KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Gediminas Rudaitis

Įvykiais grindžiamų informacinių sistemų modeliavimo ir realizavimo metodika

Magistro darbas

Darbo vadovė
prof. Lina Nemuraitė

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Gediminas Rudaitis

Įvykiais grindžiamų informacinių sistemų modeliavimo ir realizavimo metodika

Magistro darbas

Recenzentas doc. dr. Stasys Maciulevičius
2008-01-

Vadovė prof. Lina Nemuraitė
2008-01-14

Atliko IFM–2/4 gr. studentas
Gediminas Rudaitis
2008-01-14

Kaunas, 2008

SUMMARY

Methodology for Modelling and Implementation of Event-driven Information Systems

Nowadays the terms SOA (Service Oriented Architecture) and event driven systems are often used when talking about systems with large scalability, performance and interoperability. In fast changing business world it is vital to have systems that could be easily changed whenever new functionality is needed. Using service oriented architecture companies can easier adapt its business process to changed business rules.

Even though there are a lot of articles and publications about event driven systems and their advantages, but the system architect who decides to build an event driven system is facing the problem: how to model and describe events of the system. Obviously, that there are no popular and widely accepted methodology for event driven system modelling.

The goal of this work is to develop a methodology for modelling an event driven systems. This methodology describes the metamodel for event modelling and the usage of the event model in system development life cycle.

The developed methodology for modelling and implementation of event-driven information systems was used for making a model of publications portal prototype. Usage of this methodology in system design process adds more clarity, because all the events, which can be generated in the system and their types, are described in one diagram – event model. The event model elements can be used in other model diagrams, such as sequence or state machine diagrams.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

Įvadas

Pastaruoju metu informacinių technologijų pasaulyje dažnai minimos paslaugomis ir įvykiais grindžiamų sistemų architektūros. Greitai besikeičiančiame verslo pasaulyje įmonėms yra aktualu turėti sistemas, kurios greitai ir lengvai būtų priderintos prie besikeičiančių verslo procesų ir sumažintų tokių sistemų kūrimo ir palaikymo išlaidas. Naudodamos paslaugomis grindžiamą architektūrą, įmonės gali lanksčiai tvarkyti savo veiklą ir kurti sistemas, nekeisdamos turimos techninės ar programinės įrangos.

Nors literatūroje yra gausu straipsnių apie įvykiais grindžiamas sistemas ir jų privalumus, tačiau sistemos architektui, nusprendusiam sukurti įvykiais grindžiamos sistemos projektą, iškyla aktualus klausimas: kaip aprašyti ir modeliuoti įvykius, kurie ateina iš išorės arba sugeneruojami pačios sistemos ar vartotojų veiksmų su sistema metu. Akivaizdu, kad šiuo metu nėra populiarios metodikos, kuri apibrėžtų, kaip reikia atlikti įvykių modeliavimą ir jį panaudoti informacinių sistemų kūrimo procese.

Kita svarbi problema, su kuria susiduria programuotojai, yra įvykių apdorojimo ir generavimo mechanizmų realizavimas taikomųjų uždavinių serveriuose. Yra svarbu ne tik žinoti, kaip modeliuoti įvykiais grįstas sistemas, bet ir kaip geriau jas realizuoti.

Todėl šio tyrimo sritis yra įvykiais grindžiamų sistemų modeliavimo metodai ir realizavimo technologijos, tyrimo objektas – įvykiais grindžiamų informacinių sistemų kūrimo procesas.

Magistrinio darbo tikslas: įgalinti informacinių sistemų projektuotojus kurti lanksčias pokyčiams informacines sistemas, sukuriant įvykiais grindžiamų sistemų modeliavimo ir realizavimo metodiką.

Darbo uždaviniai:

- išanalizuoti sistemų, kuriose komponentai bendradarbiauja siųsdami pranešimus, modeliavimo metodus;
- išanalizuoti technologijas, kuriomis būtų galima realizuoti įvykių generavimą ir apdorojimą;
- pagal analizės rezultatus sudaryti įvykių metamodelį, kurį būtų galima taikyti įvykiais grindžiamoms sistemoms modeliuoti;
- sudaryti įvykiais grindžiamų sistemų kūrimo metodiką;
- įvertinti sudaryto metamodelio ir metodikos tinkamumą, jų pagrindu sukuriant ir realizuojant publikacijų portalo prototipą.

Klausimai, į kuriuos turi atsakyti šis tyrimas:

- kokie yra metodai modeliuoti įvykiais grindžiamas sistemas;
- kaip klasifikuoti ir modeliuoti įvykius tokiose sistemose;

- kaip taikyti įvykių modelį skirtinguose kūrimo etapuose;
- kokiomis priemonėmis realizuoti įvykių generavimą ir apdorojimą taikomųjų uždavinių serveriuose;
- kaip sukurti konkrečią įvykiais grindžiamą sistemą – publikacijų portalą;
- kaip publikacijų portale skaičiuoti vartotojų reitingus pagal jų aktyvumą.

Šiame magistriniame darbe yra pateikiama įvykiais grindžiamų informacinių sistemų modeliavimo ir realizavimo metodika. Ši metodika apibrėžia, kaip identifikuoti įvykius, sudaryti įvykių modelius ir juos panaudoti sistemų projektavimo procese. Taip pat yra išanalizuotos įvykių generavimo ir apdorojimo technologijos taikomųjų uždavinių serveriuose.

Pagal sudarytą metodiką pateikiamas publikacijų portalo prototipo projektas. Šiame portalo prototipe vartotojai gali registruotis ir publikuoti savo straipsnius, parsisiųsti kitų autorių straipsnius, juos vertinti bei rašyti komentarus. Sistema automatiškai parenka straipsnio recenzentus ir siunčia jiems pasiūlymus rašyti recenzijas; recenzentas gali priimti arba atmesti pasiūlymus. Funkcijų automatizavimas sumažina sistemos administravimo darbų apimtis ir reikalauja mažiau žmogaus darbo laiko.

Portalo vartotojai pagal jiems priskirtas roles turi atitinkamas teises. Vartotojų rangavimą pagal jų aktyvumą, t.y. parašytų straipsnių, recenzijų, komentarų kiekį ir pan., atlieka pati sistema. Vartotojo aktyvumo įvertinimui buvo sukurtas sistemos vartotojų rangavimo algoritmas. Didžiausią reitingą turintys vartotojai yra sistemos administratoriai; jiems leidžiama naudoti daugiausiai sistemos funkcijų. Sistemos administratoriai gali sukurti naujas publikacijų temas, pašalinti komentarus ir pan. Paprastas registruotas sistemos vartotojas gali publikuoti straipsnius, juos vertinti bei būti recenzentu.

Pirmame magistrinio darbo skyriuje „Įvykiais pagristų sistemų modeliavimo ir realizavimo galimybių analizė“ analizuojama tiriama problema ir analizuojami jos galimi sprendimo būdai. Šiame skyriuje aprašyta įvykių klasifikacija, kuri vėliau panaudojama sudarant įvykių metamodelį. Poskyryje „Įvykių modeliavimo metodų analizė“ aprašomi literatūroje dažniausiai minimi įvykių modeliavimo metodai, o poskyryje „Įvykių realizavimo technologijų analizė“ galimos įvykių realizavimo technologijos.

Kalbant apie įvykiais grindžiamas sistemas, dažnai minima ir tarpinė programinė įranga, kuri yra naudojama pranešimų siuntimui tarp komponentų. Poskyryje „Pranešimų siuntimas apie sužadintus įvykius“ yra apžvelgiamos tokios programinės įrangos panaudojimo galimybės ir pranešimų siuntimo būdai.

Antrajame skyriuje aprašomas įvykių metamodelis, sudarytas remiantis atliktos analizės rezultatais. Sudarant įvykių metamodelį, remtasi SARI sistemoje naudojamu įvykių modeliu [2] bei AOR (angl. *Agent–Object–Relationship*) modeliavimu [4], [5]. SARI sistemos įvykių modelis yra

tinkamas aprašyti įvykių tipams ir jų atributams, o AOR modelis – įvykius generuojantiems ir apdorojantiems komponentams. Metamodelis sudarytas panaudojant abiejų modeliavimo metodikų stipriąsias savybes.

Įvykių modelio, sudaryto atsižvelgiant į antrame skyriuje pateiktą metamodelį, panaudojimas informacinių sistemų kūrimo procese aprašytas skyriuje „Įvykių modelio panaudojimas informacinių sistemų projektavimo procese“. Šiame skyriuje aprašoma, kaip identifikuoti įvykius iš panaudojimo atvejų diagramų, aprašyti įvykių sukėlėjus ir pranešimų gavėjus klasių diagramose, pavaizduoti pranešimus sekų diagramose bei panaudoti aprašytus įvykius būsenų diagramose. Taip pat šiame skyriuje aprašomos rekomendacijos, kaip ir kokiuose informacinių sistemų kūrimo etapuose sudaryti ir panaudoti įvykių modelį, t.y. šiame skyriuje aprašoma įvykiais grindžiamų sistemų modeliavimo metodika.

Ketvirtajame skyriuje „Publikacijų portalo prototipo reikalavimų analizė ir specifikacija“ trumpai aprašomas pavyzdinis veiklos procesas, kuris turi būti kompiuterizuotas sukuriant portalą, ir išskiriami sistemos vartotojų tipai. Kiekvienam vartotojų tipui sudaryti panaudojimo atvejai yra specifikuojami lentelėmis ir sekų diagramomis. Šiame skyriuje pagal pateiktos metodikos rekomendaciją sudaromas pradinis įvykių modelis, kuris aprašo pranešimus, jų gavėjus, įvykių sukėlėjus. Šis modelis nėra tikslus ir yra papildomas sistemos projektavimo etape, kuomet tiksliai nurodomas pranešimų turinys, pranešimų siuntėjus ir gavėjus realizuojančios klasės, taip pat operacijos, kurias vykdant yra sužadinami atitinkami įvykiai.

Skyriuje „Publikacijų portalo prototipo projektas“ pateikiama kuriamos sistemos architektūra: vartotojo sąsają ir veiklos logiką realizuojančios klasės. Poskyryje „Detalus projektas“ aprašomos modelio diagramos, kurios atspindi klasių realizavimą pasirinkta Java EE 5.0 technologija. Šiame poskyryje taip pat aptariamos įvykių generavimo ir programinio agento realizacijos šios technologijos galimybėmis. Java EE 5.0 pasirinkta todėl, kad analizės metu nustatyta, jog ši technologija leidžia atlikti visų įvykių tipų generavimą ir apdorojimą, pranešimų siuntimą tarp komponentų panaudojant *JMS* (angl. *Java Message Service*) tarpinę programinę įrangą.

Sudarius klasių diagramas ir identifikavus visas klasių operacijas, poskyryje „Sistemos įvykių modelis“ yra pateikiamas detalus įvykių modelis, sudarytas remiantis trečiajame skyriuje pateiktu pradiniu įvykių modeliu, papildant jo elementus atributais, kurie nusako klases ar komponentus, realizuojančius pranešimų gavėjus ir įvykių sukėlėjus. Detalaus įvykių modelio sudarymas, remiantis reikalavimų analizės ir specifikacijos etape sudarytu modeliu, yra viena iš pateiktos metodikos rekomendacijų.

Sudaryto įvykių modelio elementai yra panaudojami modeliuojant sistemos elgseną: sudarant sekų ir būsenų diagramas. Aprašyti įvykių sukėlėjai, pranešimai ir pranešimų gavėjai yra

panaudojami sekų diagramose, o įvykiai būsenų diagramose, kuomet vaizduojamas objekto būsenos pasikeitimas esant tam tikram įvykiui.

Poskyryje „Vartotojų rangavimo algoritmas“ pateikiamas vartotojų aktyvumo vertinimo algoritmas. Pagal šį algoritmą vartotojui suteikiamas reitingas ir atitinkama rolė.

Šeštajame skyriuje „Metodikos įvertinimas remiantis publikacijų portalo prototipu“ trumpai aprašomas sukurto publikacijų portalo prototipas, jo veikimas ir įvertinama metodika, pagal užsiduotus kokybinius kriterijus.

Septintajame skyriuje pateikiami darbe naudojami terminai ir santraukos.

Darbo pabaigoje pateikiamos išvados ir literatūros, kuria remtasi rašant magistrinį darbą, sąrašas.

Darbo tematika parengtas straipsnis, kuris pateiktas konferencijai „Informacinės technologijos 2008“.

1. Įvykiais pagristų sistemų modeliavimo ir realizavimo galimybių analizė

Šiuo metu daugumos informacijos sistemų kūrimas apima atskirų, jau realizuotų komponentų integravimą, ir naujo funkcionalumo sukūrimą, o ne naujos sistemos kūrimą nuo pradžių. Šiems uždaviniams spręsti panaudojamos į paslaugas orientuotų ir įvykiais grindžiamų sistemų architektūros. Toks sprendimo metodas leidžia sukurti sistemas, kuriose komponentai yra susieti silpnais ryšiais, ir taip padidinti sistemos plečiamumo galimybes, darbo spartą ir greitą sistemų pritaikymą prie besikeičiančių verslo poreikių.

Todėl šio tyrimo sritis yra įvykiais grindžiamų sistemų modeliavimo ir realizavimo galimybės, jų privalumai ir trūkumai.

Magistrinio darbo tikslas: įgalinti informacinių sistemų projektuotojus kurti lanksčias pokyčiams informacines sistemas, sukuriant įvykiais grindžiamų sistemų modeliavimo ir realizavimo metodiką.

Šiame skyriuje analizuojami galimi įvykiais grindžiamų sistemų modeliavimo metodai, pateikiama įvykių tipų analizė ir jų skirstymas į grupes, apžvelgiamos technologijos ir programavimo priemonės, kuriomis galima realizuoti kiekvieno tipo įvykius. Analizės rezultatai panaudojami kuriant metamodelį, kurio pagrindu galima sudaryti įvykių, generuojamų įvykiais grindžiamose sistemose, modelį.

1.1. Įvykiais pagrįstos sistemos

Įvykiais pagristos sistemos yra tokios sistemos, kuriose pranešimų siuntėjai (angl. *message producers*) reaguodami į vidinius ar išorinius įvykius sukuria pranešimus ir siunčia juos pranešimų gavėjams (angl. *message consumers*), kurie tuos pranešimus apdoroja. Pranešimų siuntėjais gali būti informacinių sistemų komponentai, duomenų bazių trigeriai, jutikliai realaus laiko sistemose; priimtus pranešimus gali apdoroti komponentai, duomenų bazės ir t.t. [2], [12].

Įvykiais pagrįsta sistema taip pat gali būti apibrėžta kaip sistema, kurioje bendradarbiaujantys komponentai neturi tiesioginio ryšio vienas su kitu. Jie bendradarbiauja siųsdami pranešimus [3].

Įvykių apdorojimas ir pranešimų siuntimas kuriant informacijos sistemas yra naudojamas jau keletą metų. Kartu su į paslaugų architektūra (angl. *service-oriented architecture*), įvykiais grindžiamos sistemos tampa svarbia dalimi kuriant verslo informacines sistemas. Pagrindiniai tikslai, kurių siekiama naudojant įvykiais grindžiamą architektūrą, yra sukurti lengvai išplečiamas sistemas, kurias būtų galima greitai priderinti prie besikeičiančių verslo procesų ir kurias būtų lengva papildyti pasikeitus kliento poreikiams. Šių tikslų pasiekimas reikalauja lankstumo, kurio

negali suteikti sistemos, sukurtos panaudojant standartinę objektinę technologiją, kuriose objektai ar komponentai bendradarbiauja kviesdami metodus ir grąžindami rezultatus [1].

Pagrindinis skirtumas tarp įvykiais grindžiamos sistemos ir įprastos sistemos, kurioje komponentai (ar objektai) bendradarbiauja kviesdami vienas kito metodus ir grąžindami reikšmes, yra tas, kad įvykiais grindžiamoje sistemoje komponentai nežino apie vienas kito egzistavimą. [3], [9]. Jiems nebūtina žinoti, kokius interfeisus realizuoja konkretus komponentas ir kokie veiklos logiką realizuojantys metodai gali būti iškviesti; pakanka nusiųsti pranešimą kitam komponentui, kuris apdorodamas pranešimą įvykdys atitinkamus metodus. Šis sistemų kūrimo metodas leidžia sukurti sistemas, kuriose komponentai yra sujungti silpnais ryšiais (angl. *loosely-coupled*). Tokios sistemos yra lengviau išplečiamos, veikimo sparta yra greitesnė, jas lengviau palaikyti ir testuoti.

Norint sukurti įvykiais grindžiamą sistemą yra būtina tarpinė programinė įranga (angl. *middleware*), kuri atliktų šias pagrindines funkcijas [3]:

- leistų komponentui užregistruoti įvykį;
- leistų kitiems komponentams matyti, kokie įvykiai yra užregistruoti;
- pateiktų efektyvų įvykių apdorojimo mechanizmą. Pavyzdžiui, nebūtina užregistruoti ir siųsti pranešimus apie įvykį, jei nėra nė vieno komponento, kuriam tas įvykis būtų aktualus.

1.2. Įvykių tipų analizė

Įvykius svarbu suklasifikuoti į tipus, norint sudaryti metamodelį, kurio pagrindu būtų galima sudaryti modelius, vaizduojančius įvykius, jų apdorojimą ir sužadinimą įvykiais pagrįstose sistemose. Informacinių sistemų projektavimo literatūroje pavyko aptikti tik vieną įvykių taksonomiją: UML specifikacijoje [18] įvykiai skirstomi į signalų siuntimo, operacijų kvietimo, laiko ir pasikeitimų įvykius. Šiame darbe tokia klasifikacija buvo nepakankama, kadangi buvo tikslinga detalizuoti laiko įvykių tipus ir išskirti sistemos bei vartotojų sužadintus įvykius.

Kompiuteriniuose žodynuose galima rasti keletą įvykio apibrėžimų:

- Įvykis – reikšmingas atsitikimas tam tikru laiko momentu;
- Įvykis – veiksmas ar atsitikimas, kuris yra apdorojamas kompiuterinės programos. Tai gali būti vartotojo sukeltas veiksmas, toks kaip pelės mygtuko ar klaviatūros klavišo paspaudimas, arba sistemos sugeneruotas veiksmas [19].

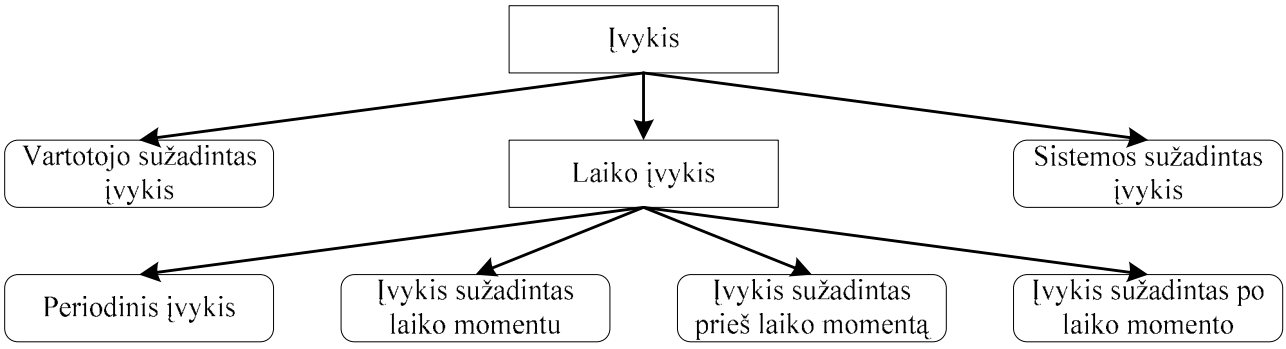
Atsižvelgiant į pastarąjį įvykio apibrėžimą, galima išskirti du įvykių tipus: vartotojo sukurtas įvykis ir sistemos sukurtas įvykis.

Dažnai informacijos sistemos įvykiai turi būti sugeneruoti automatiškai tam tikrais laiko momentais, neatsižvelgiant į vartotojo atliekamus veiksmus arba programos veikimo logiką, todėl galima išskirti dar vieną įvykių tipą: laiko įvykis. Nors šį įvykio tipą generuoja sistema, geriau jį

išskirti iš sistemos įvykių aibės ir nagrinėti atskirai, nes tokių įvykių realizavimas ir modeliavimas skiriasi nuo įprastų sistemos įvykių.

Laiko įvykis gali įvykti tam tikrais laiko momentais. Pavyzdžiui, informacijos sistema gali užregistruoti įvykį, kuris turi būti sužadintas tam tikru laiko momentu, prieš arba po tam tikro laiko momento. Toks įvykis yra sugeneruojamas vieną kartą ir apdorojamas. Be to, kartais tie patys įvykiai turi būti generuojami ciklišKai tam tikrais laiko momentais. Todėl laiko įvykius galima suskirstyti į keturis poaibius: įvykstantys tam tikru laiko momentu bei įvykstantys prieš arba po tam tikro laiko momento, ir periodiniai.

Galima įvykių klasifikavimo schema yra pateikta 1.1 paveikslėlyje.



1.1 pav. Įvykių tipai

Taigi galima išskirti tokius pagrindinius įvykių tipus:

- *vartotojo sužadintas įvykis* – įvykis, kurį sužadina vartotojo atliekami veiksmai su sistema. Tai gali būti mygtuko paspaudimas ekrane, formos lauko reikšmės įvedimas arba koregavimas;
- *sistemos sužadintas įvykis* – įvykis, kurį sužadina sistema vykdydama veiklos logiką. Pavyzdžiui, bankinėje sistemoje klientui įsigijus fondo akcijų už tam tikra nustatytą sumą, sistema sugeneruoja įvykį, kurį apdorodamas rangavimo komponentas padidina kliento reitingą;
- *įvykis, sužadintas laiko momentu* – įvykis, kuris įvyksta realiame pasaulyje arba yra sugeneruojamas sistemos tam tikru laiko momentu, nepriklausomai nuo vartotojo vykdomų veiksmų ar sistemos vykdomos veiklos logikos. Pavyzdžiui, ryšių su klientais valdymo sistemoje, sugeneruojamas įvykis naujų metų pradžioje, kurį apdorojantis komponentas siunčia pasveikinimus įmonės klientams;
- *įvykis, sužadintas prieš laiko momentą arbo po laiko momento* – įvykis, kuris yra sugeneruojamas sistemos prieš arbo po tam tikro laiko momento. Tokie įvykiai dažnai pasitaiko sistemose. Pavyzdžiui, per mėnesį nuo konferencijos pabaigos leidinys turi būti patalpintas portale (įvykis, po tam tikro laiko momento); savaitę prieš konferencijos pradžią dalyviams reikia išsiųsti pranešimus (įvykis, prieš tam tikrą laiko momentą).
- *periodinis įvykis* – įvykis, kuris generuojamas ciklišKai tam tikrais laiko momentais. Pavyzdžiui, projektų valdymo sistemoje, kartą per dieną sugeneruojamas įvykis, kuris patikrina

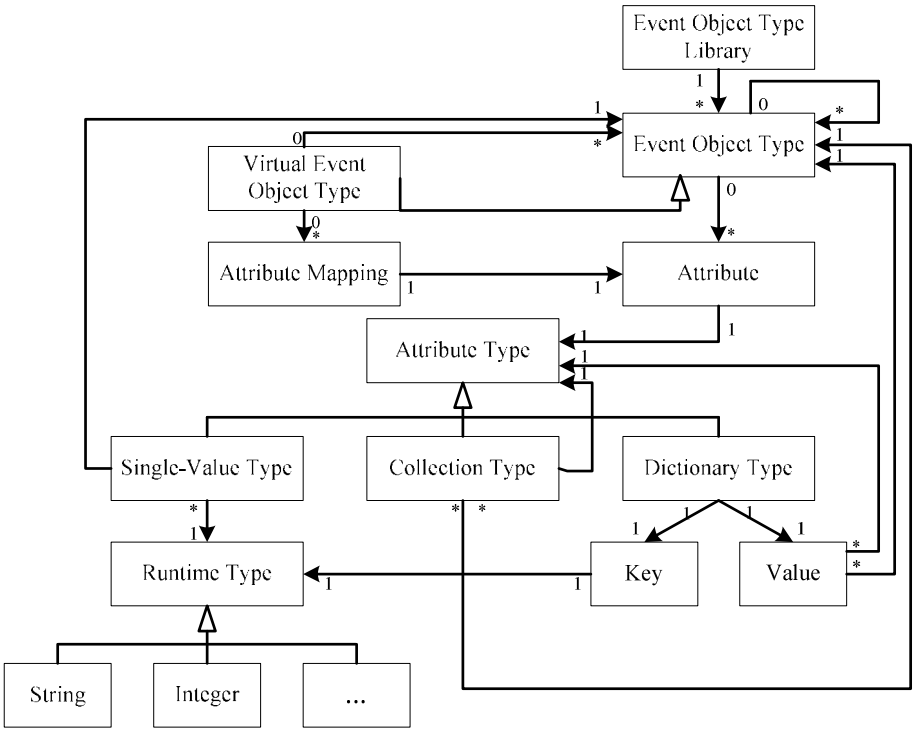
projekto užduočių atlikimą iki nurodytos datos. Jei bent viena užduotis vėluoja, apie tai pranešama projektų vadovui nusiunčiant elektroninį laišką.

1.3. Įvykių modeliavimo metodų analizė

Įvykiais grindžiamos sistemos kuriamos iš komponentų, kurie yra pranešimų siuntėjai ir gavėjai, t.y. bendradarbiavimas tarp komponentų vyksta siunčiant įvykius aprašančius pranešimus, kuriuos reikia apdoroti. Tai leidžia sukurti sistemas, kuriose komponentai yra susieti silpnais ryšiais, ir padidinti sistemų efektyvumą, plečiamumo galimybes. Tačiau tokių sistemų trūkumas yra tas, kad pranešimų gavėjai, turi suprasti pranešimų turinį ir struktūrą ir žinoti, koks įvykis turi būti apdorotas. Šios problemos sprendimas gali būti įvykių modelio sudarymas, kuriame tiksliai aprašomi visi sistemoje naudojami įvykiai ir jų struktūra. Toks įvykių modelis gali būti naudojamas kaip protokolas kuriant komponentus, kuris padėtų suprasti, kokie įvykiai yra galimi, kas juos turėtų apdoroti ir kokios yra įvykių aprašymo taisyklės.

Szabolcs Rozsnyai, Josef Schiefer ir Alexander Schatten savo publikacijoje „Įvykiais grindžiamų sistemų įvykių tipų konceptai ir modeliai“ pateikia įvykių tipų metamodelį, kuris yra naudojamas SARI įvykiais grindžiamoje sistemoje. Šis metamodelis nusako, kaip turi būti aprašomi įvykių tipai, apibūdinant įvykių pobūdį ir struktūrą. Įvykių tipų aprašymas palengvina jų apdorojimą, kadangi yra tiksliai žinoma, kokie atributai aprašo konkretų įvykį, kokie yra atributų tipai [2].

Įvykių tipų metamodelis, naudojamas SARI sistemoje, pateiktas 1.2 paveikslėlyje.



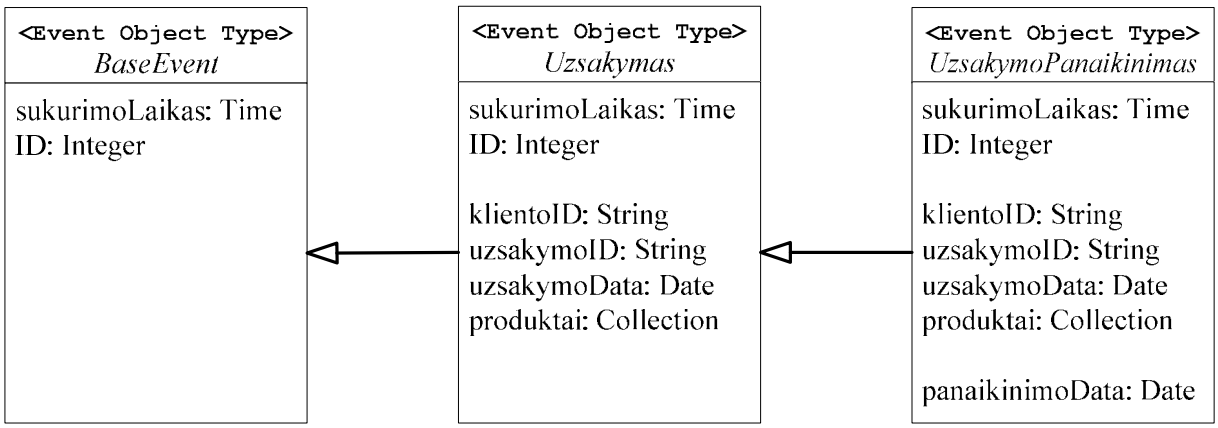
1.2 pav. Įvykių tipų metamodelis

Šiame metamodelyje pagrindinis elementas yra biblioteka (*Event Object Type Library*), kurioje saugomi visi sistemos įvykių tipai. Biblioteka tai saugykla, kurioje patalpinti metaduomenys apie įvykius ir jų atributus. Įvykio aprašymas, saugomas bibliotekoje, yra vadinamas įvykio objekto tipu. Konkretus įvykio egzempliorius yra vadinamas įvykio objektu.

Pagrindinis bibliotekos tikslas yra aprašyti visų įvykių, naudojamų konkrečioje sistemoje, semantiką. Naudojant tokias bibliotekas yra galimybė sukurti įvykių tipus, kurie gali būti panaudoti identifikuojant gaunamus pranešimus ir nustatyti, koks įvykis buvo sužadintas ir kaip jis turėtų būti apdorotas. Įvykių biblioteka gali būti saugoma duomenų bazėje ir naudojama daugelio informacijos sistemų ar taikomųjų programų.

Įvykio objekto tipas gali paveldėti kitus įvykio objekto tipus; taip pat įvykio objekto tipas gali turėti keletą atributų. Kiekvieną atributas aprašo jo tipas. Atributo tipas gali būti paprastas duomenų tipas, kitas įvykio objekto tipas, kolekcija arba žodynas. Kolekcijos susideda iš aibės reikšmių, kurios gali būti paprasto arba kito įvykio objekto tipo. Žodynai sudaromi iš aibių, kurių elementai yra identifikuojami panaudojant raktus (unikalias konkretaus tipo reikšmes).

Įvykių tipai gali sudaryti hierarchiją, t.y. gali paveldėti kito objekto atributus. SARI sistemoje pagrindinis įvykio objekto tipas yra *BaseEvent*, kurį paveldi visi įvykių objektų tipai ir kurio atributus, privalo turėti bet kuris įvykio objekto tipas. 1.3 paveikslėlyje pateiktas paveldėjimo pavyzdys, kuriame užsakymo panaikinimo įvykio objekto tipas paveldi atributus iš užsakymo įvykio objekto tipo.



1.3 pav. Įvykio objekto tipų paveldėjimas

Nors pateiktas metamodelis parodo, kokia yra įvykių modelio sudarymo struktūra, kaip įvykiai skirstomi į tipus, tačiau jis neparodo, kas konkrečius įvykius turėtų apdoroti.

Kitas modeliavimo metodas, kuriame atsispindi ne tik sistemoje sužadinami įvykiai, bet ir agentai, kurie tuos įvykius turi apdoroti, yra AOR (angl. *Agent–Object–Relationship*) modeliavimas. Nors šis metodas yra tinkamesnis modeliuoti programinių agentų sistemas, tačiau jis

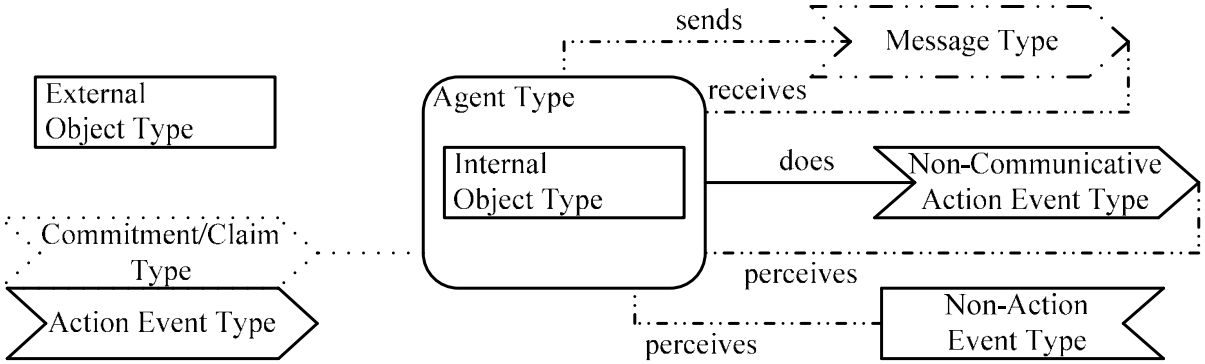
gali būti panaudotas siekiant atvaizduoti įvykių ir pranešimų tipus, bei juos apdorojančius objektus ar komponentus.

AOR modelis gali būti laikomas UML kalbos išplėtimu. Šiame modelyje esybės yra skirstomos į dvi grupes: aktyvias (agentus) ir pasyvias (realaus pasaulio objektai). AOR modelyje esybe gali būti agentas, įvykis (angl. *event*), veiksmas (angl. *action*), pareikalavimas (angl. *claim*), įvykdymas (angl. *commitment*) ar įprastas objektas [4],[5].

Yra du pagrindiniai AOR modeliai: išorinis ir vidinis. Išoriniame modelyje pateikiamas išorinio stebėtojo matomas vaizdas apie agentus, veikiančius ir bendradarbiaujančius dalykinėje srityje, t.y. šis modelis vaizduoja visus agentus, kuriuos reikia atvaizduoti sudarant statinį ir elgsenos modelius. Vidinis modelis vaizduoja konkretaus agento veikimą sistemoje, jo elgseną ir tikslus.

Pagrindiniai išorinio modelio elementai yra pavaizduoti 1.4 paveikslėlyje.

Išorinį modelį sudaro: agentai, komunikaciniai (angl. *communicative*) ir ne komunikaciniai (angl. *non–communicative*) veiksmo įvykiai, ne veiksmo įvykiai (angl. *non-action events*), pareikalavimas/įvykdymas tarp dviejų agentų, įprasti objektai, įvairūs nustatyti ryšiai (pavyzdžiui, siunčia ar atlieka), papildomos asociacijos.



1.4 pav. Pagrindiniai AOR išorinio modelio elementai

Išoriniame modelyje pavaizduoti agentai gali siųsti ir priimti pranešimus; jie gali atlikti komunikacinį veiksmą (pareikalavimą), ir laukti patvirtinimo iš kito agento. Taip pat agentas gali sužadinti ne komunikacinį įvykį, t.y. nereikalauti patvirtinimo, kad tas įvykis buvo apdorotas. Tačiau tokį įvykį užregistruoja (angl. *perceives*) kitas agentas ir apdoroja. Išoriniame modelyje gali būti atvaizduojami ir ne veiksmo įvykiai, t.y. įvykiai, kuriuos sukelia ne sistemos agentai, bet kurie taip pat yra apdorojami.

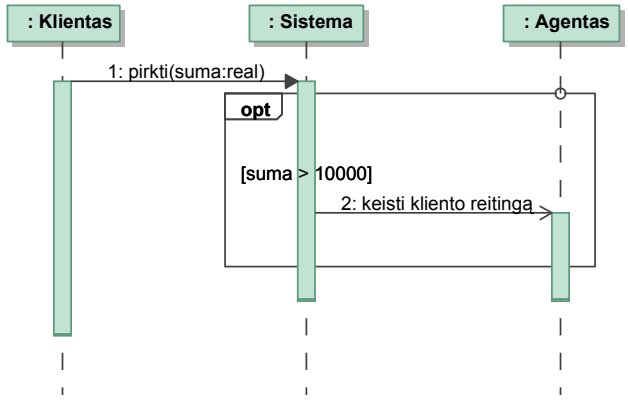
Įvykių modeliavimui galima panaudoti ir standartines UML diagramas: sekų diagramą ir būsenų diagramą.

Sekų diagrama vaizduoja sąveiką tarp objektų. Šia diagrama galima atvaizduoti objektus, kurių vienas sužadina įvykį, o kitas tą įvykį apdoroja. Kadangi dauguma įvykiais grindžiamų

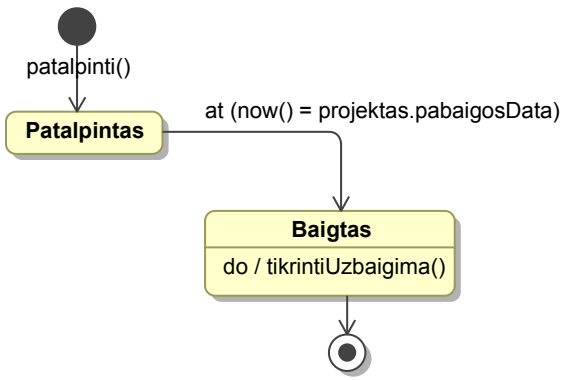
sistemų naudoja tarpinę programinę įrangą ir pranešimų siuntimą, įvykio sužadinimą ir apdorojimą galima pavaizduoti kaip pranešimo siuntimą tarp objektų. Tačiau tokios diagramos netinkamos vaizduoti įvykiams, kurie sužadinami tam tikrais laiko momentais arba periodiškai, nes nėra galimybės aiškiai pavaizduoti, kokiu momentu įvykis buvo sužadintas. Sekų diagramas geriau naudoti vaizduojant įvykius, kurie yra sužadinami vykdant tam tikrą veiklos logiką realizuojantį metodą (1.5 pav.).

UML būsenų diagrama galima vaizduoti įvykius, kurie priverčia objektą pereiti į tam tikrą būseną, kurioje atliekami veiklos logiką realizuojantys metodai. Šios diagramos taip pat gali vaizduoti ir laiko momentus, kuomet pasikeičia objekto būsena, todėl jos tinkamos atvaizduoti laiko momento ar periodinius įvykius. Tačiau būsenų diagramos dažniausiai sudaromos klasei ir vaizduoja tos klasės egzempliorių būsenas, todėl jos nėra tinkamos atvaizduoti įvykiams, kuriuos sužadina vienas objektas, o apdoroja kitas.

Būsenų diagrama, kurioje pateiktas objekto būsenos pasikeitimas tam tikru laiko momentu ir veiksmai, atliekami pasikeitus būsenai, pavaizduota 1.6 paveikslėlyje.



1.5 pav. Pranešimo siuntimas sekų diagramoje



1.6 pav. Būsenos pasikeitimas laiko momentu

1.4. Įvykių realizavimo technologijų analizė

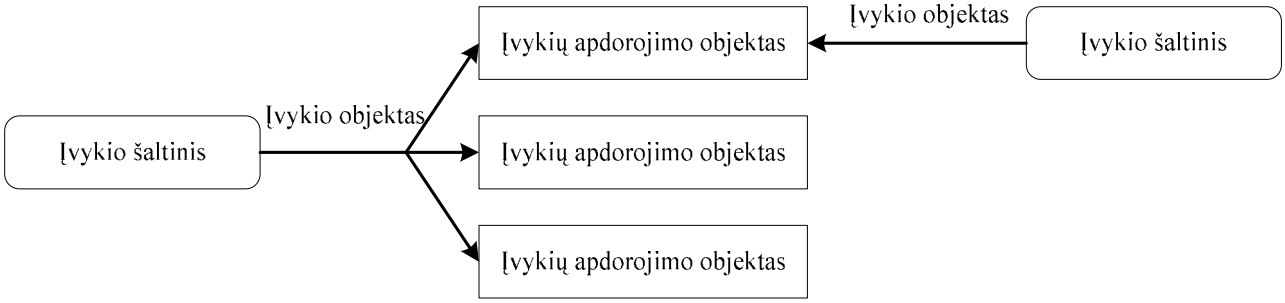
1.2 poskyryje išskirti keturi pagrindiniai įvykių tipai: vartotojo sužadintas įvykis, sistemos sužadintas įvykis, periodinis ir laiko momentu sužadintas įvykis. Šiame poskyryje apžvelgiamos kiekvieno įvykio tipo realizavimo galimybės.

- **Vartotojo sužadinamų įvykių realizavimo galimybės**

Vartotojo sužadinamas įvykis – įvykis, kurį sužadina vartotojas paspausdamas mygtuką ekrane, pakeisdamas formos lauko reikšmę, pažymėdamas keletą punktų pasirinkimo sąraše ar pan. Tokie įvykiai yra valdomi pačios sistemos ir tereikia parašyti metodus šių įvykių apdorojimui.

Java programavimo kalboje sukuriamos klasės, kurios realizuoja atitinkamus interfeisus. Pavyzdžiui, mygtuko paspaudimo apdorojimui reikalingas objektas klasės, kuri realizuoja *java.awt.event.ActionListener* interfeisą. Tokių klasių objektai naudojami įvykių apdorojimui.

Atitinkamas vartotojo sąsajos komponentas (pavyzdžiui, mygtukas) užregistruoja objektą kaip atitinkamo įvykio apdorojimo priemonę. Kuomet įvykis yra sužadinamas, jis perduodamas konkrečiam objektui, kuris įvykdo atitinkamus metodus ir apdoroja perduotą įvykį [16]. 1.7 paveikslėlyje pavaizduota kai tam pačiam įvykių apdorojimo objektui (angl. *event listener*) yra perduodami įvykiai iš skirtingų šaltinių (skirtingų vartotojo sąsajos komponentų sugeneruoti įvykiai); paveikslėlyje taip pat matyti, kad tas pats įvykis, gali būti apdorojamas kelių įvykių apdorojimo objektų.



1.7 pav. Įvykių apdorojimo schema Java kalboje

Žemiau pateiktas pavyzdys klasės, kuri realizuoja *ActionListener* interfeisą. Šios klasės objektas gali būti panaudotas apdoroti mygtuko paspaudimo įvykius.

```
public class ... implements ActionListener {
    ...
    button.addActionListener(this);
    ...
    public void actionPerformed(ActionEvent e) {
        ...
    }
}
```

Panašiai vartotojo sąsajos komponentų sugeneruojami įvykiai apdorojami ir kitose objektinio programavimo kalbose. *C#* kalboje yra sukuriamos specialios klasės (angl. *delegate*), kurios saugo nuorodas į sukurtus metodus, kurie yra užregistruoti įvykio apdorojimui. Kuomet komponentas sugeneruoja įvykį, jį apdoroja visi užregistruoti metodai.

- **Sistemos sužadinamų įvykių realizavimo galimybės**

Sistemos sužadintas įvykis yra įvykis, kurį sužadina pati sistema vykdydama veiklos logiką realizuojantį metodą. Tokie įvykiai dažniausiai sužadinami, kai atitinkama sąlyga tenkina veiklos taisyklę. Pavyzdžiui, jei klientas užsako prekių už tam tikrą sumą, sistema gali sugeneruoti įvykį, kurį apdoros klientų rangavimo komponentas. Tokie įvykiai gali būti generuojami programiniame kode, kai yra tenkinama atitinkama sąlyga, panaudojant *if..else* sakinius:

```
if (tenkinama veiklos taisyklė) {
     ...
     sugeneruoti atitinkamą įvykį
     ...
}
```

Prie sistemos sužadinamų įvykių galima priskirti ir šiuolaikinių duomenų bazių trigerių atliekamus veiksmus. Duomenų bazės trigeris – tai programinis kodas, kuris automatiškai atliekamas reaguojant į įvykius su duomenų baze arba jos lentele. Trigeriai gali uždrausti priėjimą prie duomenų, atlikti duomenų korektiškumo tikrinimą, kai jie atnaujinami arba koreguojami [17].

Daugelio šiuolaikinių duomenų bazių trigerių sukūrimo komandos yra panašios. Žemiau pateikta *ORACLE* DBVS komandos, naudojamos trigeriams kurti, struktūra:

```
CREATE [OR REPLACE] TRIGGER [schema.]trigger
    {BEFORE | AFTER}
    {DELETE | INSERT | UPDATE [OF column [, column] ...]}
[OR {DELETE | INSERT | UPDATE [OF column [, column] ...]}] ...
    ON [schema.]table
    [ [REFERENCING { OLD [AS] old [NEW [AS] new]
                   | NEW [AS] new [OLD [AS] old] } ]
     FOR EACH ROW
     [WHEN (condition)] ]
    pl/sql_block
```

Trigeriai yra sužadinami prieš arba po šalinimo, įterpimo ar įrašo atnaujinimo operacijos. Galima nurodyti, kokias lentelės kolonėlių reikšmės modifikuojant gali būti sužadintas trigeris. Taip pat trigeriuose yra galimybė sukurti kintamuosius, kurie saugo senas ir naujas modifikuojamo įrašo reikšmes. Trigerio programos tekstas yra rašomas atitinkama programavimo kalba, kurią palaiko duomenų bazių valdymo sistema. Pavyzdžiui, *ORACLE* DBVS naudoja *PL/SQL* programavimo kalba, *MSSQL* DBVS – *Transact-SQL* programavimo kalbą.

- **Laiko įvykių realizavimo galimybės**

Laiko įvykis – tai įvykis, kuris yra sužadinamas tam tikru laiko momentu. Jis gali būti sužadinamas vieną kartą, arba periodiškai, tam tikrais laiko momentais.

Laiko įvykiai sistemose gali būti realizuoti panaudojant laikmačius (angl. *Timers*). Šį funkcionalumą palaiko daugelis objektinio programavimo kalbų. Laikmačiai yra sukuriami kaip objektai, nurodant, kada laikmatis turi būti sužadintas, koks yra sužadinimo intervalas ir kokie metodai turi būti atlikti sužadinimo metu. Taip pat tokie objektai turi metodus, skirtus laikmačiams įjungti ir išjungti.

Java programavimo kalboje laikmatis realizuojamas kaip klasės *javax.swing.Timer* objektas. Žemiau pateiktas laikmačio sukūrimo ir paleidimo programinis kodas [15]:

```
int delay = 1000; //milliseconds

ActionListener taskPerformer = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        ...
    }
};

new Timer(delay, taskPerformer).start();
```

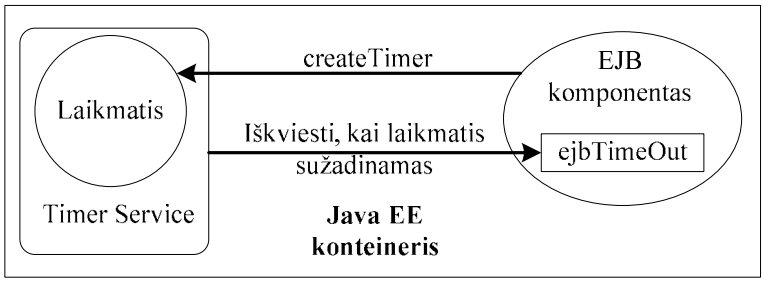
C# programavimo kalboje yra trys klasės, kurias realizavus galima sukurti laikmačius: *System.Windows.Forms.Timer*, *System.Timers.Timer* ir *System.Threading.Timer*. Kiekviena iš šių klasių buvo sukurta ir optimizuota skirtingiems tikslams. *System.Windows.Forms.Timer* klasė naudojama sukurti laikmačiui, kuris naudojamas kuriant *Windows* sistemoje veikiančias taikomąsias programas (angl. *Windows Forms Applications*), *System.Timers.Timer* gali būti panaudojamas, kuriant komponentus, kurie yra įdiegiami serveryje [14].

Kadangi dauguma šiuolaikinių verslo sistemų yra perkeliamos į Internetą, t.y. kuriamos žiniatinklio taikomosios programos, iškyla aktualus klausimas, kaip laikmačius realizuoti taikomųjų uždavinių serveriuose.

Java EE 5.0 technologija turi specialią taikomųjų programų kūrimo sąsają (angl. *Application Programing Interface*), kurioje apibrėžti interfeisai reikalingi laikmačiams kurti. Ši sąsaja vadinama *Timer Service.* Norint sukurti laikmatį Java EE taikomojoje programoje reikia [11]:

- *EJB* komponentas turi realizuoti *javax.ejb.TimedObject* interfeisą;
- Turi būti sukurtas taimeris iškviečiant metodą *createTimer*.
- Turi būti sukurtas metodas *ejbTimeOut*, kuris bus įvykdomas kai laikmatis yra sužadinamas.

1.8 paveikslėlyje pavaizduotas ryšys, tarp *EJB* komponento realizuojančio laikmatį ir laikmačių valdymo paslaugos, kurią teikia Java EE 5.0 technologiją palaikantis serveris.



1.8 pav. Java EE laikmačio veikimas

Java EE 5.0 technologija leidžia sukurti 4 tipų laikmačius. Metodai, naudojami laikmačių sukūrimui pateikti 1.1 lentelėje.

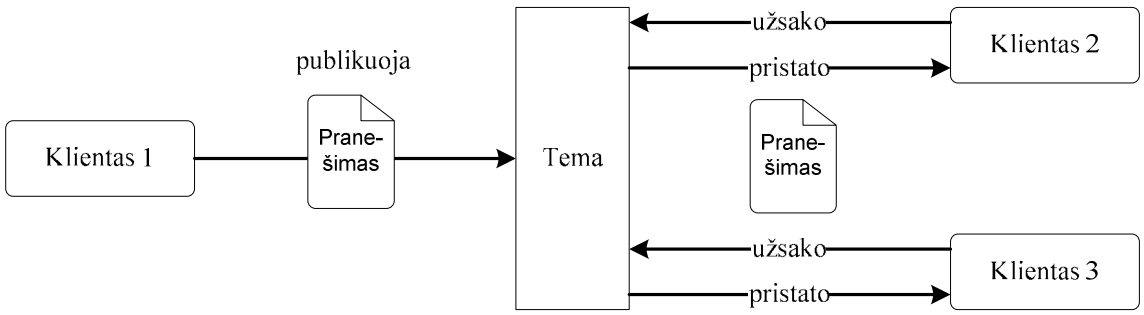
1.1 lentelė. Java EE technologijos laikmačių tipai

LAIKMAČIO TIPAS	SUKŪRIMO METODAS
Vieno įvykio laikmatis su sužadinimo laiku	`createTimer(long timeoutDuration, Serializable info)`
Vieno įvykio laikmatis su sužadinimo data	`createTimer(Date firstDate, Serializable info)`
Periodinis laikmatis su sužadinimo laiku	`createTimer(long timeoutDuration, long timeoutInterval, Serializable info)`
Periodinis laikmatis su sužadinimo data	`createTimer(Date firstDate, long timeoutInterval, Serializable info)`

1.5. Pranešimų siuntimas apie sužadintus įvykius

Kai sistema, vykdydama veiklos logikos metodą, sužadina įvykį arba kai, praėjus nustatytam laikui, sužadinamas laikmatis, kuris sugeneruoja įvykį, apie įvykį turi būti pranešama kitiems sistemos komponentams ar objektams, kurie tą įvykį apdoroja. Dažnai kuriant įvykiais pagrįstas sistemas yra naudojama tarpinė programinė įranga, kurios pagalba yra perduodami pranešimai tarp komponentų ir taip pranešama, jog sistemoje buvo sužadintas įvykis.

Yra du pagrindiniai pranešimų siuntimo ir gavimo panaudojimo būdai: publikavi–mo/užsakymo (angl. publish/subscribe) ir eilių sudarymo (angl. queuing) mechanizmai. Naudojant pirmąjį metodą, pranešimų siuntėjai pristato pranešimus tam tikrai temai (angl. topic), o tarpinė programinė įranga juos persiunčia visiems pranešimų gavėjams, kurie yra užregistruoti nurodytai temai. Dažnai tarpinė programinė įranga transformuoja pranešimų turinį taip, kad jis būtų suprantamas gavėjams [8],[12]. Publikavimo/užsakymo mechanizmo schema pateikta 1.9 paveikslėlyje.



1.9 pav. Publikavimo/užsakymo mechanizmas

Naudojant eilių sudarymo mechanizmą, pranešimai yra siunčiami į eilę, kurioje jie yra saugojami iki pranešimo galiojimo pabaigos arba iki tol, kol yra paimami pranešimų gavėjų. Skirtingai nuo publikavimo/užsakymo mechanizmo, naudojant eilių sudarymą pranešimas yra persiunčiamas tik vienam gavėjui, t.y. tam, kuris paima pranešimą iš eilės. Eilių sudarymo mechanizmo schema yra pateikta 1.10 paveikslėlyje [8].

1.10 pav. Eilių sudarymo mechanizmas

Realizuojant įvykiais grindžiamas sistemas, svarbu pasirinkti tokias technologijas, kurios leistų siųsti pranešimus tarp komponentų ar objektų. Pranešimų siuntimas privalo būti asinchroninis, t.y. siunčiant pranešimą reikia tik pranešti komponentui apie įvykį, kurį jis turi apdoroti, ir nereikia laukti atsakymo.

Literatūroje apie įvykiais grindžiamas sistemas dažnai minimi ir programiniai agentai. Programinis agentas – programinės įrangos objektas, kuris nepertraukiamai ir autonomiškai funkcionuoja tam tikroje aplinkoje ir gali bendrauti su kitais agentais ar procesais. Programiniai agentai gali apdoroti sistemos sugeneruotus įvykius.

Kuriant organizacijų įvykiais grindžiamas verslo sistemas gali būti panaudota Java EE technologija. Ši technologija gali būti panaudota realizuojant programinius agentus bei asinchroninį pranešimų siuntimą tarp komponentų.

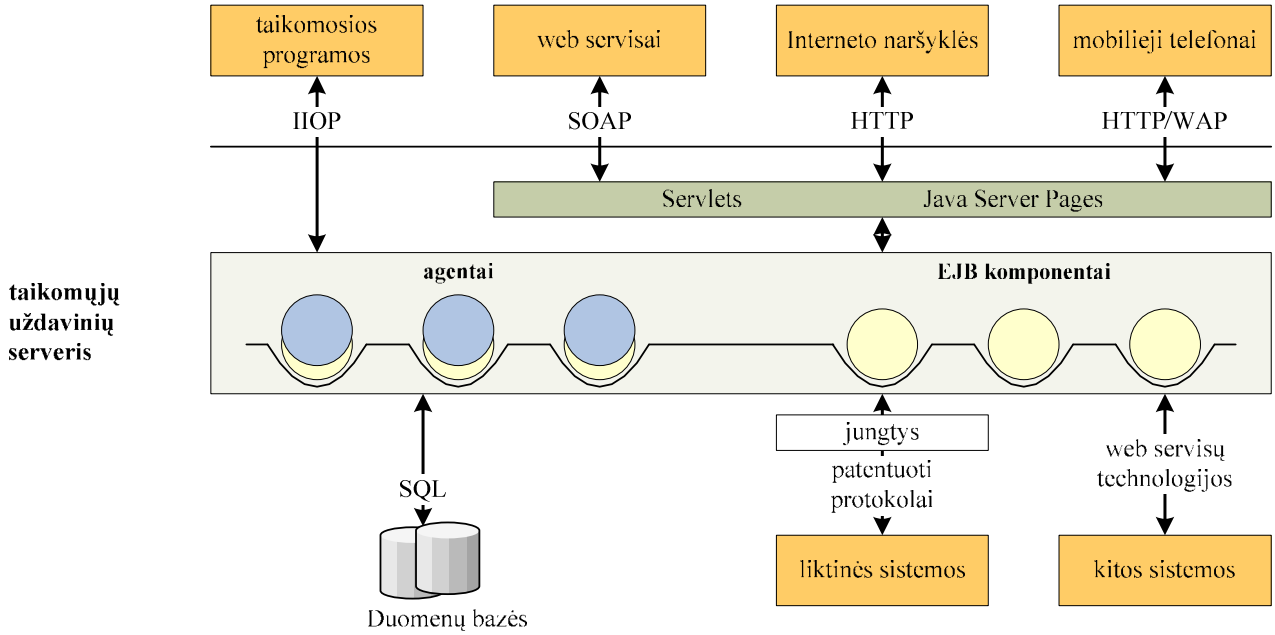
Naudojant Java EE technologiją programinius agentus galima realizuoti kaip *Enterprise JavaBeans* (*EJB*) komponentus. *Enterprise JavaBeans* – yra serverio pusės komponentas, skirtas paskirstytų, portatyvinių, saugių sistemų kūrimui Java taikomųjų uždavinių serveriuose [7]. *EJB* dažniausiai naudojami veiklos logikai realizuoti trijų lygių architektūros sprendimuose.

Serverio taikomosios programos yra įdiegiamos Java EE aplinkoje, kuri suskirstyta į konteinerius:

- Web konteinerį (angl. *Web Container*) – kuriame talpinami servletai ir *JSP* puslapiai;
- EJB konteinerį (angl. *EJB Container*) – kuriame talpinami *EJB* komponentai.

Kuriant programinius agentus kaip *EJB* komponentus, jie realizuojami pagal griežtus standartus, kuriais apibrėžiami *EJB* komponentai Java EE specifikacijoje, kartu realizuojant metodus (agentų elgseną), kuriuos turi įvykdyti programiniai agentai; agentai veikia įprastai kaip *EJB* komponentai ir gali būti patalpinti į *EJB* konteinerį. Tokiu atveju, programiniai agentai tampa portatyvūs, t.y. juos galima realizuoti bet kuriame serveryje, palaikančiame Java EE technologijas [6].

1.11 paveikslėlyje pavaizduota, programinių agentų, kaip *EJB* komponentų, realizacija taikomųjų uždavinių serveryje:



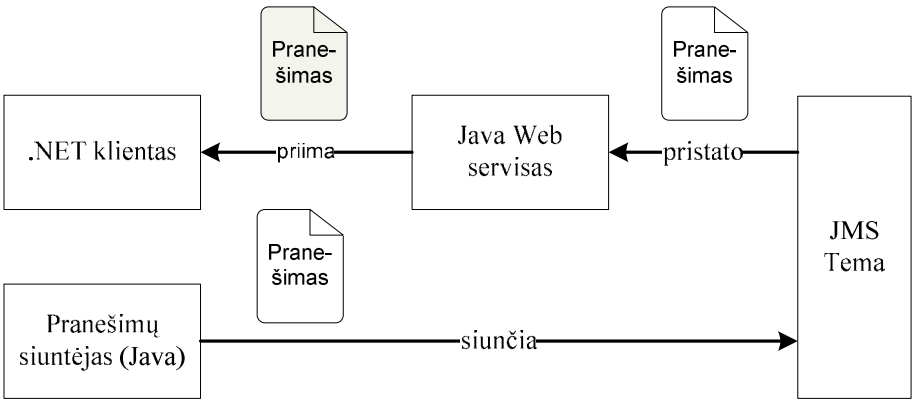
1.11 pav. Programinių agentų realizavimas EJB komponentais

Java EE 5.0 technologija pateikia *JMS* (angl. *Java Message Service*) taikomųjų programų kūrimo sąsaja, kuri leidžia kurti, siųsti, priimti ir skaityti pranešimus. Šios sąsajos pagalba galima sukurti temas (angl. *topics*) ir eiles (angl. *queues*), kurios yra naudojamos pranešimų siuntimui. *JMS* leidžia sukurti silpnais ryšiais susietų komponentų bendradarbiavimą, kuris yra asinchroninis ir patikimas [8].

Panaudojant EJB komponentus ir JMS sąsajos funkcionalumą, galima sukurti agentus, kurie priima pranešimus apie įvykius ir juos apdoroja. Žemiau yra pateiktas kodas *Message Driven Bean* komponento (atskiras *EJB* komponento variantas), kuris priima pranešimus apie įvykius ir juos apdoroja.

```
@MessageDriven(mappedName="jms/Queue")
public class SimpleMessageBean implements MessageListener {

@Resource
private MessageDrivenContext mdc;

     public void onMessage(Message inMessage) {

          TextMessage msg = null;
          try {
                if (inMessage instanceof TextMessage) {
                msg = (TextMessage) inMessage;

          } catch (JMSException e) {
     }
}
```

JMS funkcionalumą galima panaudoti kuriant sistemas su .NET karkasu. Priėjimas prie JMS temų arba pranešimų eilės realizuojamas panaudojant Web servisus. Sukurtas Java Web servisas gali priimti pranešimus ir pranešti apie įvykį .NET programai. Tokios architektūros diagrama pavaizduota 1.12 paveikslėlyje.



1.12 pav. Pranešimo siuntimas .NET klientui, panaudojant JMS servisą

1.6. Egzistuojančių su įvykiais susijusių sistemų analizė

Vienas iš magistrinio darbo tikslų yra ištirti įvykiais grindžiamų sistemų modeliavimo, realizavimo ir panaudojimo galimybes. Yra daugybė sričių, kur galima panaudoti sistemas, sudarytas iš komponentų, reaguojančių į įvykius ir bendradarbiaujančių siunčiant pranešimus: elektroninės komercijos sistemos, nuotolinio mokymo sistemos, projektų valdymo sistemos ir t.t.

Kuriant nuotolinio mokymo sistemas, yra siekiama, kad naujienos, kurias paskelbia dėstytojai arba studentai, būtų prieinamos kiekvienam studijų dalyviui. Tokiu atveju, dažniausiai yra siunčiami elektroniniai laiškai, pranešantys apie naujos medžiagos patalpinimą internete, naujų atsiskaitymų paskelbimą ir t.t. Tokius darbus automatizuoja programiniai agentai, reaguojantys į vartotojo įvykius ir siųsdami pranešimus. Analogiškai, kuriant publikacijų portalo prototipą, siekiama, kad apie naujų straipsnių patalpinimą tam tikra tema būtų pranešta dalyviams, kurie yra suinteresuoti tos temos naujienomis.

Taip pat mokymosi sistemose yra realizuojamos priminimo funkcijos, kuomet dalyviui yra siunčiamas elektroninis laiškas su pranešimu, kad reikia iki nustatytos datos pateikti atliktus darbus, arba kad po kelių dienų bus atsiskaitymas. Publikacijų portalo prototipe taip pat bus siekiama realizuoti šias funkcijas, kuomet sistemos vartotojui bus priminta apie dar neparašytas recenzijas.

Nuotolinės mokymo internetinės sistemos pavyzdys: http://webct.liedm.lt/.

Elektroninės komercijos sistemose pirkėjams ieškant prekės arba atliekant pirkimus programiniai agentai seka vartotojų veiksmus ir juos registruoja, t.y. automatiškai registruoja vartotojo įvykius. Kitam pirkėjui ieškant tokios pat prekės pagal sukauptą informaciją, programiniai agentai gali pasiūlyti pirkėjui nusipirkti tam tikras prekes, kuriomis domėjosi prieš tai pirkęs pirkėjas. Kuriamo publikacijų portalo prototipo agentai kaups informaciją apie labiausiai skaitomus ir geriausių vertinimų sulaukusius straipsnius, o sistemos vartotojui atliekant paiešką, bus pateikti straipsniai, kurie sulaukė daugiausiai susidomėjimo ir buvo geriausiai įvertinti.

Elektroninės komercijos sistemų pavyzdžiai: www.amazon.com, www.ebay.com.

Publikacijų portalų pavyzdžiai: http://www.ieee.org/web/publications/home/index.html, http://bookshop.europa.eu, http://www.scienceresearch.com.

1.7. Įvykiais grindžiamų informacinių sistemų kokybės kriterijai

Atlikta analizė rodo, kad tikslinga sukurti informacinių sistemų kūrimo metodiką, apimančią įvykių modeliavimą bei jų realizavimą. Tokia metodika įgalintų padidinti informacinių sistemų lankstumą pokyčiams ir pagreitinti tų pokyčių įgyvendinimą. Kokybinių charakteristikų, taikomų lanksčioms, greitai modifikuojamoms informacinėms sistemoms, sąrašas pagal [13] pateikiamas 1.2

lentelėje. Savybės, kurias būtų galima geriau išpildyti modeliuojant įvykius, pažymėtos trečiajame stulpelyje. Jame taip pat pažymima, kokios charakteristikos šiame tyrime specifiškai nenagrinėjamos (t.y. realizuojant prototipą jos įgyvendinamos įprastais būdais) arba kurios gali pablogėti, kadangi sistemos lankstumo, pakartotinio naudojimo ir kitų panašių savybių padidinimas paprastai šiek tiek pablogina veikimą.

1.2 lentelė. Sistemų vertinimo charakteristikos

Sistemos charakteristika	**Charakteristikos apibūdinimas**	**Įvykiais grindžiamose sistemose**
Judrumas (angl. *agility*)	Judrumas – tai sistemos galimybė būti lanksčiai ir sugebėti greitai pasikeisti.	padidėja
Lankstumas (angl. *flexibility*)	Lankstumas apibrėžia sistemos ar komponento modifikavimo ir pritaikymo naudoti kitose sistemose lengvumą.	padidėja
Junglumas (angl. *interoperability*)	Junglumas apibrėžia dviejų ar daugiau sistemų ar komponentų gebėjimą apsikeisti informacija ir ją panaudoti.	padidėja
Palaikomumas (angl. *maintainability*)	Palaikomumas apibrėžia sistemos ar komponento modifikavimo lengvumą, siekiant ištaisyti klaidas, padidinti veikimo spartą ar pritaikyti pasikeitusioje aplinkoje.	padidėja
Patikimumas (angl. *reliability*)	Patikimumas yra sistemos galimybė veikti be klaidų. Ši savybė dažniausiai matuojama vidutiniu laiku, kurį sistema dirba be klaidų.	šiame darbe neanalizuojamas
Pakartotinis panaudojimas (angl. *reusability*)	Pakartotinis panaudojimas apibrėžia sistemos ar komponento galimybę būti panaudotam daugiau nei vienoje programoje ar sistemoje.	padidėja
Veikimo sparta (angl. *performance*)	Veikimo sparta tai sistemos charakteristika, kuri matuojama laiku, reikalingu sistemai reaguoti į įvykius, arba įvykių kiekiu, apdorojamu per laiko intervalą.	gali sumažėti
Saugumas (angl. *security*)	Saugumas apibrėžia sistemos galimybę būti atspariai neleistiniems bandymams sistemos panaudojimui ir tuo pačiu metu teikti savo paslaugas teisėtiems vartotojams.	šiame darbe neanalizuojamas
Plečiamumas (angl. *scalability*)	Plečiamumas – tai sistemos galimybė išlaikyti arba padidinti savo veikimo spartą, kai sistemos vartojimo poreikis (apkrovimas) padidėja.	padidėja
Testavimo lengvumas (angl. *testability*)	Tai savybė palengvinanti testavimo kriterijų apibrėžimą ir testų veikimą, siekiant nustatyti, ar sistema veikia teisingai.	šiame darbe neanalizuojamas
Panaudojamumas (angl. *usability*)	Panaudojamumas yra: • matas, apibrėžiantis vartotojo galimybę panaudoti sistemą efektyviai; • savybė, nusakanti lengvumą vartotojui išmokti dirbti su sistema, paruoši įvedimo duomenis ir interpretuoti gautus rezultatus; • matas, apibrėžiantis, kaip vartotojai panaudoja sistemos funkcionalumo privalumus.	šiame darbe neanalizuojamas

1.8. Analizės išvados

1. Informacinių sistemų lankstumo didinimo galimybių analizė parodė, kad tikslinga kurti įvykiais grindžiamas informacines sistemas, tačiau tokių sistemų kūrimo metodikos yra vystymosi stadijoje ir nėra aiškiai apibrėžtos.

2. Įvykių klasifikacijų analizės metu nustatyta, kad pagrindiniai įvykių tipai yra vartotojo arba sistemos sužadinami įvykiai, bei tam tikrais laiko momentais sužadinami įvykiai. Tokia įvykių klasifikacija yra naudinga sudarant metamodelį, aprašant įvykių stereotipus, sudarant įvykių apibendrinimo ryšius.

3. Įvykių modeliavimo metodų analizė parodė, kad esami įvykių modeliavimo metodai apima atskirus įvykių vaizdavimo aspektus: SARI sistemos įvykių modelis yra tinkamas vaizduoti įvykių tipams ir juos aprašantiems atributams, tuo tarpu AOR modeliavime aprašomi įvykius apdorojantys komponentai ir objektai. Todėl tikslinga sudaryti įvykių metamodelį sujungiant abiejų modeliavimo metodikų stipriąsias savybes.

4. Skirtingų įvykių tipų realizavimo galimybių analizės metu nustatyta, kad vartotojo sužadinami įvykiai (pavyzdžiui, mygtuko paspaudimas ekrane) objektinėse programavimo kalbose dažniausiai realizuojami automatiškai ir tereikia suprogramuoti metodą tokio įvykio apdorojimui; sistemos sužadinami įvykiai gali būti realizuojami duomenų bazių trigeriais arba programos kodu, kuriame įvykiai yra sugeneruojami vykdant veiklos logiką realizuojančius komponentus, kai tenkinamos veiklos taisyklės; laiko įvykiai realizuojami laikmačiais, kurie palaikomi daugelyje objektinių programavimo kalbų.

5. Įvykiais grindžiamų sistemų dinamikos realizavimo modelių analizė parodė, kad pagrindinis tokių sistemų realizavimo mechanizmas yra pranešimų siuntimas tarp komponentų, kai vienas komponentas praneša apie sužadintą įvykį kitam komponentui; pagrindiniai metodai pranešimų siuntimui yra eilių sudarymo ir publikavimo/užsakymo mechanizmai.

6. Įvykiais grindžiamų sistemų realizavimo technologijų analizės metu nustatyta, kad *Java EE JMS* taikomųjų programų kūrimo sąsaja realizuoja abu pranešimų siuntimo mechanizmus ir įgalina realizuoti visus reikiamus įvykių tipus, todėl ji pasirinkta publikacijų portalo prototipo kūrimui.

2. Įvykių metamodelis

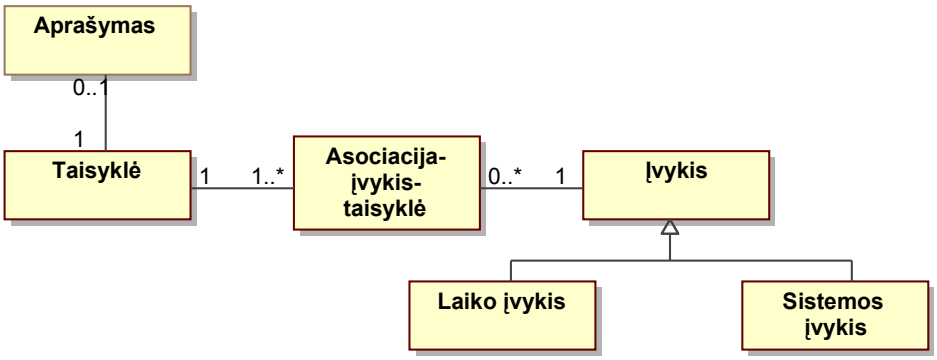
Šiame skyriuje pateikiamas įvykių metamodelis, kurio pagrindu sudarytas įvykių modelis naudojamas įvykiais grindžiamų sistemų projektavimo procese. Kitaip tariant, įvykių metamodelis apibrėžia taisykles, pagal kurias sudaromi konkrečių sistemų įvykių modeliai. Šis metamodelis apima visus analizės etape išskirtus įvykių tipus, aprašo galimas įvykių sąsajas su pranešimų siuntėjais ir gavėjais bei įvykių generavimo taisyklėmis.

2.1. Sistemos generuojamų įvykių ir laiko įvykių aprašymas metamodelyje

Analizuojant įvykiais grindžiamas sistemas, įvykiai buvo suskirstyti į tris pagrindines grupes: vartotojo generuojami įvykiai, sistemos generuojami įvykiai ir įvykiai, sugeneruojami tam tikrais laiko momentais (laiko įvykiai).

Sudarant įvykių metamodelį, įvykis yra specializuojamas į dvi grupes (2.1 pav.): sistemos įvykį ir laiko įvykį; šioms grupėms reikia apibrėžti skirtingas asociacijas su kitais metamodelio elementais.

Kartais įvykis sužadinamas tik tada, kai yra tenkinama tam tikra veiklos taisyklė, todėl įvykis asociacijos ryšiu gali būti susietas su viena ar daugiau taisyklių; tai reiškia, kad įvykis bus sužadintas tik esant atitinkamoms sąlygoms. Taisyklė gali turėti aprašymą ir būti susieta su keliais įvykiais.

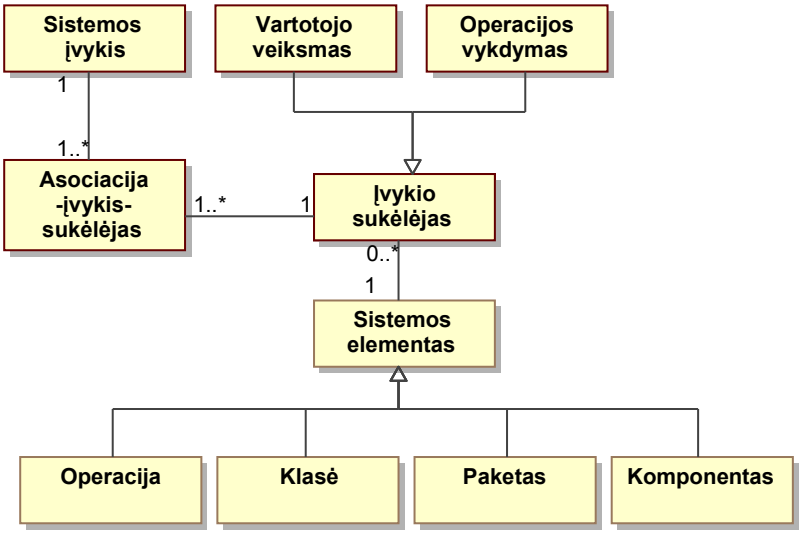


2.1 pav. Įvykių specializacija metamodelyje

Sistemos įvykis yra susiejamas asociacijos ryšiu su įvykio sukėlėju (2.2 pav.), kuris gali būti vartotojo atliekamas veiksmas, arba sistemos operacijos vykdymas, kai vykdant veiklos logiką ir esant tam tikroms sąlygoms turi būti sužadinamas įvykis. Aprašant įvykio sukėlėją, gali būti nurodytas komponentas, paketas, klasė bei klasės operacija, kurią vykdant yra generuojamas įvykis. Šie atributai nėra privalomi ir sistemos projektavimo analizės etape, kai tiksliai dar nėra žinomi, gali būti neaprašomi.

Vartotojo sukelti įvykiai taip pat yra apdorojami sistemos, iškviečiant atitinkamas operacijas (pavyzdžiui, metodą *onClick*). Tačiau skirstymas tarp įprastų operacijų ir vartotojo veiksmų

modeliui suteikia daugiau aiškumo. Sudarant įvykių modelį galima atskirose diagramose aprašyti vartotojo sužadinamus įvykius ir sistemos sugeneruojamus įvykius.

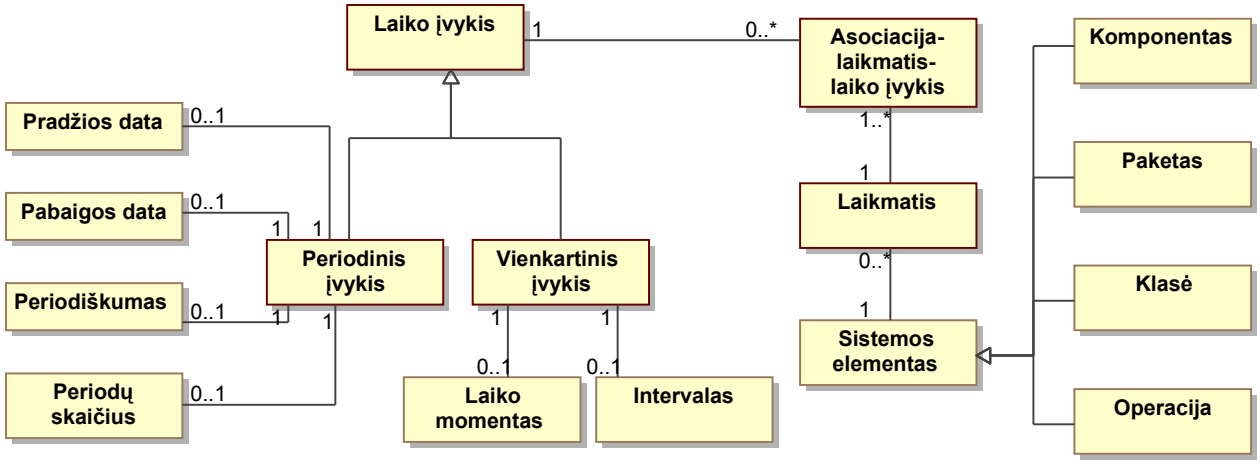


2.2 pav. Sistemos įvykių aprašymas metamodelyje

Laiko įvykis asociacijos ryšiu gali būti susietas su laikmačiu, kuris jį sugeneruoja (2.3 pav.). Laikmačiui gali būti nurodytas komponentas, paketas ar klasė, kuri jį realizuoja.

Laiko įvykiai yra specializuojami į dvi grupes: periodiniai ir vienkartiniai laiko įvykiai. Vienkartinis laiko įvykis yra sužadinamas vieną kartą tam tikru laiko momentu, todėl gali būti nurodytas laiko momentas, kada įvykis turi būti sugeneruotas. Aprašant periodinį įvykį, nurodoma pradžios data, kada turi būti sugeneruotas pirmasis įvykis, pabaigos data, po kurios periodinis įvykis yra nebegeneruojamas, bei periodiškumas (laiko intervalas, nusakantis įvykių generavimo dažnumą) ir periodų skaičius. Šie atributai nėra privalomi.

Aprašant įvykius, kurie yra sugeneruojami prieš arba po tam tikro laiko momento, yra nurodomas laiko momentas ir laiko intervalas prieš kurį arba po kurio turi būti sugeneruotas įvykis. Pavyzdžiui, jei reikia sugeneruoti įvykį, kuris turi įvykti praėjus laiko intervalui t_1, po laiko įvykio, įvykusio laiko momentu t, tai galima aprašyti vienkartinį įvykį, kuris įvyks $t + t_1$ laiko momentu.



2.3 pav. Laiko įvykių aprašymas metamodelyje

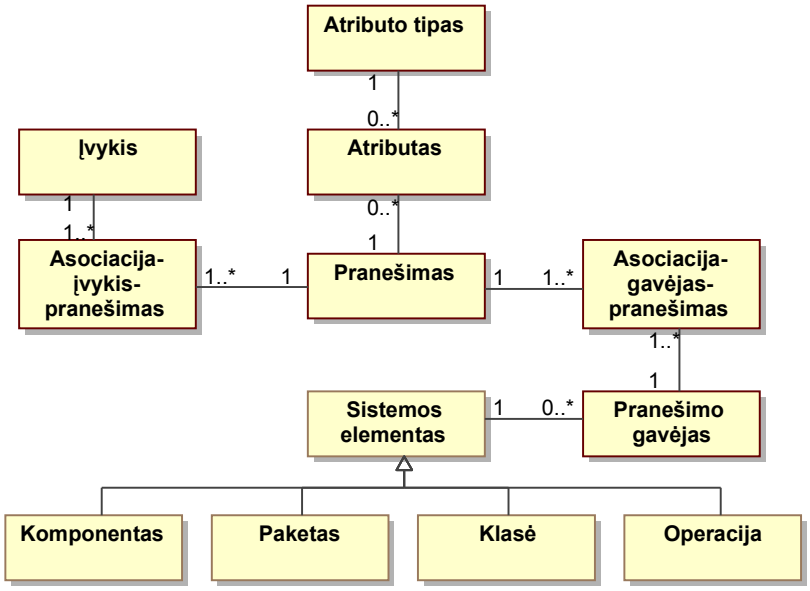
2.2. Pranešimų ir įvykių apdorojimo aprašymas metamodelyje

Sudarant įvykių modelį yra svarbu ne tik aprašyti visus galimus įvykius, bet ir nurodyti, kas juos turi apdoroti. Apie įvykio sužadinimą komponentams ar objektams, kurie turi įvykį apdoroti, yra pranešama siunčiant pranešimus, todėl taip pat svarbu modelyje atvaizduoti ir pranešimus.

Įvykis gali būti susietas su keletu pranešimų, t.y. sužadinus įvykį gali būti siunčiami vienas ar daugiau pranešimų. Norint aprašyti pranešimų turinį, galima nurodyti atributus bei jų pradines reikšmes, pagal kurias turi būti suformuotas pranešimas. Pranešimo formatas pasirenkamas priklausomai nuo realizacijos; tai gali būti XML ar paprasto teksto žinutė.

Pranešimas yra susiejamas su vienu ar daugiau pranešimų gavėjų. Pranešimų gavėju gali būti komponentas ar klasė, kuri atitinkamą įvykį apdoroja. Taip pat galima nurodyti paketą, kuriame realizuojama įvykį apdorojanti klasė. Šie pranešimų gavėjo atributai nėra privalomi ir gali būti nenurodomi sistemos projekto analizės metu, kai dar tiksliai nežinomos klasės ir komponentai, kurie bus realizuoti sistemoje.

Metamodelio fragmentas, kuriame pavaizduotas įvykių, pranešimų ir pranešimų gavėjų asociacijos pateiktas 2.4 paveikslėlyje.



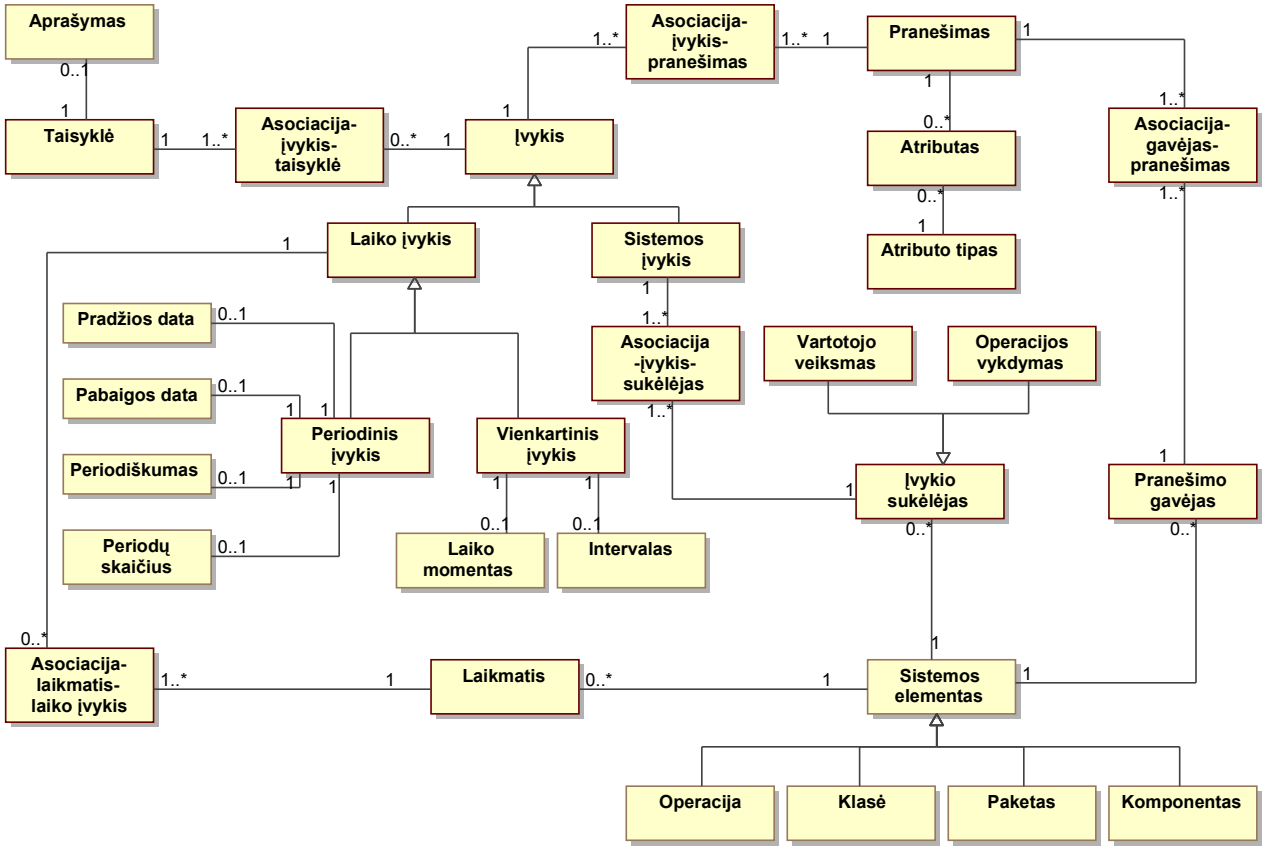
2.4 pav. Metamodelio fragmentas, vaizduojantis galimus ryšius tarp pranešimų gavėjų ir įvykių

2.3. Galutinis įvykių metamodelis

2.5 paveikslėlyje pavaizduotas įvykių metamodelis, sujungiantis įvykių aprašymo elementus bei pranešimus ir pranešimų gavėjus.

Sudarant įvykių modelį pagal aprašytą metamodelį, visi modelio elementai yra jungiami asociacijos ryšiu, nenurodant ryšio kardinalumo. Įvykių modelis gali būti sudarytas iš tokių elementų: laiko įvykis, sistemos įvykis, vartotojo veiksmas, operacija, taisyklė, laikmatis,

pranešimas ir pranešimo gavėjas. Elementų jungimo taisyklės, t.y. kokius elementus galima jungti tarpusavyje, aprašytos 2.5 lentelėje.



2.5 pav. Įvykių metamodelis

2.1 lentelė. Įvykių modelio elementų jungimo taisyklės

	PERIODINIS ĮVYKIS	VIENKARTINIS ĮVYKIS	SISTEMOS ĮVYKIS	VARTOTOJO VEIKSMAS	OPERACIJA	TAISYKLĖ	LAIKMATIS	PRANEŠIMAS	PRANEŠIMO GAVĖJAS
PERIODINIS ĮVYKIS						+	+	+	
VIENKARTINIS ĮVYKIS						+	+	+	
SISTEMOS ĮVYKIS				+	+	+		+	
VARTOTOJO VEIKSMAS			+						
OPERACIJA			+						
TAISYKLĖ	+	+	+						
LAIKMATIS	+	+							
PRANEŠIMAS	+	+	+						+
PRANEŠIMO GAVĖJAS								+	

3. Įvykių modelio panaudojimas informacinių sistemų projektavimo procese

Šiame skyriuje aprašomas įvykių modelio, sudaryto pagal anksčiau pateiktą metamodelį, panaudojimas informacinių sistemų projektavimo procese. Aprašomos įvykių identifikavimo galimybės iš panaudojimo atvejų modelių, nurodoma, kaip įvykių modelis siejasi su kitais projektavimo modeliais: klasių, sekų, būsenų diagramomis.

Paskutiniame šio skyriaus poskyryje pateikiama įvykiais grindžiamų sistemų modeliavimo ir realizavimo metodika.

3.1. Įvykiais grindžiamų sistemų projektavimo proceso problema

Literatūroje pateikiama daug metodologijų, kurios nusako kaip atlikti vartotojo reikalavimų analizę, sudaryti organizacijos veiklos modelius, aprašyti priimtus architektūrinius sprendimus ir pan. Visų šių etapų visuma aprašo projektavimo procesą, t.y. seką veiksmų, kuriuos atliekant nuosekliai gaunamas išbaigtas sistemos projektas.

Daugumą aprašytų projektavimo procesų galima panaudoti kuriant įvykiais grindžiamas sistemas, tačiau dažnai tokie projektavimo procesai nenurodo, kaip aprašyti sistemos įvykius ir juos atvaizduoti struktūriškai, arba kaip sumodeliuotus įvykius panaudoti kituose sistemos projekto modeliuose.

2 skyriuje aprašyto įvykių metamodelio pagrindu sudarytas įvykių modelis gali būti panaudojamas sudarant klasių modelį, kuomet reikia identifikuoti klases ar komponentus, kurie turi siųsti bei apdoroti pranešimus. Identifikavus reikiamas klases ir jų operacijas, galima sudaryti sekų diagramas. Šios diagramos vaizduoja bendradarbiavimą tarp klasių, siunčiančių pranešimus ar kviečiančių operacijas, todėl įvykių modelyje aprašyti pranešimai gali būti panaudojami aprašant siunčiamus pranešimus.

Įvykių modelyje aprašyti įvykiai taip pat gali būti panaudoti būsenų diagramose, nurodant įvykius, kuriuos sužadinant objektas pereina į kitą būseną.

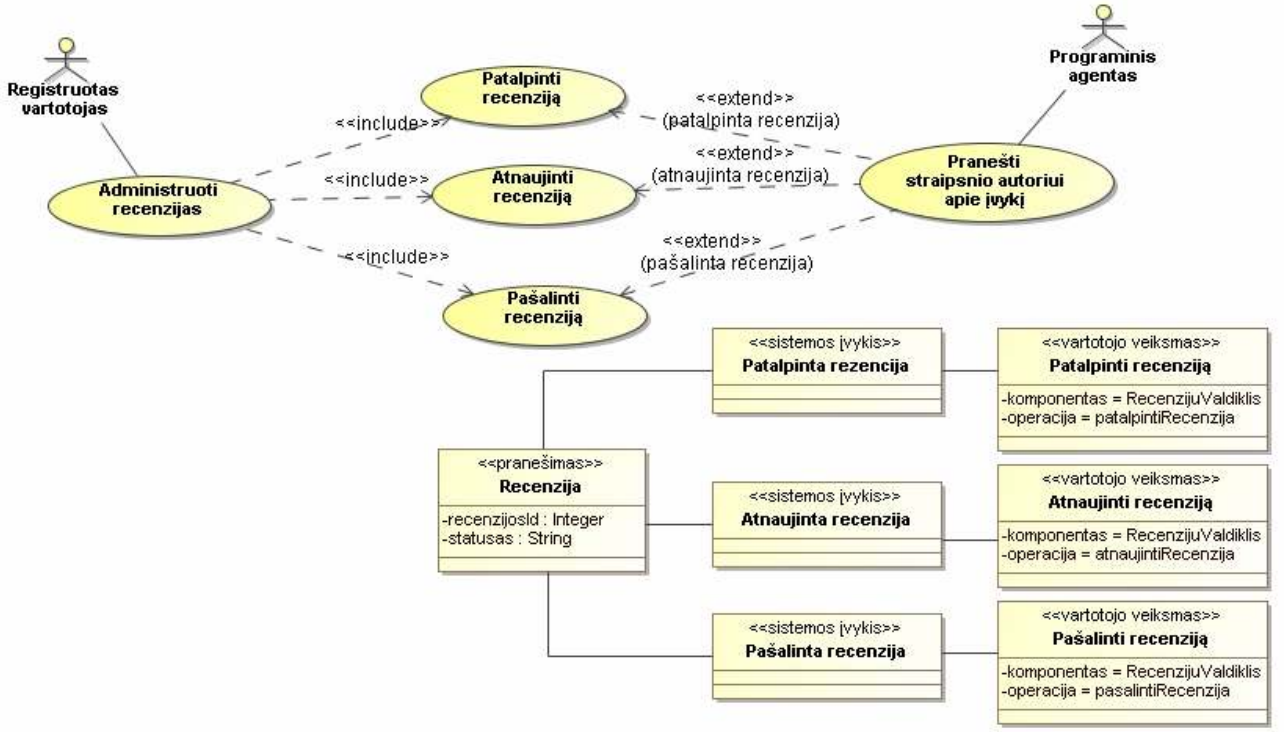
Šiame skyriuje apžvelgiamas įvykių metamodelio pagrindu sudaryto modelio elementų ryšys su *UML* panaudojimo atvejų, klasių, sekų ir būsenų diagramomis.

3.2. Įvykių identifikavimas iš panaudojimo atvejų diagramų

Panaudojimo atvejų diagramos dažniausiai naudojamos reikalavimų analizės etape aprašyti kompiuterizuojamas funkcijas. Sistemose keletas funkcijų gali būti automatizuotos, t.y. jas atlieka programiniai agentai. Programinius agentus taip pat būtina atvaizduoti panaudojimų atvejų

diagramose, nurodant, kokias funkcijas jie atlieka. Programinio agento panaudojimo atvejai (scenarijai) su kitais panaudojimo atvejais jungiami išplėtimo (angl. *extend*) ryšiu, nurodant išplėtimo taškus, t.y. sąlygas, kurioms esant yra atliekami panaudojimo atvejai. Išplėtimo taškai, aprašyti panaudojimo atvejų modelyje ir siejantys programinio agento panaudojimo atvejus su kitais panaudojimo atvejais, gali būti panaudoti įvykių identifikavimui ir atvaizdavimui įvykių modelyje.

3.1 paveikslėlyje yra pavaizduotas įvykių modelio fragmentas ir jo ryšys su panaudojimo atvejų modeliu. Panaudojimo atvejų modelio išplėtimo taškai atitinka įvykių modelio įvykius.



3.1 pav. Įvykių modelio elementų ryšys su panaudojimo atvejų modeliu

3.3. Įvykių sukėlėjai ir pranešimų gavėjai klasių diagramose

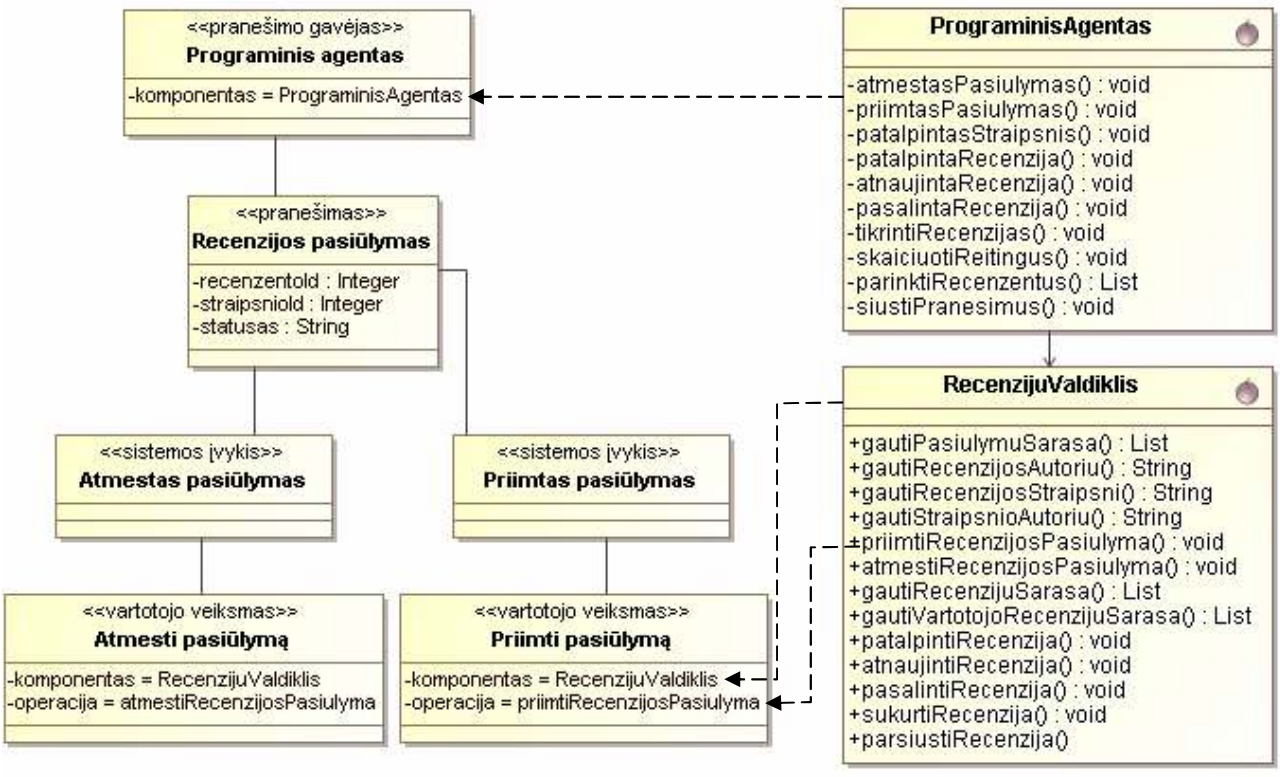
Įvykių metamodelyje nurodyta, kad įvykius gali sužadinti vartotojo veiksmai, pati sistema vykdydama veiklos logiką realizuojančius metodus arba laikmačiai. Vartotojo veiksmai taip pat yra apdorojami metodų (operacijų).

Įvykių modelyje, aprašant įvykių sukėlėjus ir pranešimų gavėjus, gali būti nurodytos klasės, komponentai ar operacijos, kurios sužadina arba apdoroja įvykius. Šie elementai yra modeliuojami klasių diagramose, todėl komponentai, klasės ar jų operacijos aprašomos įvykių modelyje gali būti atvaizduotos ir klasių modelyje.

3.2 paveikslėlyje pavaizduotas įvykių modelio ir klasių diagramos fragmentas, kuriame matyti, kad vartotojo veiksmai (įvykių sukėlėjai) yra aprašyti nurodant komponentus ir jų operacijas, kurios tuos įvykius sužadina.

Aprašant pranešimų gavėją (programinis agentas) taip pat nurodomas komponentas, kuris apdoroja gaunamus pranešimus.

Taip pat modeliuojant įvykių sukėlėjus ir pranešimų gavėjus galima nurodyti ir paketus, kuriuose yra šiuos elementus realizuojantys komponentai ar klasės.



3.2 pav. Įvykių modelio elementų ryšys su klasių diagramomis

3.4. Pranešimų siuntimas sekų diagramose

Sekų diagramos vaizduoja bendradarbiavimą tarp klasių ar komponentų. Šie elementai bendradarbiauja siųsdami vieni kitiems pranešimus arba kviesdami atitinkamas operacijas.

Sekų diagramose galima atvaizduoti šiuos įvykių modelio elementus:

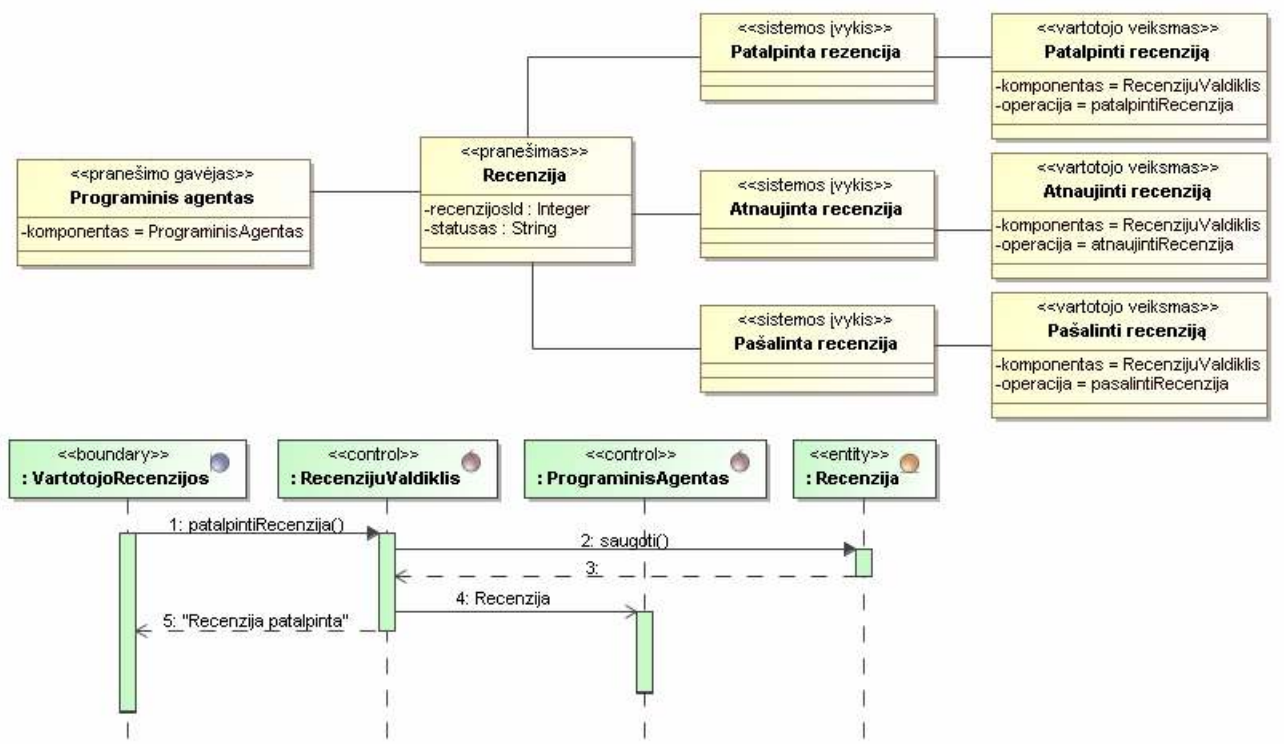
- komponentus ar klasės, kurios realizuoja įvykių sukėlėjus;
- operacijas, kurias vykdant sužadinami įvykiai;
- pranešimus, kurie yra siunčiami tarp klasių ir komponentų;
- komponentus ar klases, kurie įvykius apdoroja (pranešimų gavėjus).

Sekų diagramose nurodant siunčiamus pranešimus nevaizduojama pranešimo struktūra, t.y. nėra matoma, kokia informacija turi būti perduodama siunčiant pranešimus. Pranešimo informacija gali būti matoma iš įvykių modelio.

Sekų diagramas ir įvykių modelį sieja tokios taisyklės:

- pranešimą turi siųsti įvykio sukėlėjo elemente aprašyta klasė ar komponentas;
- pranešimas yra siunčiamas kai iškviečiama įvykio sukėlėjo elemente aprašyta operacija;
- pranešimas yra siunčiamas pranešimo gavėjo elemente nurodytai klasei ar komponentui.

Įvykio modelio fragmentas ir sekų diagrama yra pavaizduota 3.3 paveikslėlyje.



3.3 pav. Įvykio modelio ryšys su sekų diagramomis

3.5. Įvykių atvaizdavimas būsenų diagramose

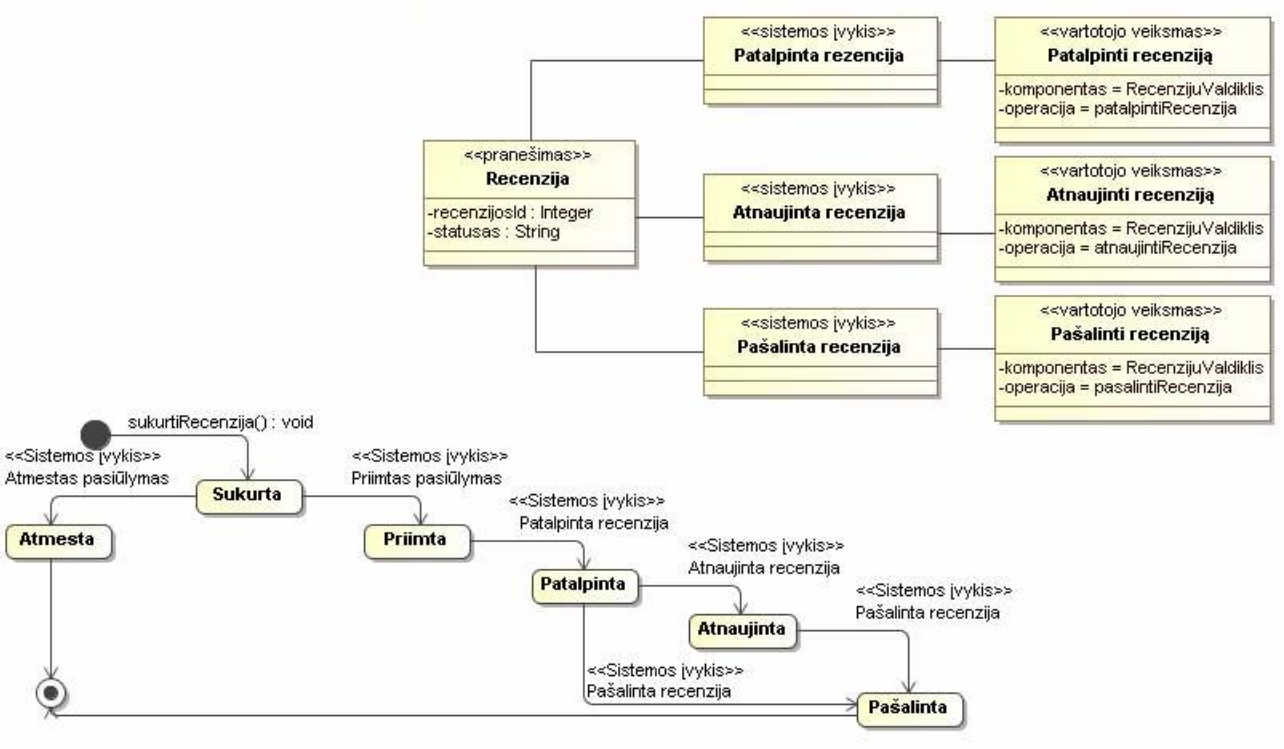
Būsenų diagramos naudojamos atvaizduoti objekto būsenų kaitą. Objektas pereina iš vienos būsenos į kitą kai yra iškviečiama atitinkama operacija, įvyksta numatytas įvykis, praeina tam tikras laiko tarpas po įvykio ir pan., todėl sudarant būsenų diagramas galima naudoti įvykio modelio elementus.

Yra trys galimi būdai panaudoti įvykio modelio elementus būsenų diagramose.

Virš perėjimus iš vienos būsenos į kitą vaizduojančių rodyklių galima rašyti įvykius (laiko ar sistemos), kuriems įvykstant objektas pakeičia būseną. Taip pat galima vaizduoti ir operacijas,

aprašytas įvykio sukėlėjo elemente. Tai reiškia, kad objektas pereina į kitą būseną, kai yra iškviečiama atitinkama klasės operacija. Trečias būdas įvykių modelio elementus panaudoti būsenų diagramose yra pavaizduoti virš būsenų perėjimus žyminčių rodyklių laiko momentus, aprašytus laiko įvykiuose (objektas pakeičia būseną tam tikrais laiko momentais).

Įvykių modelio fragmentas ir būsenų diagrama yra pavaizduoti 3.4 paveikslėlyje.



3.4 pav. Įvykio modelio ryšys su būsenų diagramomis

3.6. Įvykių modeliavimo ir realizavimo metodika

Apibendrinus 3.2–3.5 poskyriuose pateiktą informaciją galima sudaryti tokią įvykio modelio panaudojimo projektavimo procese metodiką:

1. reikalavimų etape įvykiai identifikuojami panaudojimo atvejų modelyje, panaudojimo atvejų išplėtimo taškuose;

2. identifikavus įvykius, reikalavimų analizės etape sudaromas pradinis įvykių modelis, kuriame išvardinami visi pranešimai, įvykiai, jų sukėlėjai ir pranešimų gavėjai. Šis modelis neprivalo būti detalus; jame nebūtina nurodyti klases, komponentus ir operacijas, nes neatlikus projektavimo šie elementai nėra tiksliai žinomi;

3. sistemos architektūrinio projektavimo etape reikia apibrėžiami visi komponentai ir klasės, kurie realizuos įvykių sukėlėjus ir pranešimų gavėjus. Taip pat būtina įvardinti laikmačius, kurie generuos laiko įvykius;

4. kai žinomi komponentai ir klasės, kurie realizuos įvykių sukėlėjus ir pranešimų gavėjus, reikalavimų analizės etape sudarytas įvykių modelis yra modifikuojamas: nurodomos operacijos,

kurias vykdant sužadinami įvykiai, taip pat apibrėžiami laikmačiai, kurie generuos laiko įvykius, apibrėžiama pranešimų struktūra, nurodant atributus, kurie nusako, kokie duomenys turi būti siunčiami pranešime;

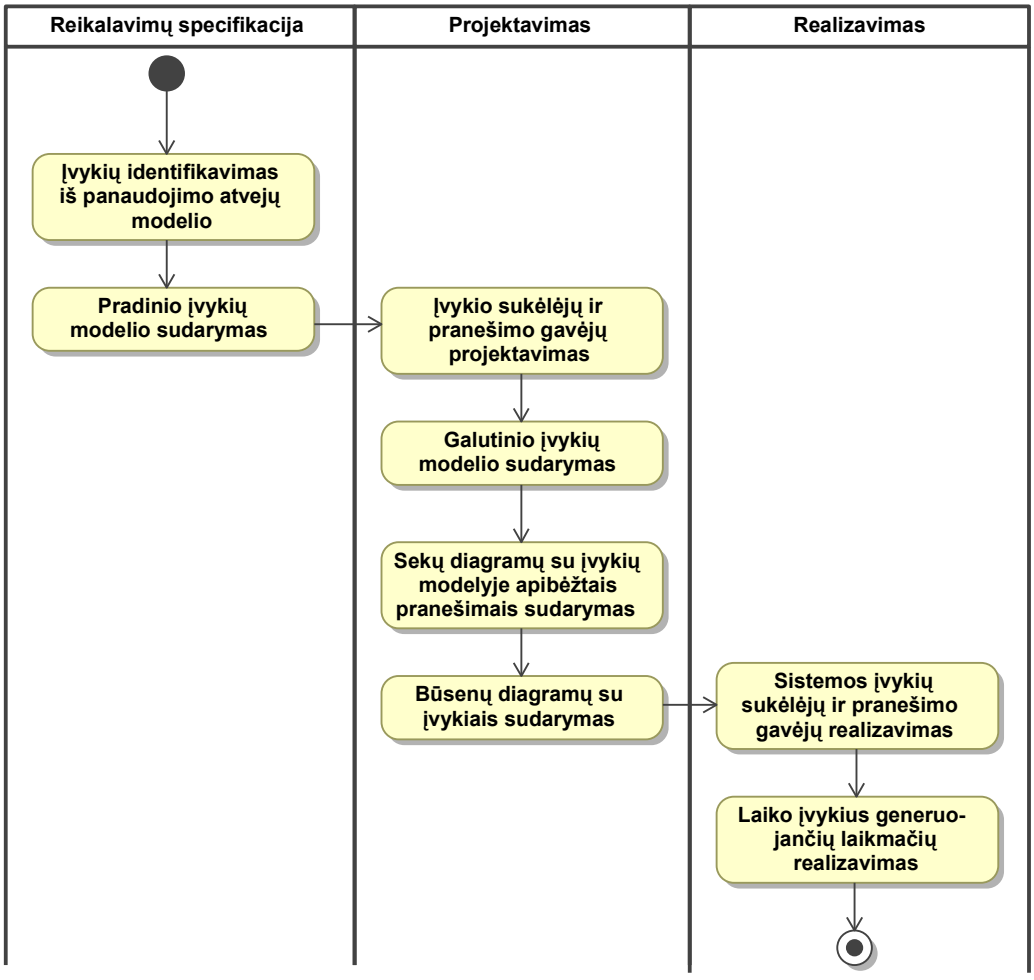
5. turint tikslų laiko įvykių modelį ir sudarant sekų diagramas, vaizduojančias bendradarbiavimą tarp klasių, panaudojami įvykių modelyje aprašyti pranešimai, nurodant kokią operaciją iškvietus yra sužadinamas įvykis ir siunčiamas pranešimas;

6. sudarant būsenų diagramas, virš būsenos pasikeitimą žyminčių rodyklių nurodomi tokie įvykių modelio elementai: įvykis, operacija arba laiko momentas. Šie elementai nurodo sąlygas, kurioms esant pasikeičia objekto būsena;

7. sumodeliuoti sistemos įvykiai realizuojami suprogramuojant komponentus, klases ir jų operacijas, kurie buvo apibrėžti įvykių modelyje prie kiekvieno įvykio. nurodant sistemos elementus, kurie konkretų įvykį realizuoja;

8. laiko įvykiai realizuojami suprogramuojant įvykių modelyje apibrėžtus laikmačius.

Įvykių modelio panaudojimo įvykiais grindžiamų informacinių sistemų projektavimo procese diagrama pavaizduota 3.5 paveikslėlyje.



3.5 pav. Įvykių modelio panaudojimas įvykiais grindžiamų informacinių sistemų projektavimo procese

4. Publikacijų portalo prototipo reikalavimų analizė ir specifikacija

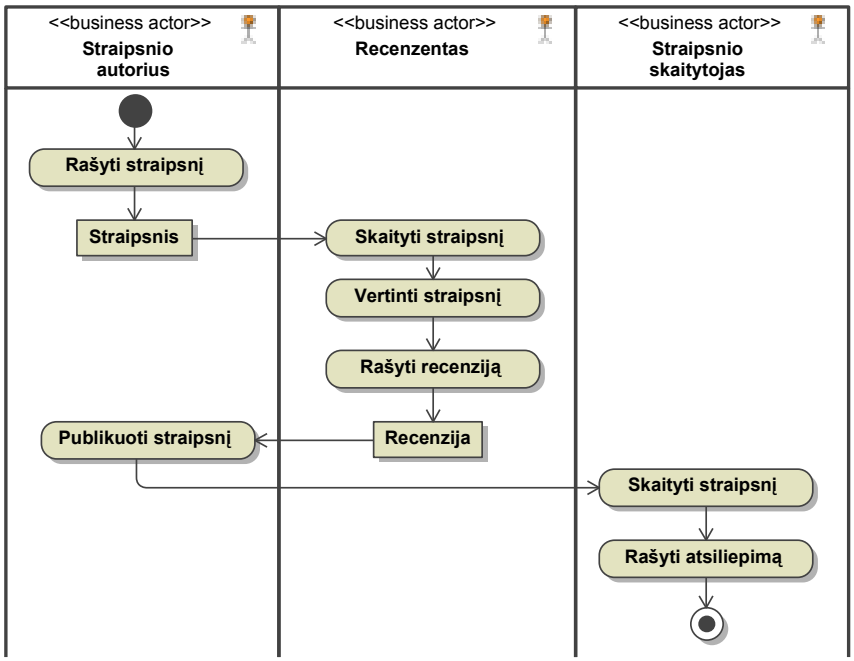
Šiame skyriuje apibrėžiami reikalavimai kuriamam publikacijų portalo prototipui, aprašomos kompiuterizuojamos sistemos funkcijos. Reikalavimai prototipui yra specifikuojami sudarant sekų diagramas ir lenteles, kuriose aprašomi vartotojo atliekamų veiksmų su sistema seka. Skyriaus pabaigoje pateikiamas pradinis įvykių modelis, aprašantis sistemos generuojamus ir vartotojo sužadinamus įvykius.

4.1. Veiklos proceso analizė

Vienas iš magistrinio darbo uždavinių yra sukurti publikacijų portalo prototipą, sudarant detalų įvykiais grindžiamos sistemos modelį ir jį realizuojant. Išsamus sistemos modelis pradedamas sudaryti analizuojant veiklą, kurią reikės kompiuterizuoti. Tam gali būti panaudojama veiklos procesų diagrama, kuri aprašo būsimų sistemos vartotojų atliekamus veiksmus.

4.1 paveikslėlyje pavaizduota veiklos procesų diagrama aprašo straipsnio autoriaus, skaitytojo ir recenzento atliekamus veiksmus. Pirmiausiai autorius parašo straipsnį, kuris atiduodamas įvertinti recenzentui. Recenzentas įvertina straipsnį ir parašo recenziją. Straipsnio autorius gali publikuoti savo straipsnį, t.y. pateikti jį laisvai prieinamą kitiems bendruomenės nariams. Straipsnio skaitytojai, perskaitę juos dominančius straipsnius, gali parašyti jų įvertinimus.

Svarbu įvertinti, kad bendruomenės narių grupės nėra griežtai atskirtos. Straipsnio skaitytojas gali būti kitų straipsnių autorius arba recenzentas. Recenzentai taip pat gali būti straipsnių skaitytojai ar rašyti savo straipsnius.



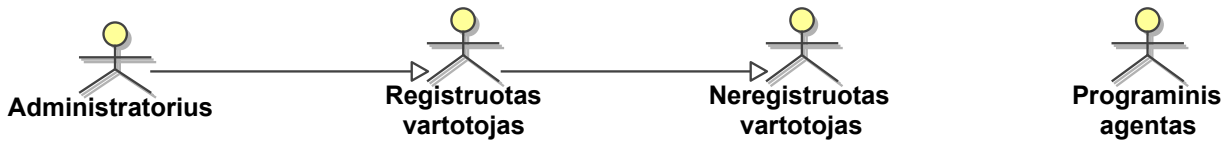
4.1 pav. Straipsnio vertinimo ir publikavimo proceso diagrama

4.2. Vartotojų tipai

Atlikus veiklos proceso analizę ir nustačius pagrindinius veikėjus, galima apibrėžti kuriamos sistemos vartotojus.

Sukurtame publikacijų portale prototipe sistemos vartotojai galės patalpinti savo straipsnius, atlikti straipsnių paiešką, recenzavimą, rašyti komentarus ir vertinti straipsnius. Neregistruotas sistemos vartotojas galės atlikti straipsnių paiešką ir perskaityti jį dominančius straipsnius. Užsiregistravęs – talpinti savo straipsnius, rašyti komentarus ir recenzuoti straipsnius. Aukščiausio lygio (turintys didžiausią reitingą) vartotojai galės atlikti sistemos administravimo darbus: įrašyti naujus temų pavadinimus, įvesti rangavimo algoritmo koeficientus.

Sistemos vartotojų tipai pavaizduoti 4.2 paveikslėlyje.



4.2 pav. Sistemos vartotojų tipai

Užsiregistravusio vartotojo reitingas pradžioje yra lygus 0. Atsižvelgiant į parašytų komentarų, recenzijų ir straipsnių kiekį, programiniai agentai perskaičiuos vartotojo reitingą. Taigi priklausomai nuo vartotojo aktyvumo, jo reitingas didėja. Reitingo skaičiavimo algoritmas atsižvelgia ne tik parašytų straipsnių kiekį, bet ir į straipsnių aktualumą (kiek parašyta komentarų ir kaip įvertintas).

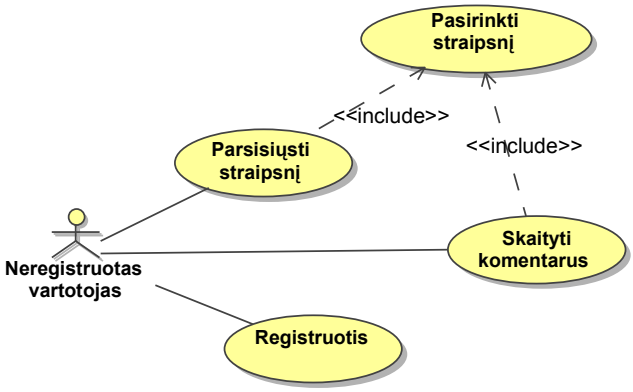
Vartotojų reitingas visuomet yra santykinis ir skaičiuojamas pagal didžiausią reitingą turinčių vartotojų aktyvumą. Pavyzdžiui, jei kuris nors vartotojas yra parašęs komentarų daugiausiai, tai jo komentarų dedamoji galutiniame reitingo įvertinime yra lygi 1; kitų vartotojų ši dedamoji paskaičiuojama komentarų skaičių padalinus iš vartotojo, kurio dedamoji yra lygi 1, parašytų komentarų skaičiaus,.

Taip pat atliekant vartotojų analizę, svarbu išskirti programinius agentus, kaip atskirus sistemos vartotojus. Šis vartotojų tipas naudojamas sudarant panaudojimo atvejų diagramas; panaudojimo atvejai, vykdomi programinių agentų, atspindi sistemos funkcijas, kurios turi buti automatizuotos.

4.3. Kompiuterizuojamos sistemos funkcijos

4.2 poskyryje, atliekant vartotojų tipų analizę, išskirti keturi vartotojų tipai: neregistruotas vartotojas, registruotas vartotojas, administratorius ir programinis agentas. Šiame skyrelyje pateiktas panaudojimo atvejų modelis, vaizduojantis kiekvieno vartotojo atliekamas sistemos funkcijas.

Neregistruotas sistemos vartotojas (4.3 pav.), gali parsisiųsti straipsnius bei skaityti jų komentarus. Norint atlikti šias funkcijas būtina norimą straipsnį surasti ir pasirinkti. Neregistruotas sistemos vartotojas taip pat gali registruotis.



4.3 pav. Neregistruoto vartotojo panaudojimo atvejų modelis

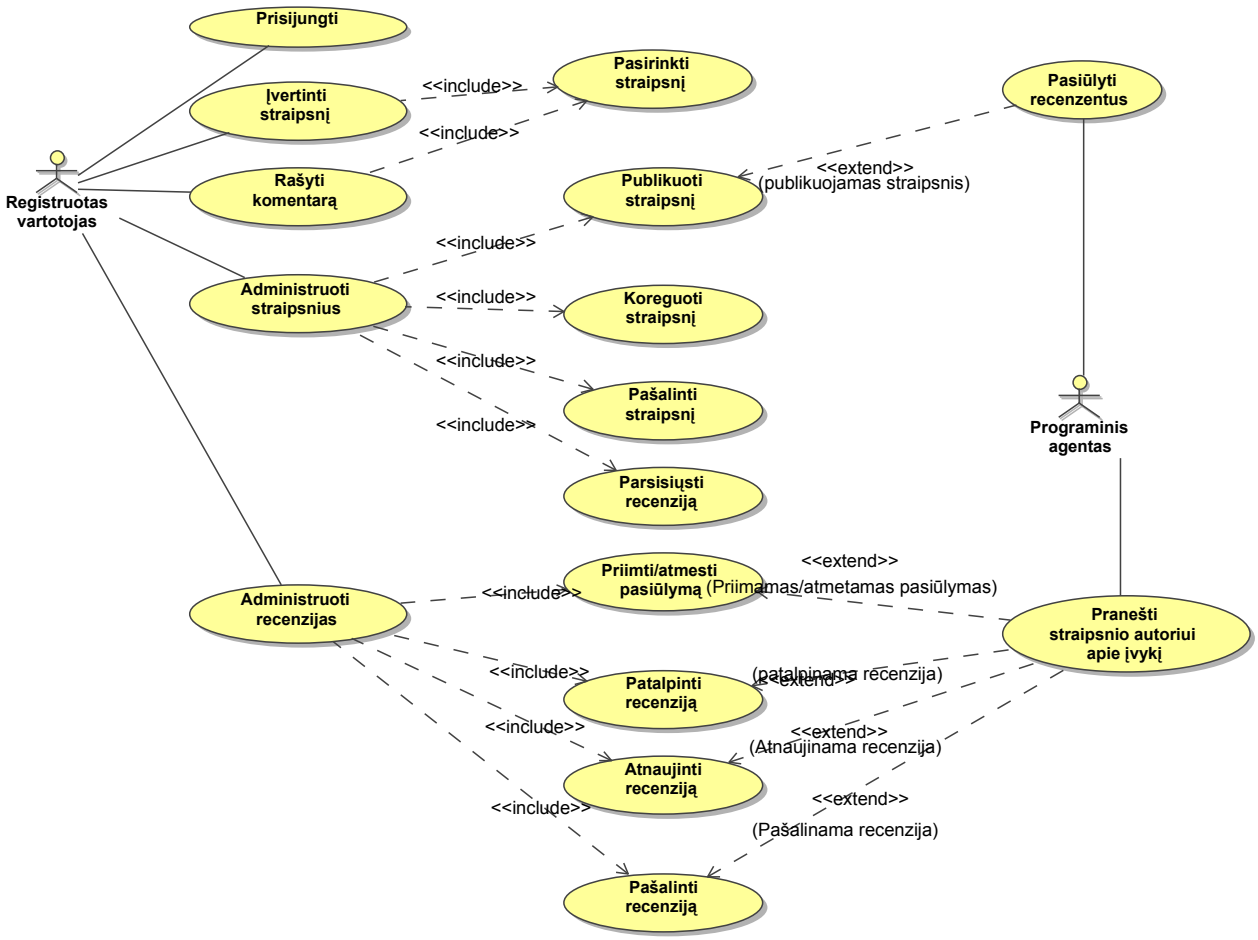
Registruotas sistemos vartotojas gali prisijungti prie sistemos ir pasirinkęs straipsnį jį įvertinti bei parašyti komentarą. Atliekant straipsnių administravimą autorius (registruotas vartotojas) publikuoja straipsnį, vėliau gali jį koreguoti ar pašalinti. Straipsnio autorius taip pat gali parsisiųsti savo straipsnio recenzijas.

Recenzijų administravimas apima tokias funkcijas: recenzijų rašymo pasiūlymo priėmimas ar atmetimas, recenzijos patalpinimas, atnaujinimas ir pašalinimas.

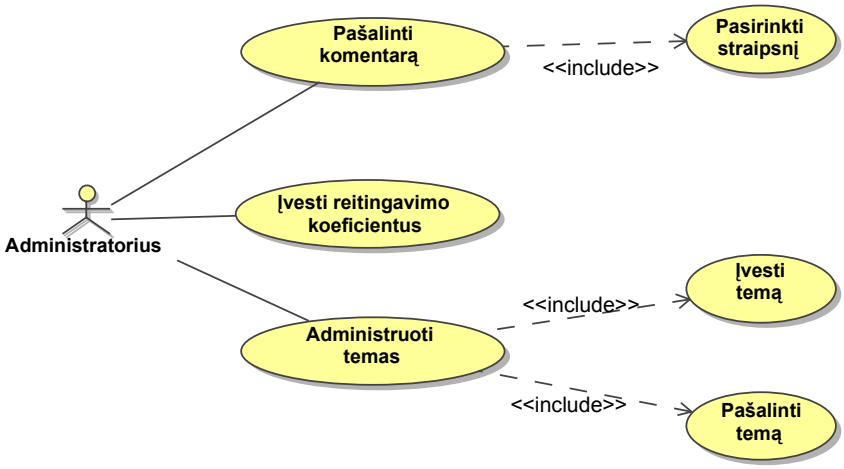
Registruotam vartotojui pasiūlymą rašyti recenziją nusiunčia programinis agentas. Jis taip pat nusiunčia pranešimus straipsnio autoriui, kuomet recenzentas atlieka vieną iš recenzijų administravimo funkcijų.

Registruoto sistemos vartotojo panaudojimo atvejai pateikti 4.4 paveikslėlyje.

Vartotojai turintys didžiausią reitingą yra sistemos administratoriai. Jie gali administruoti temas, pašalinti komentarus, įvesti rangavimo algoritmo koeficientus. Norint pašalinti komentarus, reikia pasirinkti straipsnį, kurio komentarą norima pašalinti. Administratoriaus panaudojimo atvejai pavaizduoti 4.5 paveikslėlyje.



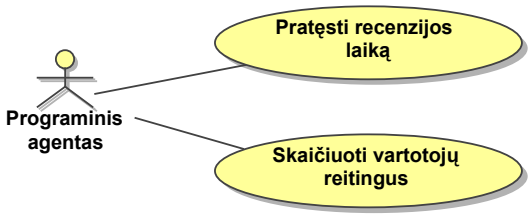
4.4 pav. Registruoto vartotojo panaudojimo atvejų modelis



4.5 pav. Administratoriaus panaudojimo atvejų modelis

Sistemoje turi būti generuojami įvykiai, kuriems įvykus, programinis agentas patikrina, ar recenzentai jau patalpino straipsnių recenzijas, ir praneša recenzentams apie neparašytas recenzijas.

Taip pat tam tikrais laiko momentais programinis agentas perskaičiuoja vartotojų reitingus pagal jų aktyvumą. Programinio agento papildomi panaudojimo atvejai pateikti 4.6 paveikslėlyje.



4.6 pav. Programinio agento papildomi panaudojimo atvejai

Kiekvieno panaudojimo atvejo paskirtis aprašyta žemiau pateiktoje lentelėje.

4.1 lentelė. Sistemos panaudojimo atvejų paskirtis

Sistemos vartotojo tipas	**Panaudojimo atvejis**	**Paskirtis**
Neregistruotas vartojas	Registruotis	Priregistruoti naują sistemos vartotoją
	Pasirinkti straipsnį	Surasti ir pasirinkti straipsnį portale
	Parsisiųsti straipsnį	Parsisiųsti publikuotą straipsnį
	Skaityti komentarus	Perskaityti straipsnio komentarus
Registruotas vartojas	Prisijungti	Prisijungti prie sistemos
	Rašyti komentarą	Rašyti straipsnio komentarus
	Įvertinti straipsnį	Įvertinti straipsnį balais
	Administruoti straipsnius	Publikuoti, koreguoti, šalinti savo publikuotus straipsnius bei perskaityti jų recenzijas
	Publikuoti straipsnį	Patalpinti straipsnį portale
	Koreguoti straipsnį	Koreguoti patalpintą straipsnį
	Pašalinti straipsnį	Pašalinti patalpintą straipsnį
	Skaityti recenzijas	Perskaityti savo straipsnio recenzijas
	Administruoti recenzijas	Patalpinti, pašalinti ir atnaujinti, parašytas recenzijas, bei priimti arba atmesti pasiūlymus rašyti recenzijas
	Patalpinti recenziją	Patalpinti straipsnio recenziją
	Pašalinti recenziją	Pašalinti patalpintą recenziją
	Atnaujinti recenziją	Atnaujinti patalpintą recenziją
	Priimti/atmesti pasiūlymą	Priimti arba atmesti pasiūlymą rašyti recenziją
Administratorius	Administruoti temas	Tvarkyti registruotų vartotojų duomenis
	Įvesti temą	Įvesti naują straipsnių temą
	Pašalinti temą	Pašalinti straipsnių temą
	Įvesti rangavimo koeficientus	Įvesti vartotojų rangavimo algoritmo koeficientus
Programinis agentas	Pranešti straipsnio autoriui apie įvykį	Pranešti straipsnio autoriui apie patalpintą, atnaujintą arba pašalintą recenziją, bei apie pasiūlymo priėmimą ar atmetimą
	Pasiūlyti recenzentus	Pasiūlyti recenzentams vertinti straipsnį
	Pratęsti recenzijos laiką	Patęsti laiką, iki kurio reikia parašyti recenziją
	Skaičiuoti vartotojų reitingus	Skaičiuoti vartotojų reitingus pagal jų aktyvumą

4.4. Nefunkciniai reikalavimai ir apribojimai

Nefunkciniai reikalavimai keliami publikacijų portalo prototipui pateikti 4.2 lentelėje.

4.2 lentelė. Žinių portalui keliami nefunkciniai reikalavimai

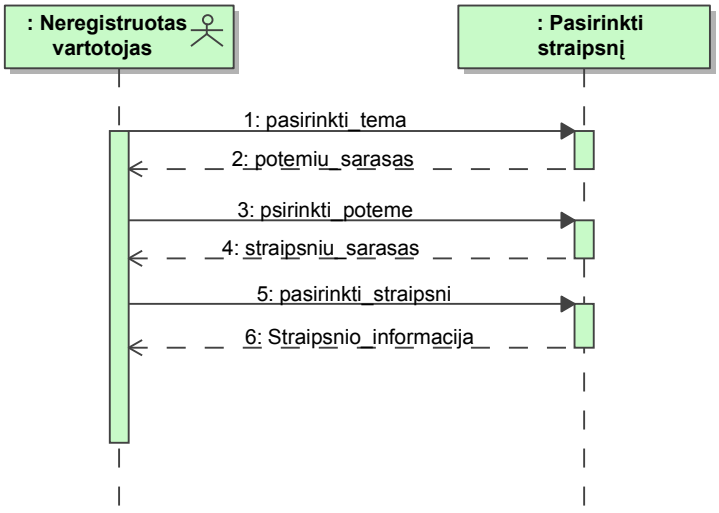
Nefunkciniai reikalavimai	Reikalavimų apibrėžimas
1. Techninei įrangai	Publikacijų portalas prototipas realizuojamas Java EE 5.0 technologi–jomis, todėl serveris, kuriame bus įdiegtas žinių portalas, turi tenkinti Java EE 5.0 taikomųjų uždavinių serverio keliamus reikalavimus techninei įrangai. Vartotojo sąsaja pateikiama naršyklėje, todėl kliento kompiuteris turi tenkinti techninius reikalavimus, kurie būtini naršyklės tinkamam veikimui.
2. Programinei įrangai	Serveryje turi būti įdiegtas *Sun Java System Application Server 9.1* taikomųjų uždavinių serveris, kuris palaiko šias technologijas: EJB 3.0 (*Enterprise JavaBeans*), Servlet API 2.3 ir JSP 1.2. Serveryje turi būti įdiegta *Derby* duomenų bazių valdymo sistema. Vartotojo kompiuteryje turi būti įdiegta naršyklė palaikanti HTML 4.0 versiją, *JavaScript* scenarijų kalbą.
3. Funkcionalumui	Programiniai agentai turi įvykdyti jiems pavestas užduotis per priimtiną laiką. Užklausos, vykdomos *Derby* duomenų bazėje, turi būti atliktos per minimalų laiką.
4. Patikimumui	Įvykus klaidoms, sistema turi apie tai pranešti vartotojui ir toliau tęsti darbą. Išsijungus serveriui, programiniai agentai turi atlikti nebaigtas užduotis iškart įjungus serverį.
5. Perkeliamumui	Sukurtas publikacijų portalo prototipas turi būti perkeliamas į kitą Java EE 5.0 technologijas palaikantį taikomųjų uždavinių serverį, nereikalaujant programinio kodo pakeitimų.
6. Saugumui	Tam tikromis sistemos funkcijomis gali naudotis tik atitinkamas teises turintys vartotojai. Vartotojas identifikuojamas prisijungimo vardu ir slaptažodžiu. Turi būti apsaugotas priėjimas prie tų sistemos funkcijų, kuriomis atitinkami vartotojai neturi teisės naudotis, t.y. vartotojo teisėmis turi būti apibrėžtos vartotojo galimybes naudotis tam tikromis sistemos funkcijomis.
7. Vartotojo sąsajai	Vartotojo sąsaja turi būti realizuota naršyklėje, pagal šiuolaikines technologijas. Meniu punktai turi būti pateikti lietuvių kalba ir suprantami paprastam sistemos vartotojui. Vartotojo sąsaja taip pat turi būti intuityvi, lengvai suprantama.
8. Duomenims	Sistemos vartotojui įvedant duomenis turi būti tikrinama jų sintaksė ir, kur įmanoma, jų semantika. Nutrūkus ryšiui tarp kliento ir serverio, turi būti atstatytas duomenų vientisumas iki paskutinės transakcijos. Duomenys turi būti saugomi *Derby* duomenų bazėje.
9. „Savęs palaikymui“	Žinių portalas turėtų veikti kaip save prižiūrintis mechanizmas, t.y. turėtų reikalauti kuo mažiau administravimo darbų.

4.5. Reikalavimų specifikacija

Neregistruoto vartotojo panaudojimo atvejų modelis pateiktas 4.3 paveikslėlyje. Neregistruotas vartotojas gali parsisiųsti straipsnį bei skaityti jo komentarus, tačiau prieš tai jam reikia pasirinkti straipsnį. Panaudojimo atvejo „*Pasirinkti straipsnį*“ specifikacija pateikta 4.3 lentelėje, o sekų diagrama, rodanti vartotojo ir sistemos sąveiką – 4.7 paveikslėlyje.

4.3 lentelė. Panaudojimo atvejo „*Pasirinkti straipsnį*“ specifikacija

Panaudojimo atvejis „*Pasirinkti straipsnį*“		
Aktorius		Neregistruotas vartotojas
Prieš sąlyga		Norima straipsnio tema duomenų bazėje užregistruota
Sužadinimo sąlyga		Vartotojas nori rasti jį dominantį straipsnį
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka temą		1.1. Sistema pateikia potemių sąrašą; jei reikiamos temos nėra, baigiamas panaudojimo atvejis
2. Vartotojas pasirenka potemę		2.1 Sistema pateikia straipsnių sąrašą; jei reikiamos potemės nėra, baigiamas panaudojimo atvejis
3. Vartotojas pasirenka straipsnį		3.1. Sistema pateikia informaciją apie straipsnį
Po sąlyga		Vartotojui pateikti duomenys apie pasirinktą straipsnį
Alternatyvūs scenarijai		
–		–



4.7 pav. Straipsnio pasirinkimo sekų diagrama

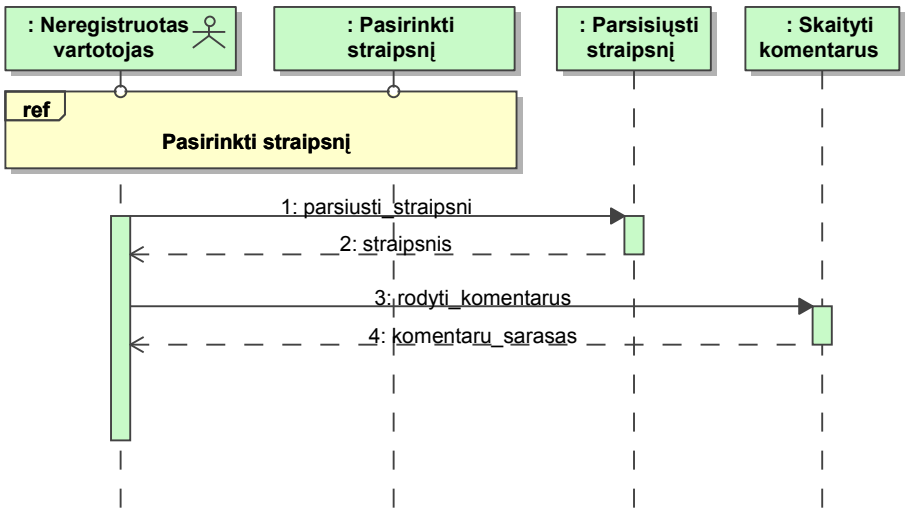
Neregistruoto vartotojo panaudojimo atvejų – „*Parsisiųsti straipsnį*“ ir „*Skaityti komentarus*“ – specifikacijos pateiktos 4.4–4.5 lentelėse, o sekų diagrama – 4.8 paveikslėlyje.

4.4 lentelė. Panaudojimo atvejo „*Parsisiųsti straipsnį*“ specifikacija

Panaudojimo atvejis „*Parsisiųsti straipsnį*“		
Aktorius		Neregistruotas vartotojas
Prieš sąlyga		
Sužadinimo sąlyga		Vartotojas nori parsisiųsti portale publikuotą straipsnį
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„*Pasirinkti straipsnį*“
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka straipsnį		1.1. Vykdomas panaudojimo atvejis „*Pasirinkti straipsnį*‘
2. Vartotojas paspaudžia parsisiuntimo mygtuką		2.1. Sistema pateikia dialogą straipsnio išsaugojimui vartotojo kompiuteryje
3. Vartotojas pasirenka vietą, kur nori išsaugoti straipsnį		3.1. Sistema parsiunčia straipsnį
Po sąlyga		Vartotojas parsisiuntęs norimą straipsnį
Alternatyvūs scenarijai		
–		–

4.5 lentelė. Panaudojimo atvejo „*Skaityti komentarus*“ specifikacija

Panaudojimo atvejis „*Skaityti komentarus*“		
Aktorius		Neregistruotas vartotojas
Prieš sąlyga		
Sužadinimo sąlyga		Vartotojas nori perskaityti portale publikuoto straipsnio komentarus
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„*Pasirinkti straipsnį*“
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka straipsnį		1.1. Vykdomas panaudojimo atvejis „*Pasirinkti straipsnį*‘
2. Vartotojas paspaudžia komentarų mygtuką		2.1. Sistema pateikia straipsnio komentarų sąrašą
Po sąlyga		Vartotojui pateikti straipsnio komentarai
Alternatyvūs scenarijai		
–		–

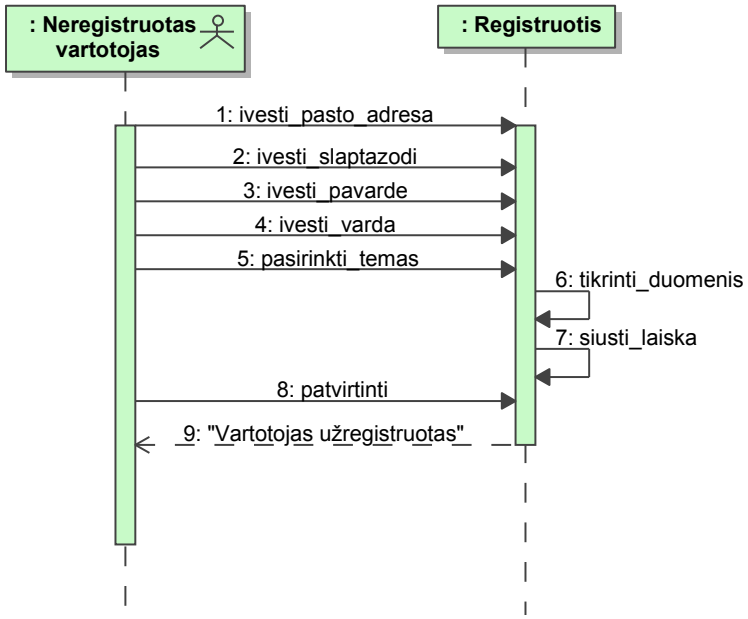


4.8 pav. Straipsnio parsisiuntimo ir komentarų skaitymo sekų diagrama

Registruotas sistemos vartotojas gali naudoti daugiau sistemos teikiamų funkcijų. Registruojantis vartotojas turi įvesti savo vardą, pavardę, slaptažodį, el. pašto adresą bei pasirinkti kompetencijos temas. Sistema patikrina įvestus duomenis ir nusiunčia el. laišką vartotojui. Vartotojui patvirtinus gautą laišką, sistema jį užregistruoja.

4.6 lentelė. Panaudojimo atvejo „*Registruotis*“ specifikacija

Panaudojimo atvejis „*Registruotis*“		
Aktorius		Neregistruotas vartotojas
Prieš sąlyga		Norimu registruotis prisijungimo vardu sistemoje kitas vartotojas neužregistruotas
Sužadinimo sąlyga		Vartotojas nori registruotis sistemoje
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas įveda pašto adresą		
2. Vartotojas įveda slaptažodį		
3. Vartotojas įveda pavardę		
4. Vartotojas įveda vardą		
5. Vartotojas pasirenka temas straipsnių, kuriems gali rašyti recenzijas		
6. Patvirtina		6.1. Sistema patikrina duomenis 6.2. Sistema siunčia pranešimą vartotojui el. Paštu
7. Vartotojas patvirtina gautą laišką		7.1. Sistema užregistruoja vartotoją
Po sąlyga		Naujas vartotojas užregistruotas
Alternatyvūs scenarijai		
6. Vartotojas neįvedė visų duomenų arba įvedė neteisingai		Sistema suformuoja pranešimą ir neregistruoja vartotojo
6. Nurodytas elektroninis paštas jau užregistruotas sistemoje		Sistema suformuoja pranešimą ir neregistruoja vartotojo
7. Vartotojas nepatvirtina gauto laiško		Sistema vartotojui nepriskiria rolių

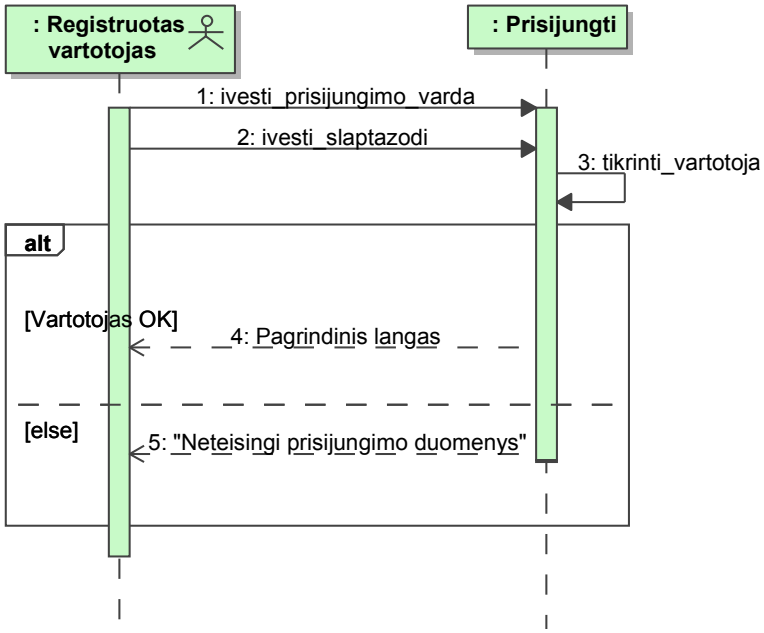


4.9 pav. Registracijos sekų diagrama

Norint naudotis visomis registruotam vartotojui ar administratoriui suteiktomis portalo funkcijomis, reikia prisijungti prie sistemos. Įvedus prisijungimo vardą ir slaptažodį, sistema patikrina įvestus duomenis ir, jei įvesti duomenys teisingi, pateikia pagrindinį langą; priešingu atveju – klaidos pranešimą.

4.7 lentelė. Panaudojimo atvejo „*Prisijungti*“ specifikacija

Panaudojimo atvejis „*Prisijungti*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs
Sužadinimo sąlyga		Vartotojas nori prisijungti prie sistemos
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas įveda prisijungimo vardą (el. pašto adresą)		
2. Vartotojas įveda slaptažodį		
3. Patvirtina		3.1. Sistema patikrina vartotoją ir pateikia pagrindinį langą.
Po sąlyga		Vartotojas prisijungęs prie sistemos
Alternatyvūs scenarijai		
3. Vartotojas sistemoje neužregistruotas		Sistema pateikia pranešimą
3. Vartotojas neteisingai įvedė prisijungimo vardą ir/arba slaptažodį		Sistema pateikia pranešimą



4.10 pav. Prisijungimo sekų diagrama

Registruotam vartotojui pasirinkus temą, pateikiamas straipsnių sąrašas. Prie kiekvieno straipsnio yra nuorodos, kurių pagalba vartotojas gali peržiūrėti straipsnio komentarus arba įvertinti straipsnį.

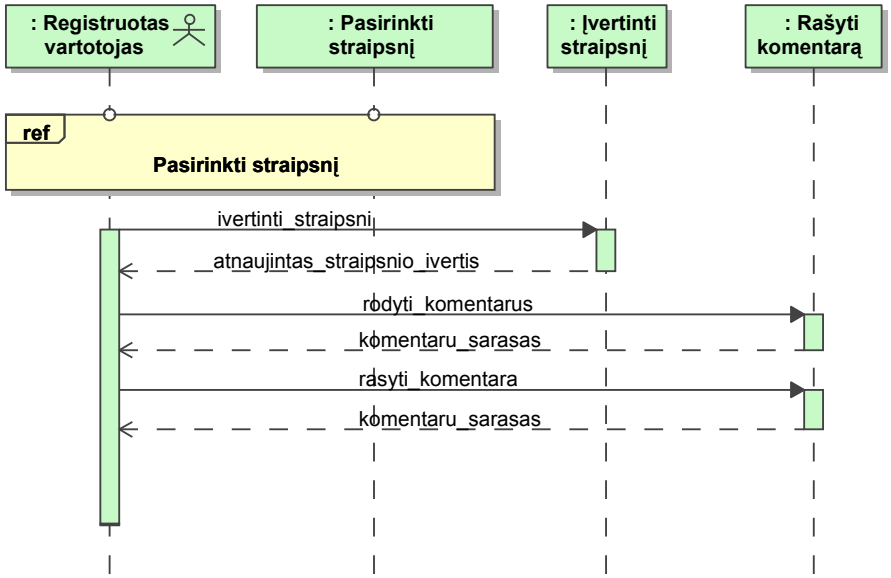
Vartotojui pasirinkus komentarų nuorodą, pateikiamas komentarų sąrašas. Vartotojas gali perskaityti straipsnio komentarus arba parašyti savo komentarą. Registruotas vartotojas taip pat gali įvertinti straipsnį.

4.8 lentelė. Panaudojimo atvejo „*Įvertinti straipsnį*“ specifikacija

Panaudojimo atvejis „*Įvertinti straipsnį*“		
Aktorius		Registruotas sistemos vartotojas
Prieš sąlyga		Vartotojas prisijungęs; vartotojas norimą įvertinti straipsnį nėra vertinęs anksčiau
Sužadinimo sąlyga		Vartotojas nori įvertinti straipsnį
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„*Pasirinkti straipsnį*“
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka straipsnį		1.1. Vykdomas panaudojimo atvejis „*Pasirinkti straipsnį*“
2. Vartotojas įvertina straipsnį		2.1. Sistema suskaičiuoja straipsnio įvertinimą 2.2. Sistema pateikia straipsnio įvertinimą
Po sąlyga		Vartotojo straipsnio įvertinimas išsaugotas duomenų bazėje
Alternatyvūs scenarijai		
2. Vartotojas straipsnį jau įvertinęs		Sistema neleidžia straipsnio vertinti

4.9 lentelė. Panaudojimo atvejo „*Rašyti komentarą*“ specifikacija

Panaudojimo atvejis „*Rašyti komentarą*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs
Sužadinimo sąlyga		Vartotojas nori parašyti straipsnio komentarą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„*Pasirinkti straipsnį*“
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka straipsnį		1.1. Vykdomas panaudojimo atvejis „*Pasirinkti straipsnį*“
2. Vartotojas paspaudžia komentarų mygtuką		2.1. Sistema pateikia komentarų sąrašą
3. Vartotojas parašo komentarą		
4. Patvirtina		4.1. Sistema išsaugo komentarą
Po sąlyga		Vartotojo komentaras išsaugotas duomenų bazėje
Alternatyvūs scenarijai		
–		–



4.11 pav. Straipsnio įvertinimo, komentaro rašymo, recenzijos parsisiuntimo sekų diagrama

Vartotojo straipsnių administravimas apima tokias funkcijas: straipsnio publikavimą, koregavimą bei pašalinimą, recenzijų peržiūrėjimą.

Publikuojant straipsnį vartotojas turi įvesti straipsnio potemę, pavadinimą, trumpą aprašymą, bei įkelti straipsnį. Sistema patikrina įvestus duomenis ir, jei duomenys teisingi, patalpina straipsnį portalo duomenų bazėje.

4.10 lentelė. Panaudojimo atvejo „*Publikuoti straipsnį*“ specifikacija

Panaudojimo atvejis „*Publikuoti straipsnį*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs; straipsnio tema užregistruota
Sužadinimo sąlyga		Vartotojas nori publikuoti straipsnį
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas įveda straipsnio potemę		1.1. Jei reikiamos straipsnio temos nėra, baigiamas panaudojimo atvejis
2. Vartotojas įveda straipsnio pavadinimą		
3. Vartotojas įveda straipsnio aprašymą		
4. Patvirtina		4.1. Sistema patikrina ir išsaugo duomenis 4.2. Sistema parenka recenzentus
Po sąlyga		Vartotojas publikavęs straipsnį portale
Alternatyvūs scenarijai		
4. Vartotojas neteisingai įvedė duomenis		Sistema pateikia pranešimą

Straipsnį koreguoti arba pašalinti galima tik tada, jei dar neprasidėjęs recenzijų rašymo laikas. Norint koreguoti ar pašalinti straipsnį, vartotojui pateikiamas straipsnių sąrašas. Pasirinktą straipsnį vartotojas gali koreguoti arba pašalinti. Jei recenzijų rašymo laikas prasidėjęs, sistema pateikia pranešimą, kad duomenų koreguoti negalima.

4.11 lentelė. Panaudojimo atvejo „*Koreguoti straipsnį*“ specifikacija

Panaudojimo atvejis „*Koreguoti straipsnį*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs; recenzijų rašymo laikas neprasidėjęs
Sužadinimo sąlyga		Vartotojas nori koreguoti straipsnį
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas prašo parodyti straipsnių sąrašą		1.1. Sistema pateikia vartotojo straipsnių sąrašą
2. Vartotojas pasirenka straipsnį		2.1. Sistema pateikia duomenų koregavimo langą
3. Vartotojas pataiso duomenis		
4. Patvirtina		4.1. Sistema patikrina duomenis 4.2. Sistema išsaugo duomenis
Po sąlyga		Straipsnio duomenys duomenų bazėje atnaujinti
Alternatyvūs scenarijai		
2. Recenzijų rašymo laikas prasidėjęs		Sistema neleidžia koreguoti duomenų
4. Vartotojas neteisingai įvedė duomenis		Sistema pateikia pranešimą

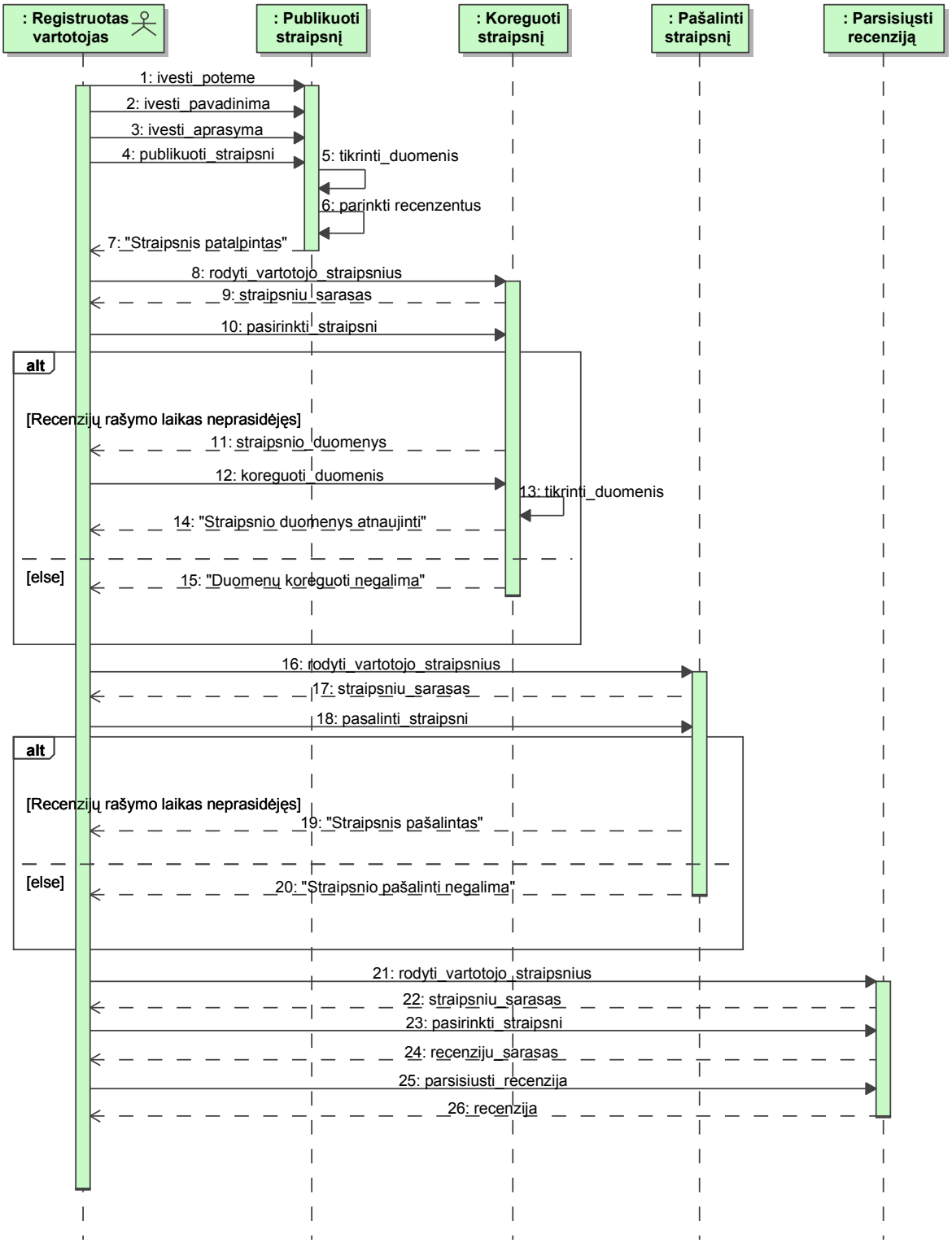
4.12 lentelė. Panaudojimo atvejo „*Pašalinti straipsnį*“ specifikacija

Panaudojimo atvejis „*Pašalinti straipsnį*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs; recenzijų rašymo laikas neprasidėjęs
Sužadinimo sąlyga		Vartotojas nori pašalinti straipsnį
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas prašo parodyti straipsnių sąrašą		1.1. Sistema pateikia vartotojo straipsnių sąrašą
2. Vartotojas paspaudžia pašalinimo mygtuką		2.1. Sistema pašalina straipsnį
Po sąlyga		Straipsnis iš duomenų bazės pašalintas
Alternatyvūs scenarijai		
2. Recenzijų rašymo laikas prasidėjęs		Sistema neleidžia pašalinti straipsnio

Norint parsisiųsti straipsniui parašytas recenzijas, vartotojui pateikiamas sąrašas straipsnio recenzijų, kurias jis gali parsisiųsti.

4.13 lentelė. Panaudojimo atvejo „*Parsisiųsti recenziją*“ specifikacija

Panaudojimo atvejis „*Parsisiųsti recenziją*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs
Sužadinimo sąlyga		Vartotojas nori parsisiųsti savo straipsnio recenziją
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas prašo parodyti straipsnių sąrašą		1.1. Sistema pateikia vartotojo straipsnių sąrašą
2. Vartotojas pasirenka straipsnį		2.1. Sistema pateikia recenzijų sąrašą
3. Vartotojas paspaudžia parsisiuntimo mygtuką		3.1. Sistema pateikia dialogą recenzijos išsaugojimui vartotojo kompiuteryje
4. Vartotojas pasirenka vietą, kur nori išsaugoti recenziją		4.1. Sistema parsiunčia recenziją
Po sąlyga		Vartotojas parsisiuntęs recenziją
Alternatyvūs scenarijai		
–		–



4.12 pav. Straipsnių administravimo sekų diagrama

Vartotojui norint administruoti recenzijas, sistema pateikia langą su recenzijų sąrašu. Šiame lange vartotojas gali patalpinti, atnaujinti arba pašalinti recenziją. Atlikus šiuos veiksmus sistema apie atitinkamą įvykį praneša straipsnio, kurio recenzija buvo administruojama, autoriui.

Autoriui patalpinus straipsnį portale, programinis agentas parenka recenzentus ir nusiunčia jiems pasiūlymus. Recenzentas gali peržiūrėti pasiūlymų sąrašą ir atmesti bei priimti atitinkamus pasiūlymus. Apie recenzento sprendimą, sistema praneša straipsnio autoriui.

4.14 lentelė. Panaudojimo atvejo „*Priimti/atmesti pasiūlymą*“ specifikacija

Panaudojimo atvejis „*Priimti/atmesti pasiūlymą*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs
Sužadinimo sąlyga		Vartotojas nori priimti arba atmesti pasiūlymą rašyti recenziją
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas prašo parodyti pasiūlymų sąrašą		1.1. Sistema pateikia pasiūlymų rašyti recenzijas sąrašą
2. Vartotojas priima pasiūlymą rašyti recenziją		2.1. Sistema pakeičia recenzijos būseną 2.2. Sistema siunčia pranešimą autoriui
Po sąlyga		Recenzijos būsena pakeista
Alternatyvūs scenarijai		
2. Vartotojas atmeta pasiūlymą rašyti recenziją		Sistema pakeičia recenzijos būseną

4.15 lentelė. Panaudojimo atvejo „*Patalpinti recenziją*“ specifikacija

Panaudojimo atvejis „*Patalpinti recenziją*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs; vartotojas priėmęs pasiūlymą rašyti recenziją; recenzijos rašymo laikas nepasibaigęs
Sužadinimo sąlyga		Vartotojas nori patalpinti recenziją
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas prašo parodyti recenzijų sąrašą		1.1. Sistema pateikia recenzijų sąrašą
2. Vartotojas patalpina recenziją		2.1. Sistema pakeičia recenzijos būseną 2.2. Sistema siunčia pranešimą autoriui
Po sąlyga		Recenzijos būsena pakeista
Alternatyvūs scenarijai		
Recenzijos rašymo laikas pasibaigęs		Sistema neleidžia patalpinti recenzijos

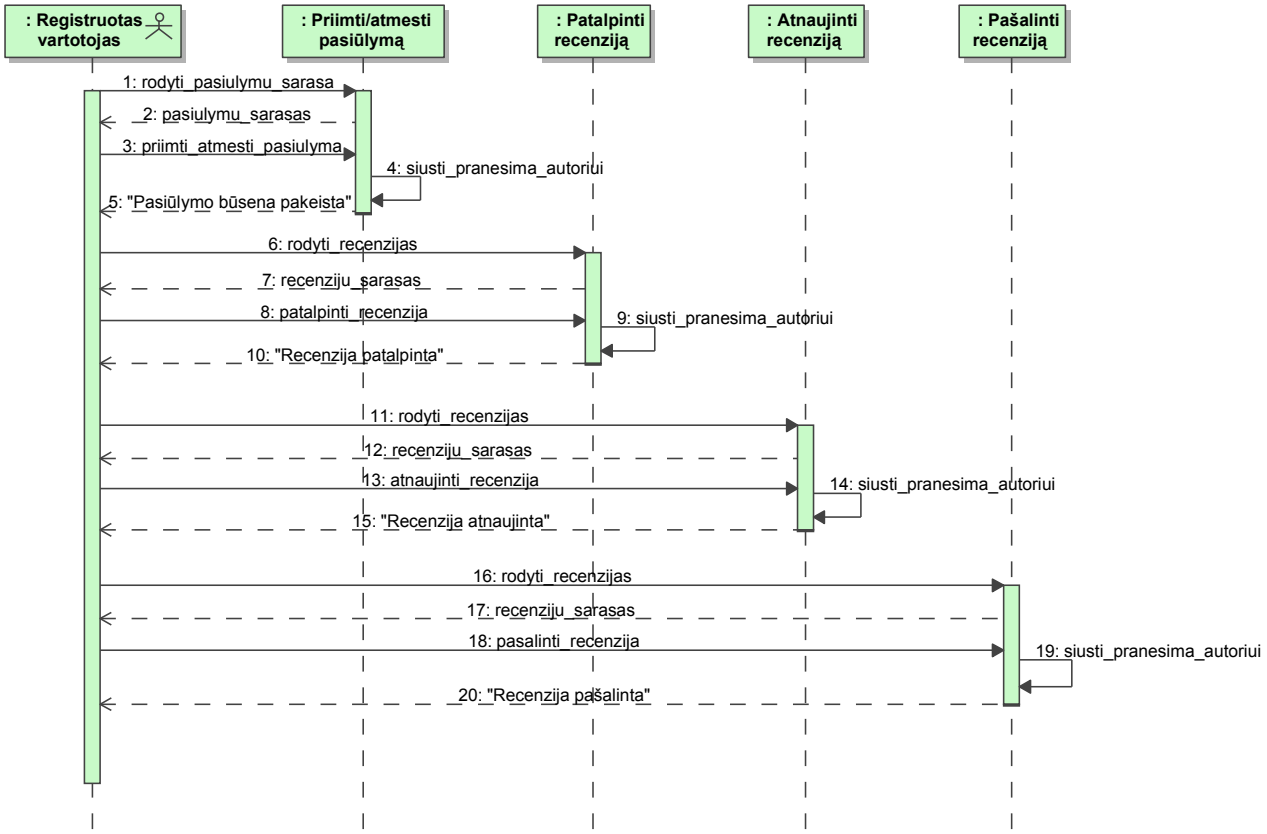
4.16 lentelė. Panaudojimo atvejo „*Atnaujinti recenziją*“ specifikacija

Panaudojimo atvejis „*Atnaujinti recenziją*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs; vartotojas patalpinęs recenziją
Sužadinimo sąlyga		Vartotojas nori atnaujinti recenziją
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**

1. Vartotojas prašo parodyti recenzijų sąrašą	1.1. Sistema pateikia recenzijų sąrašą
2. Vartotojas atnaujina recenziją	2.1. Sistema pakeičia recenzijos būseną 2.2. Sistema siunčia pranešimą autoriui
Po sąlyga	Recenzija atnaujinta
Alternatyvūs scenarijai	
–	–

4.17 lentelė. Panaudojimo atvejo „*Pašalinti recenziją*“ specifikacija

Panaudojimo atvejis „*Pašalinti recenziją*“		
Aktorius		Registruotas vartotojas
Prieš sąlyga		Vartotojas prisijungęs; vartotojas patalpinęs recenziją
Sužadinimo sąlyga		Vartotojas nori pašalinti recenziją
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas prašo parodyti recenzijų sąrašą		1.1. Sistema pateikia recenzijų sąrašą
2. Vartotojas pašalina recenziją		2.1. Sistema pakeičia recenzijos būseną 2.2. Sistema siunčia pranešimą autoriui
Po sąlyga		Recenzija pašalinta
Alternatyvūs scenarijai		
–		–



4.13 pav. Recenzijų administravimo sekų diagrama

4.14 paveikslėlyje pateikta sekų diagrama, rodanti panaudojimo atvejų, atliekamų administratoriaus, realizavimą. 4.18–4.21 lentelėse pateikta šių panaudojimo atvejų specifikacija.

4.18 lentelė. Panaudojimo atvejo „*Įvesti rangavimo koeficientus*“ specifikacija

Panaudojimo atvejis „*Įvesti rangavimo koeficientus*“		
Aktorius		Administratorius
Prieš sąlyga		Vartotojas prisijungęs ir turi administratoriaus teises
Sužadinimo sąlyga		Administratorius nori įvesti rangavimo algoritmo koeficientus
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Administratorius įveda koeficientus		
2. Patvirtina		2.1. Sistema patikrina duomenis 2.2. Sistema išsaugo koeficientus
Po sąlyga		Koeficientai duomenų bazėje įvesti
Alternatyvūs scenarijai		
2. Administratorius neteisingai įvedė duomenis		Sistema pateikia pranešimą

4.19 lentelė. Panaudojimo atvejo „*Pašalinti komentarą*“ specifikacija

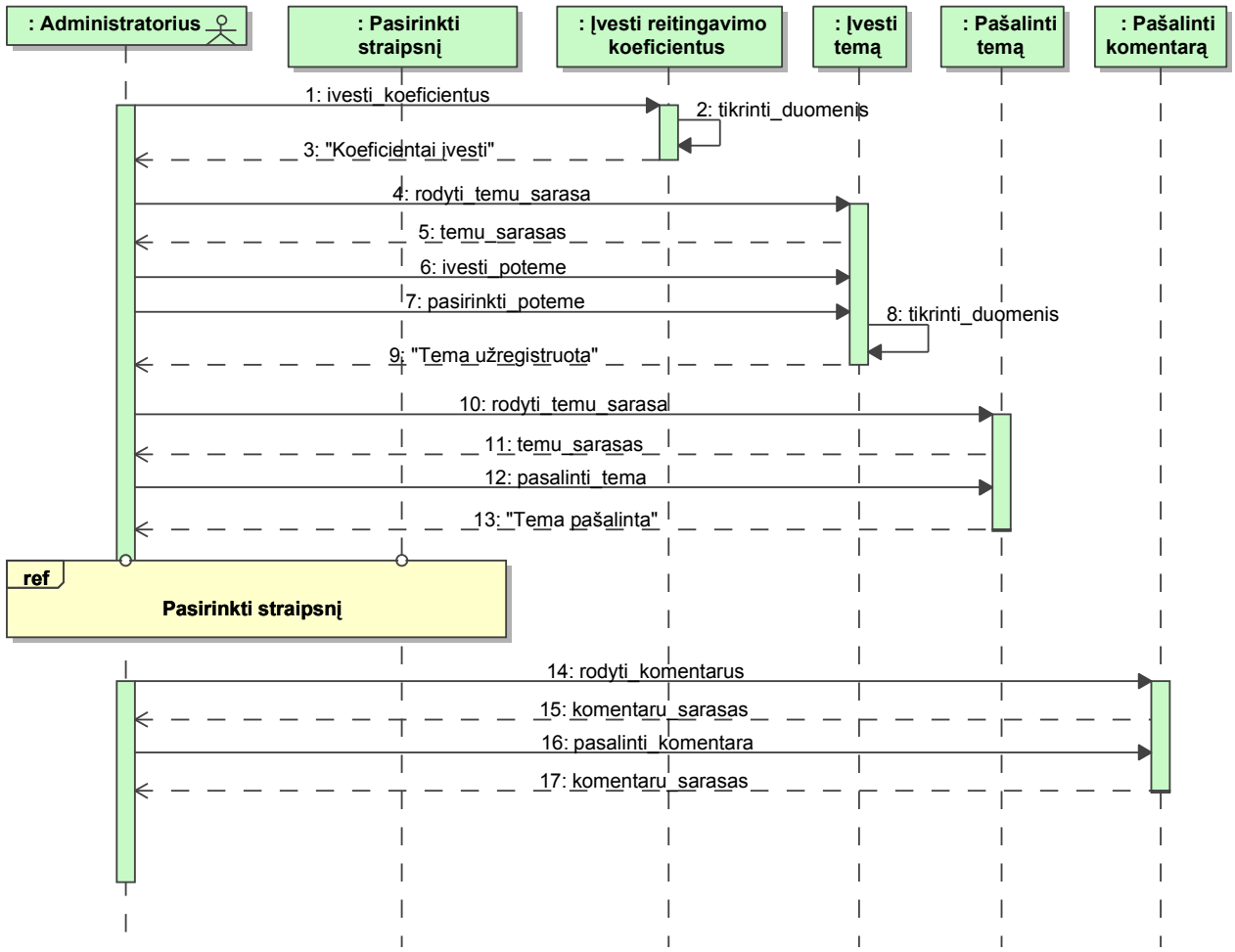
Panaudojimo atvejis „*Pašalinti komentarą*“		
Aktorius		Administratorius
Prieš sąlyga		Vartotojas prisijungęs ir turi administratoriaus teises
Sužadinimo sąlyga		Administratorius nori pašalinti komentarą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„*Pasirinkti straipsnį*“
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Administratorius pasirenka straipsnį		1.1. Vykdomas panaudojimo atvejis „*Pasirinkti straipsnį*‘
2. Administratorius paspaudžia komentarų mygtuką		2.1. Sistema pateikia straipsnio komentarų sąrašą
3. Administratorius paspaudžia pašalinimo mygtuką		3.1. Sistema komentarą pašalina
Po sąlyga		
Alternatyvūs scenarijai		
–		–

4.20 lentelė. Panaudojimo atvejo „*Įvesti temą*“ specifikacija

Panaudojimo atvejis „*Įvesti temą*“		
Aktorius		Administratorius
Prieš sąlyga		Vartotojas prisijungęs ir turi administratoriaus teises; tema, kuriai kuriama potemė, užregistruota
Sužadinimo sąlyga		Administratorius nori įvesti naują temą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
2. Administratorius įveda potemės pavadinimą		
3. Administratorius parenka temą		
4. Patvirtina		4.1. Sistema patikrina duomenis 4.2. Sistema išsaugo duomenis
Po sąlyga		Užregistruota nauja tema
Alternatyvūs scenarijai		
3. Jei administratorius neparenka temos		Sistema registruoja temą, bet ne potemę
4. Administratorius neteisingai įvedė duomenis		Sistema pateikia pranešimą

4.21 lentelė. Panaudojimo atvejo „*Pašalinti temą*“ specifikacija

Panaudojimo atvejis „*Pašalinti temą*“		
Aktorius		Administratorius
Prieš sąlyga		Vartotojas prisijungęs ir turi administratoriaus teises; šalinama tema nepriskirta straipsniui
Sužadinimo sąlyga		Administratorius nori pašalinti temą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Administratorius prašo rodyti temų sąrašą		1.1. Sistema pateikia temų sąrašą
2. Administratorius paspaudžia pašalinimo mygtuką		2.1. Sistema temą pašalina
Po sąlyga		Tema iš duomenų bazės pašalinta
Alternatyvūs scenarijai		
2. Jei tema priskirta straipsniui		Sistema suformuoja pranešimą ir temos nepašalina



4.14 pav. Administratoriaus panaudojimo atvejų realizavimo sekų diagrama

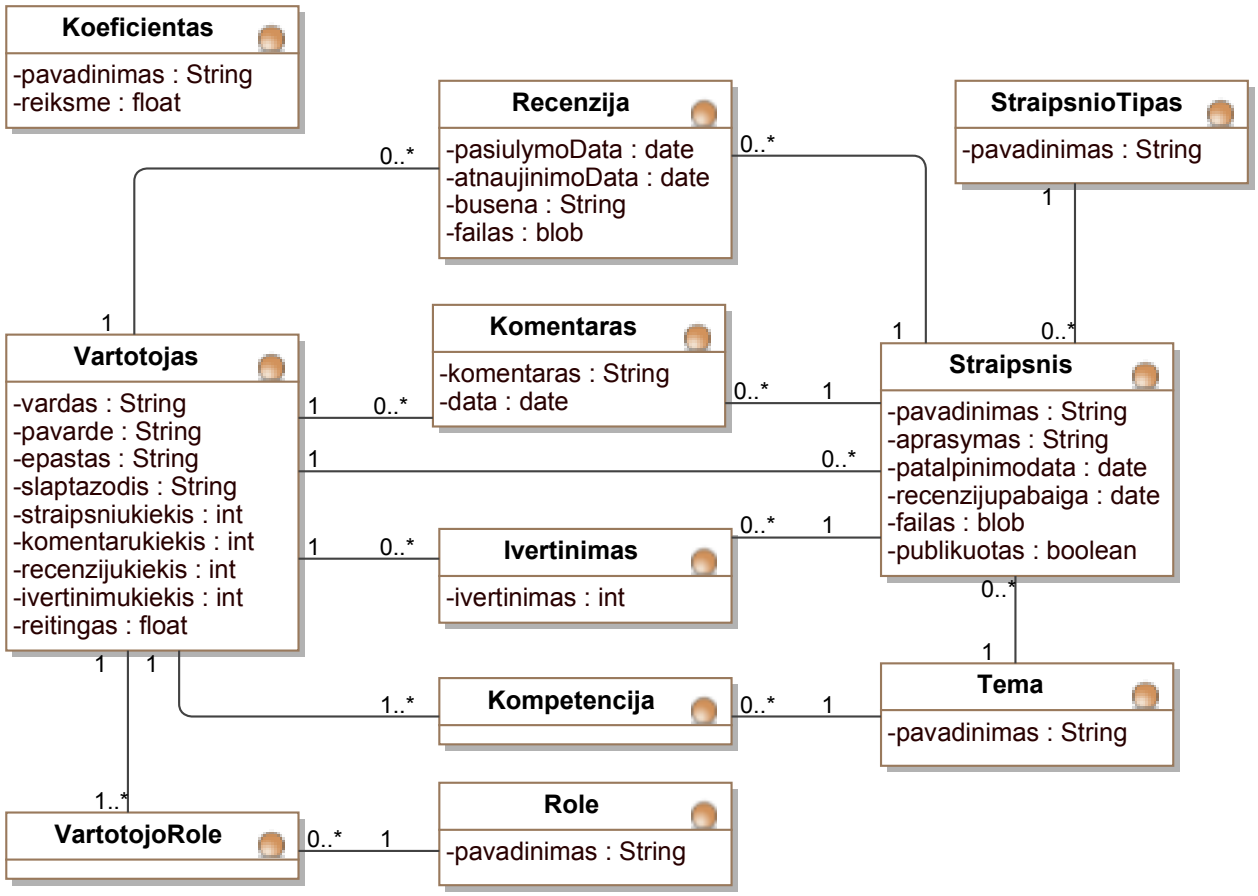
4.6. Dalykinės srities modelis

Dalykinės srities modelis pateiktas 4.15 paveikslėlyje. Šis modelis vaizduoja kuriamai sistemai aktualias esybes bei jų tarpusavio ryšius ir vėliau naudojamas duomenų bazės schemos sukūrimui.

Kiekvienos esybės paskirtis, t.y. kokią informaciją ji saugo, pateikta 4.22 lentelėje.

4.22 lentelė. Esybių paskirtis

Esybė	Saugoma informacija
Įvertinimas	Vartotojo balai skirti straipsniui įvertinti
Koeficientas	Rangavimo algoritmo koeficiento reikšmė
Komentaras	Vartotojo parašytas straipsnio komentaras
Kompetencija	Temos, kurių recenzentu gali būti vartotojas
Recenzija	Vartotojo parašyta straipsnio recenzija
Rolė	Galima sistemos vartotojo rolė
Straipsnio tipas	Straipsnio tipas
Straipsnis	Vartotojo publikuotas straipsnis
Tema	Straipsnio tema
Vartotojas	Sistemos vartotojos
Vartotojo rolė	Sistemos vartotojo rolė



4.15 pav. Dalykinės srities esybių klasių diagrama

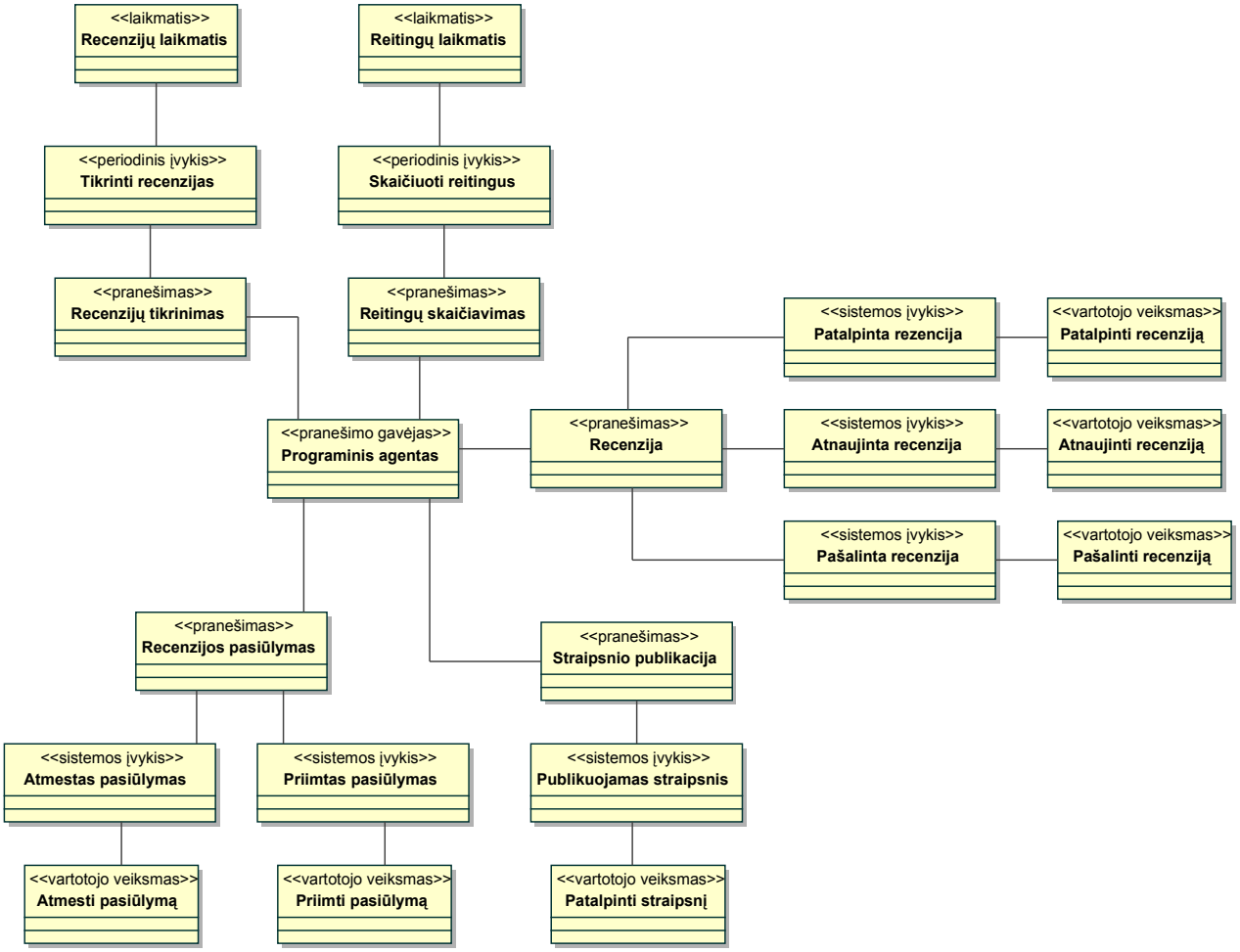
4.7. Pradinis įvykių modelis

4.16 paveikslėlyje pateiktas įvykių modelis aprašantis visus publikacijų portalo prototipo įvykius. Šis modelis sudarytas remiantis anksčiau pateiktu įvykių metamodeliu.

Įvykių modelyje yra aprašyti sistemos ir periodiniai įvykiai. Sistemos įvykius sukelia vartotojo atliekami veiksmai su sistema. Periodiniai įvykiai yra laiko įvykiai sugeneruojami tam tikrais laiko momentais.

Įvykus tam tikram įvykiui yra sukuriami pranešimai, kuriuos apdoroja įvykių gavėjai. Publikacijų portalo prototipe visus įvykius apdoroja vienintelis pranešimo gavėjas programinis agentas.

Kai recenzentas priima arba atmeta recenzijos rašymo pasiūlymą yra siunčiamas pranešimas „*Recenzijos pasiūlymas*“. Pranešimas „*Straipsnio publikacija*“ yra siunčiamas tuomet, kai vartotojas publikuoja naują straipsnį portale. Recenzentui atliekant administravimo darbus su recenzijomis yra siunčiamas pranešimas „*Recenzija*“. Pranešimai „*Reitingų skaičiavimas“* ir „*Recenzijų tikrinimas*“ yra siunčiami kai sugeneruojami laiko įvykiai.



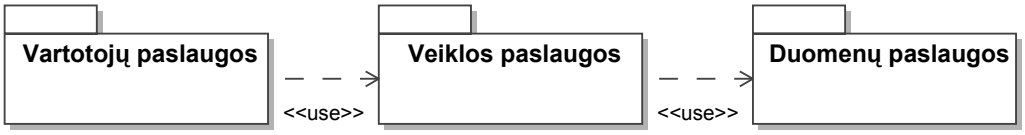
4.16 pav. Pradinis publikacijų portalo prototipo įvykių modelis

5. Publikacijų portalo prototipo projektas

5.1. Publikacijų portalo prototipo architektūra

5.1.1. Loginė visos sistemos architektūra

Publikacijų portalo prototipo loginė architektūra pateikta 5.1 paveikslėlyje. Naudojama trijų lygių architektūra atskiriant vartotojo, veiklos ir duomenų paslaugas. Vartotojo paslaugų pakete realizuojama sistemos vartotojo sąsaja, veiklos paslaugų pakete – veiklos logika. Duomenų paslaugos atspindi duomenis, kuriuos naudoja valdymo klasės.



5.1 pav. Loginė sistemos architektūra

5.1.2. Vartotojo paslaugos

Vartotojo sąsajos projektas pavaizduotas 5.2 paveikslėlyje.

Vartotojui pateikiamas pagrindinis langas, kuriame yra temų ir straipsnių sąrašai. Į kiekvieną kitą langą galima patekti pasirinkus nuorodas iš pagrindinio lango. Pagrindiniame lange yra temų ir straipsnių sąrašai. Pasirinkus temą, pateikiamas atitinkamas straipsnių sąrašas.

Prie kiekvieno straipsnio yra nuoroda į straipsnio komentarų langą.

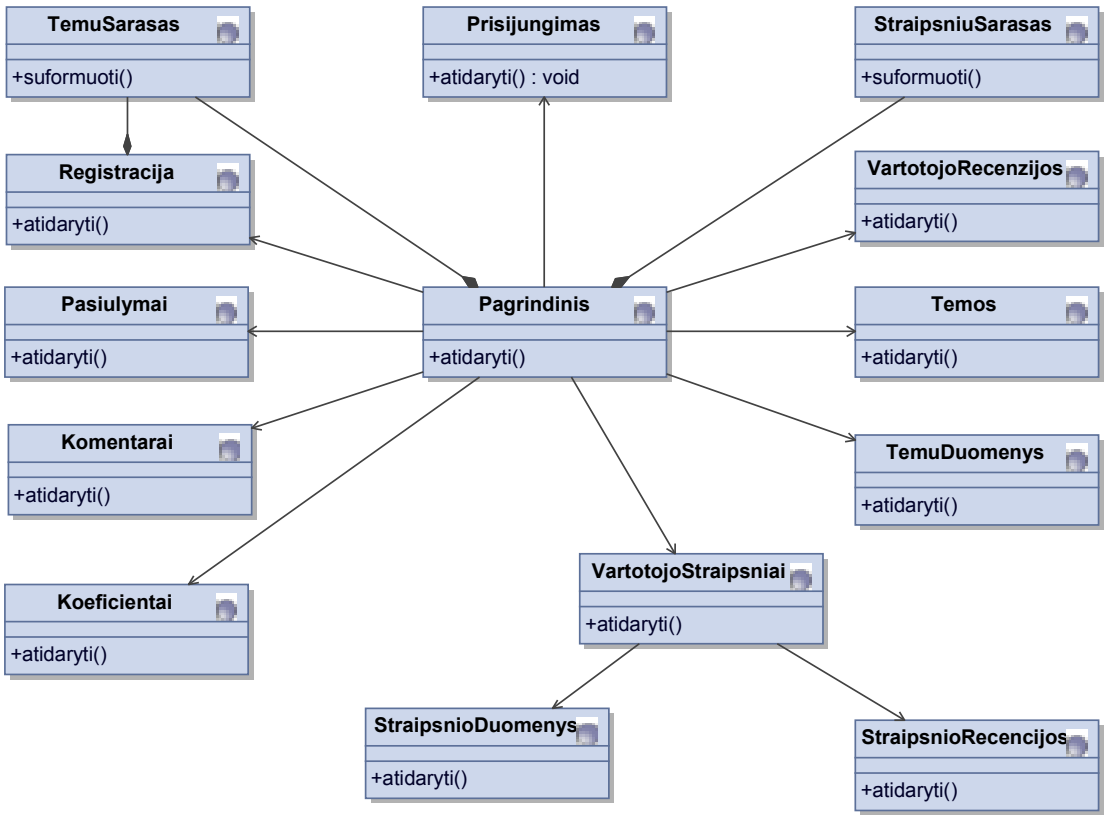
Norint naudotis visomis portalo teikiamomis funkcijomis, vartotojas turi prisijungti. Į prisijungimo langą patenkama iš pagrindinio lango.

Registruotas vartotojas gali atverti savo straipsnių ir recenzijų langus, ir atlikti jų administravimą. Iš vartotojo straipsnių lango galima patekti į straipsnio duomenų langą, kuriame galima įvesti ir koreguoti straipsnius, arba į straipsnio recenzijų langą, kuriame pateikiamas vartotojo straipsnio recenzijos.

Pasiūlymų lange registruotas vartotojas gali atmesti arba priimti programinio agento pateiktus pasiūlymus rašyti recenzijas.

Naujų vartotojų registracija atliekama registracijos lange.

Sistemos administratorius temas gali administruoti temų lange. Nauja tema sukuriama arba koreguojama temos duomenų lange. Rangavimo algoritmo koeficientai įvedami koeficientų lange.



5.2 pav. Vartotojo sąsajos modelis

5.1.3. Veiklos paslaugos

Veiklos paslaugų klasės realizuoja veiklos logiką ir yra tarpinė dalis tarp vartotojo sąsajos ir duomenų paslaugų. Valdymo klasių modelis pateiktas 5.3 paveikslėlyje. Valdymo klasės yra nepriklausomos viena nuo kitos ir vadinamos valdikliais.

Valdymo modelyje programinis agentas atlieka automatizuotas sistemos funkcijas, kurios įvardintos sudarant panaudojimo atvejų modelius (4.3 poskyris). Ši klasė naudoja vartotojo, straipsnių ir recenzijų valdiklių teikiamas paslaugas.

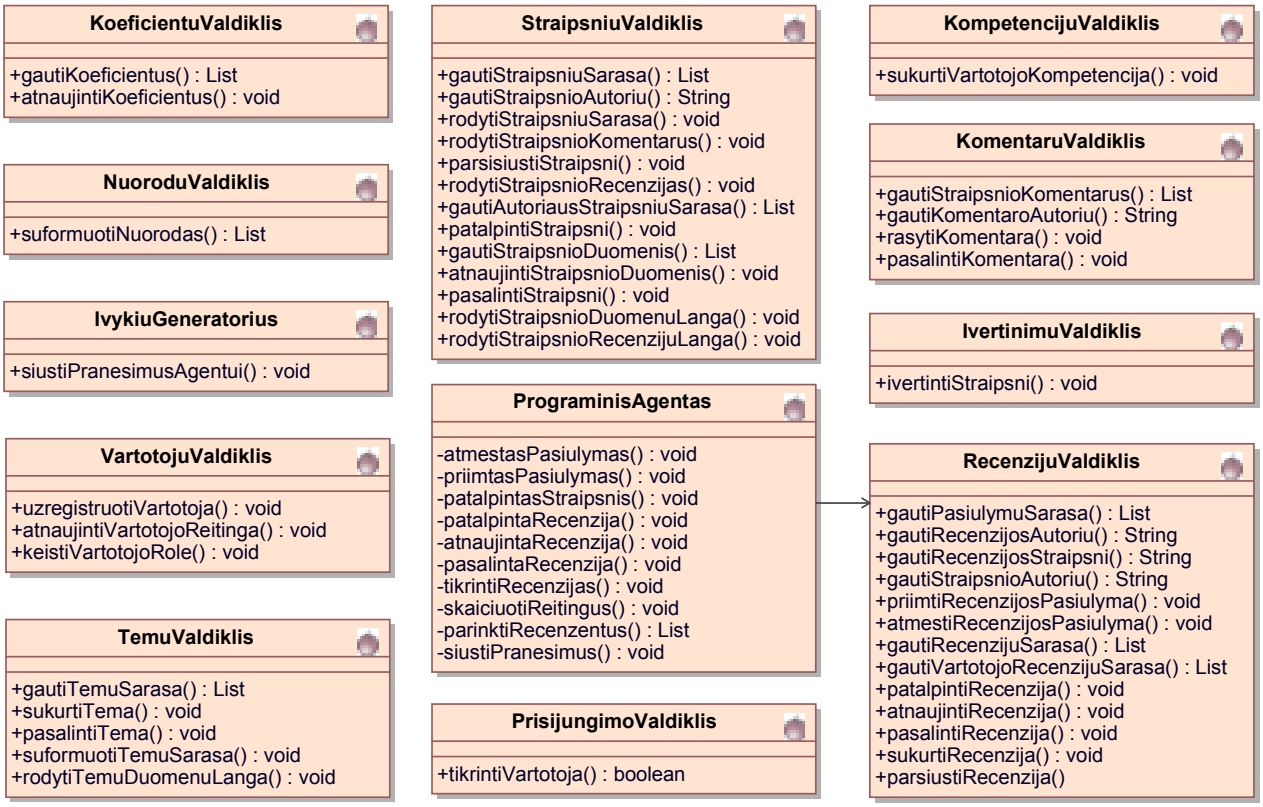
Įvykių generatorius – tai klasė, kuri atitinkamais laiko momentais programiniam agentui generuoja pranešimus.

Kiekvieno valdiklio paskirtis pateikta žemiau lentelėje.

5.1 lentelė. Valdiklių paskirtis

Valdiklis	Paskirtis
Įvertinimų valdiklis	Sukurti straipsnio įvertinimą
Įvykių generatorius	Automatiškai sukurti pranešimus agentui
Koeficientų valdiklis	Administruoti rangavimo algoritmo koeficientus
Komentarų valdiklis	Valdyti straipsnio komentarus
Kompetencijų valdiklis	Sukurti vartotojo kompetencijos sritį: temas, kurioms jis gali rašyti recenzijas
Nuorodų valdiklis	Pagal vartotojų tipą sukurti nuorodas pagrindiniame lange
Prisijungimo valdiklis	Valdyti vartotojo prisijungimą prie žinių portalo

Programinis agentas	Realizuoti automatizuotas sistemos funkcijas
Recenzijų valdiklis	Valdyti straipsnio recenzijas
Straipsnių valdiklis	Valdyti portale patalpintus straipsnius
Temų valdiklis	Administruoti temas, sudaryti temų sąrašus
Vartotojų valdiklis	Valdyti portalo vartotojus

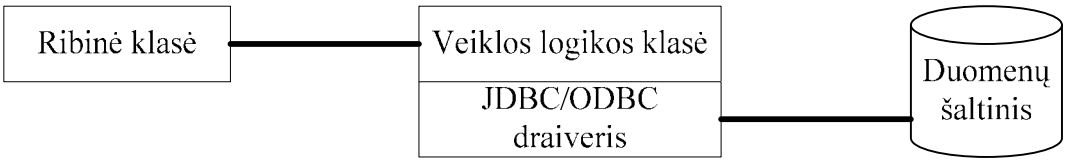


5.3 pav. Valdymo klasių modelis

5.1.4. Duomenų paslaugos

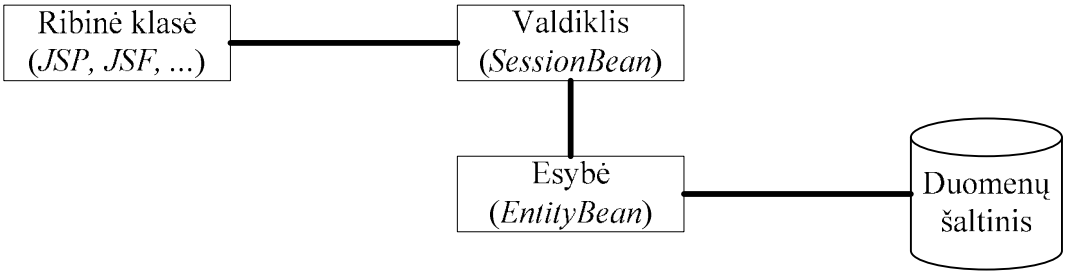
Publikacijų portalo prototipo duomenys turi būti saugomi duomenų bazėje. Veiklos logiką realizuojančios klasės kreipiasi į duomenis, esančius duomenų bazėje.

Priėjimo prie duomenų metodas priklauso nuo pasirinktos sistemos realizavimo platformos. Universalus metodas prieiti prie duomenų, saugomų duomenų bazėje, yra naudoti specialias programavimo sąsajas (angl. *Application Programming Interface*), pateikiamas kartu su realizavimo platformomis, pavyzdžiui, ODBC (*Open Database Connectivity*) ar JDBC (*Java Database Connectivity*). Naudojantis šiomis priemonėmis, duomenys pasiekiami ir atnaujinami naudojant standartinę užklausų kalbą SQL. 5.4 paveikslėlyje pateikta schema vaizduojanti, kaip naudojantis specialiomis programavimo sąsajomis pasiekti duomenis, esančius duomenų bazėje.



5.4 pav. Prieiga prie duomenų, naudojant ODBC ar JDBC programavimo sąsajas

Portalo prototipą realizuojant Java EE 5.0 technologija, priėjimą prie duomenų galima organizuoti naudojant *EntityBean* komponentus. Tokiu atveju, valdikliai kreipiasi ne tiesiogiai į duomenų bazę, bet į *EntityBean* klasės egzempliorius, kurie atspindi įrašą duomenų bazėje. *EntityBean* komponentų panaudojimas priėjimui prie duomenų bazės pavaizduotas 5.5 paveikslėlyje. Šio metodo realizavimas pasirinktas publikacijų portalo prieigai prie duomenų organizuoti.



5.5 pav. Prieiga prie duomenų, naudojant *EntityBean* komponentus

5.2. Detalus projektas

Šiame poskyryje pateiktos modelio diagramos atspindi vartotojo ir veiklos paslaugų klasių realizavimą pasirinkta Java EE 5.0 technologija. Vartotojo sąsaja realizuojama *JSP* puslapiais, o veiklos logikos klasės *EJB* komponentais. Taip pat šiame poskyryje aprašomas programinio agento ir įvykių generavimo realizavimas.

5.2.1. Klasių diagramos

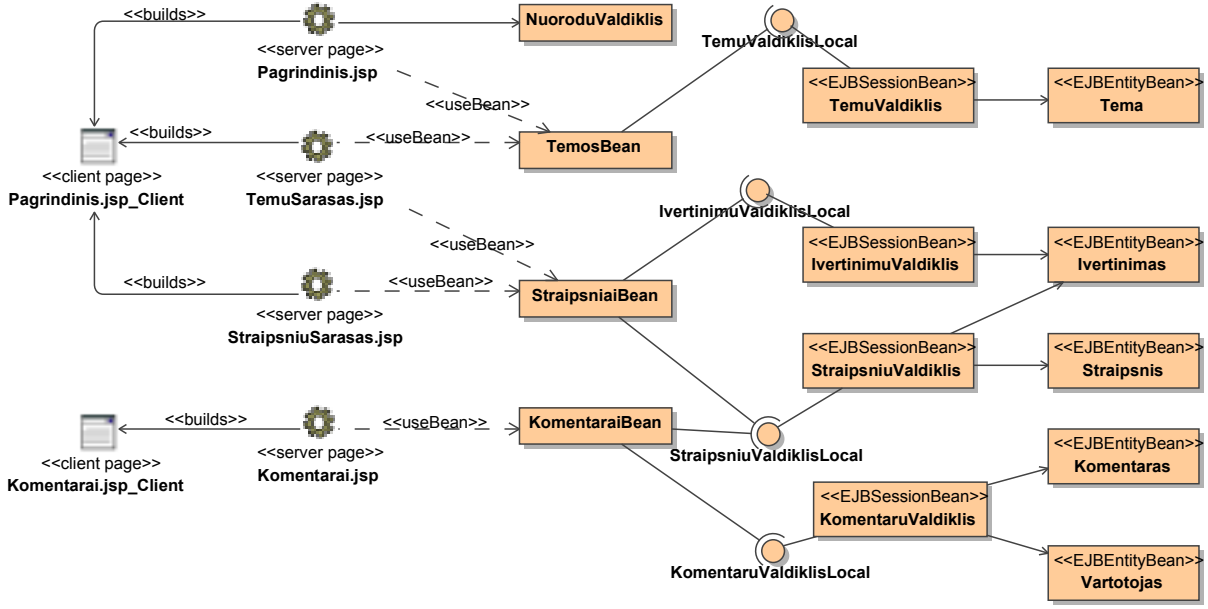
Atlikus sistemos realizavimo technologijų analizę, nuspręsta sistemą realizuoti naudojantis Java EE 5.0 technologijomis, nes šios technologijos teikiamomis galimybėmis galima automatiškai sugeneruoti įvykius, kuriuos turi apdoroti programiniai agentai, taip pat fiksuoti sistemos vartotojo įvykius ir juos atitinkamai apdoroti.

Šiame skyrelyje pateikiamos diagramos vaizduoja vartotojo sąsajos ir veiklos logikos klases, kurios realizuojamos Java EE 5.0 komponentais.

Vartotojo sąsaja realizuojama *JSP* puslapiais, o veiklos logikos klasės *EJB* komponentais. Sąsajai tarp *JSP* puslapių ir *EJB* komponentų naudojamos Java klasės, kurių metodus tiesiogiai gali kviesti *JSP* puslapiai, o pastarosios klasės per interfeisus iškviečia EJB komponentų metodus.

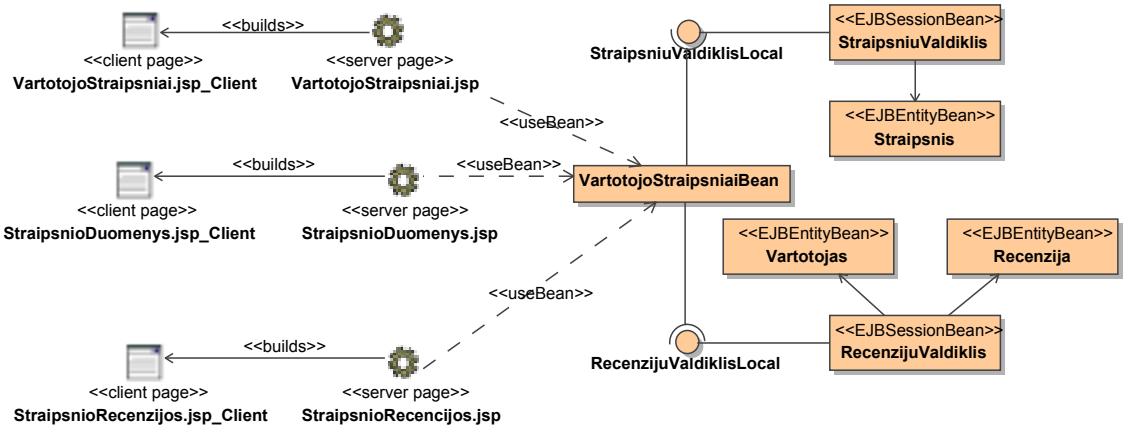
Sąsaja tarp *EJBSessionBean* komponentų ir duomenų bazės realizuojama naudojant *EJBEntityBean* komponentus.

Straipsnių skaitymą ir vertinimą, bei komentarų rašymą ir skaitymą realizuojančios klasės pateiktos 5.6 paveikslėlyje.

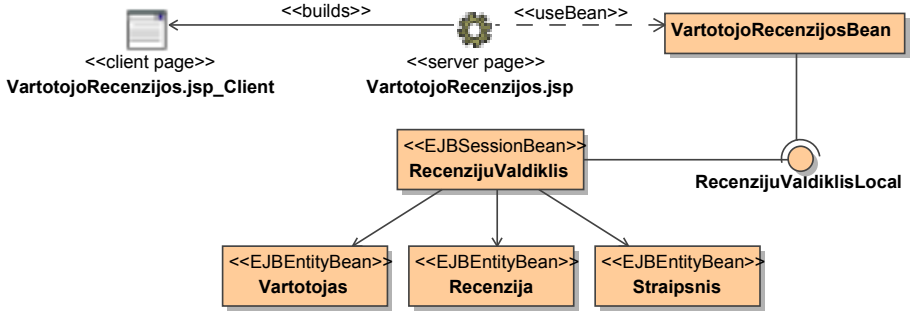


5.6 pav. Klasės, realizuojančios straipsnių, recenzijų ir komentarų peržiūrą

Straipsnių administravimą realizuojančios klasės pateiktos 5.7 paveikslėlyje, o recenzijų administravimą – 5.8 paveikslėlyje.

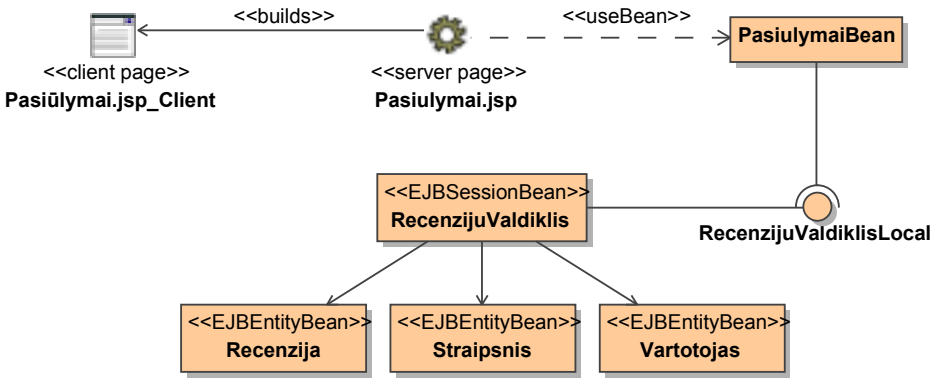


5.7 pav. Straipsnių administravimą realizuojančios klasės



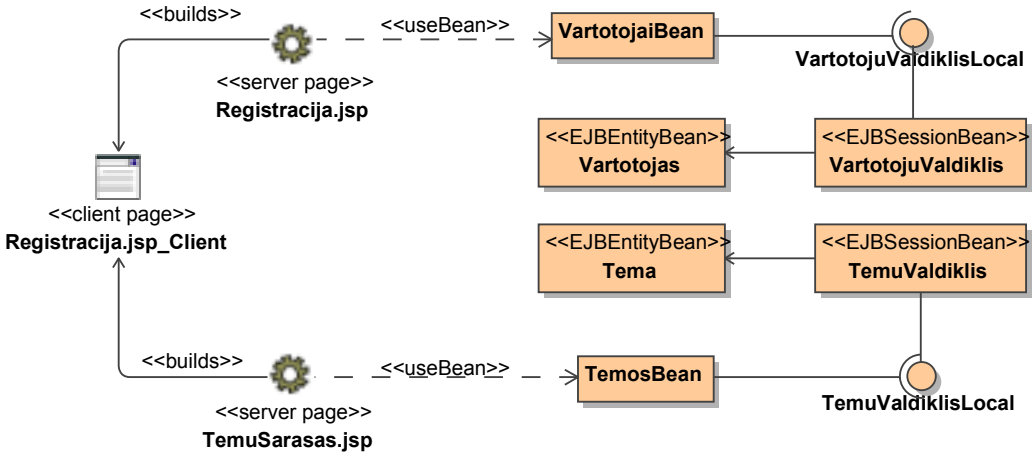
5.8 pav. Recenzijų administravimą realizuojančios klasės

Recenzijų rašymo pasiūlymų priėmimą ir atmetimą realizuojančios klasės pateiktos 5.9 paveikslėlyje.



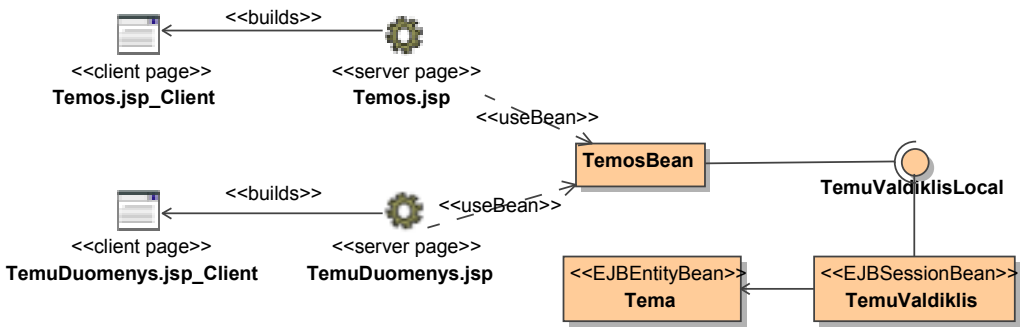
5.9 pav. Pasiūlymo priėmimą ir atmetimą realizuojančios klasės

Klasės, realizuojančios vartotojų registraciją, pavaizduotos klasių diagramoje 5.10 paveikslėlyje.

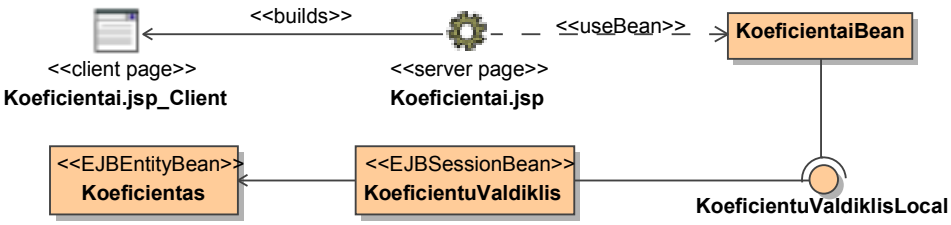


5.10 pav. Vartotojo registraciją realizuojančios klasės

Sistemos administratorius gali sukurti arba pašalinti temas, taip pat įvesti vartotojų rangavimo algoritmo koeficientus. Temų administravimą realizuojančios klasės pateiktos 5.11 paveikslėlyje, o koeficientų – 5.12 paveikslėlyje.



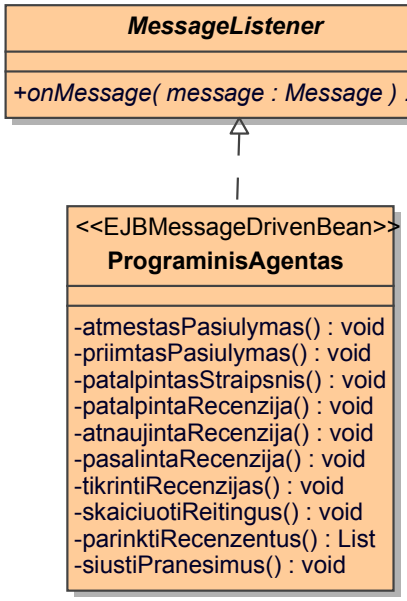
5.11 pav. Temų administravimą realizuojančios klasės



5.12 pav. Rangavimo algoritmo koeficientų administravimą realizuojančios klasės

5.2.2. Programinio agento realizacija

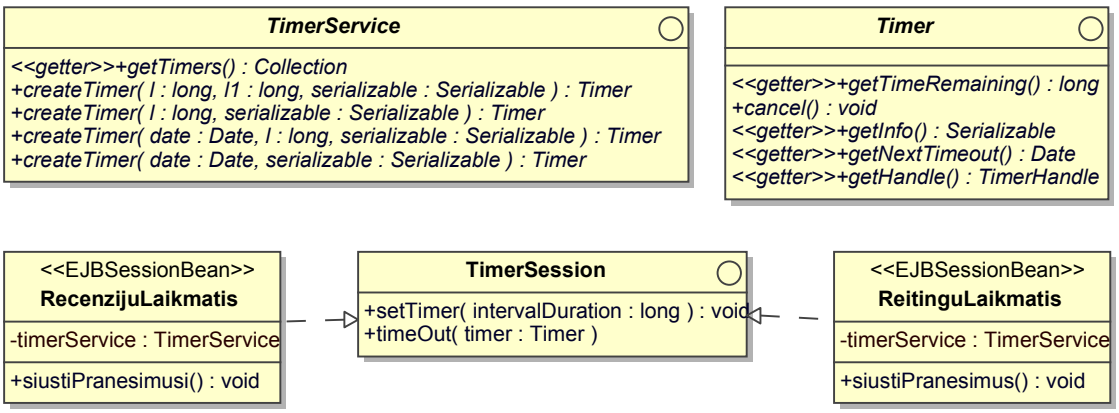
Programinis agentas realizuojamas kaip *EJBMessageDrivenBean* komponentas (5.13 pav.). Šis komponentas priima jam siunčiamus pranešimus ir pagal atitinkamą pranešimo tipą, vykdo automatizuotas sistemos funkcijas.



5.13 pav. Programinį agentą realizuojanti klasė

5.2.3. Įvykių generavimo realizavimas

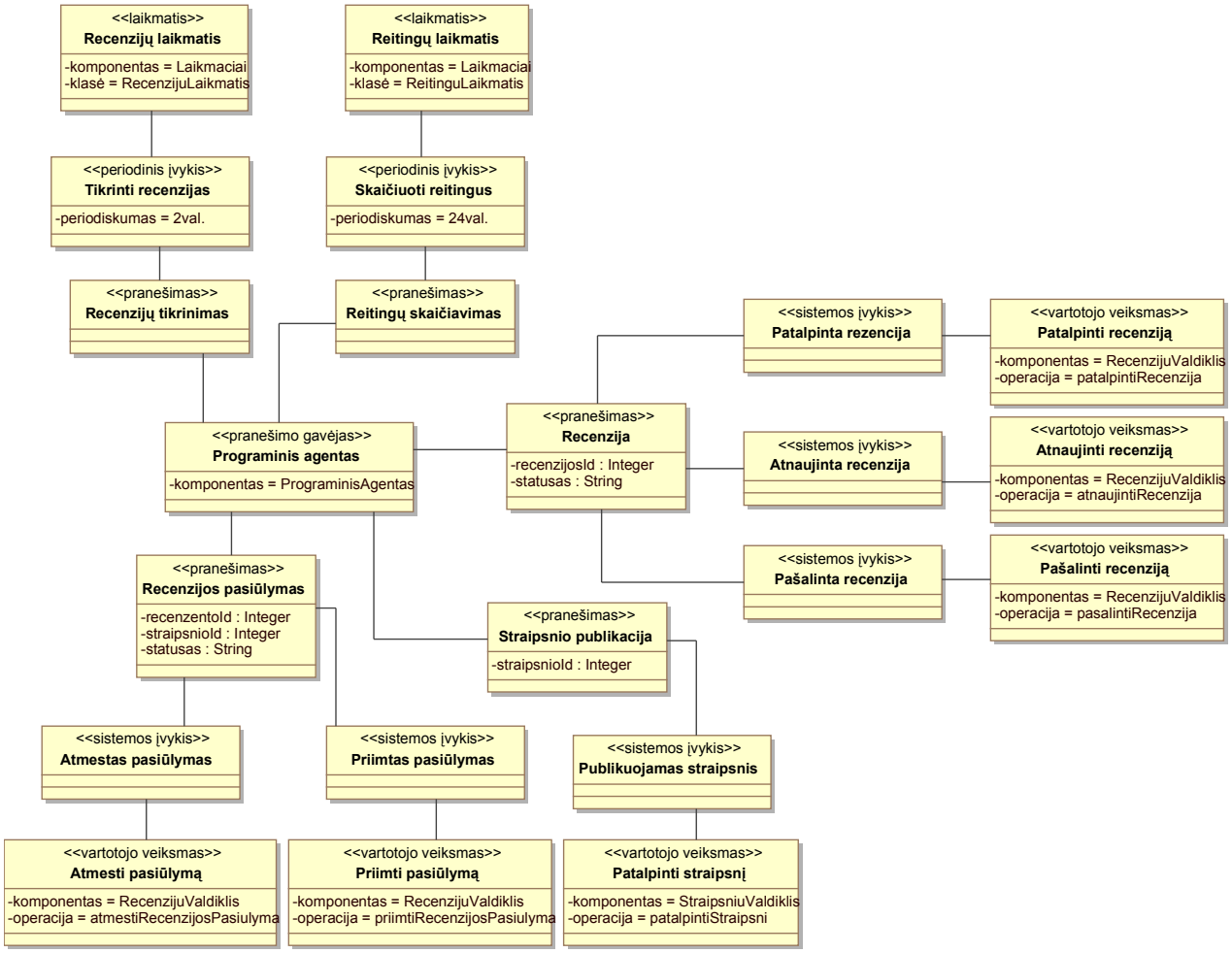
Dalį sistemos įvykių, kuriuos turi apdoroti programinis agentas, sukuria vartotojų veiksmai, kiti turi būti sugeneruoti automatiškai. Tam panaudojamos įvykių generavimo komponentai: „*RecenzijuLaikmatis*“ ir „*ReitinguLaikmatis*“ (5.14 pav.). Įvykių generatoriai, panaudojant Java EE technologiją, realizuojami klasėmis, kurios sukuria interfeiso *TimerSession* metodus. Šios klasės nustatytais laiko intervalais generuoja ir siunčia pranešimus, kuriuos apdoroja pranešimų gavėjas – programinis agentas.



5.14 pav. Laiko įvykių generavimo komponentai

5.3. Sistemos įvykių modelis

Atlikus reikalavimų analizę buvo sudarytas pradinis publikacijų portalo prototipo įvykių modelis (4.7 poskyris). Šis modelis yra papildomas sudarius sistemos klasių modelį ir žinant, kokios klasės ar komponentai realizuoja įvykių sukėlėjus, laiko įvykių generavimo laikmačius bei pranešimų gavėjus. Galutinis įvykių modelis yra pateiktas 5.15 paveikslėlyje.

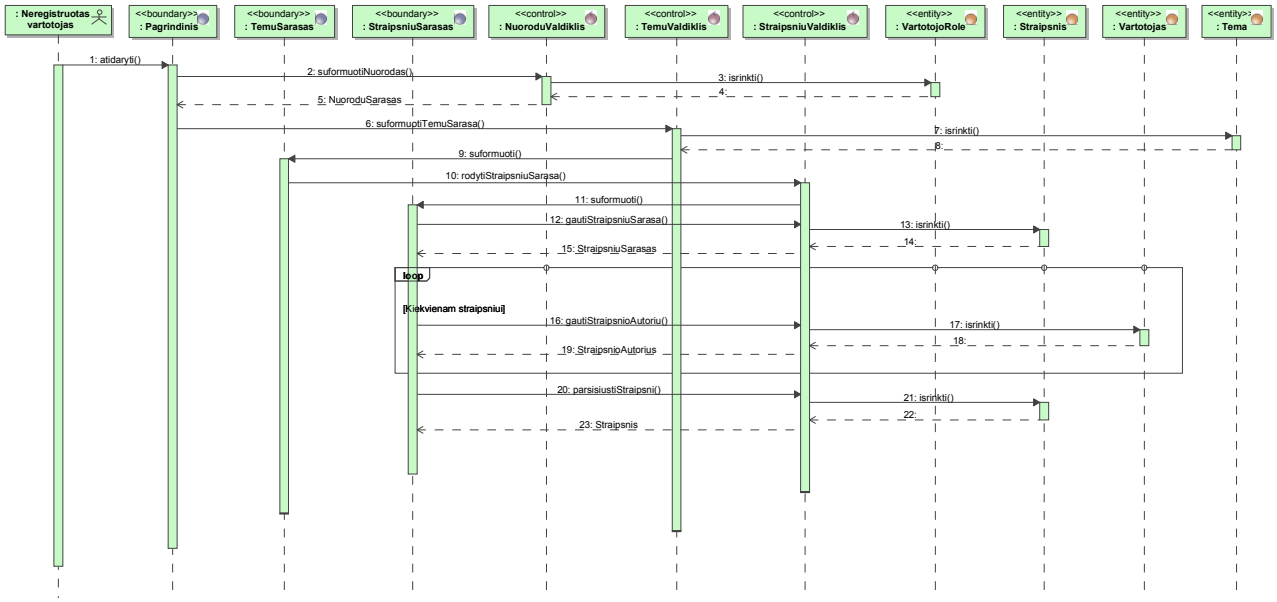


5.15 pav. Galutinis publikacijų portalo prototipo įvykių modelis

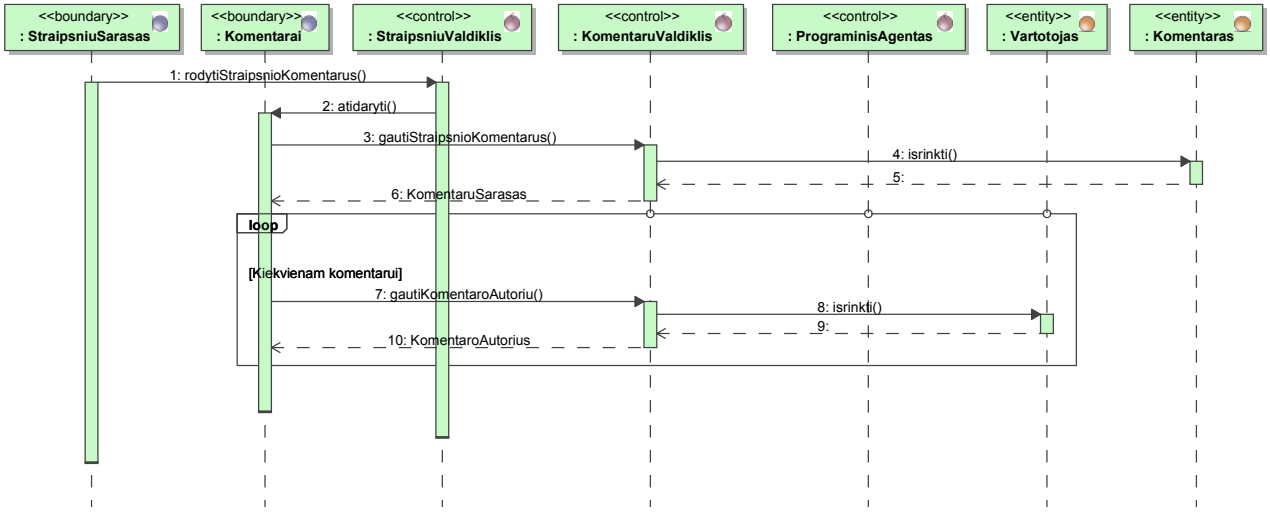
5.4. Sistemos elgsenos modelis

5.4.1. Sekų diagramos

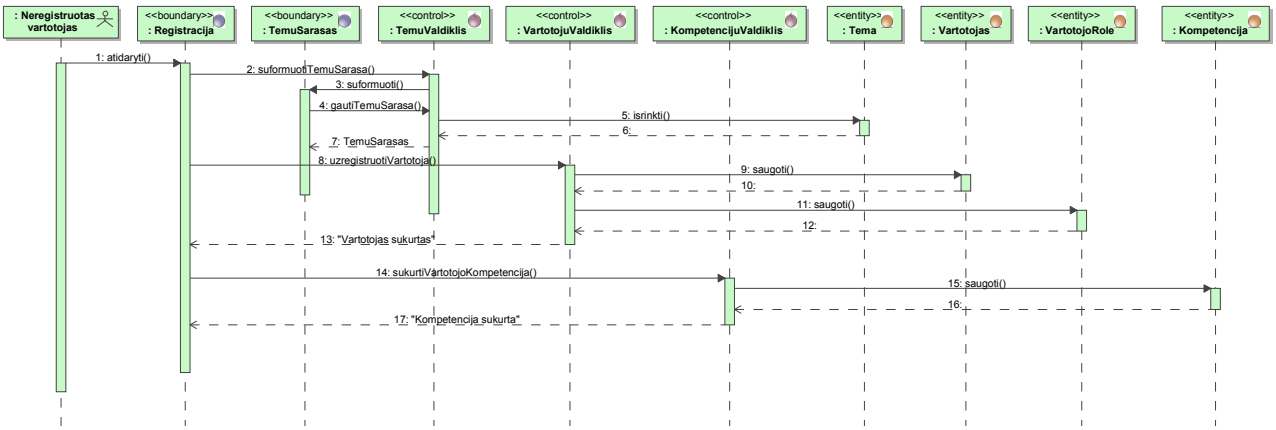
Šiame poskyryje pateikiamos sekų diagramos vaizduoja bendradarbiavimą tarp ribinių, valdymo ir esybių klasių. Pagal loginę sistemos architektūrą, bendradarbiavimas tarp vartotojo paslaugų ir esybių yra vykdomas per veiklos paslaugų klases. Šiose diagramose taip pat matyti siunčiami pranešimai, aprašyti įvykių modelyje.



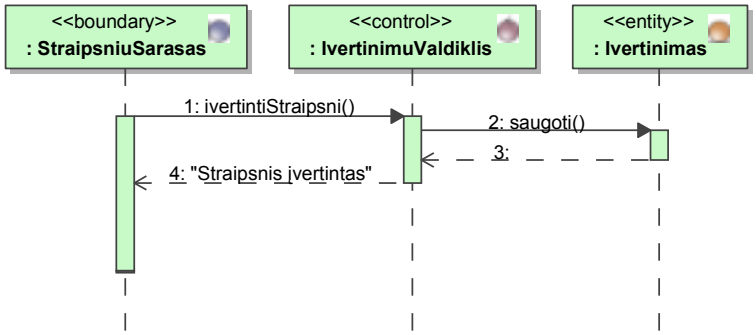
5.16 pav. Straipsnio parsisiuntimo sekų diagrama



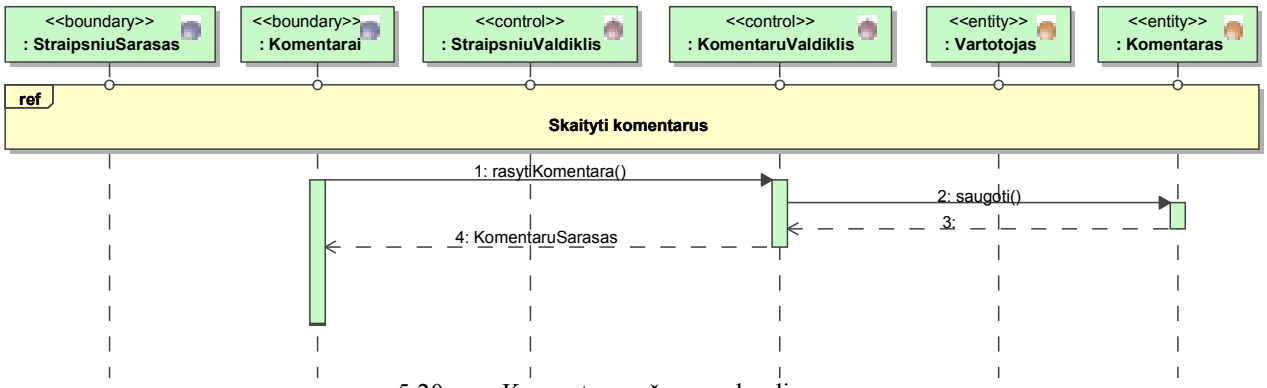
5.17 pav. Komentarų skaitymo sekų diagrama



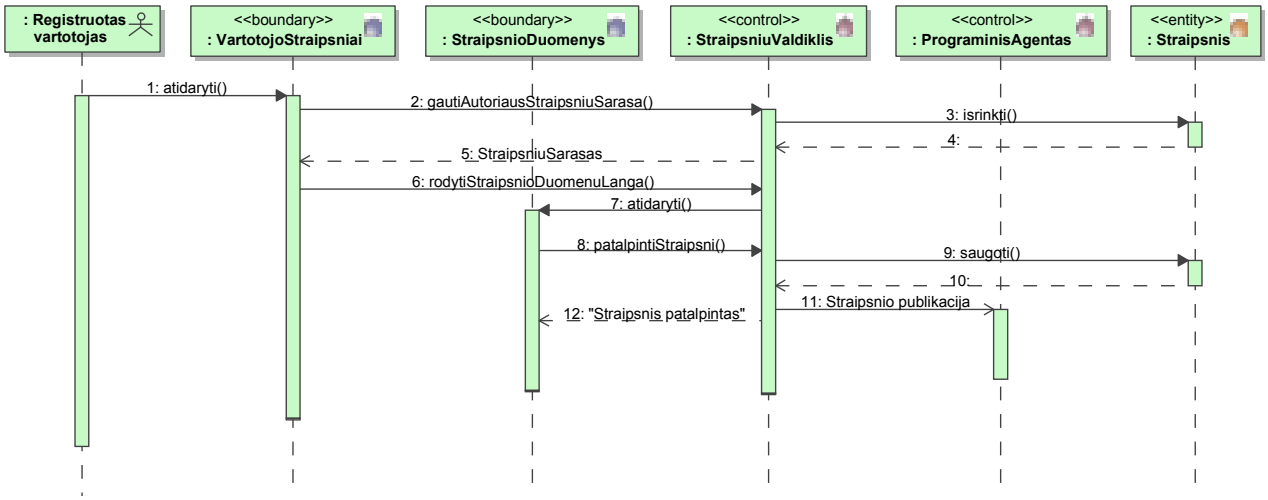
5.18 pav. Registracijos sekų diagrama



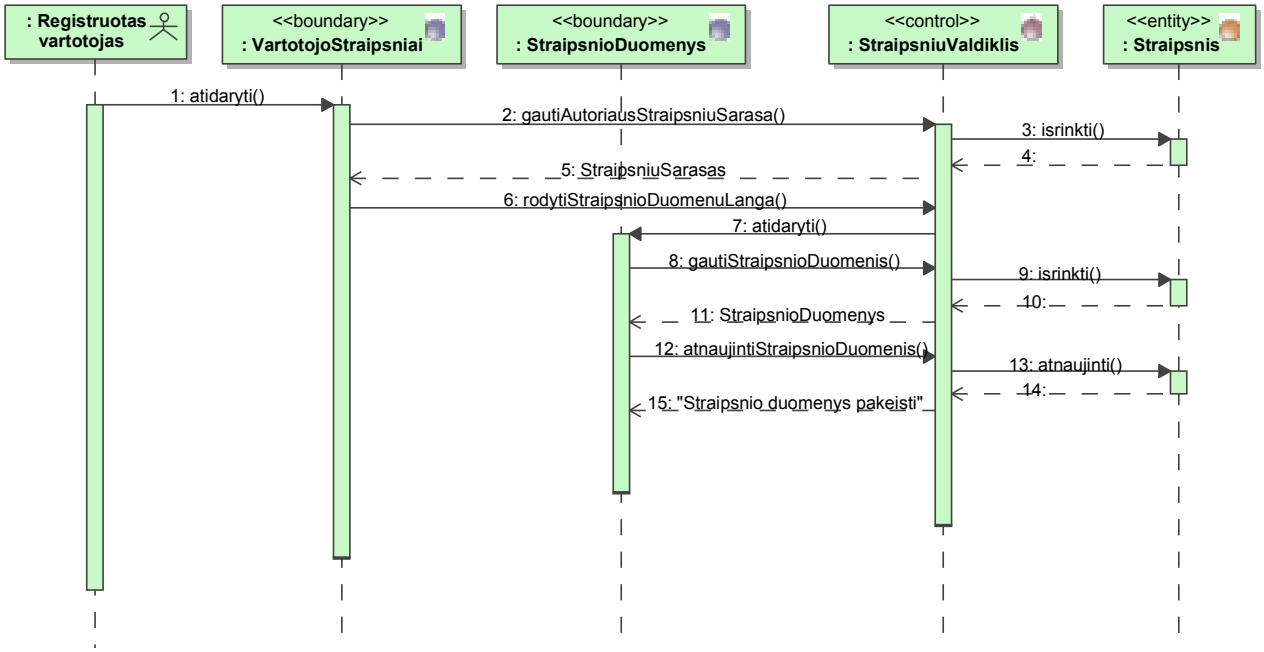
5.19 pav. Straipsnio įvertinimo sekų diagrama



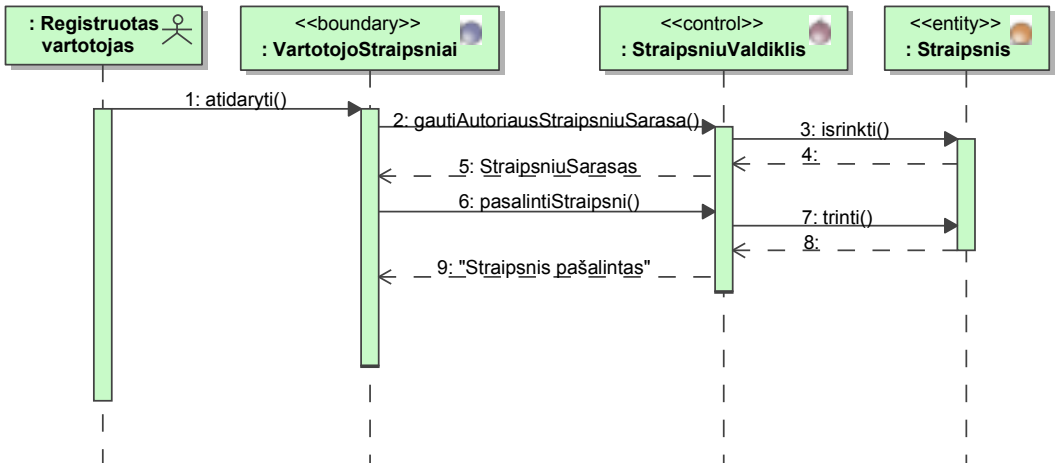
5.20 pav. Komentarų rašymo sekų diagrama



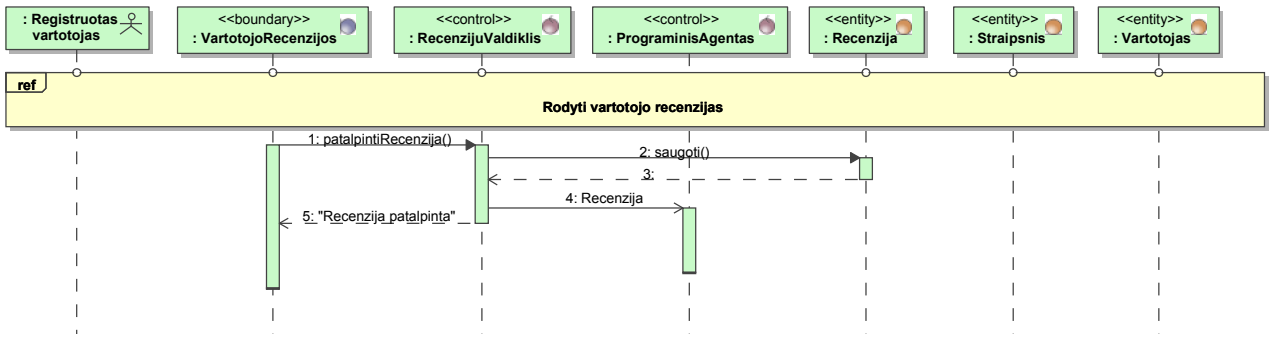
5.21 pav. Straipsnio publikavimo sekų diagrama



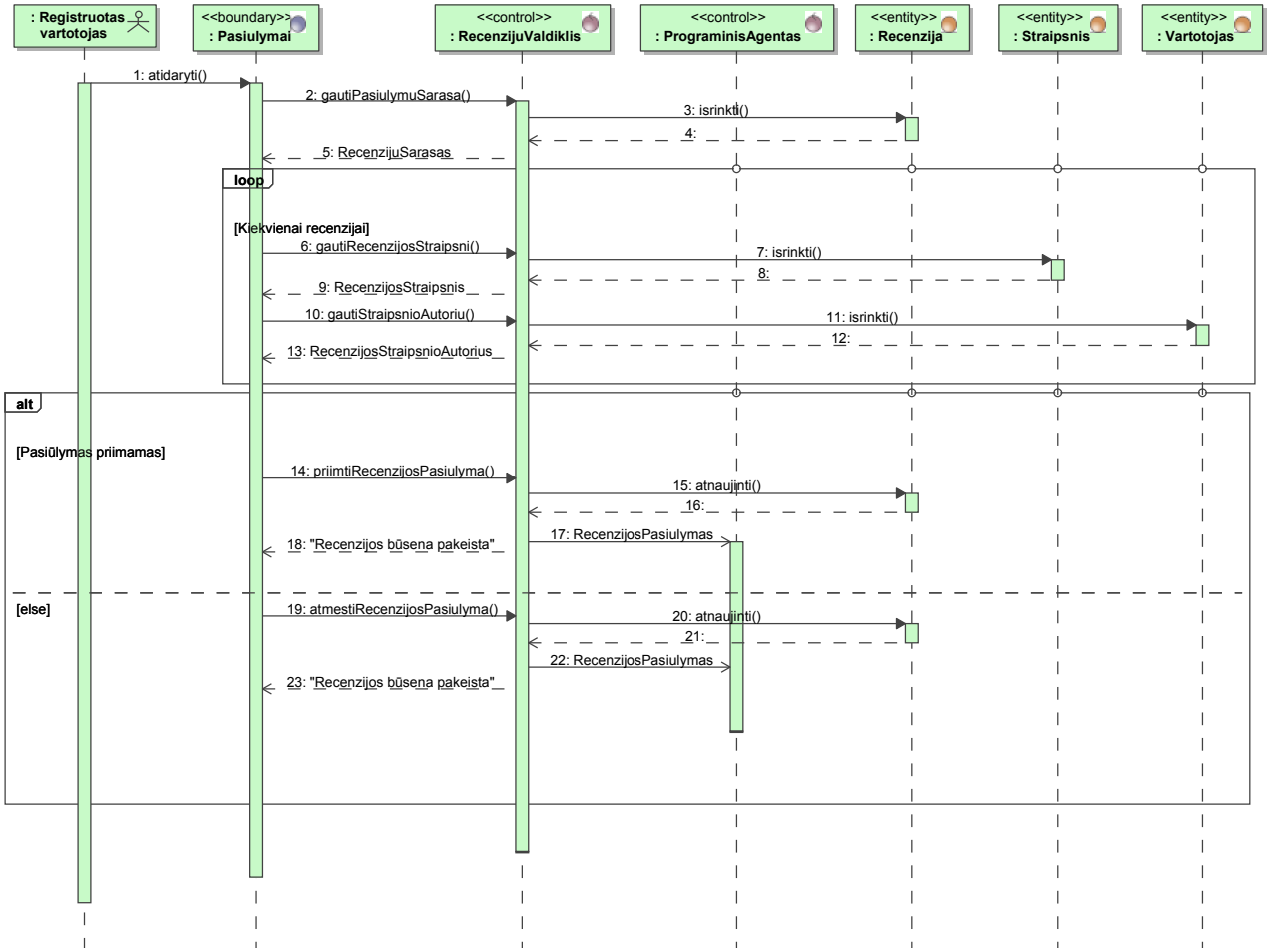
5.22 pav. Straipsnio koregavimo sekų diagrama



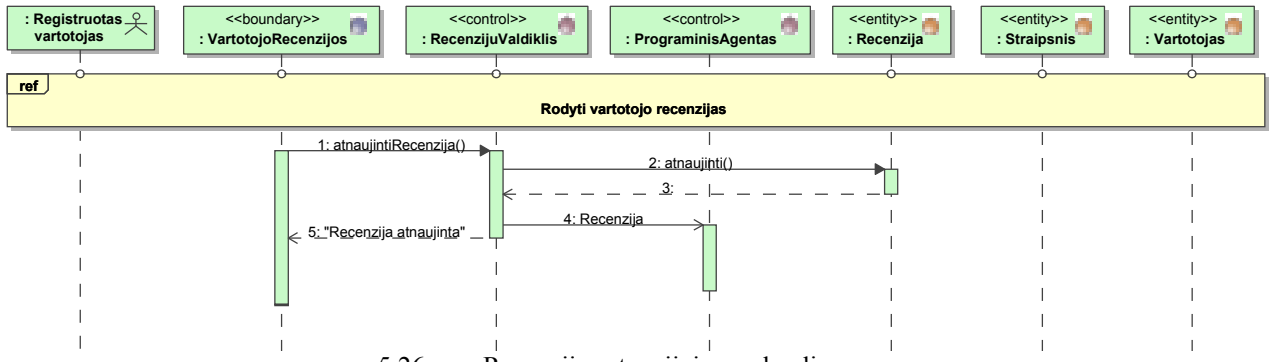
5.23 pav. Straipsnio pašalinimo sekų diagrama



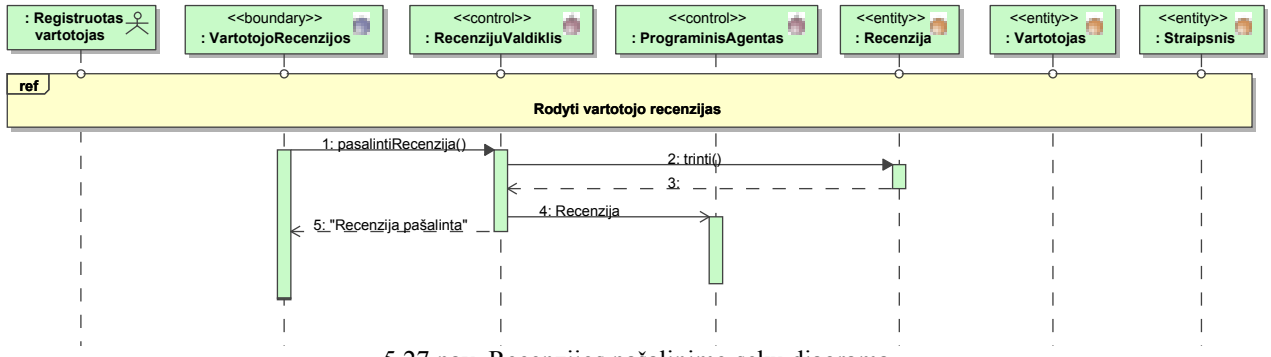
5.24 pav. Recenzijos patalpinimo sekų diagrama



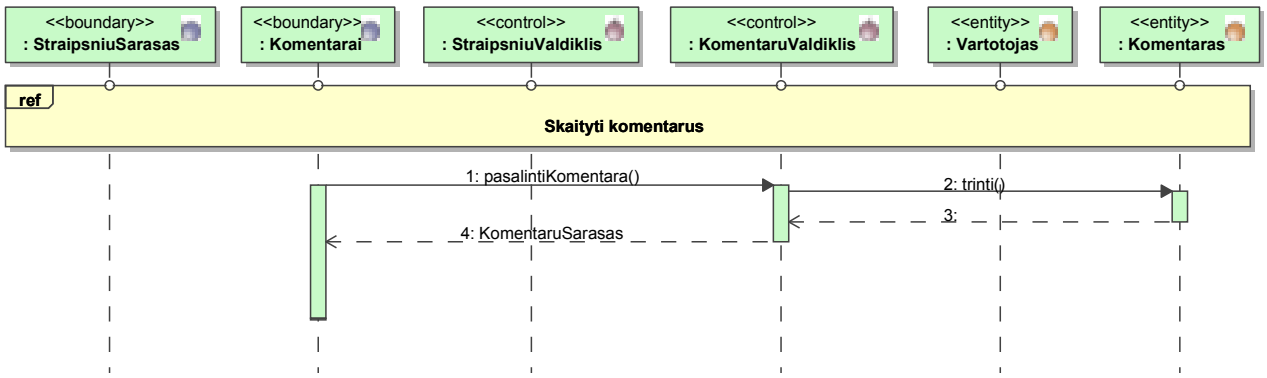
5.25 pav. Pasiūlymo rašyti recenziją priėmimo/atmetimo diagrama



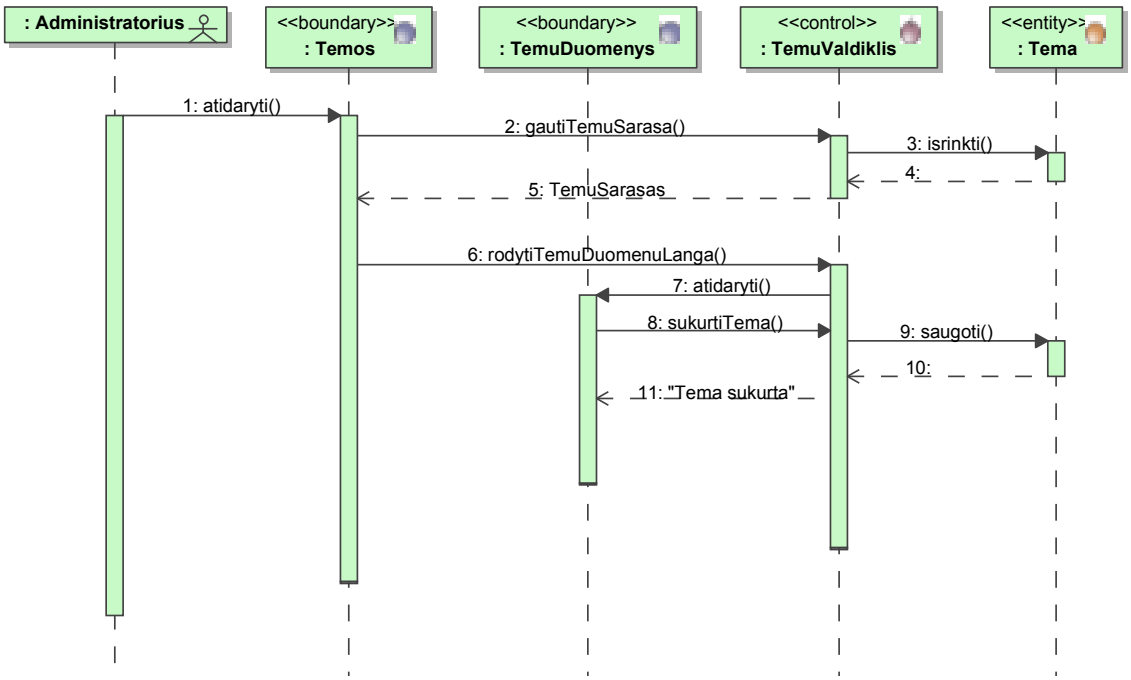
5.26 pav. Recenzijos atnaujinimo sekų diagrama



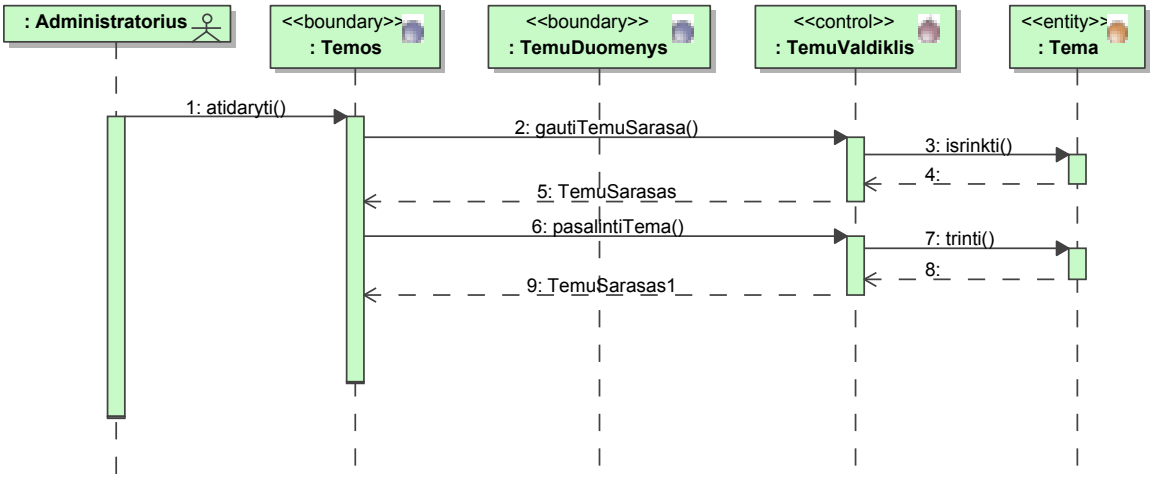
5.27 pav. Recenzijos pašalinimo sekų diagrama



5.28 pav. Komentaro pašalinimo sekų diagrama



5.29 pav. Temos įvedimo sekų diagrama

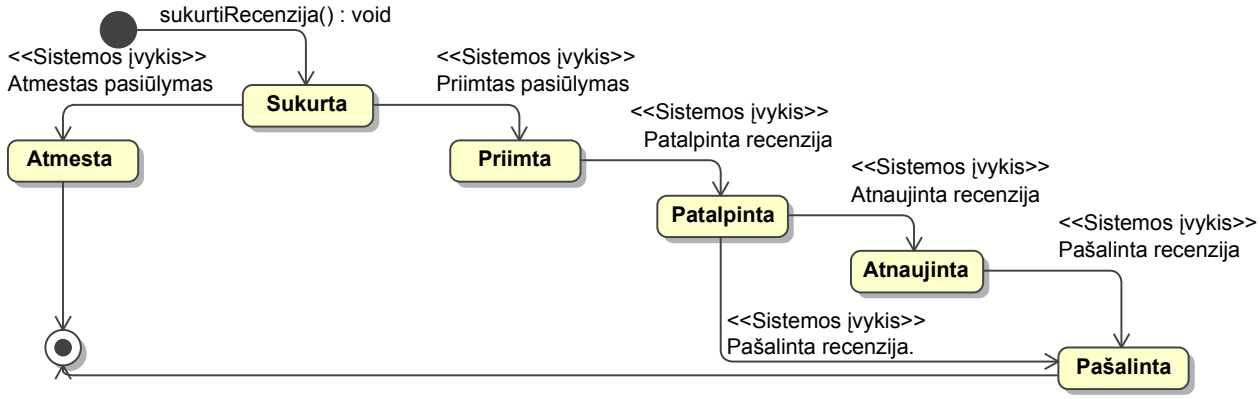


5.30 pav. Temos pašalinimo sekų diagrama

5.4.2. Būsenų diagramos

Recenzijos būsenų diagrama pavaizduota 5.31 paveikslėlyje. Programiniam agentui parinkus recenzentus yra sukuriamos recenzijos. Jei recenzija yra priimama, jos būsena tampa „priimta“, jei atmetama – „atmesta“. Kai recenzentas parašo recenziją ir patalpina portale būsena pasikeičia į „patalpintą“. Patalpinta recenzija gali būti atnaujinta arba pašalinta.

Žemiau pateiktoje būsenų diagramoje objektas pakeičia būsenas sistemoje sužadinus tam tikrus įvykius. Virš rodyklių žyminčių būsenos pasikeitimą parašyti įvykiai, kurie yra panaudoti iš įvykių modelio.



5.31 pav. Recenzijos būsenų diagrama

5.5. Vartotojų rangavimo algoritmas

Programiniai agentai nustatytais laiko momentais perskaičiuoja vartotojo reitingus ir pagal tai suteikia jiems teises. Didžiausią reitingą turintys vartotojai tampa sistemos administratoriais. Šiame poskyryje pateikiamos vartotojo reitingo skaičiavimo formulės.

Vartotojo reitingas apskaičiuojamas pagal vartotojo parašytų straipsnių kiekį ir jų aktualumą, parašytų recenzijų ir komentarų kiekį, bei straipsnių įvertinimų kiekį. Straipsnio aktualumas įvertinamas atsižvelgiant į parašytų komentarų skaičių, bei kitų vartotojų įvertinimus.

Įvedami tokie žymėjimai:

$SV_{i,j}$ – i-tojo vartotojo parašyto j-tojo straipsnio įvertinimas
Sj – j-tojo straipsnio įvertinimas
KS_j – j-tojo straipsnio komentarų skaičius
KV_i – i-tojo vartotojo parašytų komentarų skaičius
$I_{j,v}$ – j-tojo straipsnio v-tasis įvertinimas balais
R_i – i-tojo vartotojo parašytų recenzijų skaičius
V_i – i-tojo vartotojo įvertintų straipsnių skaičius
m – i-tojo vartotojo parašytų straipsnių kiekis
n – j-tojo straipsnio įvertinimų kiekis
s – straipsnių kiekis portale
t – vartotojų skaičius portale
SK – straipsnių dedamosios dalis galutiniame įvertinime

RK – Recenzijų dedamosios dalis galutiniame įvertinime
KK – Komentarų dedamosios dalis galutiniame įvertinime
IK – Įvertinimų dedamosios dalis galutiniame įvertinime

Straipsnio įvertinimas apskaičiuojamas taip:

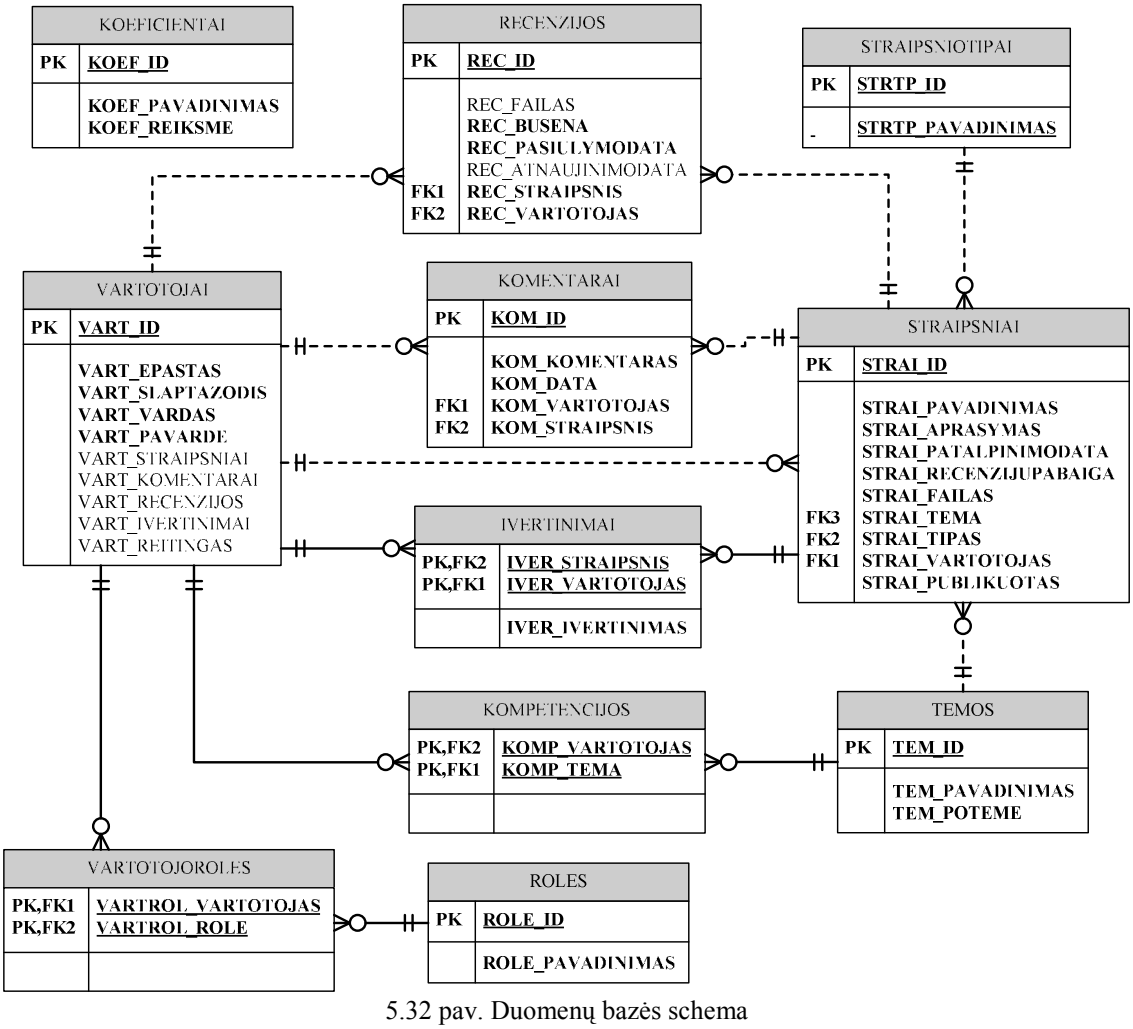
$$S_j = \frac{\sum_{v=1}^{n} I_{j,v}}{n \cdot 5} \cdot \frac{KS_j}{\max\limits_{u=1..s}\{KS_u\}}$$

Vartotojo reitingas apskaičiuojamas pagal formulę:

$$R_i = SK \cdot \frac{\sum_{j=1}^{m} \frac{SV_{i,j}}{\max\limits_{u=1..s}\{S_u\}}}{m} + RK \frac{R_i}{\max\limits_{r=1..t}\{R_r\}} + KK \cdot \frac{KV_i}{\max\limits_{k=1..t}\{KV_k\}} + IK \frac{V_i}{\max\limits_{v=1..t}\{V_v\}}$$

5.6. Duomenų bazės schema

Duomenų bazės schema pateikta 5.32 paveikslėlyje, o detalūs lentelių atributų aprašymai 5.2 lentelėje.



5.32 pav. Duomenų bazės schema

5.2 lentelė. Duomenų bazės lentelių atributai

ATRIBUTAS	TIPAS	PASKIRTIS
IVERTINIMAI		
IVER_STRAIPSNIS	INT	Pirminis raktas. Išorinis raktas. Įvertintas straipsnis
IVER_VARTOTOJAS	INT	Pirminis raktas. Išorinis raktas. Įvertinimo autorius
IVER_IVERTINIMAS	NUMERIC(2)	Įvertinimas balais
KOEFICIENTAI		
KOEF_ID	INT(10,0)	Pirminis raktas
KOEF_PAVADINIMAS	VARCHAR(50)	Koeficiento pavadinimas
KOEF_REIKSME	NUMERIC(5,2)	Koeficiento reikšmė
KOMENTARAI		
KOM_ID	INT(10,0)	Pirminis raktas
KOM_KOMENTARAS	VARCHAR(1000)	Komentaro tekstas
KOM_DATA	DATE	Komentaro parašymo data
KOM_VARTOTOJAS	INT	Išorinis raktas. Komentaro autorius
KOM_STRAIPSNIS	INT	Išorinis raktas. Komentuojamas straipsnis
KOMPETENCIJOS		
KOMP_VARTOTOJAS	INT	Pirminis raktas. Išorinis raktas. Vartotojas, kurio kompetencijos temos saugomos
KOMP_TEMA	INT	Pirminis raktas. Išorinis raktas. Vartotojo kompetencijos tema
RECENZIJOS		
REC_ID	INT	Pirminis raktas
REC_FAILAS	BLOB	Recenzijos failas
REC_BUSENA	VARCHAR(20)	Recenzijos būsena
REC_PASIULYMODATA	DATE	Recenzijos pasiūlymo recenzentui data
REC_ATNAUJINIMODATA	DATE	Recenzijos patalpinimo arba atnaujinimo data
REC_STRAIPSNIS	INT	Išorinis raktas. Straipsnis, kuriam parašyta recenzija
REC_VARTOTOJAS	INT	Išorinis raktas. Vartotojas, kuris parašė recenziją
ROLES		
ROLE_ID	INT	Pirminis raktas
ROLE_PAVADINIMAS	VARCHAR(30)	Rolės pavadinimas
STRAIPSNIAI		
STRAI_ID	INT	Pirminis raktas
STRAI_PAVADINIMAS	VARCHAR(100)	Straipsnio pavadinimas
STRAI_APRASYMAS	VARCHAR(4000)	Straipsnio aprašymas
STRAI_PATALPINIMODATA	DATE	Straipsnio patalpinimo data
STRAI_RECENZIJUPABAIGA	DATE	Data, iki kurios rašomos straipsnio recenzijos
STRAI_FAILAS	BLOB	Straipsnio failas
STRAI_TEMA	INT	Išorinis raktas. Straipsnio tema
STRAI_TIPAS	INT	Išorinis raktas. Straipsnio tipas
STRAI_VARTOTOJAS	INT	Išorinis raktas. Straipsnio autorius
STRAI_PUBLIKUOTAS	NUMERIC(1)	Parodo, ar straipsnis viešai prieinamas
STRAIPSNIOTIPAI		
STRTP_ID	INT	Pirminis raktas
STRTP_PAVADINIMAS	VARCHAR(40)	Straipsnio tipo pavadinimas

ATRIBUTAS	TIPAS	PASKIRTIS
TEMOS		
TEM_ID	INT	Pirminis raktas
TEM_PAVADINIMAS	VARCHAR(50)	Temos pavadinimas
TEM_POTEME	INT	Išorinis raktas. Parodo, kurios temos potemė
VARTOTOJAI		
VART_ID	INT	Pirminis raktas
VART_EPASTAS	VARCHAR(30)	Vartotojo elektroninis paštas
VART_SLAPTAZODIS	VARCHAR(32)	Vartotojo slaptažodis
VART_VARDAS	VARCHAR(32)	Vartotojo vardas
VART_PAVARDE	VARCHAR(32)	Vartotojo pavardė
VART_STRAIPSNIAI	NUMERIC(5,0)	Vartotojo parašytų straipsnių kiekis
VART_KOMENTARAI	NUMERIC (5,0)	Vartotojo parašytų komentarų kiekis
VART_RECENZIJOS	NUMERIC (5,0)	Vartotojo parašytų recenzijų kiekis
VART_IVERTINIMAI	NUMERIC (5,0)	Vartotojo įvertintų straipsnių kiekis
VART_REITINGAS	NUMERIC (5,2)	Vartotojo reitingas
VARTOTOJOROLES		
VARTROL_VARTOTOJAS	INT	Pirminis raktas. Išorinis raktas. Vartotojas, kurio rolė saugoma
VARTROL_ROLE	INT	Pirminis raktas. Išorinis raktas. Vartotojo rolė

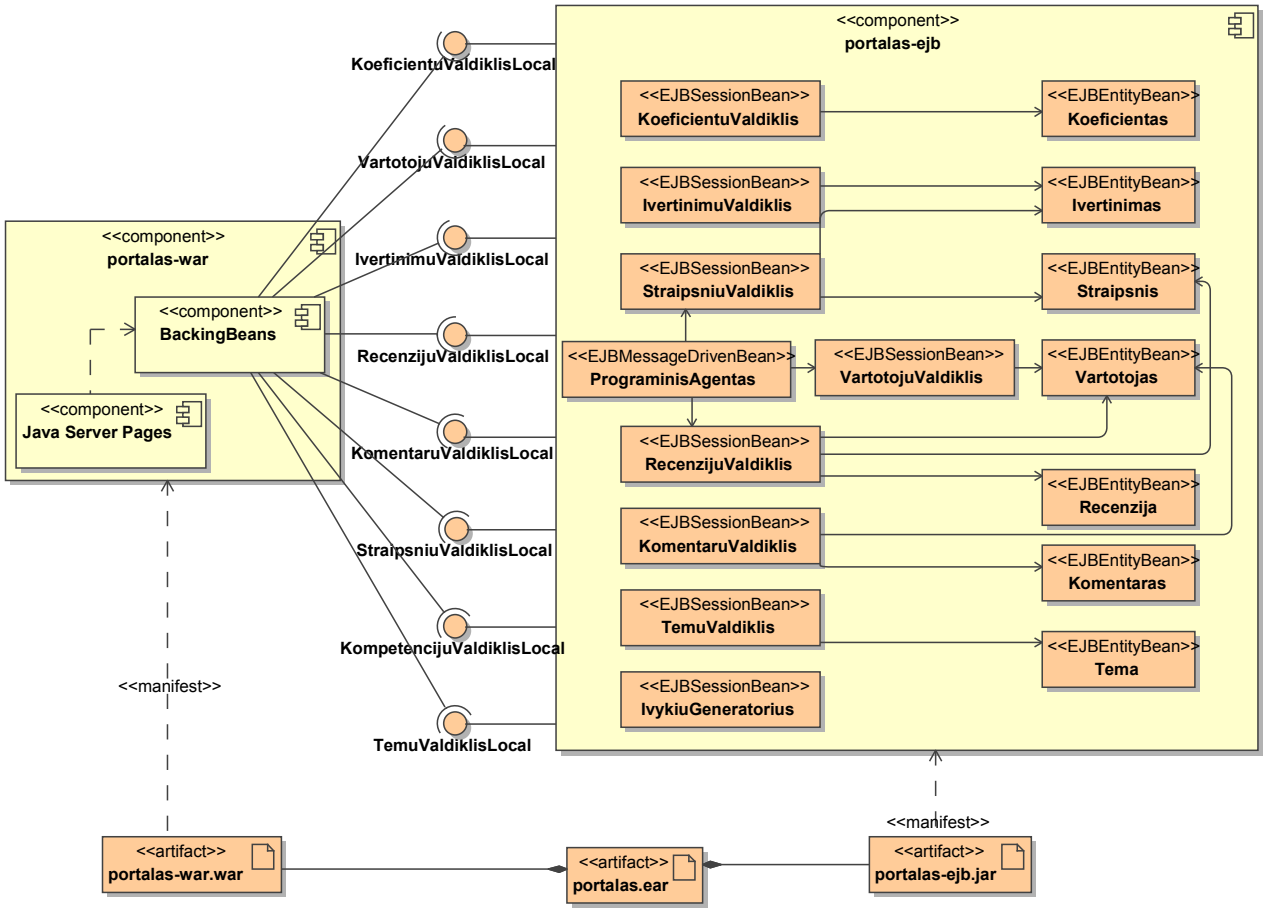
5.7. Realizacijos modelis

Vartotojo sąsaja yra realizuojama *JSP* puslapiais, kurie veiklos logikos komponentus pasiekia naudodami Java klases (*BackingBeans*) per interfeisus. Šios Java klasės, kartu su *JSP* puslapiais sudaro vieną komponentą, kuris realizuojamas artefaktu *portalas.war*.

Veiklos logiką realizuojantys *EJB* komponentai yra: *EJBSessionBean* komponentai kartu su interfeisais, bei *EntityBean* komponentai. Jie sudaro komponentą, kuris realizuojamas artefaktu *portalas.ejb*.

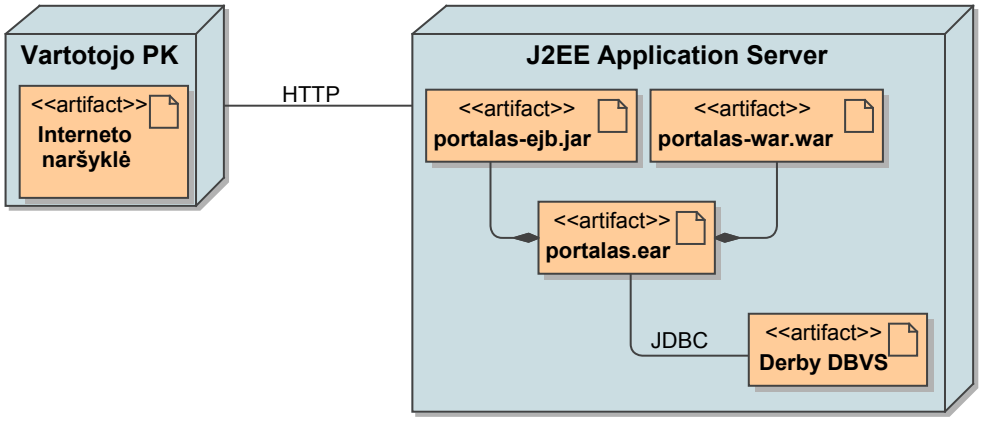
Abu vartotojo sąsajos ir veiklos logikos realizavimo artefaktai apjungiami į bendrą *portalas.ear* artefaktą.

Komponentų diagrama pavaizduota 5.33 paveikslėlyje.



5.33 pav. Komponentų diagrama

Vartotojo sąsają ir veiklos logiką realizuojantys artefaktai yra įdiegiami Java EE taikomųjų uždavinių serveryje. Vartotojas, naudodamasis savo kompiuteryje įdiegta naršykle, atveria *JSP* puslapius. Veiklos komponentai prisijungia prie duomenų bazių valdymo sistemos. Įdiegimo diagrama pavaizduota 5.34 paveikslėlyje.



5.34 pav. Įdiegimo diagrama

6. Metodikos įvertinimas remiantis publikacijų portalo prototipu

Šiame skyriuje atliekamas metodikos įvertinimas, remiantis pagal ją sukurtu publikacijų portalo prototipu. Pirmiausiai trumpai apžvelgiamas sukurto publikacijų portalo prototipo veikimas: pagrindiniai vartotojo langai ir funkcijos, kurias galima atlikti sukurtame prototipe. Metodika įgalino sudaryti konkrečios įvykiais paremtos sistemos projektą ir pagal ją sukurti sistemą, kurios struktūra yra geriau tinkama greitam plėtimui ir modifikavimui, nei įprastu būdu sukurtos. Kadangi faktiškai išmatuoti sistemos kokybines charakteristikas reikėtų didelio eksperimento, įvertinimas apsiriboja aptarimu, kaip šie kriterijai užtikrinami naudojant sukurtą metodiką. Toliau pateikiama apsibrėžtų kokybės kriterijų, pasiekiamų taikant sukurtą metodiką, analizė.

6.1. Pagrindinis publikacijų portalo prototipo langas

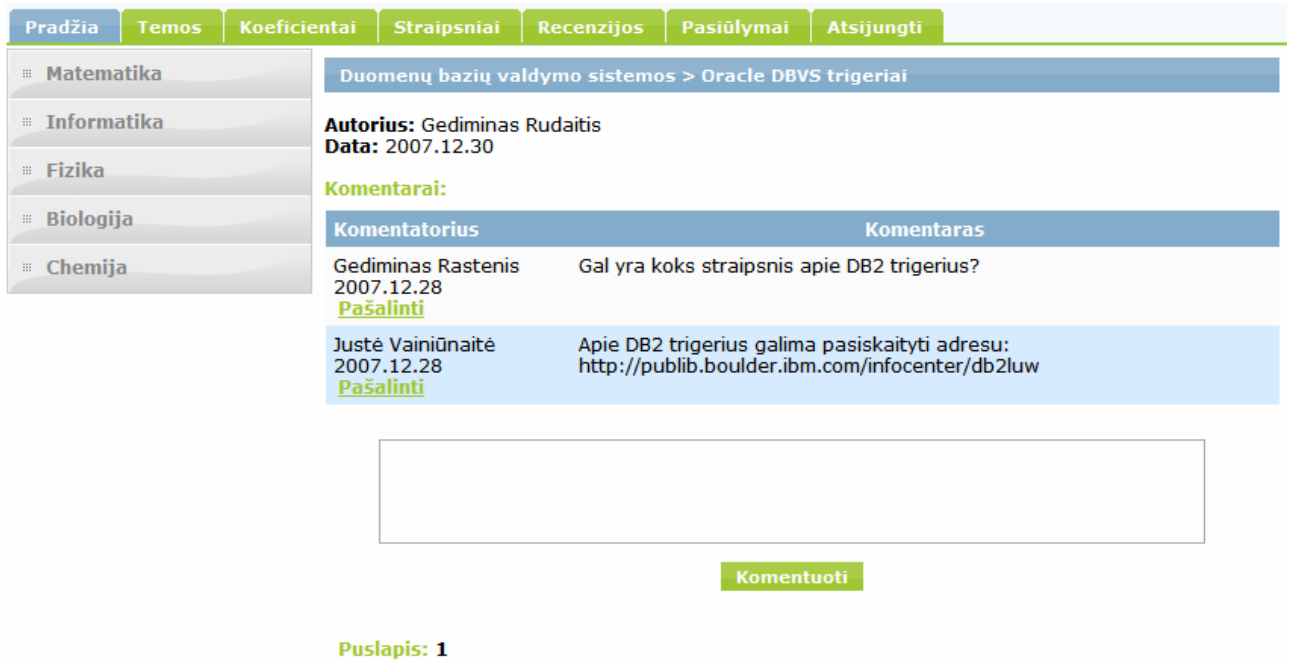
Registruotas sistemos vartotojas prisijungęs prie sistemos, patenka į pagrindinį langą, kuris pavaizduotas 6.1 paveikslėlyje. Neregistruotam sistemos vartotojui pateikiamas tas pats langas, tačiau neregistruotas vartotojas negali vertinti straipsnių, taip pat lange nėra nuorodų, leidžiančių patalpinti straipsnius ar recenzijas, peržiūrėti recenzijų rašymo pasiūlymus.

Nuorodos „Temos“ ir „Koeficientai“ yra matomos tik sistemos administratoriams.

Pagrindiniame lange pateikiamas mokslo sričių sąrašas. Pasirinkus konkrečią mokslo sritį atsiranda temų sąrašas, pagal kurį galima ieškoti konkrečios temos straipsnių, t.y. pasirinkus temą yra pateikiamas ta tema parašytų straipsnių sąrašas.

6.1 pav. Pagrindinis publikacijų portalo prototipo langas

Prie kiekvieno straipsnio yra nuoroda „Komentarai“, kurią paspaudus atsidaro langas su straipsnio komentarais (6.2 pav.). Neregistruotas sistemos vartotojas gali perskaityti komentarus, registruotas komentarus rašyti, o sistemos administratoriui pateikiama galimybė komentarus pašalinti.



6.2 pav. Straipsnio komentarų rašymo langas

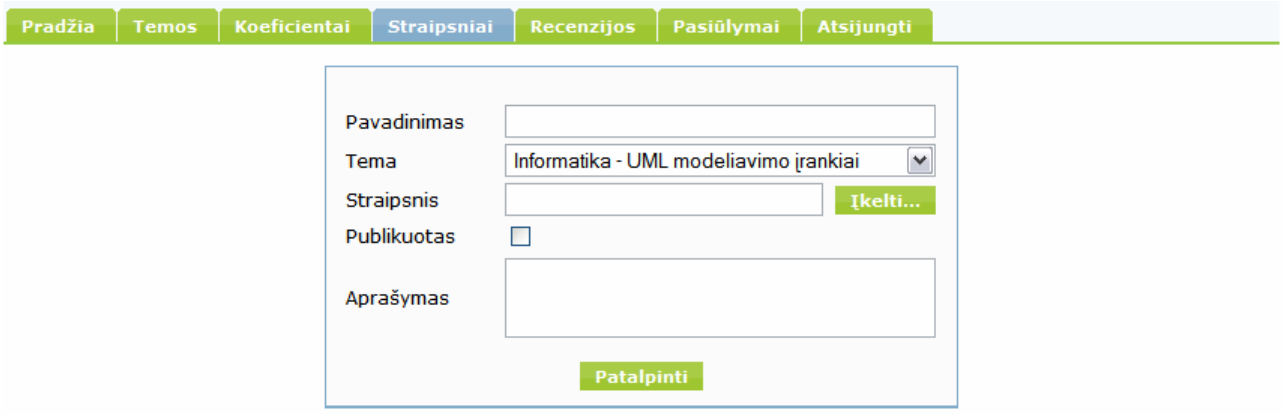
6.2. Straipsnių administravimas publikacijų portale

Registruotas sistemos vartotojas gali patalpinti savo straipsnius portale. Patalpintų straipsnių sąrašas (6.3 pav.) pateikiamas paspaudus nuorodą „Straipsniai“.

6.3 pav. Vartotojo patalpintų straipsnių sąrašas

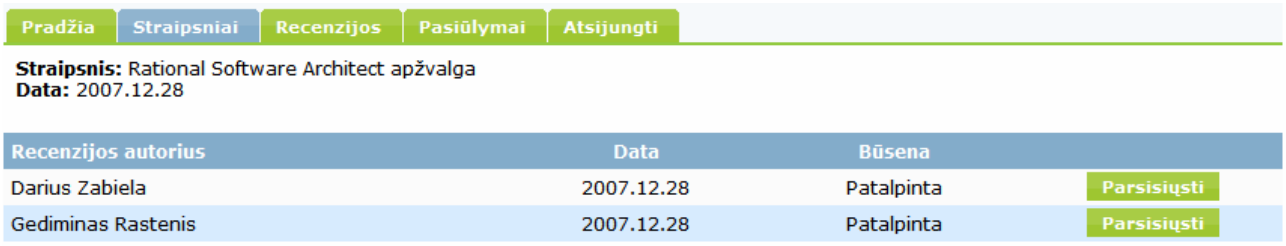
Prie kiekvieno straipsnio yra dvi nuorodos: „Koreguoti“ ir „Recenzijos“. Paspaudus pirmąją nuorodą atveriamas langas, kuriame vartotojas gali koreguoti patalpinti straipsnio duomenis. Toks pat langas (6.4 pav.) atveriamas paspaudus viršuje esančią nuorodą „Patalpinti straipsnį“, tik šiame lange pateikiami ne konkretaus straipsnio duomenys, o leidžiama įvesti naujai talpinamo straipsnio duomenis.

Kiekvienas straipsnis turi atributą „Publikuotas“, kuris parodo, ar straipsnis yra viešai prieinamas kitiems portalo vartotojams.



6.4 pav. Naujo straipsnio patalpinimo langas

Straipsnių sąrašo lange paspaudus nuorodą „Recenzijos“, pateikiamos konkretaus straipsnio recenzijos, kurias vartotojas gali parsisiųsti. Straipsnio recenzijų sąrašo langas pateiktas 6.5 paveikslėlyje.



6.5 pav. Straipsnio recenzijų sąrašo langas

6.3. Recenzijų administravimas publikacijų portale

Sistemos vartotojui patalpinus straipsnį, programinis agentas automatiškai parenka straipsnio recenzentus ir nusiunčia jiems pasiūlymus rašyti recenziją. Recenzijų pasiūlymų sąrašas (6.6 pav.) atidaromas paspaudus nuorodą „Pasiūlymai“. Vartotojas gali priimti arba atmesti pasiūlymą.

6.6 pav. Recenzijų pasiūlymų sąrašo langas

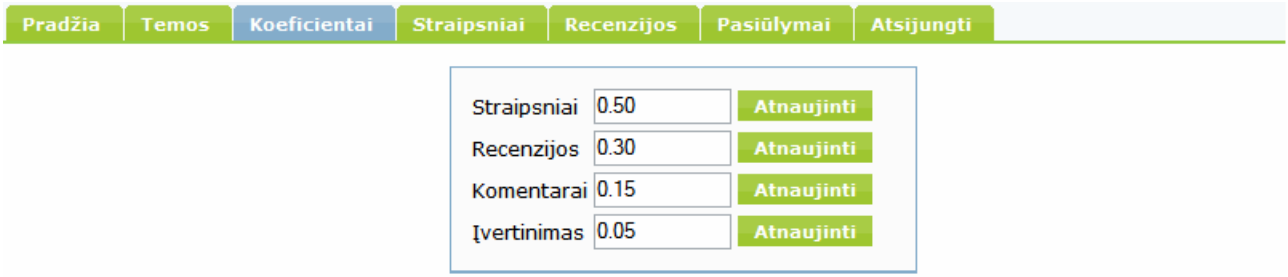
Priėmus pasiūlymą rašyti recenziją, recenzijų sąrašo lange (6.7 pav.) atsiranda naujas įrašas. Šis langas atveriamas paspaudus nuorodą „Recenzijos“. Recenzentas gali patalpinti recenziją, ją atnaujinti arba pašalinti. Apie recenzijos patalpinimą arba atnaujinimą programinis agentas nusiunčia pranešimą straipsnio, kurio recenzijos būsena pasikeičia, autoriui.

6.7 pav. Recenzento recenzijų sąrašo langas

6.4. Administratoriaus funkcijų realizavimas portale

Sistemos administratorius gali pašalinti komentarus, įvesti rangavimo algoritmo koeficientus, sukurti naujas temas ir potemes.

Rangavimo algoritmo koeficientai yra panaudojami programinio agento skaičiuojant vartotojų reitingą ir pagal tai priskiriant jiems atitinkamas roles. Rangavimo koeficientų reikšmės įvedamos lange (6.8 pav.), kuris atsidaro paspaudus nuorodą „Koeficientai“.



6.8 pav. Rangavimo algoritmo koeficientų koregavimo langas

6.5. Kokybinių kriterijų įvertinimas

1.7 poskyryje apibrėžti sistemų vertinimo kriterijai bei prielaidos, kaip įvykiais grindžiamos informacinės sistemos būtų įvertintos pateiktų kriterijų atžvilgiu lyginant su įprastomis objektinėmis sistemomis, t.y. kaip konkretus kriterijus padidėtų ar sumažėtų kuriant įvykiais grindžiamą sistemą. Šiame poskyryje kiekvienas įvertinimo kriterijus aprašomas atsižvelgiant į

pagal pateiktą įvykiais grindžiamų sistemų modeliavimo ir realizavimo metodiką sukurtą publikacijų portalo prototipą (6.1 lentelė).

6.1 lentelė. Įvykiais grindžiamų sistemų kriterijų įvertinimas

Sistemos charakteristika	**Charakteristikos apibūdinimas**
Judrumas (angl. *agility*)	Judrumas įvertinamas atsižvelgiant į sistemos lankstumą, palaikomumą, junglumą. Kadangi šitie kriterijai padidėja, todėl galima teigti, jog įvykiais grindžiamų sistemų „judrumas“ taip pat **padidėja**.
Lankstumas (angl. *flexibility*)	Kadangi įvykiais grindžiamos sistemos kuriamos iš tarpusavyje silpnai susietų komponentų, bet kurio jų pakeitimas yra žymiai paprastesnis, nei objektinėse sistemose, todėl įvykiais grindžiamose sistemose lankstumas **padidėja**.
Junglumas (angl. *interoperability*)	Įvykiais grindžiamų sistemų kūrimo metodikoje apibrėžiama, kad tokios sistemos kuriamos iš komponentų, kurie tarpusavyje siejami pranešimų siuntimu. Šis sistemų kūrimo būdas padeda pasiekti **didesnį** sistemų junglumą, nes pranešimų siuntimu susiejami komponentai yra mažiau priklausomi nei jungiami operacijų kvietimu.
Palaikomumas (angl. *maintainability*)	Įvykiais grindžiamose sistemose komponentai yra mažiau priklausomi vienas nuo kito, todėl vieno komponento pakeitimas neįtakoja kitų sistemos komponentų veikimo. Tai leidžia **padidinti** sistemų palaikomumą.
Pakartotinis panaudojimas (angl. *reusability*)	Kadangi įvykiais grindžiamose sistemose komponentai bendradarbiauja siųsdami pranešimus per tarpinę programinę įrangą ir praktiškai „nežino“ apie vienas kito realizavimo platformą, interfeisus ir t.t., tokiu atveju lengviau **padidinti** sistemų pakartotinį panaudojimą. Norint pasiekti papildomą funkcionalumą užtenka nusiųsti atitinkamą pranešimą, kurį komponentas apdoroja nesigilinant į problemas dėl komponentų sujungimo, realizavimo platformų nevienodumo.
Veikimo sparta (angl. *performance*)	Pranešimų siuntimas tarp komponentų yra asinchroninis, todėl įvykiais grindžiamų sistemų veikimo sparta **sumažėja**.
Plečiamumas (angl. *scalability*)	Įvykiais grindžiamose sistemose komponentai pranešimus siunčia per tarpinę programinę įrangą. Padidėjus sistemos apkrovimui, komponentus galima perkelti į kitas platformas (išskaidyti), nes nesvarbu, kuriame serveryje komponentas įdiegtas, svarbu yra tai, kad jis galėtų siusti ir priimti pranešimus. Ši savybė leidžia **padidinti** sistemų plečiamumą.

7. Išvados

1. Įvykiais grindžiamų informacinių sistemų kūrimo galimybių analizės metu nustatyta, kad nėra praktiškos ir išsamios metodikos, kuri nusakytų, kokius įvykių tipus reikia nagrinėti ir kaip juos identifikuoti, modeliuoti bei realizuoti projektavimo procese tam, kad būtų galima gauti lankstesnes, lengviau modifikuojamas sistemas, kurių lankstumas pasiekiamas laisvai susiejant jas sudarančius komponentus per įvykius.

2. Įvykių tipų ir jų realizavimo galimybių analizė parodė, kad įvykius tikslinga klasifikuoti pagal sudarytą taksonomiją, kurioje apibrėžiami tokie pagrindiniai įvykių tipai: sistemos sukeltas įvykis, vartotojo sužadintas įvykis ir laiko įvykis, kadangi kiekvienas iš šių įvykių tipų modeliuojamas ir realizuojamas skirtingai.

3. Projektuojant įvykiais grindžiamas informacines sistemas tikslinga naudoti šiame darbe sukurtą įvykių metamodelį bei metodiką, kuri nusako:

- kaip identifikuoti įvykius iš panaudojimo atvejų jų išplėtimo taškuose;
- kaip sudaryti įvykių modelį reikalavimų analizės ir specifikacijos etape, bei kaip jį papildyti projektavimo etape, kai suprojektuotos klasės ir komponentai;
- kaip panaudoti įvykių modelio elementus sekų bei būsenų diagramose, modeliuojant sistemos elgseną.

4. Portalo prototipo realizacijos analizė parodė, kad įvykių modeliavimo metodika ir jai tinkamos Java EE technologijos įgalina sukurti lankstesnę programinę realizaciją nei įprastu būdu, kai įvykiai neišskiriami kaip reikšmingi projekto elementai.

5. Portalo prototipo realizacija taip pat atskleidė, kad įvykiais grindžiamos sistemos pasižymi didesniu lankstumu (angl. *flexibility*), junglumu (angl. *interoperability*), plečiamumu (angl. *scalability*) ir pakartotiniu panaudojimu, tačiau tokių sistemų veikimo sparta (angl. *performance*) gali būti mažesnė, nei įprastų objektinių sistemų.

6. Metodikos taikymo galimybių analizė parodė, kad ją tikslinga taikyti kuriant programinius agentus ar sistemas, kurių funkcionavimą veikia ne tik tiesioginiai vartotojo įvykiai, bet ir laiko bei sistemos sužadinami įvykiai, kurių apdorojimą tikslinga automatizuoti.. Tokiomis savybėmis pasižymi daugelis sistemų, skirtų kompiuterizuoti veiklos procesus, todėl sukurta metodika gali padidinti daugelio informacinių sistemų kūrimo efektyvumą.

7. Darbo tematika parengtas straipsnis, kuris pateiktas konferencijai Informacinės technologijos 2008.

8. Terminų ir santraukų žodynas

angl. – terminas anglų kalba.

AOR – (***A**gent–**O**bject–**R**elationship*) – programinių agentų sistemoms modeliuoti skirtas metodas.

API – (***A**pplication **P**rogramming **I**nterface*) – taikomųjų programų kūrimo sąsaja.

DBVS – (***D**uomenų **B**azių **V**aldymo **S**istema*) – kompiuterinė programa ar programų paketas, skirtas duomenų bazės valdymui

Derby – Java kalba parašyta duomenų bazių valdymo sistema.

EJB – (***E**nterprise **J**ava**B**ean*) – Java kalba parašytas komponentas, realizuojantis dalį sistemos veiklos logikos.

JMS – (***J**ava **M**essage **S**ervice*) – taikomųjų programų kūrimo sąsaja, skirta kurti, siųsti ir priimti pranešimus Java programavimo kalboje.

SOA – (***S**ervice **O**riented **A**rchitecture*) – sistemų architektūros stilius, kai sistemos funkcionalumas sukuriamas sujungiant atskirus servisus (paslaugas).

SQL – (***S**tructured **Q**uery **L**anguage*) – struktūrizuota užklausų kalba; populiariausia iš šiuo metu naudojamų kalbų, skirtų aprašyti duomenis ir manipuliuoti jais reliacinių duomenų bazių valdymo sistemose.

UML – (***U**nified **M**odeling **L**anguage*) – vieninga modeliavimo kalba; modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti programų dokumentus.

XML – (*e**X**tensible **M**arkup **L**anguage*) – yra W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.

9. Literatūros sąrašas

[1] Peter Pietzuch, Gero Muhl, Ludger Fiege. Distributed Event-Based Systems: An Emerging Community. *IEEE Distributed Systems Online*, 2007.

[2] Szabolcs Rozsnyai, Josef Schiefer, Alexander Schatten. Concepts and Models for Typing Events for Event-Based Systems, 2006.

[3] Tarak Modi. Event-Driven Architecture: Achieving Architectural Agility, 2005.

[4] Gerd Wagner. A UML Profile for External AOR Models, 2005.

[5] James Odell. Modeling-Notation Source: AOR, 2003.

[6] Stefan Brantschen, Thomas Haas. Agents in a J2EE World, 2002.

[7] Dick Cowan, Martin Griss. Making Software Agent Technology available to Enterprise Applications, 2002.

[8] Jennifer Ball, Debbie Narson. The Java EE 5 Tutorial, 2006.

[9] Rene Meier, Vinny Cahill. Taxonomy of Distributed Event-Based Programming Systems, 2004.

[10] Sheung-On Choy, Sin-Chun, Yiu-Chung Tsang. Building Software agents to assist teaching in distance learning environments, 2005.

[11] Debu Panda. Using Timers in J2EE Applications, 2004.

[12] 4th International Workshop on Distributed Event-Based Systems [žiūrėta 2007.09.15]. Prieiga per Internetą: < http://www.cs.queensu.ca/~dingel/debs05>

[13] Gabriel Morgan. Implementing System Quality Attributes [žiūrėta 2007.12.15]. Prieiga per internetą <http://blogs.msdn.com/gabriel_morgan/archive/2007/03/20/implementing-system-quality-attributes.aspx>

[14] Comparing the Timer Classes in the .NET Framework Class Library [žiūrėta 2007.09.20]. Prieiga per Internetą: <http://msdn.microsoft.com/msdnmag/issues/04/02/TimersinNET/default.aspx>.

[15] How to Use Swing Timers [žiūrėta 2007.09.20]. Prieiga per Internetą: <http://java.sun.com/docs/books/tutorial/uiswing/misc/timer.html>

[16] Introduction to Event Listeners [žiūrėta 2007.09.18]. Prieiga per Internetą: <http://java.sun.com/docs/books/tutorial/uiswing/events/intro.html>

[17] Wikipedia. Database Trigger [žiūrėta 2007.09.19]. Prieiga per Internetą: <http://en.wikipedia.org/wiki/Database_trigger>

[18] Unified Modeling Language. Superstructure Specification Version 2.0. OMG document formal/05-07-04, 2005. Prieiga per Internetą: <http://www.omg.org>

[19] Webopedia. Event definition [žiūrėta 2007.09.15]. Prieiga per Internetą: <http://www.webopedia.com/TERM/e/event.html>

10. Priedai

10.1. Straipsnis „Įvykiais grindžiamų informacinių sistemų modeliavimo ir realizavimo metodika“

ĮVYKIAIS GRINDŽIAMŲ INFORMACINIŲ SISTEMŲ MODELIAVIMO IR REALIZAVIMO METODIKA

Gediminas Rudaitis, Lina Nemuraitė,

Kauno technologijos universitetas, Informacijos sistemų katedra, Studentų g. 50, Kaunas

Straipsnyje nagrinėjami įvykių modeliavimo metodai bei jų realizavimo galimybės. Atsižvelgiant į metodų analizę ir sudarytą įvykių taksonomiją yra pateikiamas įvykių metamodelis, kuris panaudojamas aprašant įvykius, jų sužadinimo metu siunčiamus pranešimus, pranešimų gavėjus ir siuntėjus. Šio metamodelio pagrindu sudarytas įvykių modelis yra panaudojamas įvykiais grindžiamų sistemų modeliavimo ir realizavimo metodikoje.

1. Įvadas

Pastaruoju metu informacinių technologijų pasaulyje dažnai minimos paslaugomis ir įvykiais grindžiamų sistemų architektūros. Greitai besikeičiančiame verslo pasaulyje įmonėms yra aktualu turėti sistemas, kurios greitai ir lengvai būtų priderintos prie besikeičiančių verslo procesų ir sumažintų tokių sistemų kūrimo ir palaikymo išlaidas. Naudodamos paslaugomis grindžiamą architektūrą, įmonės gali lanksčiai tvarkyti savo veiklą ir kurti sistemas, nekeisdamos turimos techninės ar programinės įrangos.

Nors literatūroje yra gausu straipsnių apie įvykiais grindžiamas sistemas ir jų privalumus, tačiau sistemos architektui, nusprendusiam sukurti įvykiais grindžiamos sistemos projektą, iškyla aktualus klausimas: kaip aprašyti ir modeliuoti įvykius, kurie ateina iš išorės arba sugeneruojami pačios sistemos bei vartotojų veiksmų su sistema metu. Akivaizdu, kad šiuo metu nėra populiarios metodikos, kuri apibrėžtų, kaip reikia atlikti įvykių modeliavimą ir jį panaudoti informacinių sistemų kūrimo procese.

Kita svarbi problema, su kuria susiduria programuotojai, yra įvykių apdorojimo ir generavimo mechanizmų realizavimas taikomųjų uždavinių serveriuose. Yra svarbu ne tik žinoti, kaip modeliuoti įvykiais grįstas sistemas, bet ir kaip geriau jas realizuoti.

Šiame straipsnyje pateikiamas įvykio metamodelis bei metodika, kuri panaudoja įvykių modelį ir yra skirta įvykiais grindžiamų sistemų projektavimui ir realizavimui. Įvykių metamodelis sudarytas remiantis SARI sistemoje naudojamu įvykių modeliu bei AOR (angl. *Agent–Object–Relationship*) modeliavimu.

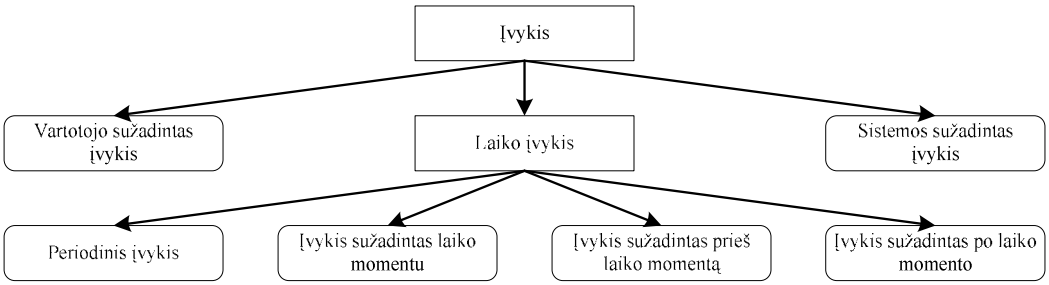
2. Modeliavimo ir realizavimo metodų analizė

2.1. Įvykių tipų analizė

Įvykius svarbu suklasifikuoti į tipus, norint sudaryti metamodelį, kurio pagrindu būtų galima sudaryti modelius, vaizduojančius įvykius, jų apdorojimą ir sužadinimą įvykiais pagrįstose sistemose. UML specifikacijoje [4] įvykiai skirstomi į signalų siuntimo, operacijų kvietimo, laiko ir pasikeitimų įvykius. Sudarant įvykių metamodelį tokia klasifikacija yra nepakankama, kadangi tikslinga detalizuoti laiko įvykių tipus ir išskirti sistemos bei vartotojų sužadintus įvykius.

Įvykis – veiksmas ar atsitikimas, kuris yra apdorojamas kompiuterinės programos. Tai gali būti vartotojo sukeltas veiksmas, toks kaip pelės mygtuko ar klaviatūros klavišo paspaudimas, arba sistemos sugeneruotas veiksmas [19]. Atsižvelgiant į pateiktą įvykio apibrėžimą, galima išskirti du įvykių tipus: vartotojo sukurtas įvykis ir sistemos sukurtas įvykis.

Dažnai informacijos sistemos įvykiai turi būti sugeneruoti automatiškai tam tikrais laiko momentais, neatsižvelgiant į vartotojo atliekamus veiksmus arba programos veikimo logiką, todėl galima išskirti dar vieną įvykių tipą: laiko įvykis. Laiko įvykis gali įvykti tam tikru laiko momentu arba ciklišk ai. Galima įvykių klasifikavimo schema yra pateikta 1 paveiksle.



1 pav. Įvykių tipai

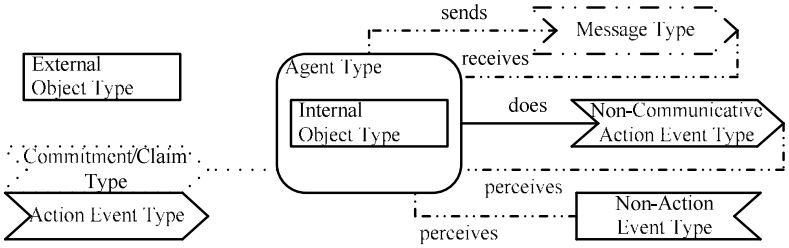
2.2. Įvykių modeliavimo metodų analizė

Įvykiais grindžiamos sistemos kuriamos iš komponentų, kurie yra pranešimų siuntėjai ir gavėjai, t.y. bendradarbiavimas tarp komponentų vyksta siunčiant įvykius aprašančius pranešimus. Tai leidžia sukurti sistemas, kuriose komponentai yra susieti silpnais ryšiais, ir padidinti sistemų efektyvumą, plečiamumo galimybes. Tačiau tokių sistemų trūkumas yra tas, kad pranešimų gavėjai turi suprasti pranešimų turinį ir struktūrą ir žinoti, koks įvykis turi būti apdorotas. Šios problemos sprendimas gali būti įvykių modelio sudarymas, kuriame tiksliai aprašomi visi sistemoje naudojami įvykiai ir jų struktūra. Kuriant komponentus, toks įvykių modelis gali būti naudojamas kaip protokolas, kuris padėtų suprasti, kokie įvykiai yra galimi, kas juos turėtų apdoroti ir kokios yra įvykių aprašymo taisyklės.

Szabolcs Rozsnyai, Josef Schiefer ir Alexander Schatten savo publikacijoje „Concepts and Models for Typing Events for Event-Based Systems“ pateikia įvykių tipų metamodelį, kuris yra naudojamas SARI įvykiais grindžiamoje sistemoje. Šis metamodelis nusako, kaip turi būti aprašomi įvykių tipai, apibūdinant įvykių pobūdį ir struktūrą. Įvykių tipų aprašymas palengvina jų apdorojimą, kadangi yra tiksliai žinoma, kokie atributai aprašo konkretų įvykį, kokie yra atributų tipai [1]. Pagrindinė šio metamodelio idėja yra įvykių aprašymas atributais. Nors SARI įvykių metamodelis parodo, kokia yra įvykių modelio sudarymo struktūra, kaip įvykiai skirstomi į tipus, tačiau jis neparodo, kas konkrečius įvykius turėtų apdoroti.

Kitas modeliavimo metodas, kuriame atsispindi ne tik sistemoje sužadinami įvykiai, bet ir agentai, kurie tuos įvykius turi apdoroti, yra AOR (angl. *Agent–Object–Relationship*) modeliavimas. Nors šis metodas yra tinkamesnis modeliuoti programinių agentų sistemas, tačiau jis gali būti panaudotas siekiant atvaizduoti įvykių ir pranešimų tipus bei juos apdorojančius objektus ar komponentus. AOR modelyje esybės yra skirstomos į dvi grupes: aktyvias (agentus) ir pasyvias (realaus pasaulio objektai). AOR modelyje esybe gali būti agentas, įvykis (angl. *event*), veiksmas (angl. *action*), pareikalavimas (angl. *claim*), įsipareigojimas (angl. *commitment*) ar įprastas objektas [4], [5].

Yra du pagrindiniai AOR modeliai: išorinis ir vidinis. Išoriniame modelyje pateikiamas išorinio stebėtojo matomas vaizdas apie agentus, veikiančius ir bendradarbiaujančius dalykinėje srityje, t.y. šis modelis vaizduoja visus agentus, kuriuos reikia atvaizduoti sudarant statinį ir elgsenos modelius. Vidinis modelis vaizduoja konkretaus agento veikimą sistemoje, jo elgseną ir tikslus. Pagrindiniai išorinio modelio elementai yra pavaizduoti 2 paveiksle.



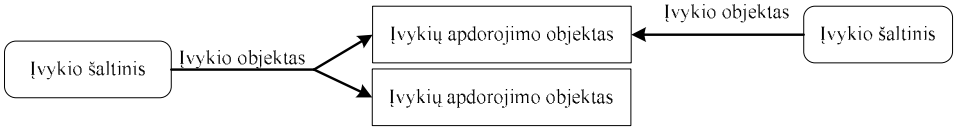
2 pav. Pagrindiniai AOR išorinio modelio elementai

2.3. Įvykių realizavimo technologijų analizė

Šiame poskyryje apžvelgiamos kiekvieno įvykio tipo, aprašyto 2.1 poskyryje, realizavimo galimybės.

- **Vartotojo sužadinamų įvykių realizavimo galimybės**

Vartotojo sužadinamas įvykis – įvykis, kurį sužadina vartotojas, paspausdamas mygtuką ekrane, pakeisdamas formos lauko reikšmę, pažymėdamas keletą punktų pasirinkimo sąraše ar pan. Realizuojant vartotojo įvykius, *Java* programavimo kalboje suprogramuojamos klasės, kurios sukuria atitinkamus interfeisų metodus. Pavyzdžiui, mygtuko paspaudimo apdorojimui reikalingas objektas klasės, kuri realizuoja *java.awt.event.ActionListener* interfeisą. Tokių klasių objektai naudojami įvykių apdorojimui. Atitinkamas vartotojo sąsajos komponentas (pavyzdžiui, mygtukas) užregistruoja objektą kaip atitinkamo įvykio apdorojimo priemonę. Kuomet įvykis yra sužadinamas, jis perduodamas konkrečiam objektui, kuris įvykdo atitinkamus metodus ir apdoroja perduotą įvykį [16]. 3 paveiksle pavaizduotas atvejis, kai tam pačiam įvykių apdorojimo objektui (angl. *event listener*) yra perduodami įvykiai iš skirtingų šaltinių (skirtingų vartotojo sąsajos komponentų sugeneruoti įvykiai);



3 pav. Įvykių apdorojimo schema Java kalboje

Panašiai vartotojo sąsajos komponentų sugeneruojami įvykiai apdorojami ir kitose objektinio programavimo kalbose.

- **Sistemos sužadinamų įvykių realizavimo galimybės**

Sistemos sužadintas įvykis yra įvykis, kurį sužadina pati sistema, vykdydama veiklos logiką realizuojantį metodą. Tokie įvykiai dažniausiai sužadinami, kai atitinkama sąlyga tenkina veiklos taisyklę. Šie įvykiai gali būti generuojami programiniame kode, kai yra tenkinama atitinkama sąlyga, panaudojant *if..else* sakinius:

```
if (tenkinama veiklos taisyklė) {
      ...
      sugeneruoti atitinkamą įvykį
      ...
}
```

Prie sistemos sužadinamų įvykių galima priskirti ir šiuolaikinių duomenų bazių trigerių atliekamus veiksmus. Trigeriai yra sužadinami prieš arba po įrašo šalinimo, įterpimo ar atnaujinimo operacijos. Galima nurodyti, kokias lentelės kolonėlių reikšmes modifikuojant gali būti sužadintas trigeris. Taip pat trigeriuose yra galimybė sukurti kintamuosius, kurie saugo senas ir naujas modifikuojamo įrašo reikšmes.

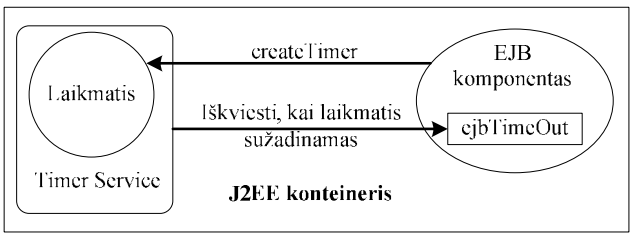
- **Laiko įvykių realizavimo galimybės**

Laiko įvykis – tai įvykis, kuris yra sužadinamas tam tikru laiko momentu. Jis gali būti sužadinamas vieną kartą, arba periodiškai, tam tikrais laiko momentais.

Laiko įvykiai sistemose gali būti realizuoti panaudojant laikmačius (angl. *Timers*). Šį funkcionalumą palaiko daugelis objektinio programavimo kalbų. Laikmačiai yra sukuriami kaip objektai, nurodant, kada laikmatis turi būti sužadintas, koks yra sužadinimo intervalas ir kokie metodai turi būti atlikti sužadinimo metu. *Java* programavimo kalboje laikmatis realizuojamas kaip klasės *javax.swing.Timer* objektas. Žemiau pateiktas laikmačio sukūrimo ir paleidimo programinis kodas [15]:

```
int delay = 1000; //milliseconds

ActionListener taskPerformer = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        ...
    }
};

new Timer(delay, taskPerformer).start();
```

Kadangi dauguma šiuolaikinių verslo sistemų yra perkeliamos į internetą, t.y. kuriamos žiniatinklio taikomosios programos, iškyla aktualus klausimas, kaip laikmačius realizuoti taikomųjų uždavinių serveriuose. Java EE 5.0 technologija turi specialią taikomųjų programų kūrimo sąsają (angl. *Application Programing Interface*), kurioje apibrėžti interfeisai reikalingi laikmačiams kurti. Ši sąsaja vadinama *Timer Service.* 4 paveiksle pavaizduotas ryšys tarp *EJB* komponento realizuojančio laikmatį ir laikmačių valdymo paslaugos, kurią teikia Java EE 5.0 technologiją palaikantis serveris.



4 pav. Java EE laikmačio veikimas

3. Įvykių metamodelis

Šiame skyriuje pateikiamas įvykių metamodelis, kurio pagrindu sudarytas įvykių modelis naudojamas įvykiais grindžiamų sistemų projektavimo procese. Kitaip tariant, įvykių metamodelis apibrėžia taisykles, pagal kurias sudaromi konkrečių sistemų įvykių modeliai. Sudarant įvykių metamodelį, įvykis yra specializuojamas į dvi grupes: sistemos įvykį ir laiko įvykį; šioms grupėms reikia apibrėžti skirtingas asociacijas su kitais metamodelio elementais.

Kartais įvykis sužadinamas tik tada, kai yra tenkinama tam tikra veiklos taisyklė, todėl įvykis asociacijos ryšiu gali būti susietas su viena ar daugiau taisyklių; tai reiškia, kad įvykis bus sužadintas tik esant atitinkamoms sąlygoms

Sistemos įvykis yra susiejamas asociacijos ryšiu su įvykio sukėlėju, kuris gali būti vartotojo atliekamas veiksmas arba sistemos operacijos vykdymas, kai vykdant veiklos logiką ir esant tam tikroms sąlygoms turi būti sužadinamas įvykis. Aprašant įvykio sukėlėją, gali būti nurodytas komponentas, paketas, klasė bei klasės operacija, kurią vykdant yra generuojamas įvykis. Šie atributai nėra privalomi ir sistemos projektavimo analizės etape, kai tiksliai dar nėra žinomi, gali būti neaprašomi.

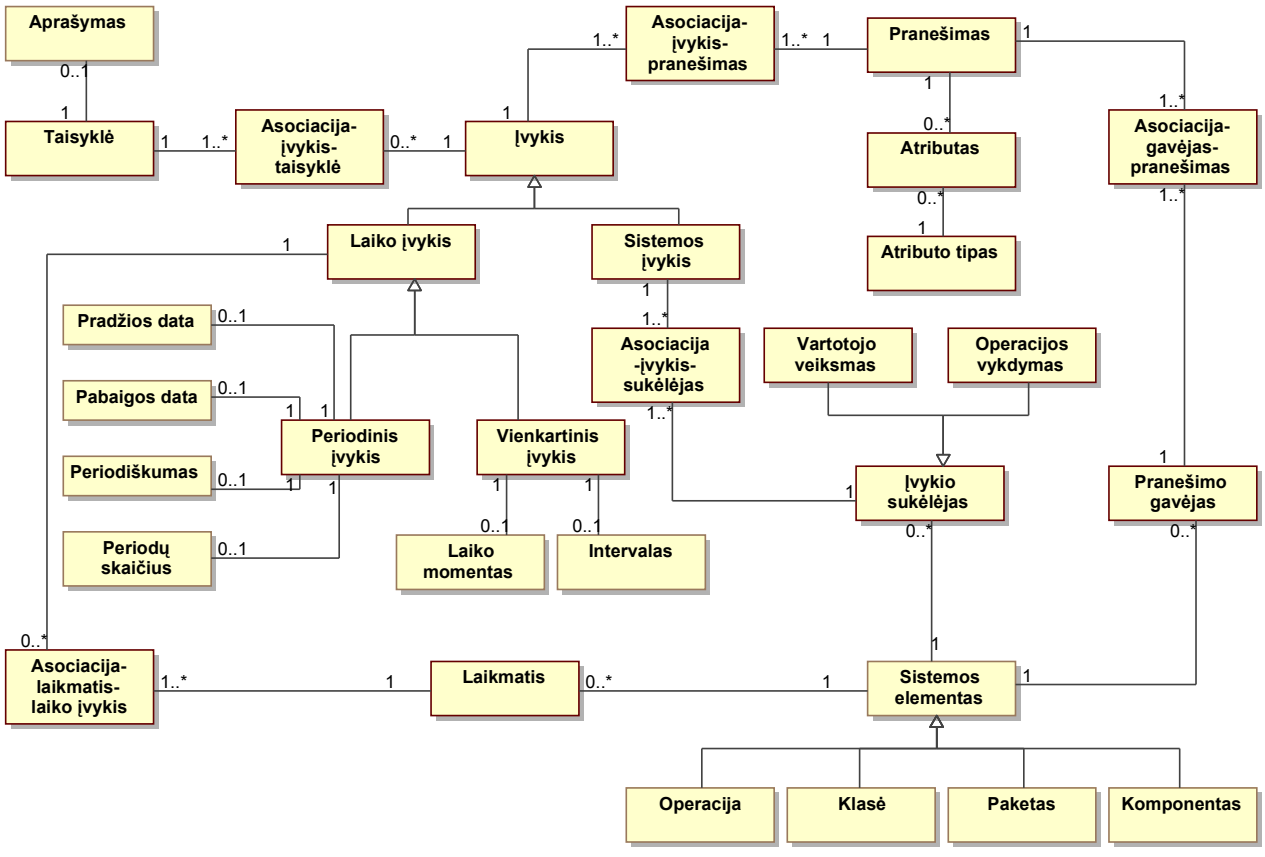
Vartotojo sukelti įvykiai taip pat yra apdorojami sistemos, iškviečiant atitinkamas operacijas (pavyzdžiui, metodą *onClick*). Tačiau skirstymas tarp įprastų operacijų ir vartotojo veiksmų modeliui suteikia daugiau aiškumo. Sudarant įvykių modelį galima atskirose diagramose aprašyti vartotojo sužadinamus įvykius ir sistemos sugeneruojamus įvykius.

Laiko įvykis asociacijos ryšiu susiejamas su laikmačiu, kuris jį sugeneruoja. Laikmačiui gali būti nurodytas komponentas, paketas ar klasė, kuri jį realizuoja. Laiko įvykiai yra specializuojami į dvi grupes: periodiniai ir vienkartiniai laiko įvykiai. Vienkartinis laiko įvykis yra sužadinamas vieną kartą tam tikru laiko momentu, todėl gali būti nurodytas laiko momentas, kada įvykis turi būti sugeneruotas. Aprašant periodinį įvykį, nurodoma pradžios data, kada turi būti sugeneruotas pirmasis įvykis, pabaigos data, po kurios periodinis įvykis yra nebegeneruojamas, bei periodiškumas (laiko intervalas, nusakantis įvykių generavimo dažnumą) ir periodų skaičius. Šie atributai nėra privalomi.

Sudarant įvykių modelį yra svarbu ne tik aprašyti visus galimus įvykius, bet ir nurodyti, kas juos turi apdoroti. Apie įvykio sužadinimą komponentams ar objektams, kurie turi įvykį apdoroti, yra pranešama siunčiant pranešimus, todėl metamodelis taip pat aprašo, kaip atvaizduoti ir pranešimus.

Įvykis gali būti susietas su keletu pranešimų, t.y. sužadinus įvykį gali būti siunčiami vienas ar daugiau pranešimų. Norint aprašyti pranešimų turinį, galima nurodyti atributus bei jų pradines reikšmes, pagal kurias turi būti suformuotas pranešimas. Pranešimas yra susiejamas su vienu ar daugiau pranešimų gavėjų. Pranešimų gavėju gali būti komponentas ar klasė, kuri atitinkamą įvykį apdoroja. Taip pat galima nurodyti paketą, kuriame realizuojama įvykį apdorojanti klasė.

Įvykių metamodelis pavaizduotas 5 paveiksle.



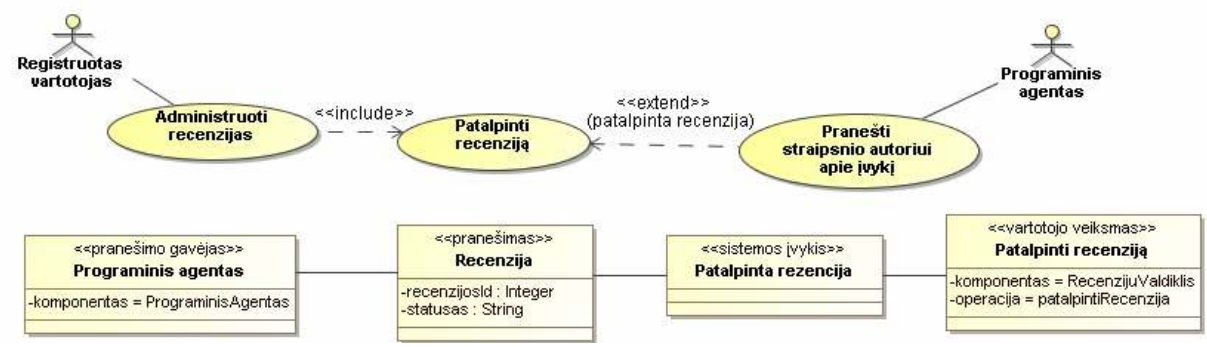
5 pav. Įvykių metamodelis

4. Įvykiais grindžiamų sistemų modeliavimo ir realizavimo metodika

Šiame skyriuje aprašomas įvykių modelio, sudaryto pagal anksčiau pateiktą metamodelį, panaudojimas informacinių sistemų projektavimo procese. Aprašomos įvykių identifikavimo galimybės iš panaudojimo atvejų modelių, nurodoma, kaip įvykių modelis siejasi su kitais projektavimo modeliais: klasių, sekų, būsenų diagramomis.

4.1. Įvykių identifikavimas iš panaudojimo atvejų diagramų

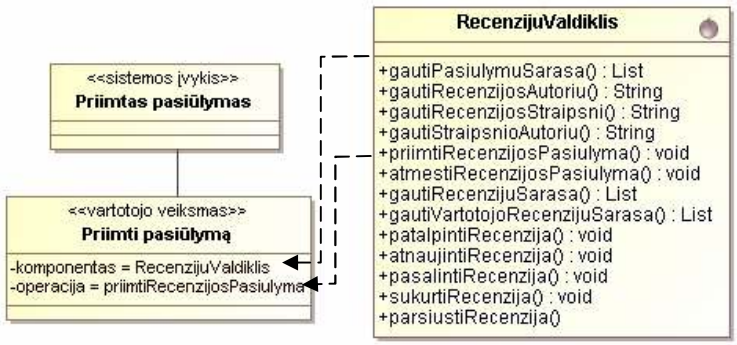
Panaudojimo atvejų diagramos dažniausiai naudojamos reikalavimų analizės etape aprašyti kompiuterizuojamas funkcijas. Panaudojimo atvejai (scenarijai) su kitais panaudojimo atvejais jungiami išplėtimo (angl. *extend*) ryšiu, nurodant išplėtimo taškus, t.y. sąlygas, kurioms esant yra atliekami panaudojimo atvejai (6 pav.). Išplėtimo taškai, aprašyti panaudojimo atvejų modelyje, naudojami įvykių identifikavimui ir atvaizdavimui įvykių modelyje.



6 pav. Įvykių modelio elementų ryšys su panaudojimo atvejų modeliu

4.2. Įvykių sukėlėjai ir pranešimų gavėjai klasių diagramose

Įvykių metamodelyje nurodyta, kad įvykius gali sužadinti vartotojo veiksmai; pati sistema, vykdydama veiklos logiką realizuojančius metodus; arba laikmačiai. Vartotojo veiksmai taip pat yra apdorojami metodų (operacijų). Įvykių modelyje, aprašant įvykių sukėlėjus ir pranešimų gavėjus, nurodomos klasės, komponentai ar operacijos, kurios sužadina arba apdoroja įvykius. Šie elementai yra modeliuojami klasių diagramose, todėl komponentai, klasės ar jų operacijos, aprašomos įvykių modelyje, atvaizduojamos ir klasių modelyje (7 pav.).



7 pav. Įvykių modelio elementų ryšys su klasių diagramomis

4.3. Pranešimų siuntimas sekų diagramose

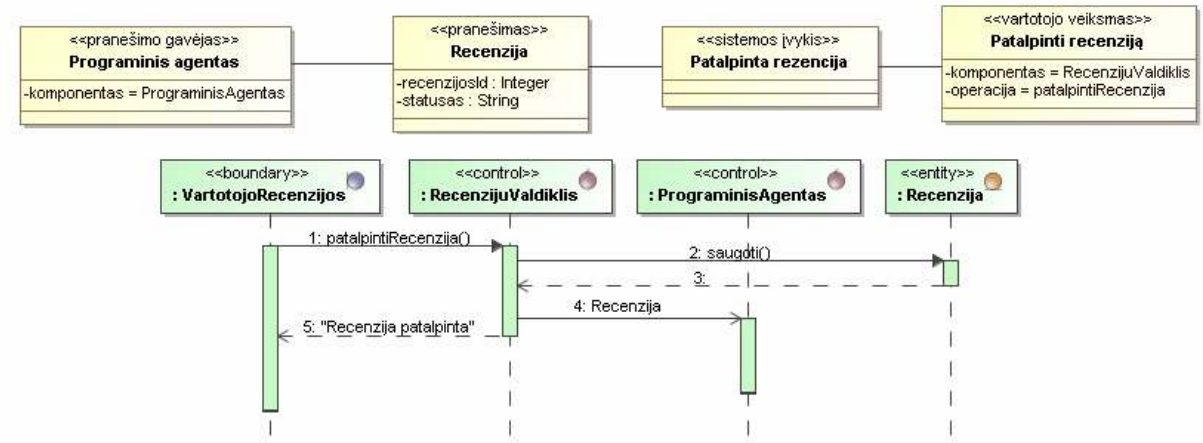
Sekų diagramos vaizduoja bendradarbiavimą tarp klasių ar komponentų. Šie elementai bendradarbiauja siųsdami vieni kitiems pranešimus arba kviesdami atitinkamas operacijas.

Sekų diagramose galima atvaizduoti šiuos įvykių modelio elementus (8 pav.):

- komponentus ar klases, kurios realizuoja įvykių sukėlėjus;
- operacijas, kurias vykdant sužadinami įvykiai;
- pranešimus, kurie yra siunčiami tarp klasių ir komponentų;
- komponentus ar klases, kurie įvykius apdoroja (pranešimų gavėjus).

Sekų diagramas ir įvykių modelį sieja tokios taisyklės:

- pranešimą turi siųsti įvykio sukėlėjo elemente aprašyta klasė ar komponentas;
- pranešimas yra siunčiamas, kai iškviečiama įvykio sukėlėjo elemente aprašyta operacija;
- pranešimas yra siunčiamas pranešimo gavėjo elemente nurodytai klasei ar komponentui.

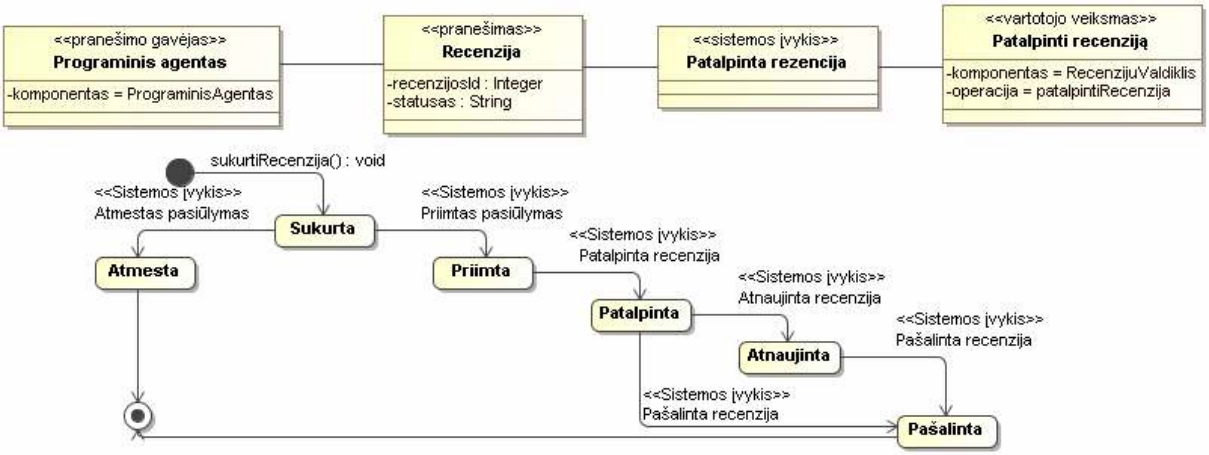


8 pav. Įvykio modelio ryšys su sekų diagramomis

4.4. Įvykių atvaizdavimas būsenų diagramose

Būsenų diagramos naudojamos atvaizduoti objekto būsenų kaitą. Objektas pereina iš vienos būsenos į kitą, kai yra iškviečiama atitinkama operacija, įvyksta numatytas įvykis, praeina tam tikras laiko tarpas po įvykio ir pan., todėl sudarant būsenų diagramas galima naudoti įvykio modelio elementus.

Yra trys galimi būdai panaudoti įvykio modelio elementus būsenų diagramose (9 pav.). Virš perėjimus iš vienos būsenos į kitą vaizduojančių rodyklių galima rašyti įvykius (laiko ar sistemos), kuriems įvykstant objektas pakeičia būseną. Taip pat galima vaizduoti ir operacijas, aprašytas įvykio sukėlėjo elemente. Tai reiškia, kad objektas pereina į kitą būseną, kai yra iškviečiama atitinkama klasės operacija. Trečias būdas įvykių modelio elementus panaudoti būsenų diagramose yra pavaizduoti virš būsenų perėjimus žyminčių rodyklių laiko momentus, aprašytus laiko įvykiuose (objektas pakeičia būseną tam tikrais laiko momentais).



9 pav. Įvykio modelio ryšys su būsenų diagramomis

4.5. Metodikos apibendrinimas

Apibendrinus 3.2–3.5 poskyriuose pateiktą informaciją, galima sudaryti tokią įvykių modeliavimo projektavimo procese metodiką:

- reikalavimų etape įvykiai identifikuojami panaudojimo atvejų modelyje, panaudojimo atvejų išplėtimo taškuose;
- identifikavus įvykius, reikalavimų analizės etape sudaromas pradinis įvykių modelis, kuriame išvardinami visi pranešimai, įvykiai, jų sukėlėjai ir pranešimų gavėjai. Šis modelis neprivalo būti detalus; jame nebūtina nurodyti klases, komponentus ir operacijas, nes neatlikus projektavimo šie elementai nėra tiksliai žinomi;
- sistemos architektūrinio projektavimo etape apibrėžiami visi komponentai ir klasės, kurie realizuos įvykių sukėlėjus ir pranešimų gavėjus. Taip pat būtina įvardinti laikmačius, kurie generuos laiko įvykius;
- kai žinomi komponentai ir klasės, kurie realizuos įvykių sukėlėjus ir pranešimų gavėjus, reikalavimų analizės etape sudarytas įvykių modelis yra modifikuojamas: nurodomos operacijos, kurias vykdant sužadinami įvykiai, taip pat apibrėžiami laikmačiai, kurie generuos laiko įvykius, apibrėžiama pranešimų struktūra, nurodant atributus, kurie nusako, kokie duomenys turi būti siunčiami pranešime;

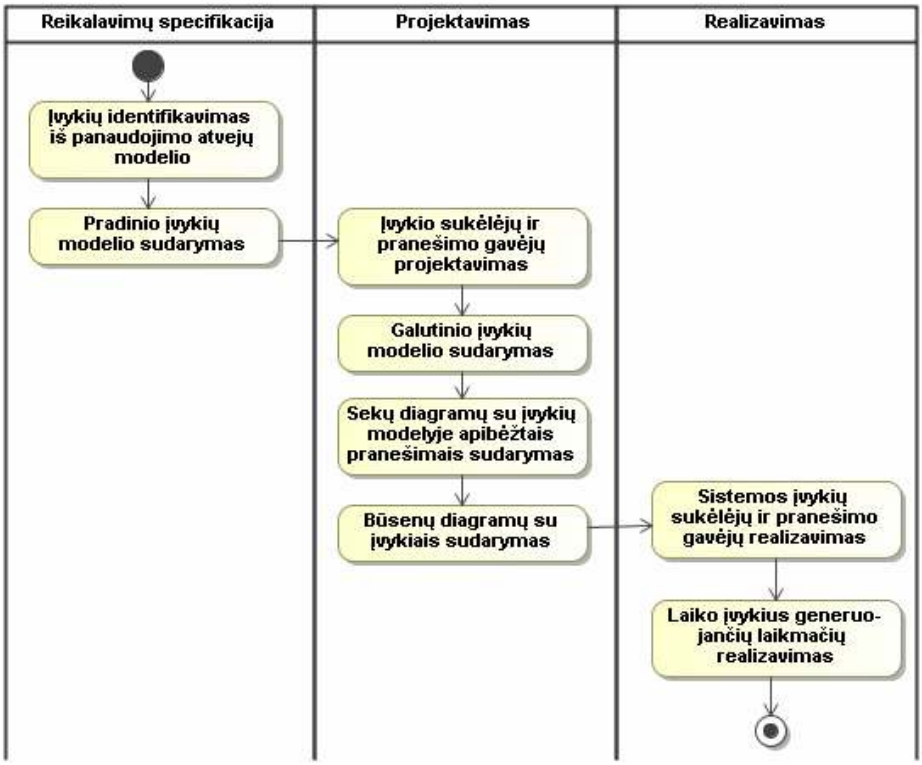
• turint tikslų laiko įvykių modelį ir sudarant sekų diagramas, vaizduojančias bendradarbiavimą tarp klasių, panaudojami įvykių modelyje aprašyti pranešimai, nurodant kokią operaciją iškvietus yra sužadinamas įvykis ir siunčiamas pranešimas;

• sudarant būsenų diagramas, virš būsenos pasikeitimą žyminčių rodyklių nurodomi tokie įvykių modelio elementai: įvykis, operacija arba laiko momentas. Šie elementai nurodo sąlygas, kurioms esant pasikeičia objekto būsena;

• sumodeliuoti sistemos įvykiai realizuojami suprogramuojant komponentus, klases ir jų operacijas, kurie buvo apibrėžti įvykių modelyje prie kiekvieno įvykio. nurodant sistemos elementus, kurie konkretų įvykį realizuoja;

• laiko įvykiai realizuojami suprogramuojant įvykių modelyje apibrėžtus laikmačius.

Įvykių modelių taikymas įvykiais grindžiamų informacinių sistemų projektavimo procese pavaizduotas 10 paveiksle.



10 pav. Įvykių modelių panaudojimas įvykiais grindžiamų informacinių sistemų projektavimo procese

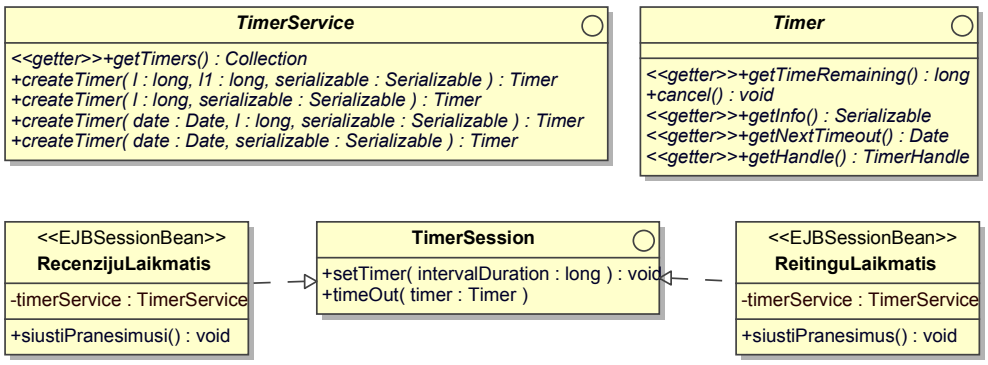
5. Metodikos išbandymas ir įvertinimas

Metodika buvo išbandyta sukuriant publikacijų portalo prototipą, kuriame sistemos vartotojai gali patalpinti savo straipsnius, atlikti straipsnių paiešką, recenzavimą, rašyti komentarus ir vertinti straipsnius. Neregistruotas sistemos vartotojas gali atlikti straipsnių paiešką ir perskaityti jį dominančius straipsnius. Užsiregistravęs – talpinti savo straipsnius, rašyti komentarus ir recenzuoti straipsnius. Aukščiausio lygio (turintys didžiausią reitingą) vartotojai gali atlikti sistemos administravimo darbus: įrašyti naujus temų pavadinimus, įvesti reitingų suteikimo taisyklių koeficientus.

Prototipe generuojami įvykiai, kuriems įvykus, programinis agentas atlieka tam tikrus veiksmus: patalpinus naują straipsnį, sudaro pasiūlymus rašyti recenzijas; praėjus tam tikram periodui, tikrina, ar recenzentai patalpino straipsnių recenzijas, siunčia recenzentams priminimus; tam tikrais laiko momentais perskaičiuoja vartotojų reitingus pagal jų aktyvumą.

Prototipas buvo realizuotas Java EE 5.0 technologija, duomenų prieigai naudojant EntityBean komponentus. Valdikliai kreipiasi ne tiesiogiai į duomenų bazę, bet į EntityBean klasės egzempliorius, kurie atspindi įrašą duomenų bazėje.

Dalį įvykių, kuriuos turi apdoroti programinis agentas. sukuria vartotojų veiksmai, kiti turi būti sugeneruoti automatiškai. Tam panaudojami įvykių generavimo komponentai: „*RecenzijuLaikmatis*“ ir „*ReitinguLaikmatis*“ (11 pav.). Naudojant Java EE technologiją, įvykių generatoriai realizuojami klasėmis, kurios sukuria interfeiso *TimerSession* metodus. Šios klasės nustatytais laiko intervalais generuoja ir siunčia pranešimus, kuriuos apdoroja pranešimų gavėjas – programinis agentas.



11 pav. Laiko įvykių generavimo komponentai

Įvykiais grindžiamų informacinių sistemų kokybines savybes, lyginant su įprastomis objektinėmis sistemomis, galima apibūdinti kriterijais, sudarytais remiantis [8]. Šių kokybinių kriterijų įvertinimas remiantis publikacijų portalo prototipu, realizuotu pagal sudarytą įvykiais grindžiamų sistemų modeliavimo ir realizavimo metodiką, pateikiamas 2 lentelėje.

2 lentelė. Įvykiais grindžiamų sistemų kokybės kriterijų įvertinimas

Sistemos kokybės kriterijus	Kriterijaus įvertinimas įvykiais grindžiamoje sistemoje
Judrumas (angl. *agility*)	Judrumas įvertinamas atsižvelgiant į sistemos lankstumą, palaikomumą, junglumą. Kadangi šitie kriterijai padidėja, todėl galima teigti, jog įvykiais grindžiamų sistemų „judrumas“ taip pat padidėja.
Lankstumas (angl. *flexibility*)	Kadangi įvykiais grindžiamos sistemos kuriamos iš tarpusavyje silpnai susietų komponentų, bet kurio jų pakeitimas yra žymiai paprastesnis, nei objektinėse sistemose, todėl įvykiais grindžiamose sistemose lankstumas padidėja.
Junglumas (angl. *nteroperability*)	Įvykiais grindžiamų sistemų kūrimo metodikoje apibrėžiama, kad tokios sistemos kuriamos iš komponentų, kurie tarpusavyje siejami pranešimų siuntimu. Šis sistemų kūrimo būdas padeda pasiekti didesnį sistemų junglumą, nes pranešimų siuntimu susiejami komponentai yra mažiau priklausomi nei jungiami operacijų kvietimu.
Palaikomumas (angl. *maintainability*)	Įvykiais grindžiamose sistemose komponentai yra mažiau priklausomi vienas nuo kito, todėl vieno komponento pakeitimas nedaro įtakos kitų sistemos komponentų veikimui. Tai leidžia padidinti sistemų palaikomumą.
Pakartotinis panaudojimas (angl. *reusability*)	Kadangi įvykiais grindžiamose sistemose komponentai bendradarbiauja siųsdami pranešimus per tarpinę programinę įrangą ir praktiškai „nežino“ apie vienas kito realizavimo platformą, interfeisus ir t.t., tokiu atveju lengviau padidinti sistemų pakartotinį panaudojimą. Norint pasiekti papildomą funkcionalumą, užtenka nusiųsti atitinkamą pranešimą, kurį komponentas apdoroja, išvengiant problemų dėl komponentų sujungimo bei realizavimo platformų nevienodumo.
Veikimo sparta (angl. *performance*)	Pranešimų siuntimas tarp komponentų yra asinchroninis, todėl įvykiais grindžiamų sistemų veikimo sparta gali sumažėti.
Plečiamumas (angl. *scalability*)	Įvykiais grindžiamose sistemose komponentai pranešimus siunčia per tarpinę programinę įrangą. Padidėjus sistemos apkrovimui, komponentus galima perkelti į kitas platformas (išskaidyti), nes nesvarbu, kuriame serveryje komponentas įdiegtas, svarbu yra tai, kad jis galėtų siusti ir priimti pranešimus. Ši savybė leidžia padidinti sistemų plečiamumą.

6. Išvados

Įvykiais grindžiamų informacinių sistemų kūrimo galimybių analizė parodė, tokios sistemos kuriamos vis dažniau, tačiau kad nėra praktiškos ir išsamios metodikos, kuri nusakytų, kokius įvykių tipus reikia nagrinėti ir kaip juos identifikuoti, modeliuoti bei realizuoti projektavimo procese tam, kad būtų galima gauti lankstesnes, lengviau modifikuojamas sistemas, kurių lankstumas pasiekiamas laisvai susiejant jas sudarančius komponentus per įvykius.

Remiantis atliktu tyrimu, galima teigti, kad įvykius tikslinga klasifikuoti pagal sudarytą taksonomiją, kurioje apibrėžiami tokie pagrindiniai įvykių tipai: sistemos sukeltas įvykis, vartotojo sužadintas įvykis ir laiko įvykis, kadangi kiekvienas iš šių įvykių tipų modeliuojamas ir realizuojamas skirtingai. Sukurtas įvykių metamodelis bei metodika nusako, kaip identifikuoti įvykius iš panaudojimo atvejų jų išplėtimo taškuose; kaip sudaryti įvykių modelį reikalavimų analizės ir specifikacijos etape, bei kaip jį papildyti projektavimo etape, kai suprojektuotos klasės ir komponentai; kaip panaudoti įvykių modelio elementus sekų bei būsenų diagramose, modeliuojant sistemos elgseną.

Portalo prototipo realizacijos analizė parodė, kad įvykių modeliavimo metodika ir jai tinkamos Java EE technologijos įgalina sukurti lankstesnę programinę realizaciją nei įprastu būdu, kai įvykiai neišskiriami kaip reikšmingi projekto elementai. Įvykiais grindžiamos sistemos pasižymi didesniu lankstumu (angl. *flexibility*), junglumu (angl. *interoperability*), plečiamumu (angl. *scalability*) ir pakartotiniu panaudojimu, tačiau tokių sistemų veikimo sparta (angl. *performance*) gali būti mažesnė, nei įprastų objektinių sistemų.

Metodiką tikslinga taikyti kuriant programinius agentus ar sistemas, kurių funkcionavimą veikia ne tik tiesioginiai vartotojo įvykiai, bet ir laiko bei sistemos sužadinami įvykiai, kurių apdorojimą galima automatizuoti. Tokiomis savybėmis pasižymi daugelis sistemų, skirtų kompiuterizuoti veiklos procesus, todėl sukurta metodika gali padidinti daugelio informacinių sistemų kūrimo efektyvumą.

Literatūros sąrašas

[1] Rozsnyai, S., Schiefer, J., Schatten, A.. Concepts and Models for Typing Events for Event-Based Systems, in ACM International Conference Proceeding Series: Proceedings of the 2007 inaugural international conference on Distributed event-based systems, Toronto, Ontario, Canada Vol. 233, 62 – 70, 2007

[2] Wagner, G. A UML Profile for External AOR Models, in Proc. of Agent-Oriented Software Engineering (AOSE) 2002, AAMAS 2002, Bologna, 99-110, 2002.

[3] Odell, J. Modeling-Notation Source: AOR, 2003. The FIPA Agent UML WebSite [žiūrėta 2007.09.15], Prieiga per Internetą: http://www.auml.org/auml/.

[4] Unified Modeling Language. Superstructure Specification Version 2.0. OMG document formal/05-07-04, 2005. Prieiga per Internetą: <http://www.omg.org>

[5] Webopedia. Event definition [žiūrėta 2007.09.15]. Prieiga per Internetą: <http://www.webopedia.com/TERM/e/event.html>

[6] Introduction to Event Listeners [žiūrėta 2007.09.18]. Prieiga per Internetą: <http://java.sun.com/docs/books/tutorial/uiswing/events/intro.html>

[7] How to Use Swing Timers [žiūrėta 2007.09.20]. Prieiga per Internetą: <http://java.sun.com/docs/books/tutorial/uiswing/misc/timer.html>

[8] Gabriel Morgan. Implementing System Quality Attributes. Skyscrapr, Microsoft Coporation, March 2007. [žiūrėta 2007.12.18]. Prieiga per Internetą: http://msdn2.microsoft.com/en-us/library/bb402962.aspx.

METHODOLOGY FOR MODELLING AND IMPLEMENTATION OF EVENT-DRIVEN INFORMATION SYSTEMS

Abstract

The paper is devoted for development of methodology for modelling and implementation of events. Taxonomy and metamodel are presented for representation of events together with related types of objects, messages, senders and receivers. Metamodel is used for identification, definition and implementation of events during the overall process of development of event-driven information systems.