psi - prezentari yt

Introducere

- trb sa armonizam 4 categorii de elemente in procesul de soft dev: resurse umane, modele de afaceri, cerinte tehnice, cerinte economice
- ce provocari putem avea in procesul de soft dev? : comunicare ineficienta in cadrul echipei, planificare defectuoasa a procesului de dezvoltare software, reticenta beneficiarilor sau utilizatorilor, testarea inadecvata a software ului, progresul rapid al tehnologiei, modificarea cerintelor clientilor, limite de timo, resurse infrastruct
- ce solutii avem? crestem nivelul de rigurozitate si disciplina => etapizam, planificam, coordonam, stabilim rezultate(livrabile) => facilitam comunicarea intre membrii echipei de dev => folosim modele => utilizam standarde
- cu materia asta de psi ar trebui sa ne clarificam solutiile de mai sus
- prin modele imi evidentiez aspecte importante si omit detalii irelevante; pot sa le construiesc cu limbaje grafice sau textuale => model care reprezinta ceva
- valoarea modelelor <= sunt rapide, intuitive si e mai simplu sa modific un model decat un cod sursa
- metodologii de dezvoltare; in majoritatea proiectelor o sa mi adopt o metodologie ca sa am un mod de lucru organizat; metodologia presupune un vocabular (imi include tipurile de livrabile care urmeaza sa fie realizate in cadrul procesului), proces (imi arata ce etape am si ce activitati desfasor), reguli si indicatii (astea imi stabilesc calitatea rezultatului si a procesului); eg. metodologii agile precum e SCRUM
- ce putem sa standardizam la un proces, din componentele 3 de mai sus: vocabularul (=notatiile folosite): eg. de notatie comuna folosita in mai multe metodologii: UML (Unified Modeling Language)
- eg de metodologii consacrate bazate pe UML: RUP(Rational Unified Process) folosite pt proiectele de dimensiuni mari; OMT (Object Management Technique); XP(Extrreme Programming): pune accentul pe codificare si incurajeaza pai programming -> sta la baza Agile; discutii frecvente si relatie stransa cu beneficiarii

UML principale caracteristici|:
- =e limbaj de notatii specific sistemelor software
- e un standard dezvoltat si intretinut de OMG (object management group)
- NU e o metodologie
- standardizeaza notatiile dar nu imi zice cum sa le fol
- imi ofera flexibilitate
- a influentat crearea unui alt limbaj numit BPMN (business process model and notation)

Diagramele UML sunt:
1. structurale: de clase, obiecte, pachete, componente, structura compusa, desfasurare, profile
2. comportamentale: cazurilor de utilizare, de activitate, stare, interactiune [eg: de secventa, comunicare, de timp, interactiunilor de ansamblu]

Aceste modele construite prin diagramele UML imi ofera 3 tipuri de perspective asupra unui sistem:
1. statica (eg: clase obiecte structura compusa componente desfasurare)
2. functionala (eg: cazuri de utilizare activitati interactiuni de ansamblu)
3. dinamica (eg: stare secventa comunicare timp)

!!! diagrame particulare: diagr de pachete => imi struct/modularizeaza limbajul & diagr de profile => extinde limbajul

!!Concept important
intrsumentele de tip *CASE = Computer Aided Software Engineering*
eg: visual paradigm => imi ofera suport pt automatizarea scrierii diagramelor UML si BPMN
- ofera suport informatic pt construirea de diagrame, pt a verifica consistenta modelelor si pt a crea legaturi intre acestea

- imi gestioneaza versiunile de modele
- imi genereaza cod sursa pornind de la diagrame
- reverse engineering sau imi genereaza modele pornind de la cod

Identificarea Cerintelor

- e una din primele etape din procesul de dezvoltare a unui sistem informatic
- faza asta e precedata de etapa de planificare (acolo fac studii de fezabilitate si analize cost beneficiu ca sa stabilesc oportunitatea construirii sistemului si cerintele globale ale acestuia)
- e super duper importanta pt ca de calitatea cerintelor depinde succesul proiectelor de soft dev
- aici culeg si documentez cerintele care pot proveni din surse multiple si pot fi culese folosind metode variate
- cu cat descopar mai multe cerinte => cu atat fac un sistem mai apropiat si bun de cerintele reale
- cerintele se pot schimba si trb sa fim preg sa operam modificari si in cadrul modelelor si in cadrul aplicatiei
- uneori, cerintele pe care le identificam sunt deja implementate => trb sa vad inainte de functionalitati am deja implementate, ca sa nu fiu redundanta
- in fct de cum le identific am 2 tipuri de cerinte: implicite (le descopar eu in calitate de analist) si explicite (mi le zice clientul/beneficiarul)
- 2 surse pt identificarea cerintelor: beneficiarul per se & artefactele
- ce sunt/ce contin artefactele? am in ele orice info existente despre sistem care pot fi culese din: documente preexistente legate de functionarea sistemului, esantioane de date, chestionare, prototipuri de interfete, scheme conceptuale
- e ff important sa intelegi organizatia: cu ce se ocupa, care sunt fluxurile de lucru, ce politici a adoptat, care sunt beneficiarii sistemului
- de ce e necesar sa implicam activ beneficiarul cand identificam cerintele? pt ca cerintele se schimba, trb sa obtinem feedback in mod frecvent, rezolvam aspecte neintelese, incercam sa eliminam ambiguitati
- BUNE PRACTICI pt analist/oricine culege cerintele unui sist informatic: foloseste un limbaj accesibil interlocutorului, lasa jargonul tehnic, incearca sa intelegi vocabularul afacerii, be respectful (asta merge pt toti oamenii, asa, in general), poate pers respectiva nu stie sa-mi raspunda la toate intrebarile => caut cai alternative de informare, trb sa ii explici beneficiarului modelele si documentele care trb validate, trb sa-i dai idei si alternative, verifica mereu info culese & last but not least, cand se modifica cerintele => informeaza beneficiarul de costurile acestor schimbari

Analiza Textuala

=ne ajuta sa identificam anumite informatii importante din descrierea textuala a cerintelor unui sistem informatic
- o aplica folosind instrumentul de tip CASE: Visual Paradigm
- pt eg. folosim o descriere succinta a principalelor cerinte ale sist informatic pe care trb sa-l modelam
- prin analiza asta imi identific principalele cazuri de utilizare a sist, principalii actori si pt a crea o diagrama a cazurilor de utilizare

Diagrame UML - Diagrama cazurilor de utilizare

- in limbajul UML, un model e reprezentat grafic printr-o diagrama
- o doagrama e o ilustrare a unei parti a realitatii descrise de model
- aceeasi diagrama poate fi utilizata in mai multe etape ale ciclului de dezvoltare al unui sistem informatic (pornim de la aspecte conceptuale, abstracte => le completam ulterior cu detalii tehnice care stau la baza generarii/scrierii de cod)
- diagr UML ofera info esentiale pt o intelegere completa a sistemului
- relatiile dintre diagrame permit crearea de modele consistente

- ROLUL ***diagramei cazurilor de utilizare*** = specifica functionalitatea de baza a unui sistem software si de a modela ce utilizator al sistemului foloseste functionalitatea; **ne arata cine lucreaza efectiv cu sistemul;** exprima **ce ar trebui sa faca un sistem** dar nu abordeaza niciun detaliu de realizare => **imi exprima asteptarile pe care le are clientul de la sistemul care urmeaza sa fie dezvoltat**
- documenteaza ce cerinte ar trb sa-mi indeplineasca sistemul => esential pt o documentare tehnica detaliata
- o folosim sa raspundem la intrebarile: CE SE DESCRIE? sistemul; CINE INTERACTIONEAZA CU SISTEMUL? actorii; CE POT FACE ACTORII? chestiile astea sunt descrise in cazurile de utilizare
- aici, actorii interactioneaza intotdeauna cu sistemul in contextul cazurilor lor de utilizare, mai precis a cazurilor de utilizare cu care sunt asociati
- actor = rol care are o interactiune cu sistemul si poate fi reprez de un om, o piesa hardware sau un alt sistem informatic
- TIPURI de actori: principali = beneficiarii sistemului (initiaza executia cazurilor de utilizare); secundari = sunt utilizati de sistem si au rolul de a oferi servicii sistemului
- CAZUL DE UTILIZARE descrie o functionalitate asteptata de la sistemul care urm sa fie dezș cupride o serie de functii care se desf atc cand se executa sistemul; modeleaza interactiunile intre actori si sistemul nostru; ofera beneficiu tangibil pt 1 sau mai multi actori care comunica cu acest caz de utilizare; SUNT ORIENTATE PE SCOP; exprima cerintele sistemului
- sunt neutre dpdv tehnologic
- mai are relatii existente intre aceste elemente; avem 3 tipuri: intre actor si use case, intre 2 use case uri, intre 2 actori
- relatia dintre un actor si un use case = asociere = descrie comunicarea care are loc intre actor si sistem in contextul unei anumite functionalitati ale acestuia; = ne arata ca are loc un schimb de info intre actor si sistem
- relatiile dintre 2 use case uri: 3 tipuri: includere, extindere, generalizare; niciodata nu discutam de o relatie simpla de asociere intre 2 use case uri
- relatia de includere: cazul A include B daca executia cazului A duce in mod obligatoriu la executia cazului B (cum e la noi Alege tip de scanare => Generare raport; sau Introdu IP => e valid => Alege tip scanare); B nu se poate executa daca nu l-am executat inainte pe A; spre exp cazul A e de baza; spre exp: atat X cat si Y presupun in mod obligatoriu Validarea a ceva & niciunul nu se poate finaliza daca nu se completeaza validarea & validarea nu se poate declansa daca X sau Y nu se declanseaza inainte; ne ilustreaza un comportament obligatoriu
- relatia de extindere ne ilustreaza un comportament optional;
- relatia de generalizare intre 2 use case uri e asemanatoare cu rel de generalizare intre 2 actori

Diagrama claselor

- descrie elementele unui sistem si relatiile dintre ele
- aceste elemente si relatii nu se modifica de-a lungul timpului: eg. elevii au un nr de inmatriculare si participa la anumite cursuri; nu-si pierd valabilitatea
- cea mai utilizata diagrama UML
- e folosita in diferite etape ale proceului de soft dev
- in faza initiala ajuta la crearea unei viziuni conceptuale a proiectului si definirea vocabularului care urmeaza sa fie folosit
- ulterior rafinam vocabularul prin introducerea unor detalii specifice unui limbaj de programare, pana la nivelul de implementare
- simplitatea sa => o face ideala pt construirea unor schite rapide care sa evidentieze structura sistemului
- cu ea pot sa-mi generez automat codul sursa
- trb sa aiba NumeClasa, -atribute, +operatii(); de asemenea pot sa definesc clase de tipul interfata
- cu stereotipuri imi definesc clase speciale care mi fac modelul mai usor de inteles: eg. entitate control si limita; =>model view control
- alte eg de clase stereotipe: enumerare, primitiva
- am reprezentari grafice distincte: eg. pt formular, control, entitate
- vizibilitatea atributelor: +public, -private, #protected, ~package; / atribut derivat
- daca un atribut accepta mai multe valori: folosesc multiplicitatea atributelor

- cand am spre exp clasa Utilizator, nu ma mai refer la actor, ma refer la datele per se ale Utilizatorului (nume, mail, parola, etc)
- regula nescrisa: cumva am mai multe relatii de asociere decat relatii de compunere; este format din …? la nivel conceptual sau fizic?
- multiplicitatile imi arata cate instante ale unei clase participa la asociere

In etapa de analiza a sistemului informatic NU includem deloc detalii legate de implementare si nu vom avea in minte o anumita tehnologie informatica sau un anumit limbaj de programare cand construim clasele si le identificam. Detaliile acestea vor interveni in etapa de proiectare atunci cand vom realiza diagrama de clase la nivel de proiectare si cand construim structurile de date persistente pe care se bazeaza sistemul nostru.

Diagrama de Activitate

=modele comportamentale care se folosesc atat in faza de analiza cat si in fata de proiectare
- in faza de analiza: prezinta detaliat secventa de actiune aferenta unui scenariu al unui caz de utilizare
- in faza de proiectar: sunt folosite pt a detalia o operatie complexa in vederea implementarii sale corecte
- nu se realizeaza pt fiecare scenariu, ci doar pt scenariile mai complexe identificate in descrierea sistemului
- se prezinta sub forma unor diagrame de flux si descrie fluxul de lucru dintr-un punct de placre pana intr-un punct de terminare, modeland in detaliu caile de decizie care pot aparea intr-o activitate
- poate fi folosita si pt a modela procesarea paralela
- se aseamana cu diagramele de procese din limbajul BPMN
- activitatea e un comportament parametrizat reprezentat sub forma de flux coordonat de actiuni
- actiunea reprezinta un singur pas in cadrul unei activitati; sunt atomice => nu mai pot fi descompuse
- si in cazul actiunilor pot utiliza restrictii, preconditii, postconditii; le fac cu nota atasata
- pt conectarea actiitatilor si actiunilor se folosesc fluxuri, aka arce =. exprima ordinea de executie a acestora; cele mai utiliz sunt fluxurile de control = modul de transfer al controlului de la o actiune la alta; alta categ = flux de obiecte -aici am calificatori (patratelele alea din capetele sagetii) merge sa punem patratelele astea doar pe actiuni, nu si pe activitati
- jetoane = mecanism virtual de coordonare care descrie cu exactitate executia; daca actiunea primeste un jeton =>poate fi executata
- noduri: initiale (bulina plina), finale: 2 tipuri: nod final al activitatii cu cer plin in cerc gol adica se termina fluxurile de control dintr-o diagrama sii nod final al fluxului cu x in cerc =: procesul se opreste in punctul ala

Diagramele de Interactiune

- modeleaza aspectele dinamice ale sistemului
- sunt alcatuite dintr-un set de obiecte si relatiile dintre ele, incluzand si mesajele pe care obiectele le trimit de la unul la altul
- sunt folosite si in etapa de analiza si in etapa de proiectare
- in etapa de analiza: implica actorii, clasele sau subsistemele specificate pana in acel moment
- in etapa de proiectare: avem si diversele clase de interfata si controllerele care sunt implicate in schimburi de mesaje la un nivel mult mai ridicat
- eg. diagrama de secventa, diagrama de comunicare (aka diagr de colaborare in UML1.4) => astea 2 diagr sunt echivalente <=> se pot transf una din alta
- DIAGR DE SECVENTA = diagrama de interact formata din obiecte, mesajele care se schimba intre ob si de dimens temporala reprezentata progresiv pe verticala => imi pun accentul pe ordinea mesajelor in functie de timp
- obiectele se aseaza de la stanga la dreapte
- obiectele exista cat au linia de viata?? (zici ca citim in palma aici)
- exemplu: nume rol = lector : Profesor = clasa

- majoritatea obiectelor exista pe tot parcursul interactiunii => au linia de viata trasata din varful diagramei pana la baza, iar altele le creez pe parcursul interactiunii
- specificarea executiei =. un dreptunghi inalt si subtire care mi indica pe ce perioada de timp face obiectul o actiune
- pot sa reprezint obiectele cu stereotipurile: *actor, boundary, entity & control* => diferentierea asta e utila mai mult in etapa de proiectare; in etapa de analiza urmarim mesajele transmise intre actori si clasa de business
- mesajele au forma de arcuri; le trasam intre obiecte & sunt de mai multe tipuri: de tip apel (call) = mesaj sincron -are linie continua si in capatul sagetii un triunghi plin; de tip raspuns (return) - are linie punctata, poate fi omis daca continutul si locul sunt evidente; mesaj asincron - linie normala si sageata ca la o sageata normala >; mesaj de autoapelare => obiectul isi trimite mesaj singur; mesaj de creare & distrugere (create & destroy) => astea incep si termina linia de viata a unui obiect ssiiii sunt optionale; le fol doar daca imi trb mentionarea explicita a acestor evenimente; grija la mesajul de distrugere, ca daca am obiecte compuse pot sa mi se distruga si alte obiecte inafara de ala pe care intentionez sa l distrug!!!!
- ORDINEA MESAJELOR in diagr de secventa e super imp!!!!!! pt ca ne arata succesiunea lor in timp: avem 3 situatii: 1. pe aceeasi linie de viata => le iau in ordine de sus in jos; 2. pe linii de viata diferite: nu conteaza care cum, ai mai multe variante posibile aici; 3. pe linii de viata diferite, dar care schimba mesaje intre ele: aici le iei tot de sus in jos, dar respectam unde se duc mesajele (daca schimb linia adica)

Limbajul BPMN

- unul din aspectele definitorii in specificarea cerintelor unui sistem informatic e legat de identificare proceselor de afaceri care vor opera in cadrul organizatiei
- proces de afaceri = o multime de activitati intercorelate executate de diferite unitati organizationale care conlucreaza pentru indeplinirea unui obiectiv al organizatiei
- activitatile incluse intr-un proces de afaceri pot fi executate manual de catre factorul uman sau prin intermediul sistemelor informatice
- daca la nivel organizational procesele de afaceri sunt esentiale pentru intelegerea modului de operare al companiilor, acestea joaca un rol imp in proiectarea si realizarea unor sisteme informatice usor adaptabile la schimbari
- BPMN (Business Process Model and Notation) = limbaj vizual specializat care ofera facilitati precum: descrierea proceselor de afaceri intr-o maniera neambigua, vizualizarea proceselor de afaceri pentru intelegerea sistematica a acestora, precum si comunicarea modelelor de proces catre cei care vor dezvolta sistemul in etapele urmatoare
- standard liber; nu propune o anumita metodologie
- Diagramele BPMN grupeaza elementele de notatii in 3 mari categorii: Obiecte de flux, Obiecte de conectare, Artefacte

Obiectele de flux:
- activitate - dreptunghi cu colt rotunjit
- eveniment - cerculet gol <- e element distinct BPMN
- in UML conditie = romb -> in BPMN sn poarta conditionala exclusiva

Obiectele de conectare:
- flux de secventa -> imi indica ordinea realizarii activitatilor
- flux de mesaj - - > arata directia de transmitere a unor mesaje
- asociere - - - pt legarea unor obiecte de tip Data sau Artefacte (2 tipuri: asociere de date . . . . > (asociez un document la o activitate; il fol ca data de intrare sau data de iesire)si asociere . . . . .)

Artefacte = notatii care permit adaugarea de info supl ce nu pot fi modelate altfel:
- adnotare =comentarii sau notitele din UML
- grup =pun impreuna sau delimiteaza anumite zone ale fluxului ce apartin unui departament functional sau unui subproces
- artefacte personalizate - in fct de dorintele celor care modeleaza procesul

Date -folosim simbolul de pagina
- Obiecte de date - pagina goala
- Date de intrare - pagina cu sageata goala
- Date de iesire - pagina cu sageata plina
- Date stocate - un cilindru mai micut (fisiere sau baze de date)

Putem folosi grupari ale actiunilor in functie de participantii la actiune(ca la UML la swinglang ceva de genu parca) => imi definesc un container cu culoare; fol un container pt un proces

pt fiecare din tipurile de obiecte de mai sus avem simboluri mai detaliate in materialul ei (vezi pdf Seminar BPMN 1)

!!!cea mai mare DIFERENTA DINTRE BPMN si UML = suportul oferit pt reprezentarea evenimentelor;
evenimentul e ceva ce se intampla in timpul unui proces de afaceri
eg.: un apel telefonic la fiecare 10 min, trimite un mesaj, a avut loc o eroare
In functie de tipul de trigger putem avea evenimente de tip diferit: tip mesaj, timp, semnal, eroare/ exceptie, conditional (modeleaza aparitia unor conditii complexe), escaladare (escaladarea anumitor situatii exceptionale)

eveniment de inceput - cerc gol simplu linie simpla contur
eveniment intermediar - cerc gol contur dublu linii simple
eveniment de sfarsit - cerc gol contur boldat cat 2 linii si plin asa

cand am un simbol in cerculet:
contur cerc simplu - eveniment de inceput care receptioneaza (un mesaj spre exp)
contur cerc dublu linii simple - eveniment intermediar care receptioneaza
contur cerc dublu linii simple iconita din cerc colorata - eveniment intermediar care trimite
contur cerc dublu linii boldate colorate intre ele iconita din cerc colorata - eveniment final care trimite
contur cerc linie simpla punctata iconita goala - eveniment de inceput care receptioneaza fara a intrerupe o alta activitate
contur cerc dublu linii punctate iconita goala - eveniment intermediar care receptioneaza fara a intrerupe o alta activitate

Porti exclusive = elemnete de modelare folosite pentru a controla modul in care fluxurile de secvente interactioneaza pe masura ce acestea se unesc/se despart in cadrul unui proces
trb sa i dau nume portii
most used - exclusiva : rombu'i gol sau are un X in el; trb sa am minim 2 linii care ies din el
inclusiva are cerc in ea; liniile care ies din poarta asta pot sa aiba loc in acelasi timp, nu se exclud intre ele; cele 2 fluxuri de reunesc la un mom dat
complexa are o * inauntru; cand conditiile nu pot fi modelate cu ajutorul notatiilor standard
paralela are un +; cu asta sincronizez sau combina fluxuri paralele sau pt a desemna inceputul unor fluxuri paralele; se reprezinta activitatile concurente; bifurcatie & jonctiune
porti bazate pe evenimente: fol pt a modela variante de continuare ale fluxului de proces care sunt determinate de aparitia diferitelor tipuri de evenimente; ce iese din rombul cu eveniment intermediar in el avem intotdeauna cate un eveniment intermediar. practic, decizia e luata de un alt participant pe baza unor date care nu sunt accesibile procesului analizat

Containerele comunica intre ele cu ajutorul fluxurilor de mesaje; NU putem avea fluxuri de secventa intre elemente aflate in containere diferite