Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 007.5
DOI: https://doi.org/10.24867/03BE32Papuga

REALIZACIJA MAŠINE PRAVILA U RUKOVANJU ALARMIMA U NADZORNO-UPRAVLJAČKOM SISTEMU

AN IMPLEMENTATION OF ALARMS RULES ENGINE IN SCADA SYSTEM

Vladimir Papuga, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisan je razvoj kao i implementacija mašine pravila (en. Rules engine) za alarmiranje u SCADA sistemima. Opisana je implementacija rešenja koje se zasniva na upotrebi expression trees i ukratko su pomenuti SCADA sistemi i značaj alarmiranja u tim sistemima.*

Ključne reči: *SCADA sistemi, Mašina pravila, Alarmni sistemi, Expression Trees*

Abstract – *This paper describes the development and implementation of Rules Engine for alarming in SCADA systems. Implementation of the solution based on expression trees is described. SCADA systems and significance of alarm in these systems are briefly described.*

Ključne reči: *SCADA systems. Rules engine, Alarming systems, Expression Trees*

1. UVOD

Od pojave upravljačkih sistema, SCADA *(Supervisory Control and Data Acquisition)* je odigrala važnu ulogu u oblasti automatizacije. SCADA sistemi nude sredstvo za nadzor i upravljanje industrijskim procesom posredstvom udaljenih uređaja. Nadzor se realizuje prikupljanjem podataka, pri čemu se podaci dobijeni od uređaja, dalje obrađuju i prikazuju korisniku – operateru. U okruženju koje kontroliše operater, koristeći SCADA-u, neophodan je alarmni sistem. Alarmni sistem služi da obavesti operatera o abnormalnim procesnim uslovima i neispravnosti uređaja. Na primer, kada god izmerena procesna vrednost (signal) prekorači očekivane granice, to je abnormalnost koja se mora proveriti kako bi se sprečile moguće posledice. Alarm se mora generisati kada se uoči da se proces ne ponaša kako je predviđeno, pre nego što se desi neki kritičan događaj, ali samo onda kada je u pitanju kritičan događaj. Svako kritično stanje trebalo bi generisati jedan alarm koji bi precizno ukazivao na problem u sistemu i tako omogućio operateru pravilnu reakciju. Kompleksnost upravljanja alarmima raste uporedno sa kompleksnošću nadziranog sistema. Tu se prikuplja velika količina podataka koji mogu "preplaviti" operatera i zbog toga alarmni sistemi moraju biti dobro dizajnirani, implementirani i održavani.

Ovaj rad fokusira se na pojednostavljenju održavanja dela alarmnog sistema zaduženog za generisanje alarma, pogotovo kada se radi o čestim izmenama (dodavanje novih uslova alarma, menjanje logike algoritama, itd.).

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Erdeljan, red. prof.

Jedno od mogućih rešenja za opisivanje stanja alarma zasnivalo bi se na upotrebi većeg broja uslovnih iskaza *(if-else, switch)*, koji bi vremenom, sa proširivanjem sistema doveli do nepreglednog koda i težeg održavanja. Kako bi se izbegao ovaj problem osmišljena je mašina pravila zasnovana na *expression trees*, čiji je cilj da omogući jednostavnu modifikaciju alarmnih pravila, samim tim i jednostavno održavanje ovog dela sistema, a da se pri tome postignu zavidne performance.

2. SCADA ALARMNI SISTEM

SCADA je akviziciono-upravljački sistem, sa primarnim funkcijama objedinjavanje i obrada svih procesnih podataka, i komunikacije sa operaterima. Tipično obuhvata više računara: servera i radnih stanica, gde se serveri brinu o prikupljanju, obradi i skladištenju informacija o nadziranom sistemu, a radne stanice prikazuju te informacije operaterima i omogućavaju im blagovremene reakcije. Objedinjavanje procesnih podataka ostvaruje se komunikacijom sa mrežom instaliranih procesnih kontrolera.

Obrada prikupljenih podataka podrazumeva proveru ispravnosti i vrednosti merenja, uz eksplicitno izveštavanje operatera o otkrivenim neregularnostima [1]. Pored prikupljanja merenja, SCADA sistem može formirati izvedene (sračunate) vrednosti, pratiti ispravnost računarske i komunikacione opreme, pratiti aktivnosti korisnika i sl. Stoga je alarmni sistem naizostavna komponenta SCADA-e i svojim performansama bitno utiče na upotrebljivost sveukupnog rešenja.

Osnovni parametri kojima se mere *real-time* performanse SCADA softvera vezani su za ukupan broj procesnih veličina koje se mogu konfigurisati u SCADA bazi, i za broj promena koje se mogu obraditi i prikazati u nekom vremenskom periodu. Zato je dizajn SCADA softvera primarno orijentisan ka efikasnosti ukupnog sistema, ali i njegovoj adaptivnosti na raznovrsne aplikativne zahteve prisutne u realnim implementacijama.

Alarmni sistem je osnovni sistem za podršku operatera za rukovanje abnormalnim situacijama i ima dve funkcije:

1. Primarna funkcija alarmnog sistema jeste da upozori operatera o situaciji koja nije normalna. Ova funkcija pomaže operateru da kontroliše buduće ponašanje složenog procesa skrenuvši mu pažnju na neželjene procesne uslove. Sistem treba da obavesti operatera o uslovima procesa koji zahtevaju blagovremenu procenu i eventualno korektivne mere u cilju održavanja ciljeva procesa u smislu sigurnosti, produktivnosti, efikasnosti, itd. Svaki alarm treba da upozori, obavesti i usmerava operatera [2].

2. Sekundarna funkcija alarmnog sistema je da beleži događaje i omogući njihovu naknadnu analizu i optimizaciju postrojenja. Treba da bude fleksibilan i da sadrži događaje, potisnute alarme i druge delove informacija koje nisu predstavljene na glavnoj listi alarma, ali koje bi mogle biti korisne za *offline* istraživanje incidenata. [2] Cilj potisnutih alarma je da obezbedi da predstavljeni alarmi u bilo kom trenutku budu relevantni za rad operatera u trenutnim procesnim uslovima i da se izbegnu "poplave" alarma. Potiskivanjem alarma ne uklanjaju se informacije iz sistema, već se samo razdvajaju informacije o alarmu na jedan ili više detaljnih nivoa. Glavni alarmni prikaz treba da obezbedi praćenje i kontrolu budućeg ponašanja sistema privlačenjem pažnje operatera prema uslovima procesa koji zahtevaju procenu ili akciju. Trebalo bi da prikaže samo alarme koji su relevantni u trenutnim uslovima procesa.
Alarmi trebaju biti relevantni za ulogu operatera, da nagoveste potrebnu akciju, da se prikazuju po stopi koju operater može da isprati i da su lako razumljivi, kako bi sistem ostao upotrebljiv za operatera u svim situacijama.

2.1 Rukovanje alarmima

Pod rukovanjem alarmima *(en. Alarm management)* podrazumeva se skup praksa i procesa koji osiguravaju efikasan alarmni sistem. *ISA18.2* standard definiše procese i postupke potrebne za kreiranje efikasnog sistema rukovanja alarmima. Jedan od najčešće primenjivanih modela jeste životni ciklus upravljanja alarmima *(en. Alarm Management Lifecycle).* Ovaj model se može primeniti na novi ili postojeći alarmni sistem [3].

3. DOSADAŠNJA REŠENJA

Realizacija jednostavnog rešenje za generisanje alarma se može zasnovati na izvršavanju brojnih uslovnih iskaza gde se npr. izmerene vrednosti porede sa željenim granicama. Mana ovakvog rešenja nije uočljiva dok je sistem mali, ali sa porastom sistema i njegove kompleksnosti raste broj, kao i kompleksnost uslovnih iskaza potrebnih za njegovo opisivanje. Ukoliko je sistem podložan čestim modifikacijama, ovakvo rešenje će dovesti do otežanog, kao i vremenski zahtevnijeg održavanja.
Kako bi se izbegao gore opisan scenario, osmišljena je mašina pravila bazirana na *expression trees* čiji je cilj da omogući jednostavno održavanje sistema, razlaganjem složenih komponenti alarmiranja na manje delove. Na ovaj način, obrada alarma opisana preko mnoštva ugnježdenih petlji se pojednostavljuje i opisuje preko skupa jednostavnih, čitljivih pravila. Modifikacija sistema vrši se jednostavnim dodavanjem ili brisanjem pravila iz konfiguracionog fajla.
Mašina pravila je ekspertski sistem baziran na pravilima i sastoji se od skupa parova tipa *"uslov - akcija"*. Ukoliko je uslov pravila zadovoljen, izvršava se akcija vezana za taj uslov. Glavna prednost ovakvog pristupa jeste da se pomoću pravila može znatno pojednostaviti opisivanje rešenja veoma složenih problema.
Ekspertski sistem je program koji je projektovan da modelira sposobnosti rešavanja problema ljudskog eksperta u nekoj oblasti. Kod ekspertskih sistema naglasak nije na obradi podataka (algoritmom, programom) kao kod konvencionalnih rešenja, već na oblikovanju (baze) znanja.
U daljem tekstu biće detaljnije opisan ekspertski sistem i mašina pravila.

4. EKSPERTSKI SISTEMI

Ekspertski sistem je inteligentni računarski program koji koristi znanje i postupke zaključivanja u procesu rešavanja problema za koje je potreban visok stepen stručnosti i iskustva iz domena kome se ekspertski sistem obraća. Osnovu ekspertskih sistema karakterišu sledeća svojstva: mora da sadrže znanje (iz konkretne oblasti – domena), da vrše zaključivanje, prosuđivanje i odlučivanje na osnovu nepouzdanih i nepotpunih informacija i da omoguće tumačenje svog ponašanja. Ekspertski sistem čini vise delova: deo za izvršavanje problema (baza znanja, mehanizam zaključivanja i globalna baza podataka) i okruženja (korisnički interfejs).
Baza znanja (knowledge base) – baza činjenica i heuristika u području za koje je namenjen ekspertski sistem.
Mehanizam zaključivanja – softver sposoban da sredi informacije iz baze znanja i da na osnovu toga izvuče zaključke.
Globalna baza podataka – radna memorija za beleženje trenutnih statusa sistema, ulaznih podataka za određeni problem.
Komunikacioni interfejs – deo koji omogućava dijalog između donosioca odluke (korisnika) i sistema.

4.1 Predstavljanje znanja u ekspertskim sistemima

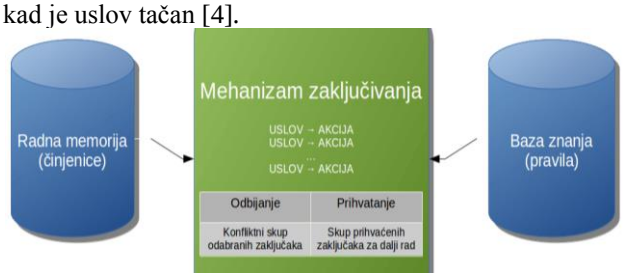
Osnovna namena reprezentacije znanja *(en. Knowledge representation)* jeste predstavljanje znanja (eksperta sistema) u simboličkom obliku. Ne postoji idealan tip predstavljanja znanja koji može pokriti i okarakterisati sve delove nekog ekspertskog sistema. U mnogim formalnim sistemima pod činjenicom, kao oblikom deklarativnog znanja, obično se podrazumeva iskaz – rečenica koja ima dve moguće vrednosti, tačna ili netačna. Osnovni modeli za predstavljanje znanja u ekspertskim sistemima su: *pravila, mreže zaključivanja, semantičke mreže, okviri, O-A-V trojke, objekti, iskazna logika i logika prvog reda.*

5. MAŠINA PRAVILA - PRODUKCIONI SISTEM PRAVILA

Mašina pravila (*en. Rules Engine)* je ekspertski sistem koji se sastoji od skupa parova tipa "uslov - akcija". Ovako definisana pravila nazivaju se produkciona pravila ili još pravila odlučivanja, pa se često u literaturi i praksi ovakvi sistemi nazivaju produkcioni ekspertski sistem ili *produkcioni sistem pravila (en. Production Rule System).* Osnovu sistema čine baza činjenica koje se smeštaju u radnu memoriju, baza znanja i procedura za izvršavanje produkcionih pravila i mehanizma zaključivanja. (slika 1).
Baza znanja - produkciona pravila u deklarativnom obliku *(uslov - akcija)* baziraju se na principima prediktivne logike, oblika *"AKO... TADA..." (IF... THEN...).* Svako produkciono pravilo sadrži uslov koji mora biti zadovoljen pre nego što se izvrši akcija.
Radna memorija (činjenice) - sadrži skup informacija koje se testiraju kako bi se zaključila istinitost nekog

uslova. Predstavlja činjenice iz globalne baze podataka koje se dovlače u radnu memoriju.

Mehanizam zaključivanja - jednostavna programska petlja koja testira istinitost uslova pravila i izvršava akcije kad je uslov tačan [4].



Slika 1: *Mašina pravila - osnove sistema*

5.1 Mehanizam zaključivanja

Osnovna struktura pravila u produkcionom sistemu pravila je prilično jednostavna. Svako pravilo ima svoj uslov i akciju. Uslov je *boolean* izraz dok akcija može biti bilo šta, ali i može biti ograničena u zavisnosti od konteksta produkcionog sistema pravila. Na primer, ukoliko sistem vrši samo validaciju, onda se akcije mogu odnositi samo na prijavljivanje grešaka. [5] Stoga je opravdana primena takvih mehanizama zaključivanja u alarmnom sistemu.

Mehanizam zaključivanja je osnovna komponenta u sistemu mašine pravila. Funkcioniše u nizu ciklusa zaključivanja. Sistem sa velikim brojem pravila i činjenica može dovesti do toga da su mnoga pravila istinita za istu činjeničnu tvrdnju. Za ova pravila se kaže da su u sukobu. Agenda upravlja redosledom izvršenja ovih konfliktnih pravila koristeći strategiju za rešavanje konflikata *(izbor pravila sa najvišim prioritetom, izbor prvog pravila, najskorije dodatog pravila, ...)*. Konflikt se rešava primenom jedne ili kombinacijom više strategija za rešavanje konflikata pravila. Nakon izvršavanja svakog pravila, ažurira se baza činjenica i mehanizam počinje novi ciklus pretraživanja skupa pravila. Sistem prestaje sa radom kada mehanizam zaključivanja ne nađe pravilo koje može izvršiti, tj. uslovi svih pravila su neistiniti.

Mehanizam zaključivanja zasnovan je na metodama *ulančavanju napred* i *ulančavanju nazad.*

5.2 Prednosti sistema sa mašinom pravila

Prednosti sistema sa mašinom pravila su mnogobrojne i najbolje se mogu opisati preko sledecih karakteristika ovih sistema:

Deklarativni pristup: Prednost ovakvog pristupa jeste da se pomoću pravila može znatno pojednostaviti opisivanje rešenja veoma složenih problema. Pravila su mnogo čitljivija od klasičnog koda.

Razdvajanje logike od podataka: Podaci su u domenu objekta, dok je logika u pravilima. Ovo u osnovi krši principe objektno-orijentisanog programiranja, ali omogućava lakše održavanje i proširivanje sistema u budućnosti.

Centralizacija znanja: Korišćenjem pravila kreira se baza znanja koja je izvršna. U idealnom slučaju, pravila su dovoljno precizno napisana da mogu poslužiti kao dokumentacija.

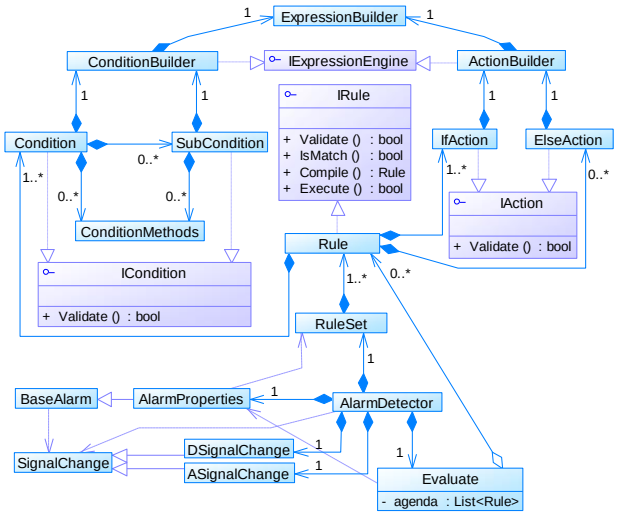
Integracija: Laka integracija u već postojeće sisteme, kao i velika *brzina* i *skalabirnost.*

Razumna pravila: Kreiranjem objektnih modela i opciono, *domen specifičnih jezika (en. Domain Specific Languages)* koji modeliraju domen problema može se omogućiti pisanje pravila koja su vrlo blizu prirodnom jeziku. Kao rezultat, opisana logika može biti razumna i ne tehničkim stručnjacima domena [6].

6. PREDLOŽENO REŠENJE

Svaka promena u nadziranom sistemu (npr. promena stanja prekidača, neko merenje, itd.) detektuje se u vidu promene signala koji predstavlja taj element. Da li će se alarm generisati i koji tip alarma će biti generisan zavisi od svojstava signala koja su promenjena, kao i od tipa samog signala. Na osnovu ovih podataka pozivaju se prethodno kreirana pravila definisana u konfiguracionom fajlu. Sva pravila čiji uslovi sadrže svojstva signala koja su promenjena evaluiraju se i u zavisnosti od rezultata evaluacije izvršavaju se odgovarajuće akcije koje mogu dovesti do generisanja alarma.

Na slici 2 prikazan je klasni dijagram koji predstavlja model na osnovu koga se implementira rešenje.



Slika 2: *Klasni dijagram modela rešenja*

Dato rešenje zasniva se na primeni *expression trees* u implementaciji mašine pravila. *Expression trees* predstavlja strukturu podataka koja sadrži izraze *(en. Expression)*, tj. strukturu stabla gde je svaki čvor izraz. Ova struktura je blisko povezana sa lambda izrazima i koristi se za njihovo opisivanje, tj. daje informaciju šta lambda izraz treba da uradi. Ova osobina *expression tree-a* omogućava dinamičnije i fleksibilinije pisanje pravila. Gubi se potreba za korišćenjem složenih uslovnih iskaza jer se pravila, tj. uslovi i akcije opisuju preko ove strukture podataka. Kako bi se dobio izvršni kod koji će izvršiti zadatak opisan u *expression tree-u*, *expression tree* je potrebno kompajlirati. Kompajliranjem dobija se delegat koji predstavlja opisan lambda izraz.

Rešenje se može podeliti na tri celine, kreiranje *expression tree-a*, kompjaliranje i izvršavanje pravila.

Kreiranje izraza: Pravila su opisana u konfigurabilnom fajlu *Rules.xml*. Klasa *RuleSet* parsira konfiguracioni fajl i kreira pravila *(en. Rule)*. Svako pravilo sadrži kolekciju uslova i dve kolekcije akcija, *IfAction* i *ElseAction*. Svaki tip alarma poseduje svojstva (*en. Property)* koja ga opisuju. Ove osobine se nalaze u klasi *AlarmProperties* i predstavljaju činjenice mašine pravila, tj. skup

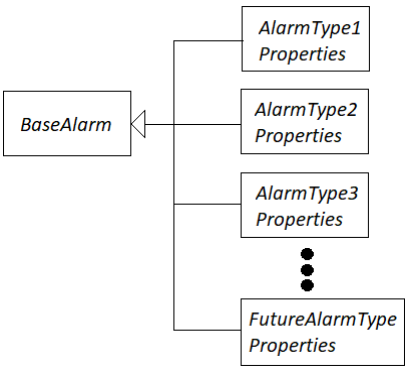
informacija koje se testiraju kako bi se zaključila istinitost nekog uslova. Ukoliko su svi uslovi u kolekciji zadovoljeni, smatra se da je uslov pravila zadovoljen i izvršavaju se akcije. Svako pravilo mora imati definisan *IfAction* - akcije koje se izvršavaju kada je uslov pravila zadovoljen. Svaki tip alarma generiše se na osnovu matematičkog izraza koji predstavlja jedan od uslova pravila, dok se drugi uslovi mogu odnositi na to da li je alarm prigušen, da li je dozvoljeno alarmiranje na konkretnoj tački, itd. Uslovi pravila kreću se od jednostavnih, gde se porede svojstva alarma sa nekom vrednošću, do kompleksnih, gde se koristi složen matematički izraz. Važna osobina *expression tree-a* je mogućnost spajanja više manjih *expression tree-a* u jedan veći. Imajući u vidu pomenutu osobinu, složeni matematički izraz se razlaže na manje celine na osnovu kojih se prave *expression trees*. U modelu, klasa *SubCondition* služi za opisivanje pomenutih manjih celina koje se objedinjuju u uslovu *(en. Condition)* pravila. Klasa *ConditionMethods* je pomoćna klasa koja ima ulogu u opisivanju složenih izraza. U ovom radu, akcije pravila predstavljaju *expression tree* gde su opisani pozivi odgovarajućih metoda od interesa. U kreiranju *expression tree-a* učestvuju klase *ConditionBuilder*, *ActionBuilder* i *ExpressionBuilder*.

Kompjaliranje *expression trees-a*: Kreirana pravila još uvek se ne mogu izvršiti. Uslovi i akcije predstavljeni su preko strukture stabla. Kako bi se dobio izvršni kod koji će neki podatak propustiti kroz stablo potrebno je izvršiti kompajliranje. Bitno je napomenuti da se u procesu kompajliranja vrši validacija *expression tree-a*. Proverava se sintaksa i tipovi korišćeni u *expression tree-u*. Pravila je potrebno kompajlirati samo jednom čime se dobija na efikasnosti.

Izršavanje pravila: Promene signala opisuje klasa *SignalChange* i one se mogu odnositi na diskretne ili analogne tipove signala. U zavisnosti o kojem tipu signala je reč proveravaju se različiti tipovi alarma. *AlarmDetector* detektuje promene signala, određuje tip signala i inicijalizuje svojstva određenih tipova alarma na osnovu pristiglih promena. Samo pravila čiji uslovi vrše neku proveru nad promenjenim svojstvima se evaluiraju. U procesu evaluacije ispituje se istinitost uslova pravila i u zavisnosti od rezultata izvršava se pravilo, tj. njegova akcija. Agenda sadrži sva pravila koja treba izvršiti i ukoliko ih ima više određuje redosled njihovog izvršavanja. Izvršeno pravilo se briše iz agende. U ovom rešenju mehanizam zaključivanja zasnovan je na metodi *ulančavanja unapred* što znači da se, nakon izvršavanja jednog pravila, analiziraju ostala pravila i ona čiji uslovi sadrže svojstvo afektovano predhodno izvršenim pravilom se dodaju u agendu. Proces prestaje kada je agenda prazna, tj. nema pravila koja čekaju na izvršenje.

6.1 Dodavanje novih tipova alarma u mašinu pravila

Svaki tip alarma opisan je svojstvima *(AlarmProperties)* koje predstavljaju činjenice mašine pravila. Dodavanje novog tipa alarma odvija se u dva koraka. Prvi korak je kreiranje činjenica, tj. dodavanje nove klase *AlarmProperties* koja sadrži svojstva koja opisuju novi tip alarma (Slika 3). Drugi korak je pisanje pravila u konfiguracionom fajlu na osnovu novo dodatih činjenica.



Slika 3: *Dodavanje novog tipa alarma u mašinu pravila*

7. ZAKLJUČAK

Osnovni cilj ovog rada bilo je pojednostavljenje održavanja alarmnog sistema u SCADA softveru uvođenjem rešenja zasnovanog na mašini pravila. Prednosti koje su dobijene implementacijom mašine pravila su pre svega razlaganje složenog algoritma alarmiranja na manje delove. Sa pojednostavljenjem logike olakšana je modifikacija alarma, u smislu dodavanja novih pravila alarma, ali i njihovog brisanja. Poboljšana je čitljivost koda i smanjen nivo kompleksnosti.

Dalji razvoj mašine pravila bio bi fokusiran na kreiranju korisničkog interfejsa *(en. User Interface)* preko kojeg bi se unosila pravila. Korisnički interfejs bi olakšao pisanje pravila i omogućio ne ekspertima sistema da na jednostavniji način unose pravila.

8. LITERATURA

[1] Branislav Atlagić: “Softver sa kritičnim odzivom u elektroenergetskim sistemima, Univerzitet u Novom Sadu, Fakultet tehničkih nauka”, 2014

[2] Norwegian Petroleum Directorate: “Principles for alarm system design”, 2001 http://www.ptil.no/getfile.php/135975/Regelverket/Alarm_system_design_e.pdf

[3] IEC 62682: “Management of Alarm Systems for the Process Industries”, 2010

[4] Miloš Radosavljević: “"Mašina pravila" ekspertski sistem dokazivanja zasnovan na pravilima”, master rad, Matematički fakultet Beograd, 2016

[5] Martin Fowler: “Domain Specific Languages”, Addison Wesley, 2011

[6] “The Rule Engine” https://docs.jboss.org/drools/release/5.3.0.Final/drools-expert-docs/html/ch01.html

Kratka biografija:

Vladimir Papuga je rođen je 1992. godine u Vrbasu. Osnovne akademske studije na Fakultetu tehničkih nauka, smer “Računarstvo i automatika”, završio je 2015. godine. Na istom fakultetu upisuje Master akademske studije 2015. godine.