RUP disciplína
Analysis & Design

Bc. Martin ROMAN

4IT421 Zlepšování procesů budování IS

27.11.2011

Obsah

1. Popis disciplíny Analysis & Design 3
2. Hlavné role v disciplíne Analysis & Design a ich popis 4
2.1. Softwarový architekt 4
2.2. Návrhár 4
2.3. Návrhár databáze 4
2.4. Revízor architektúry 4
2.5. Revízor návrhu 4
3. Popis činností ktoré sa vykonávajú počas disciplíny Analysis & Design 5
3.1. Inception phase 6
3.2. Elaboration phase 7
3.3. Construction phase 8
3.4. Transition phase 9
4. Výsledne produkty Analysis & Design 9
4.1. Analytický model 9
4.2. Konštrukčný model 11
5. Návody 12
5.1. Úvodný návrh architektúry 12
5.2. Návrh subsystémov 12
5.3. Návrh databáze 13
6. Záver a zhodnotenie 14
7. Použité zdroje a literatúra 15

1. Popis disciplíny Analysis & Design

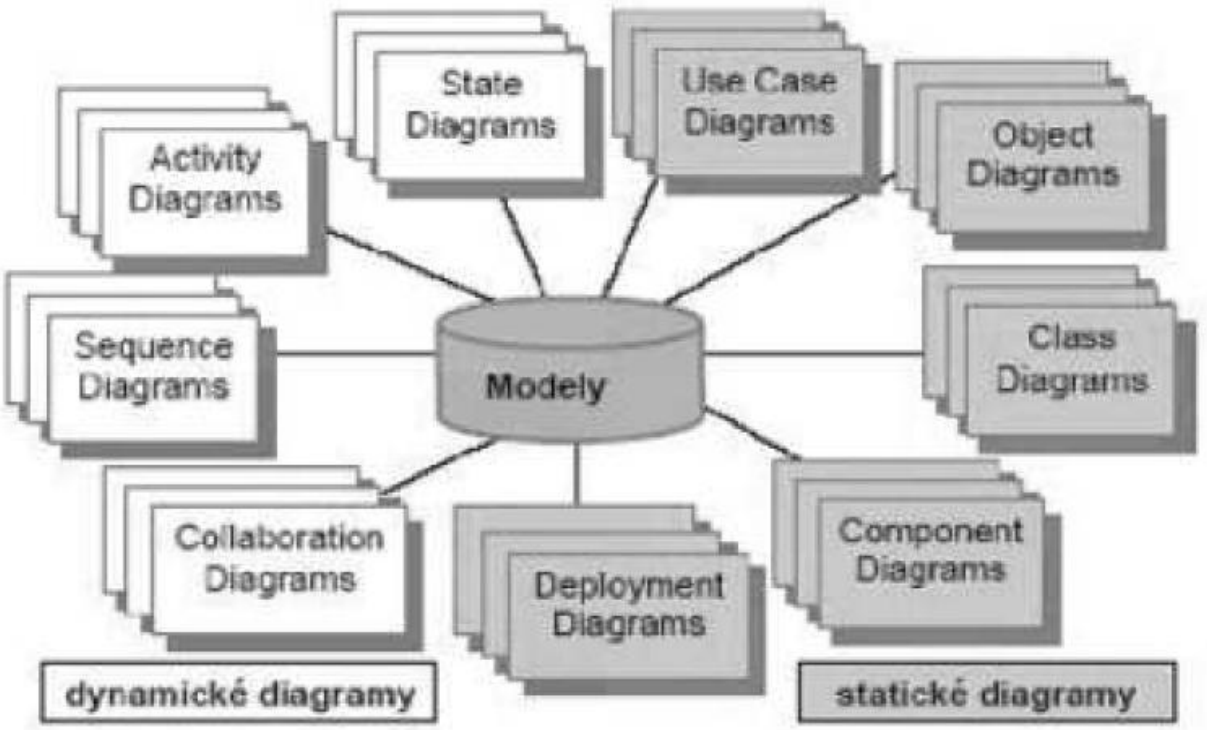
Cieľom fázy analýzy a dizajnu v metodike RUP je ukázať ako bude systém realizovaný v implementačnej fáze projektu.

Vlastnosti ktoré by mal úspešný systém spĺňať:

1. Výkon na konkrétnom implementačnom prostredí pri plnení úloh a funkcií v špecifických use-case prípadoch.
2. Úspešne zvládať všetky požiadavky zákazníka.
3. Vysoko flexibilný pri zmenách požiadaviek (je nutné aby bolo možné jednoduchým spôsobom zmeniť funkcie systému pri zmene požiadaviek)

Analýza a dizajn by vo výsledku mala mať dva výstupy a to konštrukčný model a analytický model. Konštrukčný model by mal slúžiť ako abstrakcia zdrojového kódu, to znamená že by mal byť návrhom toho ako bude zdrojový kód štruktúrovaný a napísaný.

Konštrukčný model pozostáva a z návrhu tried ktoré spolu vytvárajú dizajnové balíčky a ti sa potom zhlukujú do subsystémov s definovanými intrefacemi ktoré predstavujú, čo presne bude súčasťou realizácie systému. Tento model súčasne popisuje to, ako budú medzi sebou jednotlivé objekty navrhovaných tried spolupracovať pri vykonávaní špecifických use-case prípadoch. [i]



Obrázok 1 - Typy diagramov Zdroj: (http://a-web.mysteria.cz/upload/storage/VIS.pdf)

2. Hlavné role v disciplíne Analysis & Design a ich popis

Táto kapitola obsahuje prehľad osôb podieľajúcich sa na vývoji softvéru. U niektorých (najmä malých) projektov nemusia byť niektoré tu uvedené role vôbec zastúpené alebo že jeden pracovník súčasne pôsobí v niekoľkých rolách. Nie je tiež neobvyklé, že jedna úloha je zastúpená väčším počtom osôb (typické napr. u implementátorov).

2.1. Softwarový architekt

V priebehu celého projektu riadi a koordinuje všetky činnosti súvisiace s technológiou. Jeho úlohou je nadefinovať štruktúru jednotlivých pohľadov v rámci architektúry (tj. dekompozíciu v rámci príslušného pohľadu a vymedzenie rozhraní jednotlivých častí). Architekt by mal mať podobnú kvalifikáciu ako návrhár, na rozdiel od neho však musí mať prehľad o celom systéme bez potreby detailných znalostí jednotlivých častí.

2.2. Návrhár

Zodpovedá za popis prípadov použitia systému pomocou modelu tried, vymedzenie jednotlivých postupov pomocou dynamických modelov a určenie spôsobu ich implementácie.

2.3. Návrhár databáze

Rieši problematiku ukladania dát a práca s perzistentnými objektmi. Navrhuje databázy, štruktúru tabuliek, spôsob indexovania a výstupom jeho činnosti je dátový model.

2.4. Revízor architektúry

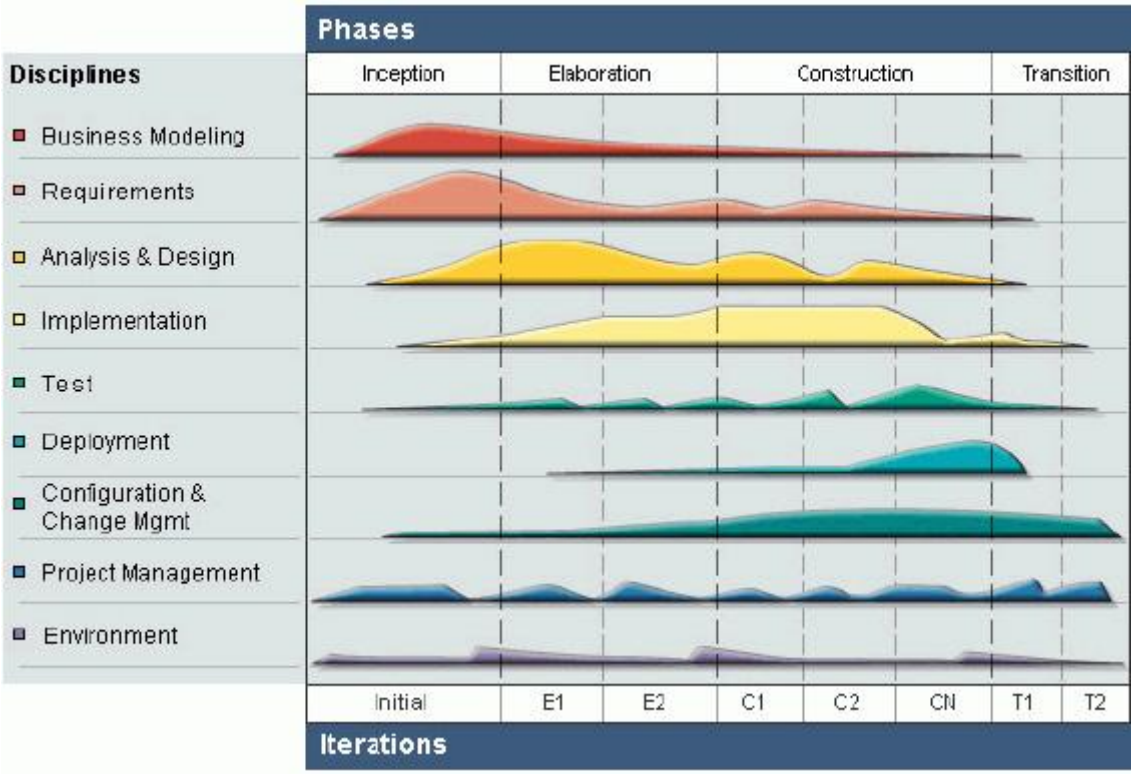
Kontroluje správnosť návrhu architektúry systému.

2.5. Revízor návrhu

Má za úlohu overiť celkovú správnosť modelu návrhu. Na rozdiel od návrhárov, ktorí musia dokonale poznať len určitú časť systému, musí revízor mať revízor návrhu prehľad o celom systéme; príliš detailné znalosti nie sú potrebné.[ii]

3. Popis činností ktoré sa vykonávajú počas disciplíny Analysis & Design

Životný cyklus projektu je v metodike Rational Unified Process rozdelený do štyroch základných fáz (pozri obrázok 4), z ktorých každá je zakončená tzv. míľnikom. Po dokončení každej fázy sa vykoná hodnotenie, či boli splnené požadované ciele. Ďalšej fáze projektu je možné začať iba v prípade splnenia všetkých požadovaných kritérií. Pri disciplíne je Analysis & Design je najviac zdrojov využívaných práve vo fáze Elaboration a na začiatku fázy Construction.



Obrázok 2 - Fázy a disciplíny vývoja podľa metodiky RUP

Zdroj: (http://www.ibm.com/developerworks/library/ws-soa-term2/)

Vývoj produktu zvyčajne nie je definitívne ukončený uvedením prvej verzie do prevádzky. Ak softvér definitívne "neodumrie", vytvárajú sa na základe novo vznikajúcich potrieb jeho ďalšie verzie. Vývoj nasledujúce generácie produktu sa z pohľadu riadenia projektu skladá opäť zo štyroch fáz. U ďalších vývojových cyklov sa ale spravidla podstatne skracuje dĺžka prvých dvoch fáz, pretože analytické činnosti boli už z väčšej časti vykonané u predchádzajúcich verzií.[iii]

Analýza a dizajn je úzko napojená na ostatné disciplíny metodiky RUP:

- The Requirements discipline – slúži ako primárny zdroj pre analýzu a dizajn
- The Implementation discipline –implementuje to čo bolo nadizajnované
- The Test discipline – má za úlohu otestovať dizajn systému
- The Environment discipline - udržiava a rozvíja podporné artefakty, ktoré sa používajú pri analýze a dizajne
- The Project Management discipline – plánuje projekt a každú iteraci[iv]

3.1. Inception phase

V tejto fáze projektu je hlavnou úlohou týmu vymedziť spolu so zákazníkom presné požiadavky ktoré ma systém spĺňať a zároveň vymedziť rozsah projektu. Rovnako je potrebné v tejto fáze vytvoriť základne use-case modely pre dôležité činnosti. Táto časť sa môže líšiť s ohľadom na to či sa vytvára úplne nový systém alebo sa len upravuje už používaný systém. Pri druhej možnosti je samozrejme táto fáza menej náročná, keďže niektoré modely by už mali byť vytvorené.

Hlavné činnosti:

- *Tvorba podnikového modelu* - Základná charakteristika podniku . Analytik podnikových procesov zobrazí pomocou use-case modelov procesy, ktoré sa týkajú vytváraného systému.

- *Analýza problémovej oblasti* - Na tomto kroku spolupracuje systémový analytik so zadávateľom. Jeho cieľom je vymedziť presné určenie systému a jeho rozsah. Rozhodujúcim faktorom sú požiadavky užívateľov, už teraz je však potrebné brať do úvahy obmedzenia dané prostredím, do ktorého budeme nový systém nasadzovať (napr. platformy, rozhrania na existujúce systémy).

- *Identifikácia požiadaviek zadávateľa* - Na základe konzultácií so zadávateľom sa spíše zoznam funkcií, ktoré by systém mal podporovať. Odporúča sa priebežne vytvárať spoločný slovník používaných pojmov, ktorý uľahčí tvorbu modelov prípadov použitia a udržanie ich konzistencie. V tomto kroku je už možné zostaviť úvodné use-case diagramy systému.

- *Plánovanie vývoja software.* V tomto štádiu by už mala byť jasne formulovaná vízia systému a určená priorita medzi jednotlivými funkciami systému. Vedúci projektu na základe poslednej verzie zoznamu use-case diagramov (seřazeného podľa priorít) a rizík aktualizuje plán vývoja software.

- *Ukončenie a zhodnotenie iterácie.* Rozsah prác, ktoré je nutné vykonať pred začatím ďalšej fázy projektu, závisí predovšetkým na tom, ako vedúci projektu zhodnotí možné problémy (napr. ak je budovaný úplne nový systém, pracovníci dodávateľa majú v tejto oblasti málo skúseností a pod). Ak je projekt považovaný za málo rizikový, stačí pre dokončenie jeho úvodnej fázy iba spresniť niektoré požiadavky zadania. V opačnom prípade je potrebné naplánovať ešte ďalšie iterácie, pri ktorých sa podrobnejšie analyzujú problémové oblasti, vytvorí detailnejšie use-case diagramy a ďalej rozpracuje úvodný návrh architektúry.

Vstupy:

- Pôvodná vízia zákazníka
- Existujúci systém (ak existuje)
- Úvodná požiadavky na systém

Výstupy :

- Spoločná vízia zákazník a dodávateľa

- Use case diagramy
- Východiskové verzia ekonomického plánu projektu
- Plán vývoja softvéru - vrátane odhadu potrebných zdrojov. Tieto údaje sa ešte môžu v neskorších iteráciách spresňovať.
- U zložitejších projektov tiež úvodný návrh architektúry (kapitola 5.1) [ii]

3.2. Elaboration phase

V tejto fáze projektu sa vychádza z výstupov Inception phase, to znamená, že sa vytvára model architektúry systémov pre ktorý je základ zoznam základných požiadaviek od zákazníka. Model môže vzniknúť jeden ale aj viacero, to záleží od potreby a náročnosti projektu. Zároveň je potrebné mať k depozícií plán vývoja ktorý bol rovnako vytvorený v predchádzajúcej fáze.

Hlavné činnosti:

- *Príprava prostredia na iterácii.* V tomto štádiu je potrebné nakonfigurovať vývojové nástroje a vytvoriť šablóny dokumentov pre analýzu a návrh.

- *Identifikácia kľúčových prípadov použitia, stanovenie priorít.* Softvérový architekt a vedúci projektu určí, ktoré use-case diagramy sa v tejto iterácii budú spracovávať. Tieto use-case budú kľúčové pre návrh architektúry. Vedúci projektu následne doplní Plán iterácie a vykoná sa detailnejšie spracovanie zvolených use-case diagramy.

- *Spresnenie návrhu architektúry.* Na základe use-case diagramov, spoločného slovníka a skúseností z praxe v danom odbore navrhne softvérový architekt spôsob rozdelenia na subsystémy a identifikuje kľúčové triedy. V tejto fáze sa takisto zohľadnia prvky súčasného systému, ktoré sa budú s novým systémom integrovať, vrátane ich rozhraní. Návrhári potom identifikujú ďalšie triedy a vykonajú spresnenie ich definície (určí ich atribúty a vzájomné väzby). Triedy, ktoré majú vplyv na architektúru systému, zapracuje systémový architekt do dokumentu "Architektúra systému".

- *Návrh subsystémov.* Softvérový architekt spresní definíciu architektúry pre potreby zvoleného implementačného prostredia (programovací jazyk, relačné databázy, distribúcie systému a i.) a nadefinuje subsystémy pre návrh. Tieto sú následne pridelené jednotlivým návrhárom. V tomto kroku možno tiež vykonať identifikáciu kľúčových prvkov dátového modelu.

- *Určenie fyzickej distribúcie systému.* Softvérový architekt v tomto kroku navrhne počiatočnú štruktúru modelu implementácie (tj fyzické členenie systému). Vstupom tohto postupu sú use-case diagramy.

Výsledkom úvodnej iterácia by mal byť prvý funkčný prototyp architektúry, jednotlivé pohľady (prípady použitia, logický, procesné, implementačné, pohľad nasadenie) by mali byť celkom podrobne popísané.

Ďalšie iterácie v tejto fáze slúžia na rozšírenie modelu návrhu a implementácie o ďalšie prípady použitia (na základe stanovených priorít). Počet týchto iterácií závisí okrem iného od rozsahu projektu a skúsenostiach dodávateľov s danou oblasťou.

Vstupy

- Spoločná vízia zákazník a dodávateľa
- Plán vývoja softvéru
- Use case diagramy
- Úvodný návrh architektúry

Výstupy

- Jeden alebo viac prototypov architektúry softvér.
- Model prípadov použitia (aspoň z 80% kompletná). Boli identifikovaní všetci aktéri a všetky kľúčové prípady použitia, k väčšine prípadov použitia existuje popis.
- Model návrhu: vykonaný návrh aspoň pre 10% prípadov použitia.
- Dátový model: boli identifikované kľúčové prvky dátového modelu (tabuľky, relácie).

Na konci fázy prípravy nastáva druhý míľnik projektu. Projekt by v tomto štádiu mal spĺňať nasledujúce podmienky:

- Vízia produktu je už nemenná.
- Bolo identifikovaných viac ako 80% všetkých prípadov použitia, aspoň polovica z tohto počtu bola už analyzovaná.
- Návrh a implementácia boli vykonané aspoň u 10% prípadov použitia.
- Návrh architektúry je už konečný, nebude dochádzať k zásadným zmenám.
- Obstarávatelia súhlasí s realizáciou software s použitím navrhovanej architektúry a plánu vývoja zhodujú sa na tom, že takýto softvér splní požiadavky vytýčené vo vízii. [ii]

Ak nie sú splnené vyššie uvedené podmienky je potrebná ďalšia iterácia.

3.3. Construction phase

Fáza Construction phase je venovaná z veľkej časti implementácii samotného systému, jeho nasadeniu a otestovaniu. Analýza a dizajn je v tejto časti projektu už menej výrazná ako v prechádzajúcich dvoch fázach. Samotná implementácia je ale závislá na výstupoch z minulých fáz, pretože sa tím vývojárov musí správne rozdeliť k vývoju jednotlivých subsystémov a systémových oblasti. Zmeny požiadaviek by sa už nemali meniť a ak zákazník požiada o zmenu tak táto je zapracovaná do dodatočných požiadaviek a je potrebná zmena nákladov aj zdrojov. Preto je vhodné mať všetky požiadavky kompletné už v Elaboration phase.

Hlavné činnosti:

- *Návrh subsystémov.* Návrhári spresní definícia vybraných tried / modulov určených v predchádzajúcich iteráciách (napr. doplní niektoré atribúty). Počas implementácie sa môžu objaviť nové skutočnosti, na základe ktorých sa vykoná úprava modelu návrhu.

- *Integrácia subsystémov*. Z komponentov vytvorených viacerými implementátormi sa na základe plánu integrácie zostaví subsystém.

Vstupy

- Architektonický model systému
- Modely prípadov použitia
- Model návrhu
- Dátový model

Výstupy

- Model návrhu: po dokončení tejto fázy by mali byť identifikované už všetky požiadavky a zapracované do modelu návrhu.
- Dátový model: úplná špecifikácia dátového modelu (tabuľky, relácie, indexy ...). [ii]

3.4. Transition phase

Táto fáza je určená k predaniu hotového systému zákazníkovi a zároveň k testovaniu samotného systému. Podľa pripomienok a testovania je možné systém odladiť ale žiadne zásadne zmeny by kvôli vysokým nákladom už nemali byť vykonávané.

Hlavné činnosti:

- *Správa požiadaviek* - V tejto fáze sa už predpokladá, že požiadavky budú takmer nemenné, nebude dochádzať k ad hoc úpravám produktu. Môžu sa ale vyskytnúť zmeny, ktoré bude treba do systému ešte zapracovať. Analýza a návrh zahŕňajú v tejto fáze hlavne drobné úpravy architektúry na základe testovania - ladenie a zmeny fyzickej distribúcie komponentov systému. [ii]

4. Výsledne produkty Analysis & Design

Ako som už na začiatku tejto práce spomenul fáza analýzy a dizajnu by mala mať dva výstupy a to konštrukčný model a analytický model.

4.1. Analytický model

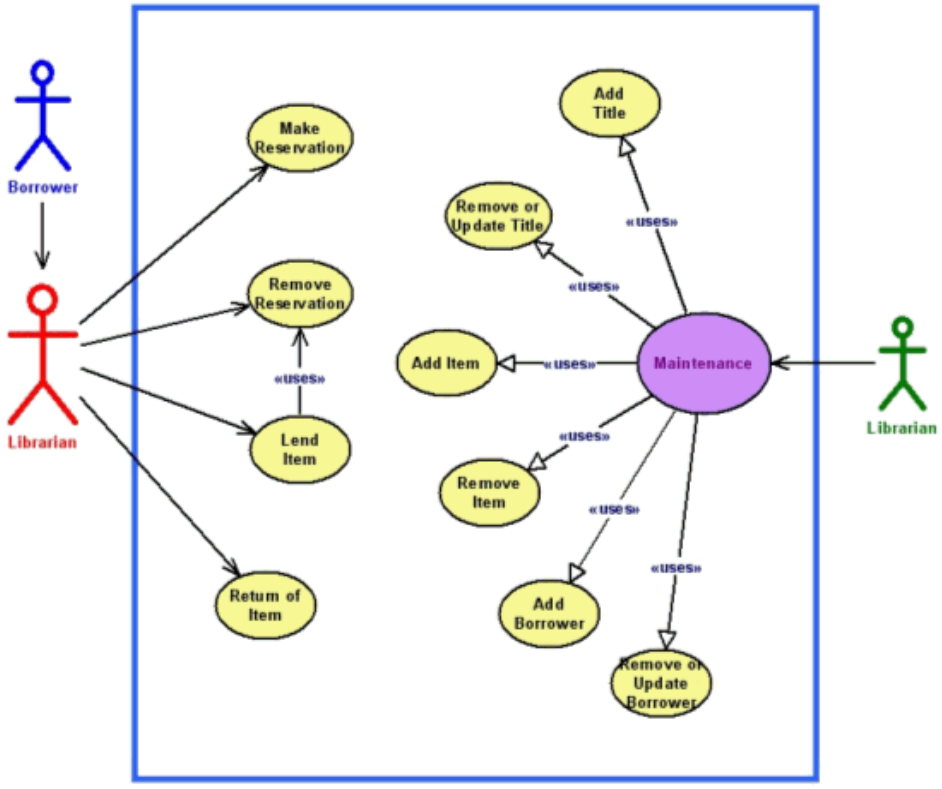
Analytický model obsahuje analýzu tried a všetkých súvisiacich artefaktov. Analytický model môže byť len dočasný a to nastáva v prípade keď sa vyvinie do konštrukčného modelu alebo sa môže prípadne stať, že analytický model sa bude využívať aj neskorších fázach projektu a bude slúžiť ako koncepčný náhľad na systém.

Analytický model je vytváraný v Elaboration fáze a upravovaný v Construction fáze. Zodpovednosť za tento model by mal mať softvérový architekt.

Hlavným cieľom tohto modelu je aby sa požadované správanie správne namapovalo na jednotlivé prvky v systéme. Na rozdiel od toho ma konštrukčný model zmeniť toto zidealizované a predbežné mapovanie do prvkov ktoré môžu byť reálne implementované.

Ďalšie body ktoré stoja za uváženie pri rozhodovaní o detailnosti analytického modelu (môžu sa líšiť od typu projektu)

- Ponechanie analytického modelu môže byť užitočné pri projektoch ktoré sa tikajú rôznych prostredí v ktorých sa budú implementovať. Pretože analytický model ja menej závislý na prostredí ako konštrukčný model.
- Veľa pozornosti treba venovať tomu aby analytický a konštrukčný model zostali v konzistencii. Ak by tieto dva modely boli udržované v príliš odlišnej hĺbke detailov bolo by veľmi nákladné ich udržiavanie. Preto si treba dať pozor na príliš veľkú detailnosťou modelov alebo naopak príliš veľkým nadhľadom.
- Ak sa analytický model prestane udržiavať veľmi rýchlo stráca na svojej hodnote pretože prestane reflektovať súčasný stav systému. Je preto potrebné rozhodnúť sa na začiatku, či bude model udržiavaný alebo nie.[v]



Obrázok 3 - Príklad analytického use-case modelu rezervácii

Zdroj: (http://www.ibm.com/developerworks/rational/library/nov06/temnenco/)

4.2. Konštrukčný model

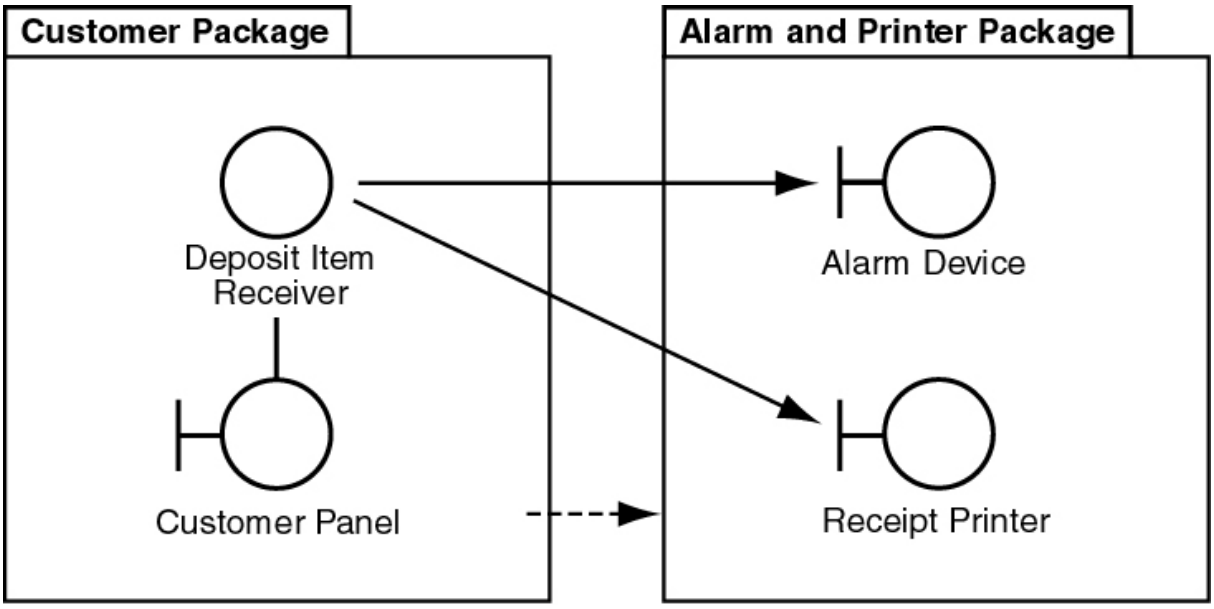
Konštrukčný model je objektový model popisujúci realizácie use-case, a slúži ako abstrakcia zdrojového kódu systému. Je detailnejší a presnejší ako analytický model z ktorého z veľkej časti vychádza.

Konštrukčný model môže byť viac či menej blízky k implementačnému modelu, záleží to do toho akým spôsobom a ako detailne budú namapované triedy, balíčky a subsystémy v implementačnom modely. Počas implementácie sa môžu často vyskytnúť malé problémy, ktoré by však nemali mať vplyv na konštrukčný model. Napríklad triedy alebo subsystémy sa počas implementácie môžu paralelne vyvíjať alebo môžu medzi nimi vznikať nové závislosti.

V každom prípade by ale mala byť zachovaná konzistencia medzi konštrukčným a implementačným modelom. Pravidlá pre toto zachovanie by mal byť definované v tzv. „Design Guidlines“.

Správny konštrukčný model by mal mať nasledujúce charakteristiky:

- Spĺňa požiadavky na systém.
- Je odolný voči zmenám.
- Je vysoko flexibilný vo vzťahu k iným modelom a k implementácii systému.
- Je zrozumiteľný voči implementácii.
- Neobsahuje informácie ktoré sú zahrnuté v kóde programu.
- Je ľahko prispôsobiteľný voči zmenám v požiadavkách.[vi]



Obrázok 4 - Jednoduchý konštrukčný model recyklačného mechanizmu Zdroj: (http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf)

5. Návody

5.1. Úvodný návrh architektúry

Úvodný návrh architektúry by sa mal vytvárať medzi fázami Inception a Elaboration. Je vhodné definovať si konkrétne artefakty, ktoré obsahujú počiatočné posúdenie architektúry. Úvodný návrh nie je určený k tomu aby bol kompletný alebo finálny, skôr zachytáva architektonický kontext a obsahuje pomerne dobre zmapovanú architektúru v jednom alebo dvoch pohľadoch.

Úvodná architektúra by mala byť spracovaná v dokumente alebo v sérii dokumentov ktoré väčšinou obsahujú:

- Stručný prehľad o systéme
- Požiadavky a potreby zákazníka
- Identifikácie a diskusie o kľúčových faktoroch
- Prepracované požiadavky zákazníka v podobe use-case
- Diskusie o kľúčových nefunkcionálnych požiadavkách a ich väzbách na potreby zákazníka
- Konceptuálnu architektúru (používanie stereotypov, diskusiu o komponentoch a zodpovednostiach, dátový model a use-case diagramy)
- Realizáciu architektúry zameranú na procesné pohľady alebo pohľady subsystémov a zobraziť čo najviac dostupných informácii o štruktúre systému. Tiež by mal byť už zobrazený predbežný plán nasadenia.
- Predbežnú implementačnú architektúru. V tomto dokumente sa treba sústrediť na posúdenie hlavných typov technológií.
- Zdôvodnenie všetkých architektonických rozhodnutí. [vii]

5.2. Návrh subsystémov

Návrh subsystémov vzniká z kompletnej analýzy tried. Ak sa trieda nedá zobraziť ako objekt ktorý je závislý iba na sebe je potrebné zaradiť ho do subsystému. Komplexné triedy môžu byť preto zobrazené ako subsystémy.

Subsystémy sú taktiež dobrým prostriedkom na identifikáciu časti systému ktorý môže byť vyvinutý samostatným týmom. Ak sú všetky spolupracujúce prvky zobrazené v balíčku spolu s ich väzbami, tak subsystém môže lepšie zobraziť ich zapuzdrenie ako jednoduchý balíček. Subsystémy sú závisle iba na interfejsoch takže žiadne vonkajšie zmeny ich neovplyvňujú.

Kdekoľvek môžu byť triedy a baličky spolu spojené do subsystémov je návrh jednoduchší, zrozumiteľnejší a flexibilnejší.

Návrh subsystémov by mal byť modelovaný v jazyku UML. Toto modelovanie by malo spĺňať nasledujúce podmienky:

- Zoskupovať triedy na definovanie väčšej granularity časti systémov
- Oddeľovať viditeľne interfejsy od vnútornej implementácie
- Každý návrh subsystémov by mal obsahovať názov a krátky popis [viii]

5.3. Návrh databáze

Databázový model je podmnožinou implementačného modelu ktorý zobrazuje logické a fyzické väzby v systéme. Rovnako obsahuje aj správanie dát v databázy ak sú definované procedúry, triggery a je dodržaná referenčná integrita.

Databázový model je tvorený v Elaboration fáze a upravovaný v Construction fáze.

Dátový model je potrebný ak sa dátová štruktúra nedajú jednoducho odvodiť z tried v konštrukčnom diagrame. Používa sa aj na mapovanie trvalých tried na trvalé dátové štruktúry a zároveň týmto diagramom môžeme tieto trvalé dátové štruktúry zobraziť a definovať. Dátový model sa najčastejšie využíva ak je konštrukčný model objektový a zároveň je trvalé úložisko založené na relačnej databáze.[ix]

6. Záver a zhodnotenie

Analýza a dizajn sú z pohľadu dôležitosti metodiky RUP jednou z najdôležitejších častí celého vývoja systému. Pri nepresnej a nespoľahlivej analýze je vysoko pravdepodobné, že celková implementácia skončí neúspešne pretože úzko nadväzuje na analyzovanie a dizajn a vychádza z ich výstupov.

Naopak pri správnej analýze a dizajne je možné očakávať, že projekty ktoré postupujú podľa metodiky RUP skončia úspešne a v predpokladanom čase a s predpokladanými zdrojmi. Okrem iného je jedným z najdôležitejších a najkritickejších faktorov dobre komunikovať so zadávateľom projektu a porozumieť jeho požiadavkám. Preto je veľmi dôležitým faktom aby role v tejto časti projektu boli zastúpené kvalifikovanými a skúsenými pracovníkmi, ktorý majú skúsenosti nielen z oblasti IT ale aj z oblasti pre ktorú je systém vytváraný.

7. Použité zdroje a literatúra

[i] ***IBM*** *Rational Unified Process*. [Online]. http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[ii] **ALDORFA, F.** *Metodika RUP*. [Online]. Diplomová práce. 2005

[iii] **BUCHALCEVOVÁ, A.** *Metodiky budování informačních systémů*. Praha: Oeconomica. 2009. ISBN 978-80-245-1540-3.

[iv] *RUP for Small Projects.* [Online]. https://kitscm.vse.cz/RUP/SmallProjects/#core.base_rup/disciplines/rup_analysis_design_discipline_29760231.html

[v] **RATIONAL SOFTWARE CORPORATION.** *Artifact: Analysis Model.* [Online]. http://www.ts.mah.se/RUP/RationalUnifiedProcess/process/artifact/ar_amdl.htm

[vi] **RATIONAL SOFTWARE CORPORATION.** *Guidelines: Design Model.* [Online]. http://www.ts.mah.se/RUP/RationalUnifiedProcess/process/modguide/md_desmd.htm#fromAnalysis

[vii] *Initial architectural design.* [Online]. http://www.softwarepractice.org/wiki/Initial_architectural_design

[viii] **IBM.** Guideline: Design Subsystem. [Online]. http://www.wthreex.com/rup/smallprojects/rup/guidances/guidelines/design_subsystem,1.2314266786534317E-305.html

[ix] **RATIONAL SOFTWARE CORPORATION.** *Artifact: Data Model.* [Online]. http://www.ts.mah.se/RUP/RationalUnifiedProcess/process/artifact/ar_datmd.htm