ELEKTROTEHNIŠKI VESTNIK 78(1-2): 36–41, 2011
EXISTING SEPARATE ENGLISH EDITION

Merjenje časa na športnih tekmovanjih z domensko specifičnim jezikom EasyTime

Iztok Fister ml., Iztok Fister

Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Smetanova 17, 2000 Maribor, Slovenija
E-pošta: iztok.fister@uni-mb.si

Povzetek. Danes si spremljanje rezultatov na športnih tekmovanjih brez elektronskega merjenja časa težko predstavljamo. Večina elektronskih merilnih naprav temelji na tehnologiji RFID, ki postaja čedalje cenejša in dostopnejša vse širšemu krogu uporabnikov. Za učinkovito delo pa te zahtevajo zmogljiv računalniški sistem, ki omogoča sprotno obdelavo dogodkov, ki nastajajo na merilnih napravah, dekodiranje rezultatov, njihov zapis v podatkovno bazo, sortiranje tekmovalcev glede na doseženi rezultat in izpis liste rezultatov. V članku razvijemo domensko specifični jezik EasyTime, ki omogoča merjenje več kontrolnih točk, na katerih želijo organizatorji spremljati vmesne rezultate, na eni sami merilni napravi in s tem zmanjšanje števila potrebnih merilnih naprav. Jezik je univerzalen, saj ga je mogoče uporabiti pri merjenju različnih športnih tekmovanj.

Ključne besede: domensko specifični jeziki, sintaktični analizator, notacija BNF, generator kode, merjenje časa, tehnologija RFID

Measuring Time in Sporting Competitions with the Domain-Specific Language EasyTime

Measuring time in mass sporting competitions is unthinkable manually today because of their long duration and unreliability. Besides, automatic timing devices based on the RFID technology have become cheaper. However, these devices cannot operate stand-alone. To work eficiently, they need a computer timing system for monitoring results. Such system should be capable of processing the incoming events, encoding and assigning results to a individual competitor, sorting results according to the achieved time and printing them. In this paper, a domain-specific language named EasyTime will be deffined. It enables controlling an agent by writing events to a database. Using the agent, the number of measuring devices can be reduced. Also, EasyTime is of a universal type that can be applied to many diferent sporting competitions.

1 UVOD

Ni še daleč čas, ko so rezultate na športnih prireditvah merili časomerilci, ki so čas merili ročno. Čas, ki ga je pokazala štoparica, so povezali s startno številko tekmovalcev in te razvrstili glede na dosežen rezultat in kategorijo. Z nastankom tehnologije RFID (angl. Radio Frequency Identification) se je merilna tehnologija pocenila (ChampionChip [7], RFID Race Timing Systems [8]) in postala dostopna tudi širšemu krogu uporabnikov (npr. športnim društvom, prirediteljem športnih tekmovanj ipd.), ki so tako postali konkurenca monopolnim podjetjem (Timing Ljubljana [9]) pri spremljanju rezultatov na manjših športnih tekmovanjih.

Prejet 4. oktober, 2010
Odobren 10. februar, 2011

Za spremljanje rezultatov na športnih tekmovanjih poleg merilne tehnologije potrebujemo fleksibilen računalniški sistem, ki omogoča spremljanje različnih športnih tekmovanj s poljubnim številom merilnih naprav in merilnih mest, sprotno beleženje časov, izpis rezultatov, zanesljivost in varnost. Fleksibilnost sistema lahko povečamo z uporabo domensko specifičnega jezika (angl. Domain Specific Language, krajše DSL) EasyTime.

Domensko specifični jeziki [1] so prikrojeni aplikacijski domeni in imajo določene prednosti pred splošno namenskimi jeziki v specifični domeni. Te prednosti se kažejo predvsem v večji izrazni moči in s tem večji produktivnosti, lažji uporabi (tudi za domenske strokovnjake, ki niso programerji), preprostejši verifikaciji in optimizaciji. Domensko specifični jezik EasyTime uporabljamo pri konfiguraciji agentov, ki skrbijo za zapis dogodka z merilnih naprav v podatkovno bazo. Agenti so tako ključni elementi merilnega sistema. Z njihovo ustrezno konfiguracijo lahko zmanjšamo število potrebnih merilnih naprav.

Struktura članka v nadaljevanju je naslednja: v drugem poglavju opisujemo probleme, s katerimi se soočamo pri merjenju časa na športnih tekmovanjih. Pri tem se osredinimo predvsem na triatlonska tekmovanja, ki so zaradi merjenja časa kar treh panog in njihovih dolžin za obvladovanje najtežja. V tretjem poglavju predstavimo DSL EasyTime. V četrtem poglavju opisujemo delovanje programa, napisanega v DSL EasyTime. Članek končamo s kratko analizo opravljenega dela in pogledom v prihodnost.

2 MERJENJE ČASA NA ŠPORTNIH TEKMOVANJIH

V praksi lahko merimo čas na športnih tekmovanjih ročno (*klasična ali računalniška štoparica*) ali avtomatsko (*merilna naprava*). Računalniška štoparica je računalniški program, ki se po navadi izvaja na delovni postaji (prenosni računalnik) in meri realni čas. Pri tem izkorišča procesorski takt, t. j. hitrost, s katero procesor izvaja računalniške instrukcije. Računalniška štoparica omogoča beleženje dogodkov, t. j. prehodov tekmovalca prek *merilnih mest* (krajše MM), podobno kot merilna naprava, vendar v tem primeru sproži dogodek operater na računalniku s pritiskom na ustrezen gumb. Operater generira dogodke v obliki trojk $\langle \#, MM, CAS \rangle$, kjer $\#$ označuje startno številko tekmovalca, MM merilno mesto, na katerem je dogodek nastal, in CAS časovni žig (angl. Timestamp), ki ga je zabeležil računalnik v trenutku nastanka dogodka in pomeni število sekund od 1.1.1970 ob času 0:0:0.

Merilne naprave danes po navadi temeljijo na tehnologiji RFID [2], pri kateri izvajamo identificiranje s pomočjo elektromagnetnega valovanja na področju radijskih frekvenc in sestojijo iz naslednjih elementov:

- čitalca oznak RFID,
- primarnega pomnilnika,
- zaslona LCD in
- numerične tipkovnice.

Na napravo lahko priključimo več antenskih polj, ki predstavljajo posamezna merilna mesta. Tekmovalci generirajo dogodke pri prehodu prek antenskega polja s pomočjo pasivnih oznak RFID, ki hranijo identifikacijsko številko tekmovalca. Ta je unikatna in se razlikuje od štartne številke. Dogodek z merilne naprave predstavimo s četvercem oblike $\langle \#, RFID, MM, CAS \rangle$, kjer je že omenjeni trojki dodana identifikacijska številka oznake $RFID$. Natančnost odčitka merilne naprave RFID je po navadi $1/10$ sekunde, kar ustreza zahtevam sodniških organizacij.

Merilne naprave in delovne postaje z nameščeno računalniško štoparico, ki so v merilnem sistemu merilna mesta, lahko priključimo na lokalno omrežje in z njimi komuniciramo prek nadzornega programa (agenta), ki teče na strežniku s podatkovno bazo. Ta se z merilno napravo poveže prek ustreznih vtičnic TCP/IP (angl. socket) z določenim protokolom. Po navadi merilne naprave podpirajo protokol *Telnet*, ki je za implementacijo zelo preprost in omogoča komunikacijo v tekstovni obliki. Agent komunicira z ročno štoparico prek prenosa datotek.

2.1 Primer: merjenje časa na triatlonih

Poseben pristop zahtevajo triatlonska tekmovanja, kjer imamo na enem tekmovanju opravka s tremi panogami, zato se temu problemu v članku tudi podrobneje posvečamo.

Triatlonsko tekmovanje so prvič izvedli v ZDA leta 1975. To tekmovanje je danes že olimpijska panoga, pri kateri tekmovalci najprej plavajo, nato kolesarijo in nazadnje še tečejo. Vse aktivnosti potekajo brez prekinitve, v skupni čas pa se štejeta tudi časa obeh menjav, t. j. ko tekmovalec preide iz vode na kolo in nato še s kolesa na tek. Triatlonskih tekmovanj je več vrst, ki se med seboj razlikujejo po dolžini posameznih prog. Zaradi preprostejše organizacije prireditelji namesto daljših prog uporabljajo raje krožne proge krajših dolžin. S tem pa povečujejo zahtevnost merjenja.

Merjenje časa na triatlonskih tekmovanjih delimo v devet kontrolnih točk (slika 1). Kontrolne točke so lokacije na tekmovalni progi, kjer želi organizator zasledovati izmerjeni čas. Ta je lahko *vmesni* ali *končni*. Na sliki 1 imamo opravka s progo za dvojni triatlon (7,6 kilometra plavanja, 360 kilometrov kolesarjenja in 84,4 kilometra teka), kjer je proga za plavanje dolga 380 metrov (20 krogov), proga za kolesarjenje 3,4 kilometra (105 krogov) in proga za tek 1,5 kilometra (55 krogov).

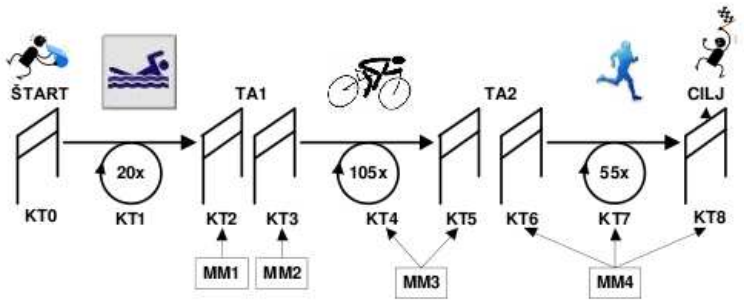
Skupni čas triatlonskega tekmovanja je sestavljen iz petih končnih (čas plavanja SWIM (KT2), čas prve menjave TA1 (KT3), čas kolesarjenja BIKE (KT5), čas druge menjave TA2 (KT6) in čas teka RUN (KT8)) in treh vmesnih časov (vmesni čas plavanja (KT1), vmesni čas kolesarjenja (KT4) in vmesni čas teka (KT7)). Pri vmesnih časih merimo tako število krogov ROUND_x kot tudi doseženi rezultat INTER_x, kjer $x = 1 \ldots 3$ označuje posamezno panogo.

Vzemimo, da imamo za merjenje tekmovanja, prikazanega na sliki 1, na voljo merilno napravo z dvema merilnima mestoma (MM3 in MM4), in da tekma poteka na eni lokaciji. V tem primeru lahko zadnji prehod čez MM3 pomeni čas KT5, prvi prehod čez MM4 čas KT6 in zadnji prehod čez MM4 končni dosežek (KT8). Merilni mesti MM1 in MM2 pomenita ročno merjenje časa. S pravilno postavljenimi kontrolnimi točkami in merilnim sistemom lahko število merilnih mest zmanjšamo za tri. Z eno merilno napravo tako izmerimo 162 dogodkov na tekmovalca (oz. 87,6%). Če temu dodamo še, da je merilna tehnologija za merjenje plavanja v jezerih oz. morju še zelo draga in zato plavanje najpogosteje merijo sodniki ročno, lahko na omenjeni tekmi avtomatsko izmerimo kar 98 odstotkov vseh dogodkov.

3 DSL EASYTIME

Z DSL EasyTime želimo opisati različna merjenja, ki jih izvajamo z merilnim sistemom, pri kompleksnejših merjenjih pa z manj merilnimi mesti pokriti več kontrolnih točk. DSL EasyTime omogoča opisovanje pravil za krmiljenje *agenta* pred zapisom dogodka, ki prihaja z merilnih mest, v podatkovno bazo. Vsak program, zapisan v tem domensko specifičnem jeziku, je treba pred izvajanjem najprej prevesti. Prevajanje je sestavljeno iz dveh faz:

- sintaktične analize in



Slika 1: Definicija kontrolnih točk pri triatlonu

- generiranja kode.

Sintaktično analizo izvaja sintaktični analizator (razpoznavalnik), ki program, zapisan v DSL EasyTime, prevede v vmesno kodo. Iz vmesne kode generator kode generira izvajalno kodo za navidezni računalnik in jo shrani v podatkovno bazo. Ker generirana koda predstavlja pravila za krmiljenje agenta, imenujemo tabelo, v katero shranimo izvajalni program, tudi tabelo pravil (angl. Rules). V nadaljevanju predstavljamo značilnosti sintaktičnega analizatorja in generatorja kode podrobneje.

3.1 Sintaktični analizator

Pred razvojem sintaktičnega analizatorja je treba definirati sintakso domensko specifičnega jezika, t. j. množico pravil, s katerimi definiramo strukturo pravilnih stavkov (gramatika), ki sestavljajo program v DSL EasyTime [6]. Sintakso DSL EasyTime predstavljamo v sintaktičnih diagramih na sliki 2, ki so najprimernejša oblika za pisanje sintaktičnih analizatorjev. Sintaktični diagrami so izrazno enako močni kot BNF (angl. Backus Naur Form) [4], t. j. notacijska tehnika za kontekstno neodvisne gramatike. Sintaktični analizator smo razvili v programskem jeziku C/C++ [3].

Tabela 1: Imena spremenljivk v tabeli $RESULTS$

Spremenljivka	Opis
ROUND_1	Število krogov pri plavanju
INTER_1	Vmesni čas 1 (KT1)
SWIM	Končni čas plavanja (KT2)
TRANS_1	Čas prve menjave (KT3)
ROUND_2	Število krogov pri kolesarjenju
INTER_2	Vmesni čas 2 (KT4)
BIKE	Končni čas kolesarjenja (KT5)
TRANS_2	Čas druge menjave (KT6)
ROUND_3	Število krogov pri teku
INTER_3	Vmesni čas 3 (KT7)
RUN	Končni čas triatlona (KT8)

Program v DSL EasyTime je sestavljen iz definicij:

- agentov,
- spremenljivk in
- merilnih mest.

Agente opisujemo s stavčnim konstruktom $Agenti$ prikazanim v istoimenskem sintaktičnem diagramu. Spremenljivke, s katerimi označujemo imena kolon v podatkovni bazi (tabela $RESULTS$) in pomenijo kontrolne točke na sliki 1, so prikazane v tabeli 1. Te morajo biti definirane pred uporabo v stavčnem konstruktu $Merilna\_mesta$, s katerim definiramo pravila, ki veljajo na določenem merilnem mestu. Pravila so oblike ⟨Predikat⟩::=⟨Operacija⟩. V DSL EasyTime pred imena spremenljivk postavimo znak *$*.

Primer definicije agentov v DSL EasyTime, s katerim opišemo merjenje na sliki 1, je prikazan v programu 1, kjer prvi agent shranjuje rezultate ročnega merjenja v mapi $/home/DC2$ in drugi agent dobiva podatke od merilne naprave, z naslovom IP 192.168.225.100 prek protokola UDP in vrat 9999, avtomatsko.

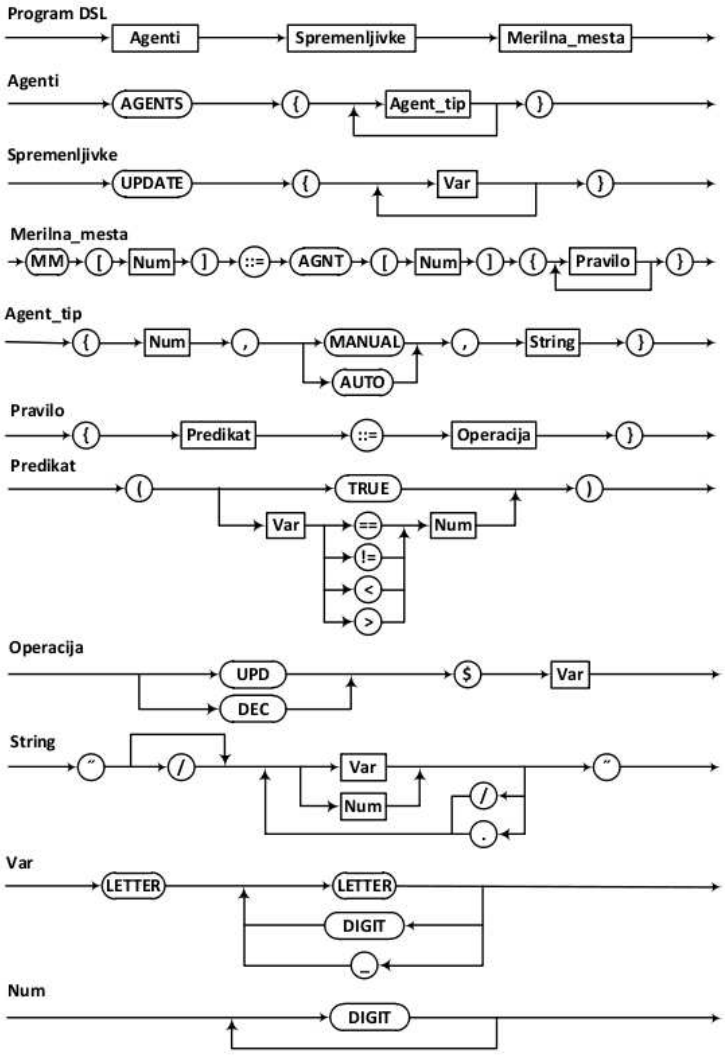
Program 1 Definicija agentov

```
1: AGENTS {
2:     {1,MANUAL,"/home/DC2/res.ets"}
3:     {2,AUTO,"192.168.225.100/UDP/9999"} }
```

Program 2 Definicija merilnih mest

```
1: MM[1] ::= AGNT[1] {
2:     { (TRUE) ::= UPD $SWIM }
3:     { (TRUE) ::= DEC $ROUND_1 } }
4: MM[2] ::= AGNT[1] {
5:     { (TRUE) ::= UPD $TRANS_1 } }
6: MM[3] ::= AGNT[2] {
7:     { (TRUE) ::= UPD $INTER_2 }
8:     { (TRUE) ::= DEC $ROUND_2 }
9:     { (ROUND_2 == 0) ::= UPD $BIKE } }
10: MM[4] ::= AGNT[2] {
11:     { (TRUE) ::= UPD $INTER_3 }
12:     { (ROUND_3 == 55) ::= UPD $TRANS_2 }}
13:     { (TRUE) ::= DEC $ROUND_3 }
14:     { (ROUND_3 == 0) ::= UPD $RUN } }
```

V DSL EasyTime določimo pravila za merilna mesta s slike 1 z izvorno kodo, prikazano v programu 2. Vsako merilno mesto je označeno s svojo identifikacij-



Slika 2: Sintaksa DSL EasyTime

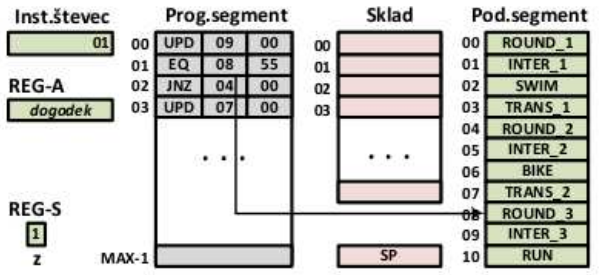
sko številko in povezano z ustreznim agentom. Pravila za merilno mesto 4 npr. določajo, da dogodek, ki prihaja z merilne naprave, najprej posodobi vmesni čas teka $INTER\_3$. Če gre za prvi prehod tekmovalca prek merilnega mesta (predikat $ROUND\_3 == 55$), posodobimo čas druge menjave $TRANS\_2$. Sledi zmanjšanje števila krogov teka (spremenljivka $ROUND\_3$) za ena. Končno agent v primeru, da je tekmovalec v zadnjem krogu (predikat $ROUND\_3 == 0$), razglasi končni rezultat, t. j. posodobi spremenljivko RUN, ki hkrati pomeni končni dosežek tekmovalca.

3.2 Generator kode

Generiranje kode [5] se izvaja samo, če se je sintaktična analiza končala uspešno. Pri neuspešnem prevodu programa sintaktični analizator izpiše poročilo o napakah. Generator DSL EasyTime generira kodo za vsako merilno mesto posebej. Generirano programsko kodo shranimo v podatkovni bazi. Tudi generator smo razvili v programskem jeziku C/C++.

Pred generiranjem kode moramo poznati arhitekturo procesorja, za katerega generiramo kodo. Prevedeni program v DSL EasyTime izvajamo na navideznem računalniku (angl. Virtual Machine). V luči paralelizacije procesov (več nitenje) za vsako merilno mesto definiramo svoj navidezni računalnik. Arhitektura navideznega računalnika (slika 3) je zelo preprosta, saj sestoji iz programskega segmenta, sklada, podatkovnega segmenta, instrukcijskega števca, programskega in statusnega registra. V programski segment naložimo generirano kodo. Na skladu izvajamo aritmetično-logične operacije. Podatkovni segment vsebuje spremenljivke iz podatkovne baze. Instrukcijski števec kaže na instrukcijo v programskem segmentu, ki jo trenutno interpretiramo. Podatkovni register (REG-A) je namenjen shranjevanju časovnega žiga dogodka, ki ga obdeluje agent. Statusni register (REG-S) določa status predikatov in je dvojiška vrednost ($TRUE$, če z==1 ali $FALSE$, če z==0).

Programske instrukcije, ki jih naložimo v programski



Slika 3: Arhitektura navideznega računalnika

segment navideznega računalnika, definiramo kot trojico $\langle op, p1, p2\rangle$, kjer op pomeni operacijsko kodo, $p1$ in $p2$ pa parametra (spremenljivka ali konstanta). Nabor instrukcij je sestavljen iz logičnih instrukcij (EQ in NEQ), operacij (UPD, DEC in STOP) in vejitvene instrukcije (JNZ). Logične instrukcije vplivajo na nastavitev statusnega registra, ki krmili delovanje vejitvene instrukcije.

Generator generira kodo iz podatkovnih struktur, ki jih zgradi sintaktični analizator. Dejansko kodo generira samo iz podatkovnih struktur, zgrajenih iz stavčnega konstrukta $Merilna\_mesta$. Imena spremenljivk, zbranih v tabeli, določajo njihove naslove v podatkovnem segmentu. Ti nastopajo v instrukcijah kot parametri, medtem ko je podatkovna struktura, zgrajena iz stavčnega konstrukta $Agenti$, namenjena konfiguraciji posameznih agentov pred izvajanjem. Primer generirane kode, zapisane v simbolični obliki v programu 2, ki predpisuje pravila za agenta na merilnem mestu 3, prikazuje tabela 2.

Tabela 2: Programska koda za merilno mesto 3

Pš	OP.K.	P1	P2
00	UPD	5	0
01	DEC	4	0
02	EQ	4	0
03	JNZ	5	0
04	UPD	6	0
05	STOP	0	0

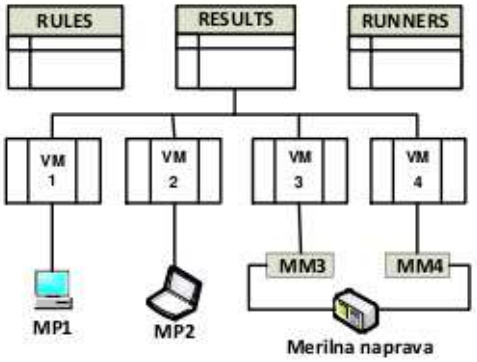
Iz primera lahko razberemo, da je generirana koda zelo optimizirana, saj iz treh vrstic izvorne kode dobimo šest vrstic izvajalne kode. Ker je vsaka instrukcija, ki jo zapisujemo v podatkovno bazo, dolga štiri bajte (operacijska koda, dva parametra in delimiter ";"), velikost prevedenega programa za podatkovno bazo ne pomeni nikakršnih težav.

4 DELOVANJE AGENTA

Agent, ki ga krmilimo s pomočjo DSL EasyTime, lahko obdeluje dogodke:

- paketno: ročni način delovanja ($MANUAL$),
- sprotno: avtomatski način delovanja ($AUTO$).

Pri paketni obdelavi agent dogodke, zbrane v tekstovni datoteki, bere in zapisuje v ustrezno podatkovno bazo. Po navadi paketno obdelujemo dogodke, ki smo jih zajeli prek računalniške štoparice. Pri tem načinu delovanja agent vsako sekundo preverja, ali datoteka, ki smo jo konfigurirali v tabeli agentov, obstaja ali ne. Če ta obstaja, zažene paketno obdelavo. Datoteko ob koncu obdelave najprej arhivira, potem pa zbriše. Sprotno obdelovanje je dogodkovno orientirano, t. j. vsak dogodek, ki ga pošilja merilna naprava, agent obdela sproti. Izvajalno okolje, ki ga zgradi program v DSL EasyTime za spremljanje triatlonskega tekmovanja s slike 1, prikazuje slika 4.



Slika 4: Izvajalno okolje programa v DSL EasyTiming

Pri obeh načinih obdelave operira agent s tabelami: pravil ($RULES$), tekmovalcev ($RUNNERS$) in rezultatov ($RESULTS$). Ob zagonu agenta poteka inicializacija navideznega računalnika, ki zajema nalaganje programske kode iz tabele pravil v podatkovni bazi, kar pomeni, da se ta naloži samo prvič. Sočasno inicializiramo tudi vrednosti spremenljivk v podatkovnem segmentu. Zapisovanje dogodka, ki ga obdeluje agent, lahko razbijemo v naslednje faze:

- rekonstrukcijo dogodka: tekmovalca identificiramo prek startne številke (#) oz. oznake RFID, merilno mesto (MM) določa številko navideznega stroja, čas (ČAS) pa je časovni žig dogodka,
- branje rezultatov: iz tabele $RESULTS$ na podlagi startne številke (#) oz. oznake RFID, preberemo trenutni rezultat tekmovalca, ki je sprožil dogodek,
- preslikava rezultatov: preslikamo prebrane rezultate tekmovalca v podatkovni segment navideznega računalnika, ki ga določa merilno mesto (MM), in napolnimo programski register z vrednostjo časovnega žiga dogodka,
- interpretacija kode: postavimo programski števec na nič in zaženemo program, naložen v ustreznem navideznem računalniku,
- zapis rezultatov: zapis rezultatov s podatkovnega segmenta ustreznega navideznega računalnika v tabelo $RESULTS$.

Program v navideznem računalniku interpretiramo sekvenčno, t. j. instrukcijo za instrukcijo, dokler ne pridemo do instrukcije STOP.

5 SKLEP

Pri razvoju univerzalne programske opreme za merjenje časa na športnih tekmovanjih se pogosto srečamo s problemom fleksibilnosti merilnega sistema. V ta namen smo razvili DSL EasyTime, ki omogoča hitro prilagajanje merilnega sistema zahtevam različnih tekmovanj. Za novo tekmovanje spremenimo izvorni program, zapisan v DSL EasyTime, ga prevedemo in ponovo zaženemo agenta. Uporaba DSL EasyTime v praksi je pokazala, da za merjenje časa na manjših športnih tekmovanjih ni treba več najemati specializiranih in zelo dragih podjetij, pri večjih tekmovanjih pa bi lahko poenostavil konfiguracijo merilnega sistema. V nadaljevanju dela želimo DSL EasyTime nadomestiti z domensko specifičnim modelirnim jezikom.

LITERATURA

[1] M. Mernik, J. Heering, A. Sloane: When and how to develop domain-specific languages, 2005, ACM computing surveys, vol. 37, no. 4, pp. 316-344
[2] K. Finkenzeller: RFID Handbook, 2010, John Willey, Chichester, UK
[3] N. Wirth: Algorithms + Data Structures = Programs, 1978, Prentice Hall PTR, Upper Saddle River, NJ, USA
[4] A.V. Aho, J.D. Ullman: The theory of parsing, translation, and compiling (Volume I: Parsing), 1972, Prentice Hall PTR, Upper Saddle River, NJ, USA
[5] A.V. Aho, J.D. Ullman: The theory of parsing, translation, and compiling (Volume II: Compiling), 1972, Prentice Hall PTR, Upper Saddle River, NJ, USA
[6] A.V. Aho, M.S. Lam, R. Sethi, J.D. Ullman: Compilers: Principles, Techniques, and Tools with Gradiance, 2007, Prentice Hall PTR, Upper Saddle River, NJ, USA
[7] ChampionChip2010, http://www.championchip.com
[8] RFIDTechnology2010, http://www.rfidtiming.com
[9] Timing2010, http://www.timingljubljana.si

Iztok Fister ml. je študent 3. letnika smeri Računalništva in informacijskih tehnologij na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru.

Iztok Fister je diplomiral leta 1983 na Univerzi v Ljubljani. V letu 2004 je magistriral na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru in v letu 2007 na isti fakulteti tudi doktoriral. Je asistent v Laboratoriju za računalniške arhitekture in jezike. Ukvarja se s programskimi jeziki, operacijskimi raziskavami in evolucijskim računanjem.