ULOGA SPECIFIKACIJA OPERACIJA #1:

Sa gledišta analize, specifikacija operacija kreira se u tački kad razumevanje analize nekog aspekta domena aplikacije može biti vraćeno nazad (feed-back) korisnicima, obezbeđujući da predlozi odgovaraju potrebama korisnika.

Sa gledišta projekta, operacijska specifikacija je okvirni rad za detaljniju specifikaciju projekta koja kasnije vodi programera kroz metodu koja je odgovarajuća implementacija operacije u kodu.

2

Uloga specifikacija operacija #2:

Operacijska specifikacija, takođe, može se koristiti da proveri da li metoda zaista odgovara svojim specifikacijama, što zapravo opisuje šta su korisnici nameravali, te dakle proverava da li su zahtevi implementirani.

3

Kada treba specificirati operacije?

• Kodirati relativno mali pod-zadatak u velikom sistemu zahteva izvesno razumevanje načina na koji će taj pod-zadatak uticati na ostale pod-zadatke, tj. kakav će biti njihov odnos. (Ako ovakvo razumevanje još uvek nije postignuto, moraju se praviti pretpostavke, i kasnije ovo može ispasti neodgovarajuće, čak katastrofalno za ceo sistem.)

• Složenost ili veličina softverskog sistema jednom dostigne određeni *prag kada isuviše rana produkcija koda postaje ekstremno neefikasna* i veoma moguće katastrofalna.

4

Kada treba specificirati operacije?

• Objektno orijentisano programiranje je uglavnom više imuno na ovu vrstu problema nego što su ostali pristupi programiranju, ali i dalje je važno opisati logičku operaciju planiranog softvera što je pre moguće.

• Modeliranje interakcije objekata je deo ovog procesa opisivanja, jer pomaže da se odredi distribucija ponašanja među različitim klasama.

• Detaljan opis inididualnih operacija sada takođe mora biti realizovan.

5

Koje operacije specificirati?

• Rumbaugh (1991) predlaže da samo operacije koje su 'računski interesantne' ili 'netrivijalne' treba specificirati.

• Allen i Frost (1998) predlažu specifikaciju svih operacija, iako nivo detaljisanja može varirati prema predviđenim potrebama projektanta.

• Drugi pristup je preporučljiv zbog problema koji mogu narasti kasnije u projektu ako nije obezbeđena puna dokumentacija. Važno je držati se makar minimalnih standarda dokumentacije, čak i za operacije koje su veoma jednostavne.

6

Uloga specifikacija:

• Svaka operacija ima određeni broj karakteristika, koje treba specificirati u fazi analize.

• Korisnici moraju potvrditi logiku, ili pravila ponašanja.

• Projektant i programer odgovorni za klasu biće glavni korisnici specifikacije jer oni moraju da znaju šta koja operacija treba da radi: da li će obavljati računanje, ili će transformisati podatke, ili će odgovarati na pitanja?

7

Dakle, kada specificirati?

• Definisanje operacija niti bi trebalo početi isuviše rano, niti bi ih trebalo ostaviti za mnogo kasno.

•Za bilo koji zadati deo modela, veoma je važno kreirati sve operacijske specifikacije pre ulaženja u *aktivnost projektovanja objekta*.

8

UGOVORI:

- Analog legalnih ili komercijalnih ugovora između ljudi ili organizacija.
- Potpisivanje ugovora uključuje obavezu da se dostave definisane usluge prema utvrđenim standardima.
- Jezikom teorije sistema, ugovor je interfejs između dva sistema.

9

Definisanje ugovora:

- Ugovor definiše ulaze i izlaze sistema kao crne kutije svojim skrivenim sporednim detaljima. Koji detalji se smatraju sporednim uvek je stvar izbora i bilo koji ugovor može specificirati da neki detalji implementacije moraju biti vidljivi drugim sistemima.
- Specifikacija pomoću ugovora znači da su operacije definisane prvenstveno pojmovima usluga koje obavljaju, i 'plaćanja' koje primaju (obično samo potpisom tj. signaturom operacije).

U objektno orijentisanom modeliranju identifikuje se priroda usluge koju obezbeđuje server-objekat i šta mora biti obezbeđeno od strane klijent-objekta da bi se dobila usluga.

10

Aspekti specifikacije operacije:

- Cilj ili svrha operacije
- Potpis operacije uključujući povratni tip (verovatno ostvaren za vreme interaktivnog modeliranja)
- Odgovarajući opis logike (naredni odeljci prikazuju neke alternativne načine opisivanja logike operacije)
- Ostale tražene operacije, bilo da su vezane za isti ili druge objekte
- Događaji preneseni na druge objekte
- Atributi postavljeni za vreme implementacije operacije
- Odgovor na izuzetke (npr. šta bi trebalo da se desi ako su parametri netačni)
- Bilo koji nefunkcionalni zahtevi koji se primenjuju

11

OPISIVANJE LOGIKE OPERACIJE:

1. Postoje operacije koje imaju sporedne efekte. Mogući sporedni efekti uključuju kreiranje ili destrukciju objektnih uzoraka, postavljanje ili vraćanje atributnih vrednosti, formiranje ili prekidanje veza sa drugim predmetima, izvršavanje računanja, slanje poruka ili događaja drugim objektima ili bilo koja kombinacija gore navedenih. Kompleksna operacija može izvršavati nekoliko ovih stvari i, dok je zadatak sasvim složen, operacija može takođe zahtevati saradnju nekoliko drugih objekata. Ovo je delimično razlog zbog koga identifikujemo oblik saradnje objekta pre detaljnog specificiranja operacija.

2. Postoje operacije koje nemaju sporedne efekte. Ovo su čisti upiti: oni zahtevaju podatke, ali ne menjaju ništa unutar sistema.

12

Apstraktne i konkretne operacije:

Kao i klase, operacije takođe mogu imati osobinu da su ili *apstraktne* ili *konkretne.*

Apstraktne operacije imaju formu koja se sastoji najmanje od signature, ponekad pune specifikacije, ali one neće imati implementaciju (neće imati metod). Tipično, apstraktne operacije su locirane u apstraktnim superklasama hijerarhije nasledja. One se uvek prebrišu konkretnim metodama u konkretnim potklasama.

Ne-algoritamski pristup opisu logike operacije:

Ne-algoritamski pristup koncentriše se na opisivanje logike operacije kao crne kutije. U objektno-orijentisanom sistemu uglavnom se ovo preferira. Pristup posebno koristan gde se, recimo, donosi struktuirana odluka, i uslovi koji određuju rezultat su lako identifikovani, ali stvarni niz koraka u postizanju odluke je nevažan.

Koriste se nealgoritamske tehnike kao što su *tabele odluka* i *parovi preduslova i postuslova*.

Tabele odluke:

Tabela odluke je matrica koja pokazuje *uslove* pod kojim se donosi odluka, *akcije* koje se mogu pojaviti kao rezultat i kako je ovo dvoje povezano.

Uslovi i akcije	Pravilo 1	Pravilo 2	Pravilo 3
Uslovi			
Da li će budžet biti premašen?	Ne	Da	Da
Da li će prelazak granice budžeta biti veći od 2%	-	Ne	Da
Akcije			
Bez akcije	X		
Pošalji pismo		X	X
Zakaži sastanak			X

Tabela opisana pomoću teksta:

Pravilo 1 **Ako budžet nije prekoračen, ništa se ne preduzima.**

Pravilo 2 **Ako je budžet prekoračen i ako izgleda da neće biti više od 2% prekoračenja, šalje se pismo.**

Pravilo 3 **Ako je budžet prekoračen i prekoračenje ima izglede da bude veće od 2%, šalje se pismo i zakazuje se sastanak.**

Preduslovi i postuslovi:

Kao što i samo ime govori, ova tehnika se koncentrise na davanje odgovora na sledeća pitanja:

- ***Koji uslovi moraju biti zadovoljeni pre nego što započne operacija?***
- ***Koji su to uslovi koji se mogu primeniti (u kom stanju može biti sistem) posle završene operacije?***

17

Preduslovi i postuslovi:

Specifikacija mora da bude dovoljno detaljna. Mora proći dva testa:

- **Korisnik treba biti u stanju da proveri da to pravilno izražava poslovnu logiku**
- **Dizajner klase mora biti u stanju da proizvede detaljan dizajn operacije za programera koji će da kodira.**

18

Algoritamski pristup opisu logike operacije:

Algoritam **opisuje unutrašnju logiku procesa ili odluke razbijanjem na male korake.**

Nivo detalja do kojeg se ovo radi varira u velikoj meri, u zavisnosti od dostupnih informacija u trenutku definisanja i razloga za definisanje.

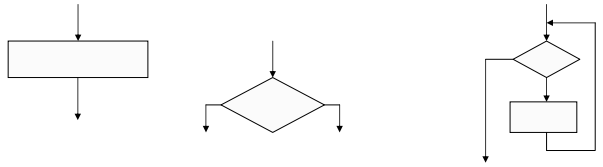
Algoritam takođe specificira niz u kom se koraci odvijaju.

Na polju računarskih i informacionih sistema, algoritmi se koriste ili kao *opis* načina na koji se zadatak za programiranje izvršava (njihova upotreba u specifikaciji operacije), ili kao *propis* za program kojim se automatizuje zadatak.

19

Kontrolne strukture u algoritmima:

Algoritmi su generalno organizovani proceduralno, što će reći da oni koriste osnovne kontrolne strukture programiranja: niz, selekcija i iteracija.



20

Izračunavanje prekoračenja budžeta kampanje:

Jedan mogući niz koraka, na veoma grubom nivou detaljisanja, uključivao bi nekoliko sledećih koraka:

1. sabrati sve individualne troškove reklama;
2. pomnožiti sumu stopom prekoračenja;
3. rezultujuća suma je ukupna cena kampanje.

21

Upotreba struktuiranog engleskog:

- 'Dijalekt' pisanog engleskog koji je negde na pola puta između svakodnevnog govora i formalnog programskog jezika.
- Kada je potrebno specifikovati operaciju proceduralno, ovo je najkorisnija i najprilagodivljivija tehnika.
- Njene prednosti uključuju mogućnosti, zadržavanja mogućnosti čitljivosti i razumevanja svakodnevnog engleskog.
- Dozvoljava konstrukciju formalne logičke strukture koja se lako prevodi na programski kod.
- Struktuirani engleski je lako pisati iterativno, na većim nivoima detaljisanja, i lako se rasklapa na komponente koje mogu biti ponovo sastavljene u različite strukture bez mnogo ponovnog rada.

22

Upotreba struktuiranog engleskog:

- Logička struktura je napravljena kao eksplicitna, kroz upotrebu ključnih reči i uvlačenja, dok se vokabular drži što više svakodnevnog govora u poslovnom kontekstu.
- Izrazi i ključne reči koje su specifične za neki posebni programski jezik izbegavaju se.
- Idealno, rezultat je nešto što ne-tehnički korisnik može da razume, prepravi ili odobri, po potrebi, dok bi to takođe trebalo da bude korisno za projektante. To znači da mora biti pogodno da se dalje razvija u detaljan programski dizajn bez poteškoća.

23

if-then-else konstrukcija:

Na primer, *if-then-else* konstrukcija koja ima samo dva moguća izlaza, prikazana je u sledećem fragmentu:

```
if client contact is 'Sushila'
   set discount rate to 5%
else
   set discount rate to 2%
end if
```

24

case konstrukcija:

```
begin case
   case client contact is ‘Sushila’
      set discount rate to 5%
   case client contact is ‘Wu’
      set discount rate to 10%
   case client contact is ‘Luis’
      set discount rate to 15%
   otherwise
      set discount rate to 2%
end case
```

25

Ugnježđena if-then-else konstrukcija:

```
if client contact is ‘Sushila’
   set discount rate to 5%
else
   if client contact is ‘Wu’
      set discount rate to 10%
   else
      if client contact is ‘Luis’
         set discount rate to 15%
      else
         set discount rate to 2%
      end if
   end if
end if
```

26

Petlja sa testom pre ulaska:

```
do while there are more staff in the list
   calculate staff bonus
   store bonus amount
end do
```

27

Petlja sa testom posle izlaska:

```
repeat
   allocate member of staff to campaign
   increment count of allocated staff
until count of allocated staff = 10
```

28

Pseudo-kod:

Pseudo-kod se razlikuje od struktuiranog engleskog po tome što je bliži vokabularu i sintaksi određenog programskog jezika.

Mada pseudo-kod liči na programski jezik po svojoj sintaksi, ipak zahteva dalji rad da bi se pretvorio u programski kod.

29

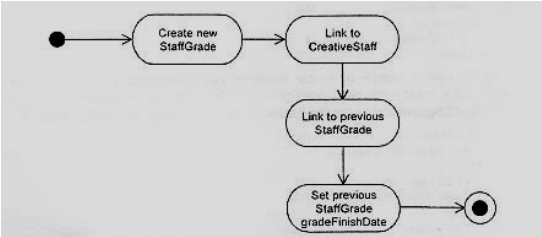
Primer pseudo-koda koji liči na C:

```
{
   { while more adverts:
      next advert;
      get advertcost;
      cumcost = cumcost + advertcost;
    endwhile;
    }
    { campaigncost + cumcost x ohrate;
     get campaignbudget;
     case campaigncost >= campaignbudget:
        return warningflag;
     endcase
     }
}
```

30

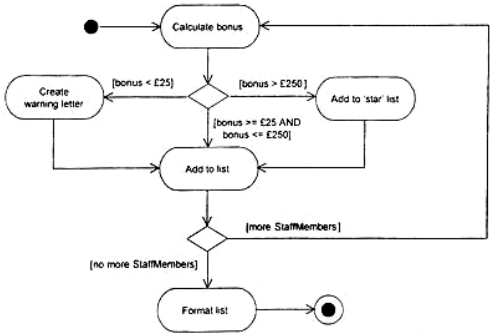
Dijagrami aktivnosti:

- **Dijagrami aktivnosti mogu se koristiti za specifikaciju logike proceduralno složenih operacija.**
- **Kad se koriste u ove svrhe, stanja aktivnosti u dijagramu obično predstavljaju korake u logici operacije.**



31

Primer:



32

JEZIK OBJEKTNIH OGRANIČENJA (OCL)

Izrazi u OCL (Object Constraint Language) su konstruisani iz skupa pre-definisanih elemenata i tipova, i jezik ima preciznu gramatiku koja omogućava konstrukciju nedvosmislenih iskaza o osobinama komponenata modela i njihove međusobne povezanosti.

Struktura OCL iskaza:

Većina OCL iskaza sadrži sledeće strukturne elemente:

- ***Kontekst*, koji definiše domen unutar koga izraz važi. Ovo je često primer specifičnog tipa, na primer objekat u klasnom dijagramu. Link (primer asocijacije) takođe može biti kontekst za OCL izraz.**
- ***Svojstvo* onog uzorka koji je kontekst izraza. Svojstva mogu uključivati atribute, asocijacije i operacije upita.**
- **OCL *operacija* koja se primenjuje na svojstvo. Operacije uključuju (ali nisu ograničene na to) aritmetičke operatere *, +, -, i /, set operatere kao što su `size, isEmpty i select,` i operatore tipa kao što je `oclIsTypeOf`.**

Ključne reči u OCL-u:

OCL iskazi takođe mogu uključivati OCL ključne reči koje uključuju logičke operatore kao što su

`and, or, implies, if, then, else, not`

i set operator `in`,
štampani podebljano da bi se razlikovali od drugih OCL termina i operacija. Zajedno sa ne-ključnim operacijama prethodno pomenutim, mogu se koristiti za definisanje sasvim kompleksnih preduslova i postuslova za operaciju.

Sintaksa specifikacije operacije u OCL-u:

OCL može specificirati mnoga ograničenja koja ne mogu biti izražena direktno u vidu dijagrama i stoga je koristan kao precizni jezik za preduslove i postuslove.

Generalna sintaksa za operacijske specifikacije je sledeća:

```
Type::operation(paramter1:type,parameter2:type):return type
   pre: parameter1 operaton
          parameter2 operation
      post:result = ...
```

Sintaksa specifikacije operacije u OCL-u #2:

Treba primetiti da kontekstni type je tip (za naše svrhe, normalno klase) koji poseduje operaciju kao karakteristiku.

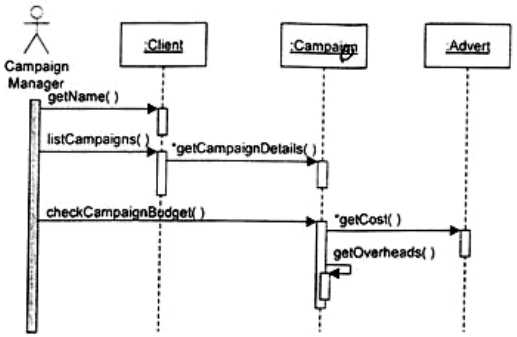
`pre`: izrazi su funkcije operacijskih parametara, dok

`post`: izrazi su funkcije njih samih, operacijskih parametara ili oboje.

OCL izrazi mogu biti napisani sa explicitnom context deklaracijom.

KREIRANJE SPECIFIKACIJE OPERACIJE:

Primer: provera prekoračenja budžeta kampanje:



Primer: `Campaign.checkCampaignBudget ( )`

Zadatak `checkCampaignBudget` za realizaciju poziva operaciju `Campaign.checkCampaignBudget ( )`.

Specifikacija za `Campaign.checkCampaignBudget ( )` je data nadalje.

Font Arial označava strukturu specifikacije,

dok `Courier oznacava kontekst`.

Komentari su italic.

Specifikacija operacije #1:

Operation specification: `checkCampaignBudget`

Operation intent: `return difference between campaign budget and actual costs.`

Specifikacija operacije #2:

operation signature: `Campaign::checkCampaignBudget ( )`
`budgetCostDifference:Money`

logic description (pre- and post-conditions):

```
pre:self -> exists
post:campaignBudget = self.estimatedCost
     committedExpenditure =
          self.adverts.estimatedCost->sum
```


Specifikacija operacije #3:

Kao što se može videti iz sekvencijalnog dijagrama, ova operacija poziva dve druge operacije i one moraju biti nabrojane. U punoj specifikaciji, bille bi zapisane pune signature, ali ovde su izostavljene.

Other operations called: `Advert.getCost,`
`self.getOverheads`

Events transmitted to other objects: none

Specifikacija operacije #4:

Attributes set: none

Ovo je operacija upita čija je jedina svrha da vrati podatke koji su već sačuvani unutar sistema.

Response to exceptions: none `defined`

Ovde bismo mogli definisati kako bi operacija trebala reagovati na uslove grešaka, npr. koja vrsta poruke za greške će biti vraćena ako se pozivajuća poruka koristi sa nevažećim potpisom (signaturom).

Non-functional requirements: none `defined`

Specifikacija operacije #5:

Nekoliko nefunkcionalnih zahteva može biti asocirano sa operacijom, ali moraju biti određeni kroz diskusiju sa korisnicima. Mogu uključivati, na primer, vreme reagovanja pod stvarnim uslovima (često upiti zahtevaju brži odgovor) ili format izlaza (recimo ako postoji kućni standard da se prekoračenje budžeta prikazuje u crvenoj boji). Međutim, ovo su prave *projektantske* teme i biće zabeležene na ovom nivou samo ako se desi da su informacije dostupne u tom trenutku.

ZAKLJUČAK:

- **Specifikacije operacija su najdetaljniji opis ponašanja modela sistema.**
- **Predstavlja važnu vezu između korisnika i projektanata i programera.**
- **Tačna specifikacija operacija je kritično važna za korektno kodiranje softvera.**
- **Ugovor je okvir za specificiranje operacije pomoću odnosa usluga između klasa.**
- **Nealgoritamske tehnike: tabele odluke i parovi preduslova i postuslova.**
- **Algoritamske tehnike: struktuirani engleski, pseudo-kod i dijagrami aktivnosti.**
- **OCL (UML's Object Constraint Language) specificira mnogo tipova ograničenja na način formalnih jezika.**

45