Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/05BE30Horvat

RAZVOJ I UPOREDNA ANALIZA PROGRAMSKIH JEZIKA ZA PAMETNE UGOVORE NA BLOKČEJNU

THE DEVELOPMENT AND COMPARATIVE ANALYSIS OF PROGRAMMING LANGUAGES FOR BLOCKCHAIN-BASED SMART CONTRACTS

Nebojša Horvat, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisani su osnovni koncepti distribuiranih sistema, blokčejn tehnologije i programskih jezika za pisanje pametnih ugovora. Razvoj blokčejn tehnologije je podeljen na tri generacije od kojih je svaka donela nove jezike za pisanje pametnih ugovora. Jezici su zasebno analizirani i na kraju je izvršena njihova komparativna analiza u cilju određivanja poželjnih osobina programskih jezika za pisanje pametnih ugovora.*

Abstract – *In this work we present the fundamentals of distributed systems, blockchain technology and languages for writing smart contracts. The development of blockchain technology was divided into three generations out of which the each one has brought some new languages for writing smart contracts. Programming languages were then separately analyzed and then a comparative analysis was done in order to determine the desired properties of languages for writing blockchain smart contracts.*

Ključne reči: *distribuirani sistemi, blokčejn, pametni ugovori.*

1. UVOD

Krajem dvadesetog veka, relacione baze podataka postaju nezaobilazni činilac informacionih sistema. One se oslanjaju na sisteme za upravljanje bazama podataka i njihova evidencija transakcija omogućava analizu podataka. Baze podataka su na početku bile pretežno centralizovane. Takve baze lakše je konstruisati i održavati, ali imaju velika ograničenja sa aspekta bezbednosti zato što baza predstavlja jedinstvenu tačku otkaza. Takođe, centralizovane baze je veoma teško skalirati. Iz navedenih razloga, i usled porasta količine podataka koje je potrebno skladištiti, dolazi do razvoja distribuiranih baza podataka. Kod distribuiranih baza može se iskoristiti memorijski kapacitet i procesorska moć svih računara na kojima je baza distrbuirana. Oporavak od otkaza dela baze mnogo je lakši nego kod centralizovanih baza, gde otkaz baze često znači i gubitak podataka.

Distribuirane baze podataka rešavaju pomenute probleme u čuvanju podataka. Međutim, u poslovanju između dve stranke postoji dodatni skup problema koje je potrebno rešiti.

U većini slučajeva, transakcije prolaze niz koraka kako bi se utvrdila njihova validnost. Najveći problem je održavati organizaciju od poverenja koja bi bila zadužena za validaciju transakcija. Takva organizacija bila bi posrednik u poslovanju stranaka. Odličan primer takve organizacije bile bi banke koje su posrednici u finansijskim transakcijama. Loša strana pomenutih rešenja je to što postoji posrednik u koga obe strane moraju imati poverenja i on je odgovoran za pravilan rad čitavog sistema.

Rešenje svih prethodno navedenih problema se nalazi u upotrebi blokčejn (engl. *blockchain* - lanac blokova) tehnologije. Blokčejn sistemi omogućavaju da se poslovanje odvija bez posrednika uz očuvanje prednosti koje pružaju distribuirani sistemi. U zavisnosti od domena problema, moguće je koristiti privatne i javne blokčejn sisteme. Javni sistemi uvode transparentnost i mogućnost svakog člana mreže da validira transakciju i lokalno čuva spisak svih transakcija. Ovakvi sisteme pružaju sigurnost, otpornost na otkaze i povećanje brzine izvršavanja transakcija. Dok je javni blokčejn otvoren i omogućava svakome da učestvuje u radu sistema, privatni blokčejn omogućava samo ovlašćenim članovima da učestvuju u radu zatvorene mreže. Svaki član ima prava i ograničenja u mreži i ona se razlikuju od člana do člana. Privatni blokčejn sistemi ograničavaju pristup podacima, te su zato idealni za konzorcijum organizacija koje međusobno sarađuju.

Prva generacija blokčejn tehnologije, poput Bitcoina [1], bila je ograničena na to da podrži samo kriptovalute uz vrlo malo mogućnosti za pravljenje drugih sistema zasnovanih na njima. Naredna generacija tehnologije, pokrenuta Ethereumom [2] i Hyperledger Fabricom [3] je uvela programsku podršku za koncept pametnih ugovora. Uz pomoć pametnih ugovora možemo da pravimo proizvoljne aplikacije zasnovane na blokčejn sistemima.

Cilj ovog rada je da se predstavi trenutno stanje u razvoju blokčejn tehnologije sa posebnim fokusom na pametne ugovore i jezike u kojima se oni pišu. Za ispunjenje pomenutog cilja potrebno je realizovati nekoliko koraka. Najpre je potrebno izvršiti analizu teorijskih osnova blokčejn tehnologije. Pod tim se podrazumeva analiza koncepata i tehnologija koje su neophodne kako bi blokčejn sistemi funkcionisali. Takođe, važno je naglasiti i objasniti kako pomenute tehnologije u međusobnoj interakciji pružaju prednosti blokčejn sistema kao što su distribuiranost, sigurnost, transparentnost, itd. Nakon toga je potrebno analizirati jezike koje svaki sistem pruža za pisanje pametnih ugovora i izvršiti uporednu analizu pronađenih jezika.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dušan Gajić, docent.

2. DISTRIBUIRANI SISTEMI

Distribuirani sistem je skup autonomnih računara koji čine celinu i iz perspektive korisnika izgledaju kao jedan koherentan sistem [4]. Prethodni iskaz obuhvata dve glavne osobine distribuiranih sistema. Prva ističe da je distribuirani sistem skup računarskih elemenata koji su nezavisni jedni od drugih; takvi elementi se najčešće nazivaju čvorovi (engl. node - čvor). Druga osobina obuhvata korisnike koji mogu biti ljudi ili aplikacije. Korisnici pri interakciji sa sistemom moraju imati osećaj kao da komuniciraju sa jedinstvenom aplikacijom, a ne sa skupom čvorova. Ovo znači da čvorovi moraju intenzivno da komuniciraju i sarađuju. Način ostvarivanja takve komunikacije među čvorovima je srž distribuiranog sistema.

Kako korisnik može da komunicira isključivo sa celim sistemom, on ne zna od kojih elemenata tj. čvorova se sistem sastoji. Podaci su takođe raspoređeni po različitim čvorovima. To znači da korisniku koji se obraća sistemu mesto čuvanja podataka nije važno, te je sistem dužan samo da obezbedi podatke. Ovo dovodi do česte replikacije podataka kako bi se pružile bolje performanse i to nazivamo transparentnošću distribuiranosti (engl. distribution transparency).

3. TEHNOLOGIJA DISTRIBUIRANE GLAVNE KNJIGE I BLOKČEJN

Distribuirana glavna knjiga ili tehnologija distribuirane glavne knjige (engl. distributed ledger technology - DLT) je vrsta distribuirane baze podataka koja pretpostavlja moguće prisustvo malicioznih korisnika (čvorova). DLT je, takođe, vrsta strukture podataka za pamćenje transakcija koje su razmeštene na više čvorova (fizičkih mašina). Ključno je da su čvorovi povezani u peer-to-peer mrežu gde svaki čvor sadrži identičnu repliku glavne knjige. Takođe, svaki čvor samostalo osvežava svoje podatke, tj. kopiju glavne knjige. Za razliku od centralizovane glavne knjige koja obično ima regulatorno telo, kod distribuirane glavne knjige svaki čvor može da upiše novu transakciju i zatim ostali čvorovi glasaju kroz konsenzus algoritam o tome koja kopija glavne knjige je validna. U trenutku kada je konsenzus postignut, svi čvorovi upisuju validnu verziju glavne knjige.

Blokčejn je distribuirana struktura podataka koja implementira distribuiranu glavnu knjigu, koja je sastavljena od lanca kriptografski povezanih blokova koji sadrže grupe transakcija. U opštem slučaju, vrši se emitovanje (engl. broadcast) svih podataka svim učesnicima u mreži. Glavna razlika između blokčejna i drugih distribuiranih baza podataka je u tome što je blokčejn projektovan kako bi se postigao konzistentan i pouzdan dogovor o zapisu događaja (npr. „ko je vlasnik čega") između nezavisnih učesnika koji mogu imati različite motivacije i ciljeve.

3.1. Generacije blokčejn tehnologije

Ukoliko posmatramo razvoj blokčejn sistema od njihovih zvaničnih početaka 2008. godine kada je Satoshi Nakamoto objavio rad o Bitcoinu, mogli bismo uvideti tri faze u razvoju ovih sistema. Prva faza bi svakako bila dominacija bitcoina i drugih kriptovaluta. Kriptovalute su bile do te mere dominantne da ljudi nisu ni znali da blokčejn može da se koristi u druge svrhe. Druga generacija ili faza razvoja bila bi pojava Ethereuma, koji koristi blokčejn tehnologiju, ne samo u svrhu pravljenja kriptovalute, već kao svetski kompjuter (engl. world computer) na kome mogu na bezbedan način da se izvršavaju proizvoljni programi (pametni ugovori). Treća generacija bili bi privatni blokčejn sistemi sa kontrolom pristupa, koji se koriste u poslovanju različitih organizacija. Primer bi bili Hyperledger Fabric i Tendermint koji pružaju platformu za razvoj poslovnih rešenja baziranih na blokčejn tehnologiji.

Kao što je već naglašeno, u **prvoj generaciji** blokčejna kriptovalute su bile dominantne. Bitcoin se posebno ističe među njima kao prva i najpopularnija kriptovaluta. Kriptovaluta je sistem za digitalni transfer vrednosti. Bitcoin se koristi za trgovinu i akumulaciju vrednosti. Korisnici mreže međusobno komuniciraju kroz Bitcoin protokol. Komunikacija se pretežno odvija preko interneta. Bitcoin protokol i sve usko povezane aplikacije su dostupne kao softver otvorenog koda i mogu se pokrenuti na velikom broju različitih računara uključujući i pametne telefone koji su danas široko pristupačni.

Ono što je Ethereum, kao predstavnik **druge generacije**, doneo kao radikalnu promenu jeste pružanje blokčejn platforme sa ugrađenim Tjuring kompletnim programskim jezikom koji može biti iskorišćen za pisanje pametnih ugovora. Takvi ugovori mogu da podrže proizvoljne transakcije za prenos stanja ili omogućavaju korisnicima da naprave proizvoljan sistem koji koristi blokčejn platformu. Proizvoljni sistemi se mogu realizovati kroz prosto pisanje pametnih ugovora u nekoliko linija koda.

Pomenute prednosti Ethereuma omogućavaju brz razvoj aplikacija, sigurnost za male aplikacije koje nemaju dovoljno resursa da brinu o bezbednosti i mogućnost da male aplikacije vrlo efikasno komuniciraju. Takve funkcionalnosti se postižu kroz kreiranje apstraktnog korenskog sloja na kome se sve aplikacije izvršavaju kroz pametne ugovore napisane u Tjuring kompletnom jeziku. Ovo omogućava svima da na lak i brz način napišu pametne ugovore koji predstavljaju decentralizovanu aplikaciju koja ima svoja proizvoljna pravila vlasništva.

U **treću generaciju** ubrajamo privatne blokčejn sisteme sa kontrolom pristupa poput Hyperledger Fabrica, Hyperledger Sawtooth i Tenderminta koji pružaju platformu za razvoj poslovnih aplikacija. Hyperledger Fabric je blokčejn sistem sa kontrolom pristupa napravljen da zadovolji potrebe poslovnih aplikacija [5]. Otvorenog je koda i ima izuzetno modularnu i konfigurabilnu arhitekturu koja može da se prilagodi potrebama velikog broja poslovnih slučajeva korišćenja. Jedna od najvećih prednosti Fabrica je to što omogućava pisanje pametnih ugovora u programskim jezicima opšte namene kao što su Go, Java i JavaScript. Značajan aspekt Fabric mreže je da je identitet korisnika u mreži poznat i da dva učesnika ne moraju da veruju jedan drugom da bi uspešno poslovali. Dovoljno je da se njihovo poslovanje formalno definiše i mreža će voditi računa o tome da su svi aspekti dogovora ispoštovani.

4. RAZVOJ PROGRAMSKIH JEZIKA ZA PISANJE PAMETNIH UGOVORA

Pored arhitekture samog sistema, koja nije previše podložna promeni, pametni ugovori predstavljaju najvažniji element svakog blokčejn sistema za programere koji ga razvijaju. Kroz pametne ugovore se realizuje sva poslovna logika sistema i zato je važno da oni budu napisani na što kvalitetniji način, kako bi što bolje oponašali poslovne procese. Međutim, kako je sva logika sistema koncentrisana u pametnim ugovorima, njihovo pisanje predstavlja proces koji zahteva najviše resursa. Pametni ugovori, na primer, mogu biti zaduženi za prebacivanje velike količine novca, tako da bilo kakva greška u njihovom kodu može dovesti do velikih i neprihvatljivih gubitaka.

Kada govorimo o **I generaciji** blokčejn sistema, najčešće podrazumevamo Bitcoin tehnologiju, kao prvu i dominantnu kriptovalutu. Bitcoin koristi Script jezik za pisanje pametnih ugovora. Script je sličan Forth programskom jeziku, koristi obrnutu poljsku notaciju i baziran je na steku. Kada kažemo da je jezik baziran na steku, to u ovom slučaju znači da ne postoje registri u kojima se čuvaju operandi ili upisuju rezultati operacija, već se sve smešta na stek. Kod obrnute poljske notacije, operandi prethode operatoru, što uklanja potrebu za zagradama.

Script je veoma jednostavan programski jezik koji je dizajniran tako da bude ograničen i lak za pokretanje na različitim hardverskim platformama čak i na ugrađenim računarskim sistemima. Potrebna je minimalna količina procesne moći da bi Script radio, ali mnoge stvari koje programski jezici opšte namene podržavaju, u njemu ne mogu biti napisane. Sva ograničenja koja nosi sa sobom su namerno postavljena od strane njegovih kreatora i služe radi povećanja bezbednosti sistema.

Glavna novina koju **II generacija** donosi sa sobom je mogućnost pisanja pametnih ugovora u Tjuring kompletnom jeziku koji nema toliko izražena ograničenja kao što je slučaj kod Bitcoinovog Script jezika. Ono što je omogućilo Ethereumu da koristi Tjuring kompletan jezik jeste uvođenje virtualizacije, tj. Ethereum virtuelne mašine (engl. Ethereum Virtual Machine - EMV) i uvođenje pojma gasa. Svaka instrukcija izvršena u pametnom ugovoru troši gas i tako sprečava napade koji onemogućavaju rad sistema (engl. Denial of Service - DoS). Ethereum programeri mogu da koriste programske jezike višeg nivoa kao što su Solidity, Serpent ili LLL. Serpent je jezik niskog nivoa koji je baziran na Pythonu, LLL je takođe jezik niskog nivoa baziran na Lispu i oba se retko koriste. Solidity je objektno orijentisan jezik visokog nivoa koji je specifično dizajniran za pisanje pametnih ugovora. Dizajniran je po ugledu na JavaScript, C++, Python i maksimalno je optimizovan da dobro radi na Ethereum virtuelnoj mašini. Solidity ima strogo definisane tipove, podržava nasleđivanje i omogućava upotrebu biblioteka i pravljenje proizvoljnih korisničkih tipova podataka.

U **III generaciji** uvodi se mogućnost pisanja pametnih ugovora kroz upotrebu jezika opšte namene. Ovo je značajna prednost, pre svega jer programeri koji razvijaju pametne ugovore ne moraju više da uče novi DSL (engl. Domain Specific Language - jezici specifični za domen) kao što je slučaj kod Bitcoina ili Ethereuma. Fabric trenutno omogućava pisanje pametnih ugovora u Golangu, JavaScriptu i Javi. Sva tri navedena jezika su izuzetno popularna tako da postoji velika verovatnoća da će programer biti ekspert u jednom od navedenih jezika i tako vrlo brzo ovladati tehnikom za pisanje kvalitetnih pametnih ugovora.

5. UPOREDNA ANALIZA PROGRAMSKIH JEZIKA ZA PISANJE PAMETNIH UGOVORA

Blokčejn sistemi su sve zastupljeniji te se javlja potreba za pisanjem velikog broja pametnih ugovora. Problem kod blokčejn programiranja je to što se ugovori moraju veoma pažljivo pisati. I najmanja greška može da dovede do pada celog sistema. Danas možemo reći da smo došli do treće generacije ovih kompleksnih sistema, tako da ćemo se osvrnuti na to kako je izgledao razvoj njihovih jezika i šta je svaka nova generacija donela kao poboljšanje.

5.1. Tjuring kompletnost

Od svih jezika koje smo obradili do sada, jedino Script nije bio Tjuring kompletan; on podržava grananje ali ne podržava „goto" naredbu tako da se u njemu ne mogu napraviti petlje. Ovo je ozbiljno ograničenje svakog programskog jezika. Script je s namerom dizajniran sa značajnim ograničenjima kako bi se povećala sigurnost pametnih ugovora. Bitcoin nema napredne sigurnosne mehanizme kao sledeće generacije, tako da je njegov sigurnosni mehanizam to što je jezik za pisanje pametnih ugovora ekstremno ograničen, te je jako teško napraviti maliciozan kod. Svi drugi programski jezici, počevši od Solidity-a i Serpenta, pa do Golanga, Jave i JavaScripta su Tjuring kompletni.

5.2. Ekspresivnost

Ekspresivna moć programskog jezika predstavlja raznovrsnost i širinu ideja koje se mogu predstaviti i napisati u tom jeziku. Krenućemo od Scripta, kako je on prvi programski jezik za pisanje pametnih ugovora i svakako ima najmanju ekspresivnu moć. Mnoge ideje se u njemu ne mogu predstaviti. Svaki sistem koji u sebi sadrži koncept ponavljanja proizvoljni broj puta, nije moguće konstruisati u Scriptu.

Solidity je objektno orijentisan, Tjuring kompletan jezik koji sa sobom donosi mnogo više funkcionalnosti u poređenju sa Scriptom. U njemu je moguće opisati eksponencijalno više sistema nego u Scriptu; međutim, i on sa sobom nosi određene nedostatke. Za razliku od drugih modernih programskih jezika, Solidity je napravljen tako da je iz njega izbačen nedeterminizam. Determinizam povećava sigurnost koda i smanjuje mogućnost za grešku, ali smanjuje ekspresivnost jezika.

Iako je Go najnoviji programski jezik koji se može koristit za pisanje pametnih ugovora na Fabricu, čini se da je on najmanje ekspresivan. Činjenica da su iz Go-a izbačeni koncepti poput nasleđivanja i generalizacije, u značajnoj meri ograničava njegovu ekspresivnost.

Java i JavaScript su najstariji i najzreliji jezici od svih spomenutih. Njihove mogućnosti su izuzetne, objektno su orijentisani, podržavaju funkcionalno programiranje, konkurentnost i sve druge koncepte koji se očekuju od programskog jezika opšte namene. Tako da su oni najekspresivniji jezici za pisanje pametnih ugovora.

5.3. Performanse

Kada se porede performanse različitih programskih jezika za pisanje pametnih ugovora koji se koriste na različitim blokčejn sistemima, vrlo je tešto izolovati performanse samog sistema od performansi programskog jezika. Takođe je vrlo važno naglasiti da se kod programskih jezika na blokčejn sistemima mnogo više ceni da količina podataka koja treba da se upiše u glavnu knjigu bude mala, nego da se podaci brzo upišu. Oprez pri skladištenju podataka je pogotovo izražen kod javnih blokčejn sistema gde se svi podaci čuvaju u jednog velikoj glavnoj knjizi čija veličina može značajno da poraste vremenom i tako optereti svaki čvor u mreži.

Script je izuzetno jednostavan jezik i vrlo je niskog nivoa. On pruža dobre performanse pri pokretanju pametnih ugovora. S druge strane, važno ograničenje Scripta je to što on čini komplikovanije pametne ugovore napisane u njemu izuzetno glomaznim. Ovo je veliki problem s obzirom da Bitcoin ima ograničenje veličine njegovih skripti.

Solidity je jezik visokog nivoa koji je dodatno interpretiran na Ethereum virtuelnoj mašini. Kod jezika koji se kompajliraju na EVM, najbitniji podatak je koliko gasa troše pri izvršavanju pametnih ugovora. Ethereum mnogo naplaćuje ugovorima koji su spori i zahtevaju mnogo računanja, tako da je značajno da programski jezik pruža maksimalne performanse. Kako je Solidity programski jezik najvišeg nivoa dostupan za pisanje pametnih ugovora na Ethereumu, to ga čini najlakšim za pisanje i najpopularnijim, ali ujedno i programskim jezikom sa najlošijim performansama na Ethereum mreži.

Što se tiče programskih jezika za pisanje pametnih ugovora na Hyperledger Fabricu, kod njih prostor nije veliki problem, s obzirom na to da je privatni blokčejn uglavnom mnogo manje opterećen. Golang bi bio prirodan izbor u potrazi za jezikom koji pruža najbolje performanse. On se kompajlira i njegov kod se direktno izvršava, za razliku od Jave za čije izvršavanje je potrebna JVM ili JavaScripta, koji je interpetirani jezik i stoga mnogo sporiji.

5.4. Bezbednost

Bezbednost je apsolutno najbitniji aspekt pri pisanju pametnih ugovora, tako da bi trebalo da je prioritet pri izboru programskog jezika.

Kao što je već rečeno u ranijem tekstu, jedina sigurnost koju Bitcoin pruža pri izvršavanju svojih skripti jeste ograničenost njegovog jezika. To čini Script ubedljivo najbezbednijim jezikom za korišćenje.

Solidity je programski jezik koji je napravljen sa mnogim ograničenjima. Razlozi za namerno kreiranje svih opisanih ograničenja leže baš u potrebi za povećanjem bezbednosti. Ukoliko uklonimo nedeterminizan, povećavamo bezbednost sistema i prostor za pojavu slučajnih grešaka.

Od spomenutih jezika opšte namene, Go je svakako najbezbedniji. Ograničenja koja su uvedena u Go su pretežno tu da bi povećala čitljivost i čistoću koda. Bezbednost Go-a i dalje nije na nivou koji pruža Solidity ili pak Script, ali je on znatno bezbedniji od Jave, a pogotovo od JavaScripta koji nema ni strogo definisane tipove.

6. ZAKLJUČAK

U prethodnim segmentima rada je opisana tehnologija koja se nalazi iza blokčejn sistema, kao i njihov razvoj koji je podeljen u tri generacije. Važno je primetiti da su kroz generacije sistemi dizajnirani s različitim idejama o njihovoj upotrebi. Bitcoin je bio zamišljen kao kriptovaluta i siguran način za razmenu novca bez potrebe za regulatornim telom. Ethereum je prepoznao mogućnosti i prednosti koje blokčejn tehnologija donosi sa sobom i težio da napravi svetski računar (engl. world computer). Ethereum je zamišljen kao platforma koja bi omogućila lak i brz razvoj blokčejn aplikacija koje bi se realizovale preko pametnih ugovora. Za razliku od Bitocina, jezik za pisanje pametnih ugovora kod Ethereuma je Tjuring kompletan i pruža mnogo veće mogućnosti od Scripta. Sledeća faza razvoja su privatni blokčejn sistemi sa kontrolom pristupa. Povećana popularnost blokčejn sistema je dovela do želje da se oni upotrebe u poslovnim aplikacijama koje imaju drugačije potrebe, koje se ne mogu zadovoljiti javnim blokčejn sistemima. Poslovne strane često zahtevaju postojanje identiteta i zabranu pristupa, tako da samo klijenti sa dozvolom mogu da učestvuju u poslovnom procesu.

Kroz razvoj blokčejn sistema, jasno se vidi da je stvar koja se najviše menjala zapravo jezik u kome su pametni ugovori pisani. Kako su pametni ugovori jedan on najbitnijih aspekata sistema i svakako deo sistema koji sadrži poslovnu logiku, važno je rukovati programskim jezikom u kome se ta logika može jednostavno i kvalitetno implementirati. Script je veoma jednostavan programski jezik u kome je bezbedno pisati pametne ugovore i lakše ih je formalno verifikovati ali je veoma neekspresivan, što ga čini veoma teškim za rad. Solidity je Tjuring kompletan i mnogo ekspresivniji od Scripta, ali i on sa sobom nosi mnoga ograničenja koja su napravljena radi njegove sigurnosti. Velika mana i Solidity-a i Scripta je to što su oni DSL-ovi. Golang, Java i JavaScirpt su prvi jezici opšte namene koji su korišćeni za pisanje pametnih ugovora. Rad sa popularnim jezicima je vrlo olakšavajući zato što su oni dobro poznati programerima i tako se brže može doći do kvalitetnih pametnih ugovora.

7. LITERATURA

[1] Nakamoto S., *Bitcoin: A Peer-to-Peer Electronic Cash System*, dostupno na: https://bitcoin.org/bitcoin.pdf poslednji pristup 20.08.2019.

[2] Vitalik Buterin, 2013. A Next-Generation Smart Contract and Decentralized Application Platform, dostupno na: https://github.com/ethereum/wiki/wiki/White-Paper, poslednji pristup 20.08.2019.

[3] Androulaki, E., Barger, A., Bortnikov, V., Cachin, 2018, April. Hyperledger fabric: a distributed operating system for permissioned blockchains

[4] Tanenbaum, A.S. and Van Steen, M., 2017. Distributed systems: principles and paradigms. Prentice-Hall., https://www.distributed-systems.net/index.php/books/distributed-systems-3rd-edition-2017/, poslednji pristup 20.08.2019.

[5] Hyperledger Fabric dokumentacija, https://hyperledg er-fabric.readthedocs.io/en/release-1.4, poslednji pristup 20.08.2019.

Kratka biografija:

Nebojša Horvat je rođen u Novom Sadu, 1995. god. Fakultet tehničkih nauka upisao je 2014. god. Osnovne akademske studije završio je 2018. god.