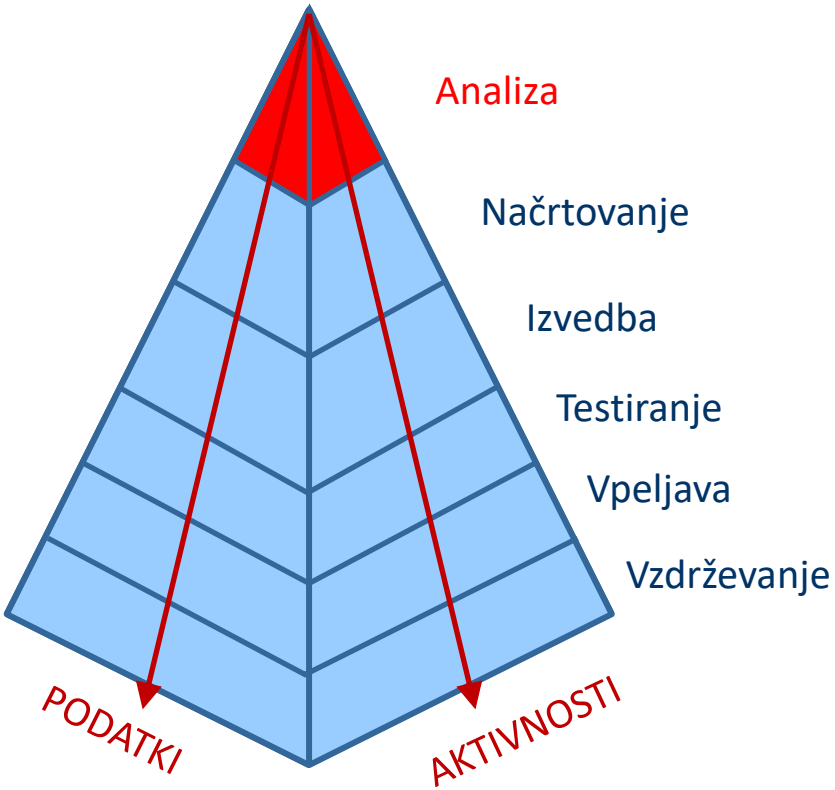
8. Modeliranje IS

KAKO pripravimo model, po katerem bomo zgradili rešitev?

Analiza IS (ponovitev)

- Glavni namen analize je izdelati razumljiv opis realnega sveta oziroma poslovnega okolja, na katerega se nanaša razvoj IS.
- Analiza daje odgovor na vprašanje, KAJ naj IS podpira. Kaj se izvaja v poslovnih funkcijah in kakšne podatke te rabijo?



Analiza IS (ponovitev)

- Analiza služi kot:
  - sredstvo za definicijo zahtev,
  - osnova za dogovor med naročnikom in izvajalcem, in
  - osnova za kasnejše faze razvoja.
- Osnovne aktivnosti analize zajemajo:
  - Zajem zahtev: zajem zahtev se nanaša na opredelitev funkcionalnosti, ki naj jo sistem podpira. Uporabniki sodelujejo z analitiki.
  - Modeliranje sistema: predstavitev zajetih zahtev v razumljivi in nedvoumni obliki. Model analize večinoma zajema več vidikov, ki so predstavljeni vsak s svojim modelom.

Modeliranje sistema

Modeliranje sistema

- Model je poenostavitev realnosti, pri čemer je abstrakcija realnosti poljubno natančna.
- Model je enostavnejši od realnosti.
  - Prikazuje le pomembne elemente in izpušča tiste, ki nas ne zanimajo.
- Modeliranje prinaša naslednje bistvene prednosti:
  - Omogoča vizualizacijo sistema.
  - Prikazuje tako statične kot dinamične lastnosti sistema.
  - Predstavlja šablono za nadaljnjo gradnjo sistema.
  - Dokumentira sprejete odločitve.
- Modele razvijamo zato, da bi sisteme bolje razumeli.

Modeliranje sistema

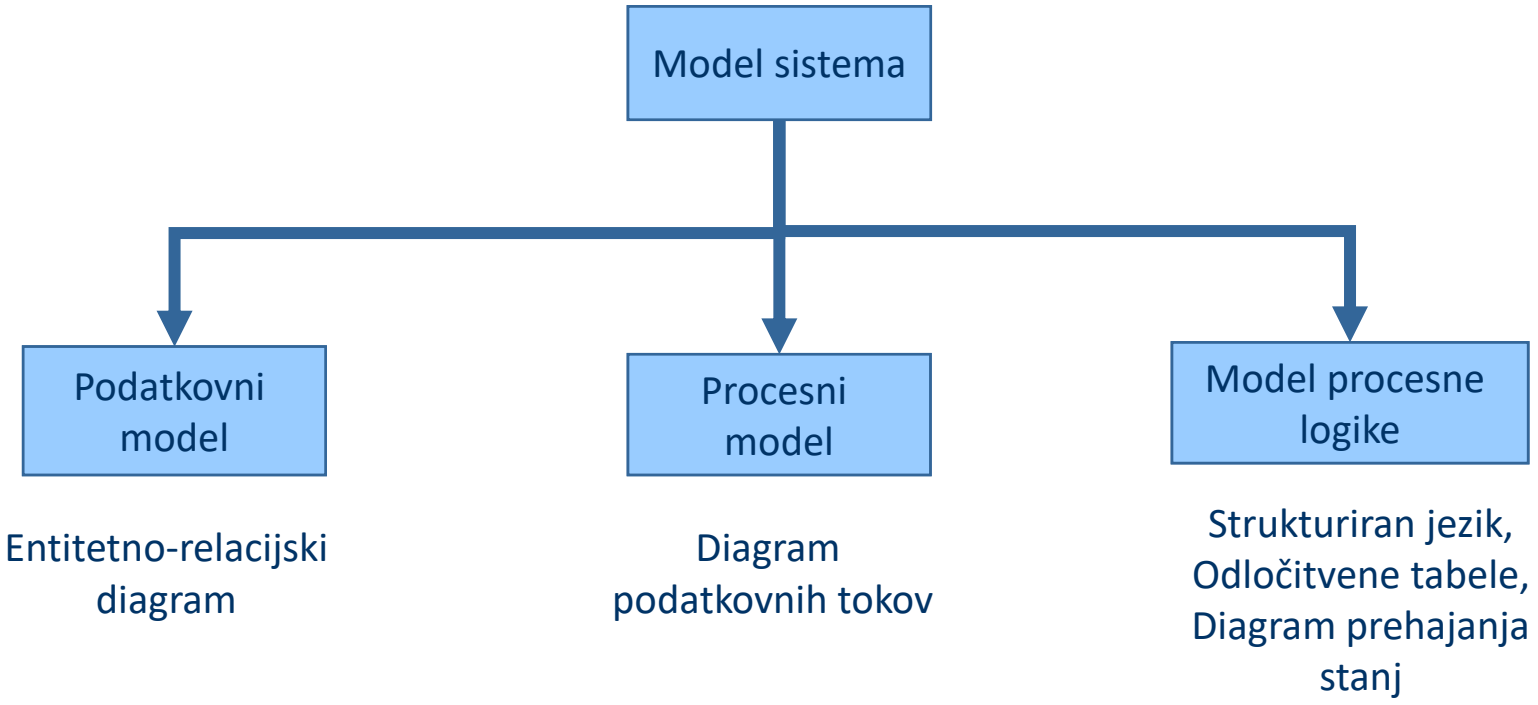
- V splošnem poznamo na področju modeliranja dva pristopa:
  - Tradicionalno modeliranje – modeliranje z vidika postopka.
  - Objektno modeliranje – modeliranje z vidika objekta.

Modeliranje sistema

- Tradicionalni pristop k razvoju IS je osnovan na postopkovni perspektivi:
  - Pogled usmerja razvijalca, da se osredotoči na potek postopkov in njihovo razgradnjo na manjše dele.
  - V praksi je pristop zelo dobro preizkušen in se veliko uporablja.
  - Zagovorniki novejših pristopov mu očitajo neprilagodljivost na vhodne spremembe.
- Modernejši pristop k razvoju IS je objektno usmerjen pristop. Osnovni gradnik takega pristopa je objekt.

Tradicionalno modeliranje

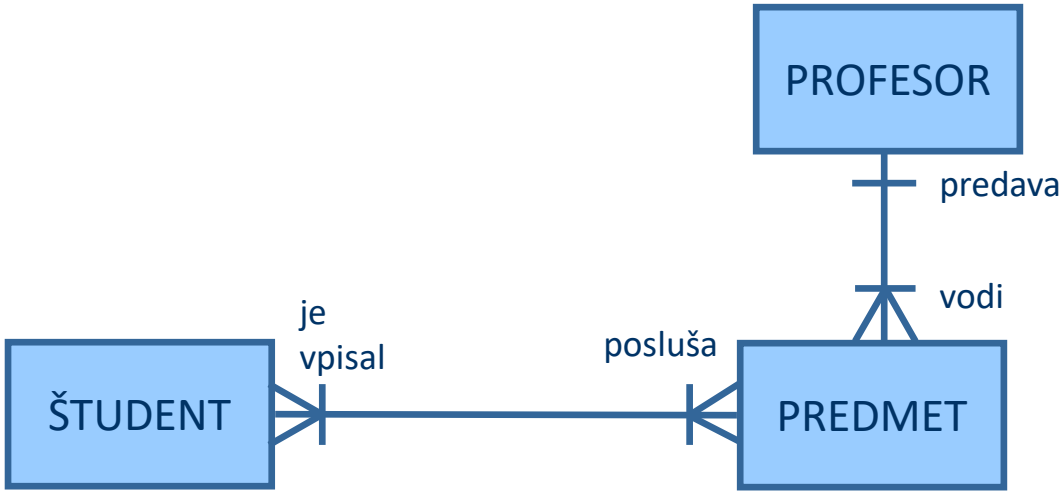
- Specifikacija sistema je sestavljena iz treh modelov, ki vsak s svojega vidika opisujejo sistem.



Tradicionalno modeliranje

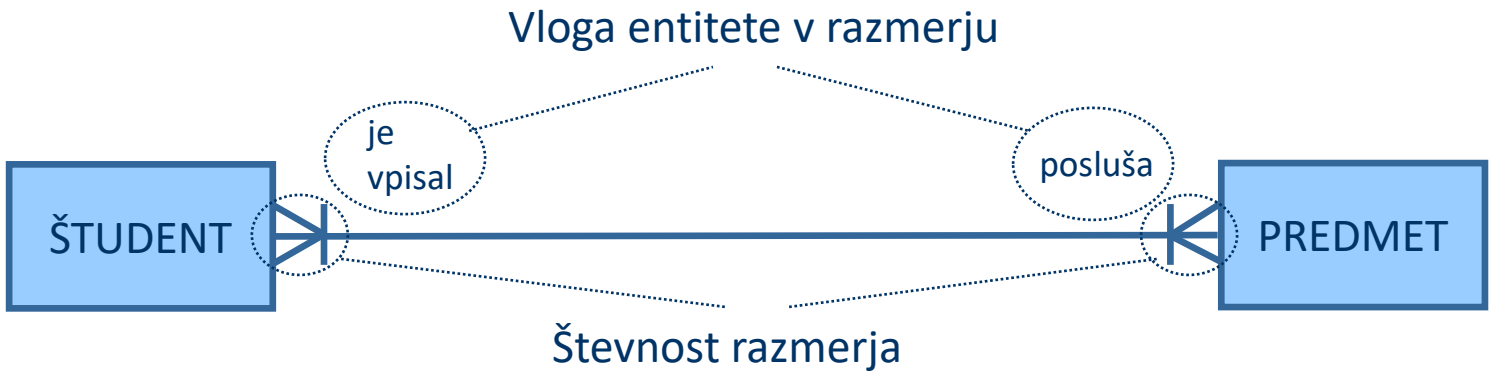
- Podatkovni model
  - prikazuje sistem s podatkovnega vidika z opisom podatkovnih struktur, ki so potrebne za delovanje sistema; zajema tudi vse povezave med podatkovnimi strukturami.
- Procesni model
  - prikazuje sistem z vidika aktivnosti ali procesov, ki se v sistemu izvajajo. Definirani so tokovi podatkov med procesi.
- Model procesne logike
  - natančneje definira procese, definirane v procesnem modelu.

E-R (entitetno-relacijski) diagram



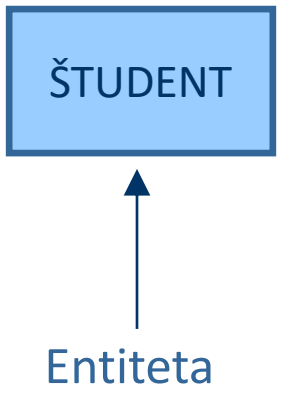
Entiteta in razmerje

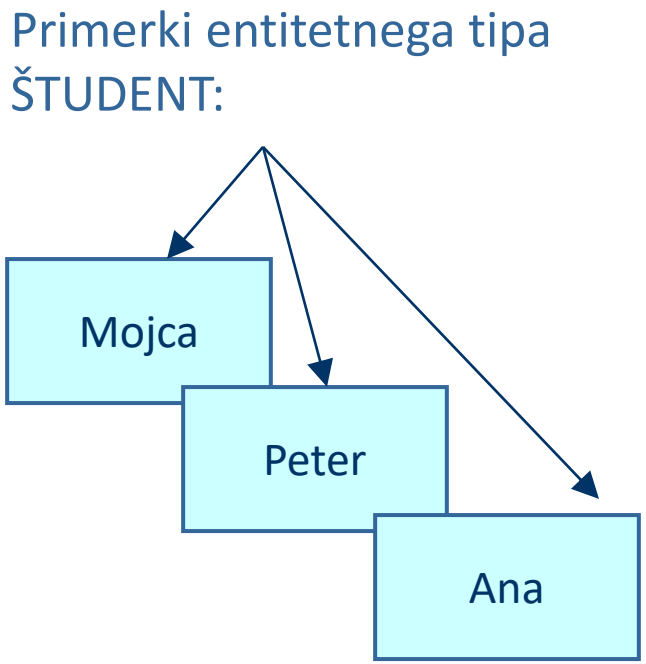
- Osnovna gradnika sta entiteta in razmerje (relacija).
- Entiteta je realni ali abstraktni predmet obravnave, značilen za področje, ki ga obravnavamo.
- Entitete realnega sveta običajno niso izolirane, temveč obstaja med njimi in drugimi entitetami pomenska povezava. To ponazorimo z razmerjem.



Entiteta – primerki

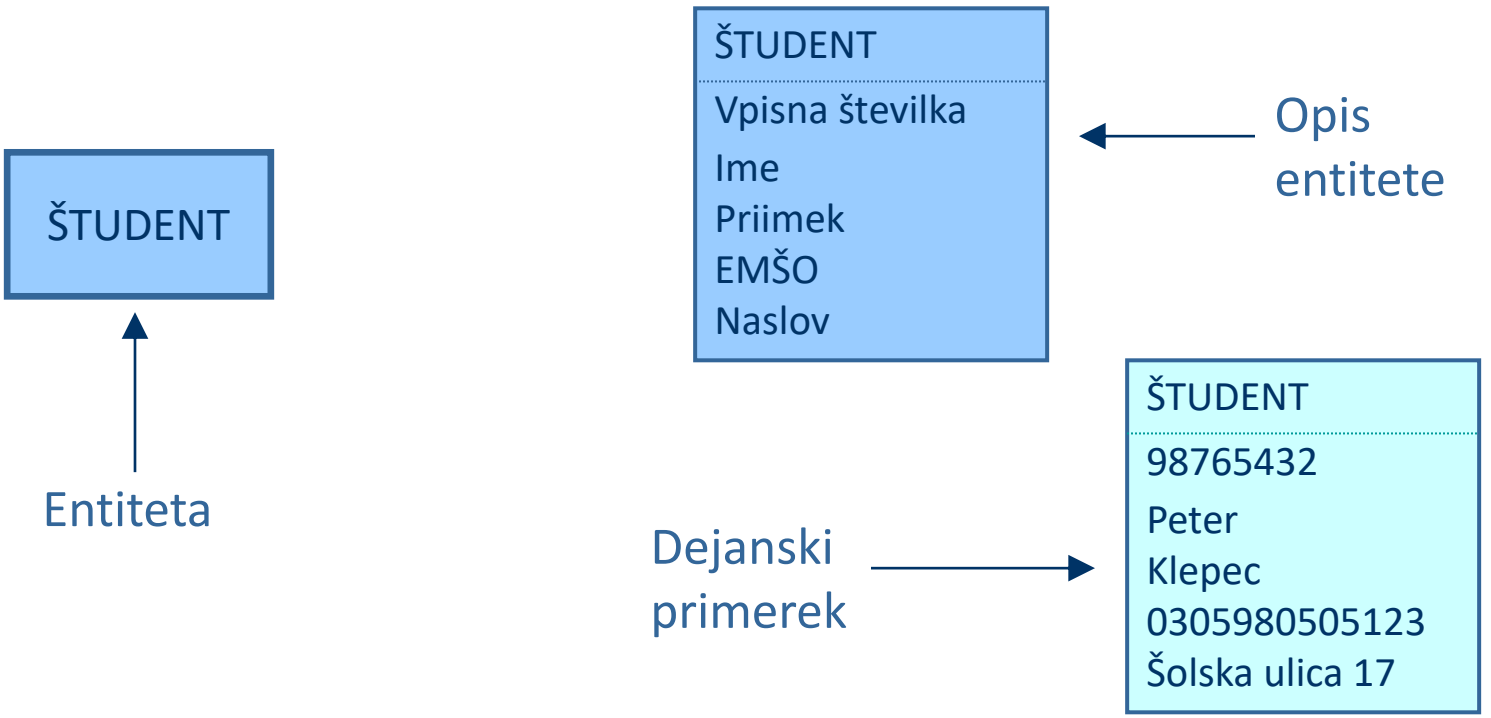
- Entiteta združuje primerke istega tipa.





Atributi

- Primerke opišemo z naborom podatkov – atributi.



Osnovne značilnosti razmerij

- Številnost označujemo z grafičnimi simboli.

Profesor predava enega ali več predmetov.
Predmet vodi natanko eden profesor.

Profesor ima lahko več asistentov.
Asistent pomaga natanko enemu profesorju.

Osnovne značilnosti razmerij

- Med dvema entitetama je lahko več razmerij.

Prvo razmerje opredeljuje predmete, ki jih študent posluša.
Drugo razmerje pove, katere predmete je študent že opravil.

Vaja dela… (1/4)

- Razviti želimo IS **MojCD**, ki nam bo pomagal pri vodenju domače zbirke CDjev.
  - Hraniti želimo podatke o glasbenih CDjih, ki jih imamo.
  - Hraniti želimo podatke o tem, kdaj in komu smo posodili posamezen CD ter če in kdaj ga je ta oseba vrnila; želimo hraniti zgodovino vseh izposoj.
  - Želimo imeti možnost enostavne statistike: kateri CDji so bili največkrat posojeni, kdo si je največ izposojal, iz katere zvrsti glasbe so bili posojeni CDji.
- Narišimo ERD diagram za tak sistem!

- *Namig: pomagajmo si z diagramom primerov uporabe (ki nam bo služil skozi celoten proces modeliranja)!*

Diagram funkcionalnosti

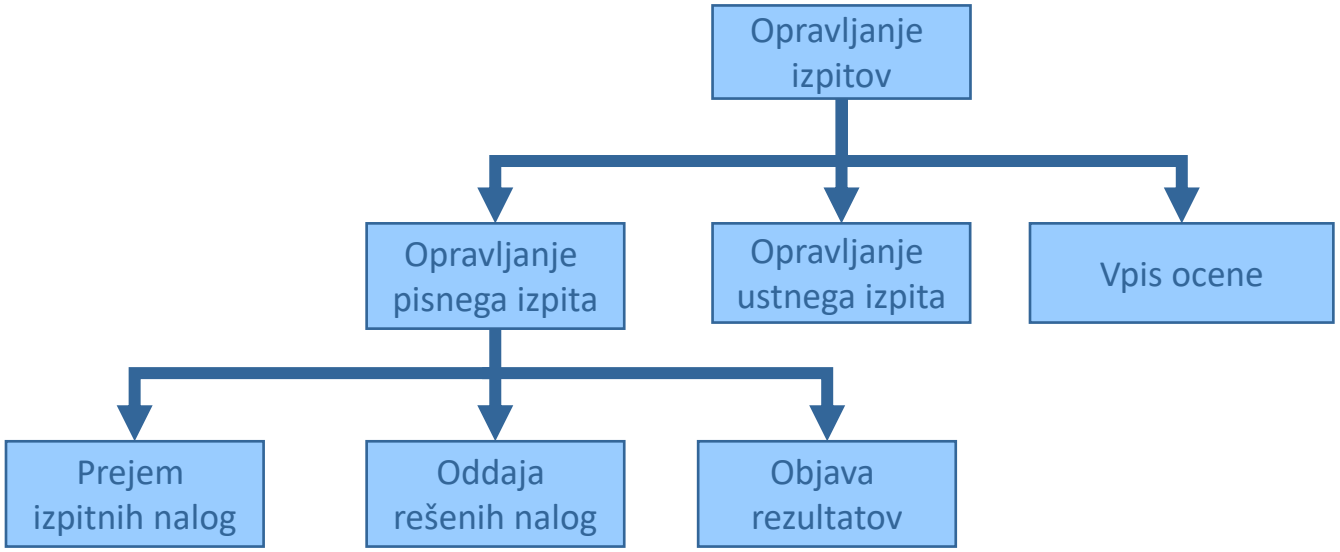
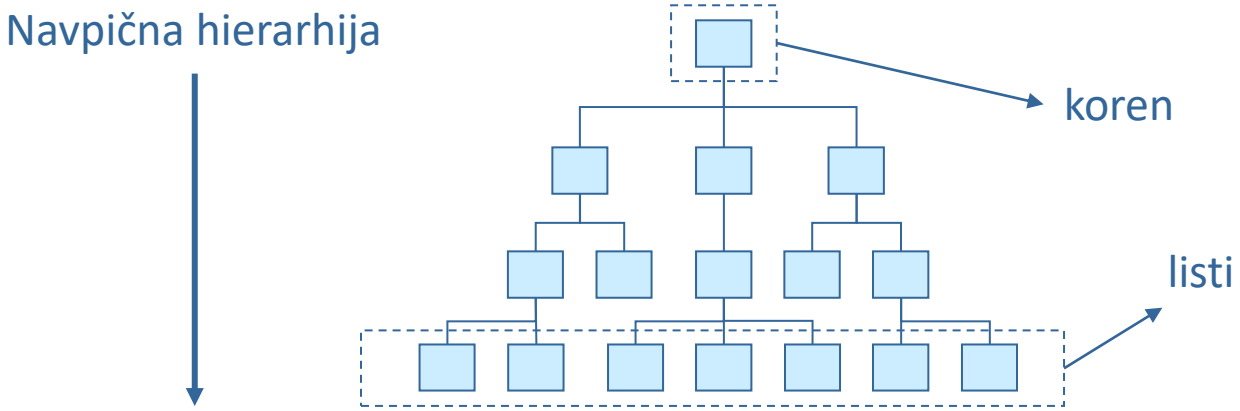


Diagram funkcionalnosti

- Z diagramom funkcionalnosti (diagram funkcionalne razgradnje) prikažemo hierarhijo funkcij, ki jih želimo s sistemom podpreti (kaj vse bo sistem omogočal).
- Hierarhijo funkcij prikažemo z navpično drevesno strukturo.
- Vsaka hierarhična struktura se začne na vrhu z eno samo vseobsegajočo enoto – korenom strukture – in se nadaljuje vse do listov, kateri predstavljajo elementarne funkcionalnosti.

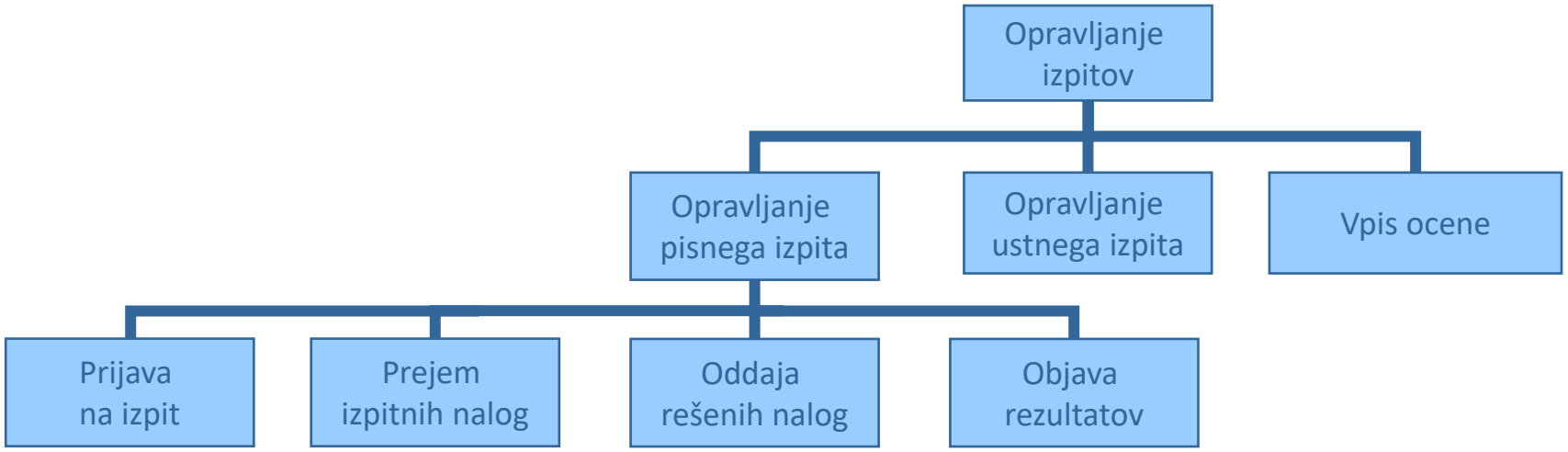


Razčlenjevanje funkcionalnosti

- V analizi pogosto identificiramo večje število funkcij – procesov (npr. nekaj sto).
- Predstavitev vseh procesov hkrati (na istem nivoju) je nepregledna, sama vsebina pa nerazumljiva.
- Uporabljamo razčlenjevanje (dekompozicijo), s čimer funkcionalnosti razvijamo od najvišjega nivoja (korena), kjer nastopajo obsežnejši procesi, do najnižjega nivoja (listov), kjer nastopajo zelo podrobni procesi.

Značilnosti diagrama funkcionalnosti

- Diagram funkcionalne razgradnje nam služi za prikaz razdelitve funkcionalnosti.
- Elementarne funkcije (listi) so dovolj enostavni, da jih lahko nedvoumno opišemo.
- Diagram funkcionalnosti nam služi kot osnova za izgradnjo diagramov podatkovnih tokov.



Vaja dela… (2/4)

- Izdelajmo diagram funkcionalosti za sistem **MojCD** (za katerega smo v prejšnjem poglavju izdelali diagram ERD).
  - Razmislimo, katere funkcionalnosti bomo zajeli v sistemu, in kako jih bomo organizirali po nivojih.

- *Namig: seveda si lahko pri tem koristno pomagamo z že izdelanim diagramom primerov uporabe!*

Diagram podatkovnih tokov

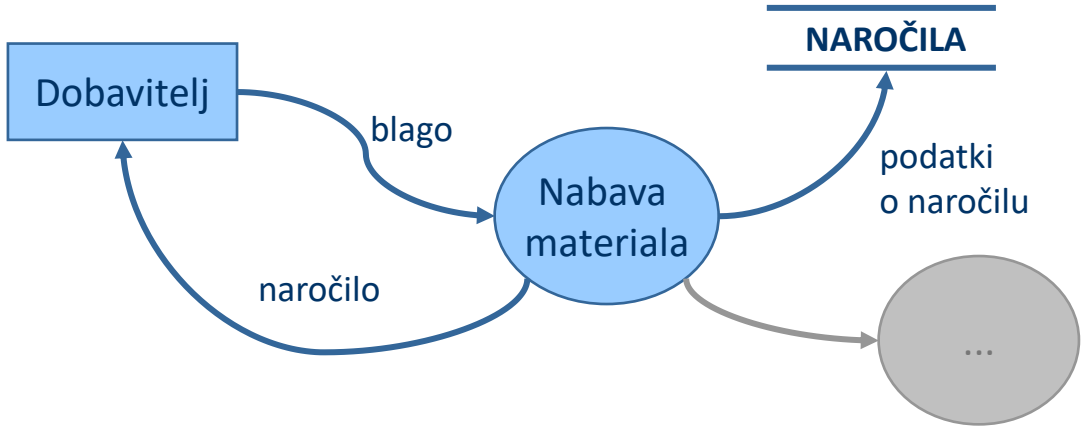


Diagram podatkovnih tokov

- Diagrame podatkovnih tokov (data flow diagram, DFD) uporabimo pri modeliranju sistema za prikaz okolja, v katerem bo sistem deloval ter za prikaz odvisnosti med procesi, ki jih bo sistem podprl.
- DFD združujejo podatkovni in procesni pogled na obravnavano področje.
- DFD sodijo med enostavnejše tehnike, saj zajemajo minimalno število gradnikov.

Osnovni gradniki DFD

- Osnovni gradniki DFD so:
  - Proces
  - Podatkovni tok
  - Podatkovna shramba
  - Zunanji izvor ali ponor

Proces

- Proces predstavlja v DFD množico aktivnosti, ki vhodne podatke pretvorijo v izhodne.
- Proces je (edini) dinamičen gradnik modela – iz vhodnih podatkov tvori izhodne po vnaprej predpisanem postopku.
- Naziv procesa je običajno glagol, glagolski samostalnik ali zaporedje besed, ki opisujejo vrsto dejavnosti. Poleg naziva procesa je procesu dodeljena številčna oznaka, ki proces enolično določa.

Notacija Yourdan-DeMarco

Podatkovni tok

- Podatkovni tok
  - predstavlja množico vhodnih ali izhodnih podatkov, ki imajo enolično definirano vsebino in strukturo.
- Podatki, ki jih tok prikazuje, so lahko:
  - Elementarni podatki (ime, znesek, šifra, ...)
  - Sestavljeni podatki, npr. dokumenti (račun, časopisni članek, izpis iz rojstne matične knjige,...)

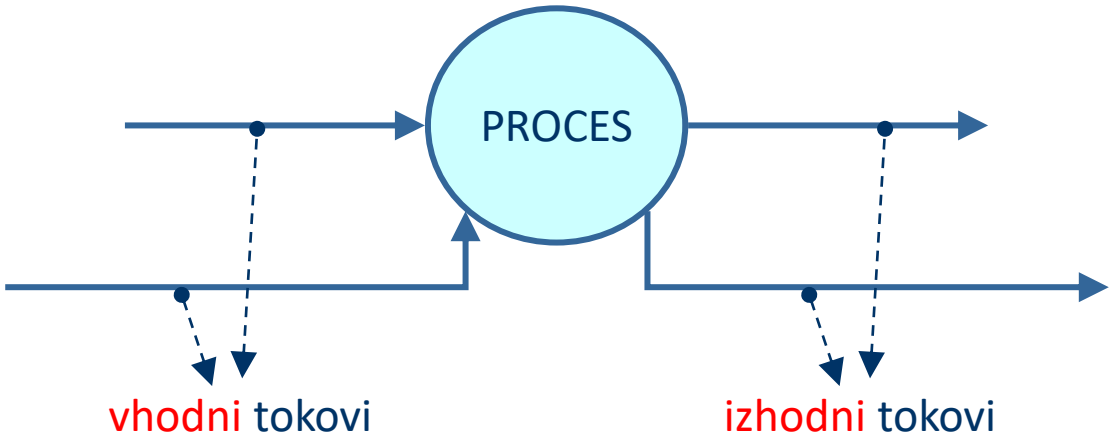
Podatkovni tok

- Podatkovni tokovi zagotavljajo procesu podatke (vhod v proces), ali obdelane podatke prenašajo iz procesa (izhod iz procesa).
- Podatke lahko obdelujejo le procesi, zato mora biti vsak podatkovni tok obvezno povezan s procesom!
  - Sam tok podatka ne more spremeniti, spremeni ga lahko le proces!

Naziv toka pove, kaj tok prenaša! Za označevanje uporabljamo samostalnike (v ednini) ali pa kombinacijo samostalnika in pridevnika.

Vhodni in izhodni tokovi

- Glede na smer prenosa podatkov ločimo:
  - Vhodne tokove: potekajo proti procesom (v proces prinašajo podatke), in
  - Izhodne tokove: potekajo iz procesov (iz procesa pošiljajo rezultate).
  - Vsak proces ima vsaj en vhod in vsaj en izhod!



Podatkovna shramba

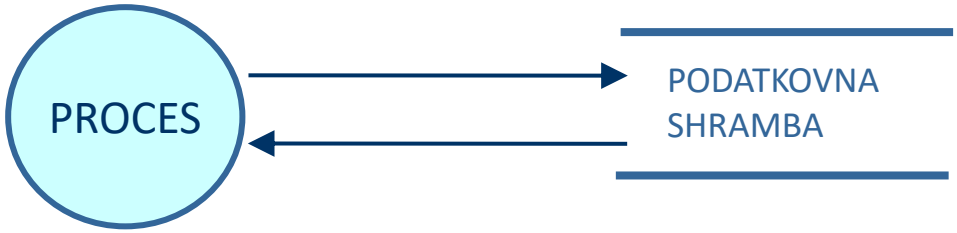
- Je koncept, ki označuje prostor za shranjevanje podatkov iz nekega procesa, z namenom, da bodo ti na voljo tudi kasneje in/ali drugim procesom.
  - V podatkovni shrambi hranimo podatke trajno, ne le za čas obdelave.
- V fazi analize se s podatkovno shrambo opisujejo logični sklopi podatkov. Njihova fizična organizacija se določi kasneje.

NAZIV PODATKOVNE SHRAMBE

Naziv podatkovne shrambe je pogosto enak nazivu vhodnih podatkovnih tokov. Shramba je podatkovni tok v mirovanju.

Podatkovna shramba

- Proces lahko opravlja dve vrsti operacij nad podatkovno shrambo:
  - Piše v shrambo (spreminjanje obstoječih podatkov, dodajanje in brisanje).
  - Bere iz shrambe.
- Pogosto proces piše in bere iz iste shrambe.

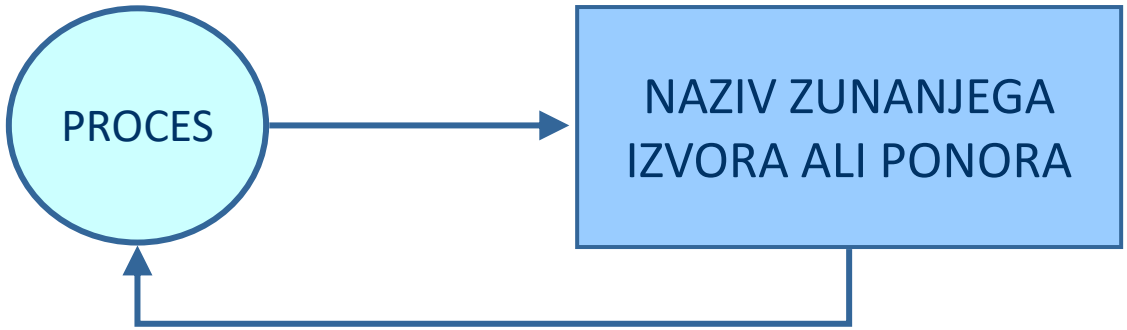


Zunanji izvor/ponor

- Zunanji izvori ali ponori podatkov so vse tiste komponente, ki sodelujejo z načrtovanim IS, niso pa njegov sestavni del. So okolje našega IS.
  - Zunanji uporabniki (posamezniki, organizacije, …) ali ostali koncepti, ki predstavljajo zunanje procese ali zunanje sisteme.
- Njihova struktura ali obnašanje nas ne zanimata.
- Zanimajo nas le podatkovni tokovi, ki zunanje izvore/ponore povezujejo s procesi v DFD.
  - Zunanji izvori vnašajo podatke v naš sistem (vhodni podatki).
  - Zunanjim ponorom pošiljamo podatke oz. rezultate iz našega sistema (izhodni podatki).

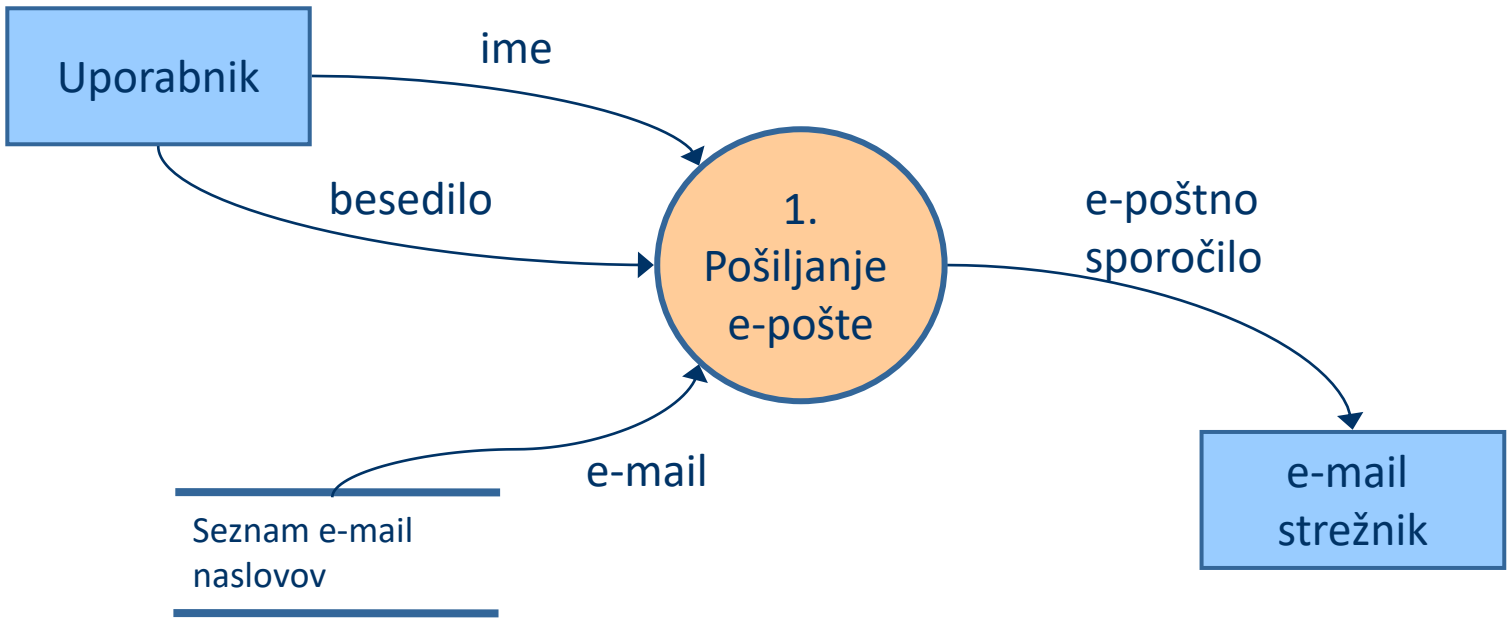
Zunanji izvor/ponor

- Vsak zunanji izvor vnaša podatke vsaj enemu (ali več) procesu v našem IS, vsak zunanji ponor prejema podatke od vsaj enega (ali več) procesa v našem IS.
- Nek zunanji sistem je lahko istočasno zunanji izvor in zunanji ponor (torej vnaša in prejema podatke).



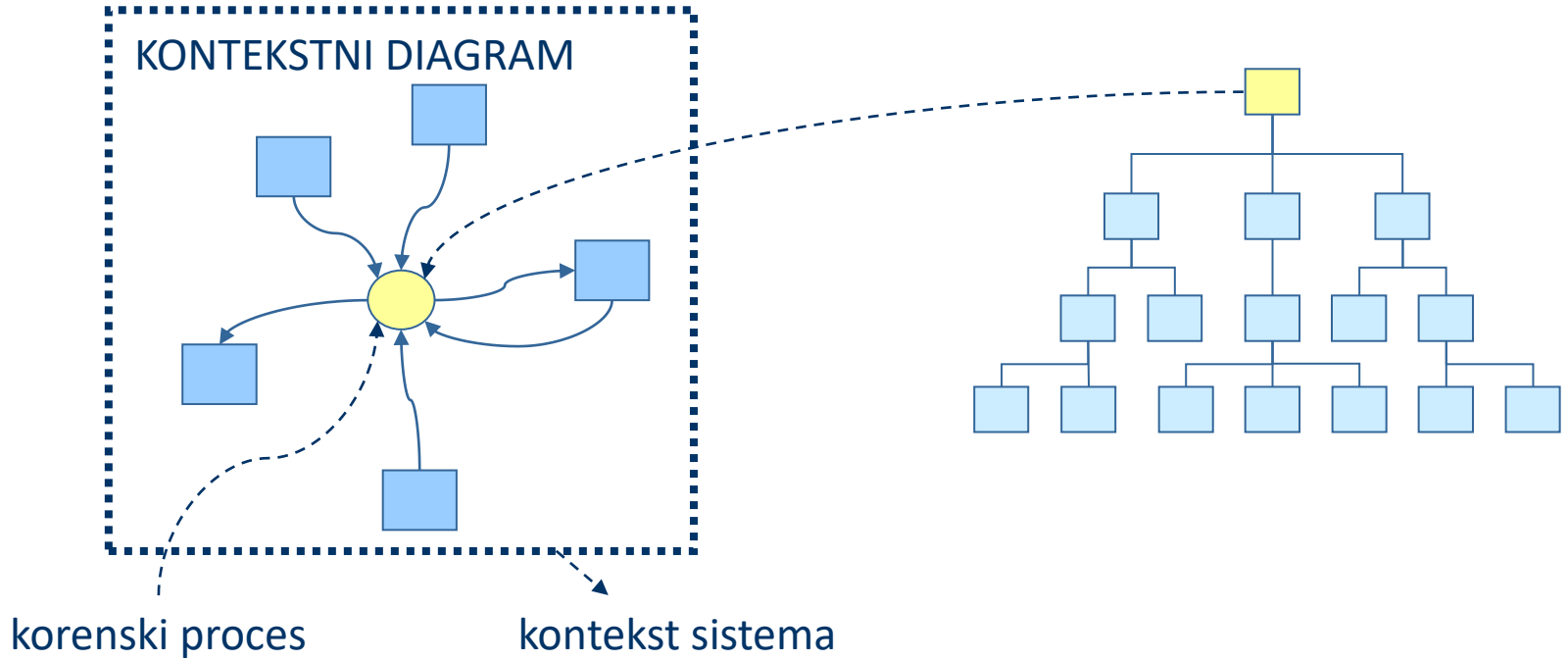
Enostaven primer DFD

- Zelo poenostavljen primer za pošiljanje elektronske pošte:



Kontekstni diagram

- Izdelavo diagrama podatkovnih tokov začnemo z enim samim procesom (korenskim procesom) na najvišjem nivoju → dobimo kontekstni diagram.

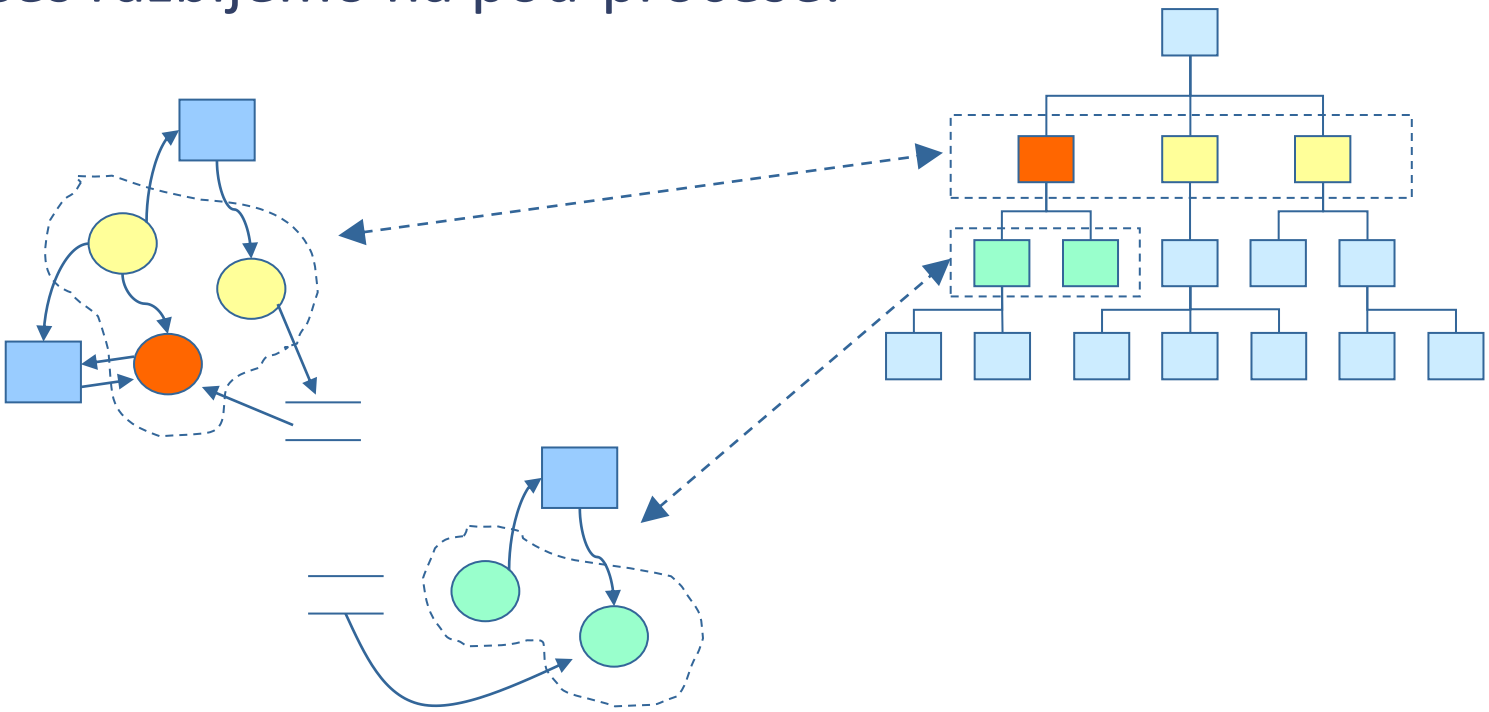


Kontekstni diagram

- Značilnosti kontekstnega diagrama:
  - Kontekstni diagram prikazuje kontekst sistema, sistem v sodelovanju z okoljem – kateri podatki prihajajo v naš sistem (in od koga) ter katere podatke pošiljamo iz sistema (in komu).
  - Kontekstni diagram ima en sam proces – korenski proces.
  - Kontekstni diagram nima podatkovnih shramb. Shrambe so namenjene odlagališču podatkov pri prenosu le-teh med procesi. Podatkovna shramba je del sistema!
  - Podatkovni tokovi med korenskim procesom in zunanjimi izvori/ponori opredeljujejo vmesnike med sistemom in okoljem.

Razčlenjevanje DFD

- Osnova za razčlenjevanje DFD je diagram funkcionalnosti.
- Za vsak proces, ki je predstavljen v diagramu na višjem nivoju, izdelamo poseben diagram podatkovnih tokov, kjer proces razbijemo na pod-procese.



Več nivojev DFD

- Sistemski (prvi) nivo diagrama podatkovnih tokov
  - Sistemski nivo (prvi nivo razčlenitve kontekstnega diagrama) predstavlja diagram podatkovnih tokov na hierarhičnem nivoju 1.
  - Diagram podatkovnih tokov na prvem hierarhičnem nivoju prikažemo z eno sliko, kjer je proces, predstavljen v kontekstnem diagramu, razčlenjen na potrebno število procesov (priporočljivo 7 ± 2).
  - Pri členjenju procesa je potrebno ohraniti vso funkcionalnost, kar pomeni, da je vsota funkcionalnosti vseh podrejenih procesov enaka funkcionalnosti nadrejenega procesa.

Več nivojev DFD

- Na naslednjem nivoju razčlenimo vsak posamezen proces s prejšnjega nivoja.
- Razčlenjevanje je smiselno do nivoja procesov, ki jih lahko opišemo z zaporedjem razumljivih korakov – elementarni procesi.
  - Za opis procesov na najnižjem nivoju diagramska tehnika DFD ni najbolj primerna, zato se uporabljajo druge tehnike, ki so del modeliranja procesne logike.

Vaja dela… (3/4)

- Izdelajmo diagram DFD za sistem **MojCD** (za katerega smo že izdelali ERD in diagram funkcionalnosti).
  - Izdelajmo kontekstni nivo, prvi sistemski nivo in po potrebi še posamezne diagrame na 2. nivoju razgradnje!

- *Namig: seveda tudi tokrat koristno uporabimo že izdelan diagram primerov uporabe, poleg tega pa še ERD in diagram funkcionalnosti!*

Model procesne logike

Minispecifikacije

Minispecifikacije

- Pri modeliranju sistema uporabimo minispecifikacije za opis procesne logike znotraj identificiranih procesov na nižjih nivojih DFD.
- Minispecifikacije služijo kot osnovni opis izvajanja aktivnosti v posameznih procesih.
- Za zapis minispecifikacij lahko v praksi uporabimo različne oblike zapisov:
  - strukturirani jezik
  - odločitvene tabele
  - odločitvena drevesa
  - diagrami prehajanja stanj

Strukturirani jezik

- Strukturirani jezik je zelo učinkovita in najbolj uporabljana metoda za zapis minispecifikacij.
- Uporablja omejen slovar in omejeno sintakso.
- Lastnost strukturiranosti mu daje oblika, temelječa na zamikih, s katerimi so nakazani tisti deli opisa, ki pomensko sodijo skupaj.
- Podpira tri osnovne proceduralne konstrukte:
  - zaporedje ukazov
  - iteracija
  - pogojni stavek

Zaporedje ukazov

- Zaporedje ukazov uporabimo za zapis aktivnosti, ki se izvajajo zaporedno.
- Oblika:

```
začetek
   stavek 1
   stavek 2
   ...
   stavek n
konec
```

Iteracija

- Iteracijo uporabimo za zapis zaporednih aktivnosti, ki naj se večkrat ponovijo.

```
ponavljaj x krat
  stavek 1
  ...
  stavek n
konec ponavljanja
```

```
ponavljaj
  stavek 1
  ...
  stavek n
dokler (pogoj) ni izpolnjen
```

```
dokler velja (pogoj) ponavljaj
  stavek 1
  ...
  stavek n
konec ponavljanja
```

Pogojni stavek

- Pogojni stavek uporabimo za zapis aktivnosti, ki naj se izvedejo le ob določenem pogoju.

```
če (pogoj) potem
  stavek 1
  ...
  stavek n
sicer
  stavek 1
  ...
  stavek n
konec pogoja
```

```
izbira
  izbor 1 (pogoj 1)
    stavek
    ...
  ...
  izbor n (pogoj n)
    stavek
    ...
  ostalo (nobeden pogoj ni izpolnjen)
    stavek
    ...
konec izbire
```

Vaja dela… (4/4)

- V sistemu **MojCD** želimo izpisati seznam vseh CDjev, ki so trenutno izposojeni.
  - Napišimo minispecifikacije za tak proces!

- *Namig: pomagamo si lahko s scenarijem iz pripadajočega primera uporabe (če ga seveda imamo izdelanega)!*