Poglavje 1

Kvalitativna analiza zanesljivosti

V predhodnih poglavjih smo zanesljivost opazovanega sistema ocenjevali *kvantitativno*, kar pomeni, da je bil cilj zanesljivostne analize usmerjen v numerično evaluacijo funkcije $R_{sys}(t)$, še pred njo pa v pridobivanje funkcije intenzivnosti odpovedovanja $\lambda(t)$. V pričujočem poglavju si bomo ogledali nekaj najpogostejših tehnik, ki omogočajo izvedbo zanesljivostne analize s pomočjo *kvalitativnih pristopov* [1]. Le te so sledeče:

- *modeliranje zanesljivosti s pomočjo Petrijevih mrež* (angl. *Petri nets* - PN),
- *funkcijska analiza* (angl. *functional analysis* - FA)
- *zanesljivostni bločni diagrami* (angl. *reliability block diagrams* - RBD),
- *analiza načinov odpovedi in njihovih efektov* (angl. *failure modes and effects analysis* - FMEA),
- *analiza dreves odpovedi* (angl. *fault tree analysis* - FTA).

Vse naštete metode nas navajajo k kritični analizi zanesljivosti sistemov, končni rezultati analize pa niso numerične ali kvantitativne narave.

1.1 Modeliranje zanesljivosti s Petrijevimi mrežami

Petrijeve mreže uporabljamo kot univerzalno orodje za modeliranje dinamičnih sistemov. Njihov obširnejši opis najdemo v delu [2], v nadaljevanju pa navedemo le nujno potrebne osnove za razumevanje kasneje predstavljenih zanesljivostnih kvalitativnih analiz na osnovi Petrijevih mrež.

1.1.1 Osnove Petrijevih mrež

Dinamiko v kakršnemkoli sistemu lahko interpretiramo kot *zaporedje izvedenih akcij*, za izvedbo katerih morajo biti izpolnjeni neki predhodno določeni *izpolnjeni pogoji*. Proženje posamezne akcije je torej pogojeno z vnaprej določeno množico izpolnjenih pogojev, rezultat proženja akcije pa povzroči izpolnjenost novih pogojev. Slednje relacije so ponazorjene z usmerjenimi povezavami, ki vodijo iz pogojev v akcije (v primeru proženja izvajanja akcij) in s povezavami, ki vodijo iz akcij v pogoje (v primeru ustvarjanja izpolnjenosti novih pogojev). Posamezen pogoj v Petrijevi mreži je lahko izpolnjen ali pa ne. V primeru izpolnjenosti pogoja govorimo o kratnosti izpolnjenosti pogoja. Tako je lahko pogoj izpolnjen enkrat, dvakrat ali večkrat, lahko pa ni izpolnjen.

Na tem mestu zapišimo formalno definicijo Petrijeve mreže povzeto po viru [3].

Definicija 1 *Petrijeva mreža je definirana kot četvorček C=(P,T,I,O), pri čemer P predstavlja končno množico pogojev, T končno množico akcij, I vhodno in O izhodno funkcijo. Množici P in T sta si tuji* ($P \cap T = \emptyset$).

Vhodna in izhodna funkcija definirata relacije ali povezave med akcijami in pogoji. Vhodna funkcija I tako za akcijo t_j določa množico pogojev $I(t_j)$, iz katerih vodijo povezave proti akciji t_j, izhodna funkcija O pa za akcijo t_j določa množico pogojev $O(t_j)$, v katere vodijo povezave iz akcije t_j. Tako so možne povezave le iz akcij v pogoje in obratno, niso pa dovoljene povezave iz pogoja v pogoj ali iz akcije v akcijo. Formalno obe funkciji običajno zapišemo v obliki prehajalnih matrik reda $m \times n$, pri čemer m predstavlja število akcij ($m = |T|$) in n število pogojev ($n = |P|$).

V izrazih (1.1) in (1.2) je predstavljen primer formalnega zapisa Petrijeve mreže s štirimi pogoji, tremi akcijami in desetimi povezavami v matrični notaciji.

$$C = (P, T, I, O), P = \{p_1, p_2, p_3, p_4\}, T = \{t_1, t_2, t_3\}, \tag{1.1}$$

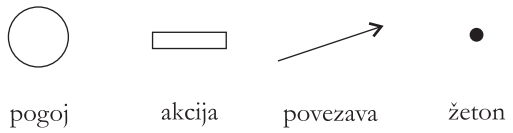
$$I = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \; O = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.2}$$

Grafično ponazoritev formalno definirane Petrijeve mreže imenujemo za *graf Petrijeve mreže*. Njegova definicija je sledeča [3]:

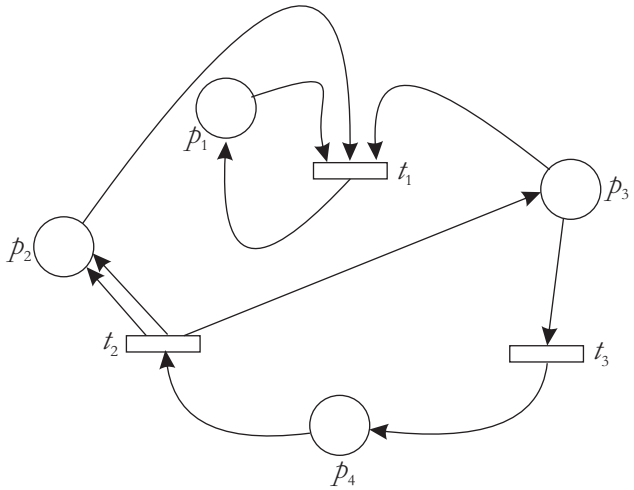
Definicija 2 *Graf Petrijeve mreže je bipartitni usmerjeni graf G=(V,A), kjer V* ($V = P \cup T$) *predstavlja množico vozlišč in A množico povezav med njimi* ($a_i = (v_j, v_k)$). *Velja naslednja relacija:*

$$\forall i : a_i = (v_j, v_k), \; (v_j \in T) \& (v_k \in P) \vee (v_j \in P) \& (v_k \in T). \tag{1.3}$$

Za tvorbo grafov Petrijevih mrež uporabljamo grafične primitive, ki so prikazani na sliki 1.1, graf Petrijeve mreže, ki smo jo formalno zapisali v izrazih (1.1) in (1.2), pa na sliki 1.2.



Slika 1.1: Grafični primitivi za tvorbo grafa Petrijeve mreže.



Slika 1.2: Primer grafa Petrijeve mreže.

Z *označitvami pogojev* v Petrijevih mrežah določamo kratnost izpolnjenosti posameznih pogojev, na samih grafih pa jo označujemo z ustreznim številom *žetonov*, ki jih vrišemo v posamezni pogoj. Formalno *označitev* Petrijeve mreže zapišemo z *vektorjem označitve*

$$o(k) = (o_1(k), o_2(k), ..., o_n(k)), \forall i : (o_i(k) \in \mathbb{N} \cup \{0\}), i = 1, .., n, \qquad (1.4)$$

pri čemer n predstavlja število pogojev ($n = |P|$), $o_i(k)$ pa število žetonov v i-tem pogoju ali kratnost izpolnjenosti i-tega pogoja. Skozi čas se v sistemu sprožajo različne akcije in načeloma se s tem spreminjajo tudi izpolnjenosti posameznih pogojev. Iz tega razloga je vektorju označitve dodan diskretni časovni atribut k.

Z *označitvijo* vseh pogojev pridemo do *označene Petrijeve mreže*, ki jo zapišemo kot petorček $M = (P, T, I, O, o(k))$. Glede na to, da definicija označitve

ne predvideva omejitve števila žetonov v posameznem pogoju, za vsako Petrijevo mrežo obstaja neskončno možno označitev, če ima mreža vsaj en pogoj. Označitev mreže $o(k)$ lahko enačimo s *stanjem*, v katerem se nahaja mreža na k-tem diskretnem koraku simulacije njene dinamike.

Proženje posameznih akcij je pogojeno s trenutno označitvijo v Petrijevi mreži. Posamezna akcija se lahko sproži, če je *omogočena*. Akcija t_i je omogočena, če se po vsaki povezavi, ki vstopa v to akcijo, lahko pripelje žeton iz pogoja, ki predstavlja izvor povezave (vhodni pogoj). To lahko formalno zapišemo z izrazom

$$o(k) \geq e(t_i) * I, \ i = 1, ..., m, \tag{1.5}$$

pri čemer je $e(t_i)$ enotski vektor dolžine m $(m = |T|)$, enica pa se nahaja na i-ti poziciji, ki jo definira indeks akcije. V primeru izpolnjenosti izraza (1.5) se akcija t_i na k-tem koraku lahko sproži.

Sprožitev akcije v splošnem odvzema žetone iz vhodnih pogojev - pogojev iz katerih vodijo povezave proti opazovani akciji t_i. Po hipnem trajanju akcije (predpostavljamo, da je čas trajanja akcije ničen ali zanemarljivo majhen) pride do posledičnega izražanja na izhodnih pogojih - pogojih do katerih vodijo izhodne povezave iz opazovane akcije t_i. Praviloma se po hipnem proženju akcije t_i odpravi po vsaki izhodni povezavi proti izhodnim pogojem en žeton. Vsak od njih se uskladišči v izhodnem pogoju. Prehajanje žetonov od opazovane - prožene akcije t_i proti izhodnim pogojem posledično privede do nove označitve sistema $o(k+1)$, ki jo zapišemo z izrazom

$$o(k+1) = o(k) + e\,[t_i]\,(O - I). \tag{1.6}$$

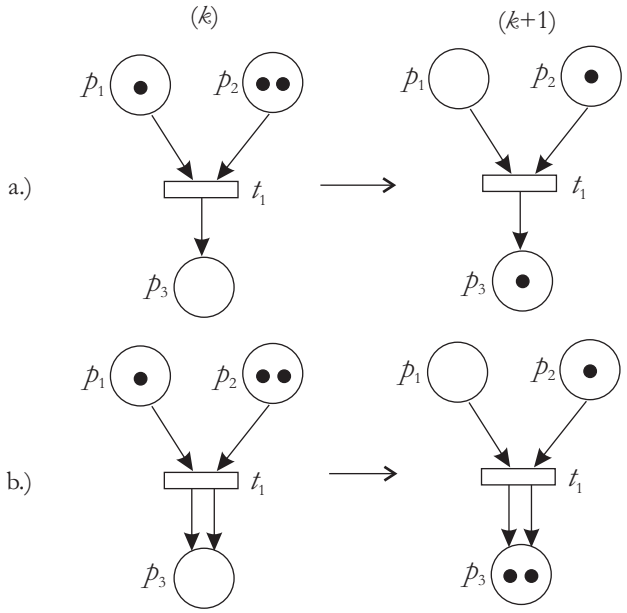
Še enkrat poudarimo, da je potovanje posameznega žetona od vhodnega pogoja preko prožene akcije do izhodnega pogoja hipno (brez časovnega trajanja). Na sliki 1.3 sta predstavljena dva zgleda prehajanja žetonov od vhodnih do izhodnih pogojev. Iz slike je razvidno, da ni nujno, da je število žetonov, ki vstopajo v akcijo, enako številu žetonov, ki iz nje izstopajo.

V primeru, da imamo na k-tem koraku v Petrijevi mreži več omogočenih akcij, se na tem koraku izvede le ena od njih, izbrana pa je nedeterministično. Slednje pomeni, da istočasno proženje več akcij v Petrijevih mrežah ni možno.

Ob analizi proženja akcij moramo biti pozorni na akcije, ki nimajo vhodnih pogojev. Tovrstne akcije so vedno omogočene, česar v realnih sistemih ne zasledimo in se posledično temu tovrstnim konstruktom v Petrijevih mrežah izogibamo.

1.1.2 Časovne Petrijeve mreže

V naslednjih razdelkih bomo osnovne pojme s področja zanesljivosti predstavili s Petrijevimi mrežami. Pojmi, ki jih bomo ponazorili z grafi Petrijevih mrež, so *(ne)popravljivost*, *življenska doba sestavnega dela sistema*, *čas servisiranja*, *redundanca*, *vrste stanj sistema*, *"hot standby"*, *"cold standby"*, *stikalo*, *diagnosticiranje*, *servisiranje* itd. Za potrebe omenjenih ponazoritev osnovni definiciji



Slika 1.3: Zgleda dinamike žetonov v Petrijevih mrežah.

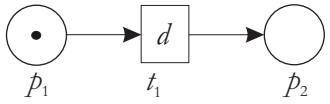
Petrijevih mrež iz predhodnega razdelka dodamo časovnost trajanja posameznih akcij [4].

Definicija 3 *Šestorček* $PN = (P, T, I, O, o(k_0), D)$ *imenujemo za razširjeno časovno Petrijevo mrežo, kjer predstavlja* P *končno množico pogojev,* T *končno množico akcij,* I *vhodno matriko in* O *izhodno matriko, pri čemer obe matriki glasita na akcije.* $o(k_0)$ *predstavlja začetno porazdelitev žetonov po indeksirani množici pogojev,* D *pa vektor pozitivnih števil vključujoč ničlo, ki posameznim akcijam določajo časovno trajanje.*

Z razliko od Petrijevih mrež definiranih v predhodnem razdelku smo tako akcijam pripisali časovno trajanje. V splošnem si trajanje interpretiramo na način ponazorjen s citatom v nadaljevanju povzetem po [4]. "*Akcija s trajanjem* d *časovnih enot se začne izvajati, ko so izpolnjeni vsi pogoji, iz katerih vodijo vhodne povezave v opazovano akcijo. Pogoji so (in morajo biti) izpolnjeni vse do konca trajanja akcije. Šele po* d *časovnih enotah se njihova izpolnjenost razveljavi*". Povedano drugače, žetoni v vhodnih pogojih ostanejo na svojih mestih d časovnih enot, šele nato pa se hipno preselijo preko akcije v izhodne pogoje, kot je to opisano v izrazu (1.6).

Na sliki 1.4 je prikazan enostaven primer grafa časovne Petrijeve mreže. Če

žeton v času k_1 prispe v pogoj p_1, se bo iz tega pogoja preko akcije t_1 hipno preselil v časovni točki $k_1 + d$ v pogoj p_2. Seveda bo to tega prišlo le v primeru, če v intervalu $]k_1 + d[$ omenjenega žetona ne bo uporabila (prevzela) kaka druga akcija, ki ji omenjeni pogoj tudi predstavlja vhod. Če bi takšna konkurenčnost obstajala, uspe v izvedbi tista akcija, ki ima krajši čas trajanja. Če bi bilo torej proženje več akcij odvisno le od enega pogoja (žetona v njem), bi se izvedla tako le ena akcija in sicer tista, z najkrajšim časom trajanja. V primeru, da bi imelo več akcij enak čas trajanja, bi se izmed njih izvedla naključno ali nedeterministično izbrana akcija.



Slika 1.4: Enostaven primer časovnosti trajanja akcije t_1 v Petrijevi mreži.

V nadaljevanju si bomo ogledali nekaj zgledov modelov kvalitativne analize zanesljivosti sistemov podanih v obliki grafov Petrijevih mrež. Večina zgledov je povzeta po viru [4].

1.1.3 Popravljive in nepopravljive entitete sistema

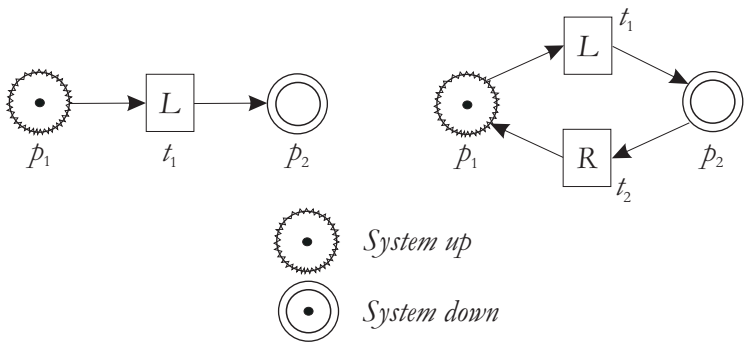
Podsisteme, funkcionalne enote, komponente, programsko opremo, skratka vse entitetne člene v opazovanem računalniškem sistemu v splošnem delimo na *popravljive* (angl. *repairable*) in na *nepopravljive* (angl. *nonrepairable*). Dogovorimo se, da bomo s črko L_i označevali predvideno *življensko dobo* i-te entitete, s črko R_i pa predviden *čas servisiranja* i-te entitete, če je le ta popravljiva.

Grafa Petrijevih mrež nepopravljive in popravljive entitete sta predstavljena na sliki 1.5. Pri tem je na sliki normalno delujoče stanje entitete predstavljeno s prisotnostjo žetona v nazobčanem pogoju (stanje *system up*), nedelujoče stanje entitete pa s prisotnostjo žetona v obkroženem pogoju (stanje *system down*).

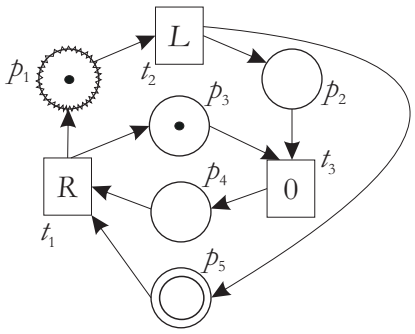
V primeru, da je za fazo servisiranja popravljive entitete potreben resurs (npr. seviser, rezervni del itd.), dobimo graf Petrijeve mreže predstavljen na sliki 1.6. Razpoložljivost resursa je na sliki ponazorjena z žetonom v pogoju z oznako p_3. Vpeljava resursa zahteva dodatno akcijo t_3 s časovnim trajanjem 0, ki resurs zaseže in prestavi na novo lokacijo (v pogoj p_4), da ga ne bi mogla med samim servisiranjem zaseči kaka druga akcija. Izpolnjenost pogoja p_4 si lahko intepretiramo kot že začeto aktivno sodelovanje resursa pri servisiranju.

1.1.4 Redundantne entitete sistema

V sisteme, v katerih hočemo doseči večjo stopnjo zanesljivosti, vključujemo večje število rezervnih ali redundantnih entitet, pri čemer ni nujno, da za normalno obratovanje sistema potrebujemo ravno vse. Običajno je dovolj, da je zmožna sistemsko breme izvajati vsaj ena od njih.



Slika 1.5: Petrijev graf nepopravljive (levo) in popravljive entitete (desno).



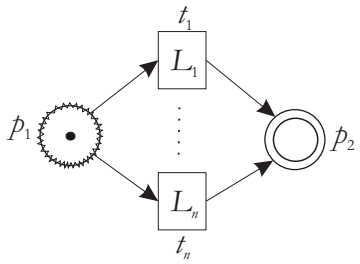
Slika 1.6: Petrijev graf popravljive entitete, ki za servisiranje zahteva prisotnost resursa s.

Na sliki 1.7 je prikazan model *neredundantnega sistema* sestavljenega iz n nepopravljivih entitet. Ko prva od njih odpove, se žeton prenese iz pogoja p_1, ki označuje delujoče stanje sistema, v pogoj p_2, ki označuje nedelujoče stanje sistema.

Življensko dobo L tovrstnega sistema ponazorimo z izrazom

$$L = \min(L_1, ..., L_n). \tag{1.7}$$

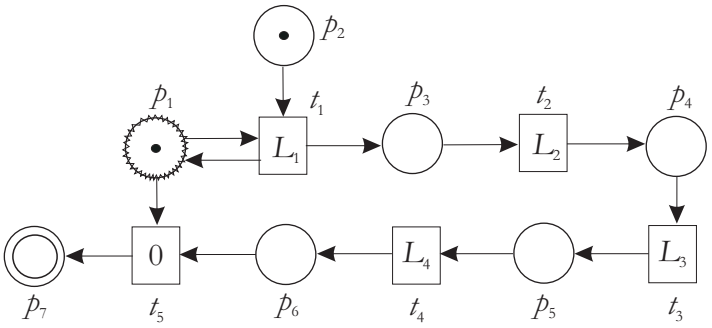
V redundantnih sistemih moramo zagotoviti tudi mehanizem detekcije odpovedi posamezne entitete in preusmeritve sistemskega bremena na neko drugo delujočo entiteto. Omenjeni funkciji vrši nova entiteta, ki jo poimenujemo za *stikalo*. Slednje je lahko zopet bodisi idealno zanesljivo, ali pa ima določeno stopnjo možnosti odpovedi. Povedano drugače običajno tudi zanj obstaja neka življenjska doba, po izteku katere predvidoma ne more več vršiti svoje funkcije.



Slika 1.7: Neredundanten sistem z n entitetami.

"Cold stanby" ali pasivna redundanca z idealnim stikalom

Predpostavimo, da imamo opravka s "Cold standby" redundanco z idealno zanesljivim stikalom in štirimi nepopravljivimi komponentami, pri čemer breme servisira le ena, ostale tri pa so v rezervi ali pasivne. Slednje se vklapljajo zaporedoma ob odpovedih predhodno delujočih komponent. Graf Petrijeve mreže za opisani primer je predstavljen na sliki 1.8. Poseben pomen ima na sliki pogoj p_2, ki predstavlja dvignjeno zastavico za veljavnost izjave "prva komponenta še ni odpovedala" in onemogoča ponovno aktivacijo prve komponente po njeni odpovedi. Življensko dobo L sistema prikazanega na sliki 1.8 ponazorimo z



Slika 1.8: "Cold standby" sistem sestavljen iz štirih nepopravljivih entitet in idealnega stikala.

izrazom

$$L = L_1 + L_2 + L_3 + L_4. \tag{1.8}$$

"Cold stanby" ali pasivna redundanca z neidealnim stikalom

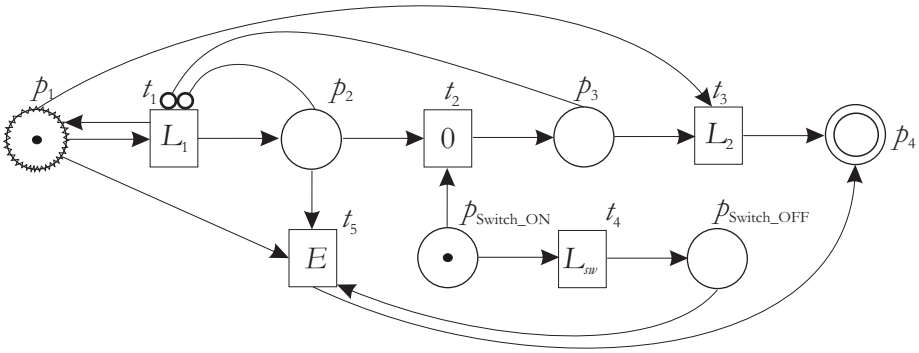
Predpostavimo, da imamo opravka z redundantnim sistemom z dvema nepopravljivima entitetama, pri čemer je za normalno delovanje sistema potrebna le ena. Ob zaznavi odpovedi prve entitete breme na drugo entiteto preklopi nei-

dealno zanesljivo stikalo. Prva v sistemu funkcijo opravlja entiteta z življensko dobo L_1, ko pa njena življenska doba poteče (pride do okvare), pa preide iz pasivnega v aktivno stanje druga entiteta z življensko dobo L_2. Model opisanega redundantnega sistema je prikazan na sliki 1.9. Ob veljavnosti relacije

$$L_{sw} > L_1, \tag{1.9}$$

kjer L_{sw} predstavlja življensko dobo stikala, se življenska doba sistema L izraža kot

$$L = L_1 + L_2. \tag{1.10}$$



Slika 1.9: "Cold standby" sistem sestavljen iz dveh nepopravljivih entitet in neidealnega stikala.

Poseben pomen na sliki 1.9 ima trajanje akcije t_5 E, ki predstavlja infinitezimalno kratko trajanje. Slednje je poljubno majhno, a večje kot nič. S tem dajemo prednost stikalu, da opravi svojo funkcijo pred prenosom žetona iz delujočega stanja sistema v nedelujoče stanje. Če stikalo ne deluje, sistem preide brez aktivacije druge entitete v nedelujoče stanje. Na sliki 1.9 vpeljemo nove *inhibirne povezave* (povezave s krožcem namesto puščice). Slednje akcijo prožijo le v primeru, če v vhodnem pogoju (pogoju iz katerega vodi ta povezava proti akciji) ni žetonov.

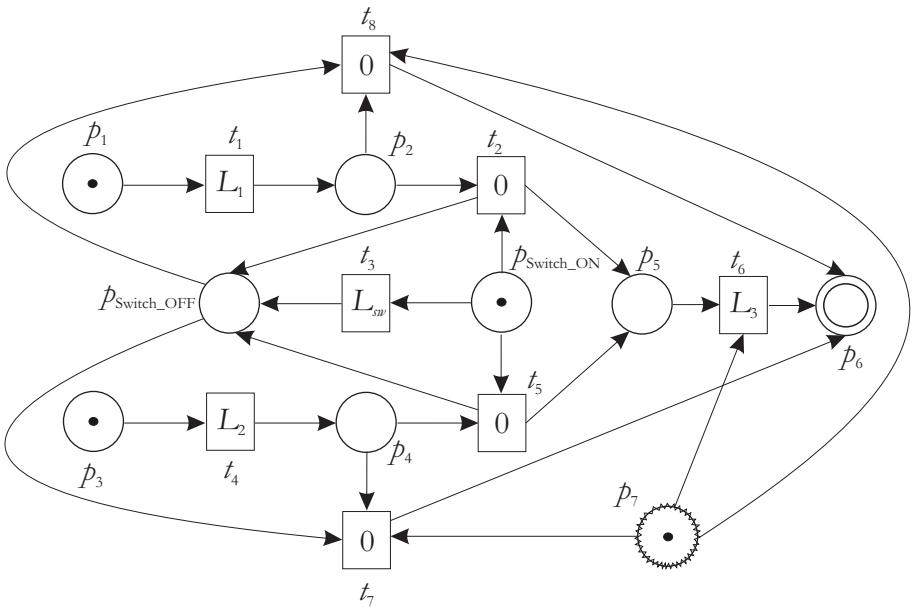
Tipično za "Cold standy" sisteme je, da lahko v primeru pasivne vključenosti posameznih obrabljivih entitet z zviševanjem njihovega števila poljubno podaljšujemo življensko dobo sistema kot celote.

Predpostavimo, da imamo opravka s sistemom, v katerem so tri nepopravljive entitete, pri čemer za normalno delovanje sistema potrebujemo le dve delujoči entiteti. Omenjeni sistem bi lahko imenovali za "2 out of 3 cold standby" sistem. Model opisanega sistema je prikazan na sliki 1.10. Na samem začetku breme servisirata entiteti 1 in 2, pasivno pa je vključena entiteta 3. Glede na prikazani model tako entiteto 1 ali 2 ob odpovedi zamenja entiteta 3, življenska doba tovrstnega sistema pa je podana z izrazom

$$L = \min(L_1, L_2) + \min(\max(L_1, L_2) - \min(L_1, L_2), L_3), \tag{1.11}$$

seveda samo v primeru, če velja relacija

$$L_{sw} > \min(L_1, L_2). \tag{1.12}$$



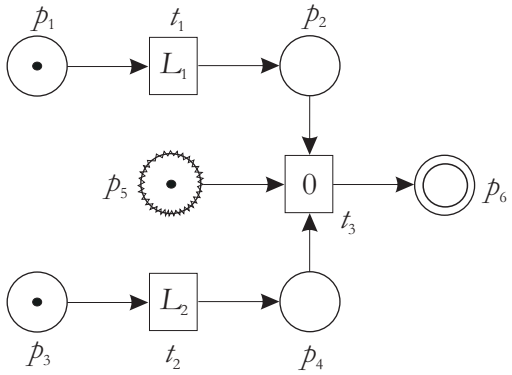
Slika 1.10: "2 out of 3 cold standby" sistem sestavljen iz nepopravljivih entitet in neidealnega stikala.

Poseben pomen imata povezavi, ki postavita stikalo v nedelujoč položaj po odpovedih prve ali druge entitete. S tem omogočimo prehod iz delujočega v nedelujoče stanje sistema, ko po vsej verjetnosti odpove preostala od prvih dveh entitet preje, kot tretja entiteta.

"Hot stanby" ali aktivna redundanca

Poleg "cold standby" ali pasivne redundance poznamo tudi "hot standby" ali aktivno redundanco. Slednje pomeni, da so vse entitete sicer aktivne, a vse ne servisirajo sistemskega bremena. Pri tem v primeru obrabljivosti entitet dosežemo predvsem eliminacijo odpovedi sistema, ki so posledica odpovedi "šibkejših" entitet, ki odpovedo pred predvidenim iztekom življenske dobe. Na sliki 1.11 je predstavljen model redundantnega "hot standby" sistema dveh nepopravljivih entitet z idealnim stikalom, ki omogoča identifikacijo odpovedi in preklop bremena na redundatno aktivno entiteto. Predvidena življenska doba tovrstnega sistema je definirana z izrazom

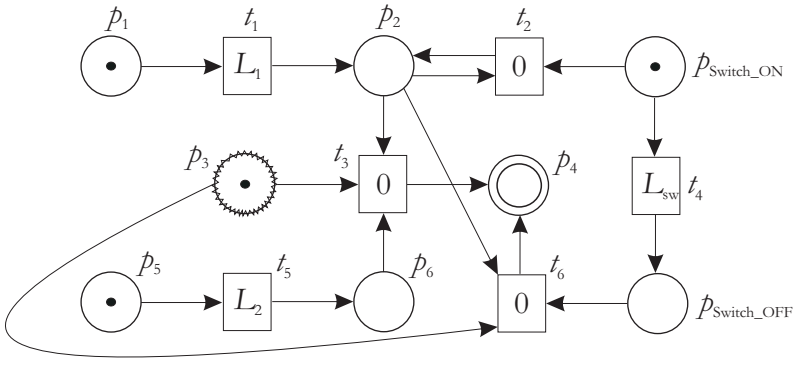
$$L = \max(L_1, L_2). \tag{1.13}$$



Slika 1.11: "Hot standby" sistem sestavljen iz dveh nepopravljivih entitet in idealnega stikala.

Če vpeljemo v predhodno opisani "1 out of 2 hot standby" redundančni sistem še nezanesljivost stikala, pridemo do modela sistema, ki je prikazan na sliki 1.12. Pri tem je najprej aktivna entiteta 1, entiteta 2 pa nastopa v "hot standby" aktivni redundanci. Neidealno stikalo po preklopu bremena odstranimo iz opazovanega sistema z odvzemom žetona. V tem primeru je predvidena življenska doba tovrstnega sistema definirana z izrazom

$$L_1 < L_{sw} : L = \max(L_1, L_2). \tag{1.14}$$

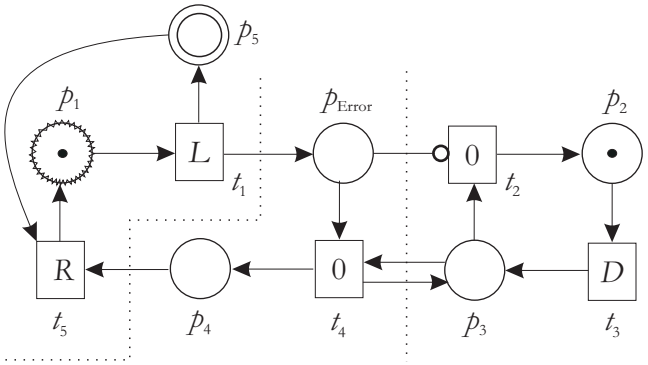


Slika 1.12: "Hot standby" sistem sestavljen iz dveh nepopravljivih entitet in neidealnega stikala.

1.1.5 Diagnosticiranje in servisiranje odpovedi posameznih entitet

V sistemih, kjer je odpoved posameznih entitet možna, je zelo pomembna prisotnost *diagnosticiranja*, ki nas zna na odpovedi bodisi opozoriti, ali celo prevzeti funkcijo stikala, ki preklopi tok delovnega bremena bodisi na že aktivno entiteto, ali pa aktivira pasivno entiteto in tok preusmeri nanjo. Eden od tovrstnih pogostih načinov je *periodično preverjanje* delovanja entitet, katerega frekvenca je odvisna od kritičnosti misije, ki jo sistem opravlja. Poglejmo si primer, kjer periodično diagnosticiranje opravljamo vsakih D časovnih period.

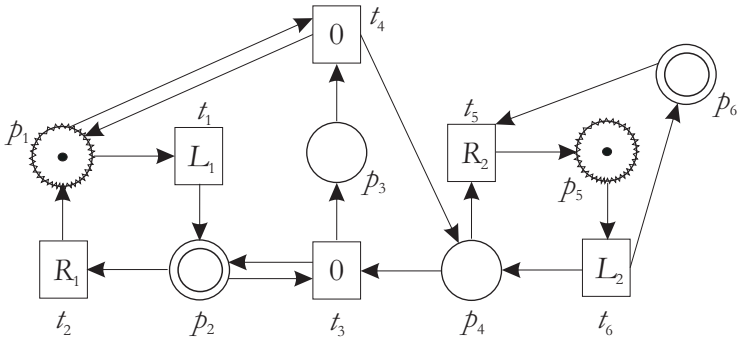
Model tovrstnega sistema je prikazan na sliki 1.13. Levi del slike predstavlja običajen popravljiv sistem s predvidenima dobama življenja (L) in servisiranja (R). Posebno vlogo igra pogoj z indeksom P_{Error}, ki predstavlja zastavico. Slednja se aktivira (v njej je žeton), ko pride do odpovedi entitete. Desni segment slike predstavlja krožno pot "diagnostika" s čimer je zagotovljena periodičnost preverjanja. Osrednji del slike omogoča hipen skok "diagnostika" v fazo servisiranja, s čimer se servisiranje sproži. Desni del zagotavlja časovno cikličnost diagnosticiranja.



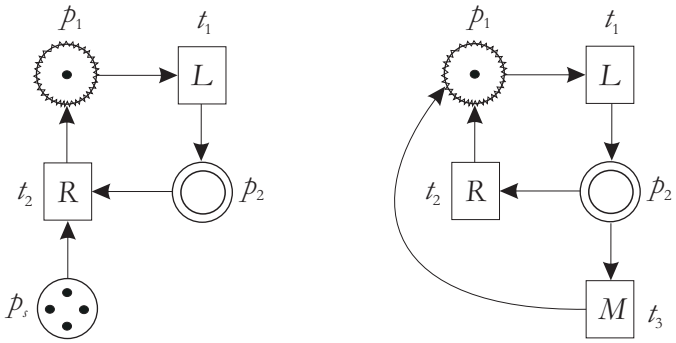
Slika 1.13: Periodična diagnostika delovanja entitete.

V nekaterih primerih imajo servisiranja entitet različne prioritete. Na sliki 1.14 imamo tako primer modela prekinjanja servisiranja entitete 2, ko nastopi odpoved entitete 1, ali povedano drugače ima serivisiranje entitete 1 prednost pred servisiranjem entitete 2. Zaradi preglednosti slike smo model poenostavili do te mere, da se je potrebno po povratku na servisiranje enitete 2 dela polotiti znova.

Pri servisiranju lahko naletimo na omejitve, ali pa imamo več različnih možnosti njegove izvedbe. Ena možnost je, da imamo na razpolago le končno število servisiranj (glej levo stran slike 1.15, kjer je to število ponazorjeno s številom žetonov v pogoju p_s), druga možnost pa je, da kombinirano samo servisiranje z enostavnejšimi zamenjavami, če je čas zamenjave M krajši od časa servisiranja R.



Slika 1.14: Prekinjanje servisiranja entitete 2 zaradi prenizke prioritete.

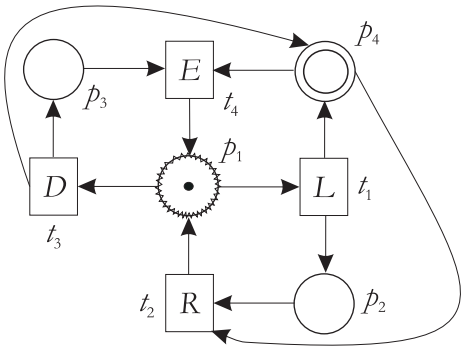


Slika 1.15: Servisiranje na osnovi razpoložljivosti resursov (levo) in opcijske menjave entitete (desno).

V primerih kompleksnejših sistemov se občasno poslužujemo strategije preventivnega vzdrževanja ali natančneje preventivne menjave posameznih entitet. Običajno imamo za to predpisane posebne procedure, ki imajo natanko določene časovne cikluse menjav. Na sliki 1.16 je prikazan model sistema, kjer menjamo entiteto vsakih D časovnih enot, samo trajanje menjave pa traja E časovnih enot. Če pred samo preventivno menjavo pride do odpovedi entitete, preide sistem v fazo popravila z običajnim servisnim časom R.

1.2 Funkcijska analiza

Za boljše razumevanje potencialnih odpovedi naj bi sistemski inženir poznal vse predvidene funkcije opazovanega sistema in njihove performančne značilnosti [1]. Razumevanje sistemskih funkcij je zajeto v sklopu kvalitativne metode *funkcijske analize*. Metoda se izvaja v sledečih korakih:



Slika 1.16: Preventivno menjavanje entitete pred iztekom njene življenske dobe.

- identifikacija vseh funkcij sistema,
- identifikacija vseh funkcij različnih *načinov delovanja* (angl. *operational modes*) sistema,
- hierarhična dekompozicija vseh funkcij sistema,
- analiza realizacije posameznih funkcij sistema,
- identifikacija povezanosti posameznih funkcij in
- identifikacija vmesnikov med posameznimi deli sistema ter sistemom in okoljem.

Pod pojmom načina delovanja sistema smatramo *normalen* način delovanja, *testni* način delovanja, način delovanja ob *prisotnosti odpovedi*, *degradiran način* delovanja itd.

Identificirane funkcije običajno razdelimo v naslednje skupine [1]:

- *esencialne* ali *primarne* funkcije sistema (angl. *essential functions*),
- *podporne* funkcije (angl. *auxiliary functions*), ki služijo kot podpora esencialnim,
- *zaščitne* funkcije (angl. *protective functions*), ki služijo kot osnova za zaščito ljudi, okolja in opreme,
- *informacijske* funkcije, ki služijo seznanjanju uporabnikov sistemov (alarmi, monitoriranje itd.),
- *vmesniške* funkcije, ki definirajo pomene vseh vmesnikov,
- *odvečne funkcije* (angl. *superfluous functions*), ki jih najdemo kot funkcijske gradnike, ki niso v uporabi (npr. del programske opreme, ki se nikdar ne izvaja itd.).

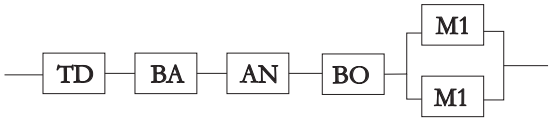
Množice funkcij si običajno niso tuje, saj lahko neka funkcija sodi v več skupin.

1.3 Zanesljivostni bločni diagrami

Zanesljivostni bločni diagrami (angl. *reliability block diagram* - RBD) so metoda za grafično predstavitev logičnih povezav sestavnih delov opazovanega sistema in njegovo zanesljivostno analizo [5], [6]. Osnove RBD diagramov smo opisali že v poglavju o teoriji zanesljivosti. Temeljijo na *serijskih* in *paralelnih vezavah* entitet, ki tvorijo sistem. Vsaka od povezanih entitet označena s četverokotnikom sovpada z eno od sistemskih funkcij ali z enim od sistemskih modulov.

Serijske vezave odražajo potrebnost delovanja vseh entitet v vezavi, paralelne vezave pa potrebnost delovanja vsaj ene od razpoložljivih entitet v posamezni vezavi.

Na sliki 1.17 je predstavljen enostavni RBD diagram vezave osnovnih sestavnih delov mobilnega telefona, pri čemer smo predpostavili, da je sestavljen iz zaslona na dotik (TD), baterije (BA), antene (AN), matične plošče z elektroniko (BO) in redundančne izvedbe pomnilnega medija (M1).



Slika 1.17: Enostaven RBD diagram mobilnega telefona.

Pri kompleksnejših RBD diagramih je zaradi preglednosti pomembna *metoda računske dekompozicije* bločnega diagrama. Na sliki 1.18 je prikazan primer kompleksnejšega RBD diagrama povzet po viru [6], izračun sistemske zanesljivosti pa je v dekompoziciji podan z izrazi

$$R_{sys}(t) = R_1(t) * R_2(t) * R_B(t) * R_{10}(t) * R_C(t), \tag{1.15}$$

$$R_B(t) = 1-[1-(R_3(t) * R_4(t) * R_5(t))]*[1-(R_6(t) * R_7(t) * R_8(t)]*[1-R_9(t)]\,, \tag{1.16}$$

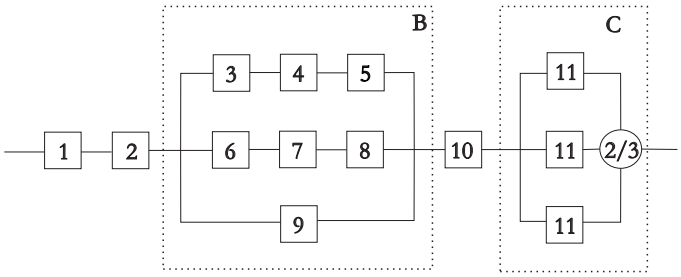
$$R_C(t) = 1-(1-R_{11}(t))^3 + 3R_{11}(t) * (1-R_{11}(t))^2. \tag{1.17}$$

Pri tem oznaka „2/3“ predstavlja „2 out of 3“ konfiguracijo sistema entitet z oznako 11.

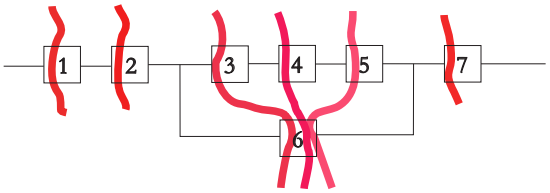
V RBD diagramih običajno iščemo najmanjše množice entitet (angl. *cut set*), pri katerih odpovedi vseh entitet v posamezni množici vodijo do odpovedi sistema kot celote [6]. Na sliki 1.19 so predstavljeni primer RBD diagrama in grafične označitve šestih minimalnih „cut set“ množic, za katere velja, da če odpovedo vse entitete posamezne množice, odpove tudi sistem kot celota. Minimalne množice za primer na sliki 1.19 so podane z izrazom

$$\{1\}\,, \{2\}\,, \{3,6\}\,, \{4,6\}\,, \{5,6\}\,, \{7\}\,. \tag{1.18}$$

Poleg iskanja minimalnih množic entitet v RBD diagramih, ki z odpovedmi vseh entitet vodijo v odpovedi sistemov, iščemo tudi takšne najmanjše množice

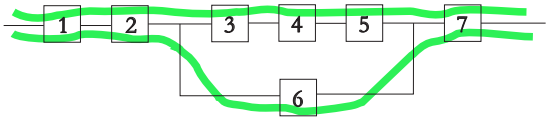


Slika 1.18: Primer kompleksnejšega RBD diagrama povzetega po viru [6].



Slika 1.19: Primer označitve minimalnih množic entitet, pri katerih odpovedi vodijo do odpovedi sistema kot celote.

entitet, znotraj katerih morajo vse entitete delovati, da bi sistem deloval kot celota (angl. *tie set*). Na sliki 1.20 je predstavljena grafična označitev dveh takšnih množic za predhodni primer. Minimalne množice za primer na sliki



Slika 1.20: Primer označitve množic entitet, znotraj katerih morajo vse entitete delovati, da bi sistem deloval kot celota.

1.20 so podane z izrazom

$$\{1,2,3,4,5,7\}\,,\{1,2,6,7\}\,. \tag{1.19}$$

V splošnem po [6] velja izraz za oceno sistemske zanesljivosti

$$1-\sum_{i=1}^{N}\prod_{j=1}^{n}(1-R_i(t)) < R_{sys}(t) < \sum_{i=1}^{T}\prod_{j=1}^{n}R_i(t), \tag{1.20}$$

pri čemer je N število „cut“ množic, T število „tie“ množic in n_j število entitet v j-ti „cut“ ali „tie“ množici. Navedeni izraz nam služi kot metoda hitre ocene

meja intervala, v katerem se nahaja vrednosti $R_{sys}(t)$ v kompleksnejših RBD diagramih.

RBD diagrami služijo običajno kot osnovni pogled na opazovani sistem, na katerem izvajamo zanesljivostno analizo. RBD prikaz je osnovni način predstavitve opazovanega sistema v večini programskih orodij, ki imajo integrirane različne metodologije ocene zanesljivosti (MIL, FMEA, FTA itd.). V okviru omenjenih orodij se posameznim entitetam RBD diagrama pripišejo (npr. izračunajo, ali pridobijo iz podatkovnih baz) ustrezne intenzivnosti odpovedovanj, posredno pa nato izračuna sistemska zanesljivost.

1.4 FMEA metoda

Ena od najpogosteje uporabljanih kvalitativnih metod za analizo in zviševanje zanesljivosti sistemov je *analiza načinov odpovedi in njihovih efektov* (angl. *failure mode and effect analysis* - FMEA) [7]. Kot metodologija predstavlja sestavni del mnogih standardov za doseganje kakovosti produktov, kot sta npr. ISO 9000 in QS-9000. Namenjena je tako za analizo *produktov*, kot tudi *delovnih procesov*, pri čemer ni usmerjena le na področje računalništva, temveč tudi na druga področja. Temelji na kritičnem pregledu produkta ali procesa in se izvaja v naslednjih korakih:

- identifikacija vseh možnih odpovedi produkta ali procesa,
- identifikacija tipičnih načinov odpovedi (angl. *failure modes*),
- analiza posledic posameznih načinov odpovedi (angl. *failure mode effects*),
- identifikacija efektov in rizikov, ki so posledice načinov odpovedi in
- ocena kritičnosti identificiranih načinov odpovedi.

Metoda je integrirana v večino programskih orodjih za ocenjevanje zanesljivosti.

1.4.1 Opis metode

V splošnem je cilj FMEA metode identifikacija odpovedi, njihovih posledic in s tem povezanih rizikov, ter posledično eventuelna eliminacija vzrokov odpovedi v produktu ali procesu. Posamezni način odpovedi, njene posledice in rizik ocenimo glede na naslednje *kriterije*:

- *resnost posledic* posamezne odpovedi S (angl. *severity of effects*),
- *pogostost pojavitev* posamezne odpovedi O (angl. *occurence of failures*),
- *možnost pravočasnega zaznavanja* odpovedi D in s tem posledično tudi eliminacije posledic in njihovega rizika (angl. *detection of failures*).

Vsak način odpovedi glede na predhodno podane kriterije subjektivno ocenimo z numeričnimi vrednostmi z intervala realnih števil $[1, 10]$. Z najmanjšo oceno 1 ocenimo minimalno (zanemarljivo) vrednost posameznega kriterija, z največjo oceno 10 pa ocenimo maksimalno vrednost posameznega kriterija. V tabelah 1.1, 1.2 in 1.3 so navedeni lingvistični opisi, ki so nam v pomoč pri izbiri numeričnih vrednosti predhodno omenjenih spremenljivk S, O in D. Tabele so povzete po viru [7]. Podatke v njih namerno navajamo v originalni obliki v angleškem jeziku, da ne bi ob prevodu spremenili njihovega pomena.

Resnost odpovedi (angl. *severity*)		
Ocena	Opis	Definicija
1	None	Failure would not be noticeable to the customer and would not affect the customer's process or product.
2	Very minor	Failure may not be readily appearent to the customer, but would have minor effects on the customer's process or product.
3	Minor	Failure would create a minor nuisance to the customer, but the customer can overcome it in the process or product without performance loss.
4	Very low	Failure can be overcome with modifications to the customer's process or product, but there is minor performance loss.
5	Low	Failure creates enough of a performance loss to cause the customer to complain.
6	Moderate	Failure results in a subsystem or partial malfunction of the product.
7	High	Failure causes a high degree of customer dissatisfaction.
8	Very high	Failure renders the unit inoperable or unfit for use.
9	Extremely high	Failure would create noncompliance with federal regulations.
10	Dangerously high	Failure could injure the customer or an employee.

Tabela 1.1: Primer opisnih kriterijev za določanje numerične ocene resnosti odpovedi [7].

Za poljuben produkt ali proces tako najprej identificiramo n možnih načinov odpovedi, nato pa za vsakega od njih subjektivno določimo vrednosti ocen S_i (resnost posledic), O_i (pogostost odpovedi) in D_i (možnost pravočasnosti zaznavanja odpovedi) $(i = 1, ..., n)$. Za vsak način odpovedi nato izračunamo vrednost spremenljivke RPN_i (angl. *risk priority number*) po izrazu

$$RPN_i = S_i * O_i * D_i, RPN_i \in [1, 1000]\,. \tag{1.21}$$

Pogostost odpovedi (angl. *occurence*)		
Ocena	Opis	Definicija
1	Remote: Failure is unlikely	One occurence in greater than five years or less than two occurences in one billion events.
2		One occurence every three to five years or 2 occurences in one billion events.
3	Low: Relatively few failures	One occurence every one to three years or 6 occurences in ten million events.
4		One occurence per year or 6 occurences in 100.000 events.
5		One occurence every six months to one year or one occurence in 10.000 events.
6	Moderate: Occasional failures	One occurence every three months or three occurences in 1.000 events.
7		One occurence every month or one occurence in 100 events.
8	High: Repeated failures	One occurence per week or a probability of 5 occurences in 100 events.
9		One occurence every three to four days or a probability of three occurences in 10 events.
10	Very high: Failure is almost inevitable	More than one occurence per day or a probability of more than three occurences in 10 events.

Tabela 1.2: Primer opisnih kriterijev za določanje numerične ocene pogostosti odpovedi [7]. Numerični podatki so odvisni od narave produkta ali procesa.

S tem dobimo n različnih RPN vrednosti, ki jih uredimo v padajočem vrstnem redu. Z vidika proizvajalca produkta ali upravljalca procesa je sedaj pomembno, da se odloči

- ali so načini odpovedi z vrha liste resnično kritični za zanesljivost produkta ali procesa in
- ali je finančni vložek za izboljšavo produkta/procesa, ki zniža oceno RPN_i za i-ti način odpovedi z vrha liste, sprejemljiv.

Z vidika kupca produkta ali upravitelja procesa bi bilo seveda idealno, da se vse vzroke posameznih načinov odpovedi, pri katerih je $RPN_i > 0$, odpravi. Slednje zahteva ustrezen finančni vložek, ki ga večinoma noben upravljalec procesov ali proizvajalec produktov ni pripravljen žrtvovati za doseganje visoke zanesljivosti svojega produkta ali procesa. Če se odločimo le za posamezne korekcije produkta/procesa, po izboljšavi segmenta, ki je vzrok za i-ti način odpovedi, analizo FMEA ponovimo. Pri tem pričakujemo, da se bo opazovani RPN_i zmanjšal in da se bo na vrh liste povzpel nek drug način odpovedi, ali pa bo tam isti način

Detekcija odpovedi (angl. *detection*)		
Ocena	Opis	Definicija
1	Almost certain	The defect is obvious or there is 100% automatic inspection with regular calibration and preventive maintenance of the inspection equipment.
2	Very high	All product is 100% automatically inspected.
3	High	An effective Statistical Process Control program is in place.
4	Moderately high	Statistical Process Control is used and there is immediate reaction to out-of-control conditions.
5	Moderate	Some Statistical Process Control is used in process and product is final inspected off-line.
6	Low	Product is 100% manually inspected using go/no-go or other mistake-proofing gauges.
7	Very low	Product is 100% manually inspected in the process.
8	Remote	Product is accepted based on no defectives in a sample.
9	Very remote	Product is sampled, inspected, and released based on Acceptable Quality Level sampling plans.
10	Absolute uncertainty	The product is not inspected or the defect caused by failure is not detectable.

Tabela 1.3: Primer opisnih kriterijev za določanje numerične ocene zmožnosti detekcije odpovedi [7].

odpovedi z nižjo vrednostjo RPN_i. Praviloma se lotimo popravkov vzrokov načinov odpovedi z najvišjimi RPN_i, ali tistih vzrokov načinov odpovedi z najbolj resnimi posledicami (največjimi S_i). Tipični primeri slednjih so načini odpovedi v sistemih, ki vršijo kritične misije.

Izvedbo FMEA metode razdelimo na sledeče faze:

1. temeljit pregled produkta/procesa,
2. identifikacija načinov odpovedi,
3. identifikacija posledic načinov odpovedi,
4. določanje S_i za vsak način odpovedi,
5. določanje O_i za vsak način odpovedi,
6. določanje D_i za vsak način odpovedi,

7. izračun RPN_i za vsak način odpovedi,
8. izdelava prioritetne lestvice odprave ali izboljšave vzrokov načinov odpovedi glede na vrednosti RPN_i in/ali S_i,
9. tehnološke izboljšave vzrokov načinov odpovedi z vrha lestvice,
10. ponoven izračun RPN_i vrednosti in tvorba nove urejene tabele (opcijski skok na točko 4).

Ena od orientacijskih mer za grobo zanesljivostno oceno produkta ali procesa, ki deloma prikriva naravo posameznih načinov odpovedi, je spremenljivka $TOTAL_{RPN}$. Izračuna se po izrazu

$$TOTAL_{RPN} = \sum_{i=1}^{n} RPN_i. \tag{1.22}$$

V kontekstu vpeljane orientacijske mere $TOTAL_{RPN}$ običajno vpeljemo tudi mejne (angl. *cut-off*) vrednosti $RPN_{cut-off_i}$. Vse vrednosti RPN_i, ki ne dosegajo ustrezne mejne vrednosti $RPN_{cut-off_i}$, ne participirajo k vsoti v izrazu (1.22).

Na posameznih specifičnih področjih, kot so npr. telekomunikacijski produkti, obstajajo sprejemljive mejne vrednosti spremenljivke $TOTAL_{RPN}$, preko katere se numerična ocena za posamezni produkt ne sme dvigniti.

1.4.2 Izvedba metode

Za potrebe izvedbe FMEA analize je potrebno sestaviti *ekipo ljudi*, ki jo bo izvedla. Običajno šteje takšna skupina 4 do 6 ljudi. Minimalno število je pogojeno s številom strokovnih področij, ki jih zaobsega posamezen produkt/proces. Tipična strokovna področja so npr. poznavanje standardov in materialov, poznavanje trga, poznavanje navad in kriterijev uporabnika itd. Različni člani v ekipo prinašajo različna znanja, izkušnje in videnja produkta/procesa. S tem je zagotovljena raznovrstnost ekipe, ki onemogoča neželjene dejavnike (npr. pristranskost) v analizi. Tipičen primer neželjenega dejavnika je sestava ekipe zgolj iz snovalcev produkta. Ti so v produkt v različnih fazah razvoja že vnesli napake, ki jih bodo v fazi analize bodisi neradi priznali, ali pa še bolj verjetno kar spregledali. Vodja ekipe mora tako formalno, kot tudi praktično dobro poznati koncept FMEA analize. V tem primeru ostalim članom FMEA metodologije ni potrebno poznati, saj jim konkretne naloge postavlja vodja. S tem ostalih članov ekipe ni potrebno dodatno izobraževati. Vodja ekipe mora imeti sposobnosti sklepanja kompromisov, vodenja projektne dokumentacije, generiranja idej in vodenja kritične diskusije, prenašanja konkretnih rezultatov v ustrezne končne predstavitve (grafi, diagrami, animacije, finančne ocene) itd.

Izvajanje FMEA metode ni trajen proces, temveč se vodi *projektno*. To pomeni, da ima izvedba analize predviden časovni začetek in konec, s čimer je časovno omejena. Pred začetkom analize je potrebno določiti

- časovni termin trajanja z začetkom in koncem,
- ekipo in njeno vodjo,
- finančna sredstva za izvedbo,
- cilje, ki jih lahko predstavlja le analiza, ali pa tudi predlogi popravkov z njihovo implementacijo,
- ustrezne ostale resurse (kontaktne osebe, materiale itd.),
- način podajanja rezultatov.

Z vidika produkta se FMEA metoda usmerja na

- varnostne rizike produkta (angl. *safety hazards*), pridobljene v fazi zasnove,
- produktne nepravilnosti (angl. *product malfunctions*), pridobljene v fazi izdelave in
- analizo preostalih pri snovanju nepredvidenih možnosti načinov odpovedovanja produkta,

z vidika procesa pa na

- vključevanje vseh dejavnikov procesa (ljudi, materialov, metodologij, okolja, opreme, itd.) in
- analizo vseh možnih načinov odpovedi procesa in posledično na analizo vpliva odpovedi procesa na lastnosti produktov.

Kot glavno slabost metode omenimo subjektivnost posameznega ocenjevalca pri dodelitvi posameznih ocen S_i, D_i in O_i. Subjektivnost se deloma izniči s povprečenjem ocen več ocenjevalcev.

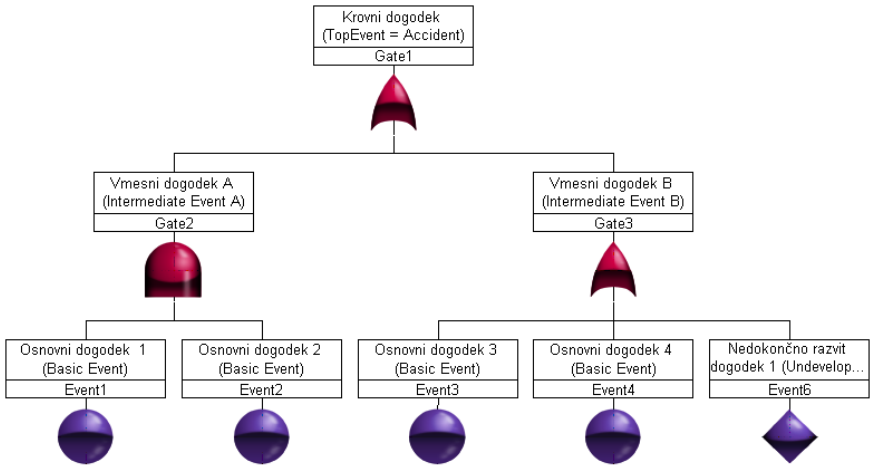
1.5 FTA metoda

FTA metoda ali *analiza dreves odpovedi* (angl. *fault tree analysis* - FTA) je namenjena kvalitativni analizi opazovanega sistema in s tem posredno zviševanju zanesljivosti sistemov [8]. Njen primarni cilj je identifikacija vseh možnih *načinov odpovedi* in identifikacija njihovih *vzrokov*. Načine odpovedi in njihove vzroke prikažemo s hierarhičnimi drevesi, kjer v korenih dreves nastopajo posamezni načini odpovedi (neželjeni dogodki), v listih dreves pa njihovi vzroki. Metoda je bila razvita v Bellovih laboratorijih l. 1962 za potrebe verifikacije produktov ameriške vojaške industrije, kot uveljavljeno metodo pa jo navajajo tudi mnogi MIL in IEC standardi.

1.5.1 Opis metode

Metoda temelji na identifikaciji n dreves, pri čemer število n sovpada s številom različnih načinov odpovedi. Posamezno drevo ponazarja logične odnose med tipom odpovedi in njegovimi vzroki. Slednji so povezani z logičnimi operatorji. V večini primerov se za logične operatorje uporabljata operaciji AND in OR, redkeje pa pragovna funkcija *k-out of n:Fail*, XOR itd.

Na sliki 1.21 je predstavljen enostaven primer FTA drevesa. Drevo je sestavljeno iz *listov* (osnovnih dogodkov - vzrokov), *vmesnih* (sestavljenih) dogodkov in *krovnega* dogodka (načina odpovedi). V spodnji vrstici slike poleg listov najdemo tudi *nedokončno razvit dogodek*. Gre za dogodek, za katerim v ozadju stoji samostojno drevo (poddrevo). Na sliki 1.22 je predstavljeno FTA drevo,



Slika 1.21: Primer splošnega FTA drevesa.

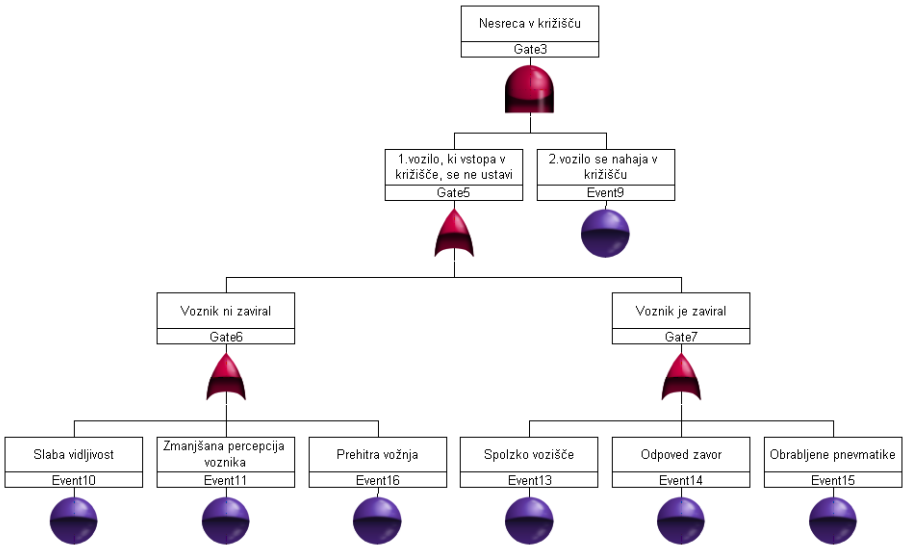
ki ponazarja eno od možnosti, ki privede do trčenja dveh vozil v križišču.

Metoda omogoča, da za porajanje vzrokov odpovedi (posameznih listov) določimo tudi verjetnosti, s čimer preko logičnih izrazov lahko pridemo do verjetnosti doseganja korenov dreves ali porajanja neželjenih dogodkov. Verjetnosti porajanja vzrokov naj bi izvirale iz meritev v fazah testiranja in eksploatacije. V splošnem lahko kreiranje drevesa ocenimo za „top-down“ koncept analize (najprej definiramo neželjene dogodke, potem pa hierarhično določamo vzroke in njihove povezave, ki vodijo do neželjenih dogodkov - načinov odpovedi).

Osnovna tendenca pri zasnovi zanesljivega sistema je v minimizaciji števila OR vrat in maksimizaciji števila AND vrat v posameznem drevesu. Slednja naj bi imela čimveč vhodov (vpeljava čimvečje redundance).

Liste kot vzroke pojavitev odpovedi klasificiramo v tri skupine in sicer v

- odpovedi opreme (angl. *equipment failures*),



Slika 1.22: Primer FTA drevesa, ki ponazarja nesrečo v križišču.

- človeške napake (angl. *human failures*) in
- zunanje dogodke.

Z vidika pojavitve krovnega neželjenega dogodka (korena) so pomembne tudi minimalne množice listov ali vzrokov (angl. *minimal cut sets*), ki še vodijo do odpovedi. Minimalne množice za primer drevesa s slike 1.21 so podane z izrazom

$$M_1 = \{E_1, E_2\}, M_2 = \{E_3\}, M_3 = \{E_4\}, M_4 = \{E_6\}. \tag{1.23}$$

1.5.2 Izvedba metode

Metodo izvajamo po naslednjih korakih:

- določimo sistema opazovanja,
- določimo n načinov odpovedi (angl. *top events*),
- za vsak način odpovedi ($i = 1, ..., n$) izvedemo naslednje:
  - zgradimo drevesno logično strukturo,
  - analiziramo vsako vejanje drevesa,
  - vsakemu listu določimo ustrezno verjetnost pojavitve,
- določimo „minimal cut-set“ množice za vsako od n dreves.

Tudi FTA metoda prinaša določene slabosti. Le te so sledeče:

- eno FTA drevo analizira le en tip odpovedi; odtod sledi, da moramo definirati relativno veliko število dreves;
- kot v primeru FMEA metode, smo tudi v primeru FTA metode soočeni s problemom subjektivnosti ocenjevanja;
- metoda zahteva veliko znanja in statističnih podatkov;
- praksa kaže, da se z vidika definiranja listov zanemarja analiza izvorov s strani človeškega faktorja in s strani zunanjih dogodkov.

1.6 Namen uporabe kvalitativnih metod

Predstavljene kvalitativne metode služijo kvalitativni zanesljivostni analizi opazovanega sistema. Uporabnika navajajo k kritičnemu pogledu na opazovani sistem, njegove funkcije, sestavne dele, identifikacijo delujočih in nedeljujočih stanj, uporabo redundance itd.

Literatura

[1] M. Rausand and A. Hoyland, *System reliability theory: Models, statistical methods, and applications.* John Wiley & Sons, Inc., 2004.

[2] M. Mraz and M. Moškon, *Modeliranje računalniških omrežij.* Založba FE in FRI, Ljubljana, Slovenija, 2012.

[3] J. L. Peterson, *Petri Net Theory and the Modeling of Systems.* Prentice Hall, 1981.

[4] W. Schneeweiss, *Petri Nets for Reliability Modeling.* LiLoLe Verlag GmbH, Nemčija, 1999.

[5] M. Xie, Y. S. Dai, and K. L. Poh, *Computing systems reliability: Models and analysis.* Kluwer Academic, 2004.

[6] P. D. T. Connor, *Practical reliability engineering.* John Wiley and Sons Ltd., 2002.

[7] R. McDermott, R. Mikulak, and M. Beauregard, *The basics of FMEA.* Productivity Inc., Portland, ZDA, 1996.

[8] W. Schneeweiss, *The Fault Tree Method.* LiLoLe Verlag GmbH, Nemčija, 1999.