění systému; využití UML)
- Průběžné zajišťování a ověřování kvality (po předání je nalezení problému dražší)
- Řízení změn (počítáme s tím, že změny nastanou, neřízení změn vede k chaosu)

Výhody RUP:

- Obecnost a mohutnost
- Iterativní přístup – včasné odhalení rizik
- Snazší správa změn
- Provázanost s notací UML, dokumentace
- Výrobce průběžně pracuje na zlepšování metodiky

- Existence doplňkových nástrojů

Nevýhody RUP:

- Komerční, tj. placený produkt
- Rozsáhlost RUP může být na škodu u malých týmů – tým stráví spoustu času implementací metodiky
- Její použití vyžaduje hluboké studium, týká se i projektových manažerů

Klasické metodiky vývoje SW jsou tedy založeny na **striktní definici postupů**.

9.9.6 Unified process

Obdobou RUP je metodika UP (Unified Process), jedná se o její nekomerční verzi.

Unified Process je řízena požadavky a riziky. Je iterativní a inkrementální. Každá iterace zahrnuje všechny fáze běžného vývoje (plánování, analýza, konstrukce, integrace, testování).

Jedna iterace zahrnuje:

1. Zahájení – stanovení proveditelnosti, vytvoření business případu, zachycení požadavků, identifikace rizik
2. Rozpracování – spustitelná verze architektury, zpřesnění rizik a požadavků, definice kvality, plán konstrukce a prostředků
3. Konstrukce – doladění požadavků, implementace
4. Zavedení – opravy chyb, příprava pracoviště k nasazení, přizpůsobení SW a pracoviště, manuály, konzultace.

9.10 Agilní metodiky vývoje software

Postupy předchozích metodik, založené na důsledné analýze a propracovaném návrhu jsou obecně nejlepší.

„Děláte web půl roku? Konkurence mezitím spustila dva…“

Zdánlivě to může vypadat tak, že neexistence zákazníkovy představy, co vlastně chce je výhodou – „něco“ mu dodáme a zákazník bude spokojen. Zákazník sdělí na konci projektu, že výsledek není to, co chtěl. Chce projekt dodělat/předělat – za původně dohodnutou cenu. Svět se mění – zákazník očekává kvalitu, ale není ochoten na ni dlouho čekat.

Tento rozpor se snaží řešit agilní metody snahou o **užší sepětí zákazníka s vývojovým týmem**.

9.10.1 2004 „Manifest agilního vývoje softwaru“

Jedinou cestou, jak prověřit správnost navrženého systému, je co nejrychleji jej vyvinout, předložit zákazníkovi a na základě zpětné vazby upravovat.

Hlavní myšlenkou agilního přístupu je tedy vtáhnout zákazníka do procesu vývoje.

Tradiční přístup - požadavky jsou stanoveny na začátku vývojového procesu a jsou neměnné. Proměnné jsou zdroje a čas.

Agilní přístup považuje za neměnné zdroje a čas, předmětem změn je funkcionalita. Na počátku projektu se stanoví nejdelší možná čas a náklady. Tým v průběhu zakázky komunikuje se zákazníkem a průběžně přehodnocuje priority.

9.10.2 Teze agilního manifestu

- Přijmout a umožnit změnu je efektivnější než se změně bránit
- Je třeba být připraven na nepředvídané události – „jedinou jistotou na projektu je změna".

9.10.3 Princip agilních metod

- Užitná hodnota pro zákazníka
- Změny jsou výhodou (pro zákazníka může být konkurenční výhodou, agilní metodiky neřeší nic, co není momentálně potřebné, protože v budoucnu se to může změnit)
- Časté dodávky (velmi krátké iterace)
- Klíčová je motivace
- Zákazníci spolupracují s týmem
- Úspěch posuzujeme podle fungování a ne podle splnění specifikace
- Zásadní je jednoduchost
- Důvěra a komunikace vedou ke kreativitě
- Jak zvýšíme efektivitu?
- Perfektní návrh a řešení (změna není považována za důkaz jeho nekvality)
- Udržitelný vývoj (přesčasy a práce v noci řeší krátkodobě problémy, ale dlouhodobě snižují produktivitu práce)

9.10.4 Tým agilního vývoje

- Do 10 členů:
    - Kouč
    - Programátoři
    - Časoměřič
    - Stále přítomný pracovník uživatele (asi nejde vždy dodržet, mohou být třeba různí pracovníci, daný pracovník je nepostradatelný)
    - Programátoři pracují ve dvojicích, které se mění
        - První programátor – vymýšlí a píše
        - Druhý programátor – oponuje, kontroluje, spolu vymýšlí
- Místnost pro odpočinek a jednání
- Důraz na využití kreativity
- Dokumentace – minimalizovat (nikdo nečte), jen přehledný zdrojový kód
- Přesčasy dlouhodobě nezvyšují produktivitu práce

Agilní vývoj omezuje:

- rizika spojená s nepřesným zadáním nebo se složitostí budovaného systému
- rizika spojená s fluktuací členů týmu
- rizika spojená s tím, že neexistuje dokumentace v obvyklém rozsahu

- rizika spojená s nedodržováním termínů a překračováním rozpočtů
-

9.10.5 Kdy není vhodné používat agilní metodiky:

- Kritické systémy, kde je nutné přesně dodržovat dohodnuté podmínky a postupy
- Rozsáhlé systémy, které se nedají dobře dekomponovat
- Nejsou k dispozici kvalitní řešitelé
- Není ochota se domlouvat o cíli za pochodu (jak uzavřít smlouvu, jak sankce za neplnění).

9.11 Přehled agilních metodik

- Adaptivní vývoj softwaru
- Extrémní programování
- Lean Development
- SCRUM
- Crystal metodiky
- Test-Driven Development
- Feature Driven Development

9.11.1 SCRUM

- Název pochází z ragby – mlýn hráčů za účelem společného dotažení míče na pozici
- 1995 Schwaber, Beedle
- Vývoj v posloupnostech, krátkých etapách, které se nazývají „**sprinty**“ (max. měsíc)
- Pro každou etapu je stanoven seznam úkolů neboli **backlog**
- Úkoly jsou seřazeny podle priority, určené ve spolupráci se zákazníkem
- Krátké **denní Scrum Meetings** - konkrétní určení činností; které položky z minulého meetingu byly dokončeny a jaké nové úkoly vznikly; předvedení výsledků zákazníkovi
- Tyto schůzky jsou zásadní, určují:
    - Shrnutí dosavadního pokroku
    - Předvedení mezivýsledků
    - Identifikace nových úkolů
    - Zvyšování soudržnosti týmu
    - Odhalování problémů v mezilidských vztazích
- SCRUM počítá s tím, že nelze naplánovat přesný průběh vývoje a proto se o to nepokouší, přesné plánování suplují denní úkoly

Charakteristika projektů podle SCRUM:

a) Flexibilní předměty dodání (obsah dodávky diktován prostředím. Např. co má být výsledkem analýzy? Někdy je lepší specifikace podle norem, někdy je lepší objektový model, jindy dodání prototypu…)
b) Flexibilní harmonogram (flexible schedule) – dodání může proběhnout později, než se očekávalo, ale zákazník s tím musí být ihned srozuměn (toto je obtížné, záleží na úrovni vztahů se zákazníkem, nebezpečí, že to bude vnímat jako problém nebo projev neprofesionality).
c) Malé týmy – ideální 3 – 6 lidí, na jednom projektu může pracovat více týmů.

d) Časté revize.
e) Spolupráce – intenzivní komunikace členů týmu, týmu se zákazníkem i zadavatelem.

Backlog – informace o vlastnostech, funkcích a činnostech, které je třeba implementovat (může být forma tzv. „User Stories“). Modifikovat může pouze manažer projektu, ostatní pouze čtou.

Riziko – silný důraz na analýzu rizik. Revize rizik na konci každé interakce, ale i v průběhu každodenních schůzek.

Sprint – základní vývojová entita (iterace) – fáze vývoj (Develop), zabalení (Wrap), revize (Review), přizpůsobení (Adjust); obvykle 30 dní.

9.11.2 Lean Development

Původně Toyota - pointou je **odstranění všeho zbytečného a minimalizace zásob** (nic nesmí ležet na skladě); později transformováno do oblasti SW.

Pravidla Lean Development:

1. odstranit vše, co je zbytečné
2. minimalizovat zásoby (minimalizovat meziprodukty)
3. maximalizovat tok (= zkrátit čas potřebný na vývoj)
4. vývoj je tažen poptávkou (rozhodnutí dělat čím jak nejpozději je to možné)
5. pracovníci mají pravomoci rozhodovat
6. hlavním cílem je uspokojovat požadavky zákazníků (teď i v budoucnu)
7. zpětná vazba (nebát se změn v učiněných rozhodnutích)
8. odstranit lokální optimalizaci (neustálá optimalizace stávajícího řešení nemají smysl)
9. vybudovat partnerství s dodavateli
10. vybudovat prostředí pro možnost neustálého zlepšování

Pokud věc nepřidává novou hodnotu, je to zbytečnost.

LD stanovuje **šest druhů plýtvání**:

1. nadvýroba (u sw nadbytečné požadavky, které pak nejsou používány)
2. čas tracený čekáním (tester čeká na dokončení funkcí – mezitím může dělat něco jiného)
3. plýtvání související s přepravou (u sw spočívá řešení v automatizaci některých procesů)
4. plýtvání související se zpracováním (vedení týmu by mělo mít přehled kdo co dělá a v jaké fázi se vývoj nachází)
5. nefektivní práce (u sw využití existujících sw nástrojů a řešení, vývojáři o nich často ani neví)
6. defekty ve výrobcích (program bez chyb téměř neexistuje, ale je třeba přijmout opatření k minimalizaci chyb)

Vývojáři musí v každém případě chápat, jak vlastně přispívají ke konečnému cíli a k čemu jejich práce vede!

Uspokojení potřeb uživatele – ten často neví co chce, požadavky se průběžně mění a občas jsou zdánlivě nesmyslné – je lépe pracovat na partnerství a užším vztahu se zákazníkem než ho na začátku nutit podepsat stostránkovou funkční specifikaci.

9.11.3 Feature Driven Development

Hlavní roli hrají vlastnosti výsledného produktu, **vlastnosti produktu řídí vývoj**. Založeno na iterativním vývoji a krátkých iteracích. Modelování – vývoj začíná vytvořením **globálního modelu** systému – z něho by mělo být patrné celkové směrování vývoje – cíl není přinést kompletní přehled funkcí, ale naznačit, z čeho se bude systém skládat a jak bude komunikovat s okolím.

Předpokládá se, že zákazníkovi jsou neustále dodávány beta verze (minimálně každých 1-3 týdne). Zákazník vidí, že vývoj postupuje vpřed (psychologický efekt) a taky má možnost do vývoje zasahovat.

Vlastnost (feature) – malý výsledek (funkčnost) užitečná z pohledu zákazníka.

Vlastnost je charakterizována:

- Měřitelnosti (Je implementovaná funkce totožná s funkcí požadovanou zákazníkem?)
- Srozumitelností (Musíme být schopni vlastnost popsat)
- Realizovatelností (Lze vlastnost dodat? Nebude její vývoj trvat příliš dlouho?)

Postup:

1. Vytvoří se seznam vlastností.
2. Seřadí se podle priority.
3. Nad seznamem probíhá vývoj – průběžně vznikají a zanikají malé týmy mající na starost implementaci konkrétní vlastnosti.
4. Po implementaci nastupuje fáze testování a integrace.

Metodika je vhodná pro menší projekty.

9.11.4 Test Driven Development

Základní myšlenka: testovací kód musí být připraven a dokončen **před** začátkem psaní kódu

1. napíšeme nový test požadované funkce tak, aby selhal
2. začleníme do projektu (do kompletní testovací sady) a ověříme, že selže
3. implementujeme požadovanou funkci
4. znovu spouštíme testy, jestliže není test úspěšný, musíme kód opravit
5. pokud test projde, zařadíme ho do testovací sady (knihovny)

Testování - implementace - návrh

Chyby jsou odchyceny při průchodu testovacím případem. Tato metodika je méně procesně orientovaná, nezabývá se tvorbou specifikací.

Výhody
kvalitní software s předvídatelným a dobře prozkoumaným chováním

Nevýhody
nutnost pevné ruky ze strany řízení projektu (pro programátory nepohodlné psát testy na počátku)

Shrnutí hlavních bodů kapitoly:

- Softwarové inženýrství – inženýrské principy s cílem dosáhnout ekonomické tvorby softwaru
- Softwarová krize – vzrůstající neúspěšnost špatně řízených projektů vývoje IS – reakce na softwarovou krizi je vznik metodik
- Přístup k vývoji IS: úkolocentrický – založený na projektovém řízení a harmonogramech a hodnotocentrický – založený na důrazu na vytvoření hodnoty pro zákazníka
- Rozdělení metodik – klasické a agilní
- Klasické metodiky – projektové řízení vývoje; Stagewise, vodopád, spirála, RUP
- Agilní metodiky – počítají se změnou, pružný vývoj, vtažení zákazníka do vývoje; Extrémní programování, SCRUM, Lean Development, Test-driven Development…
- Agilní manifest

Kontrolní otázky:

1) Co je to softwarová krize a jaké jsou příčiny neúspěchu softwarových projektů?
2) Jaké jsou specifika vývoje SW oproti jiným odvětvím?
3) Jaký je rozdíl mezi úkolocentrickým a hodnotocentrickým přístupem k vývoji SW?
4) Na čem jsou založeny klasické metodiky vývoje SW?
5) Jaké klasické metodiky vývoje SW znáte?
6) Čím se liší agilní metodiky vývoje od klasických?
7) Popište základní principy agilní metodiky SCRUM
8) Jaké jsou výhody a nevýhody klasických a agilních metodik?

Použitá literatura:

1) GUCKENHEIMER, S., PEREZ, J.. Efektivní softwarové projekty: metodiky efektivního vývoje softwaru. Vyd. 1. Brno: Zoner Press, 2007, xi, 255 s. ISBN 978-80-86815-62-6.
2) KADLEC, V. Agilní programování: metodiky efektivního vývoje softwaru. 1. vyd. Brno: Computer Press, 2004, 278 s. ISBN 80-251-0342-0.
3) PALETA, P. Co programátory ve škole neučí aneb Softwarové inženýrství v reálné praxi. Vyd. 1. Brno: Computer Press, 2003, 337 s. ISBN 80-251-0073
4) Agilní manifest, česká verze. [online] [cit 2013-05-19]. Dostupné na www <http://agilemanifesto.org/iso/cs/>

10 Testování software

V této kapitole se seznámíte s problematikou testování software. Testování je nedílná část procesu vývoje informačních systémů a mnozí producenti software již pochopili, že tato fáze může být klíčová pro úspěch tvorby SW. Oddělení pro testování software hrají srovnatelnou roli jako oddělení vývoje. V kapitole si popíšeme základní druhy softwarových testů.

Stručný obsah kapitoly:

- Black box a white box testy
- Dynamické a statické testování
- Testy specifikace
- Testování vstupů
- Testování logiky aplikace
- Testování konfigurace
- Testování kompatibility
- Testování vícejazyčného prostředí a podpory jazyků
- Unit testy
- Další typy testů
- Automatizace testování a testovací scénáře

Tato kapitola nevyžaduje žádné vstupní znalosti.

Získáte:

- Znalosti o typech testování.
- Představu, jak náročný je proces důkladného testování software

Budete umět:

- Popsat, k čemu slouží základní druhy testu software
- Vysvětlit, co je to black box a white box testování
- Objasnit účel základních typů testů software

Specifikace času potřebného pro nastudování kapitoly: 30 minut.

10.1 Testování

Testování SW má podstatný vliv na kvalitu dodaného produktu.

Náklady na odstranění chyby stoupají, v čím pozdější fázi životního cyklu aplikace je chyba nalezena.

Na druhé straně, vytvořit zcela bezchybný SW není možné. Je tedy třeba zvolit vhodný kompromis mezi množstvím testů a uvedením produktu do provozu.

Důvody neopravení chyby:

a) Není dost času
b) Není to chyba (nesprávně pochopená funkce)
c) Oprava chyby je riskantní (oprava může způsobit poškození již ověřené části)
d) Oprava nestojí za to (pravděpodobnost výskytu je nízká)

10.2 Typy testů

10.2.1 Black Box a White Box testy

Test typu Black Box znamená, že testujeme aplikaci, od které nemáme k dispozici zdrojový kód.

U White Box testů je zdrojový kód k dispozici a můžeme tedy k testování využít znalost vnitřní struktury nebo simulovat určité stavy pomocí debugeru.

Grey Box - omezená znalost interních datových a programových struktur za účelem navrhnutí vhodných testovacích scénářů, které se realizují na úrovni black box (např. testy interface).

10.2.2 Statické a dynamické testování

Statický test znamená např. revizi kódu, revizi analytických podkladů, test funkční specifikace.

Dynamické testování znamená, že testujeme aplikaci, která je spuštěna.

10.2.3 Testy specifikace

Je specifikace:

- Úplná
- Správná (nejsou v popisu chyby?)
- Přesná a jednoznačná
- Konzistentní (nejsou jednotlivé části v konfliktu?)
- Relevantní (jsou navržené funkce potřebné?)
- Proveditelná (je to realizovatelné v rámci rozpočtu a času?)
- Testovatelná

Obvyklé chyby specifikace:

- Obsahuje kvantifikátory vždy, nikdy, každý, žádný, všichni…
- Snaha někam dotlačit – je nabíledni, určitě, evidentně…
- Seznam není úplný – a tak dále, například…
- Vágní specifikace – někdy, něco, obvykle, zpravidla, většinou…
- Nejednoznačné kvantifikátory - dobrý, laciný, malý, stabilní
- If rozhodování bez specifikace else větve

10.2.4 Dynamické testování černé skříňky

a) **Test-to-pass (test splněním)** – kontrolujeme, zda aplikace vyhovuje minimální požadované funkčnosti
b) **Test-to-fail (test selháním)** – snaha najít chybu nebo aplikaci „shodit“

10.2.5 Testování dat (vstupů)

a) Test hraničních podmínek – délky řetězců, zadání čísla větší než je maximum/minimum atd.
b) Test subhraničních podmínek – testování vnitřních omezení (např. položky integer)
c) Test výchozích, prázdných, nevyplněných a nulových hodnot
d) Test na zadání neplatných nebo nesmyslných údajů

10.2.6 Testování stavů (logiky aplikace)

Součástí těchto testů by mělo být také

a) **testování race condition (souběhu).** Příkladem je např. uložení stejného dokumentu z dvou aplikací, současný přístup do databáze, sdílení portů, sdílení tiskárny, stisk klávesy v průběhu zavádění programu…
b) **test opakováním** – co se stane, když se nějaká akce neustále opakuje (např. klikání na tlačítko pro zobrazení formuláře…), hledají se např. memory leaks (zahlcení paměti)
c) **stresové testování** – test běhu aplikace při nejhorších možných podmínkách (málo paměti, málo místa na disku…)
d) **load test (zátěžové testy)** – simulace velkého počtu uživatelů, velkého počtu přístupů, velkého množství transakcí…

10.2.7 Testování konfigurace

Běží aplikace na všech možných kombinacích hardwarových komponent?

Tento typ testu je obtížný na realizace obvykle nemáme většinu HW k dispozici. Lze využít publikovaných HW specifikací…

10.2.8 Testování kompatibility

Dokáže vytvořený software pracovat souběžně s jiným SW?

Backward compatibility – zpětná kompatibilita – pracuje s předchozími verzemi tohoto software? Je datově kompatibilní?

Forward compatibility - dopředná kompatibilita – bude kompatibilní s budoucími verzemi?

10.2.9 Testování multijazyčné podpory

Lokalizace není synonymem pro slovo překlad, ale je to soubor celého komplexu opatření. Zahrnuje kromě vlastní aplikace také lokalizaci dat (např. použití UNICODE), lokalizaci horkých kláves, třídění textů, směr psaní, lokalizace textů v grafice, formátování dat.

Problémy s formátováním dat u multilanguage aplikací:

a) měrné jednotky
b) číselné údaje
c) měna (symbol a jeho umístění)
d) zobrazení datumu
e) zobrazení času (12/24)
f) kalendář (juliánský, gregoriánský, arabský…)
g) adresy (např. tvar PSČ)
h) telefonní čísla
i) velikost papíru a obálek pro tisk

Počítalo se s lokalizací od počátku návrhu?

Zásadní pro lokalizaci je nemít texty uprostřed kódu!

10.2.10 Usability test (Test použitelnosti)

Test použitelnosti - jak snadné je produkt zprovoznit a používat ho (snadnost, efektivita, přesnost, zapamatovatelnost…), formou black boxu.

Je aplikace:

- intuitivní na ovládání?
- konzistentní? (např. co se týče použité terminologie, umístění tlačítek…)
- flexibilní?
- dodržuje standardy? (např. pozice ano – ne tlačítek v dialozích)
- dostupná pro handicapované?
- V souladu s dokumentací?

10.2.11 Unit testy

Unit (jednotka) – samostatně testovatelná část aplikace. V objektově-orientovaném programování odpovídá většinou třídě. Používají se také při regresním testování.

- Provádí vývojář
- Byly použity vhodné algoritmy, návrhové vzory ?
- Ověření správnosti fungování dílčí části kódu
- Stubs, mock object, fakes, …
- Scrum, TDD – unit testy se vytvářejí před kódem

10.2.12 Downgrade test

Je možné se vrátit k předchozí verzi?

10.2.13 Test instalace

Lze aplikaci nainstalovat a poté bez problémů odinstalovat?

10.2.14 Long Term Test

Jak se aplikace chová, je-li trvale spuštěna.

10.2.15 Akceptační test

Testy prováděné při předání aplikace; většinou součástí smlouvy, někdy dohodnuté na počátku projektu. Idea je, že pokud aplikace projde akceptačními testy, měla by být objednatelem převzata a zaplacena.

10.2.16 Performance test

Test rychlostí odpovědi aplikace na činnost uživatele nebo na události.

10.2.17 Security test

Test zabezpečení aplikace. Součástí mohou být také testy implementace integritních omezení.

10.2.18 Recovery test

Vzpamatuje se aplikace po selhání HW? Co se stane, když se vypne počítač v situaci, kdy aplikace je spuštěna?

10.2.19 Regresní testy

Opakované provádění testů na již otestovaných částech. Snaha zjistit, zda oprava chyby nezpůsobila nefunkčnost jiné části aplikace.

10.3 Automatizace testování

- Monitor
- Ovladač (skript nahrazující klávesnicové vstupy)
- Maketa (stub) – přebírá výsledek sw (např. místo tiskárny se tisk ukládá do souboru)
- Záznam maker
- Generátory „šumu“

- Nástroje pro stresové a zátěžové testy (např. Microsoft Stress)
- Analytické nástroje
- Nástroje typu „hloupá opice“ (náhodné klikání nebo vstupy)
- Nástroje typu „polointeligentní opice“ – rozpozná nalezení chyby
- Nástroje typu „inteligentní opice“ – nástroj ví kde je a co dělá

10.3.1 Alfa testy

Interní testy ve vývojovém prostředí, které ale provádí někdo jiný než vývojový tým.

10.3.2 Beta testy

Zapojení externích osob do testování, např. potenciální uživatelé. Není zde jistota, že beta testeři otestují vše. Výhodou je ale např. ověření kompatibility s HW kombinacemi.

10.3.3 Strategie testování

Kolik osob, časový plán testů, stanovení metrik…

Shrnutí hlavních bodů kapitoly:

- Black Box test – nemáme k dispozici zdrojový kód
- White Box test – máme zdrojový kód, lze využít debugger
- Testování okrajových podmínek
- Zátěžové testy
- Testy instalace a odinstalace
- Stress testy a long term test
- Vícejazyčná podpora
- Alfa a beta testy
- Automatizace testování

Kontrolní otázky:

1) Proč je nutné věnovat vysokou pozornost testování software?
2) Jaký je rozdíl mezi Black Box a White Box testy?
3) Co je to dynamické testování?
4) Co se kontroluje při testech specifikace?
5) Jaké znáte základní typy testů?
6) Co je to alfa a beta testování?

Použitá literatura:

1) PATTON, R. Testování softwaru. Vyd. 1. Praha: Computer Press, 2002, xiv, 313 s. Programování. ISBN 80-722-6636-5.
2) Testování software – portál [online]. Dostupné na www: <http://testovanisoftwaru.cz/>
3) SOMMERVILLE, I.: Softwarové inženýrství. Brno: ComputerPress, 2013. ISBN: 978-80-251-3826-7

11 Metodiky Řízení IT/ICT: ITIL, COBIT, IT Governance

V této kapitole se seznámíte s řízením podnikové informatiky. Dozvíte se, jaké jsou doporučení metodik ITIL a COBIT, založené na zkušenostech z praxe.

Stručný obsah kapitoly:

- Základní pojmy z řízení podnikové informatiky
- Úvod do metodiky ITIL
- ITIL verze 2
- ITIL verze 3
- Metodika COBIT
- IT Governance

Tato kapitola předpokládá znalosti základních pojmů z informačních systémů a základů procesního řízení.

Získáte:

- Znalosti problematice řízení podnikové informatiky
- Přehled o metodikách ITIL a COBIT

Budete umět:

- Vysvětlit, čím se zabývají metodiky ITIL a COBIT
- Objasnit, jaký je vztah metodik pro řízení IT/ICT a norem ISO 9000
- Znát, co to je IT Governance

Budete schopni:

- Analyzovat stav podnikového IT/ICT a s využitím znalostí z metodik ITIL a COBIT navrhnout doporučení na optimalizaci

Specifikace času potřebného pro nastudování kapitoly: 60 minut.

11.1 Řízení podnikové informatiky

Jednou z klíčových úloh systémové integrace je **efektivní řízení fungování IT v podniku**. V konečném důsledku se jedná o poměrně složitý proces, do kterého vstupuje řada prvků – dostupné finance na pořízení, rozvoj a údržbu, personální zdroje, definování cílů, jaké má podniková informatika přinést, řízení údržby SW, zavádění nových verzí, řízení vývoje apod.

Aby výše uvedené činnosti byly efektivní a aspoň nějakým způsobem řešitelné, vznikly v průběhu minulých let metodiky a normy, které se snaží daný problém řešit. V následující části tohoto textu se seznámíme s nejčastěji používanými metodikami, a to je ITIL a COBIT a také s normami ISO, které souvisejí s řízením IT/ICT.

11.2 Základní pojmy související s ITIL

Pokud mluvíme o ITIL, znamená to, že mluvíme o **řízení IT služeb**. Abychom se mohli podrobněji zabývat o čem vlastně ITIL je, na začátku si ujasněme některé základní pojmy.

Co rozumíme pod pojmem služba? **Služba** je schopnost uspokojit předem stanovené nebo dohodnuté požadavky či potřeby.

V souvislosti s IT službami se můžete setkat se zkratkou **ICT**; ICT znamená „*Information and Communication Technology*“. ICT službou se rozumí služba, kterou IT oddělení poskytuje uživatelům vně (mimo) IT.

Je třeba zdůraznit, že **ICT služby** jsou **podpůrné služby** pro fungování procesů v organizacích (samotné IT není pro organizaci cílem; IT oddělení není zřizováno za účelem nějaké vědy nebo výzkumu, ale za účelem zajištění lepšího fungování společnosti).

Fungování ICT služeb umožňuje **ICT infrastruktura**. Jedná se o řízení jednotlivých prvků ICT (instalace, konfigurace, řízení sítě…).

Poznámka – pojmem Infrastruktura rozumíme sadu vzájemně propojených částí, které poskytují rámec pro podporu celého systému.

S fungováním ICT služeb souvisí další termín a to **ITSM** (zkratka *Information Technology Service Management*). ITSM znamená **řízení ICT služeb**, a to spíše v rovině organizačně řídicí než v oblasti technické.

CIO (Chief Information Officer) – ředitel IT útvaru

CEO (Chief Executive Officer) – výkonný ředitel

ITIL vnímá podnikovou informatiku jako proces. **Proces** je definován jako opakující se činnost, která má vstup a výstup, vlastníka a zákazníka (někoho, kdo využívá výstupy procesu). Zákazníkem v případě IT/ICT jsou zaměstnanci podniku, kterému IT oddělení poskytuje své služby.

11.3 Co je to ITIL?

A teď už se dostáváme k samotnému pojmu ITIL.

ITIL (***IT Infrastructure Library***) je **sada knižních publikací, popisujících způsob řízení ICT služeb (ITSM) a ICT infrastruktury**. K řízení IT služeb používá procesně orientovaný přístup – každá aktivita v procesu musí přinášet přidanou hodnotu pro uživatele služby. Klíčová orientace ITILu je stabilita, kvalita a spolehlivost IT služeb.

Koncepce ITIL vznikla v Anglii někdy koncem 80. let, jako vládní zakázka pro vytvoření metodiky jak řídit IT služby. Výsledkem je jakási norma, založená na **best practices** (nejlepších zkušenostech z praxe) v daném oboru. ITIL vznikl jako důsledek nízké efektivity při vytváření a nasazování IS/IT ve státní sféře (=příliš vysoké náklady na pořizování IS/IT ve srovnání s privátní sférou). Na základě zkušeností z privátní sféry byly formulovány doporučení, platné obecně.

ITIL není metodika, ale rámec pro návrh procesů ITSM založený na nejlepších zkušenostech z praxe. ITIL nenabízí konkrétní ani univerzální řešení. Je to dáno právě tím založením na „best practices" pocházejících z velice různorodého prostředí. Lidem často vadí, že ITIL je nekonkrétní a neříká přesně co se má v které situaci dělat. Každá organizace je ale svým způsobem unikátní – svým prostředím, zvyky, firemní kulturou a proto je nutné řešení přizpůsobit konkrétním potřebám.

Kvalita implementace ITIL principů je závislá na úrovni pochopení prostředí organizace a na schopnosti navrhnout a vyjednat optimální řešení.

Hlavní ideou implementace ITIL je snaha změnit IT oddělení ze servisní organizace na kvalitního poskytovatele služeb zákazníkům (byť interním). To je poměrně radikální požadavek vyžadující změnu fungování oddělení a myšlení lidí. IT oddělení často chápe svoji roli jako správa a údržba technologie, pro vedení organizace je ale IT jen prostředek, byť mnohdy nezbytný, pro zajištění fungování procesů v organizaci.

ITIL, jak už jsem uvedl, vznikl v Anglii na zakázku vlády, ale postupně se na přelomu století stal jakýmsi neoficiálním standardem pro ITSM (řízení IT služeb) a nyní už je prakticky samostatným oborem.

ITIL je komerční produkt, jeho zakoupení zahrnuje:

a) Knihovny ITIL
b) Související publikace (ISO 20000, Cobit)
c) Konzultační služby
d) Vzdělávání a školení uživatelů
e) Implementace nástrojů pro podporu ITSM

Top přínosů implementace ITIL dle poradenských firem:

- Úspora nákladů na provoz IT služeb
- Lepší kvalita a spolehlivost IT služeb (spokojenější zákazníci)

- Lepší využívání ICT zdrojů
- Menší počet výpadků ICT systémů
- Lepší úroveň komunikace mezi IT a zákazníky/uživateli

Cílem ITIL je zajištění dostupnosti, spolehlivosti a bezpečnosti IS/IT.

ITIL neřeší konkrétní podobu organizační struktury ani podobu a obsah pracovních postupů. Tyto záležitosti jsou věcí implementačního projektu. Neexistují dvě organizace, které by měly procesy ITSM dle ITIL naimplementovány naprosto stejně.

Než se podíváme podrobněji na jednotlivé části rámce ITILu, musíme definovat některé další pojmy, a to: proces, problém, incident, funkce a role.

Proces je logický sled úkolů transformujících vstup na nějaký výstup. Každý proces má definovaný cíl, kvůli kterému existuje.

Incident je nestandardní událost, která může způsobit přerušení fungování služby nebo snížení její kvality.

Problém je neznámá příčina incidentů v IT infrastruktuře.

Funkce – část, specializovaná na vykonání určitého typu práce a produkující specifický výstup

Role – množina aktivit a odpovědností, přiřazených členu týmu. Je definována v rámci procesu.

ITIL říká **KTERÉ** procesy mají být implementovány, ale neříká **JAK**.

11.4 ITIL verze 2

Druhá verze knihovny ITIL je rozdělena do několika částí, zaměřených na specifickou oblast řízení IT služeb. Poskytuje návod, jak řídit IT služby a jak poskytování IT služeb zlepšovat. Zásadní jsou knihy **Service Support** a **Service Delivery.**

Procesy popsané v Service Delivery odpovídají **taktickému řízení**, procesy popsané v knize Service Support odpovídají **operativní úrovni řízení**.

Součástí implementace ITIL verze 2 podle knihy **SERVICE SUPPORT** je:

Incident Management – popisuje proces, zajišťující co nejrychlejší obnovení dodávky služby po vzniku incidentu, minimalizuje důsledky výpadků služeb na chod organizace. Cílem je odstranit incident v co nejkratším čase.

Problem Management – proces zjišťování původních příčin incidentů.

Service Desk – účelem této funkce je poskytnout uživateli jediné kontaktní místo pro adresování požadavků.

Configuration Management – proces řízení konfigurací (součástí je udržování verzí konfiguračních prvků, poskytuje logický model infrastruktury nebo služby)

Change Management – proces řízení změn. Cílem je minimalizovat dopad incidentů vzniklých z důvodu realizace změny

Release Management – proces zajišťující distribuci a nasazení změny do IT infrastruktury. Zajišťuje soulad technického i organizačního aspektu nasazení.

Kniha **SERVICE DELIVERY** definuje:

Service Level Management - proces řízení úrovně kvality služeb (definování, schvalování, dokumentování a řízení úrovně IT služeb; servisní podpora, smlouvy se subdodavateli)

Availability Management – zodpovídá za dosažení takové úrovně dostupnosti IT služeb, která odpovídá požadavkům organizace (prostřednictvím monitorování dostupnosti IT služeb a porovnáváním těchto hodnot s požadavky na dostupnost).

Capacity Management – zodpovídá za zajištění trvale dostatečné kapacity infrastruktury tak, aby byly uspokojeny všechny požadavky organizace (současné i budoucí).

Financial Managament – zodpovídá za evidenci nákladů na IT služby, vyhodnocování návratnosti investic do IT služeb, za náklady na znovu-obnovení provozu. Poskytuje podklady pro sestavování IT rozpočtů.

IT Service Continuity Management – proces řízení schopnosti poskytování definované úrovně služeb při výpadku systémů (od selhání dílčí aplikace po kompletní ztrátu předpokladů k firemní činnosti).

V roce 2005 byla vydána norma ISO 20000 pro řízení IT služeb a tato norma vychází s procesního rámce dle ITIL.

11.5 ITIL verze 3

Od roku 2007 je již k dispozici **třetí verze publikací ITIL**. Tato verze přinesla výraznou změnu koncepce – **středem zájmu se stává IT služba a její životní cyklus**.

Verze 3 knihovny ITIL je postavená na pěti klíčových publikacích:

- Service Strategy
- Service Design
- Service Transition
- Servise Operation
- Continual Service Improvment

Service Strategy popisuje soulad mezi IT a strategickými cíli organizace, tak aby v každém stavu životního cyklu služby byla zachována orientace na hodnotu pro organizaci. (V rámci

této kapitoly jsou vysvětleny procesy jako Financial Management, Service Portfolio Management a Demand Management).

Service Design pokrývá návrh IT služeb a dává návody pro tvorbu a údržbu strategií, návrhu metrik, pracovních postupů, politik a architektury IT (do této kapitoly patří Service Level Management, Capacity Management, Availability Management)

Service Transition obsahuje postup, jak požadavky definované v rámci Service Strategy efektivně realizovat v reálném prostředí, za současného řízení rizik poruch a výpadků služeb. (Kombinuje postupy Release Managementu nebo Risk Managementu)

Service Operation – publikace obsahuje postupy pro řízení služeb v produkčním prostředí; dosažení výkonnosti a účinnosti v dodávce služeb. Zde dochází k samotnému doručení hodnoty a prakticky se ověřuje to, co bylo navrženo v designové fázi. Tato část odpovídá knihám Service Strategy a Service Delivery v ITIL v. 2, zahrnuje ale také Application Management a ICT Infrastrukture Management.

Continual Service Improvment obsahuje prostředky pro vytváření a udržování přidané hodnoty služby prostřednictvím zvyšující se kvality služeb a efektivity jejich provozu.

11.6 Implementace ITIL

Špatná komunikace mezi IT oddělením a ostatními odděleními organizace může znamenat obrovský problém. Požadavky na IT mohou být nekoordinované, neřízené, IT oddělení funguje pouze na operativní úrovni – vše se dělá hned, vše má stejně vysokou prioritu, chybí plánování…). Důsledkem může být nízká účinnost využívání IT zdrojů, nízká schopnost prezentace přínosů IT pro organizaci, nízká návratnost investic. Nebezpečným vedlejším projevem pak může být i značná závislost IT oddělení na jednotlivcích což může znamenat hrozbu pro dlouhodobou udržitelnost IT služeb.

ITIL obsahuje popisy možných způsobů a metod, jak služby (poskytované IT oddělením) evidovat a třídit. Základní podmínkou efektivního řízení IT je přijetí konceptu dodávky služeb. Knihovna ITIL obsahuje popis a doporučení pro nasazení procesů, funkcí, postupů a aktivit pro zajištění chodu služeb.

Aby bylo možné garantovat požadovanou úroveň služeb, je nutné vytvořit taková organizační a technologická opatření, která umožní takto definovanou úroveň realizovat. ITIL zároveň pracuje s měřením efektivity a výkonu IT oddělení. Organizace tím získá přehled o úrovni a kvalitě poskytovaných služeb. Aby IT oddělení mohlo kvalitně fungovat, musí pracovat s kvalitními a řízenými informacemi.

Výši nákladů do IT lze ovlivnit:

- efektivnější organizací práce IT oddělení
- zvýšení dostupnosti služeb
- implementace principů finančního řízení do IT

Jedním z výsledků implementace ITIL by měla být schopnost prezentovat náklady na určitou úroveň poskytované služby.

Základní projektové etapy jsou čtyři

a) Získání znalostí o ITIL (týká se manažerů a klíčových zaměstnanců organizace a členů implementačního týmu)
b) Zhodnocení současné situace (popis současného stavu, identifikace oblastí, které jsou již pokryty)
c) Naplánování a dosažení cílového stavu (projektový plán, realizace implementace)
d) Ověření, zda bylo dosaženo cíle

Zásady pro implementaci ITIL bychom mohli formulovat takto:

1. Rozhodnutí o implementaci ITIL musí být učiněno na úrovni nejvyššího vedení organizace
2. Projekt implementace musí mít viditelnou podporu ze strany nejvyššího vedení
3. Je vhodné v průběhu projektu nastavit očekávání všech zainteresovaných stran (příliš vysoká očekávání – zklamání i v případě relativně úspěšného průběhu, příliš nízká očekávání – výstupy se nedaří uvést do života, zúčastnění jsou výsledkem zaskočeni)
4. Předpokládá se dosažení vzájemné rovnováhy mezi třemi nezbytnými pilíři pro řízení IT: **lidí – procesů – nástrojů** (zaměstnanci musí být zaškoleni, procesy musí být zdokumentovány a implementovány, procesy ITSM musí být podporovány vhodnými SW nástroji). Nevyváženost těchto tří částí vede k neúspěchu, i kdyby byly procesy (ITIL) velmi dobře zvládnuty a navrženy.

Slabinou ITIL je absence aparátu pro měření a vyhodnocování účinnosti procesů a také obtížná vyčíslitelnost přínosů zavedeni ITIL.

11.7 Vztah mezi ISO a ITIL

Lidé se někdy ptají, jaká je souvislost mezi ITIL a ISO normami. S implementací ITIL souvisí normy ISO 9000, ISO 9001 a především ISO 20000.

ISO 9000 definuje procesní přístup k řízení organizace.

ISO 9001 vyžaduje od organizace, aby využívala při řízení své činnosti procesního řízení. Základním předpokladem systému řízení kvality je popis a jednoznačná definice procesů – stanovení co má být vykonáno, jak, kdy a kým. Cílem je zajistit, aby se pracovalo hned napoprvé dle správného procesu a aby bylo realizováno průběžné zlepšování s cílem zajistit zvyšování produktivity (menší požadavky na zdroje). ISO 9001 také zavádí „řízení neshody“, „opatření k nápravě“, a „preventivní opatření“. Analogické prvky se objevují také v metodickém rámci ITIL a následně v normě ISO 20000.

Nejvíce s ITIL souvisí ISO 20000, což je standard ISO pro řízení ITSM. ITIL je standard pro řízení IT služeb. ISO 20000 vznikla z britské normy BS15000 (vydané roku 2000). Je stručnější než ITIL, nespecifikuje „nejlepší praktiky“, ale definuje kritéria, k nimž by měla iniciativa zdokonalování IT procesů směrovat. ISO 20000 byla publikována koncem roku 2005. Nad rámec ITIL se zde nacházejí požadavky navazující na ISO 9001 a problematika řízení bezpečnosti informací.

Tzn. ISO 20000 definuje „co bychom měli dělat“, ITIL stanoví „jak bychom to měli dělat“ a pod tím je řešení „jak to udělat v konkrétní organizaci“. ISO 20000 může být organizaci udělena, aniž je implementován jeden z klíčových ITIL konceptů známý jako „Known Error“.

Na závěr si musíme zdůraznit, že ISO není automaticky recept pro získání kvality. Normy ISO dávají organizacím návody, jak dosahovat vysoké kvality, ale není zaručeno, že tyto návody budou využívány.

11.8 COBIT

COBIT (*Control Objectives for Information and related Technology*) byl vyvinut jako všeobecně přijímaný **standard pro správné postupy řízení, kontroly a auditu informačních technologií**.

ITIL je o řízení ICT z pozice CIO, COBIT je metodika pro ICT Governance, vzájemně se doplňují. Zatímco ITIL je zaměřen více na operativní a taktické řízení ICT, COBIT je posunut do strategického řízení. První verze COBIT byla publikována v roce 1996. Druhá verze, rozšířena o “Management Guidelines” v roce 1998. Třetí verze byla zveřejněna v roce 2000, aktuální čtvrtá verze pochází z prosince roku 2005. COBIT je výsledkem zkušeností auditorských společností.

COBIT je oproti ITIL volně ke stažení.

COBIT dává do souvislosti:

- IT procesy
- IT zdroje
- Informační kritéria

Základní IT procesy jsou**:**

- Plánování a organizace
- Akvizice a implementace
- Poskytování a podpora
- Monitorování

IT zdroje dle Cobit:

- Aplikace
- Informace
- Infrastruktura
- Lidské zdroje

Informační kritéria dle COBIT jsou:

- Účelnost
- Hospodárnost
- Důvěryhodnost
- Integrita
- Dostupnost
- Souhlasnost/Shoda
- Spolehlivost

COBIT pro jednotlivé **oblasti** definuje:

- Cíle řízení
  - definice požadavků ze strany businessu
  - strategický cíl ke každému procesu
  - detailní cíle jednotlivých procesů
- Procesy- definuje 34 procesů seskupených do 4 následujících domén:
  - Plánování a organizace
  - Akvizice a implementace
  - Poskytování a podpora
  - Monitorování
- Zdroje přiřazené k procesům

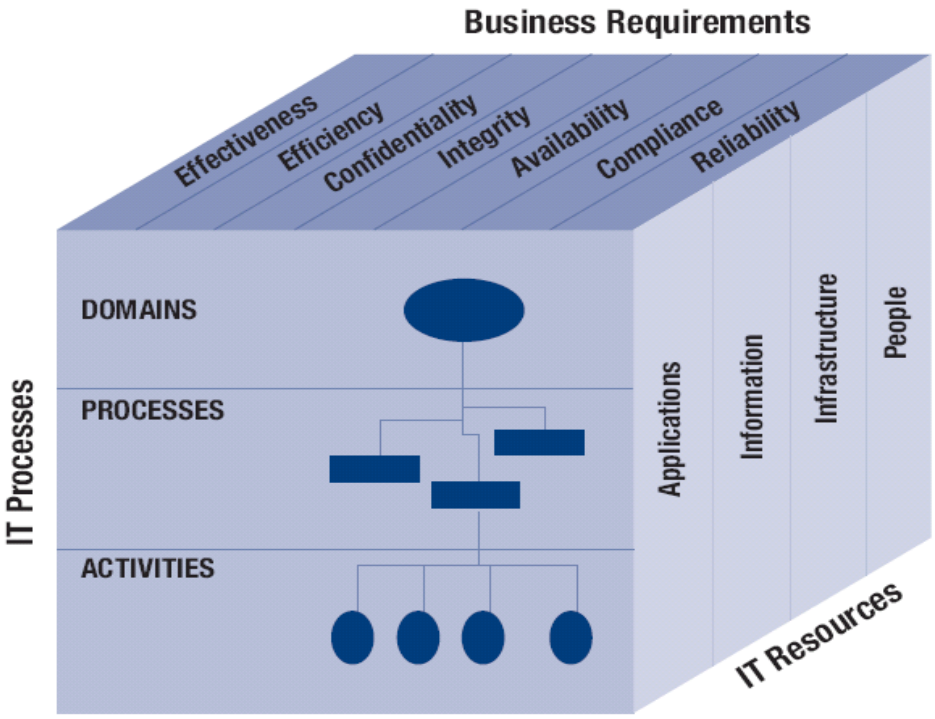
Pro každý **proces** jsou v COBITu definovány:

- Obsah a cíl

- Dílčí kontrolní cíle
- Typické aktivity a role
- Vstupy a výstupy
- Kritéria pro model vyspělosti
- Způsob měření
- Způsob auditu

Zdroje přiřazené k procesům:

- Informace (datové objekty – interní, externí atp.)
- Aplikační systémy (souhrn manuálních i automatizovaných procedur)
- Infrastruktura (HW, operační systémy, sítě, lokalizace a podpora informačních systémů)
- Lidé (znalosti, organizace, získávání, poskytování, podpora, monitoring a ohodnocení informačních systémů a služeb)



Obrázek 13 Struktura COBIT

COBIT by měl zajímat všechny, kdo mají přímou odpovědnost za obchodní procesy a technologie, ty, kdo jsou závislí na relevanci a spolehlivosti informací zapracovávaných prostřednictvím ICT a také ty, kdo poskytují služby v oblasti řízení kvality, kontroly a správy IT.

SPECIFIKA COBITU:

- Orientace na business – určeno pro poskytovatele IT služeb, uživatele, auditory, management a vlastníky podnikových procesů
- Procesní orientace - COBIT obsahuje referenční procesní model jednotlivých domén, měření výkonnosti procesů, hodnocení rizika
- Kontrolní rámec - pravidla, procedury, postupy, organizační struktura jsou definovány tak, aby poskytovaly záruku toho, že budou dosaženy podnikové cíle a že bude minimalizována pravděpodobnost jejich ohrožení
- Zaměřeno na měření výkonnosti (cíl, proces, metrika) - slouží pro podporu rozhodování o způsobu měření a kontrolování podnikového IT Základní oblasti (tzv. domény) COBIT.

11.9 IT GOVERNANCE

IT Governance (=řízení IT) je definovaná struktura vztahů a procesů, pomocí kterých lze řídit a kontrolovat organizaci tak, aby IT v maximální míře umožňovalo a podporovalo dosažení podnikatelských cílů. Přidanou hodnotou je redukce a řízení rizik nad procesy v IS.

Od ITILu a COBITu se liší tím, že **hlavním cílem je návratnost investic**.

IT Governance ovlivňuje zvýšení efektivity organizace pomocí:

- Zajištění a zabezpečení integrity, bezpečnosti a spolehlivosti strategických a jiných citlivých informací.
- Ochrany investic do IT a komunikací
- Nastavení odpovídajícího vedení a řízení informačních aktiv, na nichž přímo závisí úspěch nebo přežití organizace
- Zvyšování hodnoty podnikatelských procesů pomocí IT (vazba IT na podnikatelské procesy)

Důvody pro zavedení IT Governance:

- IT je významným kritickým faktorem úspěchu pro dosažení podnikové strategie
- IT je prvek, který umožňuje růst a vývoj podniku
- IT musí splňovat stále rostoucí nároky v oblasti regulací, povinné správy a ochrany IT aktiv

Shrnutí hlavních bodů kapitoly:

- Pohled na IT/ICT jako na službu poskytovanou IT oddělením „zákazníkům“
- Podniková informatika má procesní charakter
- ITIL – doporučení na řízení IT/ICT na základě nejlepších praktik
- Verze 2 a Verze 3 (větší důraz na procesní orientaci, 5 publikací)
- ISO 9000 – řízení kvality
- ISO 20000 – řízení kvality v IT
- COBIT
- IT Governance – řízení IT/ICT s důrazem na návratnost investic

Kontrolní otázky:

1) Co je cílem řízení IT/ICT?
2) Kdo je to CIO?
3) Jaké jsou základní principy metodiky ITIL?
4) Proč je důležité mít helpdesk?
5) Čím se zabývá problem management a incident management?
6) Čím se zabývá change management a release management?
7) Jaké jsou základní principy metodiky COBIT?
8) Co je to IT Governance?

Použitá literatura:

1) BUCKSTEEG M. & col. ITIL 2011, Brno: Computer Press, 2011. ISBN 978-80-251-3732-1
2) KOSZLAJDA, A. Zarządzanie projektami IT: przewodnik po metodykach. Gliwice: Helion, 2010. ISBN 978-832-4618-040
3) IT & Management Knowledge Base – portal. [online] Dostupné na www: <http://www.bestpractice.cz/>
4) Oficiální stránka ITIL [online]. Dostupné na www: < http://www.itil-officialsite.com/>
5) ITSM Portal [online]. Dostupné na www: < http://www.itsmportal.com/>
6) ISACA Portal [online]. Dostupné na www: < http://www.isaca.org>

12 Softwarové právo, autorský zákon, licence

V této kapitole se seznámíte se základy softwarového práva. V kapitole je popsán vztah mezi software a autorskými právy podle autorského zákona. Dozvíte se, jaké jsou základní typy licencí a jaký je rozdíl mezi licencí výhradní a nevýhradní

Stručný obsah kapitoly:

- Autorská práva na software
- Výhradní a nevýhradní licence
- Typy licencí
- Vlastnictví dat

Tato kapitola nevyžaduje žádné vstupní znalosti.

Získáte:

- Přehled o autorských právech ve vztahu k software
- Znalost základních typů softwarových licencí
- Představu o vlastnických právech na databáze

Budete umět:

- Vysvětlit, jak je software patentovatelný
- Popsat rozdíl mezi výhradní a nevýhradní licencí
- Vysvětlit vlastnické práva k pořizovaným datům

Specifikace času potřebného pro nastudování kapitoly: 30 minut.

12.1 Softwarové právo

Softwarové právo se zabývá těmito oblastmi:

a) Software a jeho vývoj – vlastnická práva na software a zdrojové kódy
b) Vlastnictví dat
c) Licencování software
d) Právní aspekty implementace a dodání software
e) Právní aspekty servisu a údržby
f) Outsourcing
g) Softwarové pirátství
h) Internetové právo
    a. Registrace doménových jmen
    b. Odpovědnost za provoz internetových stránek
    c. Internetové obchodování
    d. Autorská práva na Internetu
    e. E-government

Software je považován za **autorské dílo** a vlastnictví softwaru tedy řeší **Autorský zákon**.

Dle poslední novelizace Autorského zákoníku je vlastníkem software vždy jeho autor. Autorská práva nelze prodat, nelze se jich vzdát a jsou nepřevoditelná, jsou předmětem dědického řízení. Lze však postoupit **výkon autorských práv.** Podle znění zmíněného zákona je výkon autorských práv v případě software vytvořeného **vlastními zaměstnanci** a software vytvořeného **na smlouvu o dílo** přenesen na zaměstnavatele (zadavatele smlouvy o dílo).

Podmínkou pro automatický přenos výkonu autorských práv ze zaměstnance na zaměstnavatele je korektně sepsaná pracovní smlouva – z ní musí jednoznačně vyplývat, že tvorba software je pracovní náplní zaměstnance dle pracovní smlouvy.

Autorská práva můžeme rozdělit na osobnostní (právo na zveřejnění, právo nedotknutelnosti atd.) a majetková. Majetkové autorské právo platí v Česku 70 let a jsou předmětem dědického řízení. V případě zániku zaměstnavatele přechází výkon autorských práv na jeho právního nástupce, pokud tento neexistuje, vrací se výkon autorských práv k autorovi (zaměstnanci).

Software a databáze jsou v rámci Autorského zákona vyjmuty z „volného“ užití – nelze pořizovat kopie pro osobní potřebu. Na druhé straně, nabyvatel (uživatel) software dle par. 66 AZ má právo pořizovat kopie za účelem:

a) Zajištění běžného provozu
b) Zajištění interoperability s dalším software

Výkon autorských práv dává zaměstnavateli tyto práva:

a) Zveřejňovat, upravovat a šířit software pod svým jménem
b) Právo na dokončení software (i v případě, že skončí pracovněprávní vztah s autorem)

12.2 Právo zaměstnavatele k zaměstnaneckému dílu

a) Právo výkonu majetkových práv
b) Se souhlasem právo postoupit oprávnění výkonu majetkových práv třetí osobě
c) Právo ke zveřejnění a úpravám software
d) Právo odmítnout udělení licence zaměstnanci
e) Právo dokončení nehotového SW
f) Právo soudní obrany, dojde-li k zásahu do autorských práv
g) Zaměstnanec nemá právo na další přiměřenou odměnu

12.3 Pracovní smlouva

Pracovní smlouva by měla obsahovat tyto náležitosti:

a) Vymezení druhu práce (ne moc všeobecně ani příliš specificky)
b) Definování zodpovědnosti za práci
c) Nastavení pravidel přístupu k informacím
d) Sankci za nekvalitní práci
e) Specifikovat možnosti kontroly zaměstance zaměstnavatelem
f) Benefity a motivační mechanismy
g) Rizikové faktory eventuálně zákaz konkurence

By-li vytvořen software mim rámec vymezení druhu vykonávané práce, nemůže být software zaměstnavatelem využit bez uzavření licenční smlouvy.

12.4 Patentování SW

Co se patentovatelnosti software týče, platí ze zákona, že „**kdokoli smí vytvořit software shodné funkcionality, pokud to není protiprávní**“. Za protiprávní se považuje např. využití zdrojového kódu získaného reverzním inženýrstvím.

Předmětem patentu **nemůže být vizuální podoba software**. Výjimkou je situace, kdy dojde **k porušení zákazu nekalé soutěže**. Do této kategorie patří situace, kdy software napodobuje vzhled úspěšného software a parazituje na pověsti a úspěchu cizího SW.

Ochrana software patenty byla Evropským patentovým úřadem zamítnuta; počítačový program není považován za vynález. Výjimkou je stav, kdy software má prokazatelný technický efekt.

12.5 Licence

Licencí autor uděluje oprávnění užívat software (=realizace autorského majetkového práva).

Licence může být **výhradní** nebo **nevýhradní**. Výhradní licence je udělena výhradně jednomu nabyvateli (autor nemůže licenci poskytovat dalším subjektům). Výhradní licence může být omezena lokálně (např. na území nějakého státu) nebo časově.

Dle obsahu licence dělíme proprietární, free nebo public domain. Public domain licence znamená, že na dílo se nevztahuje žádná autorsko-právní ochrana.

Další typy licencí jsou

- OEM,
- copyleft,
- multilicence,
- transakční licence v cloudu atd.

Obsahové náležitosti licenční smlouvy

a) Specifikace SW
b) Způsob používání SW
c) Rozsah licence (množstevní, časové, územní…)
d) Odměna za licenci (jednorázová nebo formou licenčních poplatků) nebo bezúplatnost
e) Právo na přiměřenou dodatečnou odměnu (v případě, že nabyvatel vytvoří s vyžitím licencovaného sw zisk řádově neúměrný odměně za licenci

Dle Autorského zákoníku lze licenční smlouvu uzavřít i specifickým způsobem:

a) Shrink wrap – otevřením krabice
b) Click wrap – odkliknutím licenčních podmínek po instalaci SW
c) Browse wrap – potvrzením návštěvou odkazu na Internetu

12.6 Vlastnictví dat

Autorský zákon rozlišuje tyto formy databáze:

a) Databáze jako souborné autorské dílo
b) Databáze jako výsledek činnosti pořizovatele (provozovatele SW)
c) Autorská díla s volným použitím (např. texty předpisů)

Pořizovatel databáze může být právnická nebo fyzická osoba, která si databázi nechá vytvořit na smlouvu o dílo (na objednávku).

Pořizovatel disponuje „zvláštními právy pořizovatele“:

a) Právo na vytěžování (rozmnožování)
b) Právo na zužitkování (zveřejnění)

Tyto práva platí 15 let od pořízení nebo prvního zpřístupnění. V případě podstatné změny databáze naskakuje nová patnáctiletá lhůta.

Vytěžování databáze oprávněným uživatelem je zákonem omezeno: „data se nesmí kopírovat za účelem komerčního využití nebo systematicky a opakovaně“. Tím je myšleno např. vytěžování databází, které jsou součástí dostupných internetových aplikací.

Shrnutí hlavních bodů kapitoly:

- Autorský zákon
- Vlastnictví autorských práv
- Výhradní a nevýhradní licence
- Typy licencí
- Vlastnické práva k pořizování databází

Kontrolní otázky:

1) Kdo je majitelem autorský práv?
2) Jsou autorská práva přenositelná?
3) Komu patří software a autorská práva na ně, pokud byl software vytvořen jako zaměstnanecké dílo?
4) Je software patentovatelný?
5) Co znamená nevýhradní licence?
6) Jaké znáte typy licencí?
7) Jak autorský zákon upravuje vlastnictví dat?

Použitá literatura:

1) JANSA, L., OTEVŘEL P. Softwarové právo: praktický průvodce právní problematikou v IT. Vyd. 1. Brno: Computer Press, 2011, 340 s. ISBN 978-802-5134-580.
2) ŠTĚDROŇ, B. Ochrana a licencování počítačového programu. Vyd. 1. Praha: Wolters Kluwer Česká republika, 2010, xv, 199 s. Právní monografie (Wolters Kluwer ČR). ISBN 978-80-7357-555-7.
3) IT právo. Portál [online]. Dostupné na www: < http://www.itpravo.cz/>
4) Právo IT. Portál [online]. Dostupné na www: < http://www.pravoit.cz/>

13 Závěr

Dostáváme se na konec e-learningové opory. Pokud jste prostudovali všechny kapitoly a zodpověděli kontrolní otázky, jste připraveni ke zkoušce. Doufám, že Vám tato opora přinesla nové poznatky, které pro Vás budou užitečné nejen při zkoušce, ale i ve Vaší další praxi. Přeji hodně úspěchů při studiu.

Autor

Seznam literatury

1) ARLOW, J., NEUSTADT, I. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. Brno: Computer Press, 2007, 567 s. ISBN 978-80-251-1503-9.
2) BUCKSTEEG M. & col. ITIL 2011, Brno: Computer Press, 2011. ISBN 978-80-251-3732-1
3) DOUCEK, P. Řízení bezpečnosti informací. Praha: ProfessionalPublishing 2011, 240 stran. ISBN: 978-80-7431-050-8
4) FISCHER, U. Risk IT [prezentace]. Rolling Meadows, USA: ISACA, 2009. Dostupné z WWW: <http://www.isaca.org/Knowledge-Center/Standards/Documents/Risk-IT-Overview.ppt>.
5) GAWIN, B., MARCINOWSKI, B.. Symulacja procesów biznesowych: standardy BPMS i BPMN w praktyce. Gliwice: Helion, 2013. ISBN 978-832-4651-412.
6) GUCKENHEIMER, S., PEREZ, J.. Efektivní softwarové projekty: metodiky efektivního vývoje softwaru. Vyd. 1. Překlad Jan Kuklínek, Jan Pokorný. Brno: Zoner Press, 2007, xi, 255 s. ISBN 978-80-86815-62-6.
7) HOFFER, J., GEORGE, J., VALACICH, J. Modern systems analysis and design. 3rd ed. Překlad David Krásenský. Upper Saddle River, N.J.: Prentice Hall, c2002, xxxii, 733 p. ISBN 01-303-3990-3
8) HUBNER, M. Pohled nejen CIO na informační bezpečnost : příručka manažera. Praha: TATE International, 2012. ISBN: 978-80-86813-25-7
9) CHLAPEK, D., ŘEPA, V., STANOVSKÁ, I. Vývoj informačních systémů (pracovní sešit ke cvičením). Praha: VŠE, 2005. ISBN: 80-245-0977-6
10) JANSA, L., OTEVŘEL P. Softwarové právo: praktický průvodce právní problematikou v IT. Vyd. 1. Brno: Computer Press, 2011, 340 s. ISBN 978-802-5134-580.
11) KADLEC, V. Agilní programování: metodiky efektivního vývoje softwaru. 1. vyd. Brno: Computer Press, 2004, 278 s. ISBN 80-251-0342-0.
12) KANISOVÁ, H., MULLER, M. UML srozumitelně. Brno: Computer Press, 2007. 176 s. ISBN: 8025110834.
13) KOMZÁK, T. Řízení IT projektů pro úplné začátečníky. 1. vyd. Brno: Computer Press, 2013, 213 s. ISBN 978-80-251-3791-8.
14) KOSZLAJDA, A. Zarządzanie projektami IT: przewodnik po metodykach. Gliwice: Helion, 2010. ISBN 978-832-4618-040
15) MCCONNELL, S. Odhadování softwarových projektů: jak správně určit rozpočet, termín a zdroje. Vyd. 1. Překlad Jiří Fadrný. Brno: Computer Press, 2006, 317 s. ISBN 80-251-1240-3.
16) PALETA, P. Co programátory ve škole neučí aneb Softwarové inženýrství v reálné praxi. Vyd. 1. Brno: Computer Press, 2003, 337 s. ISBN 80-251-0073-1.
17) SOMMERVILLE, I.: Softwarové inženýrství. Brno: ComputerPress, 2013. ISBN: 978-80-251-3826-7

18) PATTON, R. Testování softwaru. Vyd. 1. Praha: Computer Press, 2002, xiv, 313 s. Programování. ISBN 80-722-6636-5.
19) ŘEPA, V. Analýza a návrh informačních systémů. 1.vyd. Praha: Ekopress, 1999, 403 s. ISBN 80-861-1913-0.
20) ŘEPA, V. Podnikové procesy: procesní řízení a modelování. 2., aktualiz. a rozš. vyd. Praha: Grada, 2007, 281 s. ISBN 978-80-247-2252-8.
21) SCHWALBE, K. Řízení projektů v IT. Vyd. 1. Překlad David Krásenský. Brno: Computer Press, 2007, 720 s. ISBN 978-80-251-1526-8.
22) SOMMERVILLE, I.: Softwarové inženýrství. Brno: ComputerPress, 2013. ISBN: 978-80-251-3826-7
23) STRAKA, M. Vývoj databázových aplikací. Vyd. 1. Praha: Grada, 1992, 162 s. ISBN 80-854-2443-6
24) ŠTĚDROŇ, B. Ochrana a licencování počítačového programu. Vyd. 1. Praha: Wolters Kluwer Česká republika, 2010, xv, 199 s. Právní monografie (Wolters Kluwer ČR). ISBN 978-80-7357-555-7.
25) TIETZE, P. Strukturální analýza: úvod do projektu řízení. Vyd. 1. Praha: Grada, 1993, 228 s. ISBN 80-854-2445-2.
26) TVRDÍKOVÁ, M. Aplikace moderních informačních technologií v řízení firmy: nástroje ke zvyšování kvality informačních systémů. Praha: Grada, 2008. ISBN: 80-247-2728-5.

Autor	Ing. Roman Danel, Ph.D.
Název	Analýza a projektování systémů
Vydavatel	VŠB-TU Ostrava
Rozsah	99 stran
Rok	2014
Copyright	© Roman Danel, 2014
Zdroj financování	Financováno z projektu CZ.1.07/2.2.00/28.0308 Inovace bakalářských a magisterských studijních oborů na Hornicko-geologické fakultě VŠB-TUO, spolufinancovaného Evropským sociálním fondem a státním rozpočtem České republiky

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ