Univerzitet u Novom Sadu
Fakultet tehničkih nauka

Upravljanje poslovnim procesima

4. Orkestracija procesa – prvi deo

Katedra za informatiku
nastavnik: Zarić dr Miroslav

Sadržaj

- Šabloni kontrole toka
- Petri mreže
- Procesni lanci upravljani događajima
- *Workflow* mreže
- Graf-bazirani jezici koji vode računa o zavisnosti podataka
- BPMN
- YAWL

Šabloni za kontrolu toka

- *Control Flow Patterns*
- Predstavljaju osnovni alat za izražavanje strukture – orkestraciju procesa
- Nezavisni su od konkretnog jezika za opis procesa – mogu se prikazati u različitim jezicima za opis procesa
- Poređenje podržanih šablona za kontrolu toka može poslužiti i kao sredstvo za poređenje mogućnosti različitih jezika

Osnovni šabloni za kontrolu toka

- Sekvenca
- I grananje (*and split)*
- I spajanje (*and join)*
- Ekskluzivno ILI grananje (*XOR split*)
- Ekskluzivno ILI spajanje (*XOR join*)
  - Podržavaju ih svi jezici za modelovanje procesa

Ostali šabloni za kontrolu toka

- ILI grananje (*OR split*)
- ILI spajanje (*OR join*)
- Višestruko spajanje
- Diskriminator
- N od M spajanje
- Proizvoljni ciklusi
- Implicitna terminacija

Ostali šabloni za kontrolu toka (2)

- Višestruke instance (aktivnosti)
- Odloženo odlučivanje
- Sekvencijalno izvršavanje bez prethodnog znanja u vreme dizajna procesa
- *Milestone* – referentna tačka
- Šabloni za vreme izvršavanja procesa (*runtime* šabloni)

Značenje pojmova i notacije

- Semantiku šablona posmatramo na osnovu događaja i njihovog redosleda (tj. kako određeni šablon utiče na redosled izvršavanja događaja)
- Proces posmatramo u skladu sa definicijom modela procesa:
  - P=(*N, E, type*), je model procesa koji se sastoji od skupa čvorova N, skupa grana E.
  - $N = N_A \cup N_E \cup N_G$ (skupovi modela aktivnosti, događaja i grananja)
  - E – skup grana
  - *type:* $N_G \rightarrow C$ – preslikavanje koje svakom modelu grananja (*gateway*) pridružuje određenu kontrolu toka

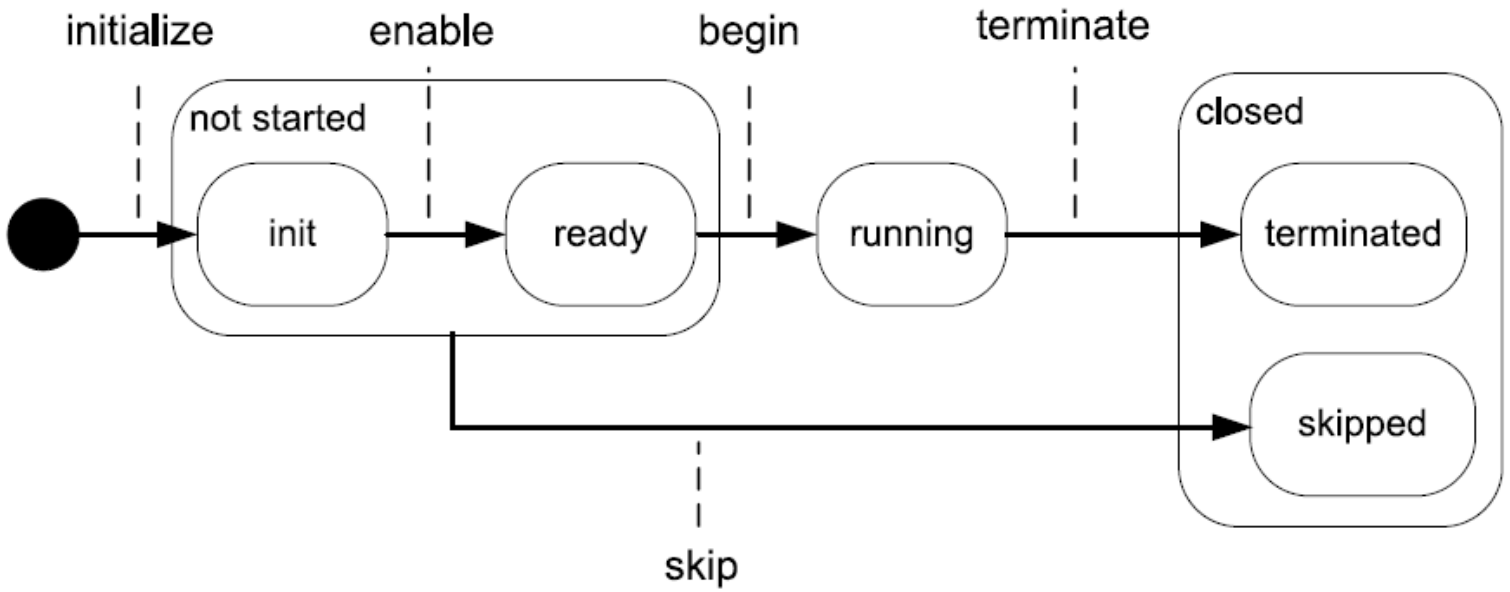
Značenje pojmova i notacije (2)

- Npr. posmatramo proces P sa modelima aktivnosti A i B, i sekvencom A → B
  - Ovakav model definiše redosled instanci aktivnosti koje su instance modela A i B u okviru jedne instance procesa
  - Instanca aktivnosti definisane modelom aktivnosti B može se pokrenuti samo onda kada je instanca aktivnosti A već završena

Značenje pojmova i notacije (3)

- Model procesa ograničava redosled događaja koji se dešavaju pri izvršavanju instanci datog procesa
- Svaki konstrukt za kontrolu toka se predstavlja grananjem (gateway-om)
  - Model aktivnosti označavamo velikim slovima (A, B...), njihove instance malim (a, b...). Ukoliko ima više instanci koriste se indeksi (a_1,a_2...)
  - Modele grananja označavamo sa G, a instance sa g

Dijagram prelaza stanja instanci aktivnosti (podsećanje)



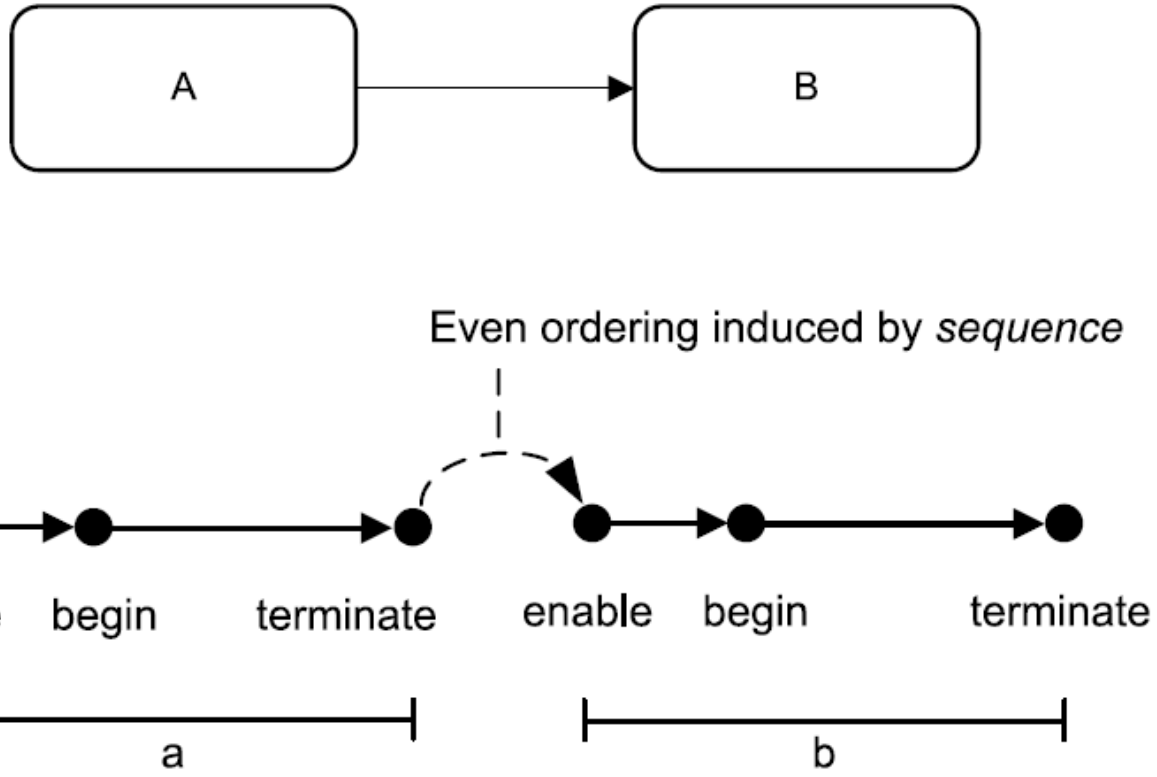
- U svim narednim primerima je
  - N_E = *{initialize, enable, begin, terminate}*

Sekvenca

- P=(*N, E, type*)
  - $A,B \in N_A$
  - $G \in N_G$
  - $E \subseteq \{(A,G),(G,B)\}$
  - *type*(G) = *Sequence*

- Sekvenca definiše redosled događaja između a i b, takav da se događaj *enable* instance b može desiti samo nakon događaja *terminate* instance a ($t_a < e_b$)

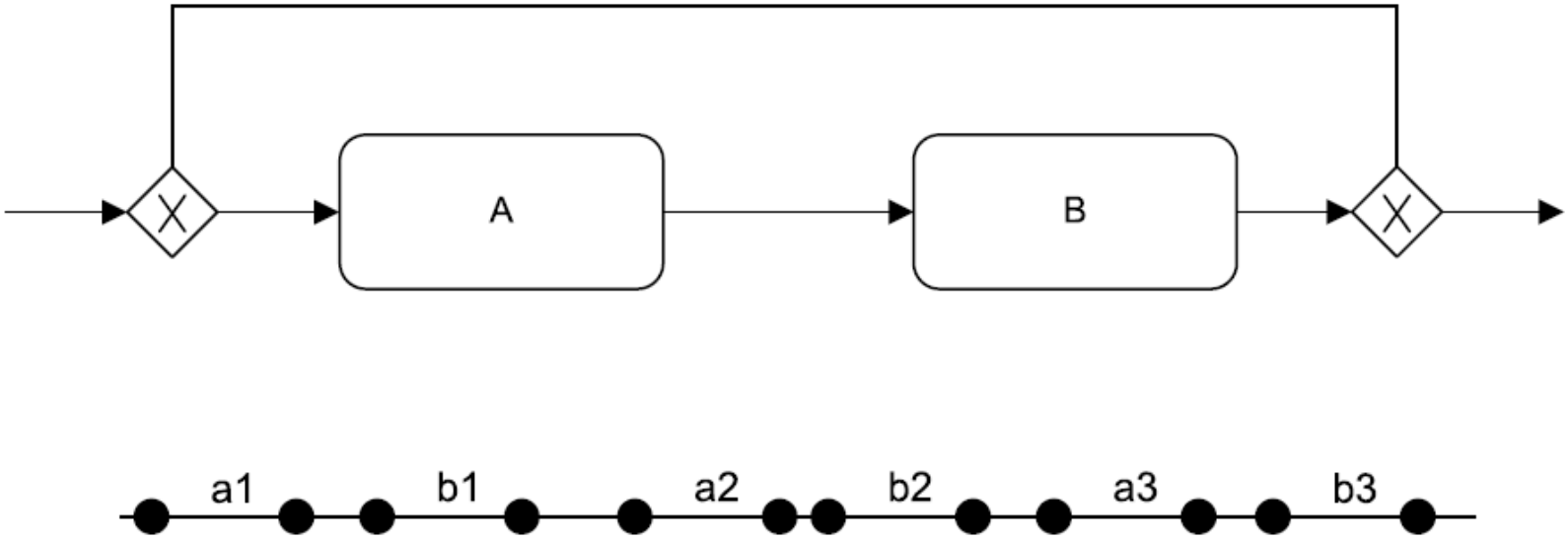
Sekvenca (2)



Sekvenca i redosled događaja

Sekvenca (3)

- Ukoliko je sekvenca deo petlje, redosled događaja mora odražavati sekvencu između pojedinih instanci aktivnosti (redosled događaja definisan sekvencom očuvan je unutar jednog ciklusa petlje)



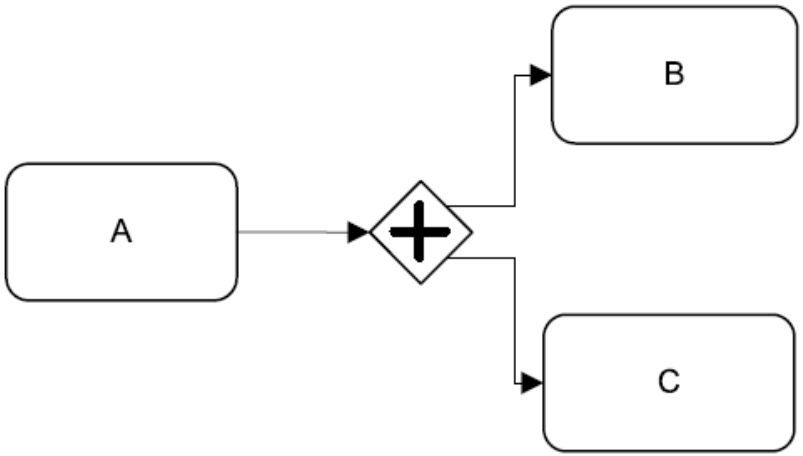
Primer: Dijagram događaja sekvence u petlji koja se izvršava 3 puta

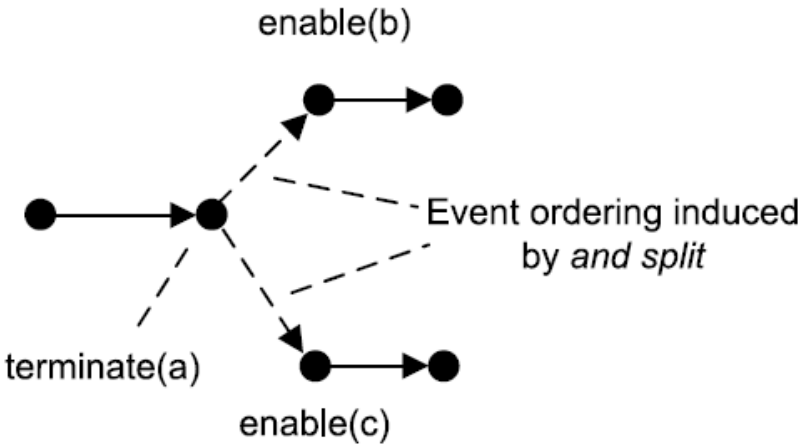
I grananje (*and split*)

- Šablon koji omogućava da se jedan tok izvršavanja multiplicira u različite konkurentno izvršive tokove.
- P=(*N, E, type*)
  - $A,B,C \in N_A$
  - $G \in N_G$
  - $E \subseteq \{(A,G),(G,B),(G,C)\}$
  - *type*(G) = *AndSplit*

- I grananje, u ovom slučaju, definiše sledeći redosled događaja: za svaki *terminate* događaj instance aktivnosti a (t_a), postoje *enable* događaji instanci b i c (e_b, e_c), koji se dešavaju nakon događaja t_a
  $t_a < e_b \wedge t_a < e_c$

I grananje


A
B
C

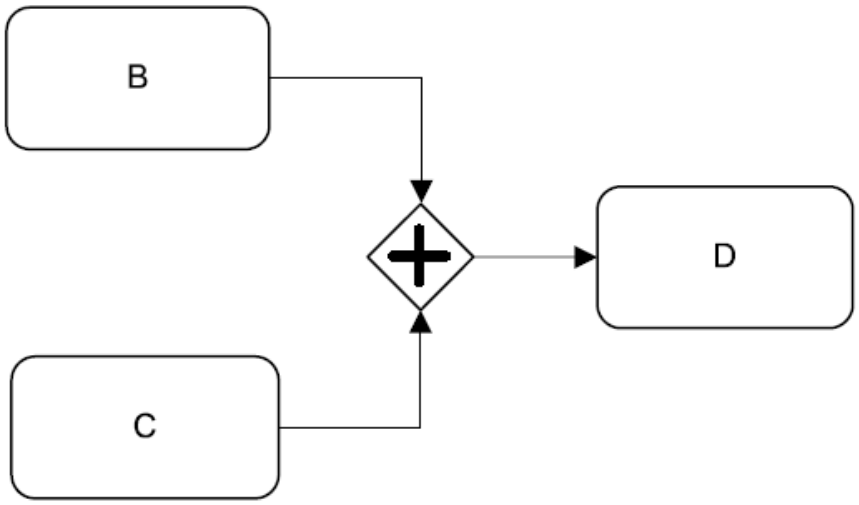

enable(b)
Event ordering induced
by *and split*
terminate(a)
enable(c)

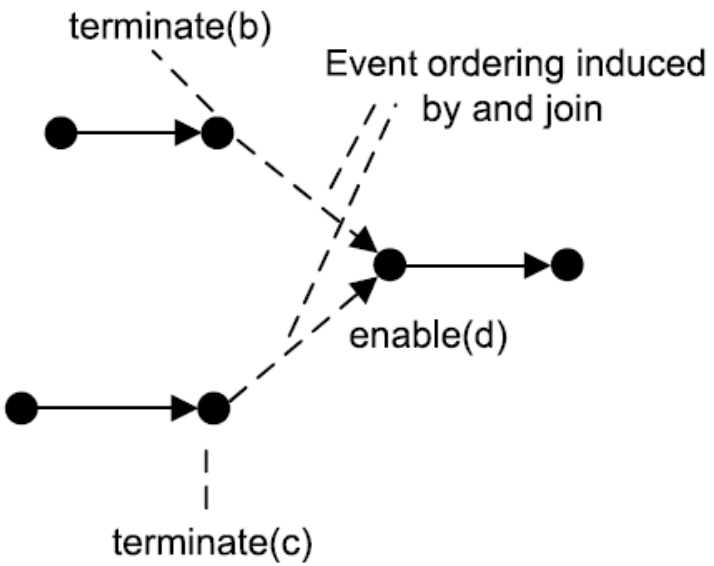
I spajanje (*and join*)

- Tačka u procesu gde se različiti konkurentno izvršivi tokovi spajaju u jedan.
- P=(*N, E, type*)
  - B,C,D $\in N_A$
  - G $\in N_G$
  - E $\subseteq$ {(B,G),(C,G),(G,D)}
  - *type*(G) = *AndJoin*

- I spajanje, u ovom slučaju, definiše sledeći redosled događaja: svakom *enable* događaju instance aktivnosti d (e_d), moraju prethoditi *terminate* događaji instanci b i c (t_b, t_c).
  $t_b < e_d \wedge t_c < e_d$

I spajanje


B
C
D

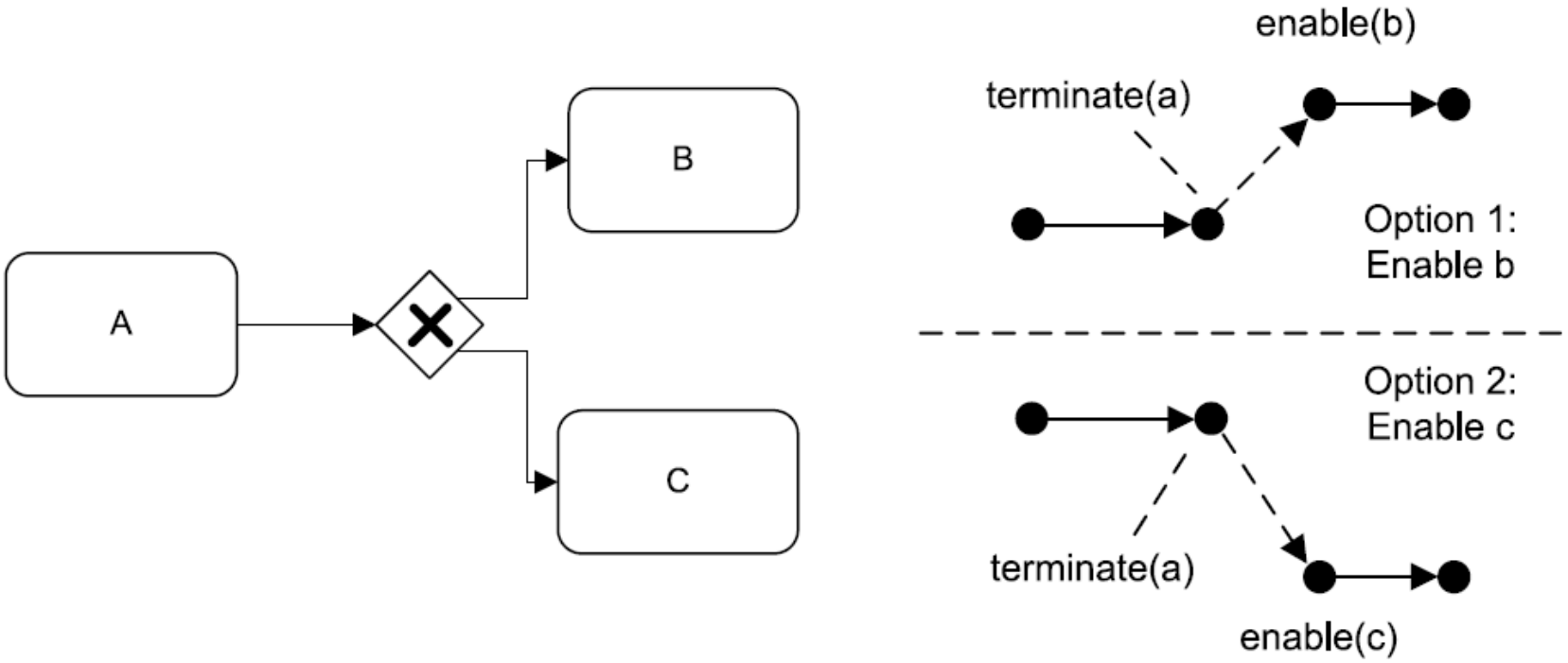

terminate(b)
Event ordering induced
by and join
enable(d)
terminate(c)

Ekskluzivno ILI grananje

- Tačka u procesu gde se bira jedna od mogućih putanja izvršavanja.
- P=(*N, E, type*)
  - A,B,C $\in N_A$
  - G $\in N_G$
  - E $\subseteq$ {(A,G),(G,B),(G,C)}
  - *type*(G) = *XorSplit*

- XOR grananje, u ovom slučaju, definiše sledeći redosled događaja: posle svakog *terminate* događaja instance aktivnosti a (t_a), desiće se *enable* događaj ili instance aktivnosti b (e_b) ili instance aktivnosti c (e_c), ali ne oba.
  $t_a < e_b \vee t_a < e_c$

Ekskluzivno ILI grananje

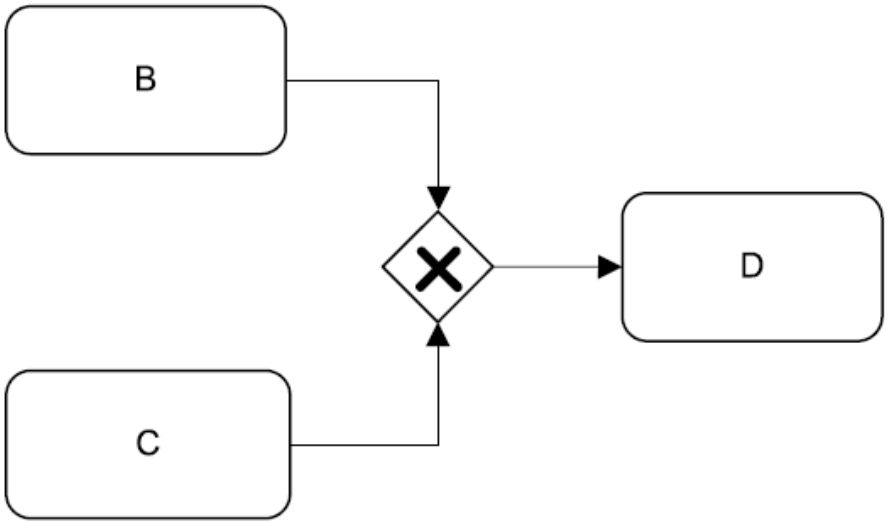


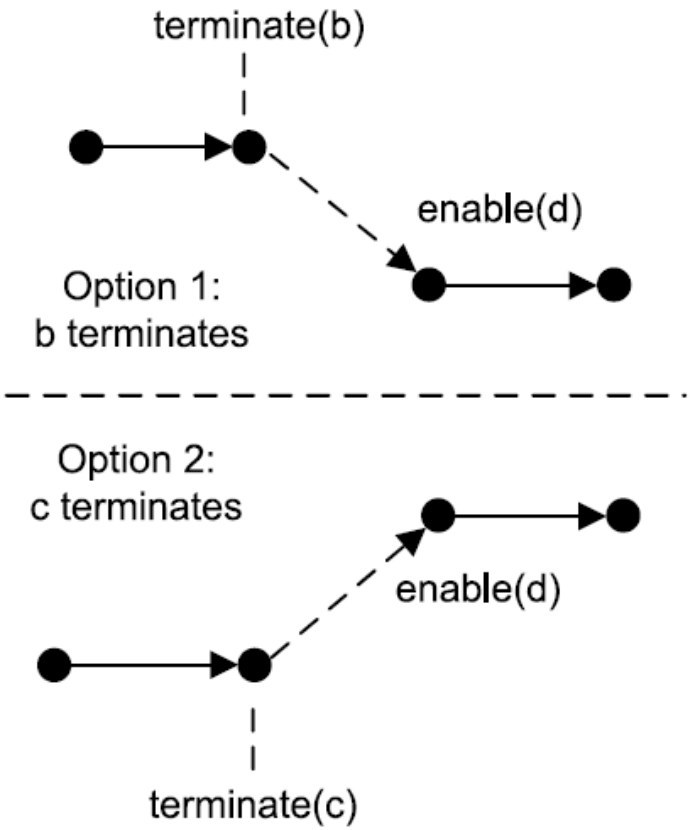
Ekskluzivno ILI spajanje

- Tačka u procesu gde se različite alternativne putanje izvršavanje spajaju bez sinhronizacije.
- P=(*N, E, type*)
  - B,C,D $\in N_A$
  - G $\in N_G$
  - E $\subseteq$ {(B,G),(C,G),(G,D)}
  - *type*(G) = *XorJoin*

- Ekskluzivno ILI spajanje, u ovom slučaju, definiše sledeći redosled događaja: *enable* događaju instance aktivnosti d (e_d), mora prethoditi tačno jedan *terminate* događaj jedne od instanci b (t_b) ili c (t_c).
  $t_b < e_d \vee t_c < e_d$

Ekskluzivno ILI spajanje



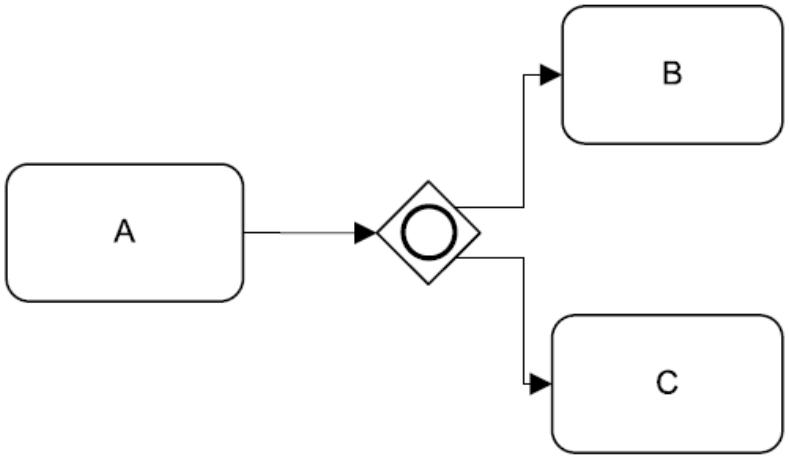


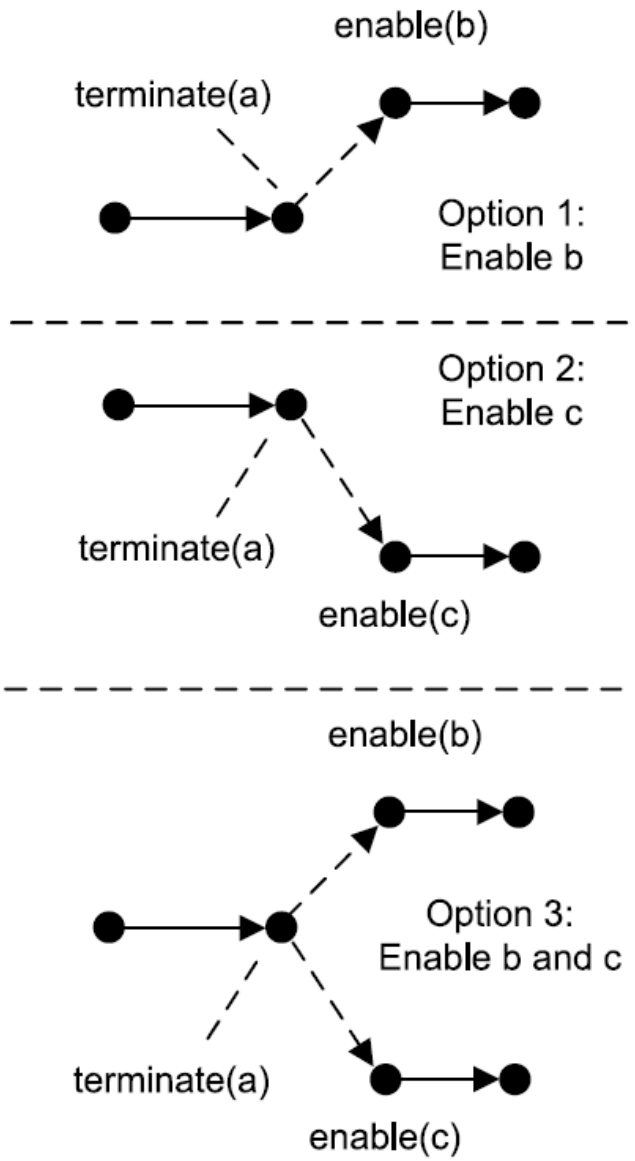
ILI grananje

- Tačka u procesu gde se bira barem jedna od mogućih putanja izvršavanja. Za ovaj tip grananja izbor bilo koje neprazne grane izvršavanja je validan.
- P=(*N, E, type*)
  - $A,B,C \in N_A$
  - $G \in N_G$
  - $E \subseteq \{(A,G),(G,B),(G,C)\}$
  - *type*(G) = *OrSplit*

- XOR grananje, u ovom slučaju, definiše sledeći redosled događaja: posle svakog *terminate* događaja instance aktivnosti a (t_a), sledi podskup *enable* događaja instance aktivnosti b (e_b) i instance aktivnosti c (e_c).
  U proizvoljnom slučaju, bilo koji neprazni podskup ovih događaja je moguć.

ILI grananje





Mogući sled događaja
u slučaju sa dve izlazne grane

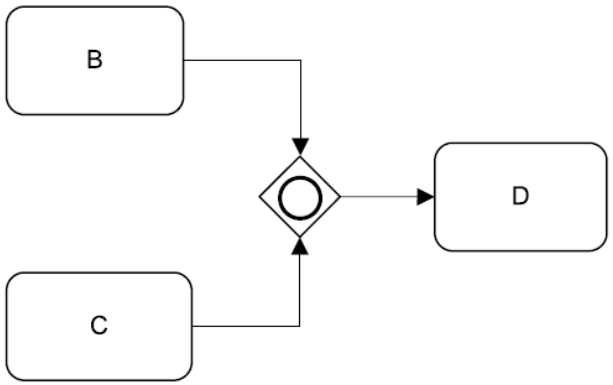
ILI spajanje

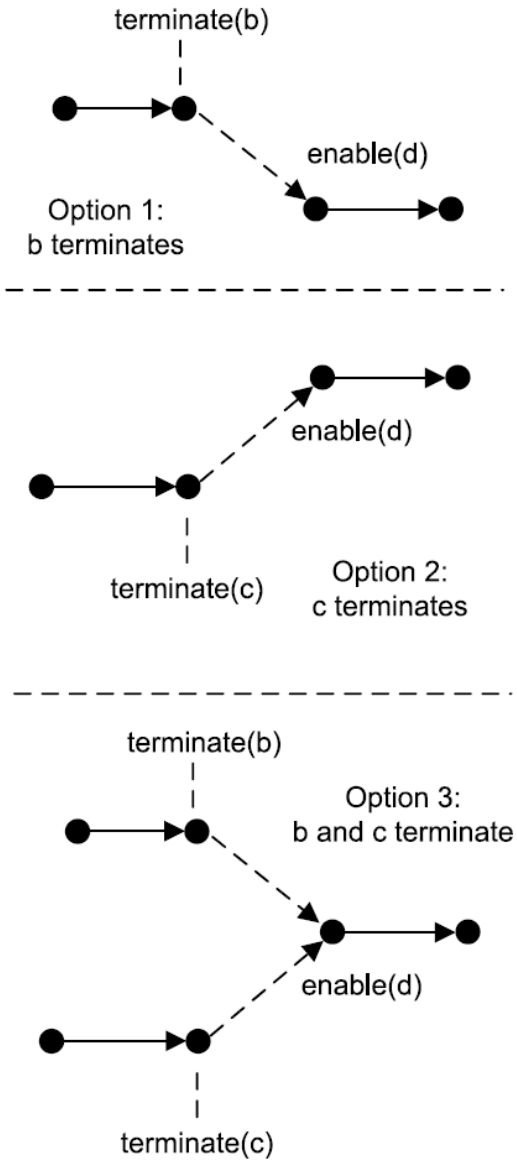
- Tačka u procesu gde se različite putanje izvršavanje spajaju. Pretpostavka je da se jedna grana izvršavanja koje je bila aktivirana ne može ponovo aktivirati dok spajanje čeka na završetak ostalih aktiviranih grana.
- P=(*N, E, type*)
  - $B,C,D \in N_A$
  - $G \in N_G$
  - $E \subseteq \{(B,G),(C,G),(G,D)\}$
  - *type*(G) = *OrJoin*
- Kada se obavljanje svih aktiviranih grananja završi, obavlja se sinhronizacija.

ILI spajanje

- Ovaj šablon je problematičan sa stanovišta da samo spajanje ne može samostalno odlučiti koliko da čeka na aktivaciju
- Npr. ako se desi *terminate* događaj instance aktivnosti b (t_b), koja je reakcija čvora spajanja?
  - Čekanje – čeka se na završetak druge ulazne grane (instanca c)
  - Okidanje – čim se desi t_b
- Bez dodatnog znanja o samoj prirodi procesa nemoguće je reći šta je ovde odgovarajuće ponašanje (čekanje na završetak instance c može trajati unedogled, a ako se izvrši aktiviranje instance d, a c se naknadno ipak završi, šta se dalje dešava?)

ILI spajanje





Višestruko spajanje (multi-merge)

- Predstavlja tačku u procesu u kome se dve ili više konkurentnih niti procesa spajaju bez sinhronizacije
- Aktivnost koja sledi iza ovakvog tipa spajanja se instancira onoliko puta koliko je bilo aktiviranih dolaznih grana
- Funckionalno je ekvivalentan *xor* spajanju, uz razliku da se ne pravi pretpostavka o tome da je aktivirana samo jedna dolazna grana

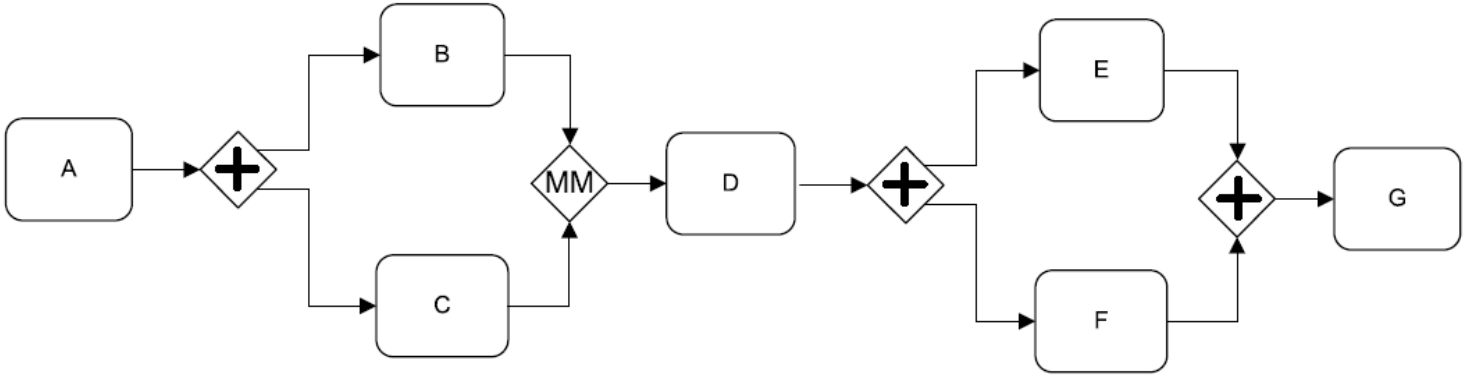
Višestruko spajanje (multi-merge)

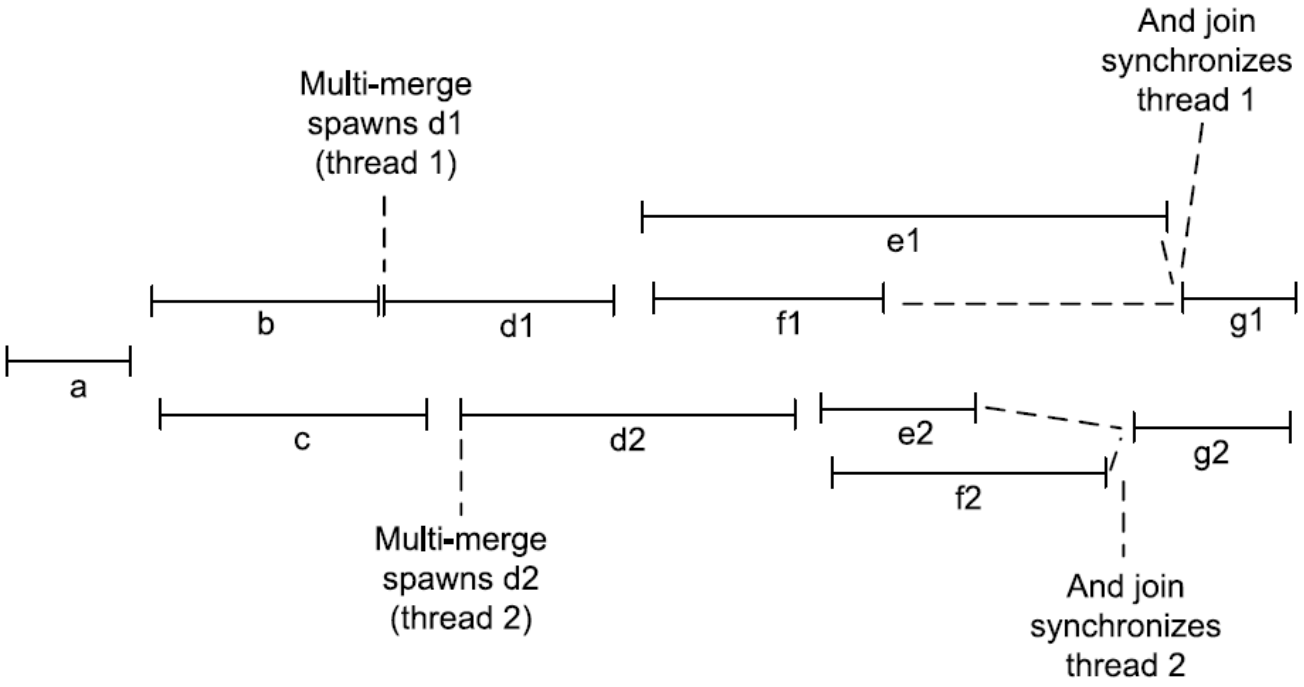
- P=(*N, E, type*)
  - B,C,D $\in N_A$
  - G $\in N_G$
  - E $\subseteq$ {(B,G),(C,G),(G,D)}
  - *type*(G) = *MultiMerge*
- U našem slučaju opis sleda događaja je: za svaki *terminate* događaj t_i, gde i$\in${b,c}, postoji po jedan *enable* događaj e_d, koji se dešava nakon odgovarajućeg t_i

Višestruko spajanje (multi-merge)

- Ovaj šablon proizvodi više niti izvršavanja procesa. Neophodno je voditi računa o ovim nitima pri svim naknadnim sinhronizacijama na *join* čvorovima

Višestruko spajanje (multi-merge)





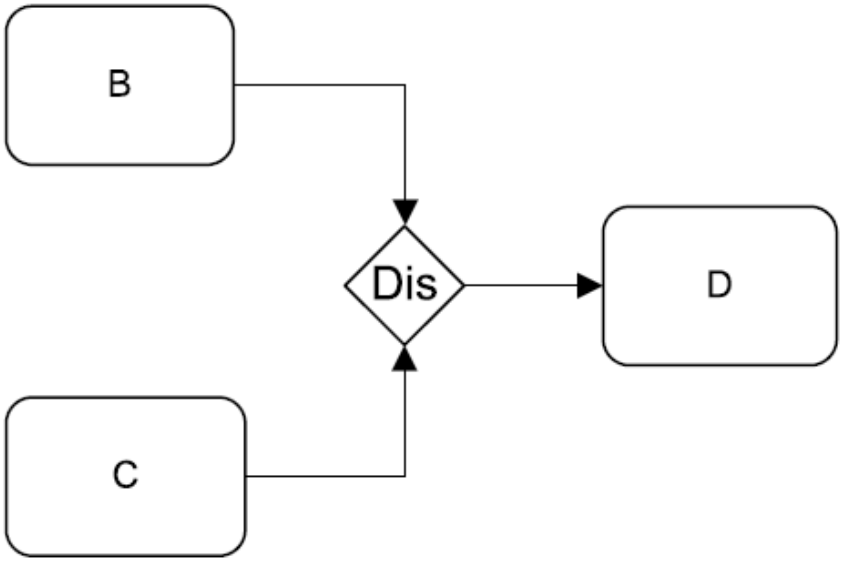
Diskriminator

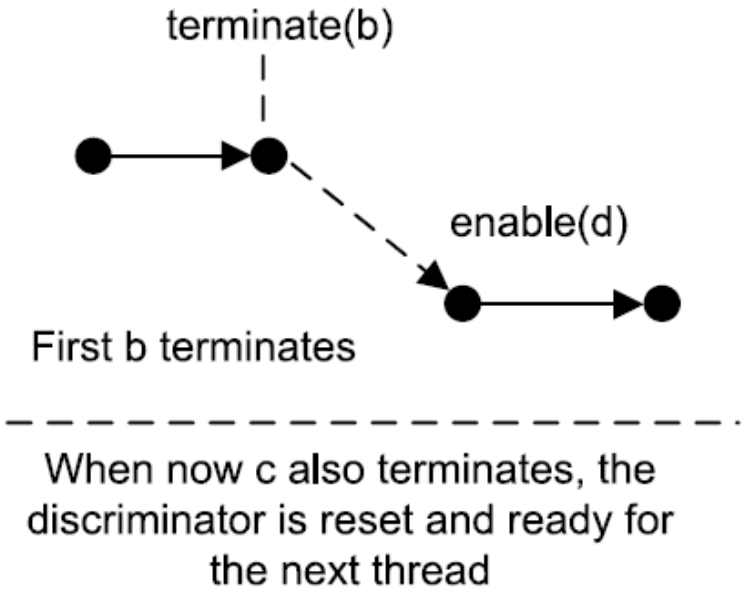
- Predstavlja tačku u procesu u kojoj se čeka da se izvrši jedna od ulaznih grana, pre nego se aktivira naredna aktivnost.
- Nakon što je naredna aktivnost jednom aktivirana, diskriminator čeka da se sve ostale ulazne grane završe, ali ne instancira niti aktivira nove naredne aktivnosti (“ignoriše” informaciju o završetku aktivnosti u ulaznim granama)

Diskriminator

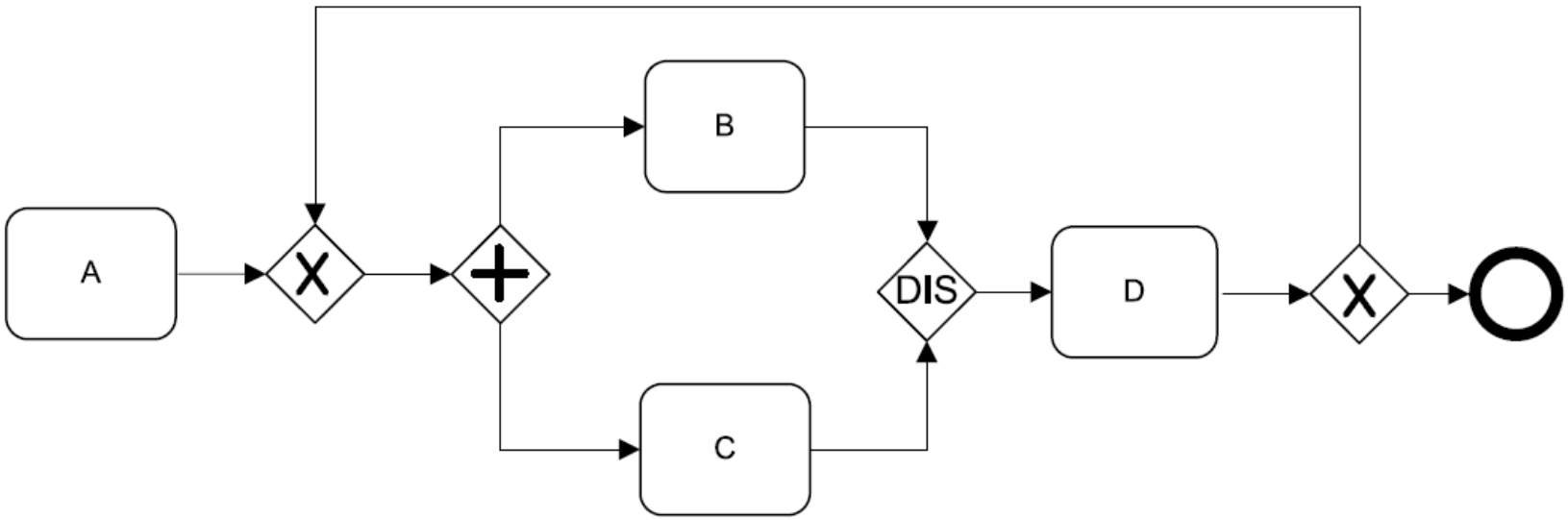
- Nakon što su sve ulazne grane završile svoje aktivnosti, diskriminator se resetuje, tek nakon toga je moguće ponovno “okidanje”
- Ovakva definicija omogućava da se diskriminator koristi u kontekstu petlji, a da pri tome nema mogućnosti konfuzije između aktivnosti obavljenih u drugom ciklusu i okasnele aktivnosti iz prvog ciklusa obrade
- P=(*N, E, type*)
  - $B,C,D \in N_A$
  - $G \in N_G$
  - $E \subseteq \{(B,G),(C,G),(G,D)\}$
  - *type*(G) = *Discriminator*

Diskriminator





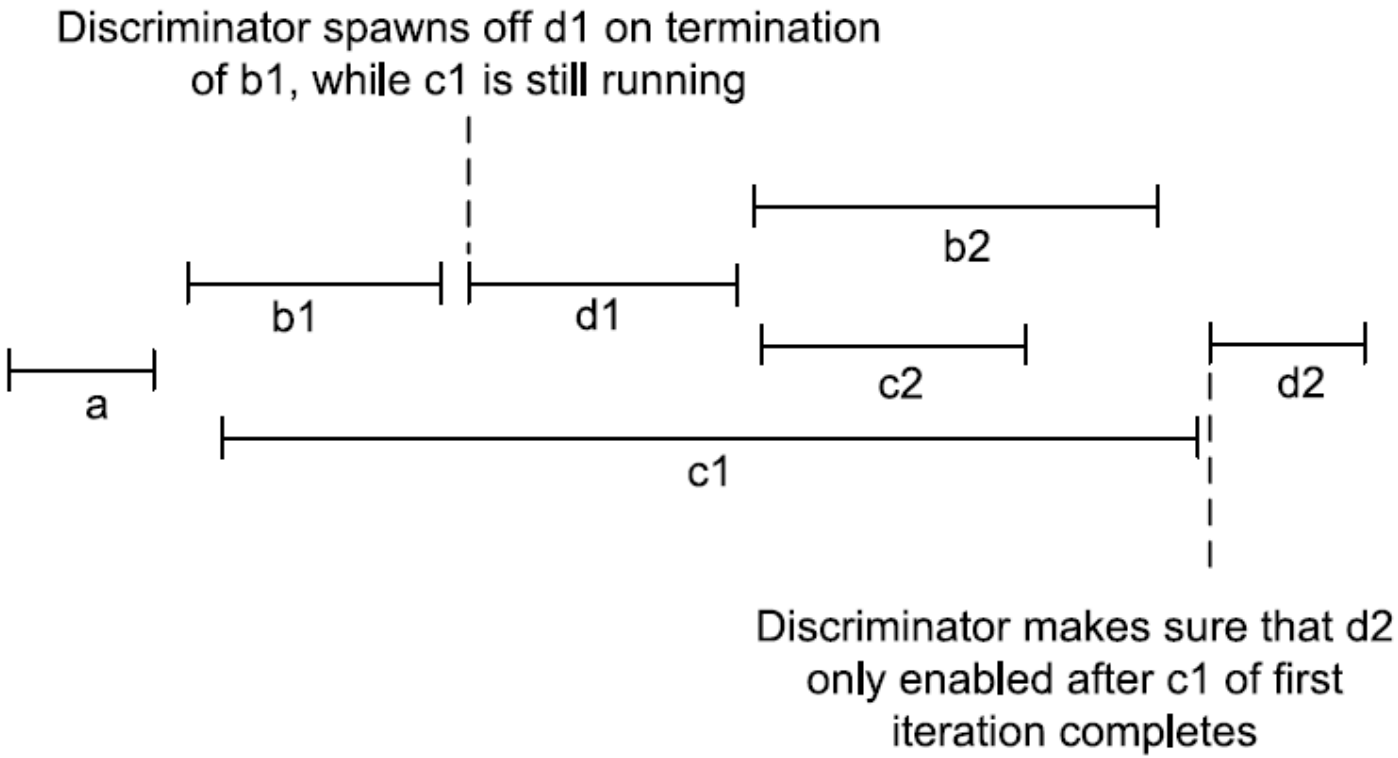
Primer: Diskriminator u petlji



Primer: Diskriminator u petlji

- Primer:
  - Nakon završene instance a, pokreću se b_1 i c_1
  - b_1 se završava prva, diskriminator “okida” i aktivira d_1
  - Kada se d_1 završi, ako je potrebna novi ciklus, kreiraju se b_2 i c_2
  - Moguće je da su istovremeno aktivne c_1 i c_2
  - Čak i ako se c_2 završi pre b_2, diskriminator obezbeđuje da se nova instanca d_2 može pokrenuti samo kada je prvi ciklus u potpunosti završen (tj. nakon reseta)

Primer: Diskriminator u petlji

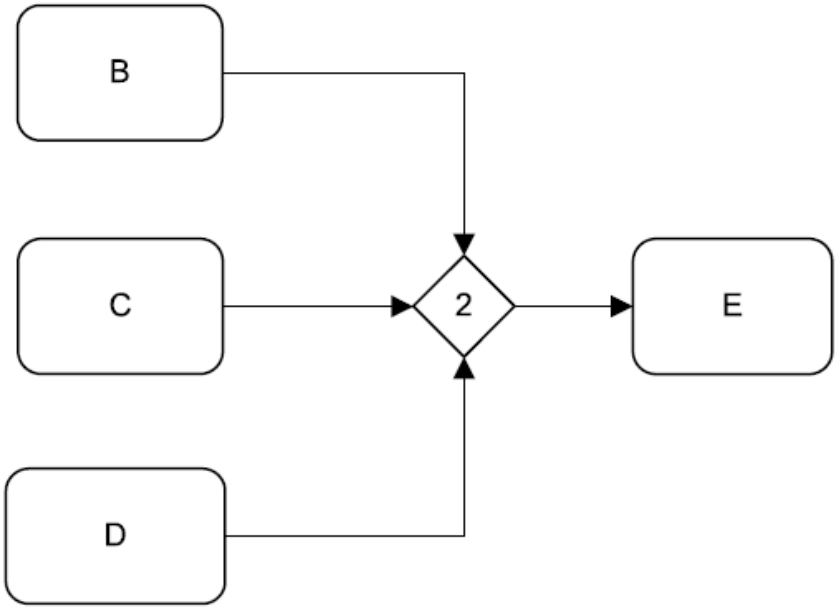


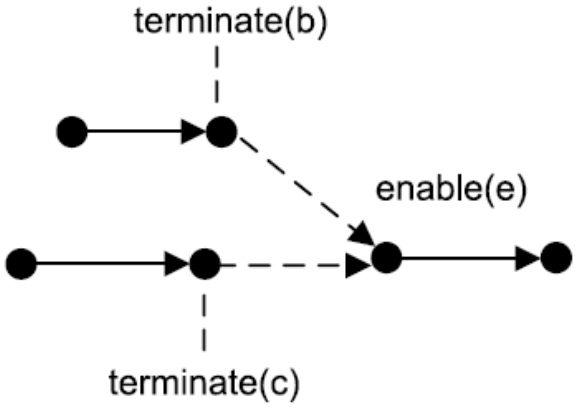
N od M spajanje

- Generalizacija diskriminatora
- Predstavlja tačku u procesu u kojoj se M paralelnih grana spaja u jednu. Čeka da se izvrši N ≤ M ulaznih grana, pre nego se aktivira naredna aktivnost.
- Nakon što je naredna aktivnost jednom aktivirana, terminacija preostalih M-N aktivnosti ulaznih grana se “ignoriše”
- Kao i kod diskriminatora kada se aktivnosti svih ulaznih grana terminiraju, čvor se resetuje

N od M spajanje

- Primer sa tri ulazne grane, za nastavak potrebne dve izvršene





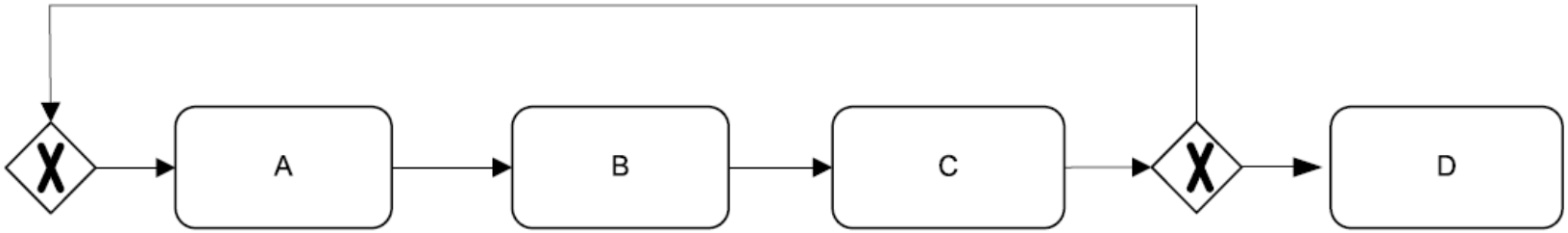
When any 2 activity instances in {b,c,d} have terminates, e can be enabled
(in the example b and c terminated)

N od M spajanje

- Konkretan primer primene bi bilo zahtevanje određenog broja ponuda (koje moraju biti dostavljene u određenom roku) pre odlučivanja o nabavci
- Bez ovakvog šablona ovo je teško za modelovanje
- Za N=M ovo je *I spajanje*
- Za N=1 ovo ipak nije XOR jer se ne zadovoljavaju svi uslovi, a ne realizuje ni višestruko spajanje, jer kod njega bi sve naknadne završene ulazne grane stvarale nove niti procesa, a 1 od N bi ih ignorisao

Proizvoljni ciklusi

- Predstavlja mesto u procesu u kome se jedna ili više aktivnosti može ponavljati

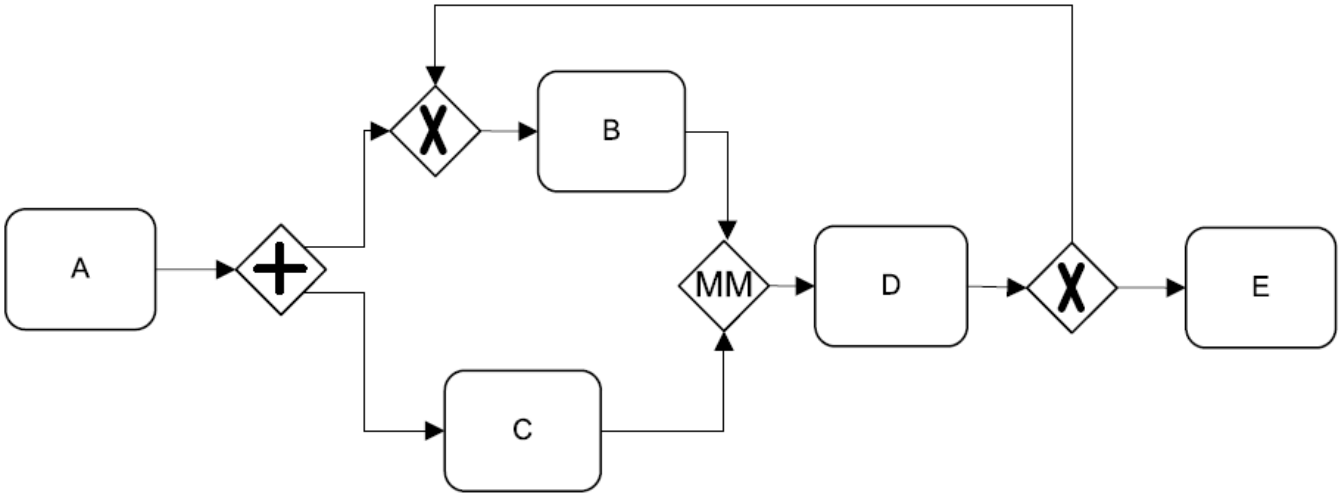


- XOR grananje odlučuje da li se izvršava još jedan ciklus, ili se nastavlja sa izvršavanjem instance d

Proizvoljni ciklusi

- Proizvoljni ciklusi se modeluju korišćenjem drugih šablona za kontrolu toka (prethodni primer koristi XOR grananje i XOR spajanje)
- Moguće su i složenije konstrukcije

Proizvoljni ciklusi



- Višestruko spajanje ovde omogućava da petlja bude deo konkurentne grane procesa.
- Iako je ovo OK sa stanovišta modela, treba biti svestan da može dovesti do toga da se različite niti procesa “pretiču” tokom izvršavanja – nekad je u tom slučaju teško utvrditi tačno ponašanje
- Kraj procesa bi mogao biti dva puta signaliziran, što nije poželjno ponašanje
- I spajanje na mestu MM spajanja bi proces dovelo u *deadlock* nakon prvog ulaska u petlju

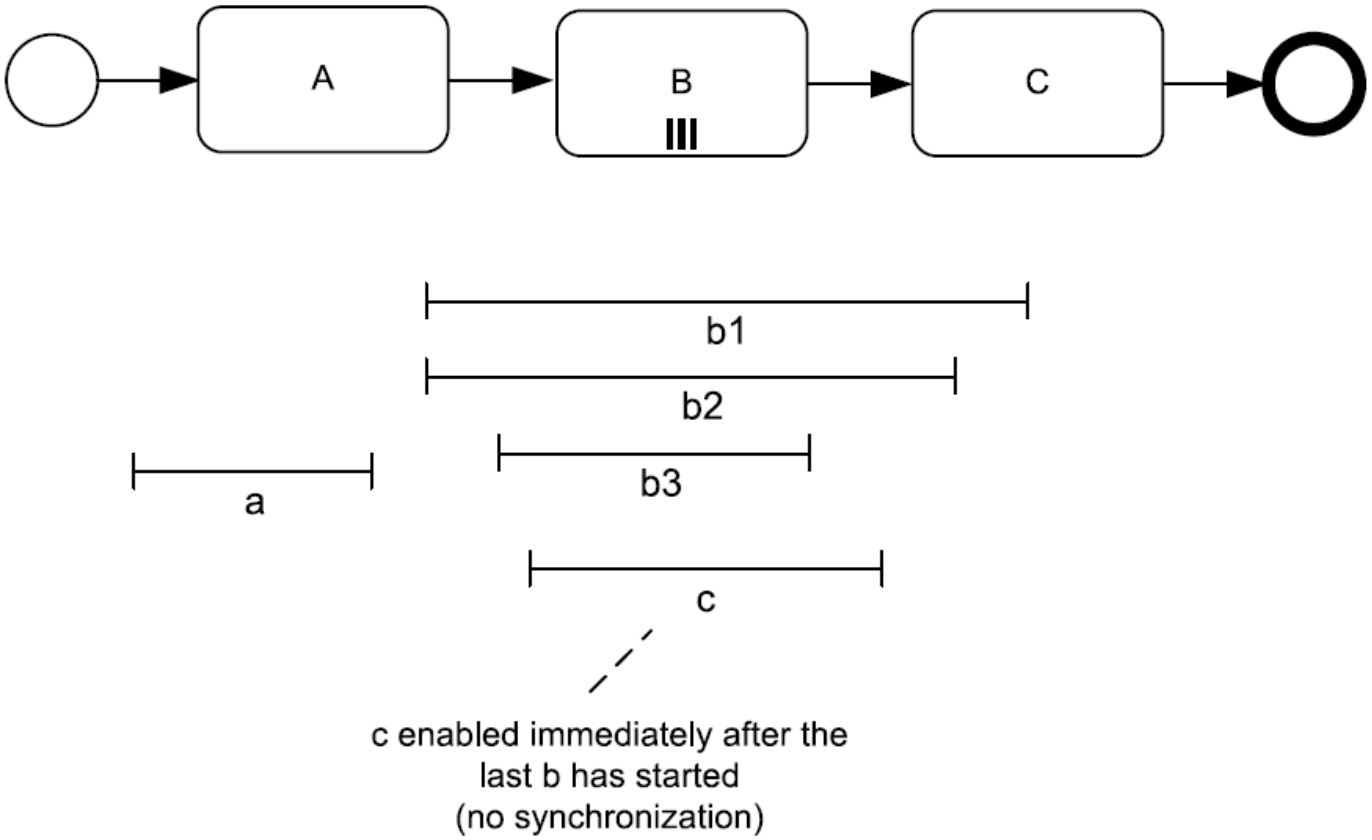
Implicitna terminacija

- Instanca procesa bi trebala biti terminirana kada više nema ništa da se obavi.
  - Ovo znači da više ne postoji nijedna instanca aktivnosti koja je u stanju *init, ready,* niti ima ijedne aktivnosti koja je trenutno u stanju *running*
  - Kao posledica ovoga ne postoji više niti jedna aktivnost koja bi mogla postati omogućena (*enabled*)
- Iako je definisana kao šablon za kontrolu toka, ona ne definiše vezu između aktivnosti, već uslov za završetak procesa.
- U procesnim jezicima, terminacija se često eksplicitno označava jer postoji samo jedno stanje procesa koje označava da je on završen.

Višestruke instance aktivnosti bez sinhronizacije

- Višestruke instance aktivnosti se zasnivaju na istom modelu aktivnosti u kontekstu poslovnog procesa
  - Npr. obrada narudžbi koja sadrži više stavki – za svaku stavku narudžbe obavlja se aktivnost provere
- Kod ovog šablona kreira se više instanci određene aktivnosti, među kojima nije neophodna sinhronizacija

Višestruke instance aktivnosti bez sinhronizacije



Višestruke instance aktivnosti bez sinhronizacije

- Kako nema sinhronizacije – c može biti aktivirana kad su aktivirane sve instance b_i dok se instance b_i nisu završile
- c može i da se završi pre nego se završe sve b_i instance
- Ovo ima svoje posledice – ovakav šablon narušava sekvencijalnost (koja proizilazi iz dijagrama)
- Drugi problem koji ovakav šablon izaziva je terminacija procesa

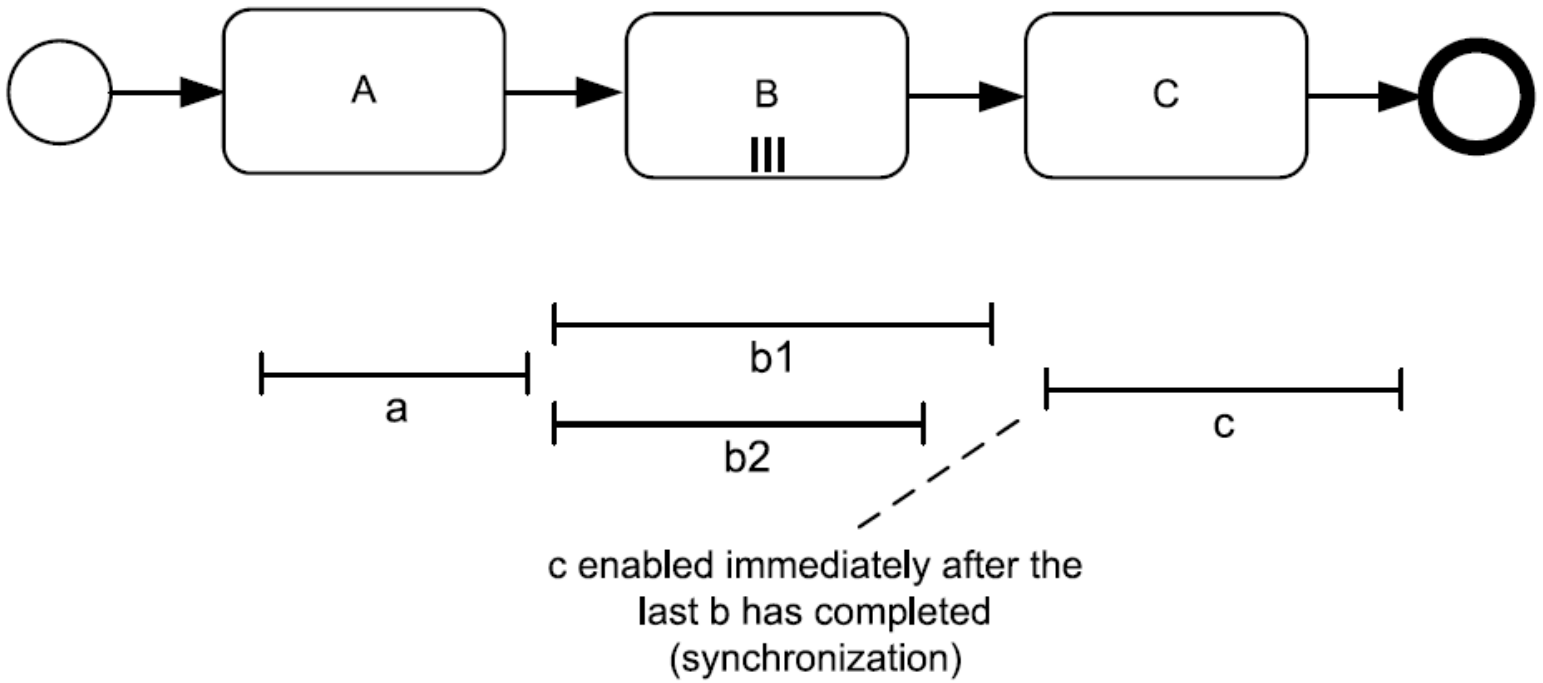
Višestruke instance aktivnosti bez sinhronizacije

- Šabloni sa višestrukim instancama procesa, razlikuju se po momentu u kome se utvrđuje broj multiplikacija instanci
- Ovaj šablon ne pravi pretpostavku kada se to dešava

Višestruke instance aktivnosti sa prethodnim znanjem u momentu dizajna modela

- Broj potrebnih instanci određene aktivnosti poznat je u momentu razvoja modela procesa.
- Instance aktivnosti su sinhronizovane, tako da se sledeća aktivnost omogućava tek kada su sve instance multiplicirane aktivnosti završene
- U zavisnosti od jezika za opis procesa mogu postojati različiti atributi kojima se u model unosi informacija o potrebnom broju instanci

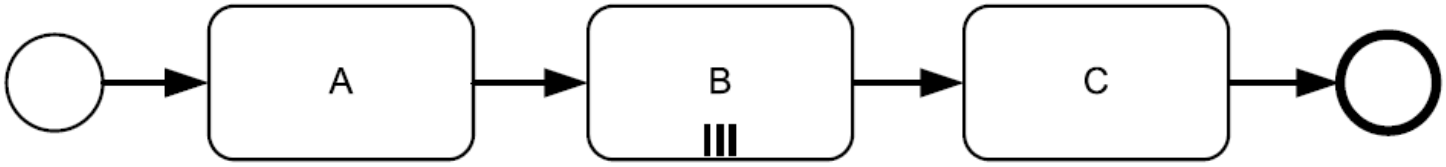
Višestruke instance aktivnosti sa prethodnim znanjem u momentu dizajna modela

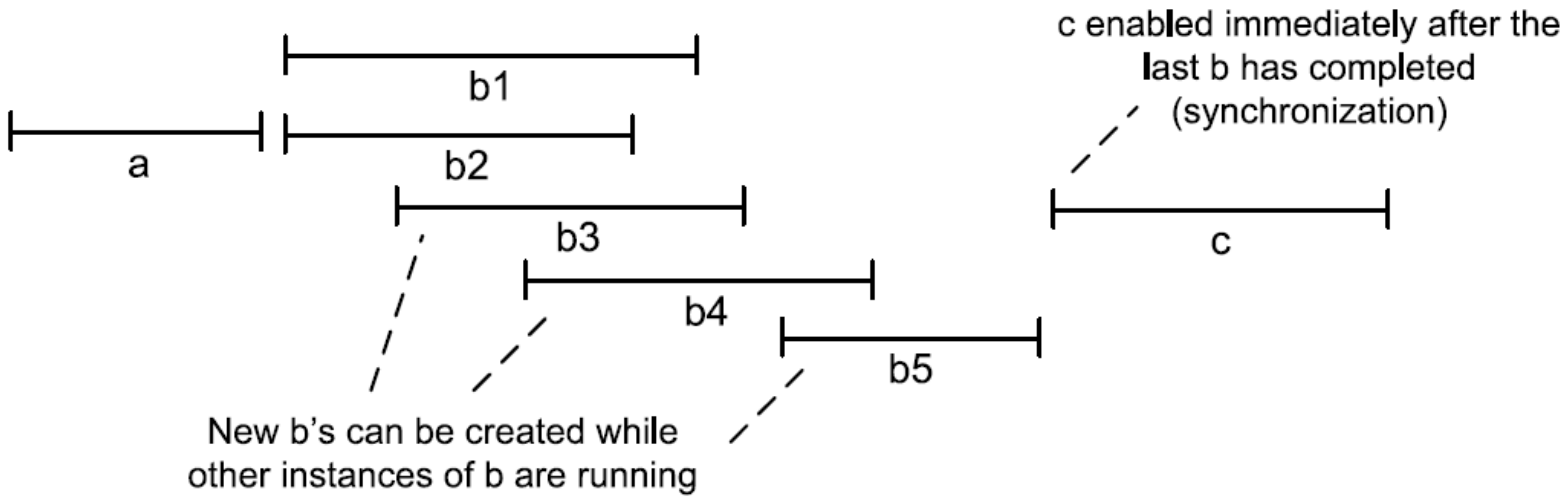


Višestruke instance aktivnosti bez prethodnog znanja u momentu dizajna modela

- Broj potrebnih instanci određene aktivnosti nije poznat u momentu razvoja modela procesa, niti u bilo kojoj fazi izvršavanja procesa pre nego se ova multiplicirana aktivnost omogući
- Nove instance date aktivnosti mogu biti kreirane u momentu dok se postojeće još izvršavaju ili su neke već i završene
- Sledeća aktivnost se omogućava kada se završi i poslednja instanca multiplicirane aktivnosti

Višestruke instance aktivnosti sa prethodnim znanjem u momentu dizajna modela

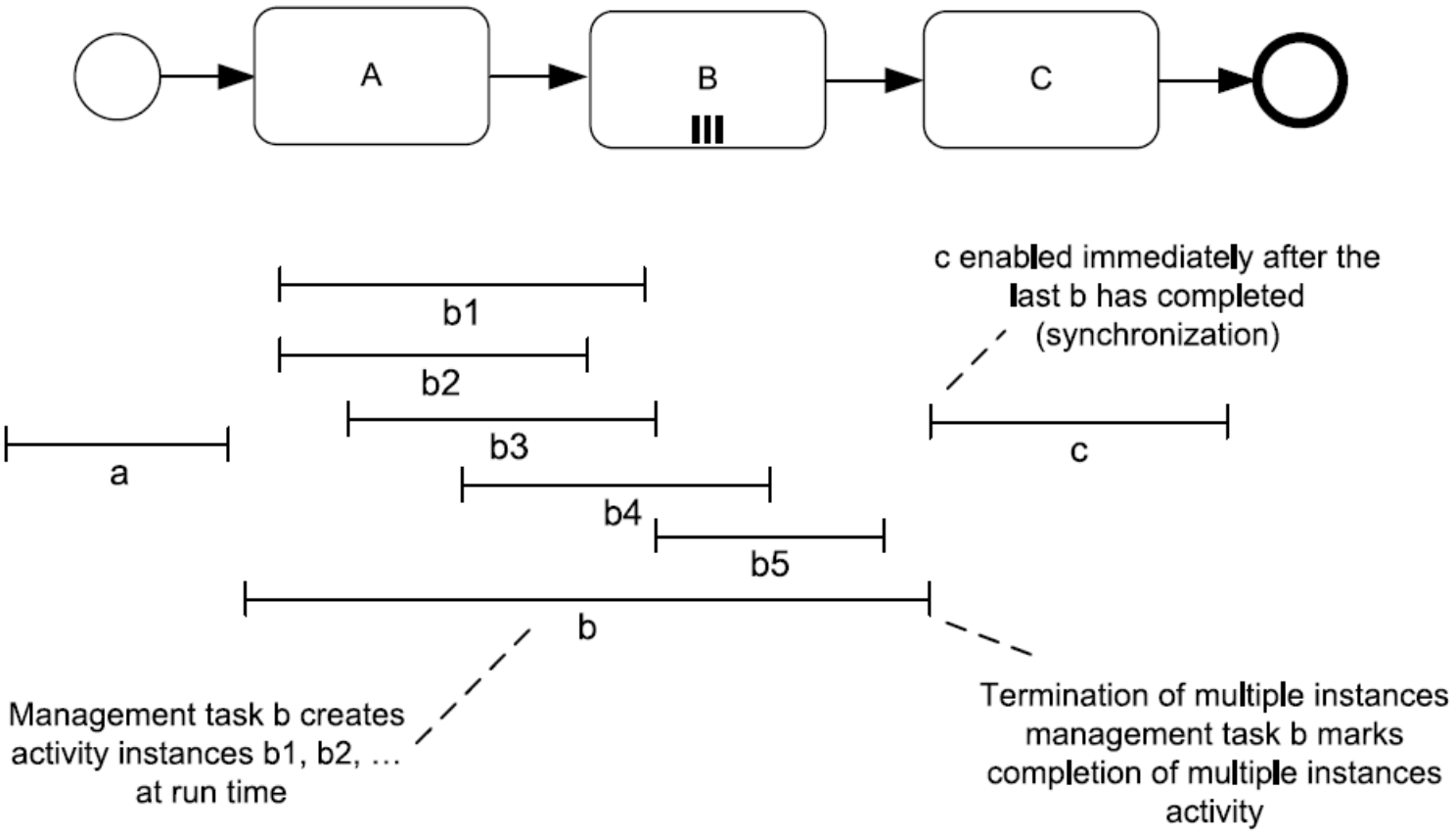




Višestruke instance aktivnosti bez prethodnog znanja u momentu dizajna modela

- Da bi se ovakav šablon implementirao neophodno je da u samom procesnom jeziku i okruženju kojeizvršava upravljanje procesom postoji način da se reši problem do kog momenta je moguće kreirati nove instance aktivnosti
  - Dok postoji bar jedna instanca b koja se trenutno izvršava
  - Postojanje “aktivnosti za upravljanje” multipliciranim aktivnostima

Višestruke instance aktivnosti bez prethodnog znanja u momentu dizajna modela



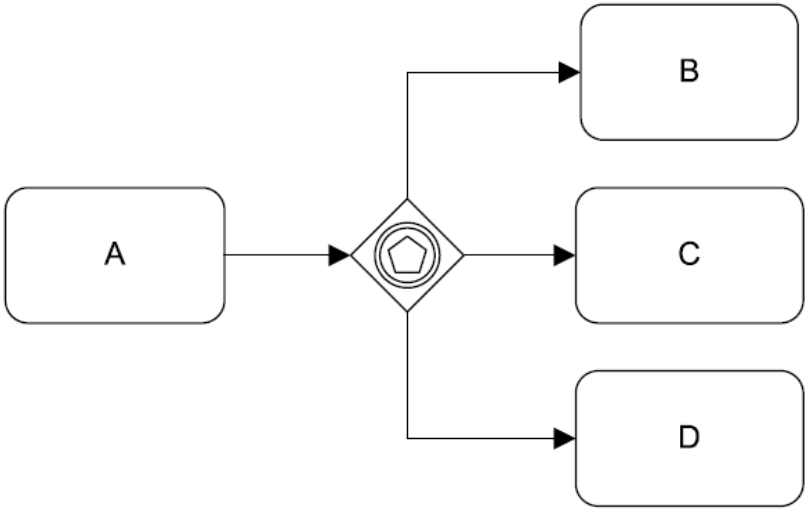
Odloženo odlučivanje

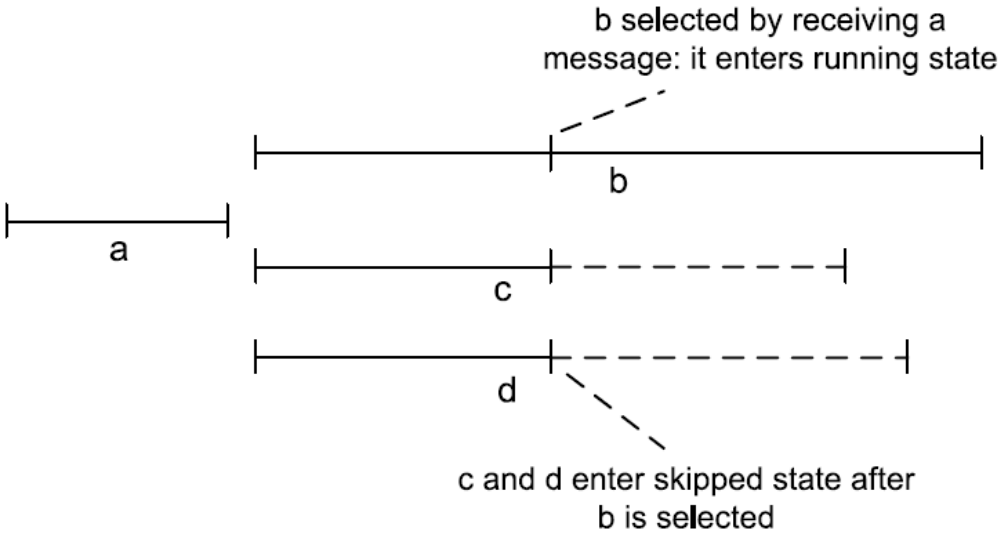
- Šablon baziran na konceptu stanja
- Ovakvi šabloni opisuju implicitno ponašanje procesa koje je bazirano ne na trenutnom slučaju koji se obrađuje, već je uslovljeno okruženjem ili drugim delovima procesa
- Obično postoji neki drugi proces koji predstavlja “okruženje”

Odloženo odlučivanje

- Odloženo odlučivanje predstavlja mesto u procesu gde se bira jedna od mogućih grana izvršavanja. Izbor nije eksplicitan (npr. na osnovu vrednosti podataka, ili na osnovu odluke korisnika) nego se okruženju ponudi nekoliko alternativa
- Okruženje aktivira jednu od alternativnih putanja a ostale grane se ne aktiviraju
- Kako se odlučivanje odlaže sve do momenta dok jedna grana nije aktivirana, ono je odloženo do poslednjeg mogućeg momenta

Odloženo odlučivanje

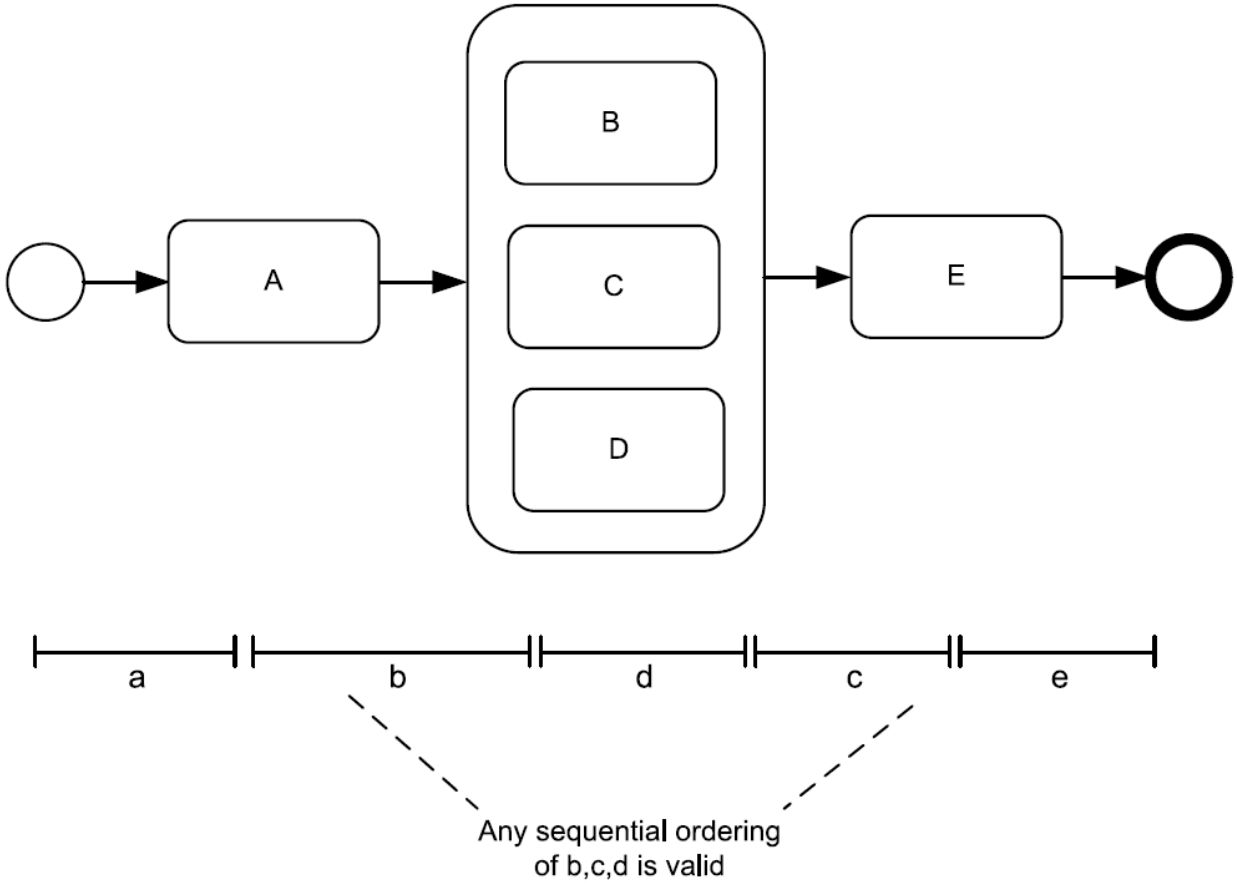




Sekvencijalno izvršavanje bez prethodnog zananja u vreme razvoja modela

- Skup aktivnosti se izvršava sekvencijalno po redosledu koji se utvrđuje u vreme izvršavanja procesa
  - Inicijalno ovaj šablon je bio poznat kao preklopljeno paralelno rutiranje (*interleaved parallel routing)*
- Koristan je u slučajevima kada određene aktivnosti treba izvesti sekvencijalno u bilo kom redosledu
- Za n aktivnosti modelovanje n! Mogućih kombinacija nije praktično

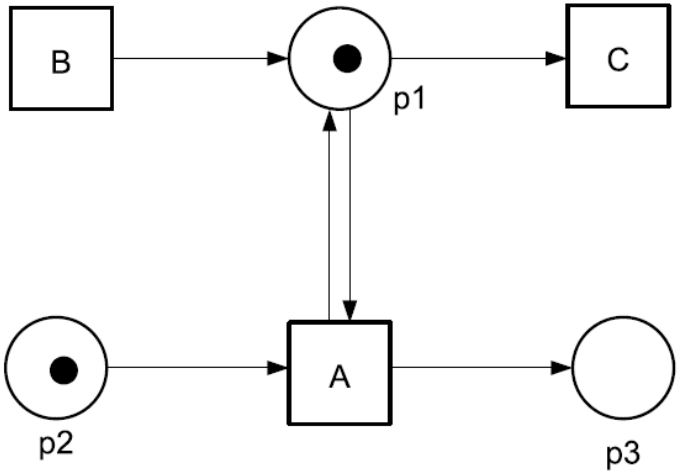
Sekvencijalno izvršavanje bez prethodnog zananja u vreme razvoja modela



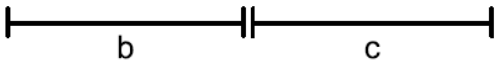
Milestone – referentna tačka

- Ovaj šablon se koristi da se određena aktivnost omogući samo kada je dostignuta određena referentna tačka (koja je pri tome još važeća)
- Primer: proces koji se sastoji od aktivnosti A, B, C. Pri definisanju procesa moguće je zahtevati da se instanca aktivnost a omogući samo ako je instanca aktivnosti b izvršena, a c još nije dovršena
- Model je dat u formatu Petri mreže jer prethodna neformalna notacija ne sadrži eksplicitnu notaciju za opis ovakvog slučaja

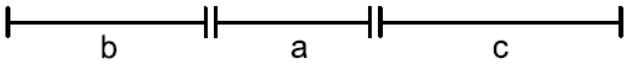
Milestone – referentna tačka



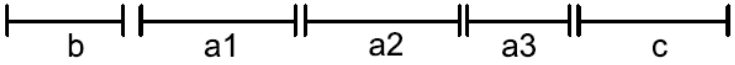
Option 1: *c* is immediately started after *b* completes, so that *a* cannot be executed



Option 2: *a* is started after *b* completes and before *c* starts



More Options: Multiple *a*'s possible, as long as *c* is not started and there are tokens in *p2*



Runtime šabloni

- Nisu deo modela procesa – karakterišu funkcionalnost okruženja za izvršavanje poslovnog procesa (*BPM system-a*)
  - Otkazivanje aktivnosti (*cancel activity pattern*)
  - Otkazivanje slučaja (cancel case pattern) – instanca procesa se zaustavlja