Skripta za pripravo na ustni izpit: Razvoj informacijskih sistemov

povzeto po predavanjih prof. Marka Bajca

3. februar 2020

Poglavje 1

Splošno o informacijskih sistemih

Informacijski sistem opredelimo kot množico medsebojno odvisnih komponent, med katere spadajo strojna in programska oprema ter ljudje, ki zbirajo, procesirajo, hranijo in porazdeljujejo podatke in s tem podpirajo delavne procese v organizaciji.

Delitev IS Informacijske sisteme delimo na **formalne** in **neformalne**. Nefromalne IS ne moremo natančno definirati, saj ne vemo kako delujejo *(npr. instinkt)*. Takšnih IS ne moremo sprogramirati.

1.1 Vrste informacijskih sistemov

Transakcijski IS pokriva vsako dnevne dogodke/transakcije (CDR - call data register).

Upravljalski (poslovodski) IS omogočajo planiranje *(npr. ali se na študijskem programu izplača povečati vpis)*

Direktorski IS

Odločitveni IS omogočajo svetovanje za sprejemanje odločitev, bazirajo na določenem modelu *(npr. odobritev kredita na banki)*

Ekspertni IS *poskušajo* nadomestiti eksperta *(npr. diagnostika pacienta na podlagi njegovih simptomov)*, osnovani so na **bazi znanja** (*if/else* pravila)

Sistemi za avtomatizacijo/ Sistemi za podporo delovni procesom

1.1.1 Dellitev IS na ravni

Operativna raven je najnižja raven, ki vključuje predvsem transakcijske sisteme, kjer hranimo veliko količino neagregiranih podatkov. Stopnja avtomatizacije je visoka, IS na tej ravni so visoko formalni.

Taktična raven vmesna raven, kjer so podatki zmerno agregirani, še vedno je prisotna visoka stopnja formalnosti in avtomatizacije *(npr. beleženje porabljenih enot v mobilnem paketu)*

Stratežka raven je najvišja ravem na kateri so prisotni večinoma agregirani podatki. Takšni sistemi so povečini neformalni (ali zgolj delno formalni), avtomatizacije je malo ali ni prisotna.

1.2 Življenjski modeli razvoja informacijskih sistemov

Faze razvoja IS Razvoj IS običajno delimo v naslednje faze:

1. analiza
2. načrtovanje
3. implementacija
4. testiranje
5. uvedba
6. vzdrževanje

Življenjski model razvoja *(SDLC - system development life cycle)* pove sosledje in način izvedbe faz v okviru razvoja IS.

1.2.1 Pristopi k razvoju IS

Zaporedni ali slapovni pristop

Ko se ena faza konča, se druga začne.

Faza	Cilj
analiza	zahteve
načrtovanje	načrt
izvedba	koda
testiranje	
uvedba	IS

Prednosti in slabosti takšnega pristopa

Prednosti	Slabosti
malo režijskega dela	prepozno testiranje
	nenaravno

Iterativni pristop

Faze razvoja so razdeljene v iteracije, v vsaki iteraciji se izvedejo vse faze. Vsaka iteracija ima nek končni izdelek, ki je del celotne končne rešitve.

Prednosti	Slabosti
povratne informacije	težko načrtovanje iteracij

Planiranje iteracij na makro ravni - release planing celoten razvoj razdelimo na makro iteracije, vsaka izmed iteracij ima kot končen izdelek nek del informacijske rešitve. Te iteracije imenujemo **release** in so omejene na 2-4 mesece. Pred zadnjim relesom moramo imeti izdelek, ki vsebuje vse minimalne zahteve projekta **MVP - minimal value product**.

Planiranje iteracij na mikro ravni - iteration planing Makro iteracije razdelimo na manjše iteracije, ki trajajo med enim in dvema tednoma. V tem času želimo razviti del rešitve, ki sestavlja končen izdelek, ki ga želimo razvit v tem releasu. Iteracije delimo še na **task-e**.

Prototipni pristop

Temelji na delnih prototipnih izdelkih. Analizo in razvoj prototipa opravimo v začetku v celoti in kot rezultat pridobimo delovni prototip. Ne **delovnem prototipu** potem izvajamo testiranje in uporabo, preko katerih se odločamo o nadaljnem razvoju in izboljšavah.

Inkrementalni pristop

Inkrementalni pristop nam omogoča, da projekt razdelimo na več samostojnih podproblemov. Vsak izmed podproblemov je samostojen del končne informacijske rešitve. Vsak del razvijamo posebej ter ga ob zaključku predamo stranki v uporabo ter nadaljujemo z razvojem naslednjega podproblema.

MoSCoW - *must, should, could, won't*

Prednosti	Slabosti
na začetku rešimo najbolj tvegane sklope	vseh IS ne moremo razdeliti na samostojne enote

1.3 Metodologije razvoja informacijskih sistemov

Metodologija je zbirka filozofij, faz, postopkov, pravil, tehnik, orodij, dogovorov med sodelavci,... za razvijalce IS.

1.3.1 Zgodovina metodologij

do začetka 1970' ni metodologij

do zgodnjih '80 se začnejo pojavljati metodologije, ki dajejo poudarek na zajemu in analizi zahtev ter načrtovanju IR.

Pojavijo se tehnike podatkovnega in procesnega modeliranja.

obdobje metodologije (do konca '90) pojavi se velika količina novih metodologij, metodologije pridobijo veliko težo pri razvoju IR.

obdobje ponovne ovenitve metodologij (danes) začenja se obdobje v katerem težke metodologije zamenjajo lahke, metodologije postanejo odvisne in prilagojene potrebam programerja.

1.3.2 Formaliziranje metodologij

Metodologije imajo različno stopnjo formalnosti, bolj kot je metodologija formalizirana natančneje določena je. Nedefinirane metodologije lahko povzročajo slabšo kvaliteto končnega izdelka, netransparentnost razvoja in težje vzdrževanje.

Po drugi strani lahko omogočajo hitrejši, lažji in bolj prilagodljiv razvoj.

1.3.3 Vrste metodologij

Metodologije lahko delimo po več kriterijih:

- življenjski cikel
  - zaporedni
  - iterativni
  - prototipni
- tehnike
  - objektna
  - podatkovna
  - procesna
  - strukturna
- teža metodologije
  - obseg (število elementov, ki jih metodologija opisuje)
  - gostota (zahtevan nivo formalnosti)
- utežitev metodologije
  - spredaj utežena - poudarek na analizi & načrtovanju
  - zadaj utežena - poudarek na kodiranju & testiranju
  - uravnotežena

Komercialne metodologije

- IE (information engineering)
- SSADM
- Rational Unified Process
- STRADIS
- agilne (lahke) metodologije
  - XP - ekstremno programiranje
  - SCRUM
  - FDD

Poglavje 2

Strukturni razvoj

Primer strukturne metodologije - informacijskih inženiring (IE)
Osnovne značilnosti IE:

- povezana množica tehnik planiranja, analize, načrtovanja, razvoja in vzdrževanja IR
- pristop *od zgoraj navzdol*
- avtomatizacija razvoja
- strateško planiranje
- povečana produktivnost
- predpostavlja: **poslovni sistemi so podatkovno, tehnični pa procesno ali dogodkovno usmerjeni**
- posebej obravnavamo podatke in posebej aktivnosti

IE delimo na 4 faze:

1. strateško planiranje (kaj?)
2. analiza (kaj?)
3. načrtovanje (kako?)
4. izvedba (kako?)

2.1 Strateško planiranje

Strteško planiranje je proces izoblikovanja IS, ki organizaciji omogoča uresničitev ciljev in posredno zagotavlja konkurečno prednost.

Cilj planiranja je povezati razvoj IS s poslovno strategijo, načrtovanje pretoka informacij in procesov ter zmanjšati čas in stroške razvoja.

Primer razdelitve dokumenta strateškega planiranja

1. pregled in analiza stanja
2. opredelitev poslovno-informacijske arhitekture
3. opredelitev informacijske vizije
4. opredelitev projektov
5. izdelava akcijskega načrta
6. spremljanje izvajanja in vzdrževanja strateškega plana

2.2 Zajem in specifikacija zahtev

Osnovni namen je opredeliti IR na način, ki bo omogočal:

- pri nakupu IR izbirati med obstoječimi rešitvami
- pri razvoju IR opredeliti osnovno funkcionalnost in druge nefunkcionalne zahteve ter omejitve za izgradnjo IR

Rezultat zajema je dokument (specifikacije) z opredeljenimi funkcionalnostmi. Zajem zahtev opravi sistemski analitiki v sodelovanju s poznavalci problemske domene oz. uporabniki.

Osnovni koraki zajema

1. zajem zahtev
2. ureditev zahtev
3. potrditev zahtev

2.2.1 Zajem zahtev

V zajemu zahtev uporabimo različne tehnike, od izbire le teh je odvisna kakovost specifikacije (razgovori, vprašalniki, opazovanje dela, analiza obstoječega sistema,...).

Napotki pri zajemu zahtev

- objektivnost
- upoštevanje vseh možnosti
- posvečanje podrobnostim
- poudarek na novih in boljših rešitvah
- brez zadržkov pri zajemanju zahtev

2.2.2 Ureditev zahtev

V tem koraku iz neformalno zapisanih zahtev oblikujemo formalno specifikacijo. **Specifikacija je dokument s formalno določenimi zahtevami, ki jih mora končni sistem zajemati.** Specifikacija je posredno določena z izbrano metodologijo, agilne metodologije določajo samo osnovne in okvirne zahteve, težke pa zahteve v celoti.

Kvalitetna specifikacija je pravilna, celovita, nedvoumna, preverljiva, konsistentna in urejena po prioriteti/stabilnosti zahtev.

Ključni vidiki zahtev obsegajo:

- funkcionalnost
- zmogljivost
- omejitve implementacije
- zunanje vmesnike

Zahteve delimo na **funkcionalne**, ki zajemajo želene funkcionalnosti sistemov, in **nefunkcionalne**, ki se nanašajo na tehnične in druge nevsebinske zahteve.

Običajna struktura specifikacije

1. opis namena IR in njenega podsistema
2. opis funkcionalnih zahtev
3. opis nefunkcionalnih zahtev
4. opis vmesnikov
5. slovar pojmov

2.2.3 Potrditev zahtev

Specifikacijo potrdimo s strani naročnika preden nadaljujemo z nadaljnim razvojem IS.

2.3 Analiza

Osnovni namen analize je izdelati razumljiv opis poslovnega okolja, na katerega se nanaša IS.

V analizi izdelamo **model sistema**, da na formalen način opredelimo potrebne podatkovne strukture in funkcije, ki te podatke uporabljajo.

KAJ naj sistem podpira? (Rezultat analize)

- model sistema
- predlog tehnične arhitekture
- prototip komponent uporabniškega vmesnika
- strategija testiranja

Postopke analize izvajajo **sistemski analitik, sistemski arhitekt** in ključni uporabnik.

2.3.1 Izdelava modela sistema

Pri strukturnem razvoju v času analize razvijemo 3 modele s katerimi si lažje viziualiziramo končni sistem

1. podatkovni modeli (konceptualni podatkovni model, entiteta-razmerje)
2. procesni modeli (diagram razgradnje, diagram podatkovnih tokov, procesni diagrami)
3. modeli procesne logike

Modeli so izdelani v polformalnih tehnikah.

Diagrami podatkovnih tokov se fokusirajo na vhodne in izhodne podatke, ki jih potrebuje/proizvede nek proces. V modelu so vidni tokovi podatkov med podatkovnimi skladišči, funkcijami in zunanjimi sistemi, ne pa tudi sosledje dogodkov.

Procesni diagram prikazuje tok dogodkov ali potek določenega procesa.

Model procesne logike dopolnjuje procesni model, predvsem tiste procese, ki niso dovolj jasno definirani. Uporabljamo različne tehnike:

- strukturiran jezik
- odločitvene tabele
- odločitvena drevesa
- diagram prehajanja stanj

2.3.2 Izdelava prototipov

Gre za neobvezen (ampak priporočljiv) korak, katerega cilj je prikazati izgled in osnovne funkcionalnosti sistema.

2.3.3 Izdelava predloge sistema (predlog tehnične arhitekture)

V okviru predloga definiramo komunikacijsko, programsko in strojno arhitekturo, ki je potrebna za vzpostavitev razvojnega, testnega in produkcijskega okolja.

Delitev okolji V času razvoja IR uporabljamo več vzporednih okolji (vsaj 2):

1. razvojno okolje
2. testno okolje (ponovljivi testi)
3. staging (kopija produkcije)
4. produkcijsko

Primer predloge

1. Arhitektura sistema
   (a) ...
2. Postopki, predpisi, standardi
   (a) ...

2.3.4 Opredelitev strategije testiranja

Opredelitev strategije testiranja je prva aktivnost v okviru testiranja.

- Kaj je predmet testiranja?
- Kdo bo testiral, kaj, kdaj in kako?
- Kje bo testno okolje?
- Struktura testov
- ...

2.3.5 Predstavitev rezultatov analize

Na koncu analize njene rezultate predstavimo končnemu uporabniku in nadaljujemo z načrtovanjem IS.

2.4 Načrtovanje

Glavni namen načrtovanja je izdelava načrta glede na specifikacije zbrane v analizi, načrt nam pove ***kako*** sistem izdelamo.

Cilj načrtovanja

- izdelati načrt IR, ki ustreza analizi in upošteva tehnične omejitve
- dokumentirati specifikacije načrta tako, da omogoča nadaljno vzdrževanje sistema
- zasnovati strategijo prehoda na novo aplikacijo

Rezultati načrtovanja

- načrt podatkovne baze
- načrt programskih modulov
- *načrt dokumentacije*
- *načrt testiranja*
- *načrt namestitve in uvedbe*

Pri načrtovanju sodelujejo: načrtovalec podatkovne baze, načrtovalec aplikacije, skrbnik podatkovne baze, izdelovalec dokumentacije, uvajalec, poslovni lastnik in končni uporabnik.

2.4.1 Izdelava načrta podatkovne baze

Iz procesnega modela, ki smo ga pripravili v analizi, oblikujemo logični in/ali fizični model podatkovne baze, ki ustreza tehnični arhitekturi.

2.4.2 Izdelava načrta programskih modulov

Namen je prikazati kako bodo bili, v analizi identificirani, procesi in funkcije v končni IR prikazani.

2.4.3 Izdelava načrta dokumentacije

Z aktivnostjo želimo določiti obseg in strukturo dokumentacije ter izbiro standardov in vzorcev. Dokumentacijo delimo na:

- uporabniško dokumentacijo
- sistemsko-tehnično dokumentacijo
  - podatkovni model

  - arhitektura sistema
  - komponente sistema
  - opis testnega, razvojnega in produkcijskega okolja
  - ...
- navodila za operativno skrbništvo
  - izdelava varnostnih kopij
  - izklop sistema
  - posodabljanje sistema
  - ...

2.4.4 Izdelava načrta testiranja

Najpomebnejša lastnost testov je njihova **ponovljivost**, v sklopu načrtovanja pripravimo načrt testiranja, ki opredeljuje:

- kdo testira
- kaj testira
- kako testira
- kdaj in zakaj testira

2.4.5 Izdelava načrta namestitve in uvedbe

Načrt namestitve in uvedbe definira uvedbo v testno in producijsko okolje, ter med drugim tudi kdo bo kaj tetsiral (v razvoju razvojna ekipa, v testnem in produkcijskem pa tudi končni uporabniki).

Načrt vsebuje:

- načrt namestitve
- načrt dodelitve pravic
- načrt prevedbe podatkov (iz starega v nov sistem)
- načrt uvajanja
- načrt za izvedbo prevzemnega in končnega testa
- načrt prehoda na nov sistem

2.4.6 Testiranje

S testiranjem želimo doseči preverjeno delujočo aplikacijo. V okviru razvoja IR se različna testiranja izvajajo različno pogosto in ob različnih časih. Teste delimo na:

- teste programskih enot
- teste integracije
- sistemske teste
- teste sprejemljivosti

Izvajalci testov so razvijalci (predvsem v razvojnem okolju), preizkuševalci (sistematično testiranje v testnem okolju) in končni uporabniki (testno in produkcijsko okolje).

2.5 Namestitev in uvedba

2.5.1 Prehod na nov sistem

Fazni pristop star sistem nadomestimo z novim v več korakih, pri čemer oba sistema delujeta medseboj. Potreben je razvoj vmesnikov, ki omogočajo komunikacijo med sistemoma. Postopno lahko sistem menjujemo po področjih, lokacijah, modulih,...

Zamenjava ali vse naenkrat velik riziko, saj star sistem ne deluje več, novega pa v produkcijskem okolju še nismo testirali.

Vzporedno delovanje starega in novega sistema. Je najbolj varen način, vendar podvaja delo končnemu uporabniku.

Poglavje 3

Objektni razvoj

Glavne razlike v primerjavi s strukturnim razvojem so pri analizi, načrtovanju in izvedbi.

Objekt lahko predstavlja fizično entiteto ali koncpetualni pojem. S stališča razvoja IR je objekt koncept, abstrakcija z natančno določenimi mejami in lastnostmi, ki so pomemben za IR. Objekt ima:

- stanje (objekt se zaveda svojega stanja - lahko *pove* svoje stanje)
- obnašanje (funkcije, ki jih objekt *dela* in kar se z njim da delati *"pisalo piše- kdo piše?"*)
- entiteto

Odnos med objektom in razredom je podobno odnosu med entiteto in entitetnim tipom.

Enkapsulacija ali skrivanje podatkov O objektu ne rabimo vedeti vsega, ampak samo tisto, kar potrebujemo za njegovo uporabo (vmesnik). Enkapsulacija zahteva, da skrijemo:

- implementacijo obnašanja, ki je na voljo prek vmesnika
- podatke znotraj objekta, ki so potrebni za implementacijo obnašanja in beležijo stanje objekta v trenutku obstoja

Dedovanje in hierhija Dedovanje uporabimo, da nekemu razredu dodelimo lastnosti nekega drugega, dedovanega razreda. *Npr. študent in delavec oba dedeujeta lastnosti kot so ime in priimek od razreda oseba.*

UML je standariziran nabor tehnik za modeliranje objektnih sistemov, med UML diagrame umeščamo: razredne, komponetne, use-case diagrame, diagrame stanj,...

3.1 Osnove procesa objektnega razvoja RUP

RUP rational unified process (proces določa kdo, kdaj, kaj dela)

Glavne značilnosti RUP

- snernice za učinkovit razvoj
- zmanjšuje tveganje, povečuje predvidljivost
- zajema in vpeljuje najboljše izkušnje (učenje iz izkušenj,...)
- pospešuje vizijo
- vpeljuje načrt za vpeljavo

6 najboljših praks v razvoju IR, ki jih vpeljuje RUP

<table>
<tr><td colspan="4">iterativen razvoj</td></tr>
<tr><td>obvladovanje zahtev</td><td>uporaba komponentne arhitekture</td><td>virtualno modeliranje</td><td>preverjanje kakovosti</td></tr>
<tr><td colspan="4">nadzorovanje sprememb</td></tr>
</table>

Arhitekrura IR določa smernice razvoja, zgradbo izdelka in sestavo skupine. RUP daje velik poudarek na arhitekturo že v začetnih iteracijah zato zahteva 4+1 pogled:

- logični pogled (analitik/razvijalec) - zgradba sistema
- izvedbeni pogled (programer) - upravljanje
- postavitveni pogled (sistemski inženir) - topologija sistema
- procesni pogled (sistemski poznavalec)
- primer uporabe (končni uporabnik) - funkcionalnost

Kaj še omogoča RUP?

- vzpostavitev in ohranitev nadzora nad projektom
- obvladovanje kompleksnosti
- vzdrževanje celovitosti sistema
- razširanje možnosti ponovne uporabe
- pospeševanje komponentno usmerjen razvoja

3.1.1 RUP faze

- začetna faza - vzpostavitev sistema, opredelitev okvirjev in načrtovanje virov
- zbiranje informacij - specifikacija zahtev, načrtovanje virov
- konstrukcija - razvoj sistema
- prevzem - predaja izdelka

3.2 Zajem zahtev

Namen zajema zahtev je doseči soglasje s stranko oz. uporabnikom, kaj naj sistem dela. Na drugi strani omogočimo razvijalcem boljše razumevanje zahtev sistema ter določimo meje sistema, ki nam zagotavljajo osnovo za načrtovanje tehnične vsebine.

RUP, za razliko od strukturnega razvoja, vpeljuje višjo stopnjo formalnosti:

- primeri uporabe
- opisi s tokovi dogodkov
- uporaba drugih tehnik UML v posebnih primerih (diagrami stanj, diagrami aktivnosti, diagrami iterakcije)

Diagram primerov uporabe prikazuje primere uporabe ter akterje, ki primere uporabe uporabljajo. Znotraj posameznih primerov uporabe določimo še tok dogodkov (vsak tok dogodkov ima osnovni in enega ali več alternativnih tokov), ki opisuje zaporedje izvajanja posameznih dogodkov.

Pri izdelavi modela uporabe najprej definiramo akterje ter primere uporabe, ki jih med seboj povežemo. V primeru odvečnih primerov uporabe, akterjev ali povezav jih pri pregledu odstranimo.

Poleg primerov uporabe in **opisov primerov uporabe** v fazi analize definiramo še **slovar** in **dodatne specifikacije**, ki zajemajo funkcionalnosti, uporabnost, zanesljivost, učinkovitost, podporo in omejitve pri načrtovanju.

3.3 Objekta analize in načrtovanja

Primerjava faz analize in načrtovanja

Analiza	Načrtovanje
razumevanje problema	razumevanje rešitve
idealiziramo načrtovanje	operacije in atributi
obnašanje	zmogljivost
struktura sistema	blizu programski kodi
funkcijske zahteve	nefunkcionalne zahteve
majhen model	velik model
	življenjski cikel objektov

3.3.1 Analiza arhitekture

Identifikacija ključnih abstrakcij skozi modele iz analize in zajema zahtev izluščimo najpomembnejše (ključne) objekte (abstrakcije).

Analiza potreb po sistemskih virih kot so trajnost, procesna komunikacija, porazdeljevanje, obvladovanje transakcij, varnost,...

Opredelitev arhitekturne ravni predstavljajo aplikacijo z različnih nivojev abstrakcije, primeri:

- model-view-controller
- pipes and filters
- blackboard

Aplikacija lahko uporablja več arhitekturnih vzorcev. Organiziranost arhitekturnih ravni pokažemo s paketi, ti so elementi sistema, ki lahko vključujejo druge elemente modela.

3.3.2 Analiza primerov uporabe

Dopolnitev opisov primerov uporabe primere uporabe dopolnimo s podatki, ki so potrebni za nadaljni razvoj.

Identifikacija potrebnih razredov za realizacijo posameznih primerov uporabe Razrede v osnovi delimo na kontrolne (en primer uporabe ima navadno enega), mejne (za vsak zunanji sistem imamo enega) in entitetne (poslovni razredi s katerimi hranimo in upravljamo podatke). Med razredi definiramo tudi povezave (te so lahko neusmerjene ali usmerjene), ki jih poimenujemo in določimo številnost.

Razdelitev odgovornosti posameznim razredom Z uporabo diagrama zaporedja iz primera uporabe definiramo kateri razred izvede katero izmed aktivnosti. Mejni razredi prevzamejo odgovornosti komunikacije z akterji, poslovni podatke, kontrolni pa (običajno) pomembnejši del toka dogodkov.

Podrobna analiza potrebnih sistemskih virov Sestavimo seznam vseh potrebnih sistemskih storitev ter razredov, ki potrebujejo sistemske storitve. V tem delu tudi opišemo lastnosti sistemskih storitev.

Poenotenje razredov pridobljenih z analizo