https://doi.org/10.37528/FTTE/9788673955056/POSTEL.2025.26

NOVI SOFTVER ZA DISKRETNU STOHASTIČKU SIMULACIJU

Marko Đogatović, Vesna Radonjić Đogatović, Nikola Matijašević
Univerzitet u Beogradu - Saobraćajni fakultet,
m.djogatovic@sf.bg.ac.rs, v.radonjic@sf.bg.ac.rs, nikola.matijasevic@sf.bg.ac.rs

Rezime: *Diskretna stohastička simulacija je metoda za modeliranje i analizu sistema kod kojih se promene stanja odigravaju u diskretnim vremenskim trenucima. Koristi se inženjerstvu, ekonomiji, saobraćaju i drugim oblastima za razumevanje složenih sistema, smanjenje troškova i rizika, donošenje upravljačkih odluka i analizu performansi. Grafički aplikativni softver mimSim Simulation Software (2024.11-2) je namenjen za izradu i simulaciju modela diskretnih stohastičkih sistema. Kao simulacionu strategiju softverski paket koristi raspoređivanje događaja uz efikasnu upotrebu generatorskih funkcija. Simulacioni model se realizuje prevlačenjem blokova kojih ukupno ima 45. Blokovi su podeljeni u tri grupe: blokovi modela, objekti i raspodele. Nakon izvršenja simulacije softver generiše izveštaj koji se može sačuvati u odgovarajućem formatu. Softverski paket je realizovan u programskom jeziku Python koji je istovremeno i interni skript jezik. Ovaj softver predstavlja odlično sredstvo za edukaciju i upoznavanje studenata sa različitim aspektima simulacije i već je našao primenu na osnovnim studijama Saobraćajnog fakulteta Univeerziteta u Beogradu.*

Ključne reči: *diskretna stohastička simulacija, softverski paket, edukacioni softver, Python*

1. Uvod

Simulacija je skup tehnika, metoda i alata za razvoj simulacionog modela realnog sistema i korišćenje tog modela u cilju opisivanja ponašanja sistema. Svrha simulacije je da razvije simulacioni model i sprovede eksperimente nad simulacionim modelom u cilju boljeg razumevanja realnog sistema [1]. Diskretna stohastička simulacija (*Discrete Event Simulation*, DES) je metodologija modeliranja dinamičkih sistema kod kojih se stanje menja u diskretnim, tačno određenim trenucima, tzv. događajima. Događaj predstavlja incident koji menja stanje celokupnog sistema. Obzirom da se promene stanja odigravaju samo kroz događaje, vremenski period između događaja je moguće preskočiti što čini ovu metodologiju izuzetno efikasnom [1, 2].

Diskretna stohastička simulacija je danas uspešno etabliran alat u inženjerstvu, ekonomiji, saobraćaju [3], zdravstvu [4] i drugim oblastima za razumevanje složenih sistema [5, 6], smanjenje troškova i rizika, donošenje upravljačkih odluka [7] i analizu performansi. Diskretnu stohastičku simulaciju najčešće koriste veliki proizvodni [5, 6] i

logistički sistemi, dok je njena upotreba ograničena u malim i srednjim preduzećima. Osnovni razlog za to su veliki troškovi licenci komercijalnih DES softvera, koji mogu da iznose i po više hiljada dolara (tabela 1), uz stalne godišnje troškove održavanja. Izrazito limitirajuće licence, kao i visoki troškovi obuke i cene rada na softveru su takođe značajni faktori ograničenog korišćenja komercijalnog DES softvera [8].

Tabela 1. Cene komercijalnih licenci simulacionih alata

Naziv simulacionog alata	**Cena**
Simul 8	$5499 - $8699
Simcad Pro	$2495 - $4950 (godišnja licenca)
ExtendSim	$4995
Arena Simulation Software	~$5000
AnyLogic	$12390 - $18990
Matlab (sa SimEvents-om)	€2345

Ova finansijska barijera proteže se i na obrazovne institucije, ometajući integraciju modeliranja i simulacije u akademske programe. Iako besplatni DES softveri i DES softveri otvorenog koda predstavljaju alternativu koja ima veliki potencijal za obrazovne institucije, oni ostaju nedovoljno iskorišćen resurs u ostalim sektorima.

Cilj rada je realizacija DES softvera koji bi mogao da se koristi u edukativne svrhe i koji bi bio jednostavan za korišćenje, a istovremeno u pogledu izrade simulacionih modela bio bi u što većoj meri u ravni sa postojećim komercijalnim DES softverom. U tu svrhu je realizovan simulacioni softver mimSim. Naravno, ovo nije jedini softver koji spada u domen besplatnog DES softvera i DES softvera otvorenog koda. Ima ih više i većina su programske biblioteke, od kojih se po svojoj kompletnosti i funkcionalnosti posebno izdvaja JaamSim.

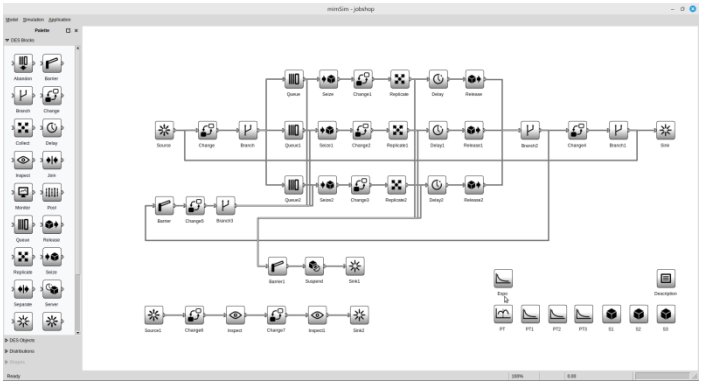
Rad je organizovan na sledeći način. Nakon uvodnog dela, u u drugom poglavlju dat je kratak opis softvera, njegove karakteristike i mogućnosti, da bi u trećem poglavlju bila prikazana arhitektura sofvera i kao i neke od komponenti koje je sačinjavaju. Primer koji ilustruje mogućnosti i daje ideju o njegovoj eventualnoj primeni je dat u poglavlju četiri. U poslednjem poglavlju su data zaključna razmatranja i navedeni su dalji pravci njegovog razvoja.

2. Opis softvera

Grafički aplikativni softver mimSim Simulation Software (2024.11-2) je namenjen za izradu i simulaciju modela diskretnih stohastičkih sistema. Interni skript jezik softvera je programski jezik Python, a softver je takođe razvijan u programskom jeziku Python [9]. Radi se o višeplatformskom (cross-platform) aplikativnom softveru koji je moguće na Linux i Windows operativnim sistemima.

Model se sastoji od blokova koji su međusobno povezani konekcijama. Konekcija se uspostavlja između izlaznog i ulaznog porta dva bloka. *Blok* obavlja izvesne operacije nad entitetima i objektima modela. *Entiteti* su dinamički objekti koji se u toku izvršenja simulacije stvaraju i uništavaju i kreću kroz blokove modela. *Statički objekti* (nazivaćemo ih samo objektima) stvaraju se na početku izvršenja simulacije i služe da ograniče kretanje entiteta kroz model, prikupljaju statistike, generišu vremena u skladu sa određenom raspodelom i izvršavaju određene operacije nad modelom. Objekti su grafički prikazani u vidu blokova modela samo bez ulaznog i izlaznog porta. Pored blokova,

entiteta i objekata u modelu postoji još jedan jedinstveni objekat koji služi za planiranje izvršenja simulacionog modela i nosi naziv *eksperiment*. Objekat eksperimenta nema grafičku reprezentaciju u modelu već mu se indirektno pristupa putem forme grafičkog interfejsa. Eksperimentom se postavljaju semena generatora slučajnih brojeva, broj simulacionih replikacija, vremenski period trajanja simulacije, prelazni period, inicijalizacioni i startni programski kod, kao i programski kod koji se izvršava po završetku simulacije. Po izvršenju simulacionog modela rezultati simulacije se prikazuju u izveštaju. Na slici 1 je dat prikaz grafičkog interfejsa sa učitanim modelom.



Slika 1. Prikaz glavnog prozora mimSim Simulation Software-a

Softver karakteriše, takozvani, drag & drop interfejs kojim se blokovi i objekti, koji se nalaze na paleti (slika 1) prevlače na list modela. Nakon prevlačenja blokove i objekte je moguće raspoređivati na listu modela po želji. Takođe, izborom ulaznog ili izlaznog porta bloka uz prevlačenje moguće je povezati dva različita bloka. List modela je beskonačne veličine i moguće ga je pomerati, uvećavati i umanjivati po želji. Blokove i objekte modela je moguće odabirati, menjati im svojstva, brisati i kopirati. Osim mogućnosti da imaju oba porta, blokovi mogu imati samo jedan ulazni port ili jedan izlazni port. Na ulazni port moguće je povezati neograničeni broj konekcija, dok broj mogućih konekcija na izlaznom portu zavisi od vrste bloka.

Realizovani softver poseduje većinu karakteristika zajedničkih za ostale DES softvere u pogledu modeliranja :

- fleksibilno kreiranje entiteta u vremenu i zavisnosti od tipa entiteta,
- kreiranje atributa i dodeljivanje vrednosti atributu u zavisnosti od entiteta,
- postavljanje entiteta u red čekanja,
- zadržavanje entiteta,
- zauzimanje i oslobađanje resursa procesa,
- promena parametara modela u zavisnosti od izvršenja simulacije, entiteta u modelu itd. i
- prilagođavanje logike upravljanja entitetima u modelu.

Gotovo da je nemoguće napraviti aplikativni simulacioni softver koji interno ne koristi neki programski jezik. Različiti softveri koriste različite jezike od kojih su neka rešenja jedinstvena za taj softver. Realizovani softver koristi Python iz nekoliko razloga. Radi se o modernom jeziku koji ima čistu i relativno jednostavnu sintaksu, popularan je i

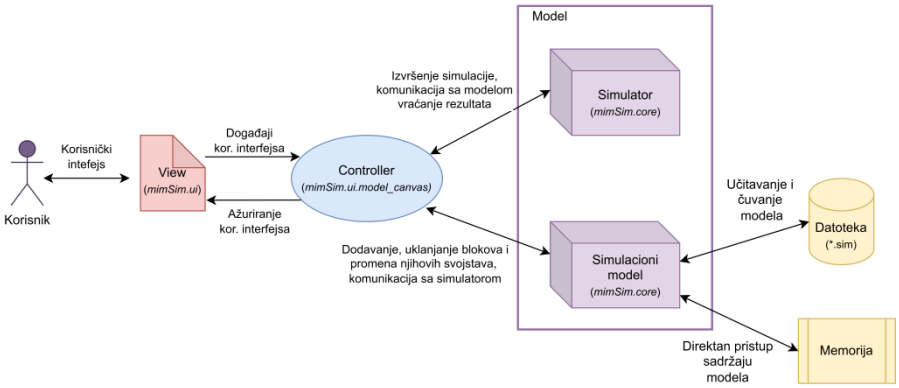
dobro dokumentovan sa mnoštvom biblioteka. Ekstenzija dokumenta modela je .sim. Radi se o komprimovanoj (zip algoritmom) tekstualnoj datoteci modela koja je data u vidu tabuliranog JSON formata. To znači da je moguće menjati sadržaj modela programski ili ručno.

Imena blokova i objekata moraju da prate pravila za imenovanje identifikatora Python jezika (promenljive, klase, ¸funkcije, itd.). Ime bloka mora da bude jedinstveno u modelu. Takođe, softver vodi računa da ime bloka bude jedinstveno i da odgovara pravilima za imenovanje identifikatora i neće dozvoliti unos imena bloka ukoliko ne zadovoljava prethodna dva uslova. Isto tako prilikom ubacivanja novog bloka u model, blok će biti ubačen sa jedinstvenim imenom. Blok je moguće preimenovati kasnije.

Izborom bloka u modelu otvara se forma svojstava objekta ili bloka. Radi se o formi kroz koju je moguće promeniti vrednosti nekim svojstvima objekta ili bloka. Forma je nemodalna što znači da je moguće istovremeno za više blokova otvoriti forme svojstava. Objektima ili blokovima je moguće pristupiti i putem Python koda korišćenjem promenljivih koje imaju isto ime kao i objekat ili blok u modelu. Neka svojstva je moguće menjati i pristupiti im, dok je nekim svojstvima samo moguće pristupiti, a nije ih moguće menjati.

3. Arhitektura softvera

Program koristi MVC (*Model-View-Controller*) softversku arhitekturu koja razdvaja korisnički interfejs od programske logike, tačnije simulatora i simulacionog modela [10]. View, grafički interfejs je realizovan u okviru `ui` paketa, dok je Model, simulator i simulacioni model realizovan u okviru `core` paketa programa. Controller, koji uspostavlja vezu između interfejsa i simulacionog modela je većim delom realizivan u `ui` paketu (u modulu `model_canvas`). U programskom jeziku Python paket je direktorijum koji sadrži jedan ili više modula (.py datoteka). Na slici 2 je prikazana MVC arhitektura mimSim softvera.



Slika 2. Model-View-Controller arhitektura mimSim softvera

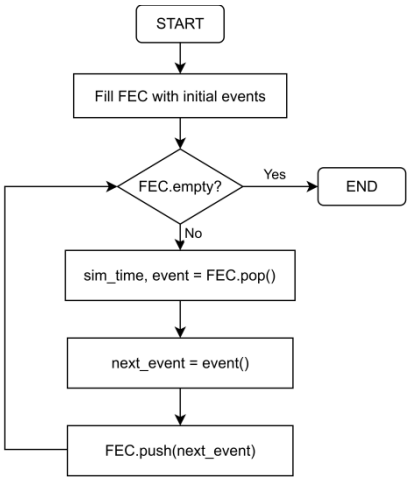
Objekti, blokovi i konekcije putem koji se uspostavljaju veze između blokova se iscrtavaju u objektu klase ModelCanvas (modul model_canvas). U tom objektu se nalaze

i reference na simulator i model. Svojstva blokova se unose i menjaju putem formi vezanih za svaki pojedinačni blok u modelu.

3.1. Simulator

U softveru se kao mehanizam pomaka vremena koristi *pomak vremena na naredni događaj*, dok se za simulacionu strategiju koristi *strategija raspoređivanja događaja*. Strategija predstavlja algoritam po kome se izvršava simulacija.

Kod strategije raspoređivanja događaja, događaji se smeštaju u listu budućih događaja (*Future Event Chain,* FEC) u kojoj se sortiraju u rastućem redosledu na osnovu vremenskog trenutka odigravanja događaja i prioriteta događaja (slika 3). Uzima se prvi događaj iz liste, vreme simulacionog časovnika se ažurira na vreme izvršenja događaja i događaj se izvršava. Izvršenjem događaja dobija se naredni događaj koji se raspodeljuje u FEC listu. Događaji u okviru strategije su realizovani korišćenjem generatorskih funkcija. Generatorska funkcija je poseban vid funkcije koja vraća objekat iteratora. Za razliku od obične funkcije koja koristi `return` službenu reč da vrati jednu vrednost i kompletira izvršenje funkcije, generatorska funkcija koristi službenu reč `yield` da vrati više vrednosti pauzirajući izršenje i čuvajući sopstveno stanje nakon svakog poziva funkcije [11]. Lista budućih događaja je realizovana korišćenjem *heap queue* strukture podataka koja je dvostruko povezana lista sa preraspodeljenim elementima unutar liste što omogućava brzi pristup najmanjem ili najvećem elementu liste. Važno je napomenuti da se u listu budućih događaja se smeštaju isključivo bezuslovni događaji [1, 2].



Slika 3. Blok dijagram strategije raspoređivanja događaja

Događaji u modelu mogu biti uslovni ili bezuslovni. Uslovni događaji čekaju da se ispuni određeni uslov da bi se izvršili (da se oslobodi resurs, da se otvori barijera, itd.), dok se bezuslovni događaji izvršavaju u vremenskom trenutku za koje je raspoređeno njihovo izvršenje (dolazak entiteta u model, završetak zadržavanja entiteta u modelu). Ovi događaji se odigravaju u blokovima modela i uvek se odigravaju nad tekućim

entitetom. Tekući entitet je entitet koji se trenutno nalazi u bloku i nad kojim se događaj odigrava.

Entiteti za koje nije moguće izvršiti uslovni događaj čekaju u redu čekanja. Red čekanja se nalazi unutar svakog bloka, ali se koristi samo kod blokova koji nemaju mogućnosti izvršenja uslovnog događaja (blok je blokiran). Situacije zbog kojih blok može postati blokiran su: nemogućnost zauzimanja dela resursa usled zauzetosti resursa u blokovima Server i Seize, nemogućnost daljeg kretanja entiteta ukoliko je barijera zatvorena u bloku Barrier, nemogućnost ulaska u red čekanja zato što je red čekanja ograničene dužine u bloku Queue itd. Uslovni događaji nad entitetom se odigravaju u sekvenci sve dok neki od blokova kroz koje se kreće entitet ne postane blokiran ili uđe u blok u kome se izvršava bezuslovni događaj. Odnosno, enititet se kreće kroz model sve dok su uslovi za izvršenje događaja zadovoljeni. Blok kod koga se završi prethodni uslovni događaj i stvoreni su uslovi da se izvrši novi događaj poziva prethodni blok da oslobodi entitet koji čeka na izvršenje uslovnog događaja. Kada je resurs u pitanju red čekanja je isti za sve blokove koji su povezani na jedan resurs. Inače, blok Queue uslovno propušta entitete i prikuplja statistike reda čekanja koji se inače nalazi ispred svakog bloka. Blok Queue može da menja veličinu i disciplinu čekanja u redu.

Blokovi u kojima se odigravaju bezuslovni događaji su blokovi Source, Delay i Server. U bloku Source se nakog pristizanja novog entiteta u model raspoređuje dolazak narednog entiteta korišćenjem vremena između dolazaka. Po ulasku u blok Delay za tekući entitet se raspoređuje događaj napuštanja bloka na osnovu vremena zadržavanja entiteta u bloku. Na sličan način kao blok Delay, blok Server po zauzimanju resursa raspoređuje događaj izlaska iz Servera, gde po izlasku entiteta oslobađa deo resursa koji je zauzeo entitet.

Ostali blokovi izvršavaju određene operacije nad entitetima, objektima ili stanjima simulacionog modela. Blok Sink uklanja entitet iz modela, blok Change menja atribute entiteta, blok Release oslobađa deo resursa, blok Branch usmerava entitet na odgovarajući blok, itd.

3.2. Simulacioni model

Simulacioni model sadrži sve blokove i objekte u modelu. Blokovi su osnovni gradivni elementi modela i kroz blokove se kreću dinamički objekti koje nazivamo entitetima. Nazivaju se dinamičkim objektima iz razloga što se stalno u toku izvršenja modela vrši njihovo stvaranje i uništavanje (uklanjanje iz modela) Entiteti sadrže samo soptvene atribute i nemaju informaciju u kom se bloku se nalaze.

Blok je zadužen da primi entitet, obavi odgovarajuću operaciju nad njim i prosledi entitet narednom bloku. Iz tog razloga blokovi sadrže reference na blok ili blokove iz kojih entitet može u njih da uđe kao i reference na blok ili blokove u koje entitet može da ode nakon izlaska iz bloka.

Kada entitet uđe u blok prvo se izvršavaju odgovarajuće operacije nad entitetom da bi on nakon toga bio smešten u red čekanja narednog bloka. Nakon toga se proverava zauzetost narednog bloka. Ukoliko blok nije zauzet oslobađa se entitet iz reda čekanja i izvršavaju se operacije nad entitetom u narednom bloku.

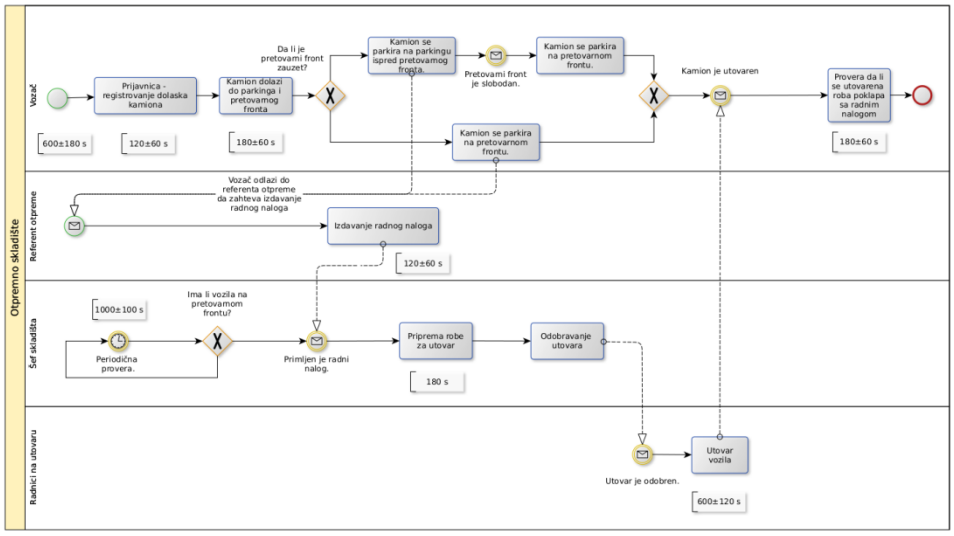
Ukoliko je blok zauzet entiteti koji pristižu ostaju u redu čekanja. Da bi entitet izašao iz reda čekanja blok u kome se on nalazi mora da pozove prethodni blok da

oslobodi entitet iz reda čekanja i prosledi ga na izvršavanje ogovarajućih operacija nad entitetom u bloku.

4. Ilustrativni primer

Kao ilustrativni primer relaizovan je simulacioni model otpremnog skladišta u kompaniji Coca-Cola HBC (CCHBC). Coca-Cola HBC ima automatsko postrojenje za punjenje bezalkoholnih pića, a svi proizvodi su uskladišteni u centralnom skladištu. Ovo skladište se koristi za transport robe ka kupcima i za razmenu robe između skladišta kompanije. Prodaja velikih količina zahteva savršenu sinhronizaciju između službi koje posluju u celoj organizaciji, a posebno saradnju i koordinaciju između službi u lancu snabdevanja [12].

Jedan od ključnih sektora u kompaniji je logistika, koja obuhvata distribuciju i skladištenje. Da bi se roba isporučila na vreme, potrebno je napraviti dobar plan po kojem će se roba za kupce pripremiti, utovariti i isporučiti. To nije jednostavan zadatak, s obzirom na uslove koji moraju biti ispunjeni: sa jedne strane je značajno vreme isporuke i stanje isporučene robe, a sa druge strane optimalno iskorišćenje kapaciteta kompanije i smanjenje troškova. Otpremna služba planira proces isporuke robe i diktira skladištu kojom brzinom će se vršiti utovar. Proces rada otpremnog skladišta je šematski prikazan na dijagramu (slika 4) korišćenjem BPMN (*Business Process Model and Notation*) notacije.

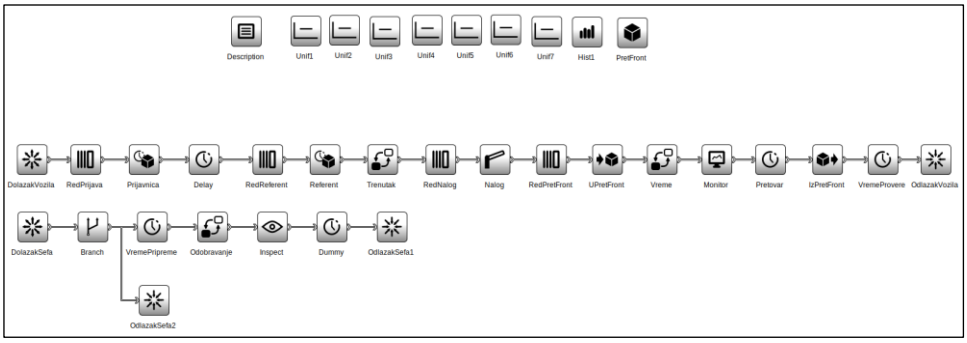


Slika 4.BPMN dijagram operacija otpremnog skladišta CCHBC

Na slici 5 je prikazan model otpremnog skladišta realizovan u mimSim softveru. Broj raspoloživih utovarnih mesta na pretovarnom frontu iznosi 2 pri čemu po jedan radnik radi na svakom utovarnom mestu.

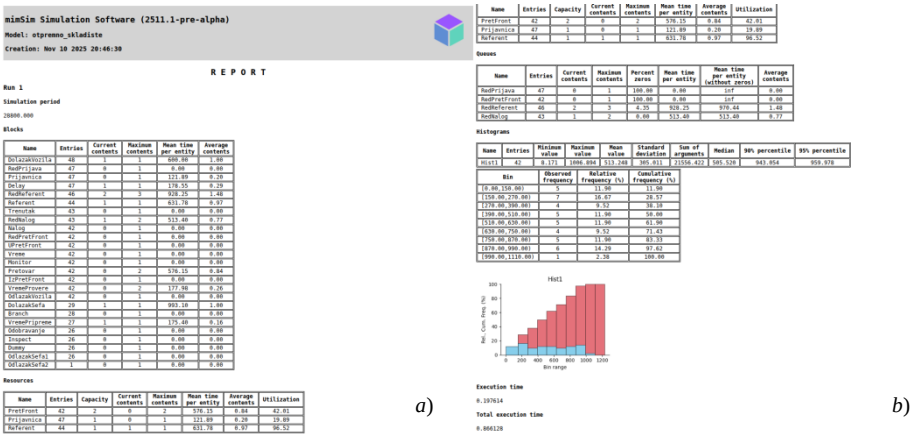
Model se uglavnom sastoji od redova čekanja (Queue), servera (Server) i blokova koji zauzimaju (Seize) ili oslobađaju (Release) resurs. Međutim, ono što je

sprecifično za ovaj model je korišćenje bloka Barrier (Nalog) koji blokira utovar kamiona dok se ne dobije dozvola od strane šefa magacina. Odobrenje se dobija otvaranjem bloka Barrier od strane bloka Inspect. Takođe, moguće je primetiti postojanje bloka Delay (Dummy) koji nema uneto vreme zadržavanja (vreme zadržavanja je 0 s). Kao što je već rečeno entitet se u modelu kreće sve dok ne postane blokiran ili dok se ne prebaci na FEC listu. Dummy blok ovde ima za cilja da prebaci entitet na FEC listu i da prekine njegovo kretanje što omogućava drugim entitetima da se kreću kroz model. Tačnije, na ovaj način entitetima koji se nalaze u redu ispred Naloga omogućava se da nastave dalje i pređu na utovar. Moglo bi se reći da Dummy server u ovom slučaju ima funkciju sličnu blok naredbi BUFFER u GPSS-u.



Slika 5. Simulacioni model otpremnog skladišta CocaCola HBC realizovan u mimSim sofveru

Na slici 6 prikazan izveštaj iz delova (slika 6a i slika 6b) koji je generisan nakon izvršenja simulacionog modela. Izveštaj prikazuje statistike blokova i objekata u modelu dobijene u toku izvršenja modela.



Slika 6. Izveštaj dobijen nakon izvršenja simulacionog modela (prikazan iz delova)

5. Zaključak

U radu je prikazan softver za diskretnu stohastičku simulaciju – mimSim Simulation Software. Motivacija za njegovu realizaciju je želja autora da se razvije softver koji bi bio konkurentan sa postojećim komercijalnim DES sofverima i koji bi mogao da se koristi kao edukativni sofver. Iako već postoje neki besplatni DES softveri i DES sofveri otvorenog koda oni često imaju specifičnu logiku izrade simulacionog modela koja pomalo odudara od standardne (npr. integrišu animaciju sa izradom modela) ili uvode neke ograničene, nedovoljno dokumentovane interne jezike što ograničava njihovu praktičnu upotrebu. U radu je dat kratak opis sofvera, objašnjena je njegova arhitektura i pojedine komponente i dat je primer koji ilustruje deo njegove funkcionalnosti.

Ova verzija mimSim sofvera predstavlja prvi korak u njegovom približavanju postojećim komercijalnim alatima. Iako je u pogledu upotrebljivosti interfejsa za kreiranje modela, blokovima i izveštajima mimSim veoma blizak nekim komercijalnim alatima, on ipak ima izvesne nedostatke koje bi trebalo u kasnijim verzijama nadomestiti. To se pre svega odnosi na mogućnosti animacije, dibagovanja i optimizacije modela, kao i na analizu ulaznih podataka (fitovanje raspodela prema podaacima dobijenih merenjem). Neke od ovih mogućnosti je relativno jednostavno uvesti (poput optimizacije i analize ulaznih podataka), dok animacija (na kojoj se već radi) i dibagovanje zahtevaju velike promene u kodu sofvera.

Na kraju, potrebno je istaći da se realizvani sofver uspešno koristi u nastavi na osnovnim akademskim studijama Saobraćajnog fakulteta na predmetu Računarska simulacija.

Literatura

[1] B. Radenković, M. Stanojević, A. Marković, *Računarska simulacija*. Beograd: Univerzitet u Beogradu - Saobraćajni fakultet i Fakultet organizacionih nauka, 1999.

[2] B. K. Choi, D. Kang, *Modeling and Simulation of Discrete Event Systems*. John Wiley & Sons, 2013.

[3] Holmes M. and Arseneau L, "A Discrete Event Simulation Tool for Conducting a Fleet Mix Study," *in Proceedings of the 14th International Conference on Operations Research and Enterprise Systems - Volume 1 : ICORES*, 2025, pp. 207-214, ISBN 978-989-758-732-0, DOI: 10.5220/0013090100003893

[4] D. Vecillas Martin, C. Berruezo Fernández, A. M. Gento Municio, "Systematic Review of Discrete Event Simulation in Healthcare and Statistics Distributions", *Aplied Sciences*, vol. 15, no. 4, article no. 1861, 2025, DOI: doi.org/10.3390/app15041861

[5] S. Chen, B. Mulgrew, and P. M. Grant, "The Role of Digital Transformation in Manufacturing: Discrete Event Simulation to Reshape Industrial Landscapes", *Applied Sciences*, vol. 15, no. 11, 2025. DOI: 10.3390/app15116140

[6] D. Qiao, Y. Wang, "A review of the application of discrete event simulation in manufacturing", *Journal of Physics: Conference Series, vol. 1802,* article no. 022066, 2021, DOI: 10.1088/1742-6596/1802/2/022066

[7] B. C. Battissacco, W. Azzolini Júnior, C. H. dos Santos, A. L. Romano, "Discrete event simulation-based digital twin: an approach focused on tactical, strategic, and

operational decisions", *Journal of Simulation*, pp. 1-19, 2025, DOI: 10.1080/17477778.2025.2531045
[8] S. Lang,, T. Reggelin, M. Müller, A. Nahhas. "Open-source discrete-event simulation software for applications in production and logistics: An alternative to commercial tools", *Procedia Computer Science*, vol. 180, pp. 978-987, 2021, DOI: 10.1016/j.procs.2021.01.349
[9] Python Software Foundation, "Python 3.13 Documentation," Python.org. Available: https://docs.python.org/3.13/ (accessed: Nov. 11, 2025).
[10] D.-P. Pop, A. Altar, "Designing an MVC Model for Rapid Web Application Development", *Procedia Computer Science*, vol. 69, pp. 1172-1179, 2014, DOI: 10.1016/j.proeng.2014.03.106
[11] D. Mertz, "PEP 255 – Simple Generators," Python Enhancement Proposals, Python.org, 2001. Available: https://peps.python.org/pep-0255/ (accessed: Nov. 11, 2025).
[12] M. Đogatović, M. Stanojević, B. Radenković, "SIM-PA: An open source-based simulation language" in *Proc. 13th Balkan Conf. Operational Research (BALCOR 13)*, Belgrade, Zlatibor, Serbia, 2013, pp. 650–659, ISBN: 978-86-7680-285-2

Abstract: *Discrete stochastic simulation is a method for modeling and analyzing systems where state changes occur at discrete points in time. It is used in engineering, economics, transportation, and other fields to understand complex systems, reduce costs and risks, support managerial decision-making, and analyze performance. Graphical application software mimSim Simulation Software (2024.11-2) is a designed for building and simulating models of discrete event systems. The software package uses an event-scheduling strategy with the efficient use of generator functions. The simulation model is built by dragging and dropping from a total of 45 blocks. The blocks are divided into three groups: model blocks, objects, and distributions. After the simulation is executed, the software generates a report that can be saved in an appropriate format. The software package is implemented in the Python programming language, which also serves as its internal scripting language. This software represents an excellent tool for educating and introducing students to various aspects of simulation and has already been implemented in undergraduate studies at the University of Belgrade - Faculty of Transport and Traffic Engineering.*

Keywords: *Discrete Event Simulation, software package, educational software, Python*

A NEW SOFTWARE FOR DISCRETE EVENT SIMULATION

Marko Đogatović, Vesna Radonjić Đogatović, Nikola Matijašević