Vanr. prof. dr Ingmar Bešić
Odsjek za računarstvo i informatiku
Elektrotehnički fakultet Univerziteta u Sarajevu

Broj: _____
U Sarajevu, 01.09.2022. godine

VIJEĆU DOKTORSKOG STUDIJA
ZA OBLAST RAČUNARSTVO I INFORMATIKA

PREDMET: **Zahtjev za pokretanjem procedure odbrane naučnoistraživačkog seminara 1.1**

U skladu sa članom 3. Odluke Vijeća Elektrotehničkog fakulteta u Sarajevu o definiranju procedure realizacije naučnoistraživačkih seminara na trećem ciklusu studija – doktorskom studiju (broj: 01-503/21 od 01.02.2021. godine), podnosim slijedeći

ZAHTJEV

za pokretanjem procedure odbrane naučnoistraživačkog seminara 1.1. kandidata Ehlimane Krupalije na temu „Unaprjeđenje procesa specifikacije elemenata uzročno-posljedičnih grafova koristeći grafički softverski alat“.

OBRAZLOŽENJE

U svojstvu akademskog savjetnika mišljenja sam da je kandidat Ehlimana Krupalija ispunila postavljeni zadatak i sve obaveze u skladu sa definiranim planom rada, uključujući objedinjavanje dostupne literature, razvoj vještina za pisanje naučnoistraživačkih radova i produbljenje znanja iz naučne oblasti Računarstva i informatike, te je napisala seminarski rad u okviru naučnoistraživačkog seminara 1.1 koji sadrži pregled literature oblasti u kojoj student planira realizirati svoje doktorsko istraživanje i tematski pripada oblasti studija studenta.

Saglasan sam i potvrđujem da je student ispunio sve obaveze u okviru odgovarajućeg naučnoistraživačkog seminara, odnosno predlažem pokretanje procedure odbrane naučnoistraživačkog seminara na temu „Unaprjeđenje procesa specifikacije elemenata uzročno-posljedičnih grafova koristeći grafički softverski alat“. Uz ovaj zahtjevu prilažem seminarski rad kandidata napisan u okviru naučnoistraživačkog seminara.

Predlaže se slijedeći termin odbrane naučnoistraživačkog seminara: utorak 04.10.2022 godine sa početkom u 10:00 u prostorijama Elektrotehničkog fakulteta Univerziteta u Sarajevu.

S poštovanjem,

Vanr. prof. dr Ingmar Bešić

Unaprjeđenje procesa specifikacije elemenata uzročno-posljedičnih grafova koristeći grafički softverski alat

Ehlimana Krupalija
Elektrotehnički fakultet
Univerzitet u Sarajevu
Sarajevo, Bosna i Hercegovina
ekrupalija1@etf.unsa.ba

Ingmar Bešić
Elektrotehnički fakultet
Univerzitet u Sarajevu
Sarajevo, Bosna i Hercegovina
ingmar.besic@etf.unsa.ba

Sažetak*—Uzročno-posljedični grafovi su jedna od često korištenih tehnika *black-box* testiranja sa mnogobrojnim praktičnim primjenama. Proces pretvaranja sistemskih zahtjeva u specifikacije uzročno-posljedičnih grafova važno je učiniti što je moguće bržim i jednostavnijim, prije upotrebe odgovarajućih algoritama za pretvaranje tako kreiranih specifikacija u tabele testnih slučajeva za *black-box* testiranje. U okviru ovog rada nalazi se detaljan pregled literature relevantne za oblast *black-box* testiranja i značaj uzročno-posljedičnih grafova u okviru ove oblasti. Nakon toga, predstavljen je grafički softverski alat za za kreiranje specifikacija uzročno-posljedičnih grafova. Alat posjeduje intuitivan korisnički interfejs i koristi standardnu prihvaćenu grafičku notaciju za predstavljanje različitih vrsta čvorova, logičkih relacija i ograničenja između čvorova grafa. Na ovaj način moguće je kreirati grafički prikaz željenih uzročno-posljedičnih grafova, a alat podržava i mogućnost zapisivanja i ponovne upotrebe kreiranih specifikacija. Svrha ovog rada je olakšavanje procesa specifikacije uzročno-posljedičnih grafova za korisnike, kako bi se uklonili neki od postojećih problema koji nastaju zbog pogrešnog razumijevanja elemenata uzročno-posljedičnih grafova. Predloženi alat evaluiran je koristeći uzročno-posljedične grafove različitih veličina iz standardne literature i koristeći druge dostupne alate. Alat je uspješno iskorišten za kreiranje uzročno-posljedičnih grafova svih veličina, dok su dostupni alati omogućili ograničene i nestandardizovane, ili nikakve grafičke reprezentacije.

Ključne riječi— uzročno-posljedični grafovi, grafički softverski alat, black-box testiranje, osiguranje kvaliteta softvera

I. UVOD

Testiranje softvera je veoma značajan proces za osiguranje kvaliteta softvera, zbog čega se željeni sistem treba testirati u toku implementacije, kao i nakon njenog završetka [1]. Uzročno-posljedični grafovi su 1970. godine predstavljeni kao nova tehnika *black-box* testiranja [2], s ciljem automatskog generisanja testnih slučajeva za željene funkcionalnosti softverskih proizvoda. Ova tehnika je kasnije usvojena kao standardna metoda *black-box* testiranja koja sadrži tri vrste elemenata: čvorove (uzroke i posljedice), logičke (Booleove) relacije i ograničenja zasnovana na ovisnosti između različitih čvorova iste vrste. Nakon kreiranja specifikacije uzročno-posljedičnog grafa, različiti algoritmi se mogu iskoristiti za pretvaranje tih specifikacija u tabele testnih slučajeva koje se zatim koriste za *black-box* testiranje željenog sistema [3] [4] [5].

Najznačajnija poteškoća pri korištenju uzročno-posljedičnih grafova nastaje zbog visoke dimenzionalnosti tabela testnih slučajeva (ukupni broj testnih slučajeva je $2^{\text{broj uzročnih čvorova}}$). Iz tog razloga, veoma je važno izvršiti ispravnu specifikaciju svih logičkih relacija i ograničenja između uzročnih i posljedičnih čvorova, kako bi se broj izvršivih testnih slučajeva smanjio koliko god je to moguće. Nažalost, definicija izvornog algoritma za pretvaranje specifikacija uzročno-posljedičnih grafova u tabele testnih slučajeva, kao i tabele istinitosti za ograničenja, su veoma neprecizno definisani, zbog čega ih korisnici često pogrešno shvataju i koriste. Nedostatak verifikacije usklađenosti specifikacije sistema sa kreiranim uzročno-posljedičnim grafom dovodi do mnogih problema sa izvođenjem testnih slučajeva iz tako kreiranih specifikacija. Svi prethodno opisani problemi su detaljno opisani u [6].

U ovom radu predstavljen je novi grafički softverski alat za kreiranje specifikacija uzročno-posljedičnih grafova. Alat je dizajniran za definisanje svih elemenata od kojih se uzročno-posljedični grafovi sastoje, uključujući različite vrste čvorova, logičkih relacija i ograničenja. Očekivani doprinosi rada uključuju:

- Prikaz detaljnog pregleda literature iz oblasti *black-box* testiranja, uključujući uzročno-posljedične grafove, njihov značaj i primjenu, kao i prednosti i mane ove tehnike u odnosu na druge korištene tehnike *black-box* testiranja.
- Uvođenje novog softverskog alata koji koristi standardnu grafičku notaciju za elemente uzročno-posljedičnih grafova. U okviru alata omogućeno je zapisivanje kreiranih specifikacija grafova u *.txt* formatu, kao i ponovno korištenje tako kreiranih specifikacija.
- Analiza skalabilnosti pristupa predstavljenog u okviru grafičkog alata za mogućnost specifikacije uzročno-posljedičnih grafova svih veličina koji su prethodno kreirani i prikazani u relevantnoj literaturi.
- Evaluacija novog alata koristeći dostupne softverske alate u odnosu na način korištenja grafičkih elemenata.

U poglavlju II objašnjen je pojam uzročno-posljedičnih grafova, kao i glavnih elemenata od kojih se grafovi sastoje. U okviru poglavlja III diskutuje se sav prethodno izvršeni rad u datoj oblasti s ciljem akcentiranja značaja uzročno-posljedičnih grafova kao tehnike *black-box* testiranja. U okviru ovog poglavlja detaljno je opisan historijski razvoj tehnike uzročno-posljedičnih grafova, kao i prethodno izvršeni rad na razvoju grafičke notacije i različitih softverskih alata namijenjenih za pomoć pri pravljenju specifikacija uzročno-posljedičnih grafova. Poglavlje IV objašnjava grafički korisnički interfejs alata za kreiranje specifikacija uzročno-posljedičnih grafova. U poglavlju V vrši se evaluacija grafičkog alata koristeći različite primjere iz relevantne literature, pri čemu se vrši usporedba generisanih specifikacija i njihove usklađenosti sa standardnom

prihvaćenom grafičkom notacijom u odnosu na postojeće pristupe. Poglavlje VI sadrži sveobuhvatnu analizu softverskog alata i uspješnosti procesa specifikacije. U ovom poglavlju dati su prijedlozi za dalje istraživanje, kao i moguća unaprjeđenja novog alata.

II. Elementi uzročno-posljedičnih grafova

Uzročno-posljedični graf je 1970. godine definisan kao pojednostavljeni Booleov graf koji sadrži formulaciju logičkih funkcija za sistemske zahtjeve [2]. Specifikacija uzročno-posljedičnih grafova i elementi od kojih se isti sastoje definisani su i detaljno objašnjeni u mnogim radovima poput [3] i [4]. U ovim radovima uzročno-posljedični graf definiše se kao kombinatorna logička mreža koja sadrži reprezentaciju formalnih sistemskih zahtjeva u obliku logičkih veza između ulaznih i izlaznih varijabli (uzroka i posljedica). Svaki uzročno-posljedični graf sastoji se od tri različite vrste elemenata: čvorova, logičkih relacija i ograničenja. Svi čvorovi grafa (bez obzira na njihov tip) mogu biti u jednom od dva stanja – aktivno (1) ili neaktivno (0). Uzročno-posljedični grafovi mogu sadržavati tri različite vrste čvorova:

1) Uzroci, koji se koriste za opisivanje različitih varijabli ili događaja koji rezultuju aktivacijom posljedica u sistemu. Uzročni čvorovi se uvijek postavljaju na lijevu stranu grafa. Označavaju se oznakom C_i (gdje $i > 0$ predstavlja broj čvora). Svaki uzročno-posljedični graf mora imati barem jedan uzročni čvor.

2) Posljedice, koje se koriste za opisivanje različitih varijabli ili događaja za čiju aktivaciju uzroci sistema predstavljaju okidač. Posljedični čvorovi se uvijek postavljaju na desnu stranu grafa. Označavaju se oznakom E_i (gdje $i > 0$ predstavlja broj čvora). Svaki uzročno-posljedični graf mora imati barem jedan posljedični čvor.

3) Međučvorovi, koji se koriste kao pomagači za opisivanje različitih logičkih relacija između uzroka i posljedica u sistemu. Svrha ove vrste čvorova je mogućnost ponovnog korištenja posljedica logičkih relacija kao uzroka za druge logičke relacije. Međučvorovi se uvijek postavljaju na sredinu grafa, između uzročnih i posljedičnih čvorova. Označavaju se oznakom I_i (gdje $i > 0$ predstavlja broj čvora). Korištenje međučvorova nije neophodno.

Postoji šest različitih logičkih relacija koje se mogu definisati između uzročnih i posljedičnih čvorova: **DIR** (direktna), **NOT** (negacija), **AND** (konjunkcija), **OR** (disjunkcija), **NAND** (Pierceova operacija) i **NOR** (Shefferova operacija). Direktna relacija i negacija su unarne, što znači da imaju samo jedan uzročni i jedan posljedični čvor, dok su ostale relacije *n*-arne, što znači da imaju $n > 1$ uzročnih čvorova i jedan posljedični čvor. Tabela istinitosti za svih šest relacija prikazana je u Tabeli I, gdje čvorovi (označeni kao N_1 i N_2) predstavljaju uzroke, a rezultat operacije predstavlja rezultujuću vrijednost aktivacije posljedičnog čvora (vrijednost 0 označava da posljedica nije aktivirana i obrnuto). Samo vrijednosti N_1 korištene su za određivanje rezultata direktne relacije i negacije, jer su ove dvije relacije unarne.

Postoji pet različitih ograničenja koja se mogu definisati između uzročnih ili posljedičnih čvorova: **EXC** (ekskluzivnost), **INC** (inkluzivnost), **EXC Δ INC** (ekskluzivna inkluzivnost), **REQ** (zahtijevanje) i **MSK** (maskiranje). Sva ograničenja osim MSK se definišu isključivo nad uzročnim čvorovima, dok se MSK ograničenje definiše isključivo nad posljedičnim čvorovima. Međučvorovi se ne mogu koristiti u ograničenjima. Ekskluzivnost, inkluzivnost i ekskluzivna inkluzivnost su *n*-arne, što znači da imaju $n > 1$ uzroka, dok su zahtijevanje i maskiranje binarne, što znači da imaju tačno dva čvora. Tabela istinitosti za svih pet ograničenja prikazana je u Tabeli II, gdje ovisno o tipu ograničenja, čvorovi N_1 i N_2 mogu predstavljati uzroke ili posljedice. Rezultat operacije predstavlja rezultujuću izvršivost testnog slučaja (vrijednost 0 označava da testni slučaj nije izvršiv i obrnuto).

TABELA I. Tabela istinitosti za sve logičke relacije

Uzroci		**Rezultujuća vrijednost posljedice**					
N_1	N_2	DIR	NOT	AND	OR	NAND	NOR
0	0	0	1	0	0	1	1
0	1	0	1	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	0	1	1	0	0

TABELA II. Tabela istinitosti za sva ograničenja

Vrijednosti čvorova		**Rezultujuća izvršivost testnog slučaja**				
N_1	N_2	EXC	INC	REQ	MSK	EXC Δ INC
0	0	1	0	1	1	0
0	1	1	1	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	0

III. Pregled literature

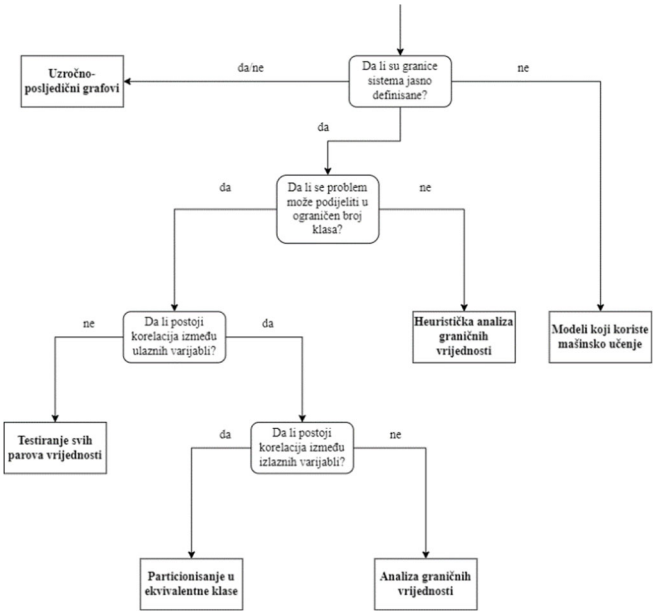
U ovom poglavlju dat je pregled relevantne literature za uzročno-posljedične grafove kao tehniku *black-box* testiranja. U poglavlju III.A. izvršeno je poređenje uzročno-posljedičnih grafova sa drugim tehnikama *black-box* testiranja. U poglavlju III.B. opisan je historijski razvoj oblasti, uključujući algoritme za pretvaranje specifikacija uzročno-posljedičnih grafova u testne module. Poglavlje III.C. opisuje prethodno izvršeni rad na razvoju grafičke notacije i različitih softverskih alata za specifikaciju elemenata uzročno-posljedičnih grafova.

A. Uzročno-posljedični grafovi kao tehnika black-box testiranja

Veoma je važno izvršiti analizu zbog čega se uzročno-posljedični grafovi trebaju koristiti za *black-box* testiranje umjesto drugih dostupnih tehnika. U [3] i [4] preporučeno je kombinovanje više različitih tehnika kada je god to moguće, jer svaka dostupna tehnika ima svoje dobre i loše strane. Budući da se većina problema sastoji od više odvojenih dijelova koji su prilagođeni za korištenje različitih tehnika *black-box* testiranja, iz tog razloga korištenje više tehnika umjesto jedne može poboljšati kvalitet testnih modula. Usporedba između različitih tehnika *black-box* testiranja izvršena je u okviru nekoliko preglednih radova poput [5], [7], [8] i [9]. U okviru ovih radova nalaze se preporuke kada se određene tehnike trebaju koristiti, a kada se trebaju izbjegavati. Pojednostavljeni dijagram ovog procesa prikazan je na Slici 1, a koncepti objašnjeni i objedinjeni u prethodno navedenim radovima biti će objašnjeni u nastavku.

Particionisanje u ekvivalentne klase je često korištena tehnika *black-box* testiranja koja se koristi kada se ulazne

varijable sistema mogu podijeliti u ograničen broj klasa, kao što je prikazano na Slici 1. Korištenje ograničenog broja klasa smanjuje veličinu testnog modula, ali rezultuje pokrivenošću sistema testovima koja nije adekvatna ukoliko se sličnost između ulaznih varijabli ne može mapirati na izlazne varijable. Iz ovog razloga, ova tehnika se najčešće kombinuje sa analizom graničnih vrijednosti. U okviru te tehnike koriste se granične vrijednosti između klasa (umjesto njihovih reprezentativnih vrijednosti), što omogućava da se i granice između izlaznih varijabli uzmu u obzir. Nedavni pristupi poput [10] i [11] pokušavaju ukloniti neophodnost kombinovanja particionisanja u ekvivalentne klase i analize graničnih vrijednosti, pritom koristeći heuristički pristup koji se može generalizovati za sisteme sa neograničenom domenom ulaznih vrijednosti (prikazano na Slici 1). I pored ovoga, svi takvi pristupi još uvijek zahtijevaju da su granice sistema eksplicitno definisane, što nije uvijek moguće. U slučajevima gdje su funkcionalne jedinice kompleksne, čak i iskusni testeri prave greške pri identifikaciji kategorija koristeći specifikacije sistema, kao što je opisano u [12]. Sličan zaključak je izveden u [13] za specifikaciju i izvršavanje testnih slučajeva u automobilskoj industriji.



Slika 1. Dijagram koji opisuje proces odabira tehnike *black-box* testiranja koja je najbolje prilagođena za dati problem

Još jedna tehnika namijenjena za poboljšavanje particionisanja u ekvivalentne klase je testiranje svih parova vrijednosti. Ova tehnika koristi ortogonalne nizove za definisanje testnih slučajeva koristeći sve različite parove reprezentativnih klasa, što dovodi do bolje distribucije testova u okviru domene testiranja. Problem nastaje kada postoji korelacija između više od dvije ulazne varijable, pri čemu se izostavlja veliki broj testnih slučajeva velike važnosti. Primjena metoda mašinskog i dubinskog učenja za *black-box* testiranje je također izvršena u prethodnom periodu, poput rekurentnih neuralnih mreža u [14], metoda mašinskog učenja u [15] (za detekciju sumnjivih dijelova koda) (eng. *code smells*) i [16] (za simulaciju znanja domenskih eksperata).

Uzročno-posljedični grafovi kao tehnika *black-box* testiranja ne posjeduju nijedan od prethodno navedenih nedostataka. Modeliranje sistema u obliku skupa uzroka i posljedica sa logičkim relacijama i ograničenjima ne dovodi do loše distribucije testova u okviru domene testiranja, niti ima ograničenja primjene samo na numeričke varijable i lako razdvojive klase. Iz tog razloga, na Slici 1 korištenje ovog pristupa preporučuje se bez obzira na granice sistema, tip i korelaciju između varijabli. Veličina testnog modula koja se dobiva kao rezultat korištenja ovog pristupa je najčešće zadovoljavajuća za sisteme sa velikim sigurnosnim zahtjevima (eng. *safety-critical systems*) za koje se prave sveobuhvatni i detaljni testni slučajevi (naprimjer, ovakav tip ugradbenih sistema opisan u [17] ili tip za navođenje projektila opisan u [18]). Na ovaj način omogućava se i testiranje različitih vrsta problema koji se ne mogu modelirati koristeći druge tehnike *black-box* testiranja (npr. sistemi koji nemaju eksplicitno definisane granice poput funkcionalnosti ugradbenih sistema koji se kontrolišu putem mašinskih interfejsa [19]).

Metode poput nasumičnog testiranja i testiranja baziranog na modelu predložene su za poboljšavanje automatizacije procesa testiranja [20] i mogu se primijeniti nad uzročno-posljedičnim grafovima za automatizaciju procesa selekcije testova. Ovo je omogućeno zbog toga što uzročno-posljedični grafovi posjeduju sva tri atributa neophodna za automatizaciju testova opisana u [21]: notaciju koja je jednostavna za korištenje, skalabilan algoritam za generisanje testnih slučajeva i kriterije za pokrivenost testovima koji se mogu koristiti za validaciju zahtjeva.

Glavni nedostatak pristupa koji koristi uzročno-posljedične grafove je neophodnost dobrog poznavanja sistema koji se modelira, što zahtijeva mnogo vremena, a najčešće i pomoć domenskog eksperta koji može ispravno odrediti uzroke, posljedice, logičke relacije i ograničenja sistema. Nedavno je napravljen pokušaj prevazilaženja ovog problema u [22], pri čemu je omogućena automatizacija pretvaranja sistemskih specifikacija u definicije uzročno-posljedičnih grafova, ali ovaj pristup je lokalizovan na korejski jezik.

Formalni parametri poput *statement ground*, koji je predložen u [23], mogu se koristiti kao pomoć pri određivanju sistemskih zahtjeva. Algoritmi poput *Ant-colony-based Generation of Test Cases* (AgeNT) [24] razvijeni su za svrhu automatskog generisanja funkcionalnih testova na osnovu formalnih sistemskih zahtjeva. Modeliranje okoline je predloženo u [25] za automatizaciju *black-box* testiranja ugradbenih sistema u realnom vremenu koji imaju ograničenja u pogledu vremenskih i sigurnosnih zahtjeva bez poznavanja dizajna sistema. Tehnike *mininga* podataka uspješno su primijenjene u [26] za pretvaranje sistemskih zahtjeva u specifikacije uzročno-posljedičnih grafova.

B. Historijski razvoj

Uzročno-posljedični grafovi izvorno su bili namijenjeni za *black-box* testiranje hardverskih logičkih kola [2]. U ovom radu predstavljen je prvi algoritam za kreiranje testnih slučajeva koristeći specifikacije uzročno-posljedičnih grafova koji radi na *brute force* principu (i koji se često naziva iscrpnim testiranjem [27]). U okviru ovog pristupa vršila bi se propagacija vrijednosti uzroka naprijed kroz graf, ali njegova implementacija nikada nije pokušana zbog hardverskih ograničenja vremena kada je rad pisan. Predloženi algoritam sadrži mnoge nekonzistentnosti. Svaki čvor može biti u jednom od tri stanja – aktivnom, neaktivnom ili nevažnom (eng. *don't care*), pri čemu se stanja obilježavaju sa 1, 0 i X respektivno. Kao rezultat toga, ukupni broj testnih slučajeva *t*

izračunava se koristeći jednačinu (1), gdje bazni broj 3 predstavlja ukupni broj mogućih stanja čvorova, a eksponent n_{uzroka} broj uzroka uzročno-posljedičnog grafa. Korištenje tri moguća stanja za čvorove u inicijalnom algoritmu rezultuje mnogo većim ukupnim brojem testnih slučajeva, nego što zaista ima mogućih kombinacija ulaznih vrijednosti. Zbog ove nekonzistentnosti, rezultujuće tabele testnih slučajeva u radu koji opisuje algoritam sadrže redundantne testne slučajeve (npr. 00001 i 00X01 su predstavljeni kao dva odvojena testna slučaja, iako 00X01 sadrži dva testna slučaja – 00001 i 00101).

$$t = 3^{n_{uzroka}} \quad (1)$$

U istom radu objašnjen je i prvi algoritam za generisanje testnih slučajeva koji koristi princip prolaska unazad kroz graf. Ovaj algoritam implementiran je u APL/360 programu koji je nazvan *Test Library Design Automation Program* (TELDAP). Algoritam radi na principu aktivacije posljedičnih čvorova, nakon čega se njihove aktivacije propagiraju unazad kroz graf kako bi se odredile vrijednosti uzročnih čvorova. Ovaj algoritam prevazišao je neke od prethodno navedenih nekonzistentnosti inicijalnog algoritma, prvenstveno jer se za izračun ukupnog broja testnih slučajeva *t* koriste dva moguća stanja čvorova (na način prikazan u jednačini (2)). Specifikacija uzročno-posljedičnog grafa se formuliše u obliku matrice, koja se zatim koristi za aktivaciju posljedičnih čvorova i iterativno određivanje vrijednosti uzročnih čvorova koje odgovaraju ovakvoj aktivaciji. Ovaj algoritam ujedno predstavlja i najčešće korišteni algoritam za određivanje testnih slučajeva koristeći specifikacije uzročno-posljedičnih grafova.

$$t = 2^{n_{uzroka}} \quad (2)$$

Slabosti inicijalnog algoritma koji koristi pristup prolaska unazad kroz graf analizirane su u [6]. Glavni problemi koji su opisani su:

- Inicijalni algoritam za generisanje testnih slučajeva koristeći specifikacije uzročno-posljedičnih grafova nije opisan na precizan način. Zbog toga sadrži mnogo nekonzistentnosti i dvosmislenosti opisane u prethodnom dijelu rada.
- Logičke relacije mogu se specificirati na više međusobno ekvivalentnih načina, što dovodi do različitih testnih modula za jedinstvenu specifikaciju sistema.
- Ukoliko se koriste tri moguća stanja čvorova, među kojima je *don't care* (X) stanje (a što je prisutno u nekim pristupima poput [3]), dolazi do pojave redundantnih testnih slučajeva.
- Različiti tipovi ograničenja nisu precizno definisani, što dovodi do njihove česte pogrešne upotrebe i rezultuje pogrešnim reprezentacijama sistema za koje se definišu uzročno-posljedični grafovi.
- Korištenje algoritma sa propagacijom vrijednosti čvorova unazad kroz graf garantuje da će testni modul sadržavati aktivacije posljedica, ali ne uzima se u obzir postojanje izvršivih testnih slučajeva bez aktivacija posljedica. Ovakvi testovi mogu biti važni za testiranje različitih vrsta sistema, ali inicijalni algoritam ih ne uključuje u testne module.

Zbog prethodno navedenih nedostataka algoritma sa propagacijom vrijednosti čvorova unazad kroz graf, izvršene su mnoge izmjene i pokušaji uklanjanja tih nedostataka. Ovi pristupi mogu se podijeliti u dvije kategorije na osnovu vrste optimizacije osnovnog algoritma [28] – pristupe za minimizaciju testnih modula, gdje se vrši pooštravanje kriterija za testni modul i pristupe za selekciju testnih slučajeva, gdje se vrši odabir testova koristeći željeni kriterij. Jedan takav pristup [29] koristi zasebna binarna stabla za predstavljanje svih posljedičnih čvorova i njihovih logičkih relacija sa uzročnim čvorovima. Nakon što se za svaki posljedični čvor zasebno generišu testni slučajevi, vrši se njihovo spajanje u jedinstveni modul, pri čemu se vrši dalja minimizacija modula, dajući prednost testnim slučajevima sa više aktivnih posljedica. Nedostatak ovog algoritma je memorijska kompleksnost operacije kreiranja binarnih stabala za svaki posljedični čvor, posebno u slučajevima kada je više posljedica povezano sa istim uzrocima putem različitih logičkih relacija.

Matematička formalizacija specifikacije uzročno-posljedičnih grafova izvršena je u [27] i [30]. U ovim radovima predstavljeni su novi algoritmi nazvani *Boolean operator testing* (BOR) i *Boolean constraint testing*. Njihova namjena je detekcija grešaka u relacijskim i Booleovim operatorima. Dalji razvoj ovog pristupa kako bi se smanjio broj prethodno navedenih dvosmislenosti i nekonzistentnosti u procesu specifikacije uzročno-posljedičnih grafova izvršen je u [31]. U ovom radu opisan je postupak kreiranja Booleovih izraza za dati uzročno-posljedični graf i formalni matematički postupak za generisanje testnih modula na osnovu tako formiranih izraza. Optimizacija ovog procesa izvršena je u [32] dodavanjem novog posljednjeg koraka u algoritam kako bi se broj testnih slučajeva u testnom modulu smanjio.

Mnogi pristupi poput [33] i [34] povezali su uzročno-posljedične grafove sa *Unified Modeling Language* (UML) modelima. U ovim radovima predloženi su načini automatske konverzije između ove dvije reprezentacije kako bi se problemi iz domene objektno-orijentisanog programiranja prenijeli u domenu uzročno-posljedičnih grafova, koja je prilagođena za *black-box* testiranje. Predloženi algoritmi koriste ekspresije *Object Constraint Language* (OCL) i ATLAS programskog jezika za kreiranje reprezentacija uzročno-posljedičnih grafova i generisanje testnih modula iz izvornih UML modela sistema.

U radu [35] uvedena je metrika pokrivenosti posljedičnih čvorova (eng. *Effect coverage* – EC) koja ima namjenu analize strukture podskupova testnih slučajeva. EC metrika određuje postotak aktivacija posljedica *Aktivacije* u odnosu na ukupni broj posljedica $n_{posljedica}$ u testnom modulu koji se analizira, na način prikazan u jednačini (3). Maksimalna vrijednost ove metrike je 100% i kvalitet testnog modula određuje se udaljenošću od njene maksimalne vrijednosti (što je vrijednost EC metrike testnog modula bliža 100%, to je veći kvalitet testnih slučajeva).

$$EC = \frac{\sum_{i=0}^{n_{posljedica}} Aktivacije[i] == 1}{n_{posljedica}} * 100\% \quad (3)$$

C. Prethodno izvršeni rad

Kao što je prethodno navedeno, u [3] i [4] preporučeno je korištenje uzročno-posljedičnih grafova za sisteme sa visokim sigurnosnim zahtjevima, kao i na druge tipove sistema koji zahtijevaju detaljne testne module koji su raspoređeni kroz cijeli sistem. Iz tog razloga, uzročno-posljedični grafovi primijenjeni su na veliki broj problema iz različitih oblasti

poput distribuisanih telekomunikacijskih sistema [36], automatizaciju procesa raspoređivanja studenata na univerzitet [37], testiranje procesa e-učenja [38], kontrolu navođenja projektila [22], železničke sisteme visokih brzina sa visokim sigurnosnim zahtjevima [39] i bankomate [40]. Zbog njihove sličnosti sa digitalnim sklopovima [4], uzročno-posljedični grafovi se mogu primijeniti na različite probleme koji zahtijevaju modeliranje logičkih relacija, kako bi se utvrdila izvršivost testnih slučajeva radi smanjenja veličine testnog modula. U nedavnim godinama uzročno-posljedični grafovi su upotrijebljeni i kao pomoć pri primjeni drugih vrsta tehnika, poput tehnike testiranja svih parova vrijednosti [41] i automatizacije procesa ekstrakcije sistemskih zahtjeva [22]. Široka primjena uzročno-posljedičnih grafova kao tehnike *black-box* testiranja naglašava značaj procesa ispravne specifikacije njihovih elemenata, što je imperativ za kreiranje *black-box* testova za željene sisteme.

Prva grafička notacija za uzročno-posljedične grafove i elemente od kojih se sastoje uvedena je u izvornom radu u [2]. U ovom radu međučvorovi se nazivaju „nepoznatim *dummy* čvorovima“ i opisano je svih šest logičkih relacija iz poglavlja II, kao i samo tri vrste ograničenja – ekskluzivnost, inkluzivnost i ekskluzivna inkluzivnost. Za opis logičkih relacija koriste se pune linije i slova (npr. A za konjunkciju, O za disjunkciju), a za opis ograničenja koriste se isprekidane linije i slova (npr. E za ekskluzivnost, I za inkluzivnost). Neki od ovih koncepata preuzeti su u daljim radovima poput [3] i [4], s tim da različiti radovi uvode različit broj logičkih relacija (npr. u [4] nisu prisutne Pierceova i Shefferova operacija). Oba prethodno navedena rada uvela su korištenje nedostajućih vrsta ograničenja koja su opisana u Tabeli II. Potpuno drugačija grafička notacija predložena je u [42], za svrhe adaptacije procesa specifikacije uzročno-posljedičnih grafova procesu specifikacije sistemskih zahtjeva. U okviru ovog rada koriste se apstraktni opisi čvorova (bez ikakve numeracije), ograničenja se zamjenjuju sa više različitih i bolje razrađenih vrsta povezanosti (pripadanje, svojstvo i interakcija), a čvorovi se nazivaju faktorima ili događajima. Sve ovo izvršeno je kako bi se proces specifikacije uzročno-posljedičnih grafova približio procesu specifikacije sistemskih zahtjeva sa ciljem smanjenja broja grešaka prisutnih u toku pretvaranja zahtjeva u specifikacije grafova.

Razvoj softverskih alata za pomoć pri procesu kreiranja uzročno-posljedičnih grafova započet je uvođenjem TELDAP-a (*Test Library Automation Program*) tokom 1970-ih godina [2]. Kao što je prethodno spomenuto, TELDAP je APL/360 program koji je prvi bio u mogućnosti procesirati definicije uzročno-posljedičnih grafova i pretvarati ih u tabele testnih slučajeva. Neki nedavno predloženi alati omogućavaju korisniku unos opisa svih čvorova, relacija i ograničenja direktno u korisnički interfejs [35] ili u obliku *.graphml* specifikacije [43], nakon čega se vrši automatsko generisanje testnih slučajeva. No, nijedan od ovih softverskih alata ne omogućava grafičko definisanje uzročno-posljedičnih grafova, jer su alati namijenjeni za primjenu različitih algoritama kako bi se za date specifikacije grafova kreirali moduli testnih slučajeva. BenderRBT je jedini dostupni alat koji koristi grafičke elemente pri kreiranju uzročno-posljedičnih grafova [44]. U okviru ovog alata omogućeno je definisanje uzročno-posljedičnih grafova, generisanje testnih modula za *black-box* testiranje, pa čak i pregled pokrivenosti koda testovima za svrhe *white-box* testiranja. Međutim, ovaj alat nije besplatan za korištenje već je komercijalne prirode, a u alatu se ne koristi standardna prihvaćena grafička notacija, što će biti detaljno analizirano u okviru poglavlja V. Problem nedostupnosti alata za testiranje softvera opisan je u [45], gdje je samo 14% svih alata bilo dostupno, a samo 1% njih je bilo besplatno za korištenje i *open-source*, što dodatno naglašava problem dostupnosti softverskih alata ove prirode.

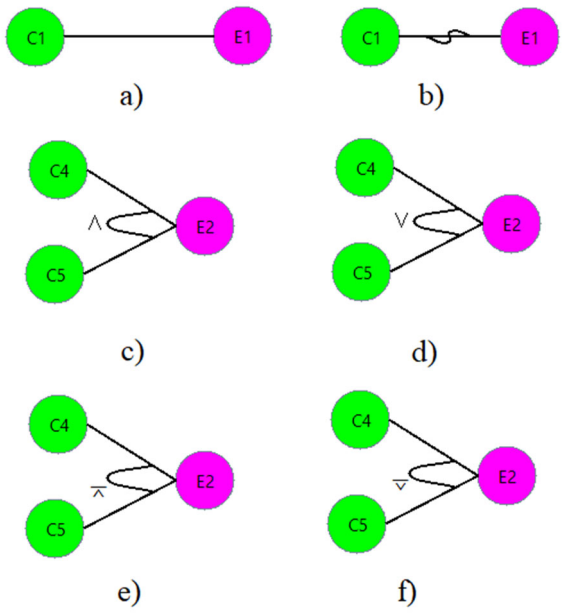
IV. Elementi predloženog softverskog alata

Kada se predloženi softverski alat pokrene, sastoji se od praznog prostora predviđenog za dodavanje čvorova, logičkih relacija i ograničenja uzročno-posljedičnih grafova. U poglavlju IV.A. opisane su funkcionalnosti alata kojima se omogućava kreiranje specifikacije uzročno-posljedičnih grafova. Poglavlje IV.B. opisuje funkcionalnost zapisivanja i ponovnog korištenja kreiranih specifikacija.

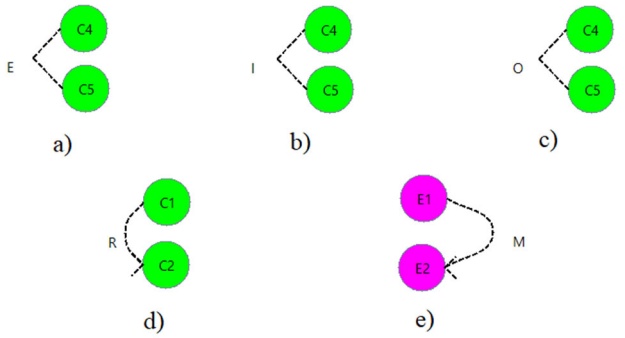
A. *Definisanje elemenata grafova*

Kako bi proces specifikacije uzročno-posljedičnih grafova bio što lakši za korisnike, u okviru predloženog alata omogućen je veliki broj funkcionalnosti za rad sa elementima grafova. Moguće je izvršiti dodavanje novih čvorova u graf, promjenu lokacije postojećih čvorova u okviru grafičke specifikacije, kao i brisanje postojećih čvorova. U okviru dodavanja i promjena lokacije koristi se intuitivna *drag-and-drop* operacija, a za brisanje se koristi lista postojećih čvorova, pri čemu je moguće izvršiti istovremeno brisanje jednog ili više čvorova. Pri dodavanju novih čvorova, novom čvoru uvijek se dodjeljuje najmanji pozitivni broj koji već nije dodijeljen nekom čvoru (što može biti veoma korisno nakon brisanja postojećih čvorova).

Slične funkcionalnosti alata omogućene su i za rad sa logičkim relacijama i ograničenjima – moguće je dodavati nove ili brisati postojeće relacije i ograničenja, pri čemu se za dodavanje vrši selektovanje željenih čvorova na prostoru gdje su isti grafički definisani, a za brisanje se koriste liste postojećih relacija i ograničenja. Grafičke reprezentacije svih vrsta logičkih relacija i ograničenja nakon njihovog dodavanja u okviru alata prikazane su na Slici 2 i 3 respektivno.



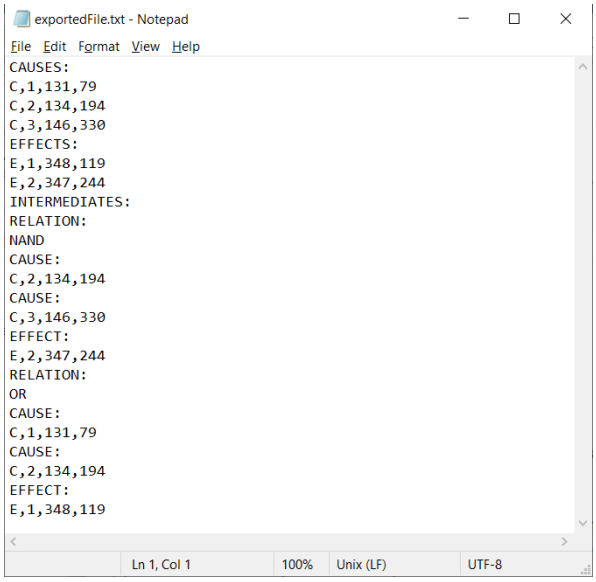
Slika 2. Način predstavljanja logičkih relacija u grafičkom softverskom alatu: a) DIR relacija, b) NOT relacija, c) AND relacija, d) OR relacija, e) NAND relacija, f) NOR relacija



Slika 3. Način predstavljanja ograničenja u grafičkom softverskom alatu: a) EXC ograničenje, b) INC ograničenje, c) EXC Δ INC ograničenje, d) REQ ograničenje, e) MSK ograničenje

B. *Ponovno korištenje specifikacija grafova*

Predloženi alat posjeduje funkcionalnost zapisivanja kreiranih specifikacija uzročno-posljedičnih grafova u *file exportedData.txt*, kao i njihovog ponovnog učitavanja u alat kako bi se mogla vršiti dalja specifikacija i željene izmjene. Kreirani *file* sadrži strukturu grafa (uključujući sve čvorove, logičke relacije i ograničenja, kao i njihove lokacije na grafu), kao što je prikazano na Slici 4. Ovako zapisani podaci su lako čitljivi i mogu se lako mijenjati i bez korištenja grafičkog alata.



Slika 4. Sadržaj *.txt file*-a u kojem je zapisana struktura uzročno-posljedičnog grafa kreirana koristeći predloženi grafički alat

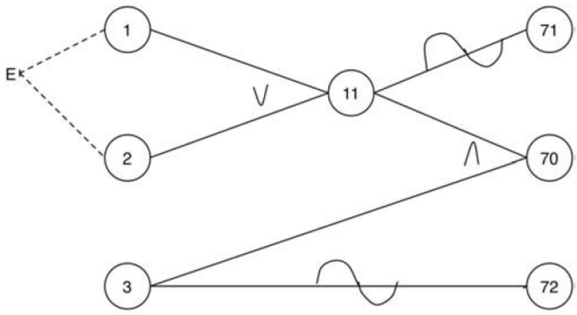
V. Evaluacija

Evaluacija predloženog grafičkog alata sastoji se od dva različita pristupa – provjere mogućnosti specifikacije različitih uzročno-posljedičnih grafova iz standardne literature, koja je obrađena u okviru poglavlja V.A, i usporedbe rezultata sa drugim dostupnim alatima, koja je obrađena u okviru poglavlja V.B.

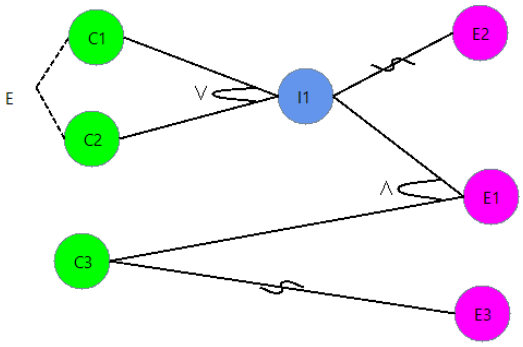
A. *Specifikacije standardnih uzročno-posljedičnih grafova*

Kako bi se validirala mogućnost uspješne specifikacije uzročno-posljedičnih grafova u predloženom grafičkom alatu, iskorišteni su sljedeći primjeri iz relevantne literature:

1) Mali uzročno-posljedični graf (koji se sastoji od ukupno $n = 6$ uzročnih i posljedičnih čvorova) iz [4], koji je prikazan na Slici 5. Prikazana reprezentacija koristi standardnu prihvaćenu grafičku notaciju i sadrži sve elemente uzročno-posljedičnih grafova, uključujući sve tri vrste čvorova, logičke relacije i ograničenja. Na Slici 6 prikazana je reprezentacija istog uzročno-posljedičnog grafa u predloženom alatu. Vidljivo je da su oba grafa skoro identična, budući da se u predloženom alatu koristi ista verzija notacije. Jedina razlika je prisutna u konvenciji imenovanja za čvorove – originalna reprezentacija koristi brojeve povezane sa sistemskim zahtjevima, dok se u predloženom alatu eksplicitno definiše tip i broj svakog pojedinačnog čvora.



Slika 5. Uzročno-posljedični graf iz [4] koji se sastoji od tri uzroka, tri posljedice, jednog međučvora, četiri logičke relacije i jednog ograničenja

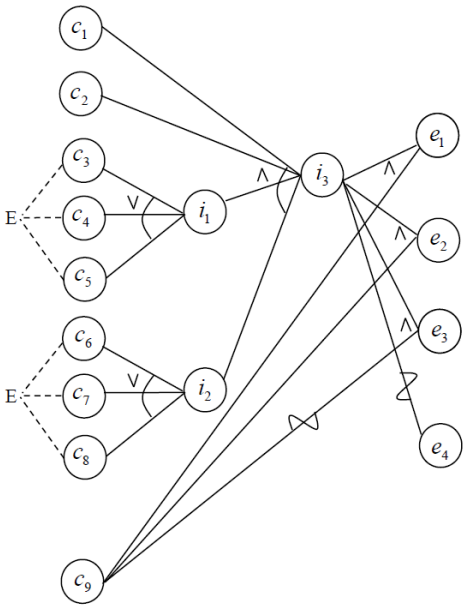


Slika 6. Reprezentacija uzročno-posljedičnog grafa sa Slike 5 u predloženom grafičkom softverskom alatu

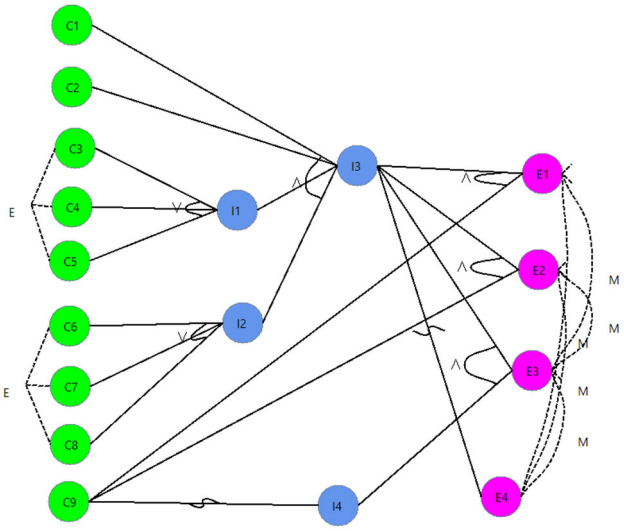
2) Uzročno-posljedični graf srednje veličine (koji se sastoji od ukupno $n = 13$ uzročnih i posljedičnih čvorova) iz [31], koji je prikazan na Slici 7. Prikazana reprezentacija također koristi standardnu prihvaćenu grafičku notaciju i sadrži sve elemente uzročno-posljedičnih grafova. Na Slici 8 prikazana je reprezentacija istog uzročno-posljedičnog grafa u predloženom alatu. Glavna razlika u odnosu na originalnu reprezentaciju je neophodnost korištenja MSK ograničenja između svih posljedičnih čvorova, što nije prikazano u originalnoj reprezentaciji, ali je implicirano jer su posljedice grafa međusobno isključive prema sistemskim zahtjevima. Dodatni međučvor I_4 dodan je u graf kako bi se rezultat NOT relacije mogao iskoristiti kao uzrok za drugu logičku relaciju, jer se NOT i AND relacije ne mogu kombinovati (što je učinjeno u originalnoj reprezentaciji).

3) Veliki uzročno-posljedični graf (koji se sastoji od ukupno $n = 25$ uzročnih i posljedičnih čvorova) iz [4] prikazan na Slici 9, koji se veoma često koristi za usporedbu

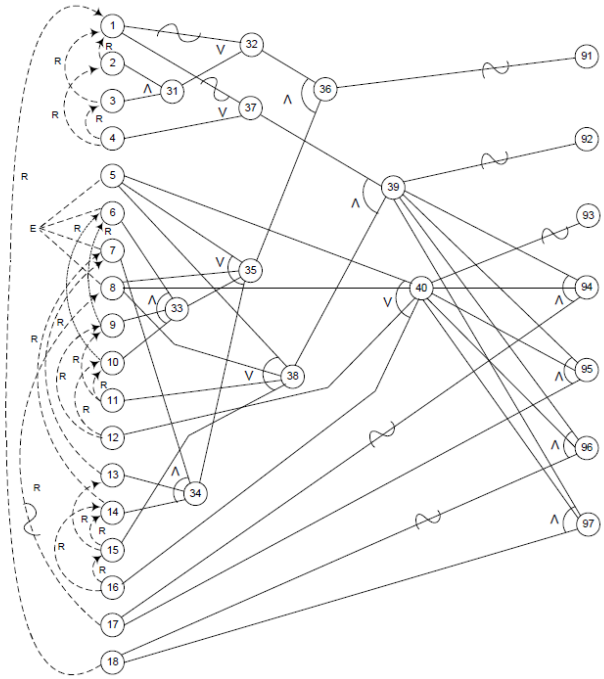
zbog svoje veličine i korištenja standardne prihvaćene grafičke notacije. Na Slici 10 prikazana je reprezentacija istog uzročno-posljedičnog grafa u predloženom alatu. Neke od razlika u odnosu na originalnu reprezentaciju već su spomenute u prethodnim primjerima, poput neophodnosti dodavanja novih međučvorova za kombinovanje unarnih i *n*-arnih logičkih relacija, kao i drugačije imenovanje čvorova. Glavna razlika se odnosi na neophodnost zamjene negiranog ograničenja zahtijevanja (NOT REQ) koje je originalno definisano između čvorova C_{17} i C_8. Korištenje ovog tipa ograničenja nije dozvoljeno u okviru predloženog alata (jer ograničenja nije moguće kombinovati sa logičkim relacijama, tj. nije dozvoljena negacija ograničenja zahtijevanja). Ova relacija je zamijenjena koristeći kombinaciju ograničenja EXC Δ INC (koje uklanja izvršivost dvije kombinacije C_{17}-C_8: 0-0 i 1-1) i REQ (koje uklanja izvršivost kombinacije C_{17}-C_8: 0-1, ostavljajući samo željenu kombinaciju C_{17}-C_8: 1-0 izvršivom).



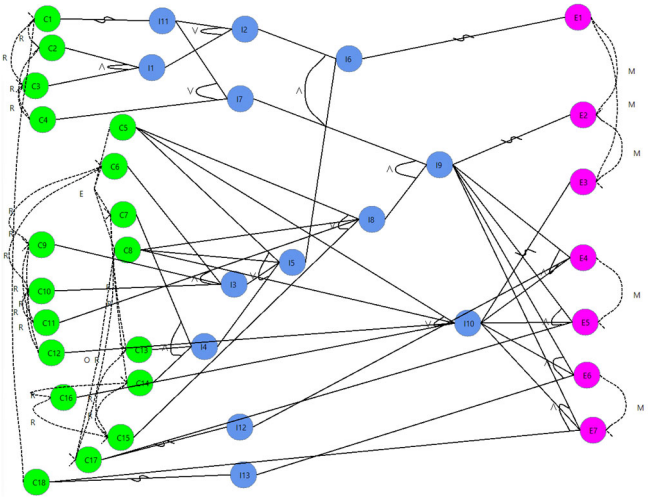
Slika 7. Uzročno-posljedični graf iz [31] koji se sastoji od devet uzroka, četiri posljedice, tri međučvora, sedam logičkih relacija i dva ograničenja



Slika 8. Reprezentacija uzročno-posljedičnog grafa sa Slike 7 u predloženom grafičkom softverskom alatu



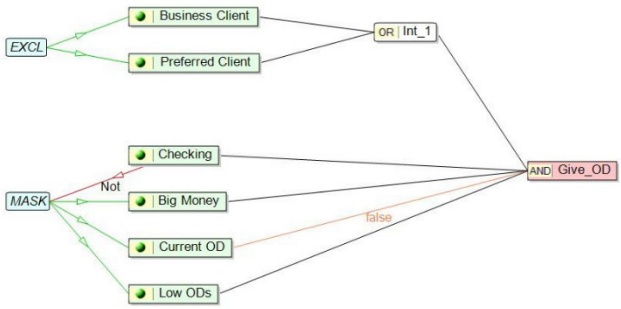
Slika 9. Uzročno-posljedični graf iz [4] koji se sastoji od osamnaest uzroka, sedam posljedica, trinaest međučvorova, dvadeset logičkih relacija i dvadest četiri ograničenja



Slika 10. Reprezentacija uzročno-posljedičnog grafa sa Slike 8 u predloženom grafičkom softverskom alatu

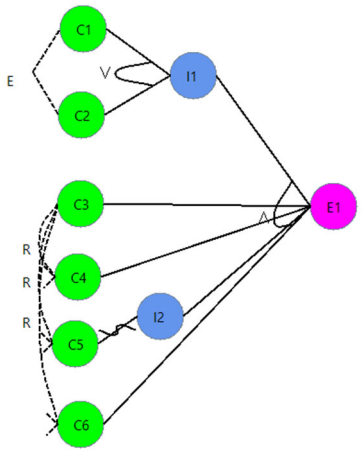
B. *Usporedba s drugim dostupnim alatima*

Jedini dostupni grafički alat BenderRBT [44] iskorišten je kako bi se razumljivost predloženog grafičkog alata usporedila sa postojećim pristupima i alatima. Ovaj alat omogućava korisnicima grafičko kreiranje specifikacija uzročno-posljedičnih grafova, ali se ne koristi standardna prihvaćena grafička notacija. Primjer grafa koji je kreiran koristeći ovaj alat [46] prikazan je na Slici 11, a specifikacija istog grafa u okviru predloženog alata prikazana je na Slici 12. Postoji veliki broj razlika u odnosu na originalnu reprezentaciju: u originalnoj reprezentaciji se ne koristi imenovanje čvorova, već opisi povezani sa sistemskim zahtjevima, kao i korištenje potpuno drugačije notacije za prikaz logičkih relacija i ograničenja.



Slika 11. Uzročno-posljedični graf iz [46] koji se sastoji od šest uzroka, jedne posljedice, jednog međučvora, dvije logičke relacije i dva ograničenja

Najznačajnija razlika u odnosu na originalnu reprezentaciju je neophodnost zamjene MSK ograničenja (koje je originalno definisano nad uzrocima, iako se ovo ograničenje smije primjenjivati samo nad posljedicama). Dodatno je nejasno značenje negacije koja je prisutna u okviru ovog ograničenja. U [44] je objašnjeno da se u BenderRBT negacija u okviru MSK ograničenja koristi za označavanje subjekta ograničenja, a ostalim vezama se predstavljaju objekti ograničenja. Ovakav opis nije u skladu sa standardnom definicijom MSK ograničenja iz Poglavlja II. Ovakva definicija zapravo označava REQ ograničenju, koje je u okviru predloženog alata primijenjeno nad odgovarajućim čvorovima grafa.



Slika 12. Reprezentacija uzročno-posljedičnog grafa sa Slike 11 u predloženom grafičkom softverskom alatu

Drugi dostupni alati poput TOUCH [43] (dostupan u [47]) i alata za testiranje uzročno-posljedičnih grafova [35] ne posjeduju grafičke elemente, jer je njihova glavna namjena primjena algoritama za pretvaranje specifikacija uzročno-posljedičnih grafova u testne module. Elementi uzročno-posljedičnih grafova se u okviru ovih alata definišu koristeći tekstualne ulaze ili *file*-ove. Iz tog razloga, predloženi alat nije bilo moguće usporediti sa drugim dostupnim alatima.

C. *Analiza rezultata*

Dobiveni rezultati ukazuju da se standardna grafička notacija za definisanje čvorova, logičkih relacija i ograničenja veoma često ne poštuje, jer se vrši dodavanje elemenata koji nisu definisani u okviru dostupne literature. Na taj način dobivaju se konfuzne specifikacije uzročno-posljedičnih grafova sa nejasnim izvršivostima testnih slučajeva. Primjeri takvih nepoštivanja prisutni su na Slici 7, gdje se unarne i *n*-arne relacije kombinuju bez upotrebe međučvorova, na Slici 9, gdje se logičke relacije kombinuju sa ograničenjima, kao i na Slici 11, gdje se ograničenja primjenjuju na pogrešan i zbunjujući način.

Prethodno objašnjeni problemi sa korištenjem standardne prihvaćene grafičke notacije su naglašeni u [6], gdje se neprecizne definicije i nedovoljno detaljna objašnjenja navode kao razlozi zbog kojih je proces specifikacije uzročno-posljedičnih grafova veoma podložan greškama. U [22] se također spominje ovaj problem i naglašava značaj usklađenosti specifikacije uzročno-posljedičnih grafova sa sistemskim zahtjevima. Dodatni problemi su uzrokovani nedostatkom standardizacije između različitih izvora literature i korištenja različitih vrsta logičkih relacija i ograničenja bez eksplicitnog definisanja tabela istinitosti.

Predloženi grafički softverski alat rješava prethodno spomenute probleme, jer ne dozvoljava korisnicima da ručno specificiraju ništa što nije podržano u okviru standardne literature. Iz ovog razloga, u okviru alata nije moguće koristiti međučvorove u okviru ograničenja ili vršiti kombinaciju logičkih relacija i ograničenja. Ovakva nepoštivanja standardne notacije nisu neophodna i mogu se remodelirati koristeći podržane standardne elemente, što je i demonstrirano u okviru svih primjera koji su korišteni za evaluaciju predloženog alata. Na ovaj način korisnicima se olakšava proces kreiranja specifikacija uzročno-posljedičnih grafova, pri čemu se mogu koristiti eksplicitno definisane tabele istinitosti za sve logičke relacije i ograničenja definisana u Poglavlju II. Ovo dovodi do strandardizovanih i lako razumljivih uzročno-posljedičnih grafova koji su u skladu sa standardnom prihvaćenom grafičkom notacijom i ne sadrže elemente koji su teški za razumjeti zbog nekonzistentnosti u odnosu na standardizovane elemente.

VI. ZAKLJUČAK

Uzročno-posljedični grafovi su popularna tehnika *black-box* testiranja koja se često koristi za kreiranje testnih modula neophodnih za izvršavanje *black-box* testova nad željenim sistemom. Problemi u okviru ovog procesa najčešće nastaju zbog nedovoljnog znanja i razumijevanja elemenata uzročno-posljedičnih grafova, što dovodi do neispravne upotrebe elemenata grafova pri kreiranju datih specifikacija uzročno-posljedičnih grafova za date sistemske zahtjeve. Nekoliko alata je kreirano za pomoć pri ovom procesu, no takvi alati se koriste za primjenu različitih algoritama za kreiranje tabela testnih slučajeva, dok samo jedan dostupni alat sadrži grafičke elemente i ne koristi standardnu prihvaćenu grafičku notaciju.

U ovom radu predložen je novi grafički alat za kreiranje specifikacija uzročno-posljedičnih grafova. Alat za cilj ima prevazilaženje postojećih poteškoća koje nastaju zbog slabog razumijevanja logičkih relacija i ograničenja uzročno-posljedičnih grafova. Alat sadrži intuitivni korisnički interfejs namijenjen da pomogne korisnicima pri brzom i efikasnom kreiranju uzročno-posljedičnih grafova, pritom koristeći standardnu grafičku notaciju, pritom zabranjujući upotrebu nestandardnh elemenata.

Usporedba između originalnih grafičkih reprezentacija uzročno-posljedičnih grafova i reprezentacija koje su kreirane koristeći predloženi alat pokazuje da je predloženi alat skalabilan i da se može koristiti za uspješno kreiranje uzročno-posljedičnih grafova svih veličina. Jedine razlike između

grafičkih reprezentacija rezultat su pogrešne upotrebe notacije u originalnim radovima, dok alat ne dozvoljava upotrebu takvih elemenata i ne dovodi do problema u razumijevanju elemenata grafova.

Rezultati dobiveni koristeći predloženi alat u usporedbi sa jedinim dostupnim grafičkim alatom pokazuju da su specifikacije grafova kreirane koristeći standardnu notaciju lakše za razumjeti u odnosu na pristupe koji ne koriste standardizovane elemente, posebno u odnosu na ispravnu primjenu međučvorova, logičkih relacija i ograničenja. Upotreba negacije u kombinaciji sa ograničenjem maskiranja zahtijevala je analizu korisničke dokumentacije BenderRBT alata kako bi se razumjelo šta ovakvo ograničenje predstavlja, a zatim je to ograničenje lako zamijenjeno sa ograničenjem zahtijevanja u predloženom alatu. Ovo znači da novi alat nudi intuitivnije i lakše razumljive izlaze za korisnike koji u svakom trenutku mogu koristiti tabele istinitosti za pomoć pri odabiru željenih logičkih relacija i ograničenja. Na ovaj način ohrabruje se korištenje standardne notacije i eksplicitnih definicija u odnosu na nestandardizovane pristupe.

Izlaz iz grafičkog alata je vizualna reprezentacija definisanog uzročno-posljedičnog grafa, kao i struktura grafa u *.txt* formatu koja se može iskoristiti za zapisivanje i ponovno korištenje željenog grafa. Korištenje ovako kreiranih specifikacija za kreiranje tabela testnih slučajeva u okviru grafičkog alata bi se trebalo omogućiti u budućnosti, kako bi se na taj način dodala nova funkcionalnost alatu. Na taj način alat bi postao još upotrebljivij i bilo bi ga moguće usporediti sa drugim dostupnim alatima koji već sadrže implementacije algoritama za kreiranje testnih modula koristeći uzročno-posljedične grafove.

LITERATURA

[1] P. M. Jacob i M. Prasanna, "A comparative analysis on black box testing strategies," u *2016 International Conference on Information Science (ICIS)*, Kochi, 2016.

[2] W. R. Elmendorf, "Automated design of program test libraries," IBM Technical Report TR 00.2809, 1970.

[3] S. L. Pfleeger i J. M. Atlee, Software Engineering: Theory and Practice, 4th ur., New Jersey: Pearson Higher Education, 2010.

[4] G. J. Myers, T. Badgett i C. Sandler, The Art of Software Testing, 3rd ur., New Jersey: John Wiley & Sons, Inc., 2012.

[5] S. Nidhra i J. Dondeti, "Black box and white box testing techniques - A literature review," *International Journal of Embedded Systems and Applications*, tom 2, br. 2, pp. 29-50, 2012.

[6] K. Nursimulu i R. L. Probert, "Cause-effect graphing analysis and validation of requirements," u *CASCON '95: Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research*, Toronto, 1995.

[7] M. E. Khan i F. Khan, "A comparative study of white box, black box and grey box testing techniques," *International Journal of Advanced Computer Science and Applications*, tom 3, br. 6, pp. 12-15, 2012.

[8] M. E. Khan, "Different approaches to black box testing technique for finding errors," *International Journal of Software Engineering & Applications*, tom 2, br. 4, pp. 31-40, 2011.

[9] N. Anwar i S. Kar, "Review paper on various software testing techniques & strategies," *Global Journal of Computer Science and Technology*, tom 19, br. 2, pp. 43-49, 2019.

[10] W.-l. Huang i J. Peleska, "Complete model-based equivalence class testing for nondeterministic systems," *Formal Aspects of Computing*, tom 29, br. 2, pp. 335-364, 2017.

[11] F. Hübner, W.-l. Huang i J. Peleska, "Experimental evaluation of a novel equivalence class partition testing strategy," *Software & Systems Modeling*, tom 18, br. 1, pp. 423-443, 2019.

[12] P.-L. Poon, T. H. Tse, S.-F. Tang i F.-C. Kuo, "Contributions of tester experience and a checklist guideline to the identification of categories and choices for software testing," *Software Quality Journal*, tom 19, pp. 141-163, 2011.

[13] K. Juhnke, M. Tichy i F. Houdek, "Challenges concerning test case specifications in automotive software testing: Assessment of frequency and criticality," *Software Quality Journal*, tom 29, pp. 39-100, 2021.

[14] M. F. Kıraç, B. Aktemur, H. Sözer i C. Ş. Gebizli, "Automatically learning usage behavior and generating event sequences for black-box testing of reactive systems," *Software Quality Journal*, tom 27, pp. 861-883, 2019.

[15] T. Guggulothu i S. A. Moiz, "Code smell detection using multi-label classification approach," *Software Quality Journal*, tom 28, pp. 1063-1086, 2020.

[16] V. Moreno, G. Génova, E. Parra i A. Fraga, "Application of machine learning techniques to the flexible assessment and improvement of requirements quality," *Software Quality Journal*, tom 28, pp. 1645-1674, 2020.

[17] M. Z. Iqbal, A. Arcuri i L. Briand, "Environment modeling with UML/MARTE to support black-box system testing for real-time embedded systems: Methodology and industrial case studies," u *International Conference on Model Driven Engineering Languages and Systems*, Oslo, 2010.

[18] Y. Sheng, S. Jiang i C. Wei, "Constructing test suites for real-time embedded systems under input timing constraints," *IEEE Access*, tom 7, pp. 20920-20937, 2019.

[19] V. Garousi, M. Felderer, Ç. M. Karapiçak i U. Yilmaz, "Testing embedded software: A survey of the literature," *Information and Software Technology*, tom 104, pp. 14-45, 2018.

[20] L. Mariani, M. Pezze i D. Zuddas, "Recent advances in automatic black-box testing," *Advances in Computers*, tom 99, pp. 157-193, 2015.

[21] R. Venkatesh, S. Ulka, A. Zare i S. Agrawal, "On generating test cases from EDT specifications," u *International Conference on Evaluation of Novel Approaches to Software Engineering*, Barcelona, 2015.

[22] W. S. Jang i R. Y. C. Kim, "Automatic generation mechanism of cause-effect graph with informal requirement specification based on the Korean language," *Applied Sciences*, tom 11, br. 24, 2021.

[23] C. Burnay, S. Bouraga, J. Gillan i I. J. Jureta, "What lies behind requirements? A quality assessment of statement grounds in requirements elicitation," *Software Quality Journal*, tom 28, pp. 1615-1643, 2020.

[24] S. Agrawal, R. Venkatesh, U. Shrotri, A. Zare i S. Verma, "Scaling test case generation for expressive decision tables," u *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, Porto, 2020.

[25] A. Arcuri, M. Z. Iqbal i L. Briand, "Black-box system testing of real-time embedded systems using random and search-based testing," u *IFIP International Conference on Testing Software and Systems*, Natal, 2010.

[26] B. Vogel-Heuser, V. Karaseva, J. Folmer i I. Kirchen, "Operator knowledge inclusion in data-mining approaches for product quality assurance using cause-effect graphs," *International Federation of Automatic Control (IFAC) PapersOnLine*, tom 50, br. 1, pp. 1358-1365, 2017.

[27] K.-C. Tai, A. Paradkar, H.-K. Su i M. A. Vouk, "Fault-based test generation for cause-effect graphs," u *CASCON '93: Proceedings of the 1993 Conference of the Centre for Advanced Studies on Collaborative Research*, Toronto, 1993.

[28] E. N. Narciso, M. E. Delamaro i F. D. L. D. S. Nunes, "Test case selection: A systematic literature review," *International Journal of Software Engineering and Knowledge Engineering*, tom 24, br. 4, pp. 653-676, 2014.

[29] P. R. Srivastava, P. Patel i S. Chatrola, "Cause effect graph to decision table generation," *SIGSOFT Software Engineering Notes*, tom 34, br. 2, 2009.

[30] A. Paradkar, K. C. Tai i M. A. Vouk, "Specification-based testing using cause-effect graphs," *Annals of Software Engineering*, tom 4, br. 1, pp. 133-157, 1997.

[31] T. Ayav i F. Belli, “Boolean differentiation for formalizing Myers' cause-effect graph testing technique,” u *2015 IEEE International Conference on Software Quality, Reliability and Security - Companion*, Vancouver, 2015.

[32] M. Agrawal i U. Badhera, “Approach for minimization of test cases from decision table generated from cause effect graph,” *International Journal of Computer Applications*, tom 172, br. 7, pp. 7-10, 2017.

[33] S. Weißleder i D. Sokenou, “Cause-effect graphs for test models based on UML and OCL,” *SoftwareTechnik-Trends*, tom 28, 2008.

[34] H. S. Son, R. Y. C. Kim i Y. B. Park, “Test case generation from cause-effect graph based on model transformation,” u *2014 International Conference on Information Science & Applications (ICISA)*, Seoul, 2014.

[35] B. Bekiroglu, “A cause-effect graph software testing tool,” *European Journal of Computer Science and Information Technology*, tom 5, br. 4, pp. 11-24, 2017.

[36] K. Nursimulu, “Cause-effect validation of requirements for distributed systems,” University of Ottawa, Ottawa, 1994.

[37] D. Jagli, T. Mamatha, S. Mahalingam i N. Ojha, “The application of cause effect graph for the college placement process,” *International Journal of Software Engineering & Applications (IJSEA)*, tom 3, br. 6, pp. 77-85, 2012.

[38] N. Gavrilović i L. Lazić, “Knowledge assessment using cause-effect graphing methods,” u *The Seventh International Conference on eLearning (eLearning-2016)*, Belgrade, 2016.

[39] L. Dou i W.-D. Yang, “Design of test case for ATP speed monitoring function based on cause-effect graph,” u *2019 CAA Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, Xiamen, 2019.

[40] J. Lal i S. Singh, “From cause to effect: An empirical study of cause-effect graphing testing techniques and its test measurement: A review,” *International Journal of Computer Science and Technology*, tom 3, br. 3, pp. 89-92, 2012.

[41] I. Chung, “Modeling pairwise test generation from cause-effect graphs as a Boolean satisfiability problem,” *International Journal of Contents*, tom 10, br. 3, pp. 41-46, 2014.

[42] F. Huang i C. Smidts, “Causal mechanism graph - A new notation for capturing cause-effect knowledge in software dependability,” *Reliability Engineering & System Safety*, tom 158, pp. 196-212, 2017.

[43] D. K. Ufuktepe, T. Ayav i F. Belli, “Test input generation from cause-effect graphs,” *Software Quality Journal*, tom 29, pp. 733-782, 2021.

[44] “The BenderRBT cause-effect graphing user manual,” Bender RBT Inc., Queensbury, 2006.

[45] S. Singhal, N. Jatana, B. Suri, S. Misra i L. Fernandez-Sanz, “Systematic literature review on test case selection and prioritization: A tertiary study,” *Applied Sciences*, tom 11, br. 24, 2021.

[46] “BenderRBT (previously SoftTest/CaliberRBT),” BenderRBT Inc., [Na mreži]. Available: https://benderrbt.com/bendersoftware.htm#rbt. [Poslednji pristup 27 August 2022].

[47] D. K. Ufuktepe, “TOUCH: Test generator for cause effect graphs,” [Na mreži]. Available: https://github.com/denizkavzak/TOUCH. [Poslednji pristup 3 August 2022].