KAUNO TECHNOLOGIJOS UNIVERSITETAS
PROGRAMŲ INŽINERIJOS KATEDRA

VITALIJ SUCKEL

AUTOMATINĖ ŠABLONINIŲ SITUACIJŲ ANALIZĖ SPORTINIŲ ĮVYKIŲ SEKOSE

Magistro baigiamasis darbas

Darbo vadovas: doc. Eimutis Karčiauskas

KAUNAS, 2014

Vadovas
doc. E. Karčiauskas

_____ _____
(parašas) (data)

Recenzentas
prof. dr. R. Butleris

_____ _____
(parašas) (data)

Studentas
V. Suckel

_____ _____
(parašas) (data)

TURINYS

SANTRAUKA

Šiame darbe nagrinėjami didelių sportinių duomenų kiekių analizės būdai. Analizei skirti duomenys saugomi darbo metu sukurtoje sistemoje ir gaunami iš išorinių šaltinių. Darbo rašymo metu sistemos MySQL duomenų bazėje saugomas didelis duomenų kiekis – virš 400 tūkstančių įrašų apie krepšinio rungtynių įvykius, todėl yra ypač svarbu rasti optimalų algoritmą ir priemones tokių duomenų analizei.

Darbo tikslas yra sukurti, realizuoti bei išbandyti duomenų analizės priemones bei algoritmus, kurie padėtų efektyviai nagrinėti sistemoje kaupiamus duomenis. Vienas svarbiausių darbo uždavinių – šabloninių situacijų išryškinimas.

Atlikus tyrimą buvo realizuotas algoritmas, pasinaudojant kuriuo buvo atlikti eksperimentai. Jų vykdymo metu aptiktos dažniausiai krepšinio rungtynėse pasitaikančios šabloninės situacijos. Tyrimo eigoje taip pat išryškintos bei išnagrinėtos kelios problemos, susijusios su duomenų gavyba. Darbe aprašyt sukurti įrankiai ir algoritmai, pateiktos jų veikimui naudojamos SQL užklausos, pavyzdžiai bei eksperimentinio tyrimo vykdymo eiga ir rezultatai.

SUMMARY

Ways of analyzing vast amount of sports events data are examined in this paper. Data for analysis is stored in the system that was developed in process of writing this thesis. At the time of writing there are more than 400 thousand records about basketball events stored in the MySQL database that system uses for it's work. Therefore it is important to find the optimum algorithm and way to analyze this data.

The main goal is to create, implement and test the algorithm which would provide ability to effectively examine data that is stored in the system. One of the most important task was to find a way to discover patterns in whole data set.

The algorithm was implemented and some experiments were performed following the research. As a result of experiments most popular basketball game events patterns were discovered. Some data mining problems were highlighted and examined during research in order to develop the optimum algorithm for analysis. Descriptions and specifications of created tools and algorithms are presented in this paper along with actual SQL queries, examples, process and results of the experiments.

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

1. ĮVADAS

Jau ilgą laiką bandoma suprasti ir sumodeliuoti žmonių elgseną tam tikrose situacijose. Tam nuolat skiriama daug dėmesio ir darbo [1], atliekami moksliniai tyrimai šioje srityje. Dauguma tokių tyrimų orientuojasi į vieno žmogaus elgesį tam tikrose situacijose, bet ne į žmonių grupės elgesį. Tačiau yra ir tam tikrų išimčių [2, 3].

Žinoma, kad žmonės susiburia į grupes tam, kad pasiektų tam tikrą tikslą. Suprasti tokios grupės elgseną, ypač jei grupė yra valdoma ir koordinuojama, yra labai svarbu [1]. Sportinėse rungtynėse žaidėjai nėra izoliuoti vienas nuo kito ir nuolatos sąveikauja. Todėl visi įvykiai turėtų būti apžvelgiami įvairiais aspektais, tame tarpe ir visos žmonių grupės (komandų) atžvilgiu. Sąryšiai tarp komandos narių ir jų veiksmų tampa kritiškai svarbūs norint suprasti ką, kodėl ir kaip komanda daro teisingai arba ne.

Šiais laikais sportinių varžybų įvykiai yra fiksuojami skaitmeninėje erdvėje. To dėka visus varžybų metu sukauptus duomenis galima atvaizduoti labai patogiais vizualiniais grafikais, greitai ir efektyviai analizuoti įvykusį žaidimą. Taip treneriai ir kiti suinteresuoti asmenys gali vaizdžiai pamatyti ir suprasti sėkmingo arba nesėkmingo žaidimo priežastis, bandyti tobulinti esamas arba, atsižvelgiant į sukauptus duomenis, kurti naujas žaidimo strategijas [1, 2].

Žaidėjų komandos efektyvumo analizė nebegali apsieiti be kompiuterinių skaičiavimų. Nuolat vystomi nauji algoritmai ir kuriamos naujos sistemos tokiai analizei atlikti. Tačiau dauguma tokių sistemų, pasinaudojant sukauptais duomenimis apie rungtynes, generuoja ir pateikia tik standartinę statistiką – grafiškai arba lentelės pavidalu atvaizduoja sukauptus duomenis, nors dažnai verta labiau įsigilinti į šių įvykių tarpusavio sąryšį ir bandyti suprasti kodėl rungtynės nuėjo viena arba kita linkme [3].

Turint omeny šios problemos sudėtingumą bei aktualumą, būtent jos dalinis sprendimas ir buvo pasirinktas šio darbo tikslu. Ištyrus šabloninių situacijų specifiką planuojama pritaikyti jau egzistuojančius arba sukurti specialų algoritmą, kurio pagalba tokių situacijų išryškinimas taptų paprastesnis.

Šio darbo kontekste bus nagrinėjamos jo metu sukurtos bei vystomos sporto įvykių analizės ir vizualizavimo sistemos *SEA*[1] (toliau – SEA) realizacija bei tobulinimo galimybės. Tam tikslui turės būti išnagrinėti ir realizuoti jau egzistuojantys sprendimai bei algoritmai, o esant reikalui ir sukurti nauji. Kadangi tyrimas vyks SEA sistemos terpėje, tiriami algoritmai turės būti pritaikyti prie šios sistemos, o keliamos prielaidos bei hipotezės nagrinėjamos jos kontekste.

[1] Sporto įvykiu analizės ir vizualizavimo sistema SEA – tai Vitalijaus Suckel magistro studijų metu sukurta programinė įranga, leidžianti kaupti ir analizuoti sportinių įvykių duomenis.

2. ANALITINĖ DALIS

2.1. Bendros srities problemos

Pasaulyje yra sukurta nemažai įrankių, leisiančių atlikti sportinių duomenų analizę. Taip pat sukurta ir ištirta dešimtys algoritmų, skirtų duomenų gavybai[2] ir analizei. Šie įrankiai ir algoritmai sprendžia įvairias problemas, susijusias su duomenų išgavimu bei analize. Kai kurios iš tokių problemų yra trivialios, ir iš pirmo žvilgsnio atrodytų, kad galėtų būti nesunkiai išspręstos. Kitos atvirkščiai, gali reikalauti ženkliai daugiau laiko, bandant ištirti problemų specifiką ir rasti jų sprendimo būdus [4].

Šiame skyriuje bus apžvelgtos kelios tipinės problemos, susijusios su sportinių įvykių registravimu ir analize.

2.1.1. Apibendrinimo kriterijai tarp skirtingų sporto šakų

Skirtingos sporto šakos turi labai skirtingus taisyklių rinkinius ir jų variacijas. Taip pat kiekvieno žaidimo tikslai yra skirtingi. Pavyzdžiui, krepšinyje tikslas yra įmesti kamuolį į krepšį, futbole – įmušti įvartį, boulinge – numušti kuo daugiau keglių. Siekiant šių tikslų žaidėjai privalo laikytis taisyklių ir apribojimų, kurie yra visiškai skirtingi. Šis faktas sukelia nemažą problemą ir rimtą klausimą – pagal kokius kriterijus galima apibendrinti skirtingas sporto šakas?

Kaip galima matyti iš skyriuje *2.3. Egzistuojantys sprendimai* apžvelgtų produktų, daugumos PĮ kūrėjų sprendimas būna gan paprastas – kiekvienai skirtingai sporto šakai kurti vis atskirą programą arba posistemę. Tačiau toks sprendimas ne visada gali būti optimalus.

Daugumai šiais laikais populiariausių sporto šakų vis gi galima rasti bendrus bruožus. Kai kurie mokslininkai išryškina 3 svarbiausias ir dažniausiai pasitaikančias tokių žaidimų būsenas [1]:

- žaidimas puolime (offensive game);
- žaidimas gynyboje (defensive game);
- pertraukos (time outs).

Tokiu būdu sugrupavus žaidimo būsenas galima lengviau įsivaizduoti į ką orientuotis aprašant skirtingas įvairių žaidimų taisykles ir bandant išryškinti šablonines situacijas.

2.1.2. Duomenų gavimas iš išorinių duomenų bazių

Norint leisti programinei įrangai naudotis ne tik savo sukauptais duomenimis, bet ir kitų sistemų kaupiamais ir agreguojamais duomenimis, yra būtina užtikrinti jų tarpusavio sąryšį. Viena svarbesnių problemų atliekant duomenų (pavyzdžiui, sportinių įvykių) iš išorinių sistemų arba duomenų bazių analizę – duomenų iš jų gavimas ir teisingas interpretavimas. Egzistuoja visa aibė sukurtų sprendimų ir protokolų tokiam tarpsisteminiam sąryšiui realizuoti.

Vienas iš projekto tikslų yra tas, kad visa duomenų analizė ir pateikimas turi veikti ir būti atliekami internetinėje erdvėje, todėl verta išanalizuoti įrankius bei technologijas, kurios skirtos sistemų surišimui šioje terpėje. Keli iš tokių apsikeitimo duomenimis būdų ir bus trumpai apžvelgti šiame skyriuje.

2.1.2.1. SOAP protokolas

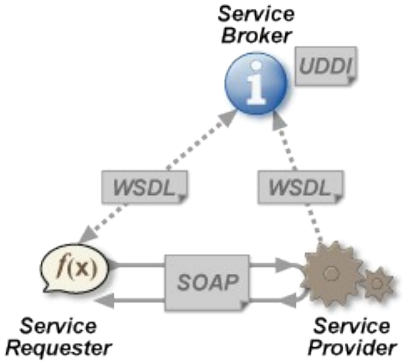
Tai protokolas, specialiai sukurtas informacijos perdavimui iš vienos web sistemos į kitą. Struktūrizuoti duomenis, kuriuos perduoda ir priima sistemos naudojant SOAP[3] protokolą, bazuojami XML žymėjimo kalba [5]. Pranešimų ir duomenų apsikeitimas vyksta virš vieno iš taikomojo lygio duomenų perdavimo protokolo (pvz. HTTP).

---

[2] Duomenų gavyba (angl. *data mining*) – tai skaičiavimo procesas, kurio metu didesniuose duomenų kiekiuose bandoma išryškinti šablonus ir modelius

[3] SOAP (angl. *Simple Object Access Protocol*) – protokolas, paprastai kartu su HTTP protokolu naudojamas XML kalbos formato struktūrizuotiems pranešimams siųsti

Protokolo veikimui reikalingos 2 sudedamosios dalys – SOAP serveris ir klientas. Serveriai dažnai turi privalomas užklausos formato taisykles, kurios yra saugomos WSDL tipo failuose. Klientas, siunčiant XML turinį į SOAP serverį, padaro užklausą, norėdamas gauti duomenis arba įvykdyti komandą. Serveris, apdorojęs užklausą, atsako klientui tuo pačiu XML formatu. Jeigu užklausa neatitinka formato taisyklių, ji yra atmetama.



1 pav. SOAP protokolo veikimas

2.1.2.2. REST architektūra

Duomenys REST[4] architektūroje turi būti perduodami nedidelėmis porcijomis, pasinaudojant standartiniais duomenų formatais (pvz. JSON, XML ar kt.). Yra visa eilė apribojimų ir reikalavimų aplinkai, kad šis metodas veiktų – pavyzdžiui, protokolas, kuriuo bus perduodami duomenis [6]:

- turi palaikyti duomenų kešavimą, siekiant pagreitinti komunikavimą tarp sistemų;
- neturi priklausyti nuo tinklo sluoksnio;
- neturi saugoti būsenų informacijos tarp eilinės "užklausa-atsakymas" poros;
- ir kt.

Reikia turėti omenyje, kad norint sistemų komunikavimą atlikti pasinaudojant REST architektūra, nuotolinė sistema turi būti tinkamai suprojektuota ir įgyvendinta. Šiais laikais vis labiau populiarėja toks architektūros stilius, kur naudojami REST principai [6, 7].

2.1.2.3. Kiti komunikavimo būdai

Komunikavimas tarp sistemų neapsiriboja vien standartiniais protokolais. Tokiais atvejais, kai išorinė sistema nesiūlo standartizuotų būdų išgauti reikiamos informacijos, naudojami specialiai tokiam atvejui pritaikyti sprendimai – rankiniu būdu aprašomas užklausos vykdymas, duomenų transportavimas ir interpretavimas.

Prieš prisirišant prie vieno iš duomenų iš išorinės sistemos gavimo metodų yra būtina sužinoti kokius iš jų dominančios sistemos siūlo. Yra tikimybė, kad kiekvienai iš tokių sistemų bus reikalingas atskiras komunikavimo aprašymas. Šis faktas yra itin svarbus projektuojant programinę įrangą. Nenumačius tokių scenarijų architektūroje, gali ateiti tokia stadija, kai PĮ pasirodys nepakankamai lanksti tam, kad integruoti į ją eilinės reikiamos išorinės sistemos duomenų.

2.1.3. Duomenų vizualizacijos problema

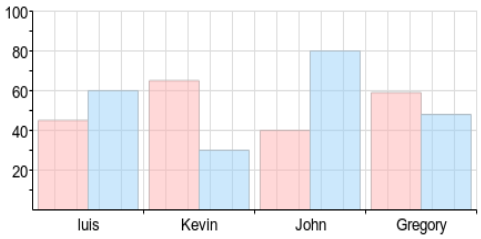
Kuriant sportinių įvykių analizės sistemą, kuri būtų patogi vartotojui, neišvengiamas yra duomenų grafinės vizualizacijos kūrimas. Įrodyta, kad grafinė vartotojo sąsaja yra patogesnė, priimtinesnė ir patrauklesnė PĮ vartotojams (išskyrus labai patyrusius vartotojus), negu komandinės eilutės aplinka [8]. Kalbant apie aktualių duomenų atvaizdavimą šis faktas turi dar didesnę svarbą, nes yra tiesiogiai susijęs su sistemos naudojimo patogumu.

Kai kurios sportinių įvykių analizės sistemos pateikia tik tekstinę statistiką [9], kurioje sunku vaizdžiai pamatyti norimą informaciją – reikia įdėmiai išstudijuoti kiekvieną aspektą, kruopščiai

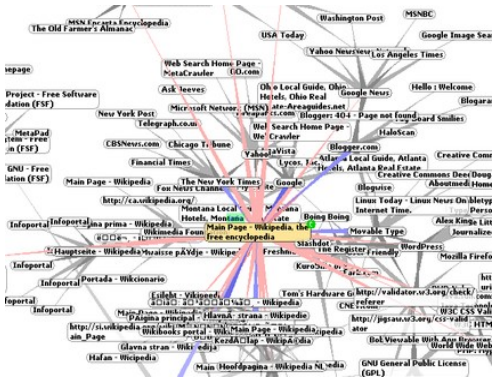
[4] REST (angl. *Representational state transfer*) - tai decentralizuotos, nesaugantis būsenų programinės įrangos architektūros kūrimo stilius, kuris veikia pasinaudojant HTTP protokolu.

įsigilinti į rodomus skaičius. Toks sprendimas nėra priimtinas paprastiems sistemos vartotojams, nes reikalauja daugiau laiko, norint priprasti prie sistemos ir jos specifikos.

Norint vizualizuoti įvairaus tipo sporto įvykių duomenis reikia pasikliauti įvairiais duomenų vizualizacijos būdais. Pavyzdžiui, pavienių žaidėjų kriterijų statistikai atvaizduoti užtenka paprastos juostinės diagramos (2 paveikslas), bet bandant išnagrinėti sąryšius tarp skirtingų žaidėjų sukurtų įvykiu gali prireikti vizualizuoti sudėtingos grafo struktūros duomenis (3 paveikslas)



2 pav. Juostinė diagrama



3 pav. Grafo vizualizavimas

Šiuo metu yra sukurta visa eilė įrankių, padėsiančių vizualizuoti įvairaus sudėtingumo duomenis internetinėje erdvėje [10], tačiau yra svarbu pasirinkti tinkamus iš jų, sprendžiant vieną arba kitą vizualizacijos problemą. Kalbant apie sportinių įvykių vizualizaciją, nemažiau svarbu pateikti duomenis, kurių analizė galėtų padėti pagerinti komandos rodiklius [11].

2.1.4. Šabloninių situacijų atpažinimas

Yra išskiriami keli būdai kaip galima atskirti įvairias šablonines situacijas sprendžiant pagal grupės narių (šiuo atveju – žaidėjų) elgseną. Vienas iš tokių būdų [12]: sistemoje yra aprašytas kuo didesnis rinkinys trajektorijų, kuriais juda pavieniai žaidėjai, o tikslas yra atpažinti ar tų judėjimų visuma atitinka kokį nors standartinį žaidimo šabloną arba taktiką. Žaidimo tipas ir pavienių žaidėjų elgsenos rinkiniai, kurie yra apibendrinti į šablonines situacijas arba taktikas, aprašomi atskirais modeliais. Į elgseną galima žiūrėti ne tik kaip į judėjimą erdvėje, bet ir kaip į objektų sukuriamus įvykius, o pastebint tarp jų nuolatinius sąryšius išskirti tokį rinkinį į šabloną.

Tam, kad atskirti ar tam tikras veiksmų rinkinys atitinka bet kokią šabloninę situaciją, dažnai yra būtina pamatyti ir atpažinti laikinus sąryšius bei tik neakivaizdžiai matomą informaciją [13]. Tam tikrais laiko momentais gali egzistuoti milžiniškas šių laikinų sąryšių kiekis, o ir reikalinga informacija kiekvienoje iš situacijų gali būti matoma tik pažiūrėjus specialiu, tokiai situacijai pritaikytu, aspektu. Tokiu būdu tampa aišku, kad ši problema – tai labai sudėtingas uždavinys. Prie to dar prisideda įvairūs papildomi faktoriai, pavyzdžiui, kad vienas ir tas pats žaidėjas gali žaisti keliose pozicijose [1].

2.2. Duomenų gavyba

Duomenų gavyba – tai skaičiavimo procesas, kurio metu didesniuose duomenų kiekiuose bandoma išryškinti šablonus ir modelius [14]. Procesas apjungia tokias sritis, kaip dirbtinis intelektas, statistika ir duomenų bazės. Pagrindinis tokios gavybos tikslas yra išgauti norimą informaciją iš didelio duomenų rinkinio ir ją transformuoti į lengviau suprantamą bei perskaitomą duomenų struktūrą tolimesniam panaudojimui. Apart pačių analizės žingsnių ir algoritmų, ši technika apima ir duomenų valdymo aspektus, duomenų paruošimą, statistinių modelių ir išvadų apžvalgas, įvairias metrikas, duomenų agregavimą ir vizualizavimą [14].

Faktinis duomenų gavybos uždavinys yra automatinė arba pusiau automatinė didelio duomenų kiekio analizė tam, kad išgauti iš anksto nežinomus arba žinomus šablonus. Priklausomai nuo duomenų specifikos, šablonai gali susidėti iš įrašų rinkinio (klasterinė analizė), neįprastų įrašų ir

situacijų (anomalijų aptikimas) ir priklausomybių (priklausomybių taisyklių gavyba) [15]. Šiam uždaviniui spręsti dažnai yra naudojamos numatytos technikos ir analizės algoritmai. Atrasti modeliai ir šablonai rezultate gali būti reprezentuojami kaip įeities duomenų santrauka, išryškinant svarbiausias vietas. Šie duomenys vėliau gali būti naudojami tolimesnėje analizėje. Pavyzdžiui, norint atlikti prognozinę analizę, gavybos procesas duomenų rinkinyje turėtų atrasti kuo daugiau šablonų grupių, kurios galėtų padėti gauti tikslesnes prognozes [15].

Duomenų gavyba apima šešias pagrindines uždavinių klases [16]:

- **Anomalijų aptikimas**. Neįprastų duomenų įrašų identifikacija. Tokie šablonai gali būti labai svarbūs, nes parodo išskirtinius atvejus. Taip pat šiai uždavinių klasei priskiriama duomenų klaidų paieška. Sportinių įvykių analizės srityje tai gali būti tokios situacijos aikštėje, kurios normaliomis sąlygomis negalėjo atsitikti (pavyzdžiui, žaidėjas 2 kartus iš eilės ateina į aikštę prieš tai jos nepalikęs). Klaidingų duomenų rinkinių interpretavimo kaip teisingų pasėkoje gali būti padarytos klaidingos išvados apie situaciją. Todėl tam tikrose srityse klaidingų duomenų analizė gali būti kritiškai svarbi (pavyzdžiui, medicinoje).
- **Priklausomybių modeliavimas**. Ieškomos priklausomybės tarp įrašų. Sporto srityje šie uždaviniai taip pat yra svarbūs. Pavyzdžiui, krepšinio komandos "Toronto Raptors" trenerio asistentas B. James naudojo šią techniką ruošiant savo komandą rungtynėms prieš kitas NBA komandas [15]. Tokio modeliavimo panaudojimo sritys yra labai plačios (ne tik sporte). Šis būdas taip pat naudojamas didelių prekybos centrų tam, kad suprasti lankytojų įpročius ir elgseną.
- **Klasterinė analizė**. Uždavinys, kurio sprendimo metu siekiama atrasti įrašų grupes ir struktūras visame duomenų rinkinyje. Taip pat yra ieškomi panašumai tarp struktūrų, atrandamos naujos šabloninės situacijos.
- **Klasifikacija**. Tai problema, kai naujus elementus arba duomenų įrašus reikia kategorizuoti. Rezultate nauji įrašai turi būti priskiriami prie vienos iš anksčiau numatytų grupių. Tokiam klasifikavimo tikslui kiekvienas įrašas yra dalinamas į matuojamus atributus, kintamuosius, galimybes ir pan. Pavyzdžiui, ši problema yra itin svarbi elektroninio pašto veikimo principe. Pasinaudojant esamais klasifikacijos problemos sprendimais yra nusprendžiama ar naujai gautas laiškas yra brukalas (angl. *spam*).
- **Regresija**. Problema, kurios sprendimo metu bandoma atrasti funkciją arba funkcijų rinkinį, kuris geriausiai atspindėtų duomenų modelį. Geriausiais skaitomi tie rinkiniai, pasinaudojant kuriais gaunama mažiausia paklaida realių duomenų atžvilgiu. Tai viena seniausių technikų, naudojamų duomenų gavybos procesuose. Įrankiams, kurie skirti šiai analizei, paduodami duomenų rinkiniai, o jie padeda atrasti matematines formules, kurios geriausiai nusako duomenų elgseną. Pasiekus gerus rezultatus formulių paieškoje ir atsiradus naujiems įrašams, duomenys įstatomi į gautas funkcijas ir gaunamos prognozės. Tačiau ši metodika veikia gerai tik su tokiais duomenimis, kuriuos galima nesunkiai kiekybiškai išmatuoti (svoris, greitis, amžius ir pan.) [15].
- **Automatinė santrauka**. Tai uždavinys, kurio metu sprendžiama duomenų reprezentavimo problemą. Visi esantys duomenys yra sutraukiami, agreguojami ir atvaizduojami vizualiai arba ataskaitos pavidale. Pagrindinė problema sprendžiant šį uždavinį yra svarbios informacijos atrinkimas visų esančių duomenų kontekste.

Duomenų gavybos procesas gali būti panaudotas netikslingai. Net tokiu atveju jo gauti rezultatai gali atrodyti tikri ir validūs, tačiau iš tikrųjų nereprezentuos jokios realios informacijos elgsenos prognozėms atlikti. Tokiu atveju dauguma iš aukščiau aprašytų uždavinių būtų išspręsti neteisingai, o naujam duomenų rinkiniui būtų neįmanoma numatyti naudojamo algoritmo elgsenos. Dažnai neteisingą technikų naudojimą nulemia noras patikrinti per daug hipotezių ir reikiamų statistinių testų neatlikimas [17]. Todėl yra svarbu verifikuoti gautus duomenis.

Duomenys gali būti analizuojami keliuose skirtinguose lygiuose [18]:

- **Dirbtinis neuronų tinklas**. Nelinijinio tipo modeliai, kurių elgsena grindžiama "patirtimi" – ankščiau matytomis situacijomis. Šio tipo modeliai savo struktūra abstrahuoja biologinį neuronų tinklą.
- **Genetiniai algoritmai**. Optimizavimui skirtos technikos. Šie algoritmai naudoja procesus panašius į tuos, kurie vyksta gamtoje ir evoliucijoje siekiant atmesti netinkamus ir palikti

labiau priimtinus variantus. Tai tokie procesai, kaip genetinė kombinacija, mutacija, natūrali atranka ir kt.

- **Sprendimų medžiai**. Medžio formos struktūros, kurios reprezentuoja sprendimų rinkinius. Šitie sprendimai nurodo taisykles bei sąlygas, kurios naudojamos naujam duomenų rinkiniui klasifikuoti.
- **Artimiausio kaimyno metodas**. Technika, kuri klasifikuoja kiekvieną įrašą duomenų rinkinyje priklausant nuo klasių kombinacijos, kurioms buvo priskirta *k* labiausiai artimų arba panašių įrašų.
- **Duomenų vizualizacija**. Vizualinis sudėtingų sąryšių ir struktūrų interpretavimas. Atvaizdavimui naudojamos diagramos, grafų vizualinės struktūros ir kiti grafiniai elementai.

Paskutinis žingsnis, kurį reikia atlikti išgavus visą reikiamą informaciją, tai patikrinti ar proceso ir algoritmo veikimo rezultate gauti šablonai atsiranda panašiu dažnumu ir didesniame įeities duomenų kiekyje. Nevisi atrasti šablonai gali būti validūs ir informatyvus. Duomenų gavybos algoritmams yra būdinga kartais atrasti tokius šablonus, kurie bendrame visų duomenų kontekste atsiranda nedažnai [17]. Tokios situacijos vadinamos persimokymais (angl. *overfitting*). Tam, kad išspręsti šią problemą algoritmas paleidžiamas su kitais duomenų rinkiniais. Išgauti šablonai testuojami su naujais duomenimis ir gauti rezultatai lyginami su prieš tai gautais [17].

Jeigu pasitvirtina prielaida, kad gauti šablonai nėra validūs, ieškoma klaidų procesuose, kurie vyko prieš algoritmo paleidimą (pavyzdžiui, duomenų paruošimo žingsnis) ir pačioje algoritmo realizacijose. Jeigu rezultatų validacijos procesas praeina sėkmingai, tai šablonai patenka į sekančią fazę ir yra interpretuojami, siekiant padaryti konkrečias išvadas [17].

2.3. Egzistuojantys sprendimai

Šiuo metu vis daugiau aplikacijų ir sistemų migruoja į internetinę erdvę dėl akivaizdžių privalumų. Sportinių įvykių vizualizavimo sistemą patalpinus į tokią erdvę išsisprendžia visa aibė problemų, tokių kaip lengvas sistemos ir jos informacijos pasiekiamumas, paprastesnis sistemos naudojimasis, įdiegimo į lokalų kompiuterį būtinumas ir pan. Yra sukurtos tik kelios tokios sistemos internetinėje erdvėje, tačiau jos visapusiškai neapima kuriamos sistemos vizijos ir dažnai yra skirtos tik patogiam konkrečių įvykių atvaizdavimui.

Šiame skyriuje bus apžvelgtos kelios iš tipinių ir charakteringųjų sistemų, skirtų sportinių įvykių registravimui ir jų analizei. Apžvelgiant esamas sistemas, bandoma įžvelgti jose panašumų ir bendrų naudingų savybių. Taip siekiama išryškinti svarbiausias tokių sistemų savybes, o galbūt ir pastebėti jose trūkumų.

2.3.1. Dartfish.com įvykių registravimo ir analizės sistema

Šis programų paketas leidžia registruoti įvykius realiuoju laiku, peržiūrėti jų statistiką, analizuoti žaidimą. Visa sistema dalinama į atskiras posistemes, kurios atlieka tam tikras funkcijas. Pavyzdžiui, pasinaudojant EasyTag posisteme leidžiama registruoti įvykius realiu laiku [19]. Taip pat yra sukurtas įrankis, leidžiantis registruoti naujus įvykius išmaniųjų telefonų pagalba. Įvykių registravimo modulis mobiliesiems telefonams buvo kuriamas taip, kad [20]:

- įvykius būtų galima registruoti taip, kad dėmesys liktų sukoncentruotas į žaidimą, o ne programos naudojimą;
- išsaugojus pastabas apie rungtynes jas galima būtų pasiekti lengvai ir greitai;
- būtų galimybė įvykius susinchronizuoti su vykusiu žaidimu, taip palengvinant jų vėlesnę analizę.

Vėliau, pasinaudojant kitais moduliais, šiuos įvykius galima analizuoti. Skirtingi PĮ paketai turi kiek skirtingas galimybes. Visas analizės procesas yra labai glaudžiai susijęs betarpiškai su rungtynių vaizdo įrašu. Sistema leidžia analizuoti įrašus, kuriuose tam tikrais laiko momentais yra pažymėti užregistruoti įvykiai. Operuojant žymėmis vaizdo įraše yra galimybė atlikti įvairias funkcijas, tarp kurių yra tokios kaip:

- atskirų laiko momentų atkartojimas;
- vaizdo arba tam tikro įvykio įrašo sukarpymas kadrais, siekiant pamatyti įvairias žaidimo smulkmenas ir subtilybes;

- galimybė piešti virš vaizdo įrašo;
- atskirų objektų atsekimas (naudinga, pavyzdžiui, stebint kamuolio trajektorijas ir pan.);
- analizuojant žaidėjų judesius atrasti tam tikras šablonines elgsenas;
- eksportuoti/importuoti duomenis į CSV formatą;
- ir kt.

Sistema naudoja pažangias technologijas žaidėjų judesiams išryškinti ir analizuoti, tačiau ji sukoncentruota į pačių žaidėjų technikos analizę, o ne į bendrą įvykių visumą.

2.3.2. Sportsec SportsCode

Sistema, leidžianti stebėti ir analizuoti sportinius įvykius realiuoju laiku. Peržiūrint rungtynes arba jų vaizdo įrašą yra galimybė išsaugoti tam tikrus įvykius, susieti juos su laiko juosta ir esant reikalui greitai pasukti vaizdo įrašą į reikiamą įvykį. Taip pat yra matomas bendras rungtynių vaizdas. Greitai veikianti sistema leidžia treneriams nedelsiant priimti efektyvius sprendimus dėl tolimesnio žaidimo.

Programinė įranga yra nesunkiai plečiama – yra galimybė rašyti savo papildomus skriptus, aprašant naujas, sistemai nežinomas sporto šakas ir jų taisyklių rinkinius. Taip pat integruotas lankstus vartotojo sąsajos valdymo įrankis.

Dirbant su vaizdo įrašu ir pildant įvykius programa siūlo tokias papildomas galimybes [21]:

- sukurti tekstinius komentarus virš vaizdo nurodytais laiko momentais;
- eksportuoti darbo rezultatus į duomenų bazę arba sukurti naują duomenų bazę, susidedančią iš užregistruotų įvykių;
- peržiūrėti scenarizuotą video (scenarijai gali būti kuriami įvairiais būdais);
- ir kt.

Programa leidžia eksportuoti tam tikrus vaizdo įrašo fragmentus į vieną arba atskirus video failus, jais pasidalinti su norimais asmenimis. Taip pat yra galimybė eksportuoti darbo rezultatus (įskaitant statistiką, patį vaizdo įrašą ir pan.) į naršyklėms suprantamą formatą. Tokiu būdu atsiranda galimybė patogiai eksportuoti duomenis rezultato talpinimui ir internete. Tačiau šita galimybė neleidžia realiu laiku stebėti įvykių internetinėje erdvėje.

Pagrindinis programos darbo langas pavaizduotas 2 priede [21].

2.3.3. StatTrak for Basketball

Programa skirta tik krepšinio įvykių registravimui ir peržiūrai. Registruojant tam tikrus įvykius, programa automatiškai paskaičiuoja statistinius duomenis ir atvaizduoja rezultatą. Leidžiama skaičiuoti atskiro žaidėjo, visos komandos arba čempionato statistiką pagal įvairius kriterijus bei filtrus. Taip pat turima galimybė pasirinkti kokius tiksliai duomenis atvaizduoti. Programos langas, atvaizduojantis žaidėjų statistiką pavaizduotas 3 priede [9].

Kaip galima matyti iš paveikslo, programa pateikia “sausą” statistiką, be jokių grafikų ar kito vizualinio pateikimo. Tokio trūkumo pasekmė yra ta, kad naudojimasis programa gali tapti labai nuobodus, reikalaujantis didesnio laiko tam, kad atrasti ir suprasti dominančią informaciją. Programoje yra integruoti iš anksto aprašyti kriterijai, pagal kuriuos galima generuoti ataskaitas, o taip pat jas eksportuoti į skirtingus formatus (pavyzdžiui, į Word failą).

Iš tikrųjų programa tik gauna vartotojų įvestus duomenis ir pagal nesudėtingas matematines formules apskaičiuoja kelias statistines reikšmes. Naudingumo atžvilgiu ji yra mažiau patrauklesnė, nes naudojama tik sumarinė įvykių statistika, kuri nepadeda padaryti išvadų apie bendrą komandos elgseną ir joje esamas problemas, bandant patobulinti arba sukurti naują, efektyvesnę žaidimo strategiją.

2.3.4. SportsML sporto įvykių žymėjimo standartas

SportsML – tai naujienų apsikeitimų standartas, paremtas XML žymėjimo kalba, sukurtas 2001 metais. Jis buvo kuriamas siekiant supaprastinti ir optimizuoti apsikeitimą sportiniais duomenimis (tokiais, kaip rungtynių rezultatai, bendra statistika, atskirų žaidėjų statistika ir pan.) [22, 23].

Šio duomenų formato galimybės leidžia į jo numatytą struktūrą patalpinti pakankamai duomenų, kad tiksliai aprašyti beveik bet kurios sporto šakos įvykius ir statistiką. Tačiau, tam tikrais atvejais, žaidimams su sudėtingesnėmis taisyklėmis ir platesne statistikos aibe kuriami atskiri šio standarto plėtiniai, pritaikyti prie konkrečios sporto šakos [23].

Laikui bėgant standartas buvo plečiamas, kol galiausiai buvo išleista patobulinta SportsML versija, pavadinta SportsML-G2, kuri yra remiama kompanijos IPTS [22]. Naujas standartas yra pilnai suderinamas su IPTC G2 šeimos standartais. Tokie standartai remiasi panašia XML blokų struktūra, kas supaprastina darbą su šiuo ir kitais šios šeimos standartais (pvz. NewsML-G2, EventsML-G2 ir kt.). Šis duomenų formatas yra tebevystomas iki šiol – tobulinama vidinė struktūra ir dokumentacija, kuriami nauji įskiepiai. Paprastumo dėlei šį formatą toliau vadinsime tiesiog SportsML.

Pagrindinės SportsML sudedamosios dalys yra [24]:

- *Taškai*. Nurodo kas laimi bei kaip keitėsi taškų skirtumas.
- *Tvarkaraščiai*. Atsako į tokius klausimus – kas su kuo, kur ir kada žaidžia.
- *Turnyrinė lentelė*. Nusako kas kokią vietą užimą, kas yra arčiausiai tam tikro tikslo.
- *Statistika*. Parodo kaip žaidėjai, komandos bei rungtynės yra matuojamos tarpusavyje, turint omenyje skirtingus kriterijus bei jų kategorijas.
- *Žinios*. Nurodo kaip reikia žiūrėti į visą aprašytą duomenų visumą, kaip siejasi aprašyti duomenys su papildomais duomenimis (angl. *metadata*) arba multimedijos failais ir kitais šaltiniais.

Sporto įvykių grupavimas yra atskiriamas į 3 dideles grupes – komandinis sportas, individualus sportas ir čempionatai. Kiekvienai iš šių grupių aprašyti naudojamos skirtingos SportsML žymės. Sekančiame pavyzdyje parodyta futbolo komandos aprašymo struktūra.

```
<team id="manutd">
   <team-metadata>
      <name first="Manchester" last="United" full="Manchester United"/>
   </team-metadata>
   <team-stats score="1" event-outcome="loss">
      <sub-score score="1" period-value="1"/>
      <sub-score score="0" period-value="2"/>
   </team-stats>
   <player id="manutd10">
      <player-metadata uniform-number="10" nationality="England">
         <name first="Wayne" last="Rooney" full="Wayne Rooney" />
      </player-metadata>
      <player-stats score="1">
         <sub-score score="1" period-value="1" />
      </player-stats>
   </player>
</team>
```

1 pavyzdys. Komandos struktūros aprašymas

Tokiu būdu aprašoma visa komanda bei jos dalyviai. Tolimesniuose aprašymuose, pavyzdžiui, įvykių ir statistikos blokuose, naudojami komandoms bei žaidėjams suteikti identifikatoriai (atributai *id*).

Šis žymėjimo standartas sprendžia visą aibę aktualių problemų, susijusių su sportinių duomenų apsikeitimu ir transportavimu. Tačiau yra ir kita probleminė sritis – šių duomenų agregavimas bei analizė. Kadangi SportsML – tai XML žymėjimo kalbos plėtinys, duomenų iš jo išrinkimui galima taikyti tokias technologijas, kaip XPath, MOXy ir kt. Tokiu atveju atsiranda galimybė išgauti atskirus duomenų rinkinius, susijusius su norimais įvykiais. Tačiau šis duomenų analizės būdas neišsprendžia pagrindinės šiame darbe nagrinėjamos problemos – sportinių įvykių tarpusavio sąryšių analizė, interpretuojant įvykių visumą kaip atskiras sekas.

2.3.5. Duomenų gavybai skirti algoritmai

Kiekvienai iš šio darbo *2.2. Duomenų gavyba* skyriuje apžvelgtų uždavinių klasių problemoms spręsti egzistuoja visa eilė įvairių algoritmų [25]. Duomenų gavyba – tai labai bendras terminas ir nenurodo konkrečių veiksmų sekų arba problemų sprendimo būdų. Tai tik bendrai aprašytos problemos bei uždaviniai. Kiekviena situacija, kai reikia analizuoti didelius duomenų kiekius, gali būti išsprendžiama įvairiai bei sukuriant savo algoritmą, kuris būtų geriausiai pritaikytas prie konkrečios situacijos. Tačiau yra išskiriamos bendros šiai problemų sričiai uždavinių klasės. Kadangi kiekviena problemų klasė sprendžia skirtingas problemas, tai ir algoritmai ir jų veikimo principai ženkliai skiriasi. Todėl nėra sukurto universalaus algoritmo, kuris spręstų visas su duomenų gavyba susijusias problemas ir uždavinius. Pilnas populiariausių algoritmų žemėlapis pateiktas šio darbo 1 priede.

Šiame skyriuje bus apžvelgti kelių problemų klasių algoritmai.

2.3.5.1. Dažnai pasitaikančių šablonų gavyba

Dažnai pasitaikančių šablonų gavyba (angl. *Frequent Pattern Mining*) – tai uždavinys, kurio metu gaunami statistiškai dažniausiai pasitaikantys šablonai. Paieška vykdoma tarp iš anksto paruoštų įrašų, kurie pasižymi tam tikromis savybėmis [26]:

- *Eiliškumas*. Turi būti galimybė visus įrašus surūšiuoti į vieną nedalomą seką.
- *Diskretumas*. Turi būti galimybė įrašą suvesti į diskrečią reikšmę, kuri padėtų atskirti skirtingus įrašų tipus.

Vienas iš algoritmų, kuris atlieka šią paiešką yra **SPADE** algoritmas (angl. *Sequential PAttern Discovery using Equivalence classes*) [27]. Jis yra plačiai naudojamas įvairiuose srityse, pastaruoju metu ypač aktyviai biologijoje. Pasinaudojant SPADE ir kitais panašiais algoritmais gaunama DNR informacija, kurios plika akimi pastebėti beveik neįmanoma. Dėl genetinių mutacijų ir evoliucijos DNR sekų duomenų bazės yra labai didelės, todėl dažnai pasitaikančių šablonų gavybos procesas gali padėti išryškinti šablonus, kuriuos būtų verta naudoti genetiniuose eksperimentuose.

Šį algoritmą taip pat būtų galima pritaikyti ir sportinių įvykių sekų analizei. Jo veikimo rezultatas parodytų kokios situacijos aikštelėje yra populiariausios (t.y. išryškintų įvykių sekas, kurios rungtynių eigoje atsiranda dažniausiai). Vėliau, analizuojant kitas rungtynes ir ieškant jose anksčiau rastų šabloninių situacijų, galima daryti tam tikras išvadas apie rungtynių rezultatą.

Algoritmo veikimas grindžiamas vertikaliu identifikatorių reikšmių sąrašu, kuriame kiekvienai iš egzistuojančių sekų priskiriamas objektų sąrašas, kuriame ji egzistuoja. Tada dažnai pasitaikančios sekos gali būti nesunkiai išryškintos pasinaudojant gautų identifikatorių sąrašų aibių sankirta [27]. Šis metodas taip pat sumažina reikalingų duomenų bazės užklausų kiekį, palyginus su kitais algoritmais, todėl sumažėja ir jo vykdymo laikas.

Pirmas žingsnis algoritmo veikime yra paskaičiuoti sekų, susidedančių tik iš vieno elemento, dažnumą. T.y. suskaičiuojama kiek kartu koks įvykis įvyko. Tas įvykdoma vos su viena duomenų bazės užklausa kiekvienam duomenų rinkiniui, grupuojant įrašus pagal reikiamus atributus.

Antrame žingsnyje skaičiuojamos sekos, susidedančios iš 2 elementų. Vertikali lentelės reprezentacija transformuojamą į horizontalią, skaičiuojama kiek kartų kiekviena elementų pora (viena po kitos) atsiranda visoje elementų sekoje. Šis žingsnis taip pat gali būti įvykdytas pasinaudojant tik viena duomenų bazės užklausa kiekvienai grupei.

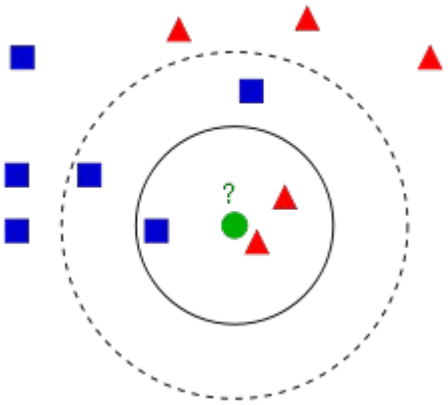
Ilgio *n* dažniausiai pasitaikančios sekos randamos apjungiant ilgio *n-1* sekų identifikatorių sąrašus. Identifikatorių sąrašo ilgis nurodo sekų, kurioje elementas yra randamas, skaičių. Jeigu šis skaičius yra didesnis, negu ilgio charakteristika kitoms sekoms, tai seka laikoma dažnai pasitaikančia. Algoritmas vyksta tol, kol randamos dažnai pasitaikančios sekos.

2.3.5.2. k artimiausių kaimynų algoritmas

k artimiausių kaimynų algoritmas (angl. *k Nearest Neighbours algorithm*, toliau – kNN) duomenų gavybos srityje naudojamas iškarto dvejose problemų klasėse – klasifikacijos ir regresijos. Šiame skyriuje bus apžvelgiamos algoritmo siūlomi klasifikacijos problemos sprendimo būdai. kNN

algoritmas nedaro jokių naujų prielaidų iš gautų įeities duomenų. Vietoje to nauji duomenys yra lyginami su jau esamais duomenimis, siekiant suklasifikuoti naują įrašą [28].

Naujo elemento klasifikacija vyksta matuojant šalia esančių elementų klasifikacijų dažnumą. Įrašas klasifikuojamas į tokią grupę, į kurią yra priskirta daugiausiai elemento kaimyninių įrašų. Kiek šalia esančių įrašų dalyvauja klasifikavimo procese nusako kintamasis *k*, kuris yra sveikas skaičius, dažniausiai nedidelis [29]. Jeigu *k = 1*, tai naujas įrašas priskiriamas tai grupei, kuriai priskirtas arčiausiai jo esantis elementas.



4 pav. kNN veikimo principas. Autorius: Antti Ajanki

Paveiksle X parodytas tokios klasifikacijos pavyzdys. Naujas elementas (žalias skritulys) turi būti klasifikuotas į vieną iš grupių – mėlyni kvadratai arba raudoni trikampiai. Jeigu *k = 3* (vertinimui naudojami 3 artimiausi kaimynai – vientisa linija), tai skritulys bus priskirtas raudonų trikampių grupei (šalia yra 2 trikampiai ir 1 keturkampis). Jeigu *k = 5* (vertinimui naudojami 5 artimiausi kaimynai – brūkšninė linija), tai naujas elementas bus priskirtas mėlynų kvadratų grupei, nes iš 5 arčiausiai esančių elementų yra 3 kvadratai ir tik 2 trikampiai.

Sprendžiant skirtingas problemas, priklausomai nuo problemos tipo, gali būti renkamos skirtingos klasifikavimo metrikos ir kintamasis *k*. Tokiems duomenims, kurie modeliuojami trimatėje erdvėje dažniausiai naudojama metrika – Euklido atstumas (atstumas tarp taškų arba vektorių trimatėje erdvėje) [28]. Specifinės metrikos yra dažnai naudingos specifinėms uždaviniams, pavyzdžiui – teksto klasifikacijai.

Siekiant išvengti kolizijų (kai vienodas išrinktų kaimynų skaičius priklauso kelioms grupėms), dažniausiai imama nelyginė *k* reikšmė. Šis faktas ypač svarbus, kai naudojamas paskirstymas tik į 2 grupes, nes tokiomis sąlygomis yra didelė kolizijų tikimybė. Taip pat yra galimybė įvairiems elementams nustatyti įvairius svorius. Pavyzdžiui, jeigu elementas *x* yra svarbesnis, negu kiti šalia esantys, tai jis gali duoti didesnį svorį įvertinimui, negu keli elementai iš kitos grupės. Dažnai naudojama technika, kurios metu yra vertinamas pats atstumas tarp skirtingų elementų. Kuo atstumas iki vertinamojo elemento yra didesnis, tuo mažesnį svorį turi tas kaimynas.

Pagrindiniai kNN algoritmo privalumai [29]:

- Nesudėtinga implementacija.
- Lankstumas paieškos dydžio atžvilgiu. Pavyzdžiui, klasės neturi būti logiškai susiję, jų loginę prasmę nustato eksperimento sąlygos.
- Klasifikatoriai gali būti lengvai atnaujinami kai imtyje atsiranda nauji elementai. Ši operacija reikalauja labai mažai resursų.
- Labai mažai parametrų: distancijos sąvoka ir *k* reikšmė.

Pagrindiniai kNN trūkumai [29]:

- Resursų atžvilgiu brangus kiekvieno elemento testavimas, nes reikia skaičiuoti atstumus tarp visų imtyje esančių elementų. Šiai problemai išspręsti yra sukurti algoritmo patobulinimai. Kai elementų skaičius yra labai didelis gali būti naudojamos specialios struktūros informacijai apie elementus saugoti (pavyzdžiui, R-medžiai ir kd-medžiai).
- Algoritmas yra jautrus iš anksto neaprašytiems ir nenumatytiems atributams, kas gali sukelti problemų, kai bandoma suskaičiuoti loginį atstumą tarp elementų.
- Taip pat algoritmas yra jautrus nesubalansuotiems duomenims, kai įrašai vienu metu gali priklausyti kelioms grupėms, arba kai nedažnai pasitaikančios klasės dominuoja daugumoje kaimynysčių.

2.4. Projekto tikslas ir funkcionalumas

Projektinės dalies tikslas yra sukurti sporto įvykių vizualaus pateikimo ir analizės programinę įrangą, veikiančią internetinėje erdvėje, kurios pagalba būtų galima pamatyti ne tik trivialią rungtynių statistiką, bet ir padaryti iš jos sekančias išvadas, įžvelgti tam tikras tendencijas,

priklausomybes ir šablonines elgsenas sportinių varžybų aplinkoje. Sistema turės būti lengvai plečiama, norint papildyti ją naujomis sporto šakomis, jų taisyklių aprašymu, statistikos skaičiavimo metodologijomis, šabloninės elgsenos aprašais ir kitais skirtingoms sporto šakoms svarbiais aspektais.

Kuriama sportinių įvykių analizės ir vizualizavimo sistema, pavadinimu SEA, kurios tikslas yra spręsti tam tikras problemas, susijusias su sportinių įvykių analize. Sistemos funkcijos yra tiesiogiai susijusios su sporto įvykių analize ir vizualizavimo rezultatais, taip pat su suinteresuotų asmenų poreikiais ir tikslais. Pagrindinės sistemos funkcijos:

1. Rungtynių sąrašo gavimas iš išorinių šaltinių;
2. Rungtynių registravimas sistemoje bei jų įvykių gavimas iš išorinių šaltinių;
3. Komandų ir žaidėjų sąrašu gavimas iš išorinių šaltinių ir registravimas lokalioje duomenų bazėje;
4. Gautų įvykių interpretavimas, agregavimas, analizė bei išsaugojimas;
5. Rungtynių grupavimas bei atvaizdavimas galutiniam vartotojui;
6. Žaidėjų paskirstymas į reikiamas komandas, jų grupavimas bei atvaizdavimas komandų ir žaidėjų sąrašuose;
7. Rungtynių įvykių atvaizdavimas;
8. Rungtynių įvykių analizė ir ja grindžiami sprendimai, filtravimas, vizualizacija;
9. ir kt.

Nors pirma sistemos versija bus pritaikyta darbui tik su NBA krepšinio čempionato įvykiais, ją nesunkiai galima praplėsti bet kuriam kitam čempionatui ir duomenų šaltiniui arba net kitai sporto šakai. Taip pat sudarytos patogios sąlygos duomenų agregavimui, todėl norint praplėsti sistemos funkcionalumą, užteks tiesiog aprašyti papildomą reikiamų sistemos vaizdų elgseną bei logiką.

2.4.1. Numatomi analizės galimybių patobulinimai

Šio darbo kontekste nagrinėjamos galimybės, kaip būtų galima praplėsti kuriamos sistemą funkcionalumą taip, kad įvykių sąryšių analizė bei šabloninių situacijų išryškinimas taptų dar efektyvesni.

Pasinaudojant šio darbo skyriuje *2.2. Duomenų gavyba* aprašytomis technikomis bei efektyviau išryškinant ir pateikiant šabloninių situacijų suvestinę atsiranda galimybė smarkiai pagerinti kuriamos SEA sistemos įvykių analizavimo gebėjimus.

3. PROJEKTINĖ DALIS

3.1. Kūrimo technologijos, įrankiai ir karkasas

Vienas iš pagrindinių sistemos nefunkcinių reikalavimų – sistema turi veikti internetinėje erdvėje. Todėl serverinės sistemos pusės programavimui buvo pasirinktos šiuo metu vienos populiariausių web programavimo technologijų – *PHP* programavimo kalba ir *MySQL* duomenų bazių valdymo sistema.

Projektavimo metu buvo numatyta nemaža sistema, todėl pasirodė tikslinga panaudoti vieną iš šiai programavimo kalbai sukurtų karkasų[5]. Buvo atlikta rinkoje esančių karkasų apžvalga. Darbui buvo pasirinktas *CakePHP* (http://www.cakephp.org/) karkasas, kuris projekto kūrėjo nuomone yra labiausiai tinkamas sistemai, užtikrinant jos greitaveiką bei plečiamumą.

3.2. Realizacija bei esminiai techniniai sprendimai

Projektas buvo vystomas iteratyviniu būdu. Prieš ir po kiekvienos iteracijos, vykstant susitikimams su užsakovu, buvo derinamas sekančios iteracijos darbų sąrašas. Šiame skyriuje bus aprašytos atskiros iteracijos bei jose priimti esminiai techniniai bei projektiniai sprendimai.

3.2.1. Pirma iteracija

3.2.1.1. Aprašymas ir užduotys

Pirma iteracija skirta tam, kad susipažinti su kuriamos sistemos specifika, išsiaiškinti pagrindinius reikalavimus bei paruošti programinę projekto bazę tolimesniam kūrimui.
Išsiaiškinti funkciniai reikalavimai:

- sistema turi gebėti išgauti krepšinio rungtynių įvykių duomenis iš išorinių sistemų;
- sistema turi atskirti skirtingus įvykių tipus;
- sistema turi išsaugoti struktūrizuotus duomenis į lokalią duomenų bazę;
- sistema turi mokėti atvaizduoti sukauptus duomenis grafiškai.

Reikalavimų išsiaiškinimo metu buvo išskirtos tokios iteracijos užduotys ir tikslai:

1. Paruošti projekto bazę – sukonfigūruoti karkasą bei paruošti jį darbui.
2. Paruošti duomenų bazės struktūrą įvykių ir kitų reikalingų duomenų saugojimui (sistema turi būti pritaikyta krepšinio duomenų saugojimui).
3. Sukurti komponento, kuris gebėtų išgauti įvairių sportinių įvykių duomenis iš išorinių šaltinių, bazinį variantą.
4. Apsispręsti dėl naudojamų šaltinių.
5. Realizuoti komunikavimą su vienu iš duomenų šaltinių.
6. Grafiškai atvaizduoti sukauptus duomenis, taip parodant, jog sistema geba operuoti ką tik išgautais duomenimis.

3.2.1.2. Realizacija ir architektūriniai sprendimai

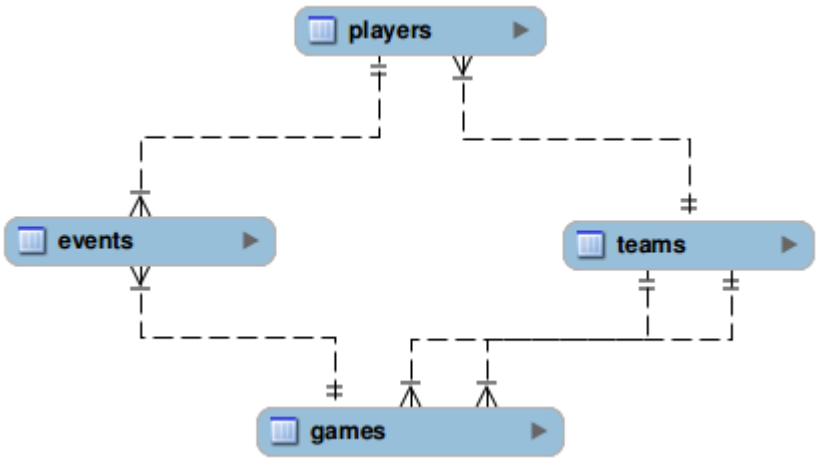
Visų pirmą buvo įdiegtas, sukonfigūruotas ir paruoštas darbui CakePHP karkasas – sukurta duomenų bazė, nustatyti prisijungimo duomenys bei teisės.

Sekantis žingsnis – duomenų bazės struktūros kūrimas. Apžvelgiant reikalavimus bei turint omeny krepšinio žaidimo specifiką, išsiskiria akivaizdžios sistemos esybės – Žaidėjas, Komanda, Rungtynės ir Įvykis. Taikomi tokie esybių sąryšiai:

- žaidėjai priklauso komandai;
- rungtynės vyksta tarp dviejų komandų (rungtynės turi nuorodas į dvi komandas);
- įvyki sukuria žaidėjas tam tikru rungtynių momentu.

Bendras esybių sąryšių vaizdas pavaizduotas 5 paveiksle

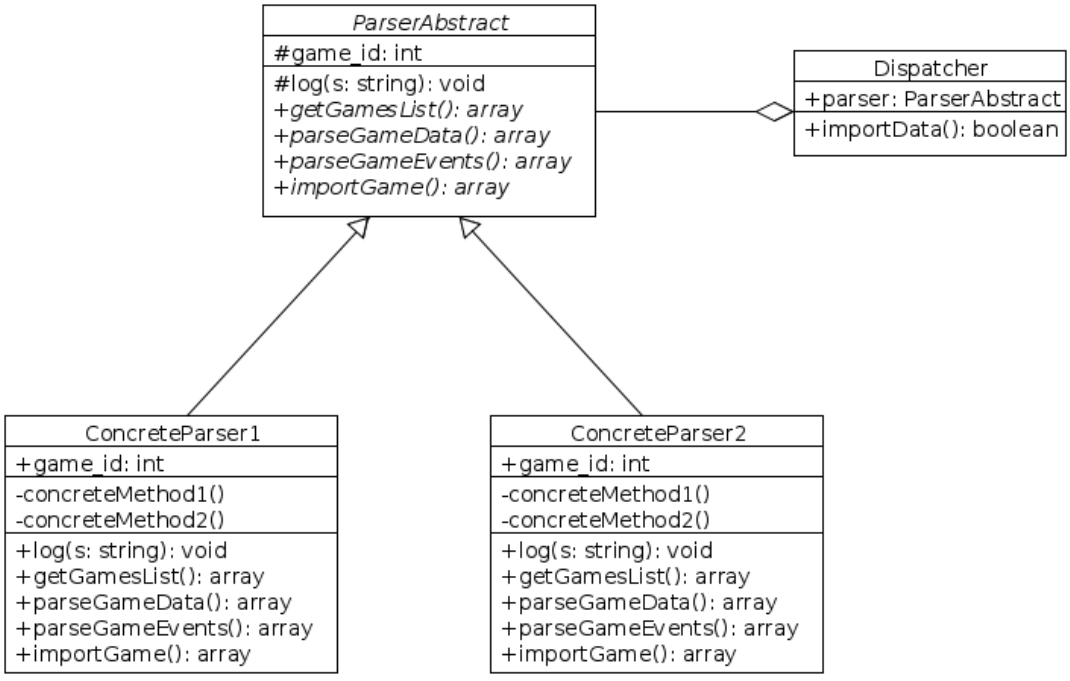
[5] PĮ karkasas (angl. *Software framework*) – programų sistemos struktūra, palengvinanti didesnių sistemų kūrimą bei jos modulių ir komponentų apjungimą tarpusavyje.



5 pav. Esybių sąryšiai

Į tą faktą, kad komponentas, užpildantis duomenų bazę iš išorinių šaltinių turės būti pritaikytas daugiau negu vienam šaltiniui, buvo atsižvelgta kuriant jo architektūrą. Sukurtas abstraktus duomenų iš šaltinio gavimo aprašas – klasė *ParserAbstract*, kurią privalės paveldėti kitos, jau konkrečiam šaltiniui pritaikytos klasės. Vaikinės klasės taip pat privalės aprašyti metodus *getGamesList()*, *parseGameData()*, *parseGameEvents()* bei *importGame()*. Taip siekiama suabstrahuoti duomenų paėmimo iš šaltinio ir išsaugojimo į lokalią duomenų bazę logiką. Taip pat naudojama pagrindinė klasė *Dispatcher*, kuri sukuria reikiamus klasės objektus, kviečia metodus reikiamu eiliškumu ir kitaip valdo visą duomenų importavimo proceso eigą.

Komponento klasių diagrama pavaizduota 6 paveiksle.



6 pav. Šaltinių duomenų paėmimo ir saugojimo komponento architektūra

Pradiniams duomenims gauti buvo pasirinktas šaltinis NBA.com. Šiuo konkrečiu atveju oficialiai nėra teikiama specialiai suformatuotų ir struktūrizuotų varžybų įvykių (pvz. XML formatu).

Todėl, norint gauti duomenis iš šio šaltinio yra tikslinga pasinaudoti *web scraping* technologija[6]. Tam, kad būtų patogiau išgauti duomenis iš tinklapio išeities kodo buvo integruota bei minimaliai pakeista *SimpleHtmlDom*[7] PHP biblioteka. Šios iteracijos ribose komponentas iš NBA.com duomenų sugeba išgauti tokius duomenis iš kiekvienų vykusių rungtynių: sėkmingi baudos/dvitaškių/tritaškių metimai, prasižengimai bei bendra rungtynių statistika.

Pasinaudojant tuo pačiu CakePHP karkasu sukurtas tinklapis, vizualizuojantis duomenų bazėje esančius duomenis. Realizuotos galimybės matyti visų sistemoje registruotų komandų, žaidėjų sąrašus, taip pat peržvelgti kiekvieno žaidėjo statistiką, pamatyti komandos sužaistas rungtynes bei jų rezultatus.

3.2.2. Antra iteracija

3.2.2.1. Aprašymas ir užduotys

Antrojoje iteracijoje tęsiamas darbas prie išorinių duomenų gavimo ir analizavimo komponento – tobulinamos jo savybės ir plečiamas funkcionalumas. Taip pat komponentas turi būti pilnai izoliuotas nuo likusios sistemos ir atlikti tik jam paskirtas užduotis vienodai ir nepriklausomai nuo karkaso aplinkos (ar tai valdiklio (angl. *controller*) kontekstas, ar konsolinės aplikacijos terpė). Duomenų vizualus pateikimas irgi turės būti tobulinamas, kai kurie duomenis turi būti atvaizduojami kitais pjūviais. Antroje iteracijoje bus įgyvendinami tokie reikalavimai:

- išorinių duomenų analizatoriaus komponentas turi gebėti parinkti visus nba.com šaltinio duomenis apie įvykius, juos struktūrizuoti ir išsaugoti;
- komponento veikimas turi būti izoliuotas nuo likusios sistemos, pasižymėti perpanaudojamumo savybėmis;
- sistema turi turėti galimybę, pasinaudojus komponento teikiamais privalumais, patogiai išgauti ir išanalizuoti dominančius šaltinius;
- duomenys turi būti saugomi tokioje struktūroje, iš kurios būtų galima atstatyti jų eiliškumą žaidimo atžvilgiu;
- iš išsaugotų rungtynių duomenų atstatyti ir vizualizuoti konkrečių žaidimų įvykių seką. Visų rungtynių chronologinė įvykių sekos[8] duomenys turi būti vaizduojami laiko juostoje.

Analizuojant gautus reikalavimus buvo išskirtos tokios užduotys šiai iteracijai:

1. Tobulinti duomenų gavimo komponento funkcionalumą – išplėsti gaunamų NBA rungtynių įvykių aibę. Turi būti išgaunami tokių įvykių duomenys:
    a. sėkmingi ir nesėkmingi dvitaškių metimai;
    b. sėkmingi ir nesėkmingi tritaškių metimai;
    c. sėkmingi ir nesėkmingi baudos metimai;
    d. kamuolių perėmimai;
    e. žaidėjų pakeitimai (atskirti kuris žaidėjas palieka aikštę ir kuris išeina);
    f. prasižengimai ir klaidos.
2. Izoliuoti komponento veikimą į CakePHP valdiklio komponentą. Taip jis galės būti naudojamas bet kuriame sistemos kontekste.
3. Sukurti CakePHP konsolinį įrankį, kurio pagalba būtų galima patogiai išgauti ir išsaugoti rungtynių duomenis. Įrankio veikimo scenarijus:
    a. Vartotojas pasirenka norimą analizatorių iš užregistruotų sistemoje (NBA, Eurolygos ar kiti šaltiniai);
    b. Įvedama pradinė data, nuo kurios bus ieškomos rungtynės;
    c. Įvedama galinė data, iki kurios bus ieškomos rungtynės;
    d. Paklausiama, ar vartotojas nori papildyti duomenų bazę kaupiamais duomenimis, ar ją pravalyti ir užpildyti kaupiamais duomenimis;
    e. Programa suranda visas rungtynes duotame laiko intervale ir išsaugo jų ir jų įvykių duomenis duomenų bazėje.

[6] *Web scraping* – technika, leidžianti išgauti duomenis iš tinklapių HTML išeities kodų.

[7] *SimpleHtmlDom* biblioteka – http://sourceforge.net/projects/simplehtmldom/

[8] Chronologinė įvykių seka (angl. *play-by-play*) – detalizuota ir eiliškumu pasižyminti informacija apie sportinius įvykius

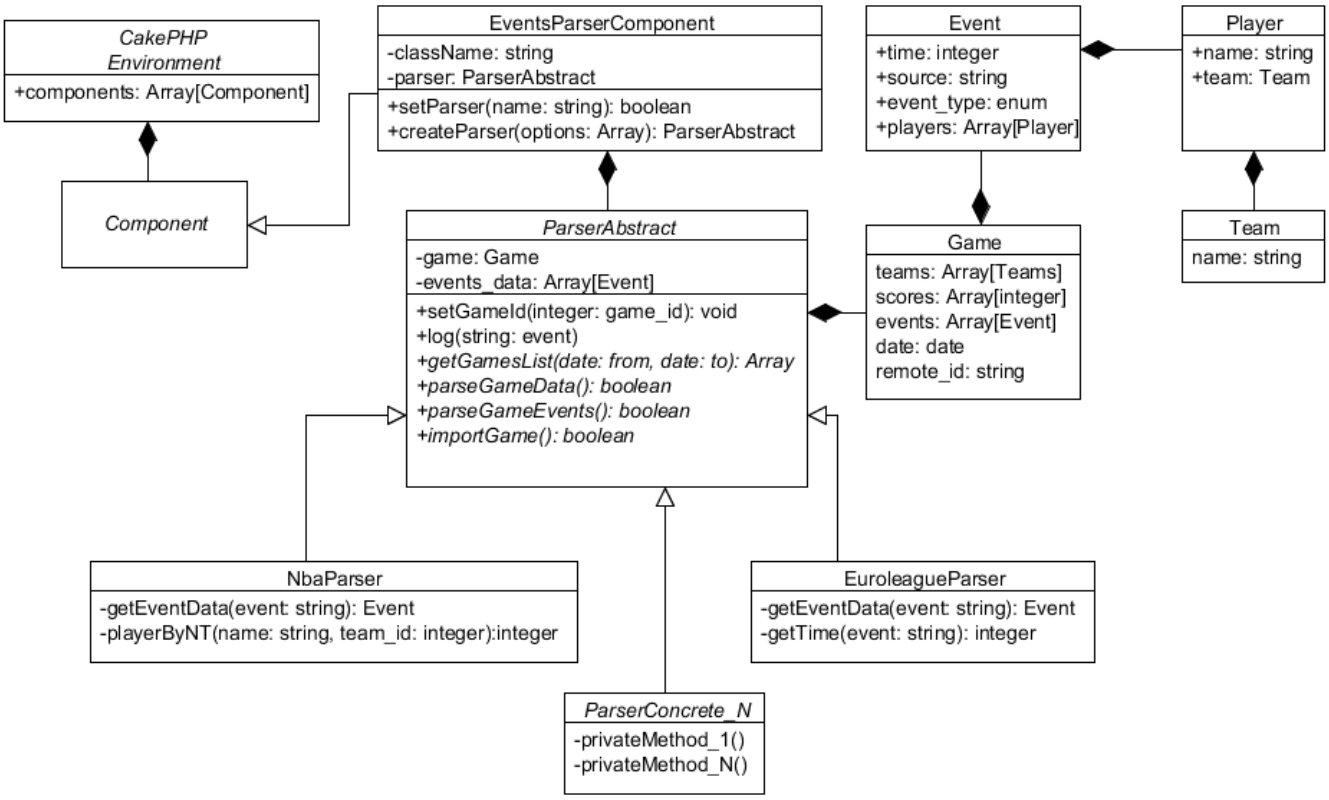
4. Patobulinti įvykio laiko momento rungtynėse apskaičiavimą, priklausomai nuo einamojo kėlinio ar kitų faktorių, galinčių įtakoti įvykių eiliškumą;
5. Realizuoti chronologinės įvykių sekos duomenų atvaizdavimą laiko ašyje.

3.2.2.2. Realizacija ir architektūriniai sprendimai

Darbas prasidėjo nuo to, kad buvo plečiama NBA gaunamų įvykių aibė. Su šia užduotimi kilo nemažai keblumų, nes atsirado sudėtinio įvykio samprata. Tai tokie įvykiai, kuriuose dalyvauja daugiau, negu vienas veikėjas. Pavyzdžiui žaidėjų apsikeitimai – vienas įvykis, bet tuo pačiu metu vienas žaidėjas palieka aikštę, o kitas išeina. Po apmąstymų ir šio fakto svarbos analizės buvo nuspręsta tokį įvykį tiesiog išskaidyti į du skirtingus įvykius (įvykis "palieka aikštę" ir "išeina į aikštę") ir jiems nurodyti tą patį rungtynių laiko momentą. Toks sprendimas leistų tam tikra prasme apjungti šiuos įvykius į vieną, bet tuo pačiu metu ir atskirti kuris žaidėjas palieka aikštę, o kuris į ją išeina.

Duomenų bazės architektūra šiam reikalavimui keistis neturėtų, tačiau iš analizatorių komponento architektūros pusės tam tikrų pakeitimų prireiks. NBA analizatoriaus metodas *getEventData()*, kuris priima įvykio originalią eilutę ir iš jos išgauna struktūrizuotą įvykį nuo šiol turės galimybę grąžinti ne vieną įvykį, o jų aibę. Taigi, NBA rungtynių analizatoriuje minėtas metodas perrašomas bei papildomas reguliariomis išraiškomis, kurios leis išgauti nesėkmingų metimų, kamuolio perėmimų, baudų ir žaidėjų apsikeitimų įvykius.

Toliau visa ši įvykių išgavimo ir analizės logika yra izoliuojama į atskirą CakePHP komponentą. Komponentas projektuojamas ir kuriamas taip, kad būtų patogu ir lengva jį plėsti, pritaikyti konkretiems atvejams bei iš esmės skirtingiems šaltiniams. Naudojamas *Strategy* projektavimo šablonas, kas užtikrina lengvą tolimesnį komponento vystymą ir plečiamumą. Komponento klasių diagrama pavaizduota 7 paveiksle.



7 pav. Duomenų gavėjo komponento architektūra

Po to, kai komponentas tampa prieinamas ir gali būti perpanaudotas bet kurioje sistemos vietoje, pradedamas kurti konsolinis analizės įrankis. Tam tikslui sukuriama nauja konsolinė CakePHP programa (arba karkaso terminais – nauja užduotis (angl. *task*)), kuri leidžia paleisti šį

įrankį komandinės eilutės terpėje. Įrankis pagal įvestą laiko intervalą ir pasirinktą analizatorių suranda visas tuo laiku vykusias rungtynes, išgauna ir išsaugo jų duomenis į lokalią duomenų bazę.

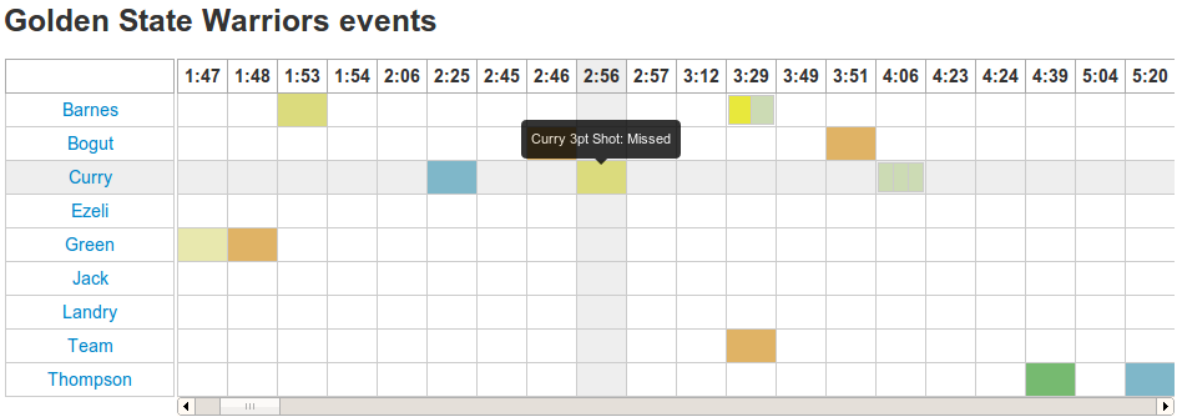
Įvykio laiko rungtynėse skaičiavimas taip pat nėra labai trivialus uždavinys. Problema tame, kad įvairiuose šaltiniuose laikas gali būti nurodomas įvairiai – pavyzdžiui šaltinis nba.com nurodo einamajame kėlinyje likusį laiką (kaip parodyta 8 paveiksle). Tačiau tam, kad struktūrizuoti duomenis iš skirtingų šaltinių ir parodyti įvykius laiko ašyje, laikas bus saugomas skaičiuojant nuo rungtynių pradžios.

START OF 1ST QUARTER		
(12:00) JUMP BALL LOPEZ VS NOAH (BOOZER GAINS POSSESSION)		
Butler Jump Shot: Missed	11:44	
Noah Rebound (Off:1 Def:0)	11:42	
Boozer Jump Shot: Missed	11:40	
	11:38	Evans Rebound (Off:0 Def:1)
	11:26	Johnson Turnover : Bad Pass (1 TO) Steal:Robinson (1 ST)
Robinson Turnover : Out of Bounds Lost Ball Turnover (1 TO)	11:24	
	11:02	Johnson Jump Shot: Missed
Team Rebound	11:01	
Noah Jump Shot: Missed	10:47	
Boozer Rebound (Off:1 Def:0)	10:46	
Boozer Putback Layup Shot: Made (2 PTS)	10:44 [CHI 2-0]	
	10:34	Johnson 3pt Shot: Missed
Belinelli Rebound (Off:0 Def:1)	10:33	
Boozer Layup Shot: Missed	10:21	
Noah Rebound (Off:2 Def:0)	10:20	

8 pav. NBA.com chronologinės įvykių sekos lentelė

Kadangi laikas rodomas einamojo kėlinio atžvilgiu, reikia į tai atsižvelgti ir tikrąjį laiko postūmį šiam šaltiniui skaičiuoti pagal formulę `Ki × KT + KT − IL`, kur Ki – kėlinio eilės numeris, KT – vieno kėlinio trukmė, IL – šaltinyje nurodytas įvykio laikas. Tokiu būdu gaunamas sekundžių, praėjusių nuo žaidimo pradžios skaičius.

Šiuos duomenis buvo nuspręsta atvaizduoti lentelėje, kur X ašis – rungtynių laikas, o Y – žaidėjai. Taip vartotojas gali glaustai matyti įvykius ir efektyviau daryti išvadas apie įvykių tarpusavyje priklausomybę. Šios iteracijos ribose sukurtas šablonas, padėsiantis atvaizduoti rungtynių įvykių seką taip, kaip parodyta 9 paveiksle.



9 pav. Chronologinės įvykių sekos vizualizacija

Šioje realizacijoje pasirinktos įvykių atvaizdavimo spalvos:

- Žalia spalva asocijuojama su baudos metimu arba dvitaškio pataikymu (ryškesnė spalva – daugiau taškų);

- Melsvai žalia spalva nurodo pataikytą tritaškį;
- Geltona spalva žymimi nepataikyti metimai. Kuo ryškesnė spalva, tuo didesnio skaičiaus taškų metimas buvo nepataikytas.
- Oranžinė spalva žymi kamuolio perėmimą;
- Raudono atspalvio langeliai nurodo pražangas.

3.2.3. Trečia iteracija

3.2.3.1. Aprašymas ir užduotys

Trečioje iteracijoje tobulinamas sukauptų duomenų atvaizdavimas ir taip pat tikslinamas kai kurių duomenų analizės komponento dalių veikimas. Šiai iteracijai buvo išryškinti tokie reikalavimai:

- žaidėjo padaryta klaida ir pražanga turi būti 2 atskiri įvykiai;
- chronologinės įvykių sekos peržiūra turi būti kuo patogesnė vartotojui;
- žaidimo peržiūros lange turi vizualiai matytis komandų taškų skirtumas tam tikru laiko momentu;
- turi būti galimybė keisti chronologinės įvykių sekos lango spalvų žymėjimus;
- turi būti galimybė pasirinkti kokius konkrečiai žaidėjus rodyti Y ašyje peržiūrint rungtynių įvykius;
- reikalinga galimybė išsaugoti žaidimo peržiūros lango nustatymus ir juos pasiekti per tiesioginę nuorodą;
- iš lentelės turi matytis koks žaidėjas yra aikštėje duotuoju momentu.

Išnagrinėjus reikalavimus buvo išskirtos užduotys šiai iteracijai:

1. Logiškai atskirti pražangas nuo kitų žaidėjų padarytų klaidų (pamestas kamuolys ir pan.);
2. Derinti chronologinės įvykių sekos atvaizdavimo šabloną taip, kad jis būtų kuo įmanoma patogesnis vartotojui:
    a. siaurinti įvykių kvadratėlius taip, kad vieno kėlinio įvykiai matytųsi be slankjuosčių;
    b. paruošti įvykių atvaizdavimo variantą be tinklelio ir langelių, bet su siauromis linijomis, kurios žymės įvykius.
3. Rungtynių peržiūros puslapyje sukurti papildomą grafiką, kuris laiko ašyje atvaizduos taškų skirtumą tarp komandų.
4. Sukurti spalvų parinkimo paletę, kurios nustatymai nurodys chronologinės įvykių sekos lentelės visų įvykių žymėjimo spalvas.
5. Sukurti papildomą žaidėjų sąrašą, kurio pagalba būtų galimybė pasirinkti kuriuos žaidėjus rodyt chronologinės įvykių sekos lentelės Y ašyje.
6. Papildyti žaidimo duomenų atvaizdavimo puslapį taip, kad matytųsi koks žaidėjas tam tikru laiko momentu yra/nėra aikštelėje.

3.2.3.2. Realizacija ir architektūriniai sprendimai

Tobulinamas duomenų iš išorinių šaltinių analizavimo komponento veikimas. Pridedami naujų įvykių tipai – nuo šiol yra registruojami tokie įvykiai, kaip žaidėjų apsikeitimai, saugomi duomenis apie tai, kada ir koks žaidėjas paliko aikštę, kada išėjo. Šitie duomenys leis tiksliau atvaizduoti situaciją aikštelėje, nes galima atsekti kurie žaidėjai tam tikru laiko momentu dalyvavo rungtynėse, o kurie buvo atsarginių žaidėjų aikštelėje.

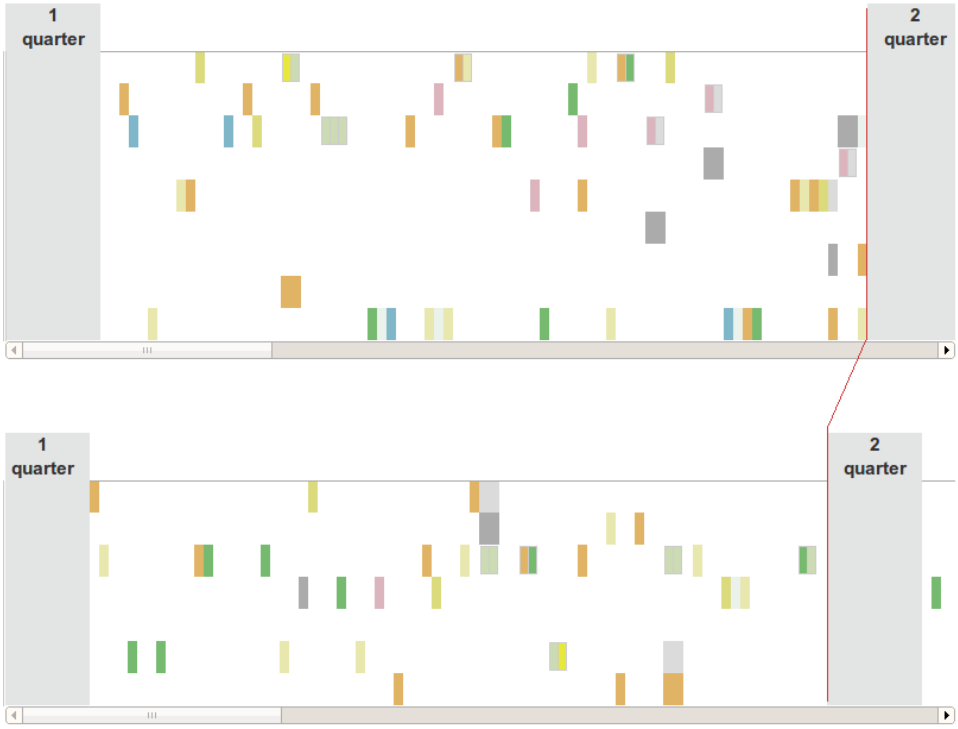
Taip pat logiškai atskiriami pražangų įvykiai – pražanga ir kamuolio pametimas išoriniame NBA šaltinyje žymimi kaip *foul*, tačiau kamuolio pametimo įvykis turi papildomą atributą ir už tai skiriami kiti baudos taškai, negu kad pražangos atveju.

Pasitarus su užsakovu buvo nuspręsta, kad yra svarbu jog chronologinės įvykių sekos lentelę galima būtų prasukti (angl. *scroll*) taip, kad iškarto matytųsi vieno kėlinio duomenys. Tokiam tikslui reikia siaurinti kvadratus, kurie žymi įvykius. Tačiau iškila problema – viršutinėje juostoje yra surašytas įvykių atsitikimo laikas rungtynių atžvilgiu, tad šriftą šioje juostoje reikėtų sumažinti tiek, kad įrašytas tekstas netektų prasmės, nes tada taptų absoliučiai neįskaitomas. To pasėkoje buvo nuspręsta atsisakyti laiko juostos, o įvykių atsitikimo laiką pernešti į informacijos burbulą (angl. *tooltip*), kuris parodomas užvedus pelę ant įvykio. Taip pat buvo atsisakyta lentelės tinklelio, kuris

vizualiai atskirdavo visus įvykius bei tuščius stačiakampius vienas nuo kito. Tinklelis liko tik tarp sudėtinių įvykių[9] stačiakampių, kad jie nesusilietų į vieną įvyki. Taip gavosi sutaupyti dar keliasdešimt lentelės vaizdo pikselių.

Buvo pastebėta viena problema. Prieš šiuos chronologinės įvykių sekos lentelės vaizdo pakeitimus, vienos sekundės "ilgis" lentelėje buvo fiksuotas. Visada buvo aišku, kad įvykis, kuris įvyko 2 kėlinio *1:43* laiko momentu, lentelėje į plotį užima 40 pikselių. Taip pat ir sudėtinių įvykių stačiakampiai užima 40 pikselių, padalinant juos tarpusavyje. Todėl buvo galima nesunkiai sinchronizuoti dvi šalia esančias skirtingų komandų lenteles taip, kad kiekviena rungtynių sekundė eitu po atitinkama kitos lentelės sekunde. Dabar sudėtinių ir paprastų įvykių stačiakampių plotis gali skirtis – per vieną sekundę įvykusių 3 įvykių neįmanoma sutalpinti į 5 pikselių plotą taip, kad jie būtų vizualiai atskirti. Taigi, nuo šiol nėra garantuojama, kad vienos ir tos pačios sekundės skiltis dviejuose lentelėse bus to pačio dydžio, todėl lentelių sinchronizavime tarpusavyje atsiranda paklaidos (10 paveikslas).

Kuo įmanoma labiau siaurinant įvykius atvaizduojančius stačiakampius buvo pasiektas numatytas rezultatas – vieno kėlinio duomenys telpa į chronologinės įvykių sekos lentelę be jokių slankjuosčių. Naujas įvykių srities lentelių vaizdas parodytas 10 paveiksle.

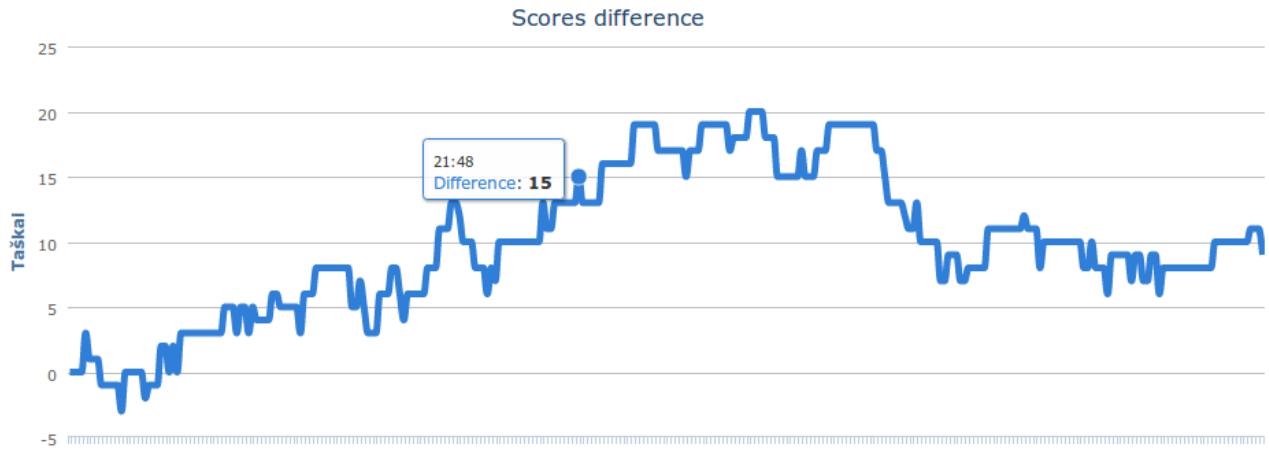


10 pav. Siaurų stačiakampių įvykių lentelė, sinchronizacijos problema pažymėta raudona linija

Pasinaudojant *Highcharts* biblioteka (kurios pagalba piešiamas ir žaidėjo statistikos grafikas) buvo paruoštas interaktyvus komandų taškų skirtumo rungtynių eigoje grafikas. Buvo naudojamas *spline* tipo grafikas, kuris suapvalina kreivių kampus. Grafiko atributai:

- x-ašis atitinka rungtynių eigą;
- y-ašyje vaizduojamas komandų taškų skirtumas. Kiekvienu laiko momentu, kuriuo įvyko taškų skirtumo pasikeitimas (kažkas pelnė taškų) atvaizduojama reikšmė skaičiuojama pagal formulę: `Y = Pt1 - Pt2`, kur `Pt1` – komandos #1 taškai, `Pt2` – komandos #2 taškai. Rezultatas pavaizduotas sekančiame paveiksle.

[9] Sudėtinis įvykis – tai tokia situacija, kai per vieną rungtynių sekundę atsitiko daugiau negu vienas, tariamai vienas nuo kito priklausomas įvykis (pvz. pražanga ir baudos mėtimai, žaidėjų apsikeitimai ir pan.)



11 pav. Komandų taškų skirtumas rungtynių eigoje

Įvairiems žmonėms dėl vienų arba kitų priežasčių gali būti patogiau žiūrėti į įvykius, jeigu jie yra pažymėti tam tikra spalva. Tam tikslui buvo sukurta visų įvykių atvaizdavimo spalvų pasirinkimo paletė – nuo šiol vartotojas pats gali dinamiškai pasirinkti kokia spalva žymėti bet kuriuos įvykius.

12 pav. Įvykių spalvų paletė

Peržiūrint chronologinės įvykių sekos statistiką dažnai pastebimos situacijos, kai norima pamatyti tik tam tikrų žaidėjų rinkinio duomenis. Ieškant šitų duomenų bendroje lentelėje ši užduotis nėra lengva – lentelėje atvaizduojami visi komandos žaidėjai, ir reikia nuolatos vizualiai išskirti ir stebėti tik dominančias įvykių eilutes. Šitos problemai buvo pasirinktas tos sprendimas – leisti vartotojui pačiam pasirinkti kokių žaidėjų įvykius atvaizduoti. Prie kiekvieno žaidėjo pavardės sukurtas trynimo ženklas, kuri nuspaudus žaidėjo eilutė yra trinama iš bendros įvykių lentelės. Taip išmetant nereikalingus žaidėjus iš statistikos galima sudaryti lentelę, susidedančią tik iš dominančių žaidėjų sąrašo. Naujas lentelės žaidėjų sąrašas pavaizduotas 13 paveiksle.

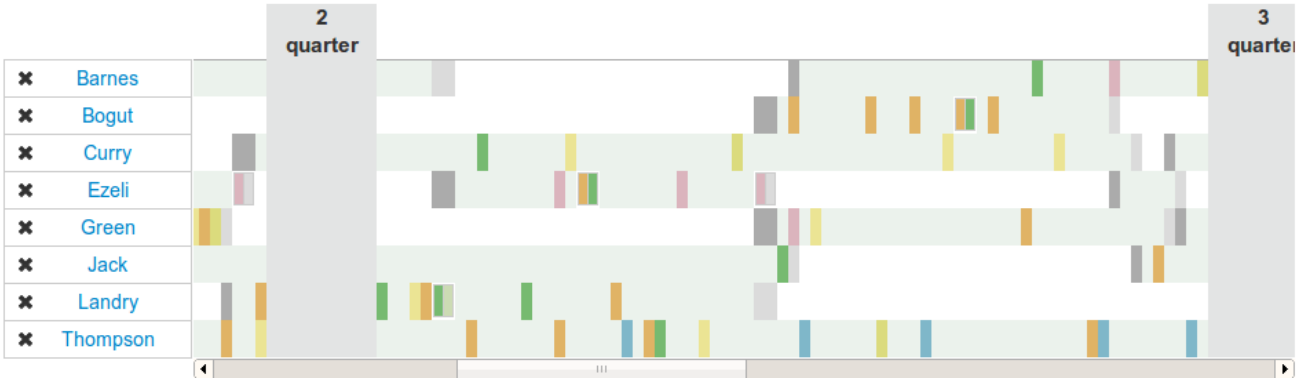
13 pav. Žaidėjų, su galimybe juos pašalinti iš lentelės sąrašas

Svarbu reikalavimas – turi būti galimybė atskirais laiko momentais vizualiai matyti kurie žaidėjai tuo metu žaidžia aikštelėje. Tam buvo nuspręsta patobulinti chronologinės įvykių sekos lentelę, žyminčią atskirus eilučių regionus skirtingomis spalvomis. Jeigu regionas nėra baltas, bet užpildytas kokia nors neryškia spalva – tai reiškia, jog tuo metu žaidėjas dalyvauja rungtynėse ir yra aikštelėje. Balti regionai nurodo, jog žaidėjas tuo metu rungtynėse nedalyvauja.

Realizuojant šį funkcionalumą iškilo problema – NBA.com šaltinyje pastoviu formatu nėra nurodyti startiniai komandų žaidėjų penketai. Tam, kad atskirti kokie žaidėjai pradeda dalyvavimą nuo pat rungtynių pradžios buvo sukurtas toks algoritmas:

1. imama kiekvieno žaidėjo rungtynių įvykių aibė;
2. iteruojama per visus įvykius iš eilės, atsižvelgiant į jų laiką rungtynėse;
3. randamas pirmas žaidėjo apsikeitimo įvykis *E*;
4. jeigu įvykis *E* nurodo žaidėjo pašalinimą iš aikštelės, reiškia jis žaidė nuo rungtynių pradžios – padaroma išvada, jog žaidėjas priklauso startiniam penketui.

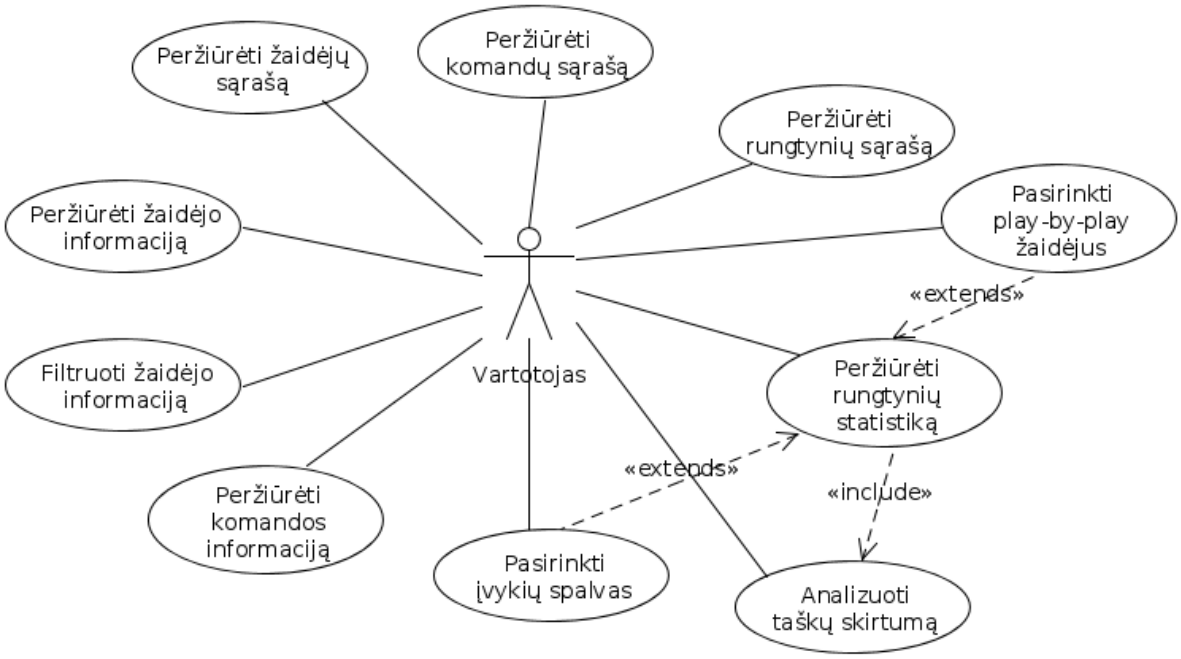
Galutinis šios iteracijos įvykių lentelės vaizdas parodytas 14 paveiksle (aikštelėje esančių žaidėjų eilučių regionai pažymėti žalsva spalva).



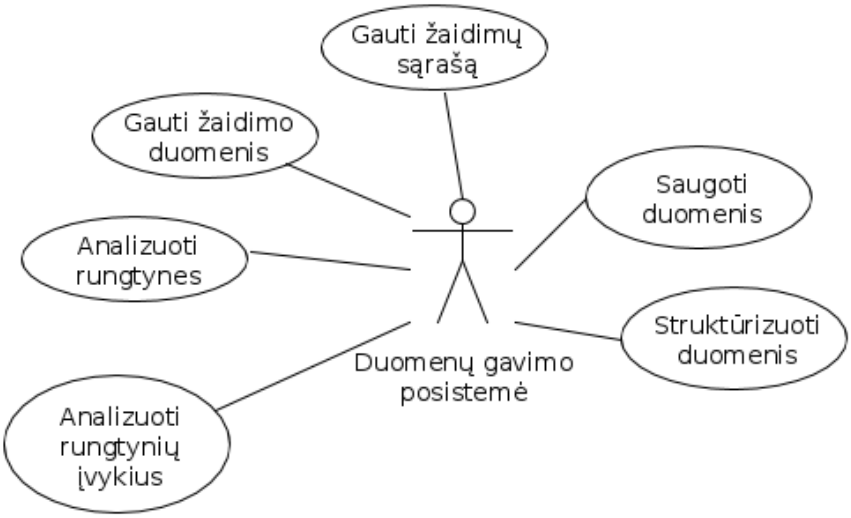
14 pav. Galutinis 3-ios iteracijos chronologinės vykių sekos lentelės vaizdas

Įvykdžius aukščiau aprašytas užduotis buvo pasiekti visi trečiosios iteracijos tikslai. Tačiau buvo išryškintos ir kelios problemos, kurias reikės spręsti tobulinant sistemą.

3.3. Sistemos panaudos atvejų diagramos



15 pav. Vartotojų panaudos atvejų diagrama



16 pav. Duomenų gavimo posistemės panaudos atvejų diagrama

4. TYRIMO DALIS

Tyrimo tikslas – išsiaiškinti kaip galima praplėsti sukurtos sportinių įvykių analizės ir vizualizavimo sistemos SEA funkcionalumą taip, kad įvykių analizė taptų efektyvesnė bei realizuoti numatytus patobulinimus. Šiam tikslui pasiekti reikia ištirti žaidimo ir sportinių įvykių specifiką, išryškinti šablonines situacijas, realizuoti mechanizmą, leisiantį grupuoti bei analizuoti sportinius duomenis.

Sukurta sportinių įvykiu analizės ir vizualizavimo sistema SEA šiuo metu veikia tik su vienos sporto šakos duomenimis – krepšinio rungtynių įvykiais, o sportinių įvykių duomenims saugoti sukurta unikali duomenų struktūra. Todėl šio tyrimo kontekste bus nagrinėjamos būtent krepšinio įvykių duomenų analizės galimybės, skirtos tik šiai sistemai.

Tyrimui nėra keliamas tikslas praplėsti palaikomų sporto šakų aibės.

4.1. Šabloninės situacijos samprata

Šabloninė situacija sporto rungtynėse – tai įvykių seka, kurios baigtis yra tam tikra numatyta situacija [12]. Sustabdžius rungtynes bet kuriuo laiko momentu, visą situaciją, esančią aikštelėje galima aprašyti tam tikru informacijos rinkiniu arba modeliu (žaidėjų padėtis aikštelėje, komandų būsenos, kamuolio padėtis, taškų skirtumas, dalyvaujančių rungtynėse žaidėjų sąrašas ir kt.). Tačiau dauguma tokių situacijų neatneš svarbios informacijos.

Pavyzdžiui, žaidėjo aikštėje judėjimo trajektoriją irgi galima traktuoti, kaip įvykių seką. Pasitelkus šia informacija galima daryti įvairias išvadas apie rungtynių eigą. Taip kai kurie mokslininkai išskiria krepšinio komandos puolimo procesą į tokius elementus (pagal žaidėjų judėjimo trajektorijas) [30]:

- Komandos žaidėjai pradeda formuotis į charakteringąsias pozicijas. Tai – pats reikšmingiausias požymis, kad prasideda pasiruošimas puolimui. Tikimybė, kad vykstančių įvykių seka reiškia būtent formavimą, yra apskaičiuojama remiantis žaidėjų nuotoliais nuo jų optimalios pozicijos ir jų judėjimo kryptimi.
- Užtvara. Tai dviejų žaidėjų kontaktas. Vienas žaidėjas stovi su kiek įmanoma mažiau judesių, kai antras prabėga labai arti pirmojo.
- Žaidėjų aikštelėje judėjimo observavimas. Pasinaudojant iš anksto paruoštais modeliais, kurie aprašo idealią žaidėjų judėjimo kryptį, yra galimybė aprašyti žaidėjo judėjimą tiesiog stebint zonas, į kurias jis įžengė.

Tačiau šio darbo kontekste tokia informacija būtų perteklinė, nes nesuteikia jokios faktinės informacijos apie atakos rezultatą. Todėl yra išskiriami atskiri, tyrimams svarbūs įvykių tipai:

- pataikytas baudos metimas;
- nepataikytas baudos metimas;
- pataikytas dvitaškio metimas;
- nepataikytas dvitaškio metimas;
- pataikytas tritaškio metimas;
- nepataikytas tritaškio metimas;
- pražanga;
- kamuolio atkovojimas;
- kamuolio praradimas;
- žaidėjas palieka aikštę;
- žaidėjas ateina į aikštę.

Analizuojant tokius įvykius yra įmanoma išryškinti standartinę situaciją aikštelėje. Šie įvykiai dažnai reiškia atakos pabaigą (sėkminga arba nesėkminga ataka), tačiau su tam tikromis išimtimis. Pavyzdžiui, pražangos įvykio tipas negarantuoja, kad su juo pasikeis ir komandos pozicija.

4.2. Sukurtos SEA sistemos įvykių analizės galimybės

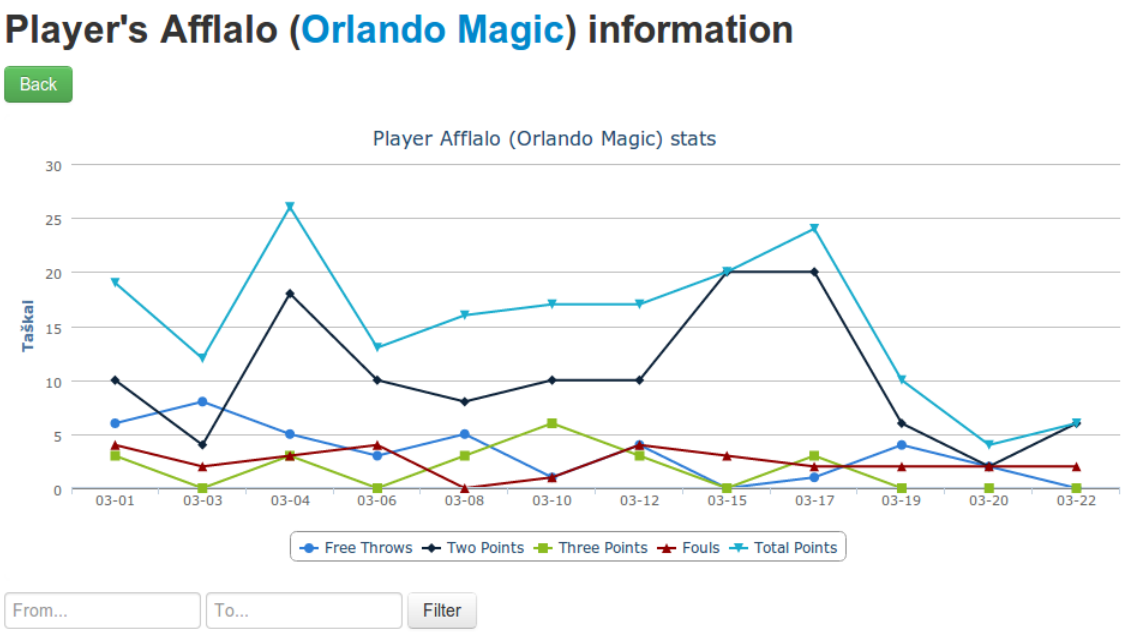
Pasinaudojant SEA sistemos galimybėmis, galima matyti sporto įvykių statistiką įvairiais aspektais. Tai leidžia efektyviai stebėti konkrečiu laiko momentu aikštėje esančią situaciją. 17

paveiksle parodytas rungtynių chronologinės įvykių sekos lentelės pavyzdys. Tai lentelės, kurios nurodo atsitikusius įvykius rungtynių eigoje. Patogumo dėlei, šie įvykiai atvaizduojami laiko ašyje. T.y. X-ašis – rungtynių laikas, o Y-ašis – konkretūs žaidėjai.

- Kairėje lentelės pusėje matomas valdomas žaidėjų sąrašas (1). Jeigu vartotoją domina tik konkretūs žaidėjai, tai nereikalingus galima išimti iš statistikos, nuspaudus kryžiuko ženklą šalia norimo žaidėjo.
- Patogumo dėlei lentelėje yra žymima kuris žaidėjas kuriuo laiko momentu dalyvavo žaidime (buvo betarpiškai aikštėje, o ne ant atsarginių žaidėjų suolelio). Žaidėjo, dalyvaujančio rungtynėse tam tikru laiko momentu, ašies fonas yra žalsvesnis, negu tada, kai žaidėjas nedalyvauja rungtynėse (2).
- Įvykiai žymimi skirtingų spalvų blokais lentelėje. Užvedus pelės kursorių virš konkretaus įvykio, parodoma papildoma įvykio informacija (įvykio laikas, įvykio dalyviai, rezultatas ir t.t.) (3). Šių blokų spalvų reikšmes galima keisti šio lango apatinėje dalyje esančio "*Events colors*" bloko pagalba.
- Kėliniai yra atskirti vertikaliais pilkais blokais (4).

17 pav. Rungtynių įvykių lentelė

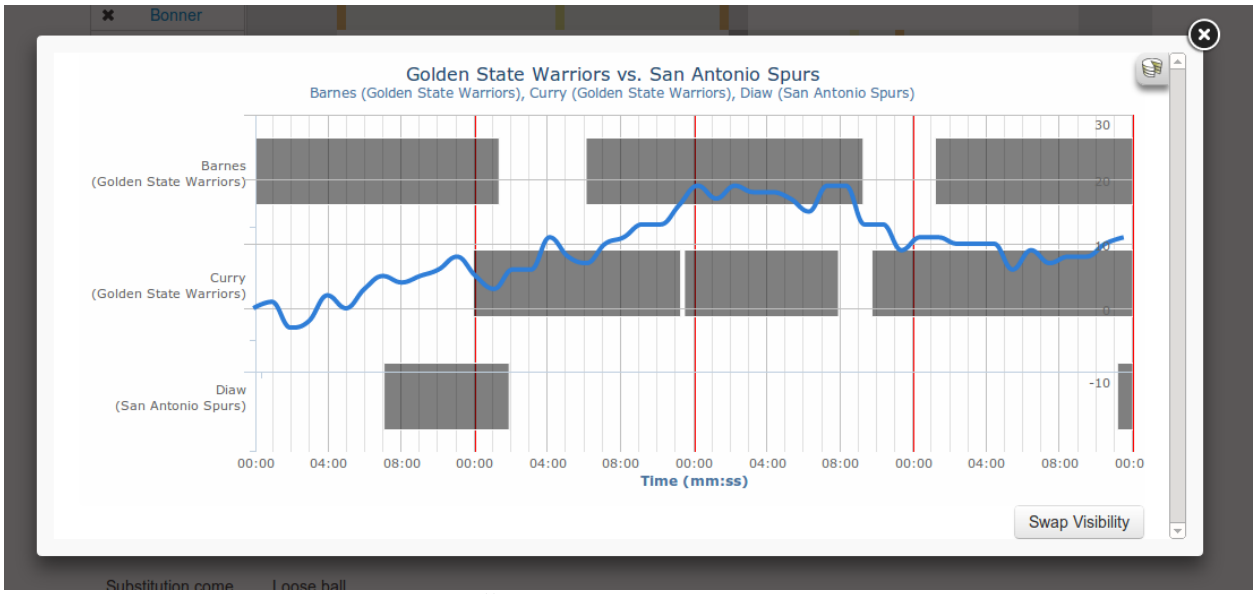
Sekančiame paveikslėlyje yra atvaizduojama pagrindinė pasirinkto žaidėjo statistika grafiko pavidalu. X-ašyje nurodytos datos, kada vyko rungtynės, kuriuose dalyvavo pasirinktas žaidėjas. Grafiko linijos ir reikšmės rodo kiek kokių įvykių žaidėjas sukūrė per konkrečias rungtynes. Mėlyna spalva žymimi taiklūs baudų metimai, juoda – dvitaškių metimai, žalia – tritaškių. Raudona spalva žymimas padarytų baudų skaičius, o šviesiai mėlyna – bendras pelnytų taškų skaičius per rungtynes. Pasinaudojant filtrais, esančiais lentelės apačioje, yra galimybė filtruoti duomenis ir analizuoti tik pasirinktame laiko rėžyje vykusių rungtynių įvykius.



18 pav. Žaidėjo statistika

Kitas svarbus sistemos grafikas, leidžiantis stebėti žaidėjų porų aikštelėje efektyvumą bei taškų skirtumo kaitą pateiktas sekančiame paveiksle. Grafiko X-ašyje yra rungtynių laiko juosta. Kėliniai yra atskiriami vertikaliomis raudonomis linijomis. Pilki blokai rodo, kada žaidėjai dalyvavo žaidime, o mėlyna kreivė rodo taškų santyki tarp komandų (šiuo atveju rodo kiek taškų komanda “Golden State Warriors” buvo priekyje prieš “Sant Antonio Spurs”. Taškų Y-ašis yra dešinėje grafiko pusėje.

Tokiu būdu atsiranda galimybė stebėti kiek efektyviai žaidžia tam tikri žaidėjų deriniai. Pavyzdžiui, iš šito grafiko yra matoma, kad trečiojo kėlinio pabaigoje, kai iš aikštelės išėjo žaidėjas “Barnes”, taškų skirtumas staigiai nukrito nuo 20 iki 10. Taip renkant skirtingas žaidėjų poras arba trejetus galima patogiai analizuoti jų efektyvumą žaidimo eigoje.



19 pav. Žaidėjų porų/trejetų efektyvumas

4.2.1. Numatomi sistemos patobulinimai

Pagrindinis SEA sistemos trūkumas yra tas, kad norint tarp visos sportinių įvykių duomenų aibės įžvelgti tam tikras šablonines situacijas, tą reikia daryti rankiniu būdu. Sistema pritaikyta tam, kad būtų galima patogiai matyti ir analizuoti duomenis, tačiau nėra numatyta būdo, kuris padėtų greitai atsekti konkrečias situacijas visų rungtynių atžvilgiu arba automatiškai rasti dažnai pasitaikančias įvykių sekas. Ištyrus šabloninių situacijų specifiką planuojama sukurti specialų įranki, kurio pagalba tokių situacijų išryškinimas taptų paprastesnis.

Iš sukauptos duomenų bazės, kurioje saugomi NBA rungtynių įvykiai galima pastebėti, kad vidutiniškai kiekvienos NBA rungtynės susideda iš 428 įvykių. Per mėnesį vidutiniškai yra sužaidžiama 236 rungtynių. Matoma, kad per mėnesį vien tik NBA lygoje yra sugeneruojama virš 100 tūkstančių registruojamų įvykių, o metų bėgyje šis skaičius viršija ir milijoną įrašų. Akivaizdu, kad atsiranda būtinybė analizuoti didelius duomenų kiekius. Tokiems uždaviniams spręsti verta pasikliauti duomenų gavybos srityje tiriamais algoritmais bei idėjomis, kurios aprašytos šio darbo *2.3.5. Duomenų gavybai skirti algoritmai* skyriuje.

4.3. Šabloninių situacijų išryškinimas

Elementarių šabloninių situacijų išryškinimui gali pakakti ir trivialaus įvykių sekos vizualizavimo. Tačiau sudėtingesnėse situacijose dažnai figūruoja daug įvairių įvykių, kas ženkliai apsunkina analizę. Tokios situacijos aprėpia skirtingus įvykių tarpusavio sąryšius bei jų kontekstą. Vienas ir tas pats įvykis, žvelgiant į jį skirtinguose kontekstuose, gali turėti skirtingą reikšmę. Tam, kad padaryti teisingą išvadą apie konkrečius įvykius ir įžvelgti juose situaciją, reikia iš anksto numatyti kuriame kontekste bus operuojama duomenimis. Išskiriami keli baziniai kontekstai.

Kontekstas, priklausantis nuo laiko. Tai tokios situacijos, kurios yra tiesiogiai priklausomos nuo įvykių atsitikimo laiko. Tokia priklausomybių grupė dalinama į 2 tipus:

1. Priklauso nuo rungtynių laiko. Išvados apie situaciją yra daromos priklausomai nuo to, kada įvykis arba įvykiai atsitiko rungtynių arba kėlinio pradžios atžvilgiu. Pavyzdžiui: įvykis atsitiko trečio kėlinio pirmų trijų minučių bėgyje.
2. Kiek laiko atžvilgiu atitolęs nuo kito įvykio. Situacija priklauso nuo įvykių grupės, kurioje kiekvienas iš įvykių yra nutolęs vienas nuo kito per tam tikrą laiko atkarpą (tikslų sekundžių skaičių arba rėžį). Pavyzdžiui: įvykis *X* atsitiko nevėliau, negu po 30 rungtynių sekundžių po įvykio *Y*.

Kontekstas, priklausantis nuo gretimai arba sąlyginai netoli vienas nuo kito esančių įvykių. Tokiose situacijose atsižvelgiama į įvykį gretimai atsitikusių įvykių kontekste. Tokie kontekstai yra panašūs į kontekstus, priklausančius nuo laiko, tačiau operuojama ne laiko vienetais, bet pačiais įvykiais. Atsižvelgiama į įvykio poziciją gretimai esančių įvykių atžvilgiu. Pavyzdžiui: įvykis *X* atsitiko po įvykio *Y* (iškarto, arba nevėliau, negu po *Z* kitų įvykių).

Priklausantis nuo konkrečios įvykių sekos. Šios situacijos priklauso nuo betarpiškai iš eilės einančių įvykių aibės. Toks kontekstas yra itin svarbus, kai bandoma pastebėti sąryšius tarp įvykių, kurie buvo sukurti skirtingų žaidėjų arba net skirtingų komandų. Pavyzdžiui: iš eilės atsitiko įvykiai *X*, *Y* ir *Z*.

Žvelgiant viename iš aukščiau išvardintų kontekstų, atsiranda galimybė juos grupuoti pagal skirtingus požymius. Analizuojant įvykį kaip duomenų rinkinį be jokio konteksto yra neįmanoma padaryti išvadų arba pastebėti šabloninių situacijų. Iš loginės pusės kiekvienas įvykis susideda iš pagrindinių atributų, išvardintų žemiau.

Įvykio tipas. Plačiau apie sistemoje naudojamus įvykių tipus kalbama šio darbo punkte *4.1. Šabloninės situacijos samprata*.

Subjektas, sugeneravęs įvykį. Tai labai svarbus aspektas kalbant apie šabloninių situacijų išryškinimą. Tai asmenys, kurie "sukuria" įvykius. Kalbant apie krepšinį teigiama, kad visus įvykius sukuria tik žaidėjai, konkrečiu laiko momentu esantys aikštelėje. Tačiau šis faktas ne visada gali atrodyti trivialus. Pavyzdžiui, suprantama, jog sprendimą dėl to, kad žaidėjas turi palikti aikštelę priima treneris ir kiti suinteresuoti asmenys, tačiau analizės atžvilgiu yra numatomi konkretūs įvykiai – žaidėjas paliko aikštelę, žaidėjas įėjo į aikštelę, kurie įvykio kūrėją nurodo būtent žaidėją, o ne trenerį.

Analizuojant įvykio *A* kontekstą ir sąryšį su kitu įvykiu *B* yra svarbu žinoti ar:

1. Įvykius *A* ir *B* sugeneravo vienas ir tas pats žaidėjas.
2. Įvykius *A* ir *B* sugeneravo skirtingi žaidėjai iš tos pačios komandos.
3. Įvykį *A* sugeneravo vienos komandos žaidėjas, o įvykį *B* – kitos.

Kaip galima matyti iš SEA sistemos galimybių analizės (šio darbo skyrius *4.2. Sukurtos SEA sistemos įvykių analizės galimybės*), kartais atsitinka tokie įvykiai, kurių autoriaus neįmanoma atsekti. Pavyzdžiui, komandinis kamuolio praradimas, kai kamuolį prarado ne vienas konkretus žaidėjas, bet visa komanda (nesėkmingai padarytas perdavimas). Tokioms situacijoms kiekvienoje iš komandų sukuriamas papildomas žaidėjas "Team", kuriam ir priskiriami tokie įvykiai. Vėliau į šiuos įvykius žiūrima taip pat, kaip ir į įvykius, sugeneruotus tikrų žaidėjų.

Įvykio laikas rungtynių pradžios atžvilgiu. Esamoje sistemoje įvykio laikas fiksuojamas rungtynių, bet ne kėlinio pradžios atžvilgiu, sekundės tikslumu. Atlikus paprasčiausius aritmetinius skaičiavimus galima sužinoti kėlinį, kuriame įvykis atsitiko, bei laiko poslinkį nuo to kėlinio pradžios. Šis atributas yra kritiškai svarbus, kai įvykiais operuojama kontekste, priklausančiame nuo laiko.

4.4. Šabloninių situacijų išgavimas

Šio darbo kontekste sportinių įvykių analizės atžvilgiu šabloninės situacijos gali būti dviejų tipų:

1. *Situacijos, kurias apskaičiavo sistema*. Tai automatiškai išryškintos rungtynių metu dažnai pasitaikančios situacijos.
2. *Pagal vartotojo suformuotą užklausą gautos situacijos*. Tai situacijos, kurios gautos analizuojant duomenis pagal vartotojo nustatytus parametrus.

Abu šie duomenų analizės tipai yra ypač svarbūs, norint efektyviai stebėti sistemoje esančius sporto įvykių duomenis. Šiame skyriuje bus nagrinėjamos abiejų šių tipų analizės algoritmo realizavimo galimybės.

4.4.1. Automatinė šabloninių situacijų paieška

Šio darbo skyriuje *2.3.5.1. Dažnai pasitaikančių šablonų gavyba* aprašytas algoritmo SPADE veikimo principas. Pritaikius šį algoritmą sukurtai SEA sistemai, galima ženkliai pagerinti vartotojo patirtį ir patogumą, analizuojant sistemoje esančius duomenis.

Kaip buvo minėta *3.1. Kūrimo technologijos, įrankiai ir karkasas* skyriuje, sistemos veikimui naudojama MySQL duomenų bazė. Užklausoms į šią duomenų bazių valdymo sistemą naudojama jau defakto reliacinėms duomenų bazėms tapusi SQL kalba[10]. Todėl realizuojant SPADE algoritmą yra svarbu jį pritaikyti būtent šiai kalbai ir sistemai.

Visas pagrindinis duomenų analizės darbas vyks pasinaudojant lentelėje *events* esančiais duomenimis. Šios lentelės struktūra pavaizduota sekančiame paveikslėlyje.

Atributas	Tipas
id	int(10) unsigned *Auto Increment*
game_id	int(10) unsigned
team_id	int(10) unsigned
player_id	int(10) unsigned
time	int(11)
source	varchar(256)
event_type	tinyint(3) unsigned

20 pav. Lentelės events struktūra

Sistemoje naudojamas *InnoDB* duomenų saugojimo variklis, kuris nėra pritaikytas konkrečiai sekoms saugoti, bet gerai tinka kitų duomenų struktūrų, naudojamų sistemoje, saugojimui. Tačiau į

[10] SQL (angl. *Structured Query Language*) – populiariausia iš šiuo metu naudojamų kalbų, skirtų aprašyti duomenis ir manipuliuoti jais reliacinių duomenų bazių valdymo sistemose

grąžintą rezultatą (eilučių rinkinį) galima žiūrėti kaip į seką. Tuomet yra svarbu, kad eilutės būtų tinkamai išrikiuotos. Šiuo atveju rezultatas turi būti išrikiuotas pagal įvykio laiką, kad būtų matoma visa rungtynių įvykių seka chronologine tvarka. Tam, kad išrikiuoti duomenis pagal norimą atributą, SQL kalboje naudojama komanda *ORDER*. Norint išgauti konkrečių rungtynių įvykių seką reikia padaryti tokią užklausą:

```
SELECT * FROM events WHERE game_id = x ORDER BY time ASC
```

, kur ***x*** – dominančių rungtynių identifikacinis numeris.

Kaip nurodyta algoritmo SPADE specifikacijoje, pirma problema, kurią reikia išspręsti – tai apskaičiuoti dominančias įvykių grupes. Įvykių grupė – tai vienas arba daugiau iš eilės einančių konkrečių tipų įvykių. Pavyzdžiui, grupė [*nepataikytas metimas, nepataikytas metimas, aikštelės palikimas*] nurodo situaciją, kai iš eilės įvyksta 2 nepataikyto mėtimo ir po jų aikštelės palikimo įvykiai.

Pagrindinis atributas, pagal kurį yra nusakomas įrašo tipas, yra atributas *event_type*, kuris vienareikšmiškai nurodo kokį įvykį reprezentuoja konkretus įrašas. Patogumo dėlei kiekvienam įvykių tipui yra priskirtas unikalus identifikatorius – sveikas skaičius. Sekančioje lentelėje nurodyta koks skaičius reprezentuoja kokį įvykį.

1 lentelė. Įvykių identifikaciniai numeriai

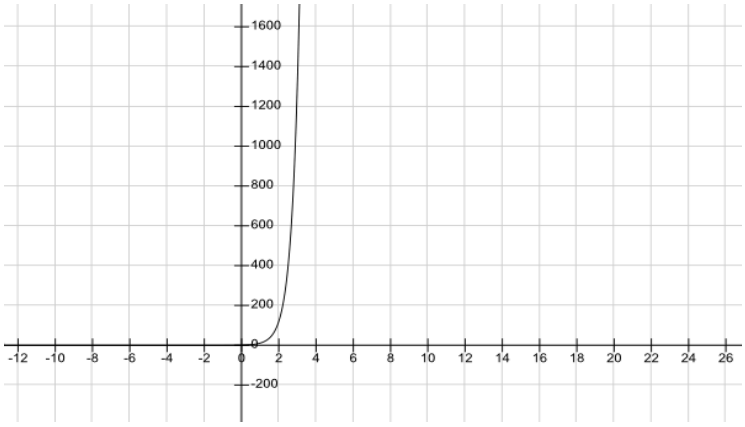
Įvykio tipas	Unikalus identifikatorius
Pataikytas baudos metimas	**1**
Pataikytas dvitaškis	**2**
Pataikytas tritaškis	**3**
Pražanga	**4**
Nepataikytas dvitaškis	**5**
Nepataikytas tritaškis	**6**
Nepataikytas baudos metimas	**7**
Perimtas kamuolys	**8**
Žaidėjas palieka aikštę	**9**
Žaidėjas ateina į aikštę	**10**
Kamuolio praradimas	**11**

Toliau šiame darbe įvykių tipai bus žymimi jų identifikatoriais, o ne pavadinimais (pavyzdžiui, #3, #5 ir t.t.).

Pagal algoritmą įvykių tipų grupės turėtų būti formuojamos dinamiškai – prieš kiekvieną vykdymą turi būti išrenkami unikalūs įvykių tipai bei iš jų formuojamos įvykių grupės. Tačiau SEA sistemos atveju įvykių tipai – tai fiksuotas identifikatorių rinkinys, susidedantis iš 11 elementų (įvykių tipų), todėl nėra būtinybės kiekvieną kartą daryti papildomų užklausų, siekiant išgauti visus sistemoje egzistuojančius įvykių tipus. Visų įvykių grupių generavimas – tai visų įmanomų įvykių rinkinių konstravimo uždavinys. Pirmoje iteracijoje grupės susideda tik iš vieno tipo, antroje – iš dviejų ir t.t. Į sekas įtraukiami visi įmanomi tipų deriniai. Sekoje vienas ir tas pats įvykio tipas gali kartotis. Problema – visų įmanomų įvykių derinių sukūrimas. Šiam tikslui galima panaudoti paprastą rekursiją ir sugeneruoti visus įmanomus sekų variantus. Formavimas vyksta tokiu principu (kabliataškio simbolis žymi sekos pabaigą):

```
#1; #2; ...; #11; #1,#1; #1,#2; #1,#3; ...; #11,#10; #11,#11; #1,#1,#1; #1,#1,#2;
...; #11,#11,#11; #1,...,#1; #11,...,#11;
```

Pagal algoritmą formavimas turi vykti tol, kol randama šablonų su apskaičiuotomis sekomis. Bendras sugeneruotų sekų skaičius yra lygus 11^n, kur *n* – maksimalus sekų ilgis. Tačiau sportinių įvykių atveju galima nustatyti fiksuotą maksimalų sekos ilgį. Tam reikia apskaičiuoti koks yra didžiausias dažnai pasitaikančių situacijų ilgis. Taip pat svarbu yra išsirinkti optimalų ilgį, nes kuo ilgesnės sekos lyginamos, tuo daugiau resursų reikalaujama, o didėjant ilgiui algoritmo sunaudojami resursai ir vykdymo laikas eksponentiškai auga. Sekančiame grafike pavaizduota kiek iteracijų reikia atlikti kokiam sekos ilgiui.



21 pav. SPADE iteracijų skaičius sporto įvykių analizei. X-ašis – maksimalus sekos ilgis, Y-ašis – iteracijų skaičius.

Su kiekviena apskaičiuota grupe skenuojami duomenys ir skaičiuojama kiek kartų jose atsiranda ieškoma įvykių seka (grupė). Tam reikia atrasti būdą kaip duomenų bazės lygmenyje galima rezultatų eilutes interpretuoti kaip įvykių sekas, kad išrinkti tik tuos duomenis, kurie atitinka testuojamą grupę. Sekanti problema, kurią reikia išspręsti – duomenų išrinkimas pagal šalia esančius įrašus.

Tam, kad užklausose naudoti sąlygas, susijusias su šalia esamais įrašais, į užklausą reikia pridėti reikiamų įrašų duomenis. Jeigu sekos ilgis yra *n*, tai užklausoje kartu reikia išgauti papildomai *(n - 1)* sekančių įrašų tipus. Pavyzdžiui, norint išgauti situacijas, kuriose egzistuoja įvykių seka *#8, #5, #8*, reikia gauti visus įrašus, kurių tipas yra #8, po kurio eina įrašas su tipu #5, po kurio eina įrašas su tipu #8. Tokio efekto galima pasiekti pasinaudojant MySQL komanda *LIMIT*[11]. Kiekvienam, sekančiam po pirmo elemento, turi būti daroma papildoma subužklausa (angl. *subquery*), išrenkant įrašo, nutolusio per reikiamą atstumą nuo pirmojo, tipą. Aukščiau pateikto pavyzdžio atveju turėtų būti suformuota ir įvykdyta tokia užklausa:

```
SELECT *,
(
      SELECT event_type
      FROM events
      WHERE time > E1.time
      LIMIT 0, 1
) AS n_event2,

(
      SELECT event_type
      FROM events
      WHERE time > E1.time
      LIMIT 1, 1
) AS n_event3
```

[11] MySQL komanda LIMIT – tai komanda, leidžianti apriboti komandos *SELECT* grąžinamų rezultatų aibę.

```
FROM events E1
WHERE
      E1.event_type = 8
HAVING
      n_event2 = 5 AND
      n_event3 = 8
ORDER BY E1.time ASC
```

Įvykdžius tokią užklausą gaunamas rezultatas, kuriame kiekviena eilutė reprezentuoja ieškomos sekos pradžios tašką (t.y. pirmą ieškomos sekos elementą). Žinant elemento, nuo kurio prasideda seka, poziciją, galima nesunkiai išgauti ir visos sekos įrašus. Tam reikia išgauti *n* eilučių, pradedant nuo rezultate gauto įvykio. Tokios užklausos rezultato pavyzdys pateiktas sekančiame paveikslėlyje.

id	game_id	team_id	player_id	time	source	event_type	n_event2	n_event3
7	1	1	4	110	West Rebound (Off:1 Def:0)	8	5	8
9	1	2	6	113	Valanciunas Rebound (Off:0 Def:1)	8	5	8
36	1	2	7	337	Gay Rebound (Off:0 Def:3)	8	5	8
45	1	2	6	377	Valanciunas Rebound (Off:1 Def:3)	8	5	8
47	1	1	4	393	West Rebound (Off:1 Def:2)	8	5	8
59	1	2	3	442	Johnson Rebound (Off:0 Def:1)	8	5	8
67	1	1	1	486	George Rebound (Off:0 Def:4)	8	5	8
69	1	2	10	495	DeRozan Rebound (Off:0 Def:1)	8	5	8
71	1	1	8	505	Hill Rebound (Off:0 Def:1)	8	5	8
83	1	1	17	575	Team Rebound	8	5	8

22 pav. Šabloninių situacijų paieškos užklausos rezultato pavyzdys

Bendru atveju užklausa konstruojama pagal tokią taisyklę:
Įvykių seka $Y = < y_1, y_2, \ldots, y_n >$, kur n – sekos ilgis

```
SELECT *,
(
      SELECT event_type
      FROM events
      WHERE time > E1.time
      LIMIT 0, 1
) AS n_event2,
(
      SELECT event_type
      FROM events
      WHERE time > E1.time
      LIMIT (i - 2), 1
) AS n_event(i),
. . .
(
      SELECT event_type
      FROM events
      WHERE time > E1.time
      LIMIT (n - 1), 1
) AS n_event(n),

FROM events E1
WHERE
      E1.event_type = y1
HAVING
```

```
      n_event2 = y2 AND
      n_event(i) = yi AND
      . . .
      n_event(n) = yn
ORDER BY E1.time ASC
```

Norint užklausą pritaikyti įvairiems įvykių kontekstams, aprašytiems šio darbo skyriuje *4.3. Šabloninių situacijų išryškinimas*, į *n_event* subužklausas reikia pridėti atitinkamas sąlygas. Pavyzdžiui, norint užtikrinti, kad visi įvykiai būtų atlikti vieno ir to pačio žaidėjo, į subužklausas pridedama sąlyga `player_id = E1.player_id`, o užtikrinant įvykių vientisumą ir faktą, jog jie visi sukurti tos pačios komandos, naudojama sąlyga `team_id = E1.team_id`. Tokios rezultatų išrinkimo galimybės suteikia lankstumo ir apribojamos tik SQL kalbos turimais apribojimais.

Tam, kad sužinoti kiek kartų duomenų rinkinyje sutinkamas ieškomas šablonas, pakanka suskaičiuoti eilučių skaičių rezultate. Tokiu būdu suskaičiavus kiek kartų duomenyse sutinkamas kiekvienas įmanomas įvykių derinys iki ilgio *n* mes galime išrikiuoti šablonus į eilę jų populiarumo tvarka.

Bendras populiariausių šabloninių situacijų gavybos algoritmas atrodo taip:

1. Sugeneruojama aibė *Y*, kurioje egzistuoja visi įmanomi įvykių deriniai. Lankstumo dėlei parametrą *n* (maksimalų sekos ilgį) gali nurodyti pats vartotojas.
2. Pasinaudojant aukščiau aprašyta užklausa, kiekvienai sekai iš aibės *Y* apskaičiuojamas skaičius, parodantis kiek kartų ši įvykių seka buvo atrasta visame tiriamame duomenų rinkinyje.
3. Rezultatas surūšiuojamas nuo daugiausiai iki mažiausiai kartų duomenyse rasto šablono.

Kadangi tokiam gavybos procesui atlikti reikia didelės duomenų imties [31], būtų netikslinga jį taikyti tik vienų rungtynių lygmenyje. Todėl nuspręsta analize atlikti laiko, o ne rungtynių kontekste. T.y. analizuoti įvykius, kurie buvo sugeneruoti datų rėžyje (nuo iki). Dėl tokių milžiniškų analizuojamų duomenų kiekio, užklausų vykdymo laikas smarkiai padidėja.

Atlikus tyrimą buvo išsiaiškinta, kad lentelėje *events* esant apie 400 000 įrašų, užklausa, reikalinga suskaičiuoti vieno šablono atsiradimo intensyvumą, užtrunka vidutiniškai 1.67 sekundės. Nesunku apskaičiuoti, kad norint išanalizuoti visas sekas, kurios susideda iš keturių elementų, prireiks 11^4 `* 1.67 = 24450` sekundžių arba beveik 7 valandų. Todėl tam, kad kiekvieną kartą pakartotinai neanalizuoti jau ankščiau išanalizuotų duomenų, buvo nuspręsta analizės rezultatus saugoti atskiroje lentelėje, į kurią vėliau daryti užklausas, norint išgauti reikiamus duomenis. Naujos lentelės struktūra pavaizduota sekančiame paveikslėlyje.

Atributas	Tipas
Id	bigint(20) unsigned *Auto Increment*
sequence	varchar(32)
date	date
count	int(10) unsigned

23 pav. Lentelės *pattern_frequencies* struktūra

Atribute *sequence* saugoma analizuojama seka; *date* – data, kuriai altlikta analizė; *count* – kiek kartų seka *sequence* buvo aptikta per dienos *date* įvykius. Vėliau, norint gauti dienos statistiką, nebereikia per naują atlikti analizės – pakanka užklausti duomenų iš šios lentelės (jeigu jie jau ten yra).

Sistema taip pat papildyta automatiniu duomenų surinkėju (įrankis pavadintas *FrequentPatternShell*). Šis komponentas nuolat "stebi" naujai atsiradusius duomenis, juos analizuoja, o rezultatą išsaugo į aukščiau aprašytą lentelę. Atlikus eksperimentinius tyrimus, aprašytus šio darbo *5.1.1. Optimalaus sekų ilgio eksperementas skyriuje* buvo nuspręsta, jog

optimalus šio įrankio nagrinėjamų sekų ilgis yra 4, todėl šio komponento pagalba nagrinėjamos sekos, kurių ilgis neviršija keturių elementų.

4.4.2. Vartotojo suformuotos parametrizuotos užklausos

Išgauti dažnai pasikartojantys įvykių deriniai gali padėti išnagrinėti rungtynių eigą ir šablonines situacijas aikštelėje. Iš tokių duomenų galima daryti tam tikras išvadas apie tuo metu esančias tendencijas bei žaidėjų elgesį. Tačiau nemažiau svarbi yra galimybė išgauti pačio vartotojo nustatytas situacijas. Šiam tikslui yra numatoma sukurti parametrizuotą algoritmą, leisiantį vartotojui išgauti tik jį dominančias šablonines situacijas.

Naujai kuriamam duomenų gavybos algoritmui yra keliami tokie reikalavimai:

- turi sugebėti aptikti vartotojo nustatytą įvykių seką konkrečiuose rungtynėse
- įvykių seka gali būti:
  - fiksuota. Įvykiai eina vienas po kito
  - toleruojama laiko žingsniu. Įvykiai turi būti nutolę vienas nuo kito nedaugiau, negu per *t* sekundžių.
- visa įvykių seka gali būti sugeneruota:
  - bet kokio rungtynių dalyvio arba
  - tos pačios komandos žaidėjo arba
  - to pačio žaidėjo.
- algoritmas turi gebėti nurodyti tikslią rastos sekos pradžios laiką.

Smulkesnė šių atributų ir jų loginės prasmės analizė aprašyta šio darbo *4.3. Šabloninių situacijų išryškinimas* punkte. Formalizuojant algoritmui keliamus reikalavimus išryškinami tokie algoritmui perduodami parametrai:

1. Įvykių seka. Maksimalus sekos ilgis ribojamas iki 4 elementų. Tam, kad pagrįsti arba paneigti šio skaičiaus tinkamumą, turi būti atliktas eksperimentas (atliekami eksperimentai aprašyti šio darbo skyriuje *5. EKSPEREMENTINĖ DALIS*). Šis parametras pateikiamas masyvo, susidedančio iš įvykių tipų identifikatorių, pavidalu.
2. Sekos tipas. Pasirenkamas arba fiksuotas sekos tipas arba toleruojamos laiko atžvilgiu sekos tipas. Antru atveju įvedamas papildomas parametras *x*, kuris nurodo maksimalų laiko intervalą tarp visų sekos įvykių. Intervalas nurodomas sekundėmis, turi būti sveikas skaičius (didesnis už 0 ir mažesnis už NBA rungtynių kėlinio trukmę – 720 sekundžių).
3. Subjektas, sugeneravęs įvykių seką (bet kokie dalyviai / tos pačios komandos žaidėjai / tas pats žaidėjas).

Tarpinis algoritmo vykdymo rezultatas, kuris yra ypač svarbus jo veikimo galutinių rezultatų gavimui – sugeneruota MySQL užklausa. Įvykdžius šią užklausą gaunamas rezultatas (eilutės) interpretuojamas kaip sekų pradžių taškai.

Vykdant fiksuotos įvykių sekos paiešką, užklausa yra konstruojama pagal tokias pačias taisykles, kaip aprašyta šio darbo punkte *4.4.1. Automatinė šabloninių situacijų paieška*. Vienintelis skirtumas yra tas, jog įstatomi parametrai ne iš sugeneruotų visų įmanomų sekų aibės, o iš vartotojo įvestų duomenų.

Skirtingai nuo *4.4.1. Automatinė šabloninių situacijų paieška* aprašyto algoritmo, konstruojant užklausą antram įvykių sekos tipui, reikia atsižvelgti ne į konkrečių šalia stovinčių įvykių tipus, o atvirkščiai – kiekvienoje subužklausoje suskaičiuoti kiek kartų sekantis ieškomas įvykių tipas pasirodo leidžiamame laiko rėžyje, pradedant nuo einamojo elemento. Dėl to bendroje užklausoje subužklausų apipavidalinimas atrodo kitaip. Konstruojant subužklausas šiam sekų tipui naudojama tokia taisyklė:

```
(
      SELECT time
      FROM events
      WHERE
            event_type = yi AND
            time > E1.time AND
```

```
            (time - E1.time) < t AND
            game_id = E1.game_id
            [papildomos subjekto sąlygos]
      LIMIT 1
) AS event(i)_count
```

Papildomos subjekto sąlygos formuojamos priklausomai nuo pasirinkto užklausos modelio. Sekančioje lentelėje pateikiamos sąlygos, kurios turi būti pridedamos į subužklausas priklausomai nuo pasirinkto subjekto tipo.

2 lentelė. Papildomos subjektų sąlygos

Subjekto tipas	**Papildoma sąlyga**
Bet koks rungtynių dalyvis	Jokios papildomos sąlygos
Tos pačios komandos žaidėjai	`team_id = E1.team_id`
Tas pats žaidėjas	`team_id = E1.team_id AND` `player_id = E1.player_id`

Tam, kad rezultatas tenkintų keliamas sąlygas, visų papildomų subužklausų reikšmės `event(i)_time` turi būti sveikas skaičius (bet ne reikšmė *NULL*). Šis faktas reikš, kad įvykdžius subužklausą bus rastas bent vienas įvykis, kurio tipas sutampa su ieškomo elemento y_i tipu ir bus gautas tikslus laikas, kada tas įvykis atsitiko. Kadangi subužklausų reikšmės generuojamos dinamiškai, į bendros užklausos *HAVING* sekciją pridedamos reikiamos sąlygos – `event(i)_time IS NOT NULL`. Sąlyga `time > E1.time AND (time - E1.time) < t` užtikrina, jog rasti rezultatai pateks į leistiną laiko rėžį (nuo einamojo elemento nedaugiau, nei *t* sekundžių).

Bendru atveju pilnos užklausos formavimas vyksta pagal tokią taisyklę:
Įvykių seka $Y = < y_1, y_2, \ldots, y_n >$, kur n – sekos ilgis,
kintamasis t – maksimalus leistinas laiko rėžis, po kurio turi būti rastas bent vienas ieškomas įvykis,
kintamasis x – analizuojamų rungtynių unikalus identifikatorius,
papildomos subjekto sąlygos – sąlygos, sugeneruotos pagal lentelėje X nurodytas taisykles. Visose subužklausose naudojamos tos pačios sugeneruotos sąlygos.

```
SELECT *,
(
      SELECT time
      FROM events
      WHERE
            event_type = yi AND
            time > E1.time AND
            (time - E1.time) < t AND
            game_id = E1.game_id AND
            [papildomos subjekto sąlygos]
      LIMIT 1

) AS event(i)_time
FROM events E1
WHERE
      E1.event_type = y1 AND
      E1.game_id = x
HAVING
      event(i)_time IS NOT NULL AND
      . . .
```

```
      event(n)_time IS NOT NULL
ORDER BY E1.time ASC
```

Suformuotos užklausos vykdymo pavyzdinis rezultatas pavaizduotas sekančiame paveikslėlyje. Pavyzdinės užklausos parametrai: $Y = < \#8, \#5 >$, $t = 10$, $x = 1$, subjektas – ta pati komanda.

id	game_id	team_id	player_id	time	source	event_type	event1_time
7	1	1	4	110	West Rebound (Off:1 Def:0)	8	112
67	1	1	1	486	George Rebound (Off:0 Def:4)	8	493
69	1	2	10	495	DeRozan Rebound (Off:0 Def:1)	8	504
168	1	1	5	1173	Hibbert Rebound (Off:1 Def:0)	8	1176
217	1	1	1	1473	George Rebound (Off:0 Def:7)	8	1479
253	1	1	5	1791	Hibbert Rebound (Off:2 Def:1)	8	1798
316	1	1	17	2323	Team Rebound	8	2325
355	1	1	11	2521	Mahinmi Rebound (Off:1 Def:4)	8	2523

24 pav. Įvykių sekos su toleruojamu laiko intervalu rezultato pavyzdys

Kaip galima matyti iš rezultato, rungtynių eigoje įvyko 8 situacijos, kai įvykis #5 įvyko nevėliau, kaip po 10 sekundžių po įvykio #8. Visus įvykius vienoje sekoje sugeneravo ta pati komanda (kiekvienai sekai nurodyta atribute *team_id*). Atributas *time* rodo pirmojo sekos elemento atsiradimo laiką (įvykio #8). Taip pat matome kada tiksliai atsitiko įvykis #5 (atributas *event1_time*). Matome, kad `event1_time - time < 10`, kas reiškia, jog parenkamos tik tos situacijos, kai laiko skirtumas tarp dviejų įvykių neviršija nurodyto $t = 10$.

4.5. Patikslinantys sporto įvykiu analizės klausimai

Tyrimo metu iškilo klausimų ir problemų, reikalaujančių papildomos analizės. Šie sukurto algoritmo veikimo aspektai nėra nagrinėjami šiame darbe:

- Algoritmo vykdymo rezultate gautos šabloninės situacijos nėra validuojamos, patikrinant gautas prielaidas su didesne duomenų imtimi. Šiam tikslui reikia išgauti duomenis iš kitų šaltinių, ko padaryti esamoje sistemoje šiuo metu nėra galimybės.
- Išgaunant įvykių sekas nėra atsižvelgiama į rungtynių kėlinių ribas. Jeigu įvykių seką kerta kėlinių riba, tai ta seka vis tiek bus pridėta prie rezultato. Nėra išsiaiškinta ir apibrėžta kaip turi būti reaguojama į tokias situacijas.
- Automatinės šabloninių situacijų paieškos atveju nėra atsižvelgiama į subjektą, sugeneravusį įvykį. Į vieną seką patenkantys įvykiai gali būti sukurti bet kurio rungtynių dalyvio (skirtingų komandų, skirtingų žaidėjų ir pan.).

Išvardintoms problemoms išspręsti reikalingi papildomi tyrimai, kurių metu būtų išsiaiškinta kaip šios situacijos keičia požiūrį į duomenų visumą. Todėl šio darbo kontekste šie klausimai nėra nagrinėjami.

5. EKSPERIMENTINĖ DALIS

Šioje dalyje bus aprašyti vykdomi eksperimentai. Eksperimentinė bazė tyrimams – tai šio darbo *4. TYRIMO DALYJE* atlikto tyrimo rezultatai (sukurtos užklausos bei algoritmai). Remiantis ištirtais bei naujai sukurtais algoritmais bus atlikti eksperimentai, leidžiantys praktiškai pagrįsti arba paneigti tyrimo metu iškeltas hipotezes, atlikti eksperimentines užklausas bei algoritmų vykdymus. Visi eksperimentai vyks pasinaudojant šio darbo rašymo metu sporto įvykių analizės ir vizualizavimo sistemoje SEA sukauptais ir išanalizuotais duomenimis. Sukauptų duomenų kiekis parodytas sekančioje lentelėje.

3 lentelė. Eksperimentams naudojamų duomenų kiekis

Duomenų tipas	**Kiekis**
Rungtynės	910
Įvykiai	389 708
Komandos	31
Žaidėjai	629
Informacija apie įvykių šablonų dažnumą	24 676 įrašai
Įvykių sekų ilgis automatinei šablonų paieškai	Nuo 1 iki 4

5.1. Automatinės šabloninių situacijų paieškos eksperimentai

Tam, kad būtų patogiau atlikti automatinę šablonų paieška, sukurta SEA sistema buvo papildyta nauju įrankiu. Kadangi teisingam ir patikimam algoritmo veikimui reikalingi didesni duomenų kiekiai, šabloninių situacijų buvo nuspręsta ieškoti ne atskirose rungtynėse, bet didesniame duomenų rinkinyje. Todėl duomenų rinkiniai yra ribojami ne vienų išrinktų rungtynių, bet ženkliai platesniu kontekstu – vartotojo pasirinktu datų rėžiu. Analizuojami visi pasirinktame laiko tarpe sugeneruoti įvykiai.

5.1.1. Optimalaus sekų ilgio eksperimentas

Taisyklingam automatinės šablonų paieškos įrankio veikimui reikalingas šio darbo *4.4.1. Automatinė šabloninių situacijų paieška* aprašyto komponento *FrequentPatternsShell* darbo rezultatas (užpildyta *pattern_frequencies* duomenų bazės lentelė). Tam, kad *FrequentPatternsShell* komponento veikimo rezultatas būtų informatyvus ir pasiekiamas laiku, jo veikimui turi būti naudojamas optimalus analizuojamų sekų ilgis. Jau žinoma, kad sukurtas algoritmas atlieka 11^n užklausų į duomenų bazę, kur n – nagrinėjamų sekų ilgis. Esant tokiam algoritmo sudėtingumui yra ypač svarbu parinkti optimalų įvykių sekos ilgį – tokį, kad algoritmo vykdymo laikas nebūtų per didelis. Tačiau išrenkami duomenys turi būti informatyvūs ir pilni.

Norint išsiaiškinti koks yra optimalus sekų ilgis, reikia paskaičiuoti kaip dažnai duomenyse sutinkami įvairių ilgių šablonai. Tam tikslui yra atliekamas eksperimentas. Jo metu bus parinkta viena diena, kurios rungtynių įvykiai bus analizuojami. Parenkama diena su daugiausiai sužaistų rungtynių, kad gauti rezultatai būtų kaip įmanoma tikslesni.

Eksperimento vykdymo eiga:

1. Parenkama data, kurios metu buvo sužaista daugiausia rungtynių. Iš šiuo metu sistemoje esamų duomenų galima daryti išvadą, kad daugiausiai rungtynių buvo sužaista 2013-03-06, todėl eksperimento metu bus analizuojami šios dienos įvykiai.

2. Apskaičiuojama kiek kartų per pasirinktą dieną įvyko šabloninės situacijos. Kiekvienam įvykių sekos ilgiui šis skaičius apskaičiuojamas atskirai. 4 priede pateikiamas šiam skaičiavimui įvykdyta SQL užklausa bei jos rezultatas.
3. Ieškomas toks sekos ilgis, pradedant kuriuo atrastų šabloninių situacijų skaičius pradeda smarkiai mažėti. Matoma, kad kai sekos ilgis viršija 4 elementus, šabloninių situacijų dažnumas staigiai sumažėja. Rasta tik 13 pasikartojančių sekų, kurių ilgis yra 5. Toks mažas pasikartojimų skaičius parodo, kad sekos nėra dažnos ir labiau panašios į atsitiktinumą, negu į šablonines situacijas. Todėl jas interpretuoti kaip šablonus būtų klaidinga.
4. Įvertinus analizei reikiamų resursų skaičių bei šabloninių situacijų atsiradimo intensyvumą, pasirenkama riba, pradedant kuria šabloninių situacijų nebus ieškoma. Šis skaičius ir yra ieškomas optimalus sekų ilgis.

Atlikus eksperimentą buvo pastebėta, jog optimalus sekų ilgis šabloninių situacijų paieškai atlikti yra 4. Eksperimento vykdymo rezultatas pateiktas 4 priede.

5.1.2. Automatinės šablonų paieškos įrankis

Automatinės šablonų paieškos įrankis analizuoja komponento *FrequentPatternsShell* sukauptus ir agreguotus duomenis. Sukurto įrankio, skirto šių sukauptų duomenų analizei, veikimas susideda iš trijų etapų:

1. Vartotojas įveda dominantį laiko rėžį (datas *nuo* ir *iki*).
2. Įrankis, pasitelkus lentelėje *pattern_frequencies* sukauptais duomenimis, pagal nurodytas datas išrenka reikiamus duomenis: populiariausių rastų šablonų penkioliktuką ir šių šablonų intensyvumą laiko atžvilgiu (kurią dieną kuris iš šių šablonų kiek kartų buvo aptiktas).
3. Atliekama surinktų rezultatų vizualizacija (lentelė bei grafikas).

Šiai globaliai įvykių paieškai nėra numatyta papildomų konteksto sąlygų. Skenuojami duomenys grupuojami tik pagal rungtynes, kuriuose jie įvyko.

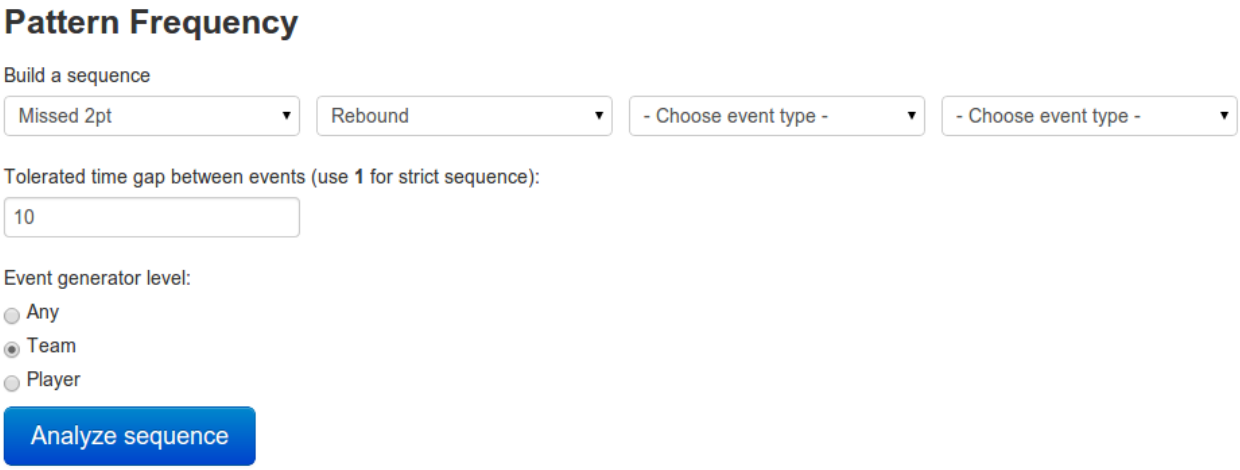
Pasinaudojant sukurtu įrankių šio eksperimento kontekste analizuojami įvykiai, sugeneruoti laiko rėžyje nuo 2013-03-01 iki 2013-03-12 imtinai. Šio eksperimento rezultatai pateikiami 5 ir 6 prieduose.

Iš 5 priede pateiktos lentelės duomenų galima pastebėti, jog pats populiariausias ir dažniausiai sutinkamas įvykių derinys yra [*nepataikytas dvitaškis*], [*atkovotas kamuolys*] (rastas 5041 šablono pakartojimas). Kaip buvo minėta šio darbo *4.6. Apribojimai* skyriuje, vykstant globaliai duomenų analizei nėra atsižvelgiama į įvykių kontekstą, t.y. išgaunami vienoje sekoje esantys įvykių tipai, nepriklausomai nuo to, kuri komanda arba žaidėjas juos sugeneravo. Todėl rezultatas yra lengvai paaiškinamas tuo faktu, kad po nepataikyto dvitaškio mėtimo kamuolys dažnai būna atkovojamas priešininkų komandos. Taip pat tarp dažniausiai pastebimų šabloninių situacijų egzistuoja sekos [*nepataikytas dvitaškis*], [*atkovotas kamuolys*], [*pataikytas dvitaškis*] (1617 pakartojimai) bei [*nepataikytas dvitaškis*], [*atkovotas kamuolys*], [*nepataikytas dvitaškis*] (1466 pakartojimai). Iš šių duomenų galima padaryti išvadą, jog komanda, atkovojus kamuolį po netaiklaus priešininkų mėtimo, dažniau sėkmingai užbaigia ataką, negu nesėkmingai.

6 priede pateiktas grafikas, kuriame pavaizduotas populiariausių šablonų pasiskirstymas laike kiekvienos dienos atžvilgiu. Galima pastebėti tendenciją, kad beveik visų kreivių reikšmės kinta proporcingai. Iš to galima daryti išvadą, kad ilguoju laikotarpiu populiariausių šablonų pakartojimo dažnumo santykis artėja prie konstantos. Taip pat skirtingų ilgių šablonai dažnai būna priklausomi vienas nuo kito, todėl, pavyzdžiui, dažniau pasitaikant šablonui < #8, #5, #2 >, dažniau pasitaikys ir šablonas < #8, #5 >.

5.2. Vartotojo parametrizuotų užklausų eksperimentai

Realizuojant šio darbo *4.4.2. Vartotojo suformuotos parametrizuotos užklausos* skyriuje sukurtą algoritmą, į sistemą buvo integruotas dar vienas naujas įrankis, skirtas įvykių analizei. Šis įrankis leidžia patogiai analizuoti sekų intensyvumą konkrečių rungtynių kontekste. Norint pradėti algoritmo vykdymą užtenka užpildyti formą rungtynių vidiniame puslapyje. Formos vaizdas pateiktas sekančiame paveikslėlyje.



25 pav. Parametrizuoto sekų paieškos algoritmo vykdymo įvesties forma

Pirmoje skiltyje įvedama ieškoma įvykių seka. Kaip buvo patvirtinta anksčiau vykdytuose eksperimentuose, ilgiausia seka, kurios prasminga ieškoti, susideda iš keturių elementų. Todėl šis įrankis taip pat riboja maksimalų sekos ilgį iki keturių elementų. Norint parinkti trumpesnę seką, likę laukai paliekami neužpildyti.

Kitas svarbus laukas – laiko atkarpa, nurodanti maksimalų tarpą tarp ieškomų įvykių sekoje. Laikas nurodomas sekundėmis. Pavyzdžiui, skaičius 10 nurodo, kad ieškoma tik tokių įvykių sekų, kurioje nurodyti įvykių tipai nutolę vienas nuo kito nedaugiau, negu per 10 sekundžių.

Trečia skiltis užpildymui – papildomas įvykių generatorių kontekstas. Šis parametras nusako ar visi įvykiai turi būti sukurti tos pačios komandos, ar to pačio žaidėjo, ar bet kurio rungtynių dalyvio. Smulkesnė įvykius generuojančių subjektų analizė pateikta šio darbo *4.3. Šabloninių situacijų išryškinimas* skyriuje.

5.2.1. Eksperimentai su žaidėjo subjektu

Šiame skyriuje bus aprašyti keli eksperimentai, kuriuose nagrinėjamų sekų generatorius visais atvejais yra vienas ir tas pats žaidėjas.

Pirmojo parametrizuotų užklausų eksperimento metu bus patikrinta ar algoritmo pateikiami duomenys yra validūs. Tam tikslui, pasinaudojant sistemoje esame rungtynių įvykių lentelėmis, bus parinktos kelios iš anksto žinomos situacijos aikštelėje ir patikrinta ar algoritmas gebės jas išskirti bendrame duomenų kontekste. Eksperimentui išrinktos 2014-01-22 vykusios rungtynės *Dallas Mavericks* prieš *Toronto Raptors*. Komandų rungtynių įvykių lentelės pateiktos 7 priede.

Kaip galima matyti iš 7 priede pateiktos rungtynių lentelės, žaidėjas *Valanciunas* ketvirto kėlinio metu, laiko momentu 7:21 atkovojo kamuolį. Vėliau pataikė dvitaškį (žalias keturkampis), o tada vėl atkovojo kamuolį (situacija pažymėta skaičiu 1). Laiko tarpas tarp šių įvykių nedidesnis, negu 30 sekundžių. Todėl į formą įvedus šiuos parametrus algoritmas turi atrasti šią situaciją ir, galbūt, kitų žaidėjų šiose rungtynėse sugeneruotas tokias pačias situacijas.

Eksperimento vykdymo rezultatas pateiktas 8 priede. Iš jo matoma, kad algoritmas aptiko šią situaciją ir ji buvo tokia vienintelė visų rungtynių metu.

7 priede skaičiumi 2 pažymėta situacija, kai žaidėjas *DeRozan* pelno 2 dvitaškius iš eilės. Atstumas tarp šių įvykių neviršija 40 sekundžių. Todėl, formoje įvedus dviejų dvitaškių iš eilės pataikymo seką bei laiko intervalą 40 sekundžių, rezultatuose turi atsispindėti ši pažymėta situacija. Šio eksperimento rezultatas pateiktas 9 priede. Iš jo galima matyti, kad algoritmas sėkmingai aptiko aukščiau aprašytą situaciją. Taip pat šių rungtynių eigoje tokį šabloną atitiko dar 3 sekos.

5.2.2. Eksperimentai su kitais įvykių subjektais

Nemažiau svarbu yra ištirti algoritmo elgseną, kai įvykius generuoja ne vienas žaidėjas, bet kiti subjektai – tos pačios komandos žaidėjai arba bet kokie žaidėjai aikštėje. 7 priede skaičiumi 3 pažymėta situacija, kai žaidėjas *Carter* nepataiko dvitaškio, o praėjus 6 sekundėms po šio įvykio

prasižengė kitas šios komandos žaidėjas *Calderon*. Įvykdžius užklausą yra grąžinama tik viena tokia situacija. Tačiau padidinus toleruojamą laiko tarpą iki 15 sekundžių, tokių įvykių sekų rungtynėse atrandama daug daugiau. Šio tyrimo rezultatas pateiktas 10 priede.

Paskutiniam eksperimentui, kuriuo metu situacijos bus analizuojami kontekstu, kuriame sekos įvykius gali sugeneruoti bet kokie rungtynių dalyviai, buvo pasirinkta tokia įvykių seka: [*nepataikytas dvitaškio metimas*], [*pražanga*], [*pataikytas baudos metimas*]. Kuo dažniau tokia seka aptinkama, tuo intensyvesnės rungtynės vyko tuo momentu. Jeigu iškarto po pražangos yra atliekamas baudos metimas tai reiškia, kad pražanga įvyko arba betarpiškai vykstant priešininko metimui arba tada, kai komanda yra surinkusi maksimalų leistiną komandinių pražangų kiekį. Tam, kad atrinkti labiausiai įtemptus momentus buvo parinktas nedidelė leistina laiko atkarpa – vos 10 sekundžių. Šios užklausos rezultatas pavaizduotas 11 priede. Iš lentelės galima matyti, kad daugiausia tokių situacijų, kai po nepataikyto metimo įvyksta pražanga, atsitinka ketvirtame kėlinyje. Tai reiškia, kad artėjant rungtynių pabaigai, žaidėjai pradeda agresyviau jungtis, ko pasėkoje gauna daugiau pražangų.

6. IŠVADOS

1. Atlikus reikiamus tyrimus ir algoritmų apžvalgą buvo sukurta SPADE algoritmo variacija, kuri pilnai pritaikyta darbui su sukurta sporto įvykių analizės ir vizualizacijos sistema SEA. Sukurtas algoritmas ir jo veikimui skirtos SQL užklausos padeda išspręsti šabloninių situacijų paieškos problemą.
2. Eksperimentinio tyrimo metu sukurtas algoritmas buvo pilnai realizuotas. Nauji įrankiai integruoti į SEA sistemą, kas leidžia patogiai analizuoti sistemoje esančius duomenis. Parametrizuotos algoritmo versijos atveju vartotojui suteikiama galimybė pačiam pasirinkti algoritmo veikimo parametrus.
3. Atlikus automatinės šablonų paieškos tyrimą, akivaizdžiai pastebima tendencija, kad skirtingų šablonų dažnumas skirtinguose laiko rėžiuose kinta proporcingai. Iš to galima daryti išvadą, kad ilguoju laikotarpiu populiariausių šablonų aptikimo dažnumo santykis artėja prie konstantos. Tai reiškia, kad pačios populiariausios šabloninės situacijos įvairiose rungtynėse aptinkamos beveik vienodu santykiu.
4. Iš tyrimo rezultatų matoma tiesioginė priklausomybė tarp kai kurių šabloninių situacijų. Atsiranda tokių atvejų, kai vieno šablono dažnumo grafiko kreivė visiškai atkartoja kito šablono aptikimo kreivę, tik su didesnėmis arba mažesnėmis reikšmėmis. Taip yra dėl to, kad skirtingų ilgių šablonai gali būti visiškai priklausomi vienas nuo kito. Tai tokie atvejai, kai ilgesnio šablono įvykių seka prasideda kitu, jau aptiktu šablonu. Galima pastebėti, kad jeigu seka A yra sekos B poaibis, tai ji visada bus aptinkama nemažiau kartų, negu seka B.
5. Tyrimo eigoje pastebėta, kad naujos SPADE algoritmo realizacijos veikimas kiekvienai įvykių grupei reikalauja daryti atskiras SQL užklausas. Egzistuoja sprendimai, leidžiantys sumažinti algoritmo veikimui reikalingų užklausų į duomenų bazę skaičių, tačiau jų realizacijai reikia atlikti papildomą analizę ir tyrimus.

7. LITERATŪRA

[1] Matej Perse, Matej Kristan, Janez Pers, and Stanislav Kovacic, “A Template-Based Multi-Player Action Recognition of the Basketball Game”, 2006. Prieiga internete < http://vision.fe.uni-lj.si/docs/matejp/CVBASE06_perse.pdf >

[2] Jin-Woo Kim, “The worth of sport event sponsorship: an event study”, prieiga internete < http://www.aabri.com/manuscripts/09382.pdf >

[3] Frane Erculj, Brane Dežman, Goran Vučkovic, Janez Perš, Matej Perše, Matej Kristan, “An Analysis Of Basketball Players’ Movements In The Slovenian Basketball League Play-offs Using The Sagit Tracking System”, 2008. Prieiga internete < http://facta.junis.ni.ac.rs/pe/pe200801/pe200801-08.pdf >

[4] Janez Perš, Stanislav Kovačič, Goran Vučkovič, “Analysis and pattern detection on large amounts of annotated sport motion data using standard SQL”, 2005. Prieiga internete < http://vision.fe.uni-lj.si/docs/janezp/1113.pdf >

[5] W3C, “SOAP Version 1.2 Part 0: Primer", prieiga internete < http://www.w3.org/TR/2001/WD-soap12-part0-20011217 >

[6] Cesare Pautasso, Olaf Zimmerman, Frank Leymann, “RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision”, 2008. Prieiga internete < http://www.jopera.org/docs/publications/2008/restws >

[7] W3C, "Relationship to the World Wide Web and REST Architectures", prieiga internete < http://www.w3.org/TR/ws-arch/#relwwwrest >

[8] Jung-Wei Chen, Jiajie Zhang, “Comparing Text-based and Graphic User Interfaces for Novice and Expert Users”, 2007. Prieiga internete < http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2655855/ >

[9] StatTrak for Basketball 3.0, prieiga internete < http://www.allprosoftware.com/bb >

[10] Banjamin Wiederkehr, “13 Javascript Libraries for Visualizations”, 2009. Prieiga internete < http://datavisualization.ch/tools/13-javascript-libraries-for-visualizations/ >

[11] Nic James, “Performance analysis to improve sport performance”, 2009. Prieiga internete < http://www.altorendimiento.com/congresos/entrenamiento/279-performance-analysis-to-improve-sport-performance >

[12] B. A. Boghossian and S. A. Velastin. “Motion-based machine vision techniques for the management of large crowds”, 1999. Prieiga internete < http://dilnxsrv.king.ac.uk/papers/ices99.pdf >

[13] Stephen S. Intille and Aaron F. Bobick, “A Framework for Recognizing Multi-Agent Action from Visual Evidence”, 1999. Prieiga internete < http://pubs.media.mit.edu/pubs/papers/TR-489.pdf >

[14] ACM SIGKDD, "Data Mining Curriculum: A Proposal", 2004. Prieiga internete < http://www.cs.uiuc.edu/~hanj/kdd_curriculum.pdf >

[15] About.com, “Datamining”, prieiga internete < http://databases.about.com/od/datamining/a/datamining.htm >

[16] Fayyad, Usama; Piatetsky-Shapiro, Gregory; Smyth, Padhraic, "From Data Mining to Knowledge Discovery in Databases", 1996. Prieiga internete < http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf >

[17] Elovici, Yuval; Braha, Dan, “A Decision-Theoretic Approach to Data Mining”, 2003. Prieiga internete < http://necsi.edu/affiliates/braha/IEEE_Decision_Theoretic.pdf >

[18] Jason Frand, “Data Mining: What is Data Mining?”, 1996. Prieiga internete <http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm >

[19] Dartfish EasyTag. Prieiga internete < http://www.dartfish.com/en/software/dartfish-easytag/index.htm >

[20] Dartfish Software. Prieiga internete < http://www.dartfish.com/en/software >

[21] Sportsec SportsCode. Prieiga internete < http://www.sportstec.com/products-sportscode >

[22] IPTS (International Press Telecommunications Council) tinklapis, prieiga internete < https://www.iptc.org/site/News_Exchange_Formats/SportsML-G2 >

[23] Sports Standards Alians, prieiga internete < http://sportsdb.org/sm >

[24] IPTS SportsML Developer Site, prieiga internete < http://dev.iptc.org/SportsML >

[25] SPMF, “A Sequential Pattern Mining Framework”, prieiga internete <http://www.philippe-fournier-viger.com/spmf/index.php?link=algorithms.php >

[26] Mabroukeh, N. R.; Ezeife, C. I., "A taxonomy of sequential pattern mining algorithms", 2010. Prieiga internete < http://dl.acm.org/citation.cfm?doid=1824795.1824798 >

[27] Zaki, M. J., "Spade: An efficient algorithm for mining frequent sequences", 2001. Prieiga internete < http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.113.6042 >

[28] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression", 2007. Prieiga internete < http://www.stat.washington.edu/courses/stat527/s14/readings/Altman_AmStat_1992.pdf >

[29] E. Fix, J. L. Hodges, “Discriminatory analysis, nonparametric discrimination: Consistency properties”, 1970. Prieiga internete < http://www.dtic.mil/dtic/tr/fulltext/u2/a800276.pdf >

[30] Jug, M., Perš, J., Dežman, B., Kovačič, S., “Trajectory based assessment of coordinated human activity.”, 2003. Prieiga internete < http://vision.fe.uni-lj.si/docs/janezp/jugICVS2003.pdf >

[31] Jiawei Han; Hong Cheng; Dong Xin; Xifeng Yan, “Frequent pattern mining: current status and future directions”, 2006. Prieiga internete < http://www.cs.ucsb.edu/~xyan/papers/dmkd07_frequentpattern.pdf >

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

CakePHP - PHP programavimo kalbos PĮ karkasas.

Chronologinė įvykių seka (angl. *play-by-play*) – detalizuota ir eiliškumu pasižyminti informacija apie sportinius įvykius.

Dažnai pasitaikančių šablonų gavyba (angl. *Frequent Pattern Mining*) – tai procesas, kurio metu gaunamos statistiškai dažniausiai duomenų imtyje pasitaikantys sekos.

Duomenų gavyba (angl. *data mining*) – tai skaičiavimo procesas, kurio metu didesniuose duomenų kiekiuose bandoma išryškinti šablonus ir modelius

K-D medis - tai duomenų struktūra su erdvės padalinimu, siekiant surikiuoti taškus k-matėje erdvėje.

kNN (angl. *k Nearest Neighbours algorithm*) - algoritmas, skirtas spręsti klasifikacijos ir regresijos klasių problemas duomenų gavyboje.

Persimokymas (angl. *overfitting*) - duomenų gavybos algoritmams būdingas bruožas, kai atrandami tokie šablonai, kurie bendrame visų duomenų kontekste atsiranda nedažnai

PĮ karkasas (angl. *Software framework*) - programų sistemos struktūra, palengvinanti didesnių sistemų kūrimą bei jos modulių ir komponentų apjungimą tarpusavyje.

R medis - tai duomenų struktūra, skirta daugiadimensinių duomenų saugojimui ir indeksavimui.

REST (angl. *REpresentational State Transfer*) - tai decentralizuotos, nesaugantis būsenų programinės įrangos architektūros kūrimo stilius, kuris veikia pasinaudojant HTTP protokolu.

SOAP (angl. *Simple Object Access Protocol*) – protokolas, paprastai kartu su HTTP protokolu naudojamas XML kalbos formato struktūrizuotiems pranešimams siųsti

SPADE (angl. *Sequential PAttern Discovery using Equivalence classes*) - algoritmas, skirtas duomenų šablonų paieškai.

Sporto įvykiu analizės ir vizualizavimo sistema SEA (angl. *Sports Events Analysis system*) – tai Vitalijaus Suckel magistro studijų metu sukurta programinė įranga, leidžianti kaupti ir analizuoti sportinių įvykių duomenis.

SportsML – tai naujienų apsikeitimų standartas, paremtas XML žymėjimo kalba

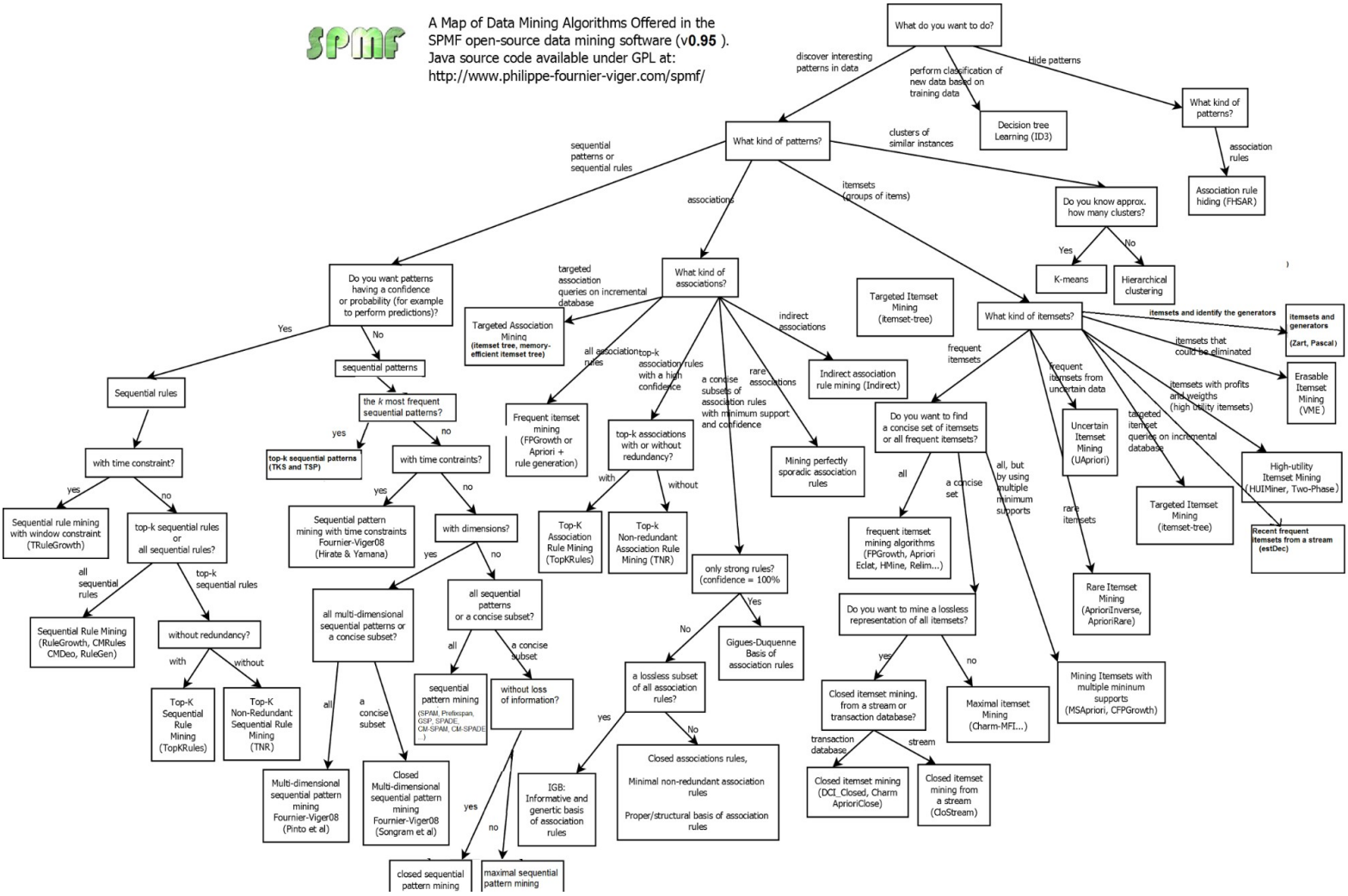
Sudėtinis įvykis – tai tokia situacija, kai per vieną rungtynių sekundę atsitiko daugiau negu vienas, tariamai vienas nuo kito priklausomas įvykis (pvz. pražanga ir baudos mėtimai, žaidėjų apsikeitimai ir pan.)

SQL (angl. *Structured Query Language*) – populiariausia iš šiuo metu naudojamų kalbų, skirtų aprašyti duomenis ir manipuliuoti jais reliacinių duomenų bazių valdymo sistemose

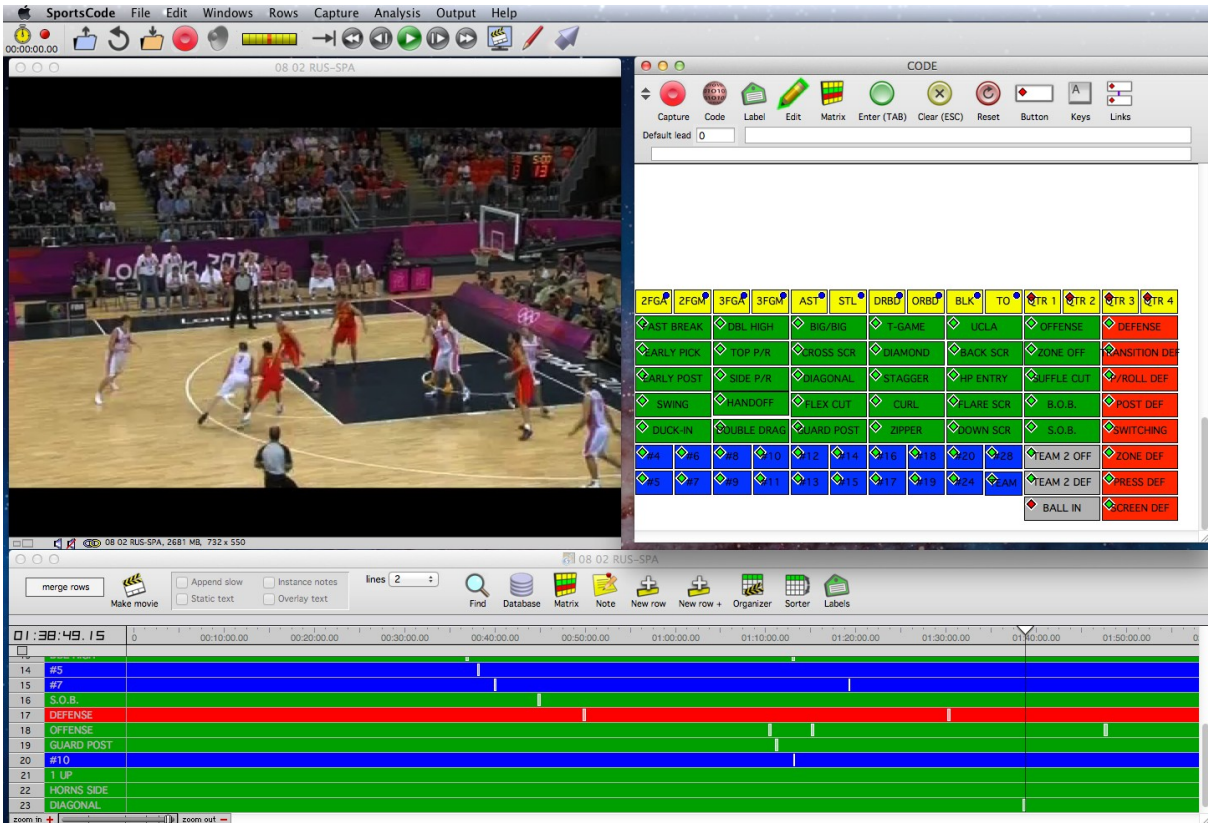
Web scraping – technika, leidžianti išgauti duomenis iš tinklapių HTML išeities kodų.

9. PRIEDAI

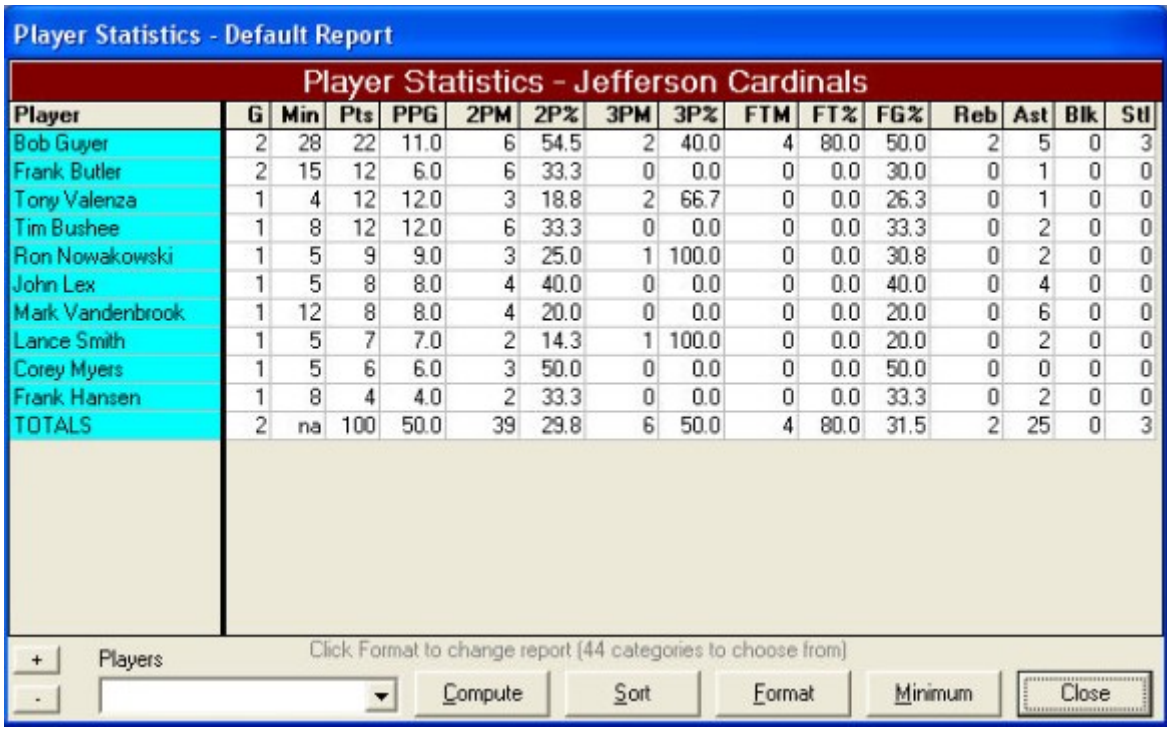
1 priedas. Duomenų gavybai skirtų algoritmų žemėlapis



2 priedas. SportsCode programos darbo langas



3 priedas. StatTrak for Basketball programos langas



Player Statistics - Default Report

Player Statistics - Jefferson Cardinals

Player	G	Min	Pts	PPG	2PM	2P%	3PM	3P%	FTM	FT%	FG%	Reb	Ast	Blk	Stl
Bob Guyer	2	28	22	11.0	6	54.5	2	40.0	4	80.0	50.0	2	5	0	3
Frank Butler	2	15	12	6.0	6	33.3	0	0.0	0	0.0	30.0	0	1	0	0
Tony Valenza	1	4	12	12.0	3	18.8	2	66.7	0	0.0	26.3	0	1	0	0
Tim Bushee	1	8	12	12.0	6	33.3	0	0.0	0	0.0	33.3	0	2	0	0
Ron Nowakowski	1	5	9	9.0	3	25.0	1	100.0	0	0.0	30.8	0	2	0	0
John Lex	1	5	8	8.0	4	40.0	0	0.0	0	0.0	40.0	0	4	0	0
Mark Vandenbrook	1	12	8	8.0	4	20.0	0	0.0	0	0.0	20.0	0	6	0	0
Lance Smith	1	5	7	7.0	2	14.3	1	100.0	0	0.0	20.0	0	2	0	0
Corey Myers	1	5	6	6.0	3	50.0	0	0.0	0	0.0	50.0	0	0	0	0
Frank Hansen	1	8	4	4.0	2	33.3	0	0.0	0	0.0	33.3	0	2	0	0
TOTALS	2	na	100	50.0	39	29.8	6	50.0	4	80.0	31.5	2	25	0	3

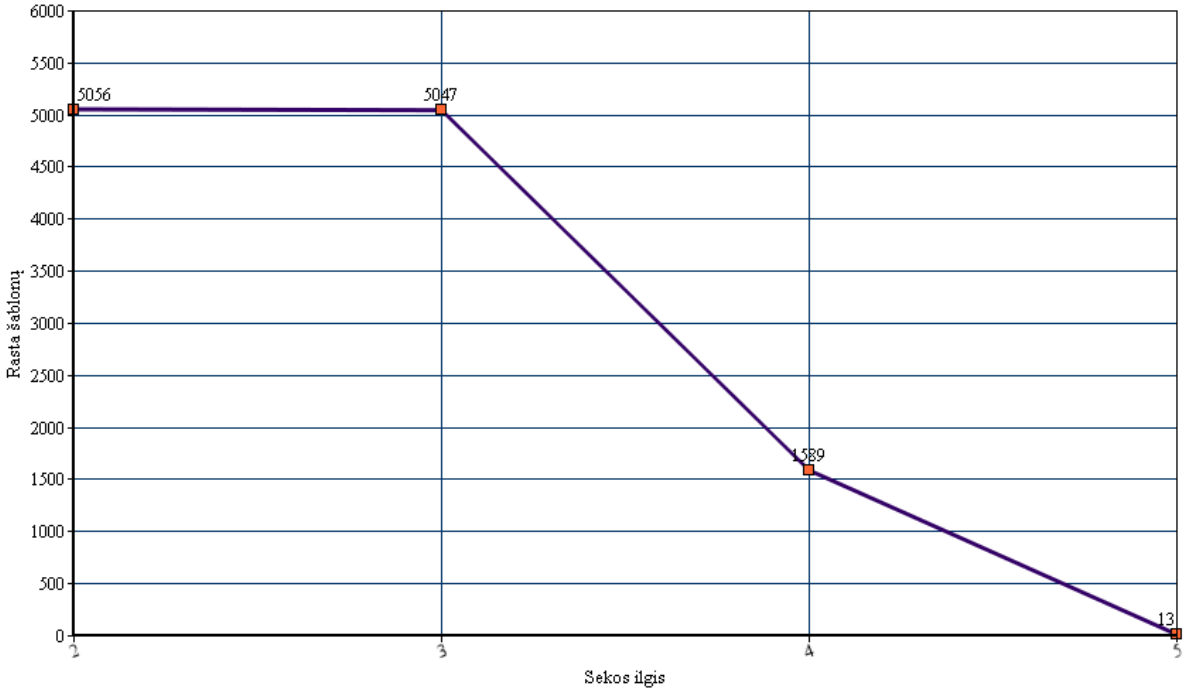
Players

Click Format to change report (44 categories to choose from)

Compute Sort Format Minimum Close

4 priedas. Optimalaus šablono sekos ilgio paieškos užklausa ir rezultatas

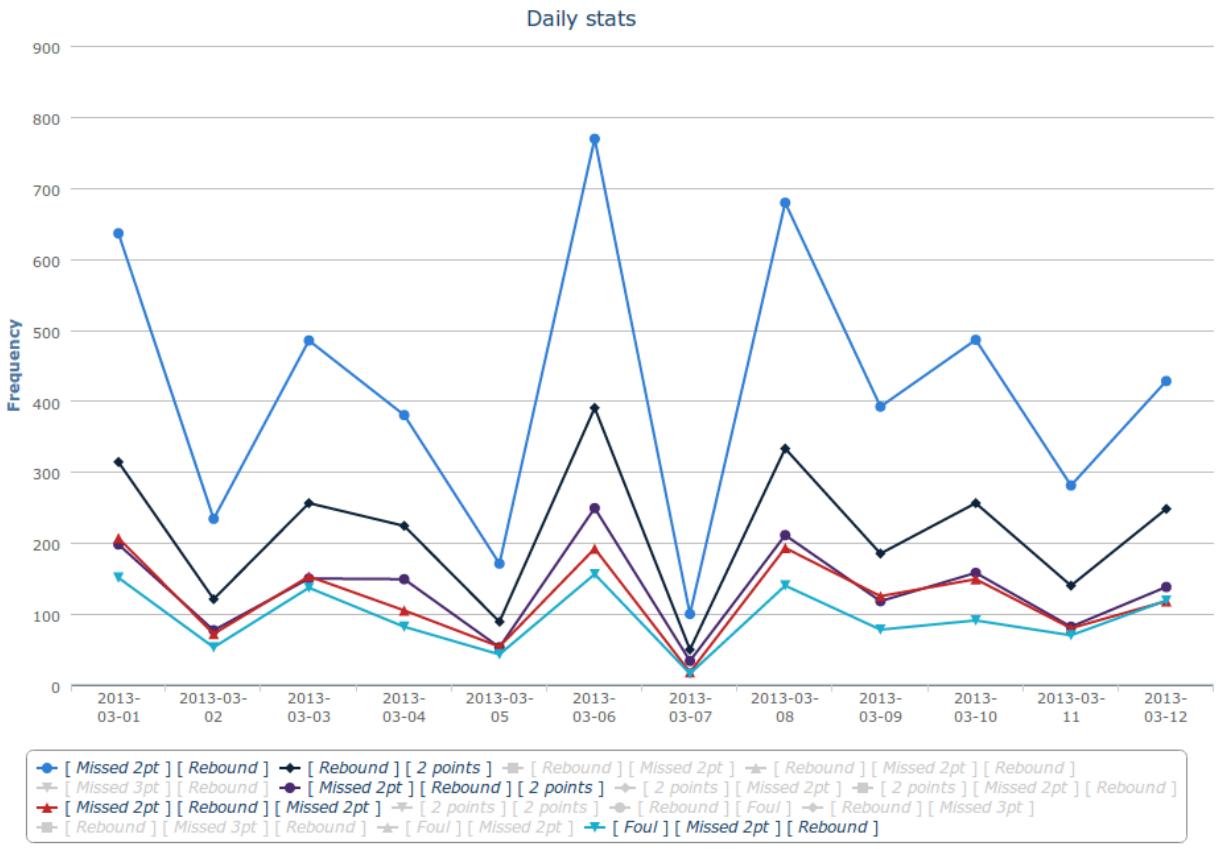
```
SELECT
 SUM(`count`) AS cnt,
 (LENGTH(sequence) - LENGTH(REPLACE(sequence, ',', '') ) + 1)
  AS seq_length
FROM pattern_frequencies
WHERE `date` = '2013-03-01'
GROUP BY seq_length
ORDER BY `cnt` DESC
```



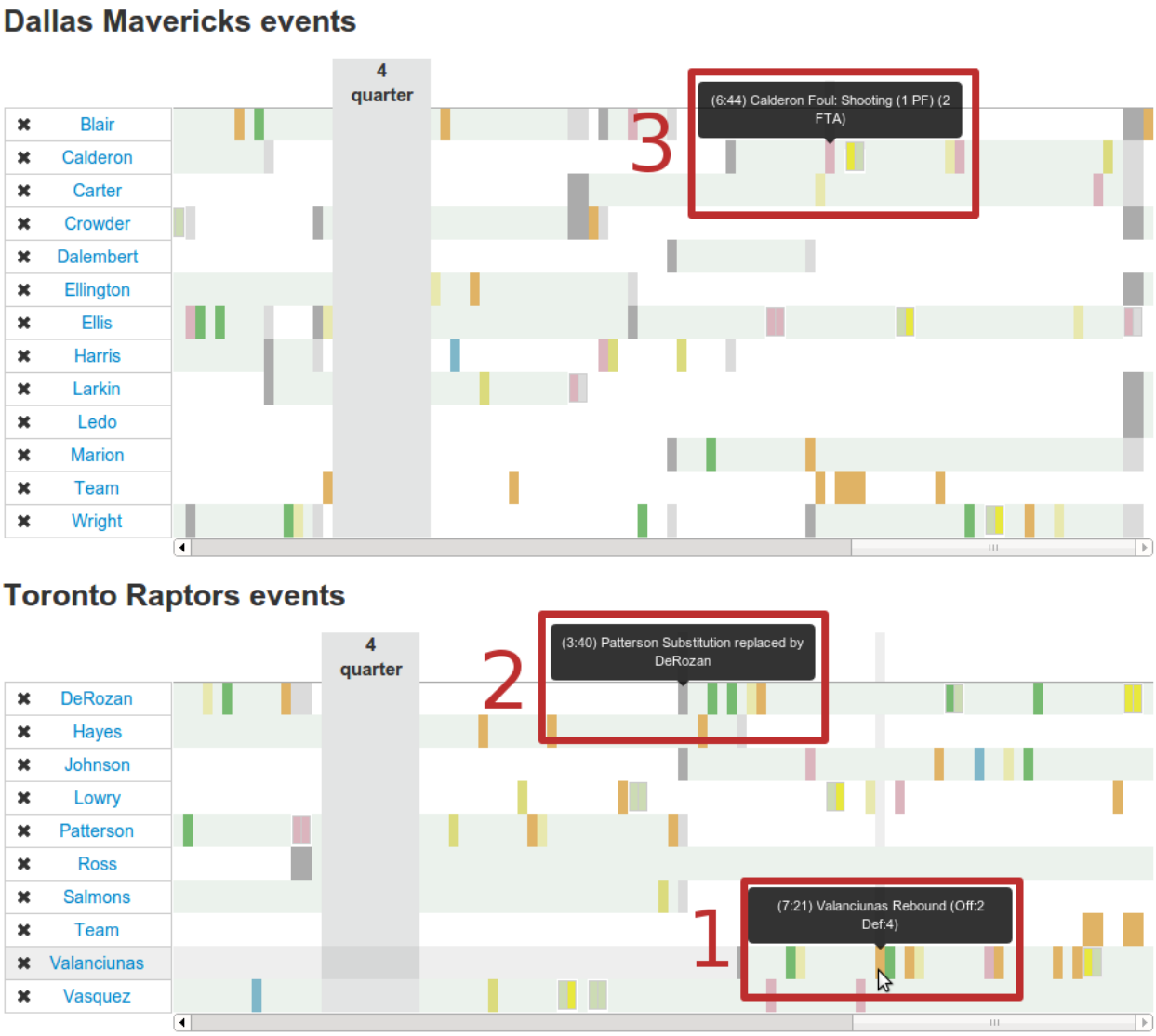
5 priedas. Automatinio šablonų paieškos įrankio darbo rezultatas (dažniausiai sutinkamų šablonų penkioliktukas)

Šablonas	Sekos ilgis	Dažnumas
[*Missed 2pt*] [*Rebound*]	2	5041
[*Rebound*] [*2 points*]	2	2606
[*Rebound*] [*Missed 2pt*]	2	2585
[*Rebound*] [*Missed 2pt*] [*Rebound*]	3	2585
[*Missed 3pt*] [*Rebound*]	2	2149
[*Missed 2pt*] [*Rebound*] [*2 points*]	3	1617
[*2 points*] [*Missed 2pt*]	2	1578
[*2 points*] [*Missed 2pt*] [*Rebound*]	3	1576
[*Missed 2pt*] [*Rebound*] [*Missed 2pt*]	3	1466
[*2 points*] [*2 points*]	2	1451
[*Rebound*] [*Foul*]	2	1408
[*Rebound*] [*Missed 3pt*]	2	1218
[*Rebound*] [*Missed 3pt*] [*Rebound*]	3	1217
[*Foul*] [*Missed 2pt*]	2	1136
[*Foul*] [*Missed 2pt*] [*Rebound*]	3	1136

6 priedas. Automatinio šablonų paieškos įrankio darbo rezultatas (šablonų aptikimo intensyvumas dienomis)



7 priedas. 2014-01-22 vykusių rungtynių Dallas Mavericks prieš Toronto Raptors įvykių lentelės



8 priedas. Jono Valančiūno sukurtos situacijos analizės rezultatas

Pattern Frequency Analysis

[*Rebound*] [*2 points*] [*Rebound*]

Team	Player	Rebound	2 points	Rebound
Toronto Raptors	Valanciunas	07:24 (quoter 4)	07:25	07:40

9 priedas. Eksperemento su dviems dvitaškių pataikymais iš eilės rezultatas

Pattern Frequency Analysis

[*2 points*] [*2 points*]

Team	Player	2 points	2 points
Toronto Raptors	Johnson	02:41 (quoter 2)	03:14
Dallas Mavericks	Ellis	01:48 (quoter 3)	02:26
Dallas Mavericks	Ellis	08:57 (quoter 3)	09:36
Toronto Raptors	DeRozan	04:16 (quoter 4)	04:52

10 priedas. Eksperimento su skirtingais komandos žaidėjais rezultatas

Pattern Frequency Analysis

[*Missed 2pt*] [*Foul*]

Team	Missed 2pt	Foul
Dallas Mavericks	04:07 (quoter 1)	04:18
Toronto Raptors	06:01 (quoter 2)	06:13
Toronto Raptors	02:05 (quoter 3)	02:19
Toronto Raptors	05:31 (quoter 4)	05:42
Toronto Raptors	06:31 (quoter 4)	06:32
Dallas Mavericks	06:41 (quoter 4)	06:47
Dallas Mavericks	08:16 (quoter 4)	08:22

11 priedas. Eksperimento su bet kokiais rungtynių dalyviais rezultatas

Pattern Frequency Analysis

[*Missed 2pt*] [*Foul*] [*Foul throw*]

Missed 2pt	Foul	Foul throw
10:55 (quoter 2)	10:56	10:56
01:44 (quoter 4)	01:52	01:52
06:41 (quoter 4)	06:47	06:47
08:16 (quoter 4)	08:22	08:22