Softvérové inžinierstvo
- disciplína, ktorá sa zaoberá tvorbou rozsiahlych softvérových systémov

Informatika
- zaoberá sa algoritmami, spôsobom práce počítačov a softvérových systémov

Softvérové inžinierstvo
- rieši praktické problémy tvorby softvéru

Softvér
- programy + dokumentácia + konfiguračné dáta

Softvérový systém
- pozostáva z niekoľkých programov, konfiguračných súborov, systémovej dokumentácie (štruktúra) a užívateľskej dokumentácie (použitie systému)

Softvérový produkt
- softvér, ktorý sa dá predať zákazníkovi

Štruktúrovaná analýza a návrh
- Metodológia založená na dekompozícii procesov a diagramov toku dát
- Metodológia (životný cyklus), Dôraz (procesy), Riziko (vysoké), Znovupoužiteľnosť (nízka), Vyzretosť (rozvinutá a rozsiahla), vhodná na dobre definované projekty so stálymi požiadavkami

Objektovo orientovaná analýza a návrh
- analýza a návrh založená na notácii objektov, ktoré zachytávajú dáta a procesy v jednom
- Metodológia (interačná/inkrementálna), Dôraz (objekty), Riziko (nízke), znovupoužiteľnosť (vysoká), vyzretosť (vyvýjajúca sa), vhodná na rizikové projekty s meniacimi sa požiadavkami

Brooksov zákon
- The Mythical Man Month
- Prečo je ťažké vytvárať veľké softvérové projekty
- Málo odladeného kódu za rok
- Ťažké rozdeliť projekt do modulov a zaistiť komunikáciu medzi nimi
- Čas a počet ľudínie sú zameniteľné
- Práca nie je paralelizovateľná– rozdelenie na menšie časti – ladenie a test ťažko paralelizovateľné
- **Ak pridáme ľudí k omeškanému projektu, projekt sa ešte viac omeška**
- Väčšina chýb nie je v kóde ale v návrhu

Práca na projekte
- Plánovanie >Testovanie sys > Testovanie modulov >kódovanie

Problémy tvorbyb softvéru
- Tvorba bez plánu, ciele sa menie v priebehu projektu, spoločný jazyk - UML

Efekt druhého systému
- Prvý pracuje, druhý sa doimplementuje (nevýkonný), tretí je v poriadku

Činnosti softvérového procesu
- Procesy, ktoré napomáhajú zaistiť, že výsledný produkt je správany, kompletný a zrozumiteľný
- Konkrétny návod na vykonanie činnosti
- Výsledky – produkty činnosti

Metodika SW procesu (pracovný postup)
- Sekvencia činností, ktoré napomáhajú pri vývoji finálneho produktu

Životný cyklus – vytvor a oprav
- Vytvor I. Verziu -> modifikuj kým zákazník nie je spokojný -> prevádzka

Životný cyklus – vodopádový
- Analýza – analýza požiadaviek a ich špecifikácia
- Návrh návrh softvérového systému

- Kódovanie – implementácia
- Testovanie – testovanie
- Údržba – údržba

Analýza

- Analýza domény, zber a definícia požiadaviek
- Zoznámenie sa so systémom
- Konzultácie s užívateľmi – zistenie cieľov a služieb systému
- Ciele a požiadavky sú definované v dokumente špecifikácie požiadaviek

Návrh

- Návrh systému a návrh SW
- Rozdelenie požiadaviek na HW a SW, definícia architektúry systému
- Identifikácia a popis zákjladných vzťahov (UML)

Kódovanie a testovanie

- Dizajnované ako moduly, triedy, programy
- Overenie špecifikácii modulov
- Poskladané, otestované a doručené zákazníkovi

Údržba

- Najdlhšia fáza životného cyklu – používanie
- Oprava chýb programu a dizajnu – rozširovanie systému

Vodopádový model

- Klady: plánovanie a kontrola, zavádza pravidlá, dodržiavanie termínov
- Zápory: sekvenčný postup, dlhý čas na vývoj, závisí od analýzy, kontrola kvality až na konci
- Modifikácie
    - inkrementálny – postupné dopĺňanie častí, systém menších vodopádov (koláč)
    - špirálový – vytváranie prototypov produktu, paralelný vývoj všetkých častí (terč)

RUP – Rational Unified Process

- Najlepšie praktiky tvorby SW
- Iterácie – chyby sa odhalie v návrhoch, na konci iterácie spustiteľný kód
- Správa požiadaviek – zisk a dokumentácia požiadaviek zadávateľa
- Komponentová architektúra - C&C, štruktúra prvkov systémuvyužívanie existujúcih a tvorba nových komponentov
- Vizuálne modely
    - UML, vo všetkých fázach projektu, zrozumiteľnosť
    - komunikácia: užívateľ – vývojár, vývojár – vývojár
- overovanie kvality – sledovanie počas celého procesu vývoja
- riadenie zmien – objektívne zmeny v priebehu riešenia – zákony
- **činnosti:** plánovanie ->biznis modelovanie ->špecifikácia požiadaviek -> analýza a návrh-> implementácia -> testovanie -> vyhodnotenie-> nasadenie
- **fázy:** zahájenie (určenie rozsahu systému), rozpracovanie (návrh architektúry systému), tvorba (produkcia, beta verzia), odovzdanie (produkcia, beta, tvorba dokumentácia)

Analýza domény a ŠP

- aké slučby sa od systému požadujú
- aké sú obmedzenia vývoja a výsledného produktu
- chyby v tejto dáze vedú k problémom v implementácii
- čím neskôr problem identifikujeme tým viac času a peňazí bude stáť oprava

Štúdia realizovateľnosti

- **Zisťujeme či IS má pre organizáciu zmysel z ekonomického a používateľkého hľadiska**
- či vieme splniť požiadavky zákazníka pomocou existujúceho HW a SW v medzia rozpočtu
- rýchla a lacná štúdia – výsledok či sa pristúpi k analýza (feasibility report)

Analýza domény a identifikácia požiadaviek

- **Zisťovanie požiadaviek na nový system pozorovaním existujúcich systémov, diskusiou s ptenciálnymi používateľmi a zadávateľom**
- zisťovanie požiadaviek, diskusiou s používateľmi a zákazníkom
- môže zahrňovať vývoj modelov a prototypov
- problémy: nerealistické, pochopenie, individuálne záujmy mng, zmeny požiadaviek

Fázy AD a ŠP

- špecifikácia požiadaviek
    - výsledok – document špecifikácie požiadaviek (DŠP)
    - dve formy
        - zákazník – používateľské požiadavky, vysokoúrovňový popis požiadaviek
        - vývojár – systémové požiadavky, podrobná špecifikácia systému
- Validácia požiadaviek
    - Realistickosť, konzistentnosť, úplnosť
    - Korekcia možných chýb v DŠP
    - Doplnenie nivch požiadaviek do DŠP

Všeobecný model analýzy požiadaviek

- Porozumenie aplikačnej doméne – používatelia, reporty, formuláre, procedúry
- Zber požiadaviek – dotieravosť, nestrannosť, detail, nové pohľady
- Klasifikácia požiadaviek
- Riešenie konfliktov
- Určenie priorít
- Kontrola požiadaviek

Biznis modelovanie

- **Vyjadruje ako popísať víziu organizácie, pre ktorú je system vyvíjan a ako následne túto víziu použiť pri popise procesov, rolí a zodpovednosti**
- Zlepšiť pochopenie medzi zákazníkom a SW inžinierom
- Diagram biznis procesov
    - Diagram biznis prípadov použitia
        - Asociácia, extend, include, boundary, actor, biznis prípad použitia
    - Digram systémových procesov
        - Začiatočný stav, aktivita, rozhodovací blok, podmienka, koncový stav, vetvenie
    - Digram biznis tried

Metódy zberu požiadaviek

- Interview - zber faktov, názorov, emócie, plánovanie (checklist, stretnutie)

- Dotazníky – výber respondentov, reprezentácia všetkých používateľov, vzdialené vykonanie, uzatvorené otázky, spôsoby: náhodný výber, kritériá, vyhovujúci ….
- Pozorovanie – ťažko získať objektívne data (limitovaný čas, počet osôb, ľudia pracujú inak)
- Joint Application Design (JAD)
    - Spája kľúčových používateľov, manažerov a systémových analitikov
    - Cieľ je súčasne zozbierať požiadavky od všetkých kľúčových účasníkov
    - Vedúci stretnutia, používatelia, sponzori, manažéri, tajomník, analytici
- Prototypovanie
    - Rýchly prevod požiadaviek na pracujúci system
    - Keď požiadavky nie sú jasné, vela používateľov, rozsiahly návrh
    - Cieľ je vývoj špecifikácie pre finálny systém

Špecifikácia požiadaviek

- Skutočný system (Biznis modelovanie)
- Model skutočného systému (Špecifikácia požiadaviek)
- Model softvéru (systému)
- **Cieľ je čo má navrhovaný SW system robiť (nie AKO!),** určiť funkcie, vytvoriť zadanie projektu
- Delenie požiadaviek
    - Možnosť prechádzať db objednávok, vhodné prehliadanie dokumentov – **FUNKČNÉ**
    - Automatické spomalenie vlaku ak prejde na červenú – **DOMÉNOVÉ**
    - Max veľkosť systému 4MB, kódovanie, jazyk, … - **MIMOFUNKČNÉ**
- delenie podľa úrovne popisu
    - užívateľská špecifikácia – zrozumiteľné pre používateľa
    - systémová špecifikácia – podrobnejšia pre vývojárov, presná

Diagramy UML

- model prípadov použitia (use case model)
- popis kontextu systému a popis funkčných požiadaviek
- **actor**
    - **prvok okolia modelovaného systému (človek, HW, iný SW system)**
    - používa system, zadáva vstupy, preberá výstupy, neriadi systém
    - udrzuje system (admin), ext. HW (znímač), iné (SW posluprácujúce systémy)
- **prípad použitia (use case)**
    - **základná funkcia systému z vonkajšieho pohľadu (klienta)**
    - Vyjadruje chovanie systému
    - Poskytuje aktorovi výsledok
- Vzťahy
    - asociácia, závislosť
    - generalizácia (zamestnanec [predavač, účtovník], editácia [obrázku, odpovede])

Scenár

- **postupnosť činností v komunikácii aktora so systémom**
- štrukturovaný text, diagram UML
- Schema
    - ID a názov scenára, stručný popis, actor, ktorý inicializuje UC
    - A priori podmienky – vstupné podmienky, musia byť splnené pred začiatkom vykonávania UC
    - Kroky v scenári – činnosti aktora a systému
    - A posteriori podmienky – výstupné podmienky, musia byť splnené po skončení vykonávania UC
    - Actor ktorý dostane výsledok UC
- Kto zadáva? Kde zadáva? Aké údaje?

Validácia požiadaviek

- Metódy validácie požiadaviek
    - Preskúmanie – manuálna kontrola požiadaviek
    - Generovanie testovacích prípadov – tvorba testov požadaviek–časté odhalenie problému
    - Prototypovanie – predvedenie spustiteľného modelu zákazníkovi
    - Automatická analýza konzistencie – ak sú požiadavky vo forme modelu
- Správa požiadaviek
    - **Process riadenie zmien systémových požiadaviek**
- Požiadavky z hľadiska vývoja – trvalé, nestále
- Proces zmeny požiadaviek
    - Analýza problému a špecifikácia zmeny
    - Analýza zmeny a určenie jej ceny
    - Implementácia zmeny
- Sledovateľnosť požiadaviek – matica závislostí požiadaviek – schopnosť sledovať požiadavky

Analýza

- **Proces rozdelenia komplexného problému na menšie časti, za účelom ich lepšieho pochopenia**
- Cieľ je vytvoriť analytický – konceptuálny model
- Zachytenie podstatných požiadaviek a charakteristických rysov systému
- Objektovo orientovaná analýza a návrh
    - Prístup modelujúci system ako skupinu spolupracujúcich objektov
- Objektovo orientovaná analýza
    - Aplikuje techniky objek. modelovania za účelom analýzy funkcionálnych požiadaviek
- Objektovo orientovaný návrh
    - Spracováva analytické modely za účelom vytvorenia špecifikácie implementácia
- Pravidlá tvorby diagramov
    - Tvorený v doménovom jazyku
    - Minimalizácia vzťahov
    - Len prirodzená dedičnosť
    - Čo najjednoduchší model pre max počet používateľov
- Výstupy - Činnosti
    - Architektonická trieda
    - Analýza tried

    - Analýza balíčkov
    - Analýza prípadov použitia
- Vstupy: doménový model, model požiadaviek, model UC, popis architektúry
- **Analytická trieda**
    - **Trieda, ktorá reprezentuje základné data a chovanie a ktorá nezachytáva SW a HW podro.**
    - Hrubá abstrakcia, mapuje jasne definovanú vlastnosť, minimum väzieb
- Metóda stereotypov
    - Hraničné triedy – všetko s čím komunikujú aktéri (rozhrania, formuláre)
    - Entitné triedy – objekty reálneho sveta, system ich udržuje dlhšiu dobu (student)
    - Riadiace triedy – koordinujú správanie systému (nastavujú obsah entitných tried)

Sekvenčný diagram

- **Zobrazuje časovo utriedenú interakciu medzi objektami za účelom vykonania podstatných častí prípadu použitia**
- Vychádza zo scenára
- Aktori komunikujú iba s hraničnými objektami
- Hraničné objekty len s riadiacimi aktormi
- Riadiace objekty so všetkými okrem aktorov

Diagram tried

- **Zachytáva statický pohľad na logickú štruktúru systému modelovanú triedami ich atribútmi, operáciami a vzájomnými vzťahmi**
- Zachovanie prehľadnosti a jednoduchosti bez zanášania implementačných detailov
- Trieda – abstrakcia skupiny podobných objektov
- Pre inštancie definuje – typ, vlastnosti (atribúty), správanie (operácie)
- Asociácia – dlhší vzťah medzi inštanciami
- Agregácia – vzťah celok/časť
- Kompozícia – silnejší typ agregácie, inštancia časti len v jednom celku, ak sa celok zmaže tak sa zmažú aj časti
- Primárne vlastnosti atribútov – najdôležitejšie logické atribúty, primárne operácie
- Generalizácia – zdola nahor – hľadajú sa triedy so spoločnými vlastnosťami → nadtrieda
- Špecializácia – zhora nadol – existujúce triedy zjmňujeme pomocou podtried
- Polymorfizmus – vytvárame asociácie k rodičovským triedam

Analýza balíčkov

- Zoskupovanie tried
- Abstrakcia združovania – je to kontajner a vlastník modelovaných prvkov
- **Univerzálny mechanizmus zoskupovania prvkov a diagramov**
- Definovanie hraníc vnutri modelu, zapuzdrený menný priestor, vnáranie balíčkov

Architektonická analýza

- Zoskupovanie tried do množiny súdržných balíčkov
- Štrukturované do oddielov a vrstiev
- Minimalizácia vzťahov

Realizácia prípadov použitia

- Modelované interakcie medzi objektami
- Popis spolupráce inštancií analytických tried za účelom dosiahnutia požadovaného správania sa systému
- Ciele
    - Zistenie interakcií analytických tried
    - Zisťovanie zasielaných správ (primárne operácie, primárne atribúty, primárne vzťahy)
    - Aktualizácia modelov

Modely tried projektu

- **Doménový model tried – výsledok biznis modelovania**
- **Konceptuálny model tried výsledok analýzy**
- **Implementačný model tried – výsledok návrhu (UML) a implementácie (kód)**

Návrh

- Ciele analýzy
    - Logický model tvoreného systému
    - Analýza požiadaviek z pohľadu problémovej domény
- Ciele návrhu
    - **Presná špecifikácia spôsobov ako to implementovať**
    - Zlúčenie technických riešení (architektúra, GUI, distribúcia, perzistencia objektov)
- Aspekty
    - Kompatibilita
    - Rozširovateľnosť
    - Udržateľnosť
    - Modulárnosť
    - Odolnosť voči chybám
    - Znovupoužiteľnosť
    - Použiteľnosť
    - Bezpečnosť
- Činnosti
    - Návrh architektúry systému (rozdelenie do podsystémov alebo komponentov)
        - Vstupy: model požiadaviek, prípadov použitia, analýzy, popis architektúry
        - Výstupy: podsystém, rozhrania, návrhové triedy, model nasadenia (všetko náčrt)
        - Kroky: podsystémy – vrsty a oddiely – topológia – komunikácia – spôsob riadenia
    - Podrobný návrh systému (každá časť dobre popísaná – podsystém)
        - Vstupy: model požiadaviek, podsystém (náčrt), rozhrania (náčrt)
        - Výstup: podsystém (úplný), rozhrania (úplné)
    - Dôraz na rozhrania
- Podsystém
    - **je množina elementov, ktorá je same systémom a komponentom väčšieho systému**
    - obsahuje aspekty systému s podobnými vlastnosťami
    - hranice volíme taka by komunikácia prebiehla vo vnútri systému
    - identifikujeme podľa služieb, ktoré poskytuje
    - Služba - **množina funkcií, ktoré majú rovnaký základný účel**
- Návrhové modely
    - Podsystémov
    - Návrhových tried
        - Vstup: realizácia UC, návrhová trieda, rozhrania, analytická trieda (náčrty)

        - Výstup: návrhová trieda (úplná), rozhranie (úplné)
    - Rozhraní
    - Návrchových realizácií prípadov použitia
    - Diagramov nasadenia
- Identifikácia paralelizmu
    - Identifikácia podsystémov, ktoré musia a ktoré nesmú pracovať paralelne
- Mechanizmy riadenia
    - Riadené procedurálne
    - Riadené udalosťami
    - Paralelné systémy

Návrh II

- Význam analytického modelu
    - Nové osoby v projekte
    - Porozumenie systému po dlhej dobe
    - Pochopenie systému –uspokojovanie požiadaviek
    - Sledovateľnosť požiadaviek
    - Plánovanie údržby a rozširovania
    - Pochopenie logickej architektúry
- Realizácia prípadov použitia
    - Vstupy: model požiadaviek, prípadov použitia, analýzy, návrhu, nasadenia
    - Výstupy: realizácia UC – návrh, rozhranie, návrhové triedy, podsystém (náčrty)
- Ciele realizácie prípadov použitia
    - Odhaľovanie nových nefunkčných požiadaviek a tried
    - Identifikácia návrhových vzorov
    - Overenie realizácie UC
- Modely tried projektu
    - Doménový model tried (výsledok biznis modelovania)
    - Konceptuálny model tried (výsledok analýzy)
    - Implementačný model tried (výsledok návrhu UML a implementácie (kód)

Implementácia

- **Cieľ je vytvoriť fungujúci SW**
- Produkty implementácie
    - SW component: zdrojový, binárny, vykonateľný súbor
    - UML diagram komponentov: štruktúra a závislosť medzi komponentami
    - UML diagram nasadenia: štruktúra HW, rozmiestnenie SW komponentov
- Preklad: trieda (UML) -> zdrojový kód (programovací jazyk) -> vykonateľný kód (strojový kód)
- Stratégia je postup ako sa realizujú jednotlivé SW súčiastky
- Stratégie
    - testovanie zhora nadol
    - testovanie zdola nahor
    - jednofázové testovanie
    - sendvičové testovanie
    - testovanie porovnávaním
- verifikácia – validácia
    - reálny system – konceptuálny model

    - reálny system -> softvérový model (validácia, robí, čo má?)
    - konceptuálny model -> softvérový model (verifikácia, je bez chýb?)
- akceptácia (užívateľ testuje, či systém spĺňa zadanie
    - správnosť (plni požiadavky)
    - robustnosťodolný voči chybám)
    - výkonnosť (pamäťová a časová náročnosť)
    - dokumentácia (úplnosť a zrozumiteľnosť)
- alfa testovanie : vývojárska firma, užívateľ sledovaný vývojármi, známe prostredie nasadenia
- beta testovanie : užívatelia na svojich počítačoch, rôznorodé prostredie, modifikácia > servis pack

Nasadenie

- inštalácia systému (u zákazníka)
- testovanie po inštalácii (beta)
- plán zálohovania a obnovy
- uvedenie do rutinnej prevádzky

Dokumentácia

- diagramy UML
- zdrojové texty programov
- užívateľská príručka
- CASE – generovanie dokumentácie

Agilné metodiky

- Základné princípy
    - Iteratívny a inkrementálny vývoj
    - Priama osobná komunikácia v time
    - Stale spojenie so zákazníkom
    - Časté testovanie a dôraz na zdrojový kód
    - Pripravenosť reagovať na nepredvídané udalosti

Extrémne programovanie

- Jediným exaktným, jednoznačným, zmerateľným, overiteľným a nezpochybniteľným zdrojom informácií je zdrojový kód
- Účinný, efektívny, ľahký, flexibilný a zábavný spôsob vývoja. Veľmi rozumný
- Premenné (Kvalita, šírka zadania, náklady, čas)
- Hodnoty (komunikácia, jednoduchosť, sptná väzba, odvaha, respect)
- Činnosti (testovanie, písanie zdrojového kódu, počúvanie, návrh – všetko zo všetkým)
- Výhody
    - Práca v súlade s inštinktmi
    - Iteratívna a Inkrementálna
    - Priamy postup k cieľu, bez formalít
    - Podpora v IDE, CASE
- Nevýhody
    - Detail jednoduché, zložité vykonať
    - Ťažké zavedenie, nie vhodné pre každého človeka

SCRUM

- Cieľom tímu je “dotlačenie lopty” na požadovanú pozíciu
- Dopredu nevieme úplne presne čo bude pri vývoji nutné robiť. Zavádza teda každodenné stretnutia ktoré prinesú slabé, zato časté nové myšlienky
- Kľúčové pojmy: flexibilné predmety dodania, flexibilný harmonogram, male tímy, časté revízie, spolupráca, backlog, šprint, riziko, scrum meeing
- Fázy: plánovanie a architektúra, šprint (vývoj, obalenie, zhodnotenie, doladenie), uzávierka
- Výhody
    - Reakcie na zmeny
    - Sloboda voľby riešenia
    - OOP
    - Prepracovaný spôsob odhadu pracnosti
- Nevýhody
    - Skôr súhrn vzorov
    - Ťažké zaveddenie
    - Nie pre každého človeka

Test drive development

- Testovací kód musí byť dokončený ešte pred začiatkom písania testovaného kódu
- Kroky: nový test – spustenie testu – realizácia zmeny – koniec

Crystal Clear

- agilná metodika :)

Testovanie

- SW chyba – softvér nerobí to, čo by podľa špecifikácie produktu mal
- Vznik chýb: Špecifikácia > Návrh > kód > iné
- Dobrý tester
    - Cieľom SW testera je vyhľadávať chyby a čo najskôr zaistiť ich opravu
    - Zvedavý, neúnavný, perfekcionalista, dobrý úsudok, programátorské a doménové znalosti
- Nie je možné otestovať kompletne (vela vstupov a výstupov, špecifikácie je subjektívna)
- Axiómy
    - Nikdy nepreukáže, že chyby neexistujú
    - Čím viac chýb nájdete, tým viac chýb tam je
    - Nie všetky nájdené chyby sa opravia
    - Je ťažké povedať, kedy je chyba chybou
    - Špecifikácia produktu nikdy nie je konečná
    - Testeri nie sú najobľúbenejšími členmi týmu
    - Testovanie je presná technická disciplína