

Vysoká škola ekonomická v Praze  
fakulta informatiky statistiky

Aplikovaná informatika



*Seminární práce*

**RUP – disciplína Analysis & Design**

**Předmět : 4IT421 – Zlepšování procesů budování IS**

**Autor : Petr Soukup**

**E-mail : xsoup22@vse.cz**

**Studijní program : E**

**Rok : 2010/11**

## OBSAH

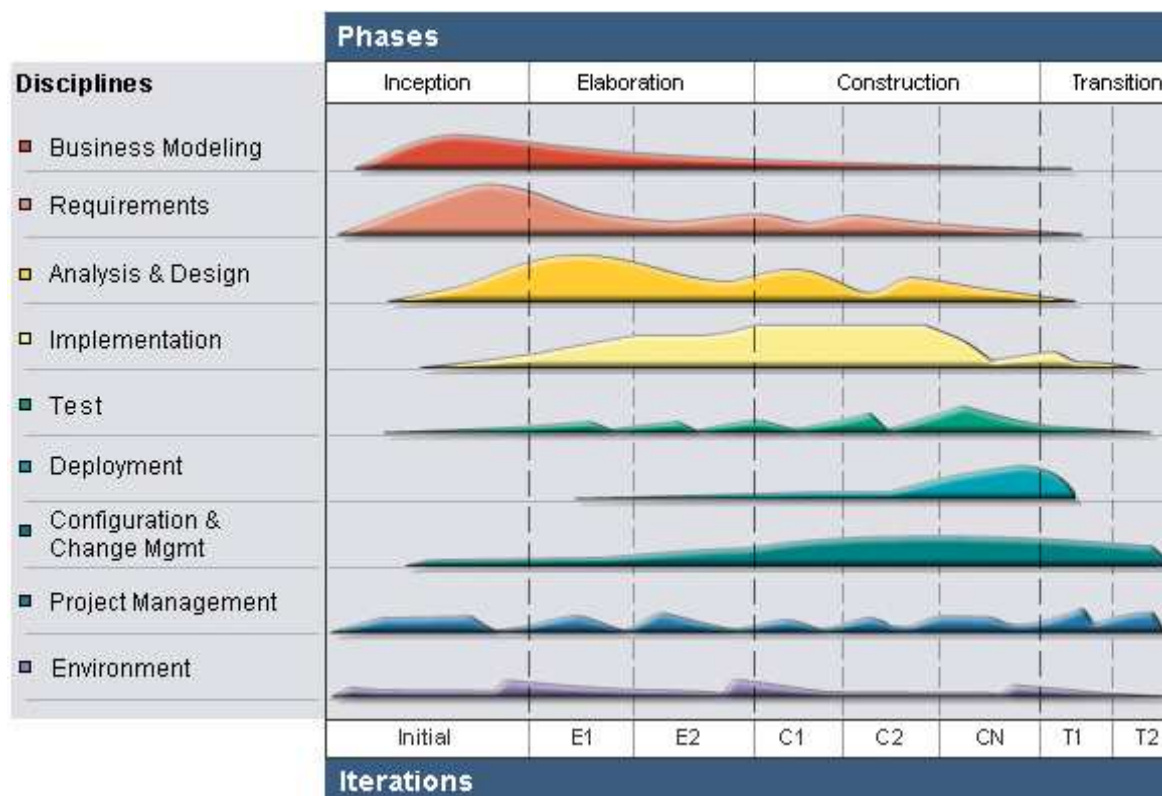
<b>1. ÚVOD .....</b>	<b>- 3 -</b>
1.1. ŽIVOTNÍ CYKLUS RATIONAL UNIFIED PROCESS VE ZKRATCE .....	- 4 -
1.2. CHARAKTERISTIKA (IDENTIFIKACE) DISCIPLÍNY ANALYSIS AND DESIGN .....	- 4 -
1.3. CÍL PRÁCE.....	- 5 -
<b>2. ANALÝZA VS. DESIGN .....</b>	<b>- 6 -</b>
2.1. ANALÝZA .....	- 6 -
2.2. DESIGN .....	- 6 -
<b>3. PRACOVNÍ PROCES ANALÝZA A DESIGN.....</b>	<b>- 7 -</b>
3.1. DEFINE CANDIDATE ARCHITECTURE .....	- 8 -
3.1.1. <i>Analýza Architektury</i> .....	- 9 -
3.1.2. <i>Use Case Analýza</i> .....	- 11 -
3.2. REFINE THE ARCHITECTURE .....	- 12 -
3.2.1. <i>Identify Design Mechanisms</i> .....	- 14 -
3.2.2. <i>Identify Design Elements</i> .....	- 15 -
3.2.3. <i>Describe the Run-time Architecture</i> .....	- 15 -
3.2.4. <i>Describe Distribution</i> .....	- 16 -
3.2.5. <i>Review the Architecture</i> .....	- 17 -
3.3. ANALYZE BEHAVIOR .....	- 18 -
3.3.1. <i>Identify Design Elements</i> .....	- 18 -
3.3.2. <i>Use-Case Analysis</i> .....	- 18 -
3.3.3. <i>Design the User Interface</i> .....	- 18 -
3.3.4. <i>Prototype the User-Interface</i> .....	- 19 -
3.3.5. <i>Review the Design</i> .....	- 19 -
3.4. DESIGN COMPONENTS .....	- 20 -
3.4.1. <i>Use-Case Design</i> .....	- 21 -
3.4.2. <i>Subsystem Design</i> .....	- 22 -
3.4.3. <i>Class Design</i> .....	- 22 -
3.4.4. <i>Review the Design</i> .....	- 23 -
3.5. DESIGN THE DATABASE .....	- 24 -
3.5.1. <i>Class Design</i> .....	- 24 -
3.5.2. <i>Database Design</i> .....	- 24 -
3.5.3. <i>Review the Design</i> .....	- 25 -
<b>4. ROLE.....</b>	<b>- 26 -</b>
<b>5. VÝSTUPY .....</b>	<b>- 27 -</b>
<b>6. ZHODNOCENÍ .....</b>	<b>- 29 -</b>
<b>7. ZDROJE.....</b>	<b>- 30 -</b>

# 1. Úvod

**Rational Unified Process** je metodika určená k vývoji software. Lze ji považovat za rámec, který je rozšiřitelný a přizpůsobitelný potřebám konkrétního projektu.[8] Cílem této metodiky je zajistit vývoj vysoce kvalitního software, který splňuje požadavky koncových uživatelů v časovém harmonogramu a v stanovené hranici rozpočtu. Metodika je založena na obecných praktikách (best practices) softwarového vývoje používaných a prověřených celou řadou odborníků na značném množství projektu. Ve zkratce se jedná o tyto tzv. best practices:[8, 9]

- **Iterativní vývoj softwaru**
- **Správa a řízení požadavků**
- **Použití komponentové architektury**
- **Vizuální modelování softwaru**
- **Průběžné zajišťování a ověřování kvality**
- **Řízení změn**

Rational Unified Process je rozdělen do čtyř hlavních (Inception, Elaboration, Construction, Transition) vývojových fází na ose X a několika (Tvorba podnikového modelu, Správa požadavků, Analýza a Design, Implementace, Test atd.) disciplín na ose Y.



**Obr. 1** – Procesní struktura, dimenze Rational Unified Process[7]

Pro přechod do následující fáze je nezbytné splnit stanovená kritéria tzv. milníky pro právě vyvíjenou oblast.

Je třeba neopomenout hlavní kouzlo této metodiky, jež spočívá v iteracích. Použití metodiky Rational Unified Process umožňuje vytvoření funkčního prototypu systému po

dokončení každé iterace, neboť každá z iterací je svým způsobem malý „vodopád“ obsahující všechny tyto disciplíny.[2] Jinými slovy řečeno v rámci každé iterace proběhnou činnosti vázané na princip modelování, následují specifikace požadavků, analýza a návrh, implementace, testování a rozmístění (instalace). K tomu probíhá celá řada činností z podpůrných toků týkajících se správy konfigurací, řízení projektu a přípravy prostředí, ve kterém je systém vyvíjen a bude nasazen. Způsob, jakým iterace a fáze probíhají, by měl být nakonfigurován podle specifických potřeb projektu a používaných technologií.

V rámci Rational Unified Process jsou k dispozici čtyři základní prvky, které prakticky řídí vývoj softwaru:

- Role/Pracovníci (Role)
- Činnosti (Activities)
- Meziprodukty (Artefacts)
- Pracovní procesy (Workflows)

## **1.1. Životní cyklus Rational Unified Process ve zkratce**

V první fázi Inception (Zahájení) pracovníci definují rozsah projektu a jeho obchodní případ, vizi projektu. Hlavním úkolem je vyjasnění a přesná definice požadavků všech zainteresovaných stran a stanovení rozsahu budoucího systému.

V následující fázi Elaboration (Příprava), pracovníci analyzují podrobněji požadavky projektu ve smyslu návrhu tzv. úvodního modelu architektury, dle kterého v dalších fázích dojde k implementaci vyvíjeného systému.











Ve fázi Construction (Konstrukce) pracovníci vytvoří z modelu návrhu architektury spustitelnou verzi systému, tj. vytvoří zdrojových kód, programy, knihovny, potřebné datové soubory atd. (Vzhledem k typicky velkému rozsahu prací patřících do této fáze je právě tato fáze nejvhodnějším kandidátem k využití iterativnosti. Výsledkem každé z iterací musí být funkční verze systému, přičemž každá další verze je pouze zpřesněním a dotažením předchozí k vyššímu stadiu dokončení). Konečně v poslední fázi Transition (Předání) vývojáři poskytnou systém koncovým uživatelům.[1]

## **1.2. Charakteristika (Identifikace) disciplíny Analysis and Design**

Cílem disciplíny Analýza a Design je transformovat požadavky zákazníka na konkrétní návrh systému, vytvořit robustní architekturu systému a přizpůsobit návrh systému tak, aby odpovídal prostředí pro implementaci.

Hlavní část práce na disciplíně Analýza a Design z největší části probíhá v druhé fázi životního cyklu Rational Unified Process fázi Elaboration (Příprava). Disciplína Analýza a Design jako i ostatní disciplíny se prolínají všemi fázemi, které kombinují definované etapy, techniky a postupy s jinými součástmi vývoje jako jsou dokumenty, modely, manuály, kódy, a další v rámci jednotného rámce životního cyklu Rational Unified Process. Modely vytvořené v tomto pracovním procesu (Analýza a Design) identifikují jak funkční, tak nefunkční požadavky, které zákazník požaduje.

Je důležité zmínit hlavní jazyk UML, jenž je nezbytný v disciplíně Analýza a Design ovládat. Výstupy této disciplíny nejsou ničím jiným než-li modely jazyka UML.

	Inception	Elaboration	Construction	Transition
Perform Architectural Synthesis				
Define a Candidate Architecture				
Analyse Behaviour				
Design Components				
Design the Database				
Refine the Architecture				

**Obr. 2** - Rozložení disciplíny Analýza a Design v životním cyklu Rational Unified Process[5]

### 1.3. Cíl práce

Tato práce se zaměřuje na detailní popis procesu disciplíny Analýza a Design v metodice Rational Unified Process ver.: 7.5.1 ve variantě metodiky pro menší projekty.

## 2. Analýza vs. Design

### 2.1. Analýza

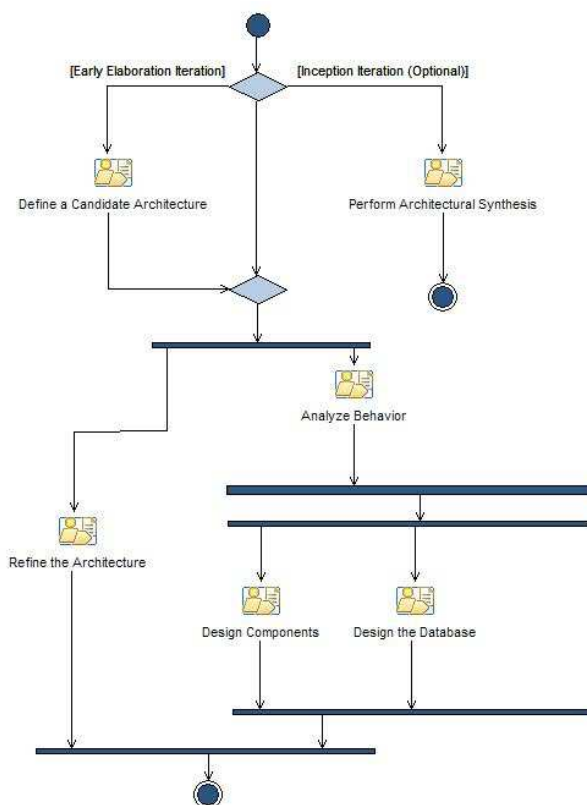
Analýza odpovídá na otázku „**Co**“ má být vytvořeno.[9] Cílem analýzy je analyzovat požadavky na systém do podoby jednotlivých subsystémů, tříd, analyzovat objekty, třídy, subsystémy, události systému, reakce systému atp.. Analýza ignoruje požadavky na funkčnost i omezení, které vyplývají z implementačního prostředí. Výsledkem analýzy je téměř ideální obraz systému.

### 2.2. Design

Design odpovídá na otázku „**Jak**“ systém vytvořit.[9] Cílem designu je přizpůsobit výsledky analýzy omezením vyplývajícím z nefunkčních požadavků analýzy tedy implementačnímu prostředí, požadavkům na výkon systému atp.. Design upřesňuje analýzu a zaměřuje se na optimalizaci a integraci systému a současně zajišťuje úplné pokrytí požadavků.

### 3. Pracovní proces Analýza a Design

V Inception fázi se disciplína Analýza a Design snaží zjistit, zda systém dle představ zákazníka, je vůbec možné posoudit zda-li je řešení možné realizovat z hlediska dostupných technologií (Činnost: **Perform Architectural Synthesis**). Tento krok je možné vynechat, pokud si pracovní tým je jist, že neexistují rizika typu, že budoucí systém je něco neobvyklého, invenčního a že neexistují rizika, která by v pozdějších krocích mohla položit vývoj systému.



Obr. 3 – Pracovní proces disciplíny (Workflow) [7]

Elaboration fáze se zaměřuje na vytvoření „úvodního návrhu architektury“ systému (Činnost: **Define a Candidate Architecture**), jenž je výchozím bodem pro následné zpřesnění funkcionalit, přizpůsobení systému a prioritně pro design systému. V případě, že architektura systému již existuje (ať už proto, že byla vytvořena např.: v předchozích projektech), zaměřuje se činnost paralelně spíše na zpřesnění architektury (Činnost: **Refine the Architecture**) a analýzu chování (Činnost: **Analyze Behavior**).

Následně, co jsou identifikovány počáteční prvky a analyzováno požadované chování systému, vytvoří pracovníci přesnější návrh funkcionality analyzovaných komponent a navrhnu hierarchickou strukturu meziproduktů, modelů pro implementaci (Činnost: **Design Components**). Souběžně s touto činností, je řešen návrh databáze (Činnost: **Design the Database**). Návrhář databáze určí perzistentní třídy a navrhne k nim příslušné datové struktury atp..

### 3.1. Define Candidate Architecture

Cílem této činnosti je vytvoření úvodního návrhu architektury systému. Úvodní návrh architektury můžeme přirovnat ke globálnímu návrhu z hlediska návrhu systému podle dimenzí MMDIS. Jde především o zmapování všech podstatných okolností, jimiž je dán obsah a struktura IS. Je třeba zaměřit se na celek na úkor detailů, jenž budou rozpracovány v dalších iteracích.[13] Pracovníci - Softwarový architekt a Designer definují architektonické vzory, identifikují architektonicky významné případy užití, provedou use case realizaci prvků v budoucí architektuře IS. Neopomenuta je i analýza stávajícího majetku zákazníka. Aktivita Define Candidate Architecture je rozdělena na dvě části. Obsahuje úlohy Analýza architektury a Analýza případů užití.

Hlavním výstupem je **úvodní návrh architektury** vyjádřený převážně v modelu Analysis Model – skládající se z diagramů tříd, sekvenčních diagramů, use case diagramů které popisují logické plnění funkčních požadavků.

Analýza tříd, jejich odpovědností a následné vytvoření class diagramu je jedním z prvních a základních kroků analýzy navrhovaného programového systému.[20, 4] Je nutno říci, že v tomto stádiu by se ale spíše než o komplexní návrh systému mělo jednat o nástin řešení, s větší mírou abstrakce analyzovaného systému. Jakýsi idealistický obraz IS, na jehož základě bude možné posoudit proveditelnost designu.

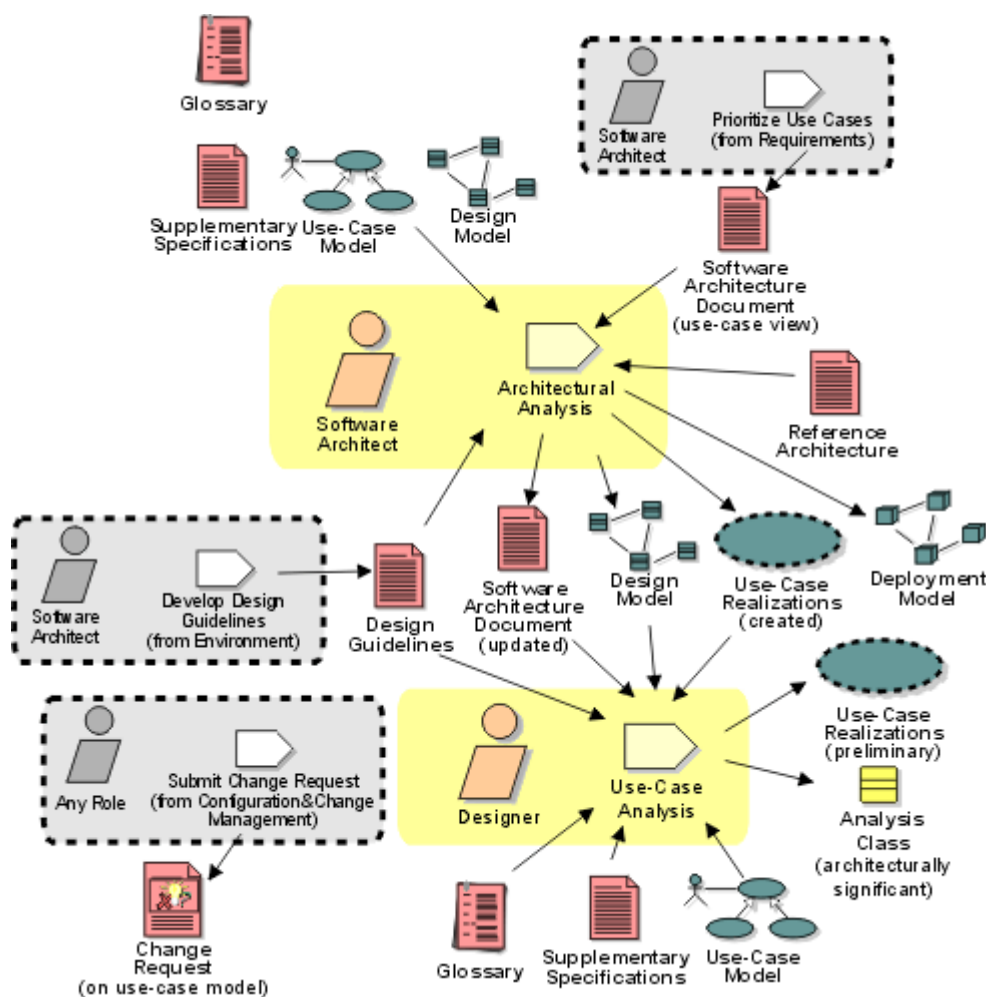
Detailní výstupy této činnosti (viz. kap. 5) jsou:

- Use-Case Realization
- Deployment Model
- Design Guidelines
- Software Architecture Dokument
- Design Model
- Analysis Class
- Analysis Model

Cíle tohoto detailního pracovního procesu jsou:

- Vytvořit první nástin architektury systému
  - Definovat počáteční soubor architektonicky významných prvků, které mají být použity jako základ pro detailní analýzu
  - Definovat počáteční nastavení analýzy mechanismů
  - Definovat počáteční vrstvu a organizaci systému
  - Definovat Use-Case Realizace
- Identifikovat Analysis Classes z architektonicky významných případů užití
- Aktualizovat Use-Case Realizace v interakci s analyzovanými třídami





**Obr. 4 – Pracovní proces Define Candidate Architecture[19]**

### 3.1.1. Architectural Analysis

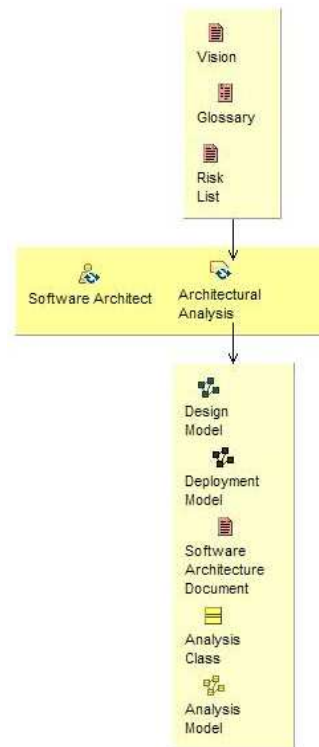
Sub-proces činnosti Define Candidate Architecture, architektonická analýza se zabývá vytvořením **analytického modelu** systému. Opírá se o zkušenosti nabyté z podobných již existujících systémů nebo problémů. Analytický model je tvořen pomocí analytických modelů (diagram tříd, diagram procesů, use case atd.) V existujících systémech, kde je již architektura dobře definována, lze architektonický rozbor vynechat a přistoupit rovnou k činnosti Refine the Architecture. Architektonická analýza je především přínosem při vývoji nových systémů a bezprecedentní. Jedná se o hrubý návrh IS či vizi budoucího stavu IS, která zachycuje jednotlivé komponenty IS/IT a jejich vzájemné vazby, obsahuje základní stavební bloky IS/IT. V tomto úkolu figuruje pracovní role Software Architect, který je zodpovědný za výstupy a prováděnou práci.[19]

Architektura IS/IT je grafické a písemné vyjádření celkové koncepce IS/IT, která v sobě zahrnuje základní představu o:

- struktuře IS v návaznosti na organizační strukturu podniku,
- funkcích, které bude IS zabezpečovat v návaznosti na procesy podniku,
- provozu a bezpečnosti celého systému, vazbách na okolí.

## Význam Architektury IS/IT:

- Vytváří relativně stabilní rámec řešení IS/IT.
- Je významným komunikačním prostředkem mezi tvůrci a vedením podniku.
- Zajišťuje stabilitu vývoje IS/IT při rychlém technologickém vývoji IT.
- Význam ekonomický - minimalizuje náklady na chybně zadané projekty a rekonstrukce.



**Obr. 5** - Pracovní proces Analysis Architecture [7]

### Kroky:

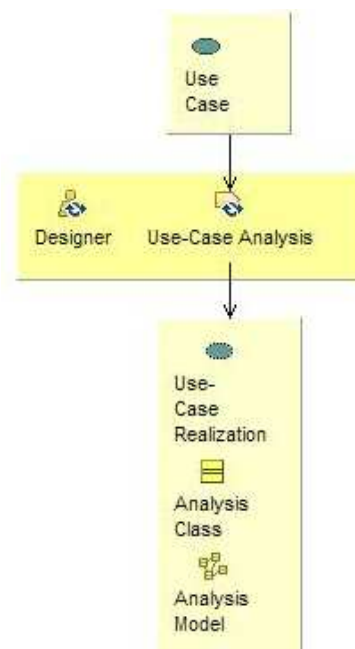
- Develop Architecture Overview – vytvořit architektonický přehled, usnadňuje zúčastněným stranám pohled na systém
- Survey Available Assets – identifikace majetku důležitého pro projekt, vytvořit seznam aktiv, které jsou použitelné pro projekt atd
- Define the High-Level Organization of Subsystem – vytvoření úvodní struktury systému - v této fázi zaměřenou většinou na dvě úrovně Aplikační a Business-specific pro design model
- Identify Key Abstractions
- Develop High Level Deployment Model - vybudování vysokoúrovňového modelu nasazení
- Identify Analysis Mechanisms – Analýza mechanismů IS
- Create Use-Case Realizations - Vytvoření use case realizace
- Identify Stereotypes and Interactions
- Review The Results - Přezkoumání výsledků

### 3.1.2. Use Case Analysis

Use Case analýza slouží k zachycení dalších informací nezbytných k pochopení požadovaného vnitřního chování systému, které mohou chybět z popisu use case popsaného zákazníkem ve fázi projektu Inception. Obecně use case poskytuje podrobné informace o chování systému nebo aplikace, která je vyvíjena. Zachycuje vazby mezi aktéry a funkcemi systému a mezi funkcemi navzájem.[13] Při použití některých architektur lze use case diagramy vhodně využít i ve fázi designu k popisu komponent. V tomto úkolu figuruje pracovní role Designer, který je zodpovědný za výstupy a prováděnou práci.

Podrobnější cíle:

- Identifikovat třídy, které vystupují v tocích událostí use case
- Přidělit funkce chování do identifikovaných tříd, pomocí use case realizací
- Identifikovat odpovědnosti, atributy tříd a asociace mezi třídami
- To note the usage of architectural mechanisms



**Obr. 6** - Pracovní proces Use Case Analysis [7]

Kroky:

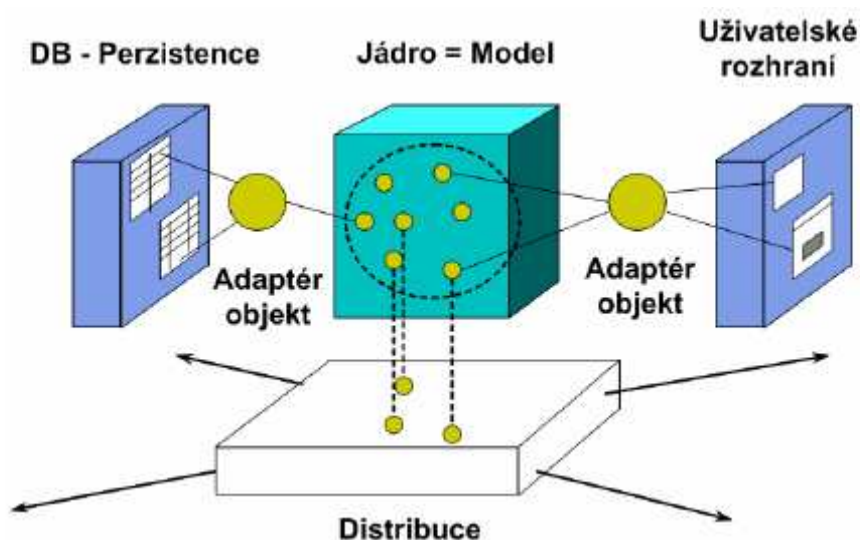
- Supplement the Use-Case Description - zachytit další informace nezbytné k pochopení požadovaného vnitřního chování systému, který by mohl být chybějící z popisu use case psaného pro systém zákazníky
- Find Analysis Classes from Use-Case Behavior
- Distribute Behavior to Analysis Classes – distribuovat analýzu chování do tříd
- Describe Responsibilities – popsat odpovědnosti
- Describe Attributes and Associations – popsat atributy a asociace mezi třídami
- Reconcile the Analysis Use-Case Realizations
- Qualify Analysis Mechanisms
- Establish Traceability

Review the Results

### 3.2. Refine the Architecture

Pokud se doposud nevyskytly nějaké větší problémy vyžadující více iterací a s tím tedy přímo úměrné zdržení projektu, přechází projekt do fáze zpřesnění analyzované architektury. Tato činnost v průběhu projektu představuje jakýsi přirozený přechod od „analýzy“ k „designu“ IS. Činnost Refine the Architecture běží paralelně s designem systému, kdy dochází jednak k designu IS, druhak k upřesnění designu v rámci každé iterace.

Architektonický design je klíčovou fází vývoje informačního systému. Jeho kvalita rozhoduje o úspěchu celého projektu. Při tvorbě designu systému pracovníci vychází z analytických modelů, předem definovaných v předchozích krocích analýzy IS. Jádrem systému vytvořené v etapě analýzy bude postupně doplněno o níže uvedené tři vrstvy a rozpracováno do detailů.



Obr. 8 – Architektura softwarového systému [15]

Meziprodukt analytický model bude postupně doplňován o aspekty týkající se implementace a nasazení systému. Cílem etapy návrhu, jež můžeme říci, že právě začíná v rámci toku činností zabývajících se analýzou a designem, je vytvoření **design modelu**.

Výsledkem architektonického designu je jasně specifikovaná architektura IS. Návrhový model upřesňuje model analytický ve světle skutečného implementačního prostředí. Představuje tak abstrakci zdrojového kódu, jinými slovy řečeno, reprezentuje „výkresovou“ dokumentaci určující, jak bude zdrojový kód strukturován a napsán.[15]

Softwarový Architekt na základě use case analýzy identifikuje jednotlivé prvky pro design IS. Systém je mimo jiné množina komponent (prvků), která interaguje, aby splnila nějaký cíl, tzn. pracovníci budou nejdříve identifikovat budoucí komponenty systému. Na základě analytických modelů, společného slovníku, zkušeností z praxe v daném oboru Softwarový architekt **navrhne** způsob rozdělení IS na subsystémy, identifikuje klíčové třídy apod. Rovněž zohlední prvky stávajícího systému, které se budou s novým systémem integrovat, včetně jejich rozhraní.

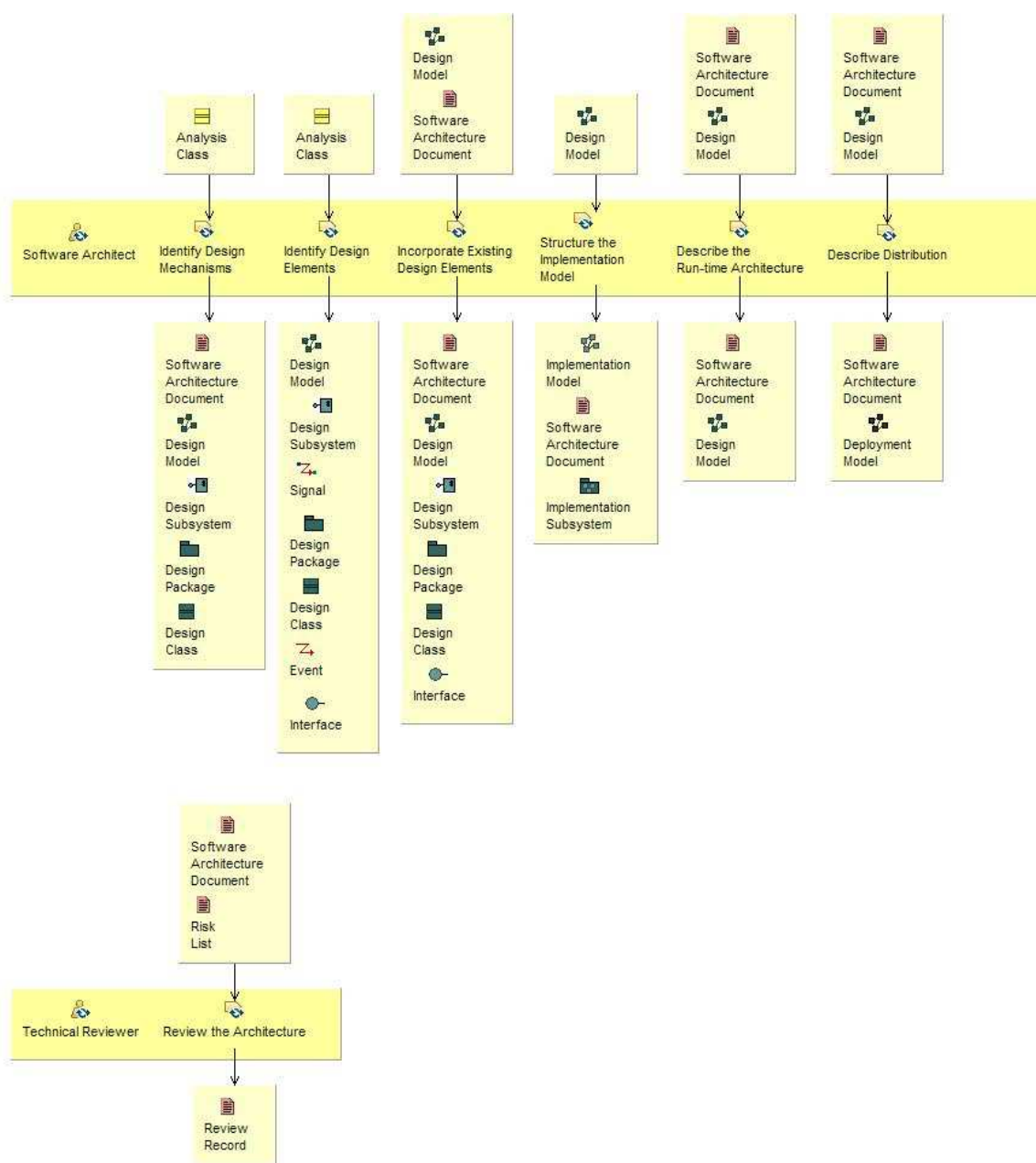
Dále je řešena např.: souběžnost procesů či fyzická distribuce systému. Činnost Refine the Architecture obsahuje několik úloh, které dále popíši: Identify Design Mechanisms, Identify Design Elements, Incorporate Existing Design Elements, Structure the Implementation Model, Describe the Run-time Architecture, Describe Distribution a úlohu Review the Architecture.

Detailní výstupy této činnosti (viz. kap. 5):

- Design Class
- Design Model
- Design Package
- Design Subsystem
- Software Architecture Document
- Event
- Interface
- Signal
- Interface
- Implementation Model
- Implementation Subsystem
- Deployment Model
- Review Record

Cíle tohoto detailního pracovního postupu jsou:

- Přirozený přechod od analýzy k návrhu a identifikaci:
  - Vhodný **design** prvků z **analýzy**
  - Vhodný design mechanismů z analýzy
- Udržet konzistenci a integritu architektury a zajistit, aby:
  - Nově identifikované prvky designu v současné iteraci byly integrovatelné s již existujícími prvky
  - Zajistit maximální znovupoužití dostupných komponent a prvků designu
- Popsat souběžnost procesů a fyzickou distribuci systému
- Zajistit/uspořádat model implementace tak, aby přechod mezi návrhem a implementací byl bezproblémový



**Obr. 9 - Pracovní proces Refine the Architecture [7]**

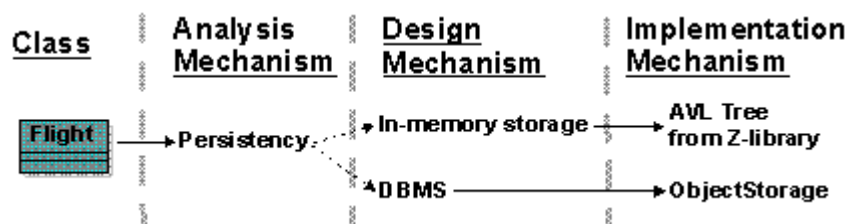
### 3.2.1. Identify Design Mechanisms

Úkolem Softwarového Architekta v této úloze je rozšířit analýzu mechanismů o detailní mechanismy pro design systému. Identifikace mechanismů pro zajištění správné funkce systému při sdílených zdrojích závisí velmi na programovacím jazyku zvoleném pro implementaci a na cílové platformě.[13]

Jedná se jinými slovy o doplnění analýzy mechanismů o některé detailní informace z implementačního prostředí. Definování kritérií na architekturu systému. Softwarový architekt by měl mít na paměti stále zachování abstraktnosti - což je velmi obtížné, tzn. stále ještě není design model upřesněn pro konkrétní implementační prostředí a celkový design. K upřesnění dojde v pozdějších iteracích.

Objekty systému jsou různorodé a mohou proto vyžadovat rozdílnou podporu systému (např.: požadavek na rychlost odezvy na vzdáleném serveru apod.). Ukázkovým příkladem

Identify Design Mechanismus je např.: určit požadavky objektů IS a identifikovat podporu pro prvky IS.



Kroky:

- Categorize Clients of Analysis Mechanismus
- Inventory the Implementation Mechanisms
- Map Design Mechanisms to Implementation Mechanisms
- Dokument Architectural Mechanisms

### 3.2.2. Identify Design Elements

V této úloze figuruje rovněž pracovní role Softwarový architekt. Cílem je popsat strukturální vlastnosti systému. Prvky IS jsou obecně elementární, dále nedělitelné části systému, které představují jeho rozkladové (dekompoziční) části. Tedy taková část systému, která tvoří na dané rozlišovací úrovni nedělitelný celek, jehož strukturu nechceme, nebo nemůžeme rozlišit.[6] Identifikace a následná dekompozice klíčových komponent je, dá se říci, jednou z nejtěžších částí OO návrhu.

Při definici systému nebo subsystému obvykle začínáme u standardního rozdělení na komponenty modelu, funkce a rozhraní. V případě komplexních návrhů je potřeba ale dále postupovat s dekompozicí jednotlivých komponent, jenž bude vykonáno v pozdějších iteracích.[13]

Identifikace klíčových komponent vyžaduje schopnost objevovat i vynalézat. Objevit je třeba klíčové abstrakce či mechanismy. Vynalézt je třeba zobecněné abstrakce či takový mechanismus, kterým objekty vzájemně spolupracují. Vychází se analytickým modelem tříd vytvořeným ve fázi analýzy, který je postupně zpřesňován, obohacován o implementační detaily, a tak transformován na designový model tříd. Identifikaci tříd často předchází slovní analýza. Jak postupovat v tomto sub-procesu, tedy jaké kroky je třeba vykonat je popsáno níže. [6, 3]

Kroky:

- Identify and Specify Events and Signal - identifikace a specifikace externích a interních událostí, na které musí IS reagovat
- Identify Classess, Aktive Classes and Subsysteme – identifikace prvků (komponent)
- Identify Subsystem Interfaces – identifikace rozhraní pro komunikaci sub-systémů

### 3.2.3. Describe the Run-time Architecture

Softwarový architekt má v této úloze za úkol popsat architekturu procesů systému. Architektura procesů systému vyjadřuje strukturu nezávislých procesů popisující běh systému. Popis běhového prostředí systému je vyjádřen ve smyslu aktivních objektů jejich instancí a jejich vztahů, procesů k procesům operačního systému. Je třeba identifikovat

procesy a jejich životní cyklus, distribuovat prvky modelu mezi procesy apod. Postup se skládá z několika na sebe navazujících kroků. Tyto kroky jsou velmi důležité pro úspěšnost zavedení celého systému jakosti orientovaného na procesy.

Běžové prostředí zbavuje programátory břemene psaní kódu pro úkoly, jako je kreslení textu na obrazovku nebo připojení k internetu. Poskytuje tak abstraktní vrstvu, která skrývá složitost nabízených služeb operačního systému.[16]

Kroky:

- Analyze Concurrency Requirements
- Identify Processes and Threads – Identifikovat procesy(vlastníky procesů) a vlákna, které budou v IS existovat.
- Identify Process Lifecycles – Identifikovat kdy jsou procesy a vlákna vytvářeny a kdy zanikají
- Identify Inter-Process Communication Mechanisms - Identifikovat prostředky, které procesy a vlákna se bude komunikovat. (Typická komunikace mezi procesy mechanismy zahrnuje - Shared memory, Mailboxes, Rendezvous)
- Allocate Inter-Process Coordination Resources
- Map Processes onto the Implementation Environment - Při tomto určení je třeba vzít v úvahu současný stav firmy, ale tento stav nelze pouze popsat jako procesy. Především se musí vzít v úvahu očekávání a zkušenosti zákazníka, vlivy společnosti, dopady na životní prostředí a další vlivy, které působí na firmu zvnějšku a vlivy, kterými působí firma na své okolí.
- Map Design Elements To Threads of Control

### 3.2.4. Describe Distribution

Tento úkol se vztahuje v metodice Rational Unified Process pouze na distribuované systémy. Distribuované systémy neboli systémy rozmístěné na větší ploše jsou postavené především na komunikaci. Jsou určitým způsobem použity navzájem komunikující počítače, programovatelné automaty nebo jednočipové procesory a činnost celého systému je ovlivněna fungováním a vzájemnou komunikací jeho jednotlivých částí. [17] Procesní architektura musí navrhnout systém tak, aby vytěžoval rovnoměrně všechny dostupné procesory. Každá oblast, kde se distribuované systémy používají, klade jiné nároky na výkonnost použité komunikační sítě, na rychlost odezvy, na množství přenášených dat. Prvním krokem při distribuci prvků (komponent) na fyzické procesory je separace programových komponent, které neobsahují žádné aktivní objekty, a samotných aktivních objektu. V dalším kroku je potřeba identifikovat dostupné procesory, které využijeme k vykonávání systému. Třetím krokem je vlastní rozmístění programových komponent obsahujících pouze pasivní objekty a rozmístění aktivních objektu na dané procesory.[13]

Kroky:

- Analyze Distribution Requirements – Analyzovat rozsah požadavků pro distribuovaný systém
- Define the Network Configuration - definovat nastavení a topologii sítě
- Allocate System Elements to Nodes – rozmístění programových komponent



### **3.2.5. Review the Architecture**

Review the Architecture má za cíl odhalit jakákoli neznámá nebo vnímaná rizika v plánu nebo rozpočtu projektu, zjistit případné chyby architektonické analýzy a identifikace.

Architektonické chyby jsou známy jako nejtěžší na opravu a nejškodlivější v dlouhodobém horizontu projektu. Dále detekovat potenciální nesoulad mezi stanovenými požadavky a navrhnutou architekturou: nereálné požadavky, nebo chybějící požadavky. Vyhodnotit jednu nebo více konkrétních architektonických flexibilit: výkon, spolehlivost, modifikovatelnost, bezpečnost. Identifikovat možnosti opětovného použití.

Kroky:

- General Recommendations
- Recommended Review Meetings
- Allocate Defect Resolution Responsibilities

### 3.3. Analyze Behavior

V kapitole 3.2.2 jsem nepatrně nastínil proces tzv. dekompozice IS. Analýza chování systému právě vede k dekompozici zkoumaného systému na subsystemy, či prvky a vazby mezi nimi. Systém tedy můžeme graficky zobrazit jako reprezentaci fyzické konfigurace reálných objektů, nebo jejich skupin s příslušnými vazbami nebo propojeními. [6] Hlavním cílem a jedním z kroků dekompozice v této činnosti je transformovat popsané chování systému do souboru prvků subsystemů. Určit, jak analyzované třídy zapadají do logické architektury (hlavní subsystemy a třídy) IS.

Chování je projevem dynamiky systému. Dynamika je schopnost vyvolat změnu systému, zejména jeho stavu. Chování prvků ve vzájemných interakcích, které vyplývají ze struktury systému, určuje chování celého systému. Popisujeme je pomocí vstupních, stavových a výstupních charakteristik. [6]

V této činnosti může také dojít k úvodnímu design uživatelského rozhraní. Až doposud se činnosti zabývaly více tzv. nefunkčními požadavky IS, v tomto kroku je však úkolem zajistit potřebnou funkcionalitu IS.

Tato činnost zahrnuje úkoly Identify Design Elements, Use-Case analysis prováděné Designerem, Design the User Interface prováděné Designerem, Prototype the User-Interface, Review Design prováděné Design Reviewer.

Detailní výstupy této činnosti jsou:

- Design Class
- Design Model
- Design Package
- Design Subsystem
- Event
- Interface
- Signal
- Analysis Class
- Analysis Model
- Use-Case Realization
- Navigation Map
- User-Interface Prototype
- Review Record

#### 3.3.1. Identify Design Elements

Dochází k doplnění a upřesnění prvků systému v další iteraci viz. 3.2.2.

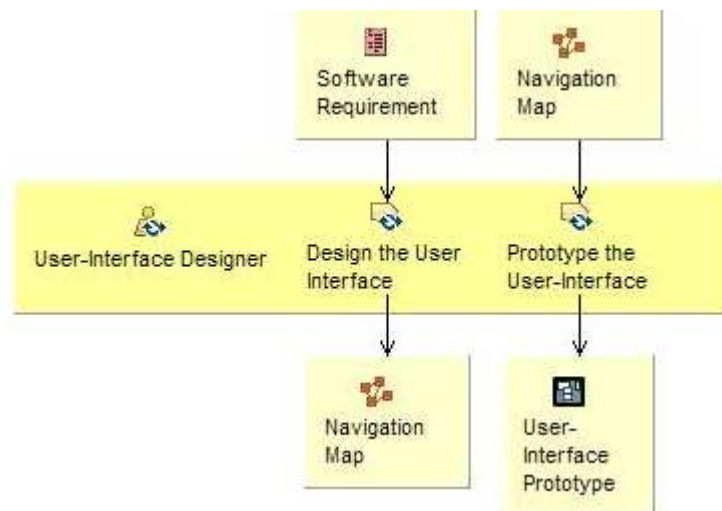
#### 3.3.2. Use-Case Analysis

Dochází k doplnění a upřesnění prvků systému v další iteraci viz. 3.1.2

#### 3.3.3. Design the User Interface

Cílem sub-procesu Design the User-Interface je identifikovat hlavní prvky IS uživatelského rozhraní, dále vytvořit navigační mapu jakožto orientační mapu IS uživatelského rozhraní. Úloha Design the User-Interface společně s následující úlohou Prototype the User-Interface jsou prováděny rovněž ve více iteracích skrz Elaborační fázi.

První iterace návrhu UI zahrnuje identifikaci a návrh klíčových uživatelských prvků a navigaci mezi nimi. K návrhu UI je doporučeno využít metody tzv. Storyboarding či Personas, jenž jsou efektivními technikami, které pomohou lépe pochopit, jak by se mělo UI chovat. Jakmile je rozhodnuto o prvním návrhu UI, začíná sub-proces Prototype the User-Interface. Tato úloha vysvětluje způsob chování GUI designu s důrazem na použitelnost.



**Obr. 10** - Pracovní proces Design the User Interfaces [7]

Kroky:

- Describe the Characteristics of Related Users
- Identify the Primary User-Interface Elements – Identifikace hlavních uživatelských prvků
- Define the Navigation Map – Definovat navigační mapu
- Detail the Design of the User-Interface Elements -

### 3.3.4. Prototype the User-Interface

Tento subprocess popisuje vývoj prototypu UI systému s cílem získat zpětnou vazbu použitelnosti. Hlavní výhodou vývoje „prototypu“ verze UI je eliminace pozdějších investic do úpravy UI systému. Vývoj konečného UI by měl být zahájen až po celkové shodě UI.

Při navrhování a prototypování uživatelského rozhraní s cílem potvrdit a ověřit použitelnost systému je důležité úzce spolupracovat s potenciálními uživateli UI. První prototyp UI většinou podporuje pouze podmnožinu vlastností systému. V následujících iteracích je prototyp rozšiřován, postupně přidáváno širší pokrytí vlastností systému.

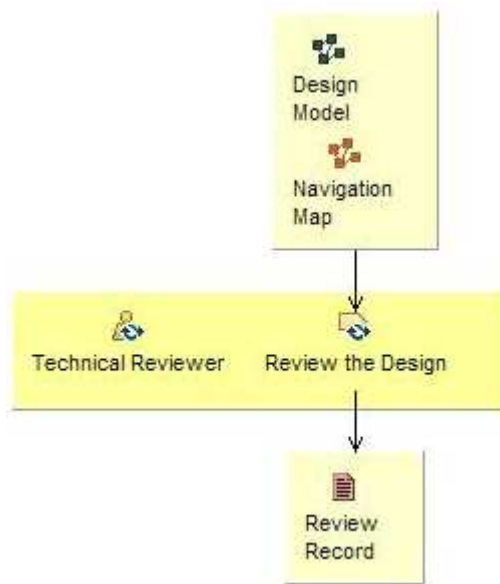
Kroky

- Design the User-Interface Prototype – Návrh prototypu UI
- Implement the User-Interface Prototype – Implementace prototypu UI
- Get Feedback on the User-Interface Prototype – Získat zpětnou vazbu UI

### 3.3.5. Review the Design

Tato úloha definuje, jak provést revizi designu a jak postupovat v přezkoumání výsledků. Ověřit, že model designu splňuje požadavky na systém, a že slouží jako dobrý základ pro jeho

implementaci. Zajistit, že model designu je konzistentní s ohledem na obecné navrhované zásady směrnic.



**Obr. 11** - Pracovní proces Review the Design [7]

Kroky:

- Review the Design Model as a Whole – globálně upřesnit design model
- Review each Use-Case Realization – upřesnit use case realizace
- Review each Subsystem (and its contents) or Class (if the system is small) – upřesnit subsystémy a třídy
- Review Design Guidelines -
- Prepare Review Record and Document Defects -

### 3.4. Design Components

Jak již bylo řečeno architektonický design je klíčovou fází vývoje informačního systému. Jeho kvalita rozhoduje o úspěchu celého projektu. Při tvorbě designu systému vycházíme z analytických modelu, předem definovaných v analýze IS.

Ve zkratce, během designu tedy: [13]

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

Složitý systém je nutno popsat velkým počtem diagramů. Po celý životní cyklus Rational Unified Process je tedy nutné respektovat při modelování softwarového systému principy, umožňující vytvářené modely, obvykle reprezentované celou řadou diagramů, vhodným způsobem strukturovat a lépe tak spravovat celou dokumentaci vytvářeného produktu.

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

Těmito pohledy jsou komponentová a procesní architektura. Komponentová architektura se zaměřuje na třídy, tedy statické aspekty systému. Rozděluje systém na jasné identifikovatelné, vzájemně propojené komponenty, řeší jejich vnitřní stavy a přechody mezi nimi. Architektura procesu se naproti tomu věnuje instancím tříd (dynamické aspekty systému). Dělí systém na procesy, řeší jejich vzájemné interakce a koordinaci.[13]

Cílem této činnosti je Softwarový architekt navrhne definici modelu pro potřeby zvoleného implementačního prostředí (programovací jazyk, relační databáze, distribuce systému aj.) Pokud je k dispozici definice základních designových prvků viz., vytvoří k nim návrhář přesnější popis jejich funkcionality, aby je bylo možné implementovat jako komponenty. Navrhuje se též hierarchická struktura modelu pro implementaci. V tomto kroku lze též provést identifikaci klíčových prvků datového modelu.

Tato činnost se skládá z úloh Use-case Design, Class Design a Subsystem Design. Za tyto činnosti je zodpovědný Designer, za činnost Review the Designz odpovídá Technical Reviewer.

Detailní výstupy této činnosti jsou:

- Design Model
- Use-Case Realization
- Design Class
- Design Model
- Design Subsystem
- Interface
- Review Record

### 3.4.1. Use-Case Design

Tato úloha definuje použití interakcí, zejména pak sekvenčních diagramů, které popisují chování systému. Sekvenční diagramy jsou velmi užitečné, pokud je chování systému nebo subsystémů především popsát synchronní komunikaci. Asynchronní zasílání zpráv, a to zejména v systémech řízené událostmi, je často lépe popsány z hlediska státních strojů a spolupráci, což umožňuje kompaktní způsob, jak definovat možné interakce mezi objekty.

Asynchronní zprávy hrají důležitou roli v reálném čase, a používají se pro komunikaci mezi instancemi. Rozšiřují návrh tříd o detaily jako jsou operace a metody, přidávají sekvenční a komunikační diagramy.

Cíle:

- Zpřesnit use case realizace o termíny interakce
- Zpřesnit navrhované třídy o požadavky na operace tříd
- Zpřesnit navrhované třídy a subsystémy o požadavky na operace

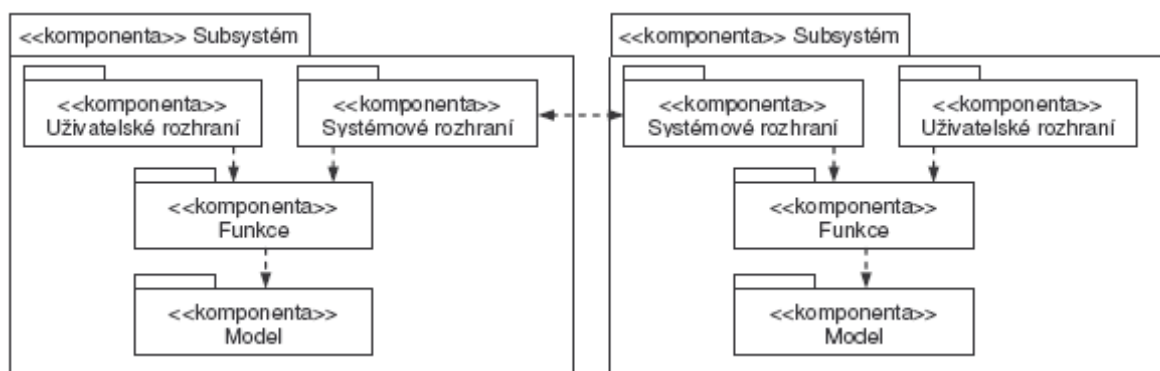
Kroky:

- Create Use-Case Realizations
- Describe Interactions Between Design Objects
- Simplify Sequence Diagrams Using Subsystems (optional)
- Describe Persistence-Related Behavior
- Refine the Flow of Events Description
- Unify Design Classes and Subsystems
- Evaluate Your Results

### 3.4.2. Subsystem Design

Subsystem (tzv. balíček) je podmnožina prvků systému, kterou můžeme popsat jako samostatný systém se specifickou charakteristikou. Balíček je obecný mechanismus umožňující organizovat prvky do skupin s cílem redukovat složitost modelovaného systému. Tyto balíčky mohou sdružovat libovolné části modelovaného systému. Pokud se bude jednat o části, které spolu souvisejí z hlediska zajištění např.: nějaké společné funkce či účelu, pak hovoříme o designu subsystémů. [6, 15]

U jednodušších systémů není problém využít standardní architekturu tzv. Model View Controller. Celý systém tak bude složen pouze ze třech komponent: model, funkce a uživatelské rozhraní. Designovat tímto způsobem komplexnější systémy je nevhodné. Složitější IS je třeba rozdělit na více částí (sub-systémů), které spolu budou komunikovat pomocí jasně definovaných rozhraní. Každý subsystém se tak bude skládat dále z vlastních sub-komponent. Jak je vidět na následujícím obrázku, každý subsystém obsahuje navíc komponentu Systémové rozhraní (system interface). Na rozdíl od komponenty uživatelského rozhraní, která umožňuje interakci s uživatelem, stará se systémové rozhraní o zpřístupnění funkcí jiným sub-systémům.



Obr. 12 - Architektura subsystémů [15]

Cíle:

To define the behaviors specified in the subsystem's interfaces in terms of collaborations of contained design elements and external subsystems/interfaces.

To document the internal structure of the subsystem.

To define realizations between the subsystem's interfaces and contained classes.

To determine the dependencies upon other subsystems

Kroky:

- Distribute Subsystem Behavior to Subsystem Elements
- Document Subsystem Elements – zdokumentovat sub-systémy
- Describe Subsystem Dependencies – popsat závislosti sub-systémů

### 3.4.3. Class Design

Proces návrhu tříd má poskytnout požadované chování, které odpovídá návrhovým rámcům. Zahrnuje také návrh rozhraní a je konzistentní s designem interakcí.

Diagram tříd, jenž je jedním z modelů designu, se používá na popis statické struktury systému, zejména na několika úrovních abstrakce. Začíná se analytickým modelem tříd, který

je postupně zpřesňován, obohacován o implementační detaily (například o viditelnosti atributů a metod, datové typy apod). Dále do modelu přidává třídy uživatelského rozhraní (presentation classes) a třídy obsluhující systémové události (control classes). [4] Z jedné třídy v analytickém modelu se tedy může stát v designovém modelu více návrhových tříd. Díky tomu, že diagram tříd zachycuje pravidla modelovaného systému, je nejdůležitějším podkladem jak pro forward engineering, tak pro reverse engineering. [10]

Oproti ostatním konstrukčním prvkům jako jsou subsystemy, balíčky a collaborations, popisují, jak jsou třídy rozděleny nebo jak spolupracují. Třídy jsou prvky designu, které vykonávají skutečnou práci systému.

Cíle:

Zajistit, aby třídy obsahovaly požadované chování z use case realizace

Zajistit, To ensure that sufficient information is provided to unambiguously implement the class

To handle nonfunctional requirements related to the class

To incorporate the design mechanisms used by the class

Kroky:

- Use Design Patterns and Mechanisms
- Create Initial Design Classes
- Identify Persistent Classes
- Define Class Visibility
- Define Operations
- Define Methods
- Define States
- Define Attributes
- Define Dependencies
- Define Associations
- Define Internal Structure
- Define Generalizations
- Resolve Use-Case Collisions
- Handle Nonfunctional Requirements in General
- Evaluate Your Results

#### **3.4.4. Review the Design**

Ověřit, že design model splňuje požadavky na systém, a že bude sloužit jako dobrý základ pro implementaci. Zajistit, aby byl design model v souladu s obecnými zásadami návrhu.

Kroky:

- General Recommendations
- Review the Design Model as a Whole
- Review Each Design Use-Case Realization
- Review Each Design Element
- Review Design Guidelines
- Prepare Review Record and Document Defects

### 3.5. Design the Database

Design the Database se skládá z úloh: Database Design, který provádí Database Designer, Class Design, provádí jej Designer, a Review the Design, provádí jej Design Reviewer. V této činnosti návrhář databáze určí perzistentní třídy a navrhne k nim příslušné datové struktury, definuje mechanismus a strategii pro načítání a ukládání dat, aby byla splněna kritéria pro výkonnost systému.

Podrobnější cíle:

- provést mapování tříd z designu do datového modelu
- optimalizace datového modelu (datových struktur) z hlediska výkonnosti
- optimalizace přístupu k datům (použití indexů a jejich typů apod.)
- definice charakteristik uložení dat (velikost stránky apod.)
- definice referenčních tabulek a přednastavených hodnot atributů (např. uložení nejpoužívanější tabulky na nejrychlejší disk, optimalizace vyrovnávacích pamětí apod.)
- definice pravidel datové a referenční integrity
- distribuce chování třídy do databáze (uložené procedury apod.)
- přezkoumání výsledku (ověření kvality a integrity datového modelu)

Detailní výstupy této činnosti jsou:

- Design Class
- Design Model
- Data Model
- Review Record

#### 3.5.1. Class Design

Viz. 3.4.3

#### 3.5.2. Database Design

Základní návrh databáze se obecně realizuje pomocí tzv. logického a posléze fyzického datového modelu. Datový model obsahuje informace o tom, co přesně v datové sadě je, jak je daný objekt nebo jev reprezentován, jaké atributy jsou objektům a jevům přiřazeny, jakých hodnot jejich atributy mohou nabývat a mnohé další užitečné informace.

Fyzický datový model a jemu předcházející logický datový model, by měl být tím prvním krokem, který návrhář chystající se vytvořit nějakou datovou sadu podnikne. Je zcela jedno, zda budou data vznikat vektorizací, digitalizací, terénním průzkumem nebo jiným způsobem. Předtím než vznikne první objekt v nově vytvářené datové sadě nebo databázi, předtím než vůbec vznikne prázdná datová sada nebo databáze, je nezbytné připravit důkladný logický datový model. Z něj je následně odvozen fyzický datový model, který respektuje možnosti cílového formátu, ve kterém budou data nakonec uložena.



Kroky:

- Develop Logical Data Model (Optional) - Logický datový model popisuje datové struktury v obecné rovině, nezávisle na konkrétním databázovém stroji. V přehledu vlastností tak nenalezneme ani informace o způsobu uložení dat, ani náhled SQL pro vytvoření jednotlivých objektů
- Develop Physical Database Design - Fyzický datový model zobrazuje data s nižší abstrakcí než koncepční datový model, ze kterého vychází. Obsahuje navíc fyzickou reprezentaci dat pomocí proměnných a relací pomocí relačních tříd. Tyto třídy jsou největší změnou oproti koncepčnímu modelu. Zaznamenávají relace N:M tak, že shromažďují hodnoty primárních klíčů z obou tříd v relaci. Tato dvojice je pak identifikována vlastním primárním klíčem.

### **3.5.3. Review the Design**

Tato úloha definuje, jak provést revizi návrhu databáze a jak řešit přezkoumání výsledků. Cílem je ověřit, že Design Model splňuje požadavky na systém, a že slouží jako dobrý základ pro jeho implementaci.

Kroky:

- General Recommendations
- Review the Design Model as a Whole
- Review Each Design Use-Case Realization
- Review Each Design Element
- Review Design Guidelines
- Prepare Review Record and Document Defects

## 4. Role

**Database Designer** – návrhář databáze je odpovědný za detailní návrh databáze zahrnující: tabulky, indexy, omezení, trigery, ukládací procedury a ostatní specifické databázové konstrukce potřebné pro ukládání, vyvolání či odstranění přetrvávajících objektů

**Designer** - identifikuje a definuje odpovědnosti, operace, atributy a vztahy konstrukčních prvků. Zajišťuje, že design je v souladu se softwarovou architekturou, a je uveden do stavu, kdy může následovat implementace systému.

**Software Architect** - má celkovou odpovědnost za řízení velkých technických rozhodnutí, ohledně architektury systému. To obvykle zahrnuje identifikaci a dokumentaci architektonicky významných aspektů systému, včetně požadavků, návrhu, implementace a nasazení tedy ze všech „pohledů“ systému.

**System Analyst** - vede a koordinuje požadavky elicítace vymezením funkčnosti IS

**Technical Reviewer** - organizuje odpovědnost za plnění úkolů a vývoj prací na produkty ve skupinách. Je třeba zvážit i dovednosti potřebné pro úlohy a různé přístupy. Může zaměstnancům přidělovat role. Účelem Technického reviewera je, aby se dospělo k technicky dokonalejší verzi přezkoumávaného produktu, zda je v souladu doporučeními či zavedení alternativních přístupů

**Test Designer** - určuje vhodné metody, nástroje a pokyny k provedení požadovaných testů

**User-Interface Designer** - koordinuje návrh uživatelského rozhraní. Zahrnuje sběr požadavků na použitelnost UI a navrhuje prototyp uživatelského rozhraní v rámci těchto požadavků.

## 5. Výstupy

Work Product	Cíl
Analysis Model	<p>Model analýzy je užitečný pro lepší porozumění požadavkům dříve než se vytvoří design model. Pomocí tohoto teoretického popisu IS se dostává pracovníkům do rukou silná zbraň pro zpracování mnoha funkcionálních požadavků, přehledným srozumitelným a systematickým způsobem.</p> <p>Model analýzy je složen z analytických modelů (Class diagram, Process diagram, State diagram atd.)</p>
Navigation Map, User-Interface Prototype	<p>Je souhrn způsobů, jakými lidé (uživatelé) ovlivňují chování strojů, zařízení, počítačových programů či komplexních systémů. Projekty s velkým a složitým uživatelským rozhraním by měly zvážit návrh uživatelského rozhraní.</p> <p>Typy rozhraní:</p> <ul style="list-style-type: none"> <li>• grafické uživatelské rozhraní – nejrozšířenější rozhraní pro desktop</li> <li>• textové uživatelské rozhraní – s menu, tlačítky, ovládání klávesnicí a myší</li> <li>• příkazový řádek – příkazy se zadávají zápisem pomocí klávesnice</li> <li>• braillovský řádek – zařízení pro převod textu do slepeckého písma</li> <li>• atd.</li> </ul>
Design Model	<p>Upřesňuje model analýzy ve světle skutečného implementačního prostředí. Model návrhu tak představuje abstrakci zdrojového kódu, jinými slovy řečeno, reprezentuje „výkresovou“ dokumentaci určující jak bude zdrojový kód strukturován a napsán.</p>
Design Class, Design Package	<p>Třídy jsou základní součástí každého objektově orientovaného návrhu. Objektově orientovaný návrh je standardně používán na většině projektů. Specifikuje množinu tříd, rozhraní a jejich vzájemné vztahy. Diagramy tříd slouží k vyjádření statického pohledu na systém.</p> <p>Balíček je mechanismus obecného použití umožňující organizovat elementy do skupin s cílem redukovat složitost modelovaného systému.</p>
Use-Case Realization	<p>Poskytuje most z případů užití k designu. Účelem tohoto dokumentu je popsat jednotlivé případy užití a propojit pohled případů užití s logickým a datovým pohledem.</p>
Interface	<p>Zařízení, program nebo formát zajišťující spojení mezi jinými zařízeními nebo programy.</p>
Design Subsystem	<p>Subsystémy sdružují libovolné části modelovaného systému. Pokud se bude jednat o části, které spolu souvisejí z hlediska zajištění nějaké společné funkčnosti či účelu pak hovoříme o vytváření tzv. subsystémů.</p>
Event	<p>Dokument zaznamenávající posloupnost požadavků účastníka systému a reakcí systému</p>
Data Model	<p>Používá se k popisu logické a případně i fyzické struktury dat.</p>
Deployment Model	<p>Model nasazení definuje fyzickou alokaci částí systému (jeho komponent) na fyzické zdroje (výpočetní uzly/servery).</p>

Reference Architecture	Referenční architektura slouží k urychlení vývoje a snížení rizika opětovného použití osvědčených řešení.
Software Architecture Document (SAD)	Architektura softwaru dokumentu se používá k ucelenému architektonickému přehledu o systému. Tento přehled umožňuje porozumět systému a zachytit klíčové stavební rozhodnutí informačního systému. SAD je výsledkem procesu skládající se z - sady modelů a popisů - celého životního cyklu Rational Unified Process.

## 6. Zhodnocení

Práce na Analýze a Designu jsou ve velké většině rozděleny mezi pracovníky Softwarový Architekt, Designer, a Designer Databáze.

Jak je patrné z této práce, Rational Unified Process je velice komplexní a obsáhlá metodika. Popisuje fáze vývoje softwaru již od raných počátků, radí, čím bychom měli začít, jak bychom měli dále postupovat až do nasazení výsledného produktu. Rational Unified Process díky nadefinovanému pracovnímu procesu všech svých kroků vysvětluje jednotlivé kroky, jejich sled v rámci procesu, co se očekává od každé role jaké jsou potřebné vstupy a jak by měl vypadat vzorový výstup.

Ačkoliv navigace v metodice (distribuována jako sada HTML stránek) je velmi intuitivní, nezkušený člověk se v ní velmi rychle ztratí. Každý proces odkazuje na mnoho dalších míst, ke každému procesu či artefaktu je definován seznam kroků, které je potřeba mít hotov ještě před zahájením. A každý tento předchůdce má opět svůj seznam předpokladů. Opět se nám zde objevuje potřeba mít pro implementaci RUP řídícího člověka, který dokáže zbytek týmu v případě potřeb navigovat.

Disciplínu Analýza a Design považuji za stěžejní milník v metodice. Ze statistických údajů vyplývá jako druhá nejvíce časově náročná fáze. Kvalita výstupů rozhoduje o budoucím úspěchu celého projektu a dodržení stanoveného ať už časového tak i finančního plánu.

## 7. Zdroje

[1]

ALBRECHT, Jakub. *Využití metodiky RUP při vývoji webové aplikace menšího rozsahu v prostředí PHP 5*. Praha. Vysoká škola ekonomická, 2008. Diplomová práce. Vysoká škola ekonomická.

[2]

ALDORF, Filip. *Metodika RUP*. Vysoká škola ekonomická v Praze. Praha, 2008. Dostupné z: <http://objekty.vse.cz/Objekty/RUP>. Diplomová práce. Vysoká škola ekonomická v Praze.

[3]

BUHNOVÁ, Bára. Ilustrační příklad v jazyce UML. In: 2005 - 2010 . Dostupné z: <http://www.fi.muni.cz/~buhnova/PV167/prikklad.html>

[4]

BUCHALCEVOVÁ, Alena; PAVLÍČKOVÁ, Jarmila; PAVLÍČEK, Luboš. *Základy softwarového inženýrství - materiály ke cvičení*. 1.vyd. Praha : Vysoká škola ekonomická, 2007. 222 s. ISBN 987-80-245-1270-9.

[5]

COLE, Leonardo, Eduardo Kessler PIVETA a Augusto SAMPAIO. *RUP Based Analysis and Design with Aspects*. [online]. Federal University of Santa Catarinas. 15 [cit. 2011-12-22]. Dostupné z: [http://www.cin.ufpe.br/~lcn/publications/SBES2004\\_Cole\\_Piveta\\_Sampaio.pdf](http://www.cin.ufpe.br/~lcn/publications/SBES2004_Cole_Piveta_Sampaio.pdf)

[6]

HRONEK, Jiří. *Informační systémy* [online]. Univerzita Palackého. Olomouc, 2007 [cit. 2011-12-22]. Dostupné z: <http://phoenix.inf.upol.cz/esf/ucebni/infoSys.pdf>

[7]

IBM CORP. *Rational Unified Process: RUP for Small Projects* [online]. 7.5.1. IBM, 2006 [cit. 2011-12-22]. Dostupné z: <https://kitscm.vse.cz/RUP/SmallProjects/>

[8]

KADLEC, Václav. *Živě.cz: Rational Unified Process: základní pojmy*. 5.8.2003. [cit. 2011-12-22]. Dostupné z: <http://www.zive.cz/Clanky/Rational-Unified-Process-zakladni-pojmy/sc-3-a-113011/default.aspx>

[9]

KRUCHTEN, Philippe. *The Rational Unified Process An Introduction, Second Edition*. Addison Wesley, March 14, 2000. Second Edition. ISBN 0-201-70710-1.

[10]

PENDER, Tom. *UML Bible*. 1.vyd. Indianapolis: Wiley Publishing, Inc., 2003. ISBN 0-7645-2604-9.

- [11]  
PÉRAIRE, Cécile , Mike EDWARDS, Angelo FERNANDES, Enrico MANCIN a Kathy CARROLL. *The IBM Rational Unified Process for System z* [online]. IBM Redbooks, 2007 [cit. 2011-12-22]. Dostupné z: [ibm.com/redbooks](http://ibm.com/redbooks)
- [12]  
PIVETA , Eduardo Kessler a Augusto Jun DEVEGILI. Aspects in the Rational Unified Process' Analysis and Design Workflow. [online]. Universidade Luterana do Brasils. 7 [cit. 2011-12-22]. Dostupné z: <http://mypages.iit.edu/~akkawif/workshops/aosd/09-piveta.pdf>
- [13]  
ŘEPA, Václav, Václav SYNÁČEK, Petr HAMERNÍK a Ondřej DIVIŠ. a kol. *Metodika vývoje informačního systému s pomocí nástroje Power Designer* [online]. Praha, červen 2006 [cit. 2011-12-22]. verze 1. Dostupné z: [http://opensoul.iquest.cz/forum/docs/publications/Metodika\\_vyvoje\\_IS\\_06\\_2006.pdf](http://opensoul.iquest.cz/forum/docs/publications/Metodika_vyvoje_IS_06_2006.pdf)
- [14]  
VOJTEK, David . *Úvod do GIT a Základy geoinformatiky* [online]. VŠB-TU Ostrava, 14.12.2009 [cit. 2011-12-22]. Dostupné z: [http://gis.vsb.cz/vojtek/index.php?page=git\\_c/cviceni06](http://gis.vsb.cz/vojtek/index.php?page=git_c/cviceni06)
- [15]  
VONDRÁK, Ivo. *Úvod do softwarového inženýrství* [online]. Ostrava, 2002 [cit. 2011-12-22]. verze 1.1. Dostupné z: [http://vondrak.cs.vsb.cz/download/Uvod\\_do\\_softwaroveho\\_inzenyrstvi.pdf](http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf)
- [16]  
Běhové prostředí. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-12-22]. Dostupné z WWW: <[http://cs.wikipedia.org/wiki/B%C4%9Bhov%C3%A9\\_prost%C5%99ed%C3%AD](http://cs.wikipedia.org/wiki/B%C4%9Bhov%C3%A9_prost%C5%99ed%C3%AD)>.
- [17]  
Distribučované řídicí systémy. 2005 [cit. 2011-12-22]. Dostupné z: <http://dce.felk.cvut.cz/drs/>
- [18]  
Mapování procesů. In: *Procesní model systému managementu jakosti*. Dostupné z: <http://www.komora-khk.cz/business/documents/?soubor=moduly/5-jakost/06-procesni-model-systemu-managementu-jakosti/06-01-mapovani-procesu.pdf>
- [19]  
*Rational Unified Process* [online]. 2002.05.01.01. Rational Software Corporation, 1987 - 2001 [cit. 2011-12-22]. Dostupné z: <http://www.ts.mah.se/RUP/RationalUnifiedProcess/index.htm>
- [20]  
*The analysis model* [online]. IBM Corporation, 2004 - 2005 [cit. 2011-12-22]. Dostupné z: <http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.ibm.rsa.nav.doc/topics/canalysismodel.html>