STANJA I DOGAĐAJI:

Svi objekti imaju *stanje*.

***Tekuće stanje je rezultat događaja* koji su se pojavili objektu i određeno je tekućim vrednostima atributa objekta i linkovima koje ima sa drugim objektima.**

Neki atributi i linkovi jednog objekta su značajni za određivanje njegovog stanja, a drugi nisu.

UML definicija stanja:

Stanje je okolnost u toku života jednog objekta ili interakcija u toku koje on zadovoljava neke uslove, ostvaruje neku akciju ili čeka neki događaj. ...Konceptualno, objekat ostaje u jednom stanju u nekom vremenskom intervalu. međutim, semantika dopušta modelovanje 'protočnih' stanja koja su trenutna, kao i prelaze koji nisu trenutni.

Moguća stanja:

- Moguća stanja koja jedan objekat može imati ograničena su klasom.
- Objekti nekih klasa imaju samo jedno moguće stanje.

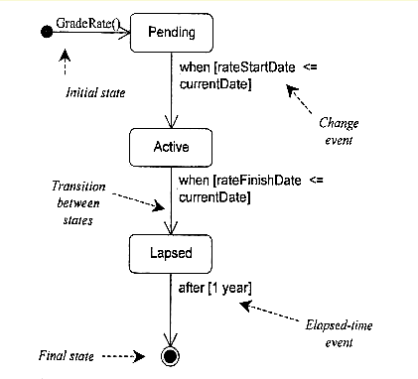
Kretanje iz jednog stanja u drugo naziva se *prelazak* i on je izazvan nekim *događajem*.

Kada se pojavi okidajući događaj, kaže se da je *prelazak upaljen*.

5

Dijagram stanja:

Prelazak se predstavlja dijagramom stanja:



6

Događaj i prelazak:

- *Prelazak se predstavlja punom linijom sa strelicom od izvornog ka ciljnom čvoru.*
- Događaj je pojava stimulusa koji može da okine promenu stanja i koji je značajan objektu ili aplikaciji.
- Kao što se skup objekata definiše klasom čiji su oni uzorci, događaji se definišu *tipom događaja* čije uzorke predstavljaju događaji.
- Ono što se modeluje zapravo su tipovi događaja, ali jednostavno ih zovemo događajima.
- Događaj može imati parametre i vraćenu veličinu, i u OO sistemu on se implementira porukom.

7

Tipovi događaja #1:

Događaj promene javlja se kada se ostvari (postane *true*) uslov. Označavaju se *ključnim rečima* kada iza njih sledi Bool-ov izraz u zagradama.

Pozivni događaj javlja se kada objekat primi poziv za jednu od njegovih operacija ili od drugog objekta ili od sebe . Pozivni događaji odgovaraju pozivnim porukama i označavaju se signaturom operacije kao okidačem prelaska.

8

Tipovi događaja #2:

Signalni događaj **javlja se kada objekat primi signal (asinhronu komunikaciju). Označava se signaturom pozvane operacije..**

Između pozivnih i signalnih događaja nema sintaktičke razlike. Podrazumeva se da će se razlikovati po imenima.

Događaj isteklog vremena **izazvan je istekom projektovanog vremena posle specificiranog događaja. Predstavljaju se vremenskim izrazima unutar zagrada, a prethodi im ključna reč *after* i ako nije navedeno startno vreme, odnosi se na poslednji ulazak u tekuće stanje.**

9

Osnovna sintaksa poziva ili signalnog događaja:

`event-name '(' parameter-list ')'`

gde lista parametara ima oblik:

`parameter-name ':' type-expression`

odvojenih zarezom.

10

OSNOVNA NOTACIJA:

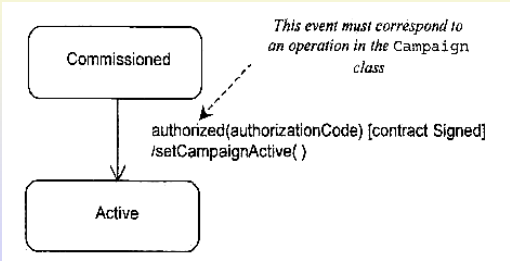
Inicijalno stanje **(startna tačka) životnog ciklusa obeležava se malim popunjenim krugom.**

Krajnja tačka **životnog ciklusa (konačno stanje) predstavlja se okom bika.**

Sva *ostala stanja* predstavljaju se pravougaonikom sa zaobljenim uglovima i trebalo bi da su označeni imenom.

11

Prelazak:



12

Format prelaska za pozivne i signalne događaje:

```
signatura događaja '('vodeći-uslov')' '/' izraz-akcije
```

Signatura događaja ima formu:

```
ime-događaja '('lista-parametara')'
```

Vodeći uslov je Bulov izraz koji se izračunava u trenutku paljenja događaja.

Izraz–akcije izvršava se kada neki događaj izazove paljenje prelaza.

13

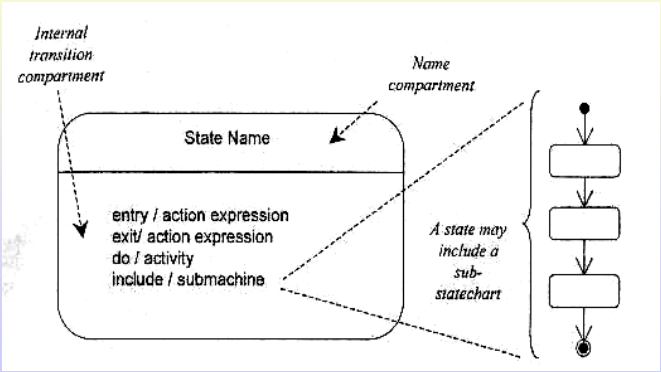
Unutrašnje (interne akcije)

Izrazi akcije mogu biti pridruženi i stanju, ne samo prelazu. One mogu biti okinute događajima koji ne menjaju stanje, ili izazivaju ulazak u stanje, ili događajem koji rezultuje u izlasku iz stanja.

Interni događaji mogu imati dva tripa: *ulazni događaj* i *izlazni događaj*. Oni se opisuju ključnim rečima *entry* i *exit*, respektivno.

14

Unutrašnje akcije i aktivnosti za jedno stanje:



15

Sintaksa aktivnosti:

Aktivnosti prethodi ključna reč *do*:

```
'do' '/' ime-aktivnosti '(' lista parametara ')'
```

Može se prikazati da stanje ima podstanja korišćenjem ključne reči *include* posle koje sledi ime sadržanog poddijagrama ili podmašine.

Ugnježdavanjem dijagrama stanja moguće je predstaviti vrlo složeno ponašanje.

16

Sintaksa internog prelaska:

Generalno, interni prelaz javlja se kao odgovor na događaj i rezultuje u akciji, ali ne izaziva promenu stanja.

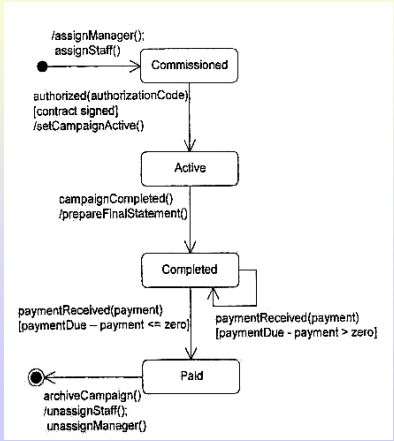
Format internog prelaza koji nije ulaz u- i izlaz iz stanja je:

```
ime-događaja '(' lista-parametara ')'
             '[' vodeći-uslov ']' '/' izraz-akcije
```

Ime događaja može se pojaviti više puta ako su vodeći uslovi različiti.

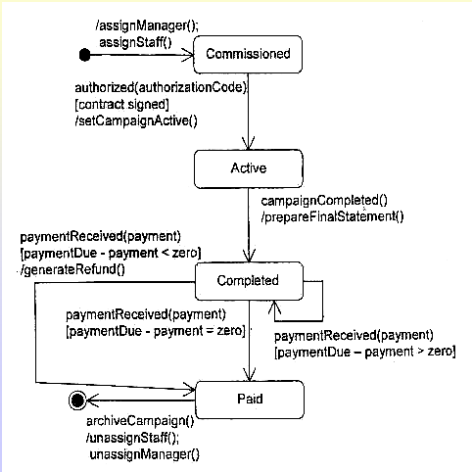
17

Dijagram stanja za klasu `Campaign`:



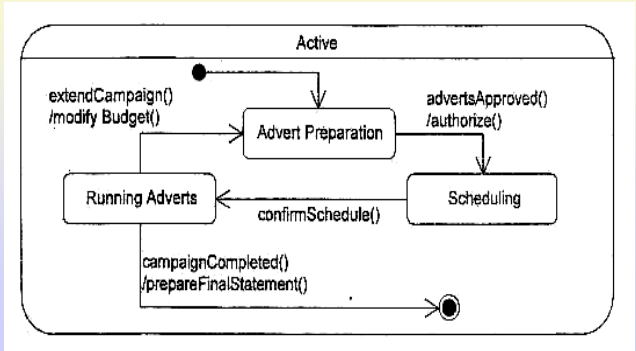
18

Dorađen dijagram stanja za klasu `Campaign`:



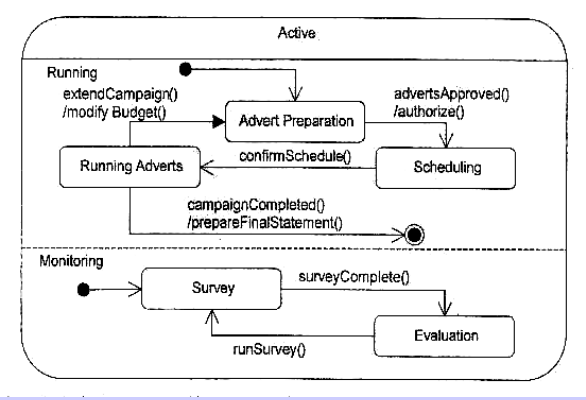
19

Stanje `Active` u dijagramu za `Campaign` koje prikazuje ugneždena stanja:



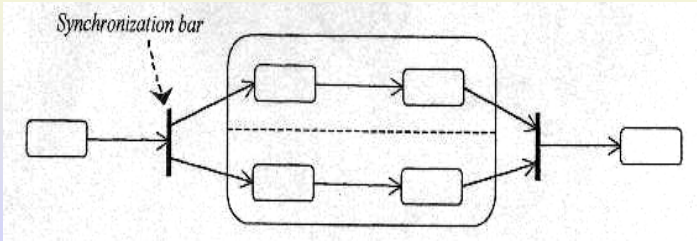
20

Stanje `Active` sa konkurentnim podstanjima:



21

Sinhronizovani konkurentni kontrolni putevi (thread):



22

Priprema dijagrama stanja:

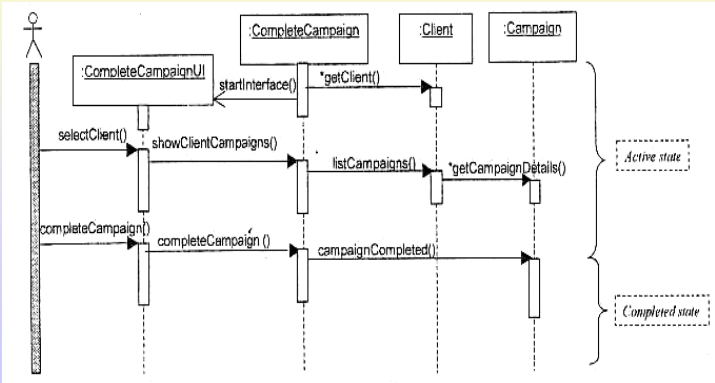
Dijagrami stanja mogu se praviti sa različitih perspektiva.

Dijagram stanja za klasu može se smatrati opisom načina na koje korisnički slučajevi mogu da utiču na objekte u klasi. Korisnički slučajevi dodaju napredak dijagramima interakcije (sekvencijalni dijagrami i dijagrami saradnje) i ovo se može koristiti kao polazna tačka kod dijagrama stanja.

Dijagrami interakcije prikazuju poruke koje prima jedan objekat u toku izvršavanja korisničkog slučaja. Prijem poruke od strane objekta ne znači obavezno događaj koji izaziva promenu stanja.

23

Sekvencijalni dijagram za korisnički slučaj `Zabeležiti kompletiranje kampanje`:



24

Pristup ponašanja – koraci u pripremi dijagrama stanja #1:

1. Ispitati sve dijagrame interakcije svake klase sa intenzivnim porukovanjem.

2. Identifikovati dolazeće poruke na svakom dijagramu interakcije koji mogu da odgovaraju događajima. Takođe, identifikovati moguća rezultujuća stanja.

3. Dokumentovati ove događaje i stanja na dijagramu stanja.

4. Obraditi dijagram stanja da prihvati nove interakcuije kad postanu evidentne i dodati izuzetke.

Pristup ponašanja – koraci u pripremi dijagrama stanja #2:

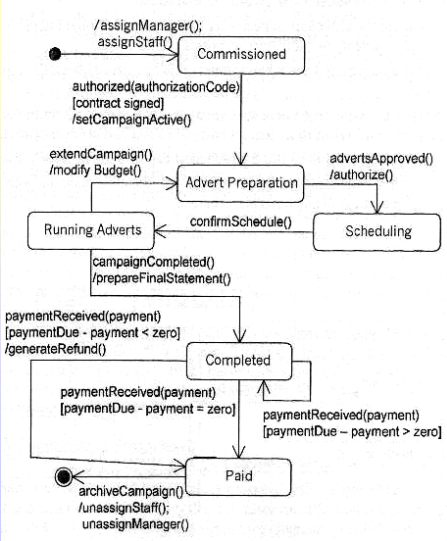
5. Razviti ugnježdene dijagrame stanja.

6. Ispitati dijagram stanja sa stanovišta konzistentnosti sa korisničkim slučajevima. Proveriti da li su ograničenja odgovarajuća.

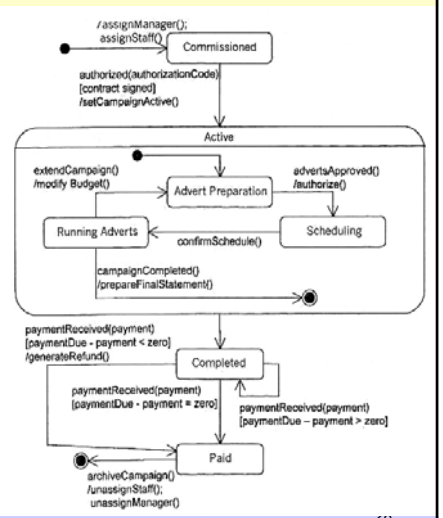
7. Iterirati korake 4, 5 i 6 dok dijagram stanja dobije neophodan nivo detaljnosti.

8. Proveriti konzistentnost dijagrama stanja sa dijagramom klasa, dijagramom interakcije i ostalim dijagramima stanja.

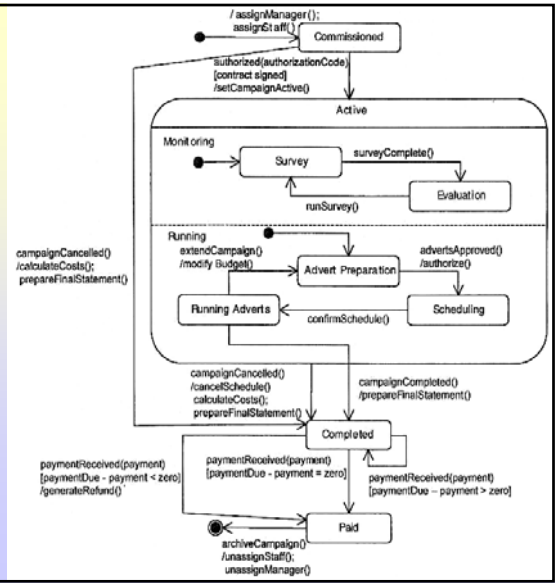
Inicijalni dijagram stanja za klasu `Campaign` – pristup ponašanja



Dorađeni dijagram stanja za klasu `Campaign`



Finalna verzija dijagrama stanja za klasu `Campaign`



Pristup ciklusa života - koraci u modelovanju stanja #1:

1. Identifikovati najveće događaje u sistemu.

2. Identifikovati svaku klasu koja ima izgleda da na ove događaje ima odgovor zavisan od stanja.

3. Za svaku od ovih klasa formirati first-cut dijagram stanja razmatranjem tipičnog ciklusa života jednog uzorka klase.

4. Ispitati dijagram stanja i obraditi da bi obuhvatio detaljnije ponašanje događaja.

30

Pristup ciklusa života - koraci u modelovanju stanja #2:

5. Poboljšati dijagram stanja uključivanjem alternativnih scenarija.

6. Ispitati dijagram stanja sa stanovišta konzistentnosti sa korisničkim slučajevima. Proveriti da li su ograničenja odgovarajuća.

7. Iterirati korake 4, 5 i 6 dok dijagram stanja dobije neophodan nivo detaljnosti.

8. Proveriti konzistentnost dijagrama stanja sa dijagramom klasa, dijagramom interakcije i ostalim dijagramima stanja.

31

Provera konzistentnosti dijagrama stanja sa drugim modelima:

•Svaki događaj treba da se pojavi kao dolazeća poruka odgovarajućeg objekta na dijagramu interakcije.

•Svaka akcija treba da odgovara izvršenju operacije na odgovarajućoj klasi, a možda i slanjem poruke drugom objektu.

•Svaki događaj treba da odgovara operaciji na odgovarajućoj klasi (treba primetiti da ne korespondiraju sve klase događajima).

•Svaka izlazna poruka poslata sa dijagrama stanja mora da korespondira operaciji na drugoj klasi.

32

Kvalitetni dijagrami #1:

•Imenovati svako stanje jedinstveno tako da odražava ono što se događa u toku njegovog trajanja.

•Ne koristiti kompozitna stanja osim ako je ponašanje stanja prirodno kompleksno.

•Ne prikazivati previše kompleksnosti na jenom dijagramu stanja.

•Vodeće uslove koristiti pažljivo da bi se obezbedilo da dijagram stanja opisuje ponašanje nedvosmisleno.

33

Kvalitetni dijagrami #2:

•Većina prelazaka pali se završetkom stanja.

•Mnoge poruke poslate samo–reflektujućem kodu koristiti ponovo umesto akcija okinutih događajima.

•Stanja ne obuhvataju ponašanja zavisna od stanja, asocirana sa klasom.

34

Zaključak:

Specifikacija kontrolnog aspekta aplikacije je važan aspekt i analize i projektovanja.

Delimično se može opisati dijagramima interakcije, ali oni se fokusiraju samo na korisničke slučajeve i operacije.

Da bi se obuhvatila potpuna ograničenja kontrole za svaku klasu, moguće je modelovati uticaj događaja na klasu i modelovati rezultujuće promene stanja sa njihovim ograničavajućim uticajem na ponašanje.

Potrebno je samo formirati dijagrame stanja za klase koje imaju promene u ponašanju zavisne od stanja.

UML-notacija dijagrama stanja omogućava konstrukciju detaljnih modela koji mogu da sadrže ugnježdena stanja i koriste konkurentna stanja za prikazivanje kompleksnog ponašanja.

35

Kraaaaaaj analizeeeeee!