Curs 3

Colectarea cerinţelor

Suport de curs bazat pe ***B. Bruegge and A.H. Dutoit***

"Object-Oriented Software Engineering using UML, Patterns, and Java"

Sumar Curs 3

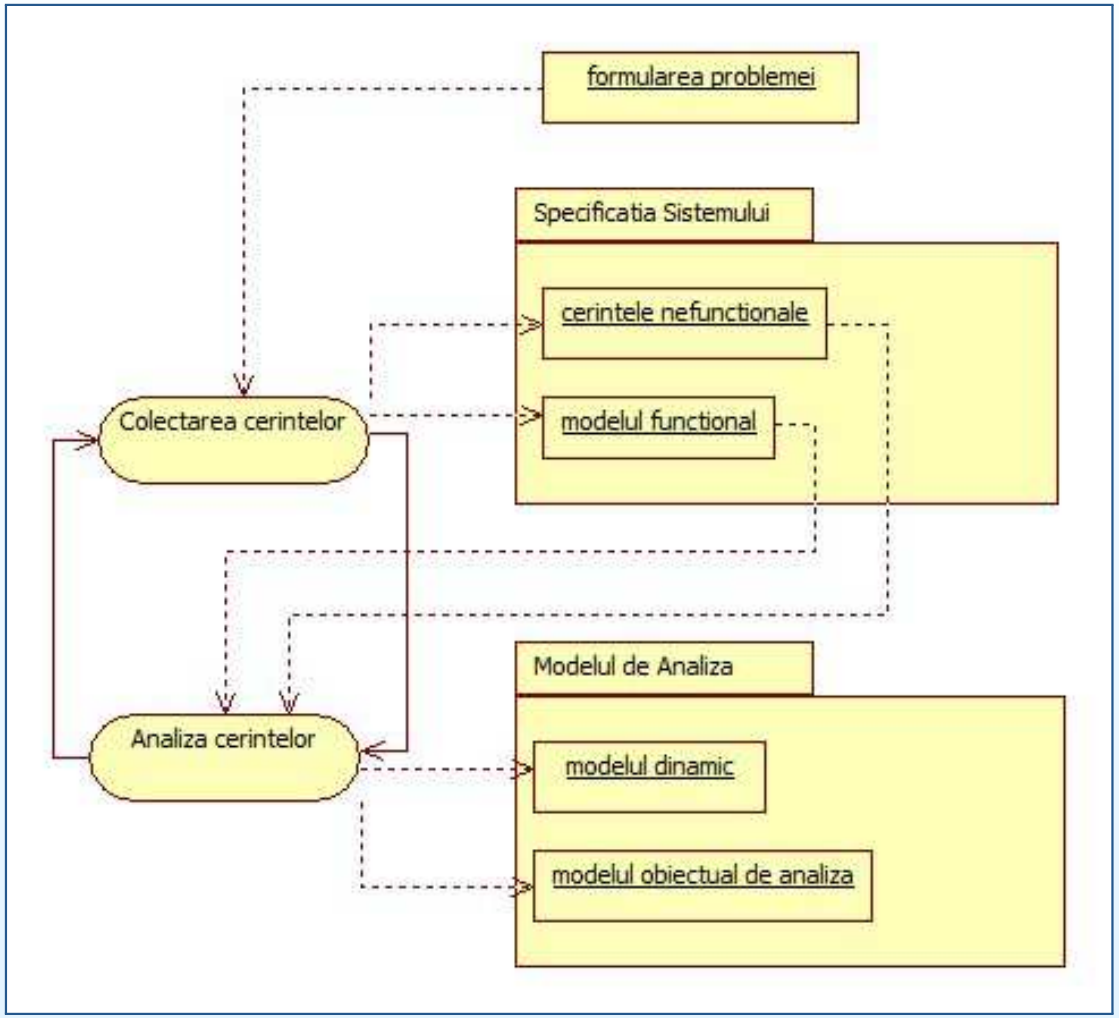
- Cerinţe. Ingineria cerinţelor
- Colectarea cerinţelor - concepte
- Colectarea cerinţelor - activităţi tehnice
- Colectarea cerinţelor - management

Cerinţe. Ingineria cerinţelor

- O *cerinţă* (eng. *requirement*) reprezintă un element de funcţionalitate pe care sistemul trebuie să îl ofere sau o constrângere pe care trebuie să o îndeplinească
- *Ingineria cerinţelor* (eng. *requirements engineering*) este un subdomeniu al ingineriei softului, având drept scop definirea cerinţelor sistemelor soft ce urmează a fi construite
- Activităţi ale ingineriei cerinţelor
  - *Colectarea cerinţelor* (eng. *requirements elicitation*) $\Longrightarrow$ *specificaţia sistemului* (contract între client şi dezvoltatori)
  - *Analiza cerinţelor* (eng. *analysis*) $\Longrightarrow$ *modelul de analiză*
- Atât specificaţia sistemului, cât şi modelul de analiză reflectă aceeaşi informaţie, ele diferă însă prin rolul lor şi notaţia utilizată
  - Specificaţia sistemului este exprimată în limbaj natural şi serveşte ca şi instrument de comunicare cu clienţii/utilizatorii
  - Modelul de analiză este exprimat într-o notaţie semiformală sau formală şi serveşte ca şi instrument de comunicare între dezvoltatori

Cerinţe. Ingineria cerinţelor (cont.)

- Activităţi şi produse ale ingineriei cerinţelor (diagramă UML de activităţi)
  - Activităţile ingineriei cerinţelor se focusează exclusiv pe aspectele externe ale sistemului (perspectiva utilizatorilor externi asupra sistemului)
  - Modelele lor reprezentând aceleaşi aspecte, cele două activităţi se desfăşoară concurent şi iterativ



Colectarea cerinţelor

- Colectarea cerinţelor necesită colaborarea între grupuri de participanţi cu background diferit (clienţi/utilizatori - experţi în domeniul problemei vs. dezvoltatori - experţi în dezvoltare de soft)
  - Corectarea erorilor de comunicare introduse acum (rezultând în specificarea greşită a unor funcţionalităţi, funcţionalităţi lipsă, probleme la nivelul interfeţelor utilizator) în etapele ulterioare ale dezvoltării este deosebit de costisitoare în termeni de timp/buget
  - Ca urmare, metodele de colectare a cerinţelor au drept obiectiv îmbunătăţirea comunicării între aceste grupuri
    - Ex.:*Colectarea cerinţelor bazată pe scenarii* (eng. *scenario-based requirements elicitation*)
      - *Scenariu* = descriere a unui exemplu concret de utilizare a sistemului, în termenii unei secvenţe de interacţiuni între utilizator şi sistem
      - *Caz de utilizare* = abstractizare care descrie elementele comune ale unei clase de scenarii

Colectarea cerinţelor bazată pe scenarii

- Dezvoltatorii colectează cerinţele prin observarea şi chestionarea utilizatorilor în mediul lor
- Se realizează două tipuri de scenarii
  - Iniţial - scenarii descriind modul curent de desfăşurare a proceselor de lucru în mediul utilizatorilor
  - Ulterior - scenarii prescriptive, ilustrând funcţionalităţile care urmează a fi oferite de către noul sistem
- Scenariile realizate sunt validate de către clienţi şi utilizatori prin inspectare şi prin manipularea unor prototipuri ale interfeţei grafice cu utilizatorul, oferite de către dezvoltatori
- Pe măsură ce definiţia sistemului se stabilizează, dezvoltatorii şi clientul convin asupra unei specificaţii a sistemului constând în descrierea cerinţelor funcţionale, a celor nefuncţionale, a cazurilor de utilizare şi a scenariilor aferente

Activităţi ale colectării cerinţelor bazate pe scenarii

- *Identificarea actorilor*
  - Dezvoltatorii identifică diferitele tipuri de utilizatori a căror activitate va fi susţinută de către viitorul sistem
- *Identificarea scenariilor*
  - Dezvoltatorii observă utilizatorii în mediul lor şi descriu un set de scenarii detaliate aferente funcţionalităţilor tipice oferite de către noul sistem
  - Scenariile sunt folosite pentru comunicarea cu utilizatorii şi aprofundarea înţelegerii domeniului problemei de către dezvoltatori
- *Identificarea cazurilor de utilizare*
  - Ulterior stabilizării scenariilor, dezvoltatorii derivă din acestea cazurile de utilizare ce definesc viitorul sistem
- *Rafinarea cazurilor de utilizare*
  - Dezvoltatorii verifică completitudinea specificării cerinţelor prin detalierea fiecărui caz de utlizare şi descrierea comportamentului sistemului în situaţii de excepţie

Activităţi ale colectării bazate pe scenarii (cont.)

- *Identificarea relaţiilor dintre cazurile de utilizare*
  - Dezvoltatorii factorizează comportamentele comune mai multor cazuri de utilizare şi identifică dependenţele dintre acestea
  - Această activitate permite verificarea consistenţei specificaţiei sistemului
- *Identificarea cerinţelor nefuncţionale*
  - Dezvoltatorii, clienţii şi utilizatorii convin asupra constrângerilor legate de performanţele sistemului, resursele utilizate, securitatea şi calitatea sa, modalitatea de documentare, etc.

Colectarea cerinţelor - concepte

- Cerinţe funcţionale
- Cerinţe nefuncţionale
- Completitudine
- Consistenţă
- Claritate
- Corectitudine
- Realism
- Verificabilitate
- Trasabilitate
- Inginerie Greenfield
- Re-inginerie
- Ingineria interfeţelor

Cerinţe funcţionale

- *Cerinţele funcţionale* (eng. *functional requirements*) descriu interacţiunile dintre sistem şi mediul acestuia, independent de implementare
  - Mediul = utilizatorii + alte sisteme cu care sistemul în cauză interacţionează
- Ex.: Specificarea cerinţelor funcţionale ale sistemului *SatWatch* - ceas care se actualizează automat (fără intervenţii externe)
  - *SatWatch* este un ceas de mână care afişează timpul pe baza locaţiei curente. *SatWatch* utilizează sateliţi GPS pentru a determina locaţia curentă şi structuri interne pentru a asocia acelei locaţii un fus orar.

    SatWatch actualizează ora şi data la schimbarea fusului orar sau a graniţelor politice. Proprietarul ceasului nu trebuie să îl reseteze niciodată, ca urmare ceasul nu are butoane de control.

    SatWatch utilizează sateliţi GPS pentru determinarea locaţiei curente, ca urmare are aceleaşi limitări ca şi restul dispozitivelor GPS (ex. incapacitatea de a determina locaţia în anumite contexte, ex. tuneluri/pasaje subterane). În timpul perioadelor de black-out, *SatWatch* consideră că nu se schimbă fusurile orare sau regiunile politice. Imediat după ieşirea din intervalul de black-out, se realizează actualizările aferente.

Cerinţe funcţionale (cont.)

SatWatch are un afisaj pe două linii, cea de sus indicând timpul (ora, minute, secunde, fus orar), iar cea de jos data (zi din saptămână, număr zi din lună, luna, an).

În cazul modificării graniţelor politice, proprietarul poate actualiza softul ceasului prin utilizarea dispozitivului *WebifyWatch* (oferit împreună cu ceasul) şi a unui calculator conectat la Internet.

- Cerinţele funcţionale anterioare surprind doar interacţiunile dintre sistemul *SatWatch* şi mediul extern (poprietarul, sateliţii GPS, dispozitivul *WebifyWatch*), fără a referi detalii de implementare (procesor, limbaj, tehnologie, etc.)

Cerinţe nefuncţionale

- *Cerinţele nefuncţionale* (eng. *nonfunctional requirements*) descriu diverse tipuri de constrângeri impuse asupra sistemului, ortogonale pe aspectele legate de funcţionalitate
- Categorii de cerinţe nefuncţionale (*de calitate*), conform modelului FURPS+ ([Grady 1992], [IEEE Std. 610.12-1990])
  - *Utilizabilitate* (eng. *usability*)
    - Denotă uşurinţa cu care un utilizator poate învăţa să opereze, să pregătească intrări sau să interpreteze ieşiri ale unui sistem sau ale unei componente
    - Ex. de cerinţe privind utilizabilitatea: diferite conveţii adoptate la nivelul interfeţelor grafice (şabloane de structurare, scheme de culori, logo-uri, fonturi), help online, ghid de utilizare detaliat
  - *Performanţă* (eng. *performance*) - referă atribute cuantificabile, precum:
    - *timpul de răspuns* (rapiditatea reacţiei sistemului la inputul utilizatorului)
    - *puterea de calcul* (volumul de calcule efectuate într-un interval de timp)
    - *acurateţea rezultatelor*
    - *disponibilitatea* (măsura în care un sistem este operaţional şi accesibil atunci când se doreşte utilizarea lui)

Cerinţe nefuncţionale (cont.)

- *Fiabilitate* (eng. *reliability*)
  - Reprezintă abilitatea sistemului sau componentei de a îndeplini funcţiile cerute în condiţiile stabilite, pentru o anumită perioadă de timp
  - Ex. de cerinţe privind fiabilitatea: un timp mediu precizat între eşecuri, abilitatea de a face faţă unor atacuri de securitate, etc.
  - Referintă şi cu termenul de *dependabilitate* şi acoperind, pe lângă *corectitudine* (eng. *correctness* - conformanţa la specificaţii) şi *robusteţea* (eng. *robustness*) - gradul în care un sistem sau o componentă poate funcţiona corect în prezenţa unor intrări invalide sau a unor condiţii excepţionale şi *siguranţa* (eng. *safety*) - o măsură a absenţei unor consecinţe catastrofale în mediu
- *Suportabilitate* (eng. *suportability*)
  - Denotă uşurinţa modificării sistemului după instalare, incluzând *adaptabilitatea* (eng. *adaptability*) - abilitatea de a modifica sistemul pentru a opera cu noi concepte din domeniul problemei, *mentenabilitatea* (eng. *maintainability*) - abilitatea de a schimba sistemul pentru a opera cu noi tehnologii sau a repara defecte şi *internaţionalizarea* (eng. *internationalization*) - abilitatea de a schimba sistemul penru a opera cu limbi/unităţi de măsură/formate numerice străine

Cerinţe nefuncţionale (cont.)

- Categorii adiţionale de cerinţe calificate drept nefuncţionale în modelul FURPS+ (*pseudo-cerinţe* sau *constrângeri*)
  - *Cerinţe privind implementarea* (eng. *implementation requirements*)
    - Sunt constrângeri care vizează utilizarea unei anumite platforme hardware, a unui anumit limbaj de programare sau a anumitor instrumente
  - *Cerinţe privind interfaţa* (eng. *interface requirements*)
    - Sunt constrângeri impuse de către alte sisteme cu care sistemul în cauză interfaţează
  - *Cerinţe privind modul de operare* (eng. *operation requirements*)
    - Sunt constrângeri privind administrarea şi gestiunea sistemului în mediul său de operare
  - *Cerinţe privind instalarea* (eng. *packaging requirements*)
    - Sunt constrângri legate de livrarea sistemului, cum ar fi suportul folosit pentru instalare
  - *Cerinţe legale* (eng. *legal requirements*)
    - Sunt constrângeri legate de licenţe, legi, certificări
    - Ex.: cerinţe privind obligativitatea oferirii accesului la un anumit soft persoanelor cu dizabilităţi

Cerinţe nefuncţionale (cont.)

- Ex.: cerinţe de calitate pentru *SatWatch*
  - Orice utilizator care ştie să citească un ceas digital şi cunoaşte abrevierile legate de fusurile orare trebuie să poată folosi sistemul fără manual de utilizare (utilizabilitate)
  - *SatWatch* trebuie să afişeze fusul orar corect în maxim 5 minute de la ieşirea dintr-o perioadă de black- out (performanţă)
  - Deoarece *SatWatch* nu dispune de butoane, nu trebuie să apară nici o eroare care să necesite resetarea acestuia (fiabilitate)
  - *SatWatch* trebuie să admită actualizări prin intermediul interfeţei seriale *WebifyWatch* (suportabilitate)
- Ex.: constrângeri pentru *SatWatch*
  - Softul *SatWatch* va fi scris integral în Java, pentru conformanţă cu standardele companiei (implementare)
  - *SatWatch* se conformează interfeţelor fizice, electrice şi soft definite de *WebifyWatch 2.0* (interfaţă)

Completitudine, consistenţă, claritate şi corectitudine

- Validarea continuă a cerinţelor reprezintă un imperativ în procesul de dezvoltare
- Validarea cerinţelor presupune verificarea completitudinii, consistenţei, clarităţii şi corectitudinii acestora
  - *Completitudine*
    - Specificarea cerinţelor se consideră a fi completă dacă surprinde toate aspectele de interes pentru clienţi/utilizatori (au fost descrise toate scenariile posibile, inclusiv cele de excepţie)
  - *Consistenţă*
    - Specificarea cerinţelor se consideră a fi consistentă dacă oricare două cerinţe sunt non-contradictorii
  - *Claritate/non-ambiguitate*
    - Specificarea cerinţelor se consideră a fi neambiguă dacă o aceeaşi cerinţă nu admite interpretări distincte
  - *Corectitudine*
    - Specificarea cerinţelor se consideră a fi corectă dacă reprezintă fidel interesele clientului şi ale dezvoltatorilor, fără a include elemente nedorite
- Corectitudinea şi completitudinea sunt greu de apreciat/certificat, mai ales anterior existenţei sistemului

Realism, verificabilitate şi trasabilitate

- Specificarea cerinţelor se consideră a fi *realistă* dacă sistemul poate fi implementat cu constrângerile impuse
- Specificarea cerinţelor se consideră a fi *verificabilă* dacă ulterior dezvoltării sistemului pot fi proiectate teste care să dovedească conformanţa acestuia cu specificaţia
  - Ex.: cerinţe greu/imposibil de verificat
    - Sistemul *SatWatch* trebuie să aibă un interval mediu între eşecuri de 100 de ani
    - Produsul trebuie să aibă o interfaţă utilizator bună
    - Produsul trebuie să fie fără erori
- O specificare a cerinţelor are proprietatea de *trasabilitate* dacă fiecare cerinţă poate fi urmărită, de-a lungul procesului de dezvoltare, până la funcţiile sistem care o implementează şi reciproc
  - Trasabilitatea cerinţelor e o constrângere critică pentru dezvoltarea testelor şi analiza impactului schimbărilor asupra sistemului

Trasabilitatea cerinţelor

- *Trasabilitatea* (eng. *traceability*) reprezintă abilitatea de a urmări evoluţia unei cerinţe
  - Presupune urmărirea cerinţei de la origine (cine a introdus-o? cărei nevoi client corespunde?) până la nivelul acelor componente ale sistemului pe care le afectează (ce componente implementează cerinţa? ce cazuri de test verifică îndeplinirea sa?)
  - Trasabilitatea permite dezvoltatorilor să motiveze completitudinea sistemului, testerilor să justifice conformanţa acestuia cu cerinţele, proiectanţilor să înregistreze argumentele din spatele deciziilor luate şi echipelor de întreţinere să evalueze efectul schimbărilor
- Trasabilitatea poate fi urmărită prin întreţinerea de referinţe între documente, modele şi cod
  - Fiecare element individual (cerinţă, componentă, clasă, operaţie, caz de test) primeşte un identificator unic
  - O dependenţă este apoi documentată prin intermediul unei referinţe conţinând identificatorii componentelor sursă şi destinaţie

Inginerie Greenfield, re-inginerie, ingineria interfeţelor

- Caracteristicile activităţii de colectare a cerinţelor depind de sursa acestora
  - *Ingineria Greenfield*
    - Procesul de dezvoltare începe de la 0, nu există un sistem anterior
    - Cerinţele sunt furnizate doar de client şi utilizatori
  - *Re-inginerie*
    - Reproiectarea şi reimplementarea unui sistem existent, ca urmare a unor modificări la nivelul proceselor de lucru sau a unor modificări tehnologice
    - Funcţionalitatea sistemului poate fi eventual extinsă, însă scopul său rămâne acelaşi; majoritatea cerinţelor sunt extrase din vechiul sistem
  - *Ingineria interfeţelor*
    - Reproiectarea şi reimplementarea doar a interfeţei unui sistem existent
- În cazul ingineriei Greenfield şi a reingineriei, dezvoltatorii trebuie să acumuleze cât mai multe cunoştinţe relativ la domeniul problemei
  - Surse: descrieri ale proceselor de lucru, documentaţie distribuită noilor angajaţi, manuale ale vechiului sistem, note ale utilizatorilor, interviuri cu clienţii şi utilizatorii

Colectarea cerinţelor - activităţi tehnice

- Identificarea actorilor
- Identificarea scenariilor
- Identificarea cazurilor de utilizare
- Rafinarea cazurilor de utilizare
- Identificarea relaţiilor între cazurile de utilizare
- Identificarea cerinţelor nefuncţionale

Identificarea actorilor

- Permite identificarea frontierei sistemului şi a perspectivelor din care acesta trebuie abordat
- Întrebări utile pentru identificarea actorilor
  - Care sunt grupurile de utilizatori a căror activitate este susţinută de sistem?
  - Care sunt grupurile de utilizatori care execută funcţiile principale ale sistemului?
  - Care sunt grupurile de utilizatori care execută funcţii secundare, precum întreţinere sau administrare?
  - Care sunt echipamentele hardware şi sistemele software externe cu care sistemul curent va interacţiona?
- Întrebările anterioare conduc la identificarea unei liste de entităţi ce urmează a fi rafinată, în general, într-un număr mai mic de actori, diferiţi din perspectiva utilizării sistemului
  - Se unifică entităţile care utilizează aceleaşi interfeţe
  - Se elimină entităţile care, cel mai probabil, nu vor interacţiona în mod direct cu sistemul

Identificarea scenariilor

- În colectarea cerinţelor, dezvoltatorii şi utilizatorii concep şi rafinează o serie de scenarii, pentru a dobândi un nivel comun de înţelegere relativ la funcţionalitatea sistemului
- Întrebări utile pentru identificarea scenariilor
  - Care sunt sarcinile pe care utilizatorul doreşte să le execute sistemul?
  - Care sunt informaţiile pe care le poate accesa actorul? Cine creează aceste date? Pot fi ele modificate sau şterse? De către cine?
  - Care sunt modificările externe despre care actorul trebuie să informeze sistemul? Cât de des? Când?
  - Care sunt evenimentele despre care sistemul trebuie să informeze actorul? Cu ce periodicitate?
- Pentru a răspunde întrebărilor anterioare, dezvoltatorii consultă diverse documente cu informaţii privind domeniul problemei
  - manuale ale unor sisteme anterioare, manuale procedurale, standarde ale companiei, note şi documente elaborate de utilizatori, interviuri cu clienţii şi utilizatorii

Identificarea scenariilor (cont.)

- Ex.: Scenariul *depozitIncendiat* al cazului de utilizare *Raportează*Urgenţă al *SGA*

Nume	depozitIncendiat
Instanţe	bob, alice : OfiţerTeren
actori	john : Dispecer
Flux de evenimente	1. Trecând prin dreptul unui depozit, Bob simte miros de fum. Partenera sa, Alice, activează funcţia *Raportează urgenţă* pe terminalul SGA. 2. Alice introduce adresa clădirii, o scurtă descriere a locaţiei şi un nivel de alertă. Zona fiind aglomerată, solicită o echipă de pompieri şi mai multe de medici. Confirmă datele şi aşteaptă confirmarea dispecerului. 3. John, dispecerul, este alertat de un semnal sonor al staţiei sale de lucru. Verifică informaţiile trimise de Alice şi confirmă primirea lor. Alocă o echipă de pompieri şi două de medici şi ii trimite lui Alice ora estimată a sosirii acestora. 5. Alice primeşte confirmarea şi estimarea.

Identificarea cazurilor de utilizare

- Ex.: Cazul de utilizare *Raportează Urgenţă* al SGA

Nume	*Raportează Urgenţă*
Participanţi	Iniţiat de *OfiţerTeren* Comunică cu *Dispecerul*
Flux de evenimente (scenariu normal)	1. *Ofiţerul* activează funcţia *Raportează urgenţă* a terminalului. 2. Sistemul *SGA* afişează un formular *Ofiţerului*. 3. *Ofiţerul* completează formularul, inserând nivelul de alertă, tipul, locaţia şi o scurtă descriere a situaţiei. Poate propune şi eventuale soluţii la situaţia de urgenţă. După completare, *Ofiţerul* trimite formularul. 4. Sistemul primeşte formularul şi notifică *Dispecerul*. 5. *Dispecerul* consultă informaţia primită şi apelează cazul de utilizare *DeschideCazNou*. *Dispecerul* optează pentru una dintre soluţiile propuse şi confirmă primirea formularului. 6. Sistemul afişează confirmarea şi soluţia aleasă *Ofiţerului.*
Condiţii de intrare	*Ofiţerul* este logat în sistem.
Condiţii de ieşire	*Ofiţerul* a primit confirmarea de la *Dispecer* SAU o explicaţie privind motivul eşecului tranzacţiei.
Cerinţe de calitate	Confirmarea *Dispecerului* ajunge în maxim 30 de sec. după trimitere.

Identificarea cazurilor de utilizare (cont.)

- Ghid de descriere a cazurilor de utilizare
  - Numele cazurilor de utilizare trebuie să fie construcţii verbale, care să indice scopul utilizatorului (ex. *Raportează Urgenţă*, *AlocăResurse*)
  - Numele actorilor trebuie să fie substantive (ex. *OfiţerTeren*, *Dispecer*)
  - Frontiera sistemului trebuie să fie clară, distingându-se între acţunile actorilor şi cele ale sistemului
  - Evenimentele care definesc cazul de utilizare trebuie formulate la modul activ
  - Relaţia de cauzalitate între două evenimente consecutive trebuie să fie clară
  - Fluxul normal al unui caz de utilizare trebuie să descrie o tranzacţie completă (ex.: fluxul normal al cazului *Raportează Urgenţă* descrie toţi paşii, de la iniţierea raportului şi până la primirea confirmării finale)
  - Excepţiile/alternativele se descriu separat
  - Descrierea unui caz de utilizare trebuie să evite referirea directa a componentelor interfeţei grafice cu utilizatorul
  - Descrierea unui caz de utilizare nu trebuie să depăşească două-trei pagini. În caz contrar, cazul se descompune, folosind relaţiile între cazuri de utilizare

Rafinarea cazurilor de utilizare

- Focusul acestei activităţi îl constituie completitudinea şi corectitudinea specificării cerinţelor
  - Dezvoltatorii identifică funcţionalităţi neacoperite de scenariile descrise (cazuri rare sau excepţii) şi le documentează prin rafinarea cazurilor de utilizare existente sau introducerea unor noi cazuri
  - Spre deosebire de identificarea iniţială a actorilor şi cazurilor de utilizare, focusată pe definirea frontierei sistemului, etapa de rafinare introduce detalii suplimentare privind funcţionalităţile oferite de sistem şi constrângerile asociate
- Sunt detaliate următoarele aspecte
  - Informaţia manipulată de sistem
  - Interacţiunile dintre actori şi sistem
  - Drepturile de acces (ce cazuri de utilizare poate invoca fiecare actor)
  - Cazurile de excepţie (sunt identificate şi specificate)
  - Funcţionalitatea comună (este factorizată)

Rafinarea cazurilor de utilizare (cont.)

- Ex.: Rafinarea cazului de utilizare *Raportează**Urgenţă*

Nume	*Raportează**Urgenţă*
Participanţi	Iniţiat de *Ofiţer**Teren* Comunică cu *Dispecerul*
Flux de evenimente	1. *Ofiţerul* activează funcţia *Raportează urgenţă* a terminalului. 2. Sistemul *SGA* afişează un formular *Ofiţerului*. *Formularul include componente privind nivelul de alertă, tipul urgenţei (general, incendiu, accident auto), locaţia, descrierea şi resursele solicitate.* 3. *Ofiţerul* completează formularul, *inserând cel puţin nivelul de alertă şi descrierea situaţiei*. El poate propune şi eventuale soluţii la situaţia de urgenţă *şi poate să solicite anumite resurse*. După completare, *Ofiţerul* trimite formularul. 4. Sistemul primeşte formularul şi notifică *Dispecerul*. 5. *Dispecerul* verifică informaţia primită şi *creează un nou Eveniment în baza de date prin invocarea cazului de utilizare DeschideCazNou*. *Toate informaţiile din formularul primit sunt asociate automat evenimentului creat. Dispecerul alege un răspuns prin alocarea de resurse la eveniment (prin cazul de utilizare AlocaResurse)* şi confirmă primirea formularului *printr-un scurt mesaj către ofiţer.* 6. Sistemul afişează confirmarea şi răspunsul ales *Ofiţerului.*
...	

Identificarea relaţiilor

- Identificarea relaţiilor dintre actori şi cazurile de utilizare (comunicare, incluziune, extindere, generalizare) permite reducerea complexităţii modelului, eliminarea redundanţelor şi a potenţialelor inconsistenţe
- Euristici privind folosirea relaţiilor de incluziune şi extindere
  - Utilizarea relaţiei de incluziune pentru factorizarea comportamentului comun mai multor cazuri de utilizare
  - Utilizarea relaţiei de extindere pentru modelarea comportamentului opţional, excepţional sau cu frecvenţă redusă de apariţie
  - Utilizarea judicioasă a relaţiilor, pentru a evita suprastructurarea modelului funcţional. Un singur caz de utilizare mai complex se poate dovedi a fi mai inteligibil decât un număr prea mare de cazuri mai simple

Identificarea cerinţelor nefuncţionale

- Întrebări ce permit identificarea cerinţelor nefuncţionale aferente modelului FURPS+
  - Utilizabilitate
    - Care este nivelul de expertiză al utilizatorilor?
    - Ce standarde de interfeţe sunt familiare utilizatorilor?
    - Ce documentaţie trebuie pusă la dispoziţia utilizatorilor?
  - Fiabilitate (incluzând robusteţe, siguranţă şi securitate)
    - Cât de fiabil/robust trebuie să fie sistemul?
    - Este repornirea sistemului o alternativă viabilă în cazul unui eşec?
    - Sunt admisibile pierderi de date? Ce volum?
    - Cum trebuie să gestioneze sistemul excepţiile?
    - Există cerinţe privind siguanţa?
    - Există cerinţe privind securitatea?
  - Suportabilitate
    - Care sunt extinderile probabile ale sistemului?
    - Cine întreţine sistemul?
    - Există probabilitatea de a porta sistemul pe alte platforme hardware/software?

Identificarea cerinţelor nefuncţionale (cont.)

- Performanţă
  - Există sarcini critice din punct de vedere al timpului?
  - Câţi utilizatori concurenţi trebuie să suporte sistemul?
  - Ce dimensiuni se estimează că va avea depozitul de date?
  - Care este cel mai slab timp de răspuns acceptat de utilizatori?
- Implementare
  - Există constrângeri privind platforma hardware?
  - Există constrângeri impuse echipei de testare?
  - Există constrângeri impuse echipei de întreţinere?
- Interfaţă
  - Va interacţiona sistemul cu alte sisteme?
  - Cum sunt exportate/importate datele la nivelul sistemului?
  - Există standarde utilizate de client care se aplică noului sistem?
- Instalare
  - Cine va instala sistemul?
  - Pe câte staţii va fi sistemul instalat?
  - Există constrângeri de timp privind instalarea?

Identificarea cerinţelor nefuncţionale (cont.)

- Operare
  - Cine gestionează sistemul după punerea în exploatare?
- Legal
  - Care este procedura de licenţiere?
  - Există taxe rezultând din utilizarea anumitor componente/algoritmi?
  - Există penalizări în cazul căderii sistemului?

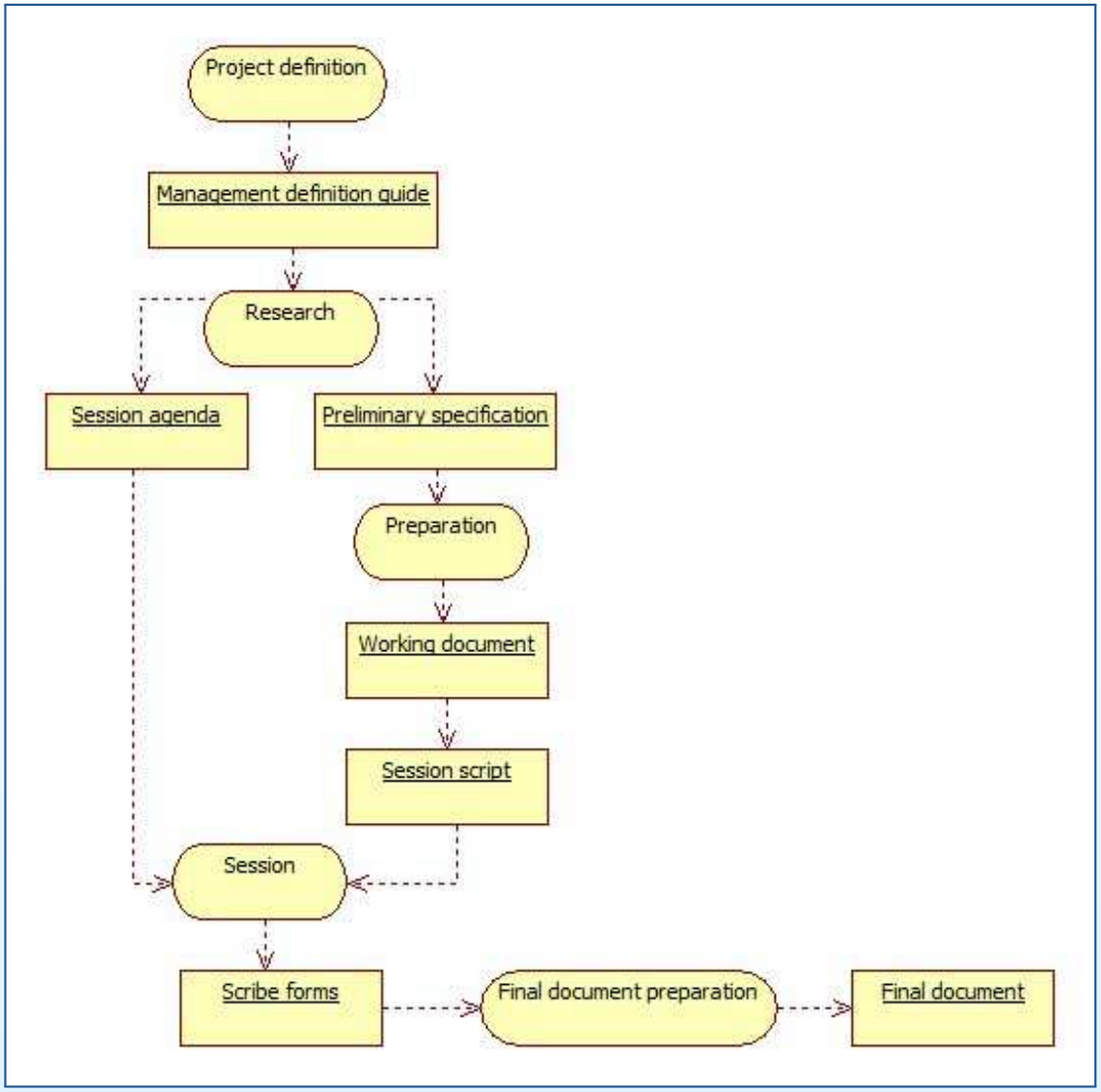
Colectarea cerinţelor - management

Metoda JAD

- *JAD* (*Joint Application Design* [WoodSilver 1989]) este o metodă de colectare a cerinţelor dezvoltată la IBM la sfârşitul anilor ’70
- Eficacitatea sa provine din faptul că activitatea de colectare a cerinţelor se desfăşoară într-o singură sesiune workshop, cu participarea tuturor celor interesaţi în dezvoltarea sistemului (clienţi, utilizatori, dezvoltatori), precum şi a unui moderator de sesiune specializat
- Produsul final al workshopului (documentul final JAD) reprezintă specificaţia completă a sistemului
- Fiind un document redactat de comun acord cu toti participanţii, specificaţia JAD minimizează riscul unor modificări ulterioare ale cerinţelor în procesul de dezvoltare
- Succesul unei sesiuni JAD depinde, de cele mai multe ori, de calificarea moderatorului

Metoda JAD (cont.)

- Activităţi JAD



Activităţi JAD

- *Project definition*
  - Moderatorul JAD intervievează project managerul şi clientul, pentru a identifica scopul şi obiectivele proiectului, informaţiile fiind reţinute în *Management Definition Guide*
- *Research*
  - Moderatorul JAD intervievează utilizatorii sistemului, colectează informaţii relativ la domeniul problemei şi elaborează un prim set de cazuri de utilizare. De asemenea, deschide o listă de probleme ce se doresc a fi abordate în cadrul workshop-ului. Rezultatele acestei activităţi sunt reprezentare de *Session agenda* - agenda de lucru şi *Preliminary specification* - specificaţia preliminară a sistemului
- *Preparation*
  - Moderatorul JAD pregăteşte sesiunea de lucru, creează o primă versiune a documentului final JAD - *Working document* şi compune o echipă constând din project manager, client, utilizatori şi dezvoltatori selectaţi.

Activităţi JAD

- *Session*
  - Moderatorul JAD coordonează echipa în elaborarea specificaţiei sistemului. O sesiune JAD durează 3-5 zile. Se definesc scenariile, cazurile de utilizare şi prototipurile interfeţei cu utilizatorul. Toate deciziile sunt documentate (=> *Scribe forms*).
- *Final document preparation*
  - Moderatorul pregăteşte documentul final - *Final document*, revizuind documentul de lucru pentru a include toate deciziile luate în sesiune. Documentul final reprezintă specificaţia completă sistemului, aprobată în cadrul sesiunii. Acesta este distribuit participanţilor pentru inspectare. Urmează o sesiune de 1-2 ore în care participanţii dezbat modificările şi finalizează documentul.

Referinţe

- [WoodSilver, 1989] J. Wood, D. Silver, *Joint Application Design*, Wiley, New York, 1989.
- [Grady 1992] R. Grady, *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, Englewood Cliffs, NJ, 1992.
- [IEEE Std. 610.12-1990] Institute of Electrical and Electronics Engineers, *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*, New York, NY, 1990.
- Use Case Analysis demos

  Lesson1, `https://www.youtube.com/watch?v=KeMgPqLCkuo&t=19s`
  Lesson2, `https://www.youtube.com/watch?v=8qyHeqInnFU`
  Lesson3, `https://www.youtube.com/watch?v=iex3QbSo61c`
  Lesson4, `https://www.youtube.com/watch?v=3OWNQddbrOs`
  Lesson5, `https://www.youtube.com/watch?v=FrdCX9Gcc4g`