KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Justas Sperauskas

APPFUSE KARKASO TYRIMAS IR TAIKYMAS

Magistro darbas

Darbo vadovas
prof. Eduardas Bareiša

KAUNAS, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Justas Sperauskas

APPFUSE KARKASO TYRIMAS IR TAIKYMAS

Magistro darbas

Recenzentas
doc. D. Rubliauskas
2008 05 26

Vadovas
prof. Eduardas Bareiša
2008 05 26

Atliko
IFM-2/2 gr. stud.
Justas Sperauskas
2008 05 26

KAUNAS, 2008

TURINYS

Analysis and use case of Appfuse

Summary

In this paper we tried to look into J2EE project start and development problem. There are so many development tools to be used in project that causes the complexity. The used tools must be configured to communicate with each other and has different style of configuration. Build, testing, deployment tools must be implemented and m1anaged for every J2EE project. We analyze the solution Appfuse, which eliminates the problems mentioned above. Appfuse is open source project and program, which help fast and efficient software development. The proofs of concepts were illustrated by creating the Information System for Dormitories.

PAVEIKSLIUKŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

1. ĮVADAS

1.1 Dokumento paskirtis

Šis dokumentas yra magistro baigiamojo darbo aprašomasis raštas. Šiame darbe yra visos magistrinio darbo sudėtinės dalys.

1.2 Pagrindimas

Pastaruoju metu informacinių sistemų kūrimas vystosi pagal Lehmano [1] dėsnius:

- Pastovi programinės įrangos kaita - kiekvienas PĮ produktas turi būti atnaujinamas ir tobulinamas, kitaip jis taps vis mažiau ir mažiau tinkantis.
- Didėjantis sudėtingumas - sukūrus produktą, jo sudėtingumas nuolat auga, nebent skiriamas dėmesys jam sumažinti.

Pradedant kurti programinės įrangos produktą sistemos architektas turi svarbią užduotį - parinkti technologijas. Architektai dažnai parenką technologijos tas, kurias geriausiai išmano. Kuriant daugiasluoksnes sistemas naudojami įvairūs jau esami komerciniai arba atviro kodo karkasai, technologijos. Kad ir kokių technologijų derinį jis pasirinktų - priemones tenka apjungti ir nuolat konfigūruoti jų tarpusavio sąryšius. Kuriant didelius, daugiasluoksnius, internetinius projektus reikalingas masyvas įvairiausių programinių servisų: nuo dinaminio puslapio HTML generavimo iki verslo domeno objektų saugojimo. Vietoj to, kad kurtų visus šiuos servisus nuo pradžios, programuotojai naudojasi jau sukurtais ir laisvai prieinamais sprendimais.

Kompanijos Sun J2EE platforma teikia specifikaciją ir bendras sąsajas tiems servisams Java programavimo kalbai. J2EE leidžia programinės įrangos kūrėjams kurti konkurencingus sprendimus. Visiems programinės įrangos kūrimo etapams naudojamos įvairios technologijos. Mes analizuosime etapus nuo programinės įrangos infrastruktūros sukūrimo iki jo galutinio pristatymo. J2EE projekto kūrimo pradžioje daugelis programuotojų susiduria su komponentų ir įvairių modulių integracijos problema. Analizuojama technologija Appfuse yra skirta palengvinti programuotojų ir architektų darbą. Šis karkasas yra junginys daugelio technologijų ir atsakingas už jų sąveiką, darbo automatizavimą, lengvą programinės įrangos kūrimą ant sukurto maketo. Mes sukūrėme internetinę Bendrabučių informacinė sistemą. Naudojome Appfuse architektūros maketą su Struts2, Spring, Hibernate kaip pagrindinėmis technologijomis.

Įvairiuose šaltiniuose teigiama, kad pradedant naują J2EE projektą būtina naudoti karkasus ir taip pat akcentuojama, jog juos sukonfigūruoti yra sudėtinga. Mes kaip sprendimą nurodėme Appfuse ir pademonstravome jo panaudojimą.

1.3 Tikslas

Ištirti ir pritaikyti karkasą Appfuse internetinio projekto kūrimui ir įvertinti karkaso teikiamus privalumus bei trūkumus.

1.4 Uždaviniai

Darbo uždaviniai yra tiesiogiai išvedami iš projekto tikslo:

1. Ištirti iš aprašyti J2EE projektų kūrimo problematiką dėl įvairių technologijų panaudojimo kuriant programinę įrangą.
2. Išanalizuoti karkaso Appfuse paskirtį ir panaudojimo galimybes.
3. Ištirti ir aprašyti vienos Appfuse diegiamos architektūros technologijas ir jų sąveiką.
4. Sukurti projektą naudojantis Appfuse karkasu.
5. Atlikti sukurtos PĮ kokybės tyrimą.

1.5 Santrauka

Šiame darbe mes įvardinome J2EE projekto kūrimo problematiką. Ištyrėme įrankį Appfuse, kuris yra skirtas programuotojo darbui palengvinti. Appfuse atlieka J2EE internetinio projekto infrastruktūros generavimą, kompiliavimą, veiksmų automatizavimą ir kitas naudingas užduotis. Šį įrankį panaudojome Bendrabučių informacinei sistemai sukurti ir taip įsitikinti Appfuse nauda. Projektas buvo pradėtas greitai ir kodo rėmai leido programuoti šiuolaikiškai pagal trijų lygių architektūros schemą. Nereikėjo gilintis į sudėtingų technologijų, tokių kaip Spring ir Hibernate integravimą. Dokumente aprašėme naudojamos priemonės suteikiamus privalumus ir panaudojimą tyrimą.

1.6 Dokumento struktūra

Skyrius 1: Įvadas. Šiame skyriuje pateikiamas dokumento kontekstas.

Skyrius 2: Analitinė dalis. Šiame skyriuje mes apžvelgiame esamas šiuolaikinės technologijas. Pateikiama literatūros apžvalga susijusi su J2EE projektavimu. Išanalizuojamos kuriamos sistemos problemos ir galimi sprendimo būdai.

Skyrius 3: Projektinė dalis. Šiame skyriuje pateiktas realizuotos sistemos aprašymas. Išdėstomi pagrindiniai architektūros aspektai.

Skyrius 4: Tyrimo ir eksperimentinė dalis. Šiame skyriuje atliekamas Appfuse karkaso galimybių tyrimas. Tyrimas atliekamas analizuojant sukurtą projektą.

2. ANALITINĖ DALIS

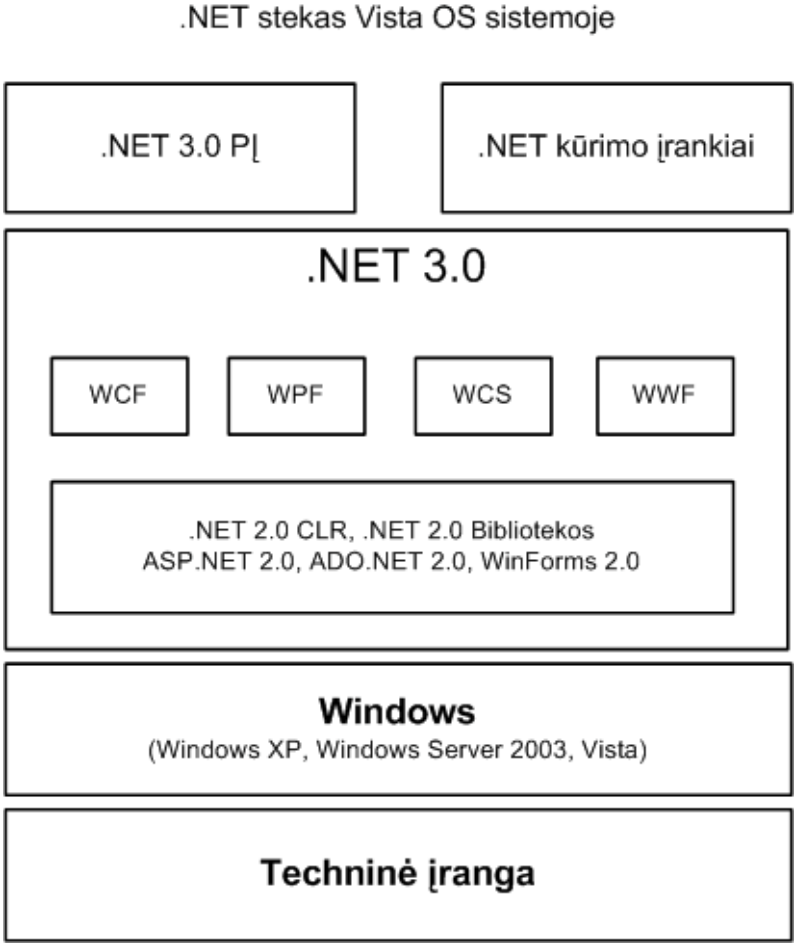
2.1 Internetinių sistemų technologijos

Interneto naršyklė vis dažniau naudojama kaip įrankis darbui su centralizuota sistema. Su tokiomis technologijomis kaip Java, JavaScript, Flash ir kitomis įgalina piešimą, garsą, prieiga prie klaviatūros ir pelės. Internetinių technologijų kūrėjai dažnai prideda kliento pusės programavimo (tokių kaip JavaScript) funkcionalumui praplėsti. Toliau apžvelgsime tris pagrindines internetinių sistemų programavimo technologijas.

2.1.1 .NET platforma

2002 metų sausį buvo pateiktas „.NET Framework“, kaip atskiras komponentas prie operacinės sistemos arba sistemos kūrimo įrankiuose, taip pat buvo pateikimas kartu su Microsoft Visual Studio .NET (tuo metu tai buvo Visual Studio .NET 2002).

Microsoft.NET [2] yra produktas naudojamas kurti intelektualias ir verslo lygio internetiniams produktams. Galima apibrėžkime skirtumus su J2EE technologija taip: .NET yra strategija, o J2EE yra standartas kuriuo laikantis produktai yra kuriami. .NET yra perrašytas Windows DNA (angl. Windows Distributed interNet Architecture), kuris buvo skirtas kurti verslo klasės programoms. .NET karkasas pakeitė šią technologiją pridėdamas internetinius servisus ir pagerindamas programavimo kalbų palaikymą (1 pav.). Reikėtų paminėti, kad .NET platforma palaiko visas programavimo kalbas išskyrus Java. .NET platforma naudoja ODBC programavimo sąsaja prisijungimui prie duomenų bazių (RDBMS). Naudojant ODBC galima kurti programas, kurios nekeičiant kodo ar keičiant tik nežymiai leidžia naudotis skirtingomis duomenų bazių valdymo sistemomis. Norint per ODBC interfeisą naudotis tam tikra duomenų baze, jai turi būti parašyta ODBC tvarkyklė.



1 pav. .NET 3.0 architektūra

Ši platforma suteikia kitoms programoms galimybę naudotis daugybe jau paruoštu įvairių bibliotekų (pvz., duomenų bazių komponentus, formų komponentus ir kt.). Negalima nepastebėti, kad šis projektas yra pilnai pritaikytas ir palaikomas tik Windows OS platformoms. Tai pagrindinis skirtumas nuo Java platformos, kuri yra visiškai multiplatforminė.

2.1.2 JAVA (J2EE) platforma

J2EE sukūrė Sun Microsystems, tačiau esama ir alternatyvių šios sistemos išpildymų. Tai reiškia, kad J2EE specifikaciją sukūrė ši kompanija, todėl ji gali patvirtinti ar sukurtas produktas gali būti vadinamas kaip šios technologijos išpildymas. Teisingai parašyti J2EE moduliai veikia nepriklausomai nuo to, kokio gamintojo J2EE supanti sistema naudojama. Ši standartinė daugialypių programų kūrimo (Java platformos dalis) platforma, paremta moduliniais komponentais, vykdomais programų serveryje. Ji kurta su tikslu supaprastinti komplikuotus kūrimo, diegimo ir priežiūros procesus daugialypiams sprendimams.

Ši architektūra yra pagrįsta Java programavimo kalba. Nuostabiausias dalykas šioje programavimo kalboje yra tai, kad su Java sukurtas kodas vieną kartą, gali buti diegiamas į bet kurią operacinę sistemą. Trumpas kūrimo aprašymas:

1. Programuotojas parašo kodą su Java.

2. Java kodas kompiliuojamas į baitkodą, kuris yra multiplatforminis ir tarpinis tarp programos kodo ir mašininio kodo.
3. Tada kodas yra paruoštas paleidimui su Java virtualia mašina (angl. the Java Runtime Environment (JRE))

Viena iš pačių svarbiausių sričių šiuolaikinėje programų inžinerijoje – duomenų valdymas, kuriuo remiasi dauguma šiuolaikinių informacinių sistemų. Java turi JDBC (Java DataBase Connectivity) biblioteką, kuri yra skirta reliacinių duomenų bazių valdymui pagrįstam SQL kalba ir nepriklausomam nuo duomenų bazės sistemos tiekėjo (Oracle, SQL Server, DB2 ir kt.). JDBC apibrėžia sąsajas, kurių realizacijos turi būti pateikiamos duomenų bazių sistemos JDBC tvarkyklėse (angl. drivers).

2.1.3 PHP platforma

PHP – plačiai paplitusi dinaminė interpretuojama programavimo kalba, sukurta 1997 m. ir specialiai pritaikyta interneto svetainių kūrimui [3].

PHP sintaksė panaši į daugelį struktūrinių kalbų, ypatingai į C bei Perl. PHP kalba yra atviro kodo ir tai yra viena priežasčių, dėl ko kalba yra nors ir nesudėtinga, bet gana lanksti – veikia daugumoje operacinių sistemų, palaiko nemažai reliacinių duomenų bazių bei veikia su dauguma interneto serverių – CGI, FastCGI, ISAPI ir kitais protokolais.

Nors ir PHP yra dažniausiai naudojama interneto puslapių kūrimui, PHP yra labai galingas įrankis atlikti kitas funkcijas komandinėje eilutėje.

2.1.4 J2EE, ASP.NET, PHP palyginimas

Šiame skyriuje bus aptariamos trys populiariausios internetinių projektų kūrimo platformos. Aprašomi pliusai ir minusai šių platformų pagal patyrusio programuotojo internetinį įrašą ir komentarus [4].

Pagal sintaksės kriterijus Java kalba yra maloniausia ir labiausiai objektiškai orientuota. PHP turi įvairių sunkiai skaitomų simbolių bei funkcijų pavadinimų ir daugiau panaši į procedūrinio programavimo kalbą. ASP.NET galima rašyti įvairiomis kalbomis, bet jei naudojant VB.NET tai yra visai maloni kalba.

Kalbos išmokimo sudėtingumas taip pat yra svarbus kriterijus ir jis geriausiai vertinamas su PHP kalba, nes ją labai paprasta išmokti. PHP turi daug pradžiamokslio dokumentacijos. ASP.NET taip pat yra gana nesunkiai išmokstama, o J2EE yra sudėtingiausia pagal šį kriterijų.

Multiplatformiškumas yra svarbus, bet ne visada reikalingas. J2EE ir PHP projektai gali veikti ant įvairiausių operacinių sistemų, o ASP.NET tik su Microsoft operacinėmis sistemomis.

Duomenų bazių priėjimai yra išvystyti labai gerai su visomis platformomis. Integruotų programavimo aplinkų yra apstu J2EE sistemoms, jų netrūksta ir ASP.NET, bet PHP šių aplinkų kiekiu negali pasigirti. J2EE turi daugiausia nemokamų bei keletą mokamų programavimo aplinkų Eclipse, IntelliJ, Netbeans, Sun Studio ir t.t.. ASP.NET turi labai gerą Visual Studio įrankį, kuris yra labai gerai vertinamas. Vertinant pagal objektiškai orientuotą programavimą J2EE yra geriausia, o PHP silpniausia. Saugumo klausimu taip pat pasižymi J2EE, nes PHP ir ASP.NET dažniausiai yra nulaužiamos ir turi įvairių klaidų. ASP.NET veikia tik ant Windows sistemų, todėl tai susiję ir su operacinės sistemos saugumo problemomis.

ASP.NET turi mažiausiai programavimo bibliotekų ir karkasų, o PHP bei J2EE turi jų apsčiai. Bendruomenės gausumu ir palaikymų gali pasigirti visos šios aptariamos platformos, nes internete yra labai daug forumų, grupių ir programuotojų, kur galima spręsti iškilusias problemas.

Kainos atžvilgiu galima išskirti ASP.NET, nes ji yra komercinė ir mokama platforma, o J2EE ir PHP yra atviro kodo ir turi daug nemokamų įrankių.

2.1.5 Internetinių platformų apibendrinimas

Aprašytos trys platformos [4] yra pačios populiariausios internetiniams projektams (taip pat būtina paminėti „Ruby on Rails“, Django ir kt.) ir jos turi privalumų bei trūkumų vienos prieš kitas. Platforma turi būti parenkama pagal projekto funkcijas, dydį, apimtį, bei pagal išanalizuotus kriterijus. Nebūtų galima priimti išvadų, kuri platforma yra geriausia, nes tai priklauso ir nuo projekto bei žmonių.

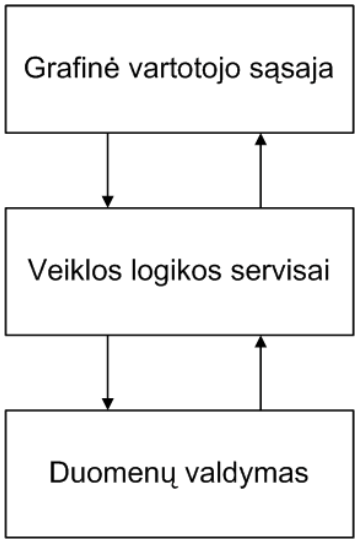
Nors dauguma internetinių svetainių yra kuriamos su PHP arba mod_perl technologijomis, yra labai didelis pasirinkimas visų platformų internetinių sistemų karkasų (angl. Web application frameworks), kurios automatizuoja procesus, leisdamos programuotojui dirbti su aukšto lygio aprašymais. Naudojimas minėtų karkasinių sistemų dažniausiai sumažina klaidų buvimą sistemoje, dėl kodo paprastumo ir susikoncentravimo prie šios karkasinės technologijos. Tačiau sistemos yra pastoviame nulaužimo pavojuje, todėl saugumas yra vienas iš svarbiausių uždavinių.

Šiuo metu Java technologijos tampa vis populiaresnės kuriant internetines sistemas. Tai ypač pastebima verslo sistemoms. Tam skirtos J2EE Java programavimo platforma su komponentais (angl. JavaServer Pages, servlets, client-side applets, Enterprise Java Beans , JDBC ir kt.). Taip pat .NET platforma yra dažniausiai naudojama verslo sistemoms kurti, kai pasirinkta aplinka yra Microsoft.

2.2 J2EE projektų technologijos

Kuriant šiuolaikinius konkurencingus J2EE projektus yra svarbu kurti ekonomiškai ir efektyviai tiek kiek įmanoma. Vienas iš būdų padidinti efektyvumą yra kompleksiškumo sumažinimas. Naudojant šiuolaikinius atviro kodo karkasus kompleksiškumas yra gana didelis. Todėl reikia sprendimo, kuris atliktų dalį kompleksiško darbo automatiškai, leistų laikytis priimtų standartų, bei sumažintų klaidų tikimybę. Todėl šiame skyriuje aprašysime šiuolaikinius karkasus naudojamus J2EE projektuose. Organizacijų lygio programinė įranga kuriama naudojant daugiasluoksnę trijų lygių architektūra (angl. Three-tier architecture).

Yra daug gyvenimiškų principų, kurie puikiai tinka projektuojant programinę įrangą. Vienas iš tokių — dar romėnų imperijos kūrimo laikais taikytas „skaldyk ir valdyk“ [6]. Bet kurią sudėtingesnę programinės įrangos sistemą reikia suskaidyti į dalis, kad būtų galima efektyviai padalinti kūrimą grupei žmonių. Čia labai svarbu tiksliai apibrėžti tų dalių bendravimo principus ir sąsajas. Vienas iš plačiausiai taikomų sprendimų — sluoksniuotos posistemių architektūra, kurioje bendravimas tarp sluoksnių yra griežtai ribojamas hierarchijos — aukščiausio lygio sluoksnis bendrauja tik su po juo esančiu sluoksniu, šis su dar žemesniu ir t.t. Nėra leidžiamas bendravimas iš žemesnių sluoksnių į aukštesnius arba „peršokant“ tarpinius sluoksnius. Toks sistemų organizavimas labai palengvina sistemų palaikomumą. Sluoksniuota architektūra (2 pav.) dažnai jungiama su MVC (angl. Model-View-Controller) šablonu, kuris siūlo skaidyti programinę įrangą į vartotojo sąsajos, duomenų modelio ir veiklos logikos dalis.



2 pav. Sluoksniuota trijų lygių architektūra pagal MVC projektavimo šabloną

Sluoksniuotos architektūros, pagrįstos MVC šablonu, schema puikiai tinka taikyti daugumoje programinės įrangos sistemų, kadangi tiek vartotojo sąsaja, tiek duomenys yra praktiškai bet kurioje programinėje įrangoje, o didesnėse verslo optimizavimui skirtose sistemose dažniausiai yra pakankamai sudėtinga veiklos logika, kuri gali kisti priklausomai nuo įmonės strategijos, todėl ją verta išskirti kaip atskirą modulį [5]. Toks organizavimas leidžia realizuoti nesunkiai praplečiamą ir palaikomą programinę įrangą bei lengvai paskirstyti darbus.

Šiuo metu egzistuoja labai daug jau sukurtų karkasų vienam ar keliems šios architektūros sluoksniams realizuoti. Pavyzdžiui duomenų valdymo lygiui yra tokie sukurti J2EE atviro kodo karkasai: Hibernate, iBatis, jPersist, JDBCPersistence, OJB [7]. Šie karkasai yra duomenų valdymo lygio realizacija ir atlieką duomenų perkėlimą iš RDBMS į objektiškai orientuotus kalbos objektus. Naudojantis šiais karkasais nereikia rūpintis kokią duomenų bazės sistema bus panaudotą, nes karkasas gali ir moka dirbi su dauguma populiariausių RDBMS. Duomenų struktūroms aprašyti naudojama XML kalba, o manipuliacija su duomenimis, kuri yra neintuitivi ir neįkelta į karkasą, yra aprašoma universalia kalba panašia į SQL. Pavyzdžiui, Hibernate karkase ši kalba yra HQL ir nepriklausomai nuo RDBMS gali sėkmingai manipuliuoti duomenimis, bei teikti servisus aukštesniems lygiams, šiuo atveju veiklos logikos servisų lygiui [8].

Grafinės vartotojo sąsajos ir veiklos logikos servisų lygiams yra bendri karkasai tokie kaip Struts2, Spring MVC, JSF, Cocoon, Apache Velocity, WebWork [9]. Šie karkasai yra paminėtų lygių realizacijos ir naudojasi duomenų valdymo lygio mechanizmais. Karkasai palengvina, supaprastina šių lygių logikos programavimą. Pavyzdžiui, Struts2 įveda specialius tagus (angl. custom-tags), kurie palengvina vartotojo sąsajos programavimą, t.y. lentelių formavimas, rikiavimas, duomenų formos, datų įvedimas ir t.t.. O veiksmai tarp grafinės sąsajos ir veiklos logikos konfigūruojami XML kalba. Veiklos logika Struts2 karkase programuojama pagal Servlet specifikaciją su įvairiais patobulinimais t.y. įvedimo ir rezultatų logiką aprašant XML failuose, bei naudojant Spring karkaso teikiamus privalumus [10].

2.2.1 J2EE projekto parengimas

Projekto parengimas apima daug darbo su klientais ir dokumentacija. Šiame darbe nėra analizuojama reikalavimų išgavimo problematika ar projekto valdymo subtilumai. Šiame darbe pateikiamas karkasas, kuris palengviną techninį darbą. Dažnai įmonėse dirba techniniai projektų vadovai, kurie atsakingi už architektūros parinkimą pagal projekto poreikius. Taip pat jie atsakingi už technologijų suintegravimą ir stabilų jų veikimą, bei optimalų komunikavimą tarp skirtingų karkasų. Naudojant tokį įrankį kaip Appfuse visi šie darbai yra padaromi automatiškai, tik lieka išsirinkti architektūrą iš siūlomų ir peržiūrėti ar ji atitinka visus reikalavimus kuriamam projektui. Darbe analizuojami tik internetiniai projektai kuriami su J2EE platformai. Kadangi internetiniai projektai kuriami ne vieną dešimtmetį, todėl jau projektams yra nusistovėję tipiniai reikalavimai jų funkcionalumui. Dauguma šiuolaikinių projektų yra kuriami daugiasluoksnės architektūros, daugiakalbiai, su duomenų baze, vartotojų registracijomis bei autorizacija ir saugumu, taip pat įvedamų duomenų tikrinimų. Visą paminėtą funkcionalumą ir daugiau gali

suteikti Appfuse karkaso sudaromiems projektams. Todėl galima teigti, kad yra tik maža dalis internetinių J2EE projektų, kuriems Appfuse neteiktų privalumų.

2.2.2 J2EE projekto vykdymas

Projekto vykdymas techniškai gali būti pradėtas nuo jau užbaigtos specifikacijos arba dar jos neturint. Tai yra projekto vadovo atsakomybė parinkti projekto vykdymo metodologiją, ar bus vykdoma krioklio metodu ar iteracinių, ar kitais Agile principais paremtais metodais.

Programa karkasas yra masyvas klasių, kuris įgyvendina perpanaudojamą dizainą programai arba dažniau vienam programinės įrangos lygiui [11]. Kur programos kodas kviečia bibliotekos klasę atlikti vienam ar kitam platformos servisui, o karkasas kviečia programos kodą ir taip valdo vykdymo seką. Programuotojas rašo kodą karkasui, kad jį iškviestų vykdymo metu.

Viename straipsnyje minimi dešimt didžiausių pavojų kuriant J2EE projektą [12] :

- Nemokėjimas Java, nesupratimas J2EE
- Perprojektavimas
- Neatsiejimas prezentacijos lygio nuo biznio
- Programinė įranga nepaleidinėjama kol kuriama
- Blogas pasirinkimas įrankių
- Nežinojimas pasirinktų įrankių
- Projektavimas nekreikiant dėmesio į apkrovą ir išplėčiamumą
- Senoviškas kūrimo procesas
- Nenaudojimas karkasų
- Projekto plano grindimas pagal rinką, o ne pagal techninius faktus

2.2.3 J2EE projektų kūrimo užduočių vykdymo įrankiai

Projekto parengimas veikimui nėra paprasta užduotis. Šis darbas reikalauja griežtai apibrėžtų veiksmų su kompiliatoriais ir rezultatų failais. Projekte dažnai būna daug failų ir kompiliavimas reikalingas su skirtingais kompiliatoriais. Taip pat reikalingas ir duomenų bazės paruošimas. Šiuos veiksmus J2EE projektams geriausiai atlieka parengimo karkasai (pvz.: ANT, MAVEN2). Šie karkasai atlieka aprašytus veiksmus tam, kad gautume vienokį ar kitokį rezultatą. Veiksmai gali būti tokie kaip kodo kompiliavimas, projekto testavimas, duomenų paruošimas ir kt..

ANT – atviro kodo programavimo įrankis sudėtingoms programoms iš išeities kodo „pagaminti“. ANT reikalingas, kai naudojamas ne vienas kompiliatorius, dalis išeities kodo bylų turi būti generuojama automatiškai, programą norima iš karto automatiškai testuoti ir kitais sudėtingais atvejais. Ant turi daug modulių įvairioms užduotims atlikti; prireikus nesunku parašyti ir savo specifinių [13] . Ant užduotis rašoma XML.

MAVEN - taip pat atviro kodo Java projektų valdymo ir gamybos įrankis. Yra panašus į ANT įrankį, bet turi paprastesnį konfigūravimo modelį, rašomą XML. Pagrindinė Maven raktinė funkcija yra tinklinis pasirengimas. Tai reiškia, kad Maven gali parsisiųsti priedus iš įvairių Java atviro kodo ar kitų organizacijų prieigų per tinklą. Dėl šio privalumo šis karkasas tampa svarbiausiu Java projekto rengimo įrankių.

2.3 Projektų kūrimo įrankis Appfuse

Su Appuse J2EE projektas gali būti pradėtas labai greitai. Programuotojas pasirenką vieną iš Appfuse pasiūlytų architektūrų ir vienos komandos pagalba sugeneruoja projekto infrastruktūrą. Galimos architektūros yra:

- JSF Basic
- Spring MVC Basic
- Struts 2 Basic
- Tapestry Basic
- JSF Modular
- Spring MVC Modular
- Struts 2 Modular
- Tapestry Modular
- Core

Projekto infrastruktūra sugeneruojama vienos komandos pagalba: mvn archetype:create.

Kitos komandos pateikiamos lentelėje (1 lentelė).

1 lentelė. Appfuse mvn komandos

Komanda	**Aprašymas**
`mvn appfuse:full-source`	Parsiunčia ir įdiegia pilną PĮ išeities kodą
`mvn appfuse:gen -Dentity=Name`	Generuoja testų, logikos klasių, grafinės sąsajos, duomenų bazės failus iš objektinės

	JAVA klasės
`mvn appfuse:gen-model`	Generuoja testų, logikos klasių, grafinės sąsajos, objektines klases iš duomenų bazės
`mvn jetty:run-war`	Įdiegia duomenų bazę, praleidžia testus ir įdiegią programinę įrangą
`mvn install eclipse:eclipse`	Sukuria projekto failus pasirinktai integruotai programavimo aplinkai
`mvn install`	Atlieka visus testus ir sukuria instaliacinį failą

2.3.1 Appfuse naudojami karkasai

Appfuse naudoja atviro kodo laisvai prieinamus jau sukurtus karkasus ir įrankius. Šių priemonių sąrašas yra gausus:

- Acegi Security
- Acegi JSF Components
- ASM
- AspectJ
- Clickstream
- Commons Lang
- Commons Logging
- Commons BeanUtils
- Commons Collections
- Commons DBCP
- Commons FileUpload
- Commons IO
- Core JSF Validator
- Display Tag
- DWR
- EhCache

- Facelets
- Hibernate
- iBATIS
- JavaMail
- jMock
- JSTL
- JUnit
- Log4J
- MyFaces
- Jakarta ORO
- OSCache
- Scriptaculous
- SiteMesh
- Spring
- Spring Modules
- Struts
- Struts Menu
- Tapestry
- The DHTML Calendar
- URL Rewrite Filter
- Velocity
- WebTest
- Wiser
- XFire

Iš šio sąrašo matome, kad daugumą įrankių yra panaudojami daugelyje projektų. Tokie įrankiai kaip kalendorius, vartotojų prisijungimas ir t.t. yra naudojami beveik visuose internetiniuose projektuose, todėl šiuolaikiniai programuotojai jų nekuria nuo nulio, o pasiima jau sukurtus, laisvai prieinamus.

2.3.2 Klaidų ir išimčių apdorojimas

Bet kokios stambesnės programos kūrimo proceso didelę dalį sudaro klaidų ir išimčių apdorojimas. Vien šis skyrius nusipelno atskiro straipsnio, kuriame būtų aprašyti visi apdorojimo niuansai.

Teoriškai bet kokio didelio proceso loginė seka veiks teisingai tik su labai maža aibe duomenų, kurie yra „taisyklingi“. Bet kokie „netaisyklingi“ duomenys gali nutraukti visą susietų procesų grandinę. Iš būdo, kaip apdorojamos išimtys, įspėjimai ir klaidos, galima atskirti puikią programinę įrangą nuo niekam tikusios.

Iš savo patirties galiu pasakyti, kad išimtis geriausia apdoroti kaip įmanoma aukštesniame lygyje. Žmogus, besikreipiantis dėl tam tikros specializacijos programos, gali nieko nesuprasti iš sistemos gilumo atėjusio pranešimo, tačiau paprastas pranešimas, kad sistemoje įvyko klaida, bus daug prasmingesnis.

Daugybė Java išplėstinės API metodų apdoroja įvairias išimtis, kurios visos yra bendros klasės Exception poklasiai. Keletas direktyvų gali padėti geriau apdoroti išimtinius atvejus:

1. Išimčių persiuntimas. Vienas kitame esančių metodų kvietimo atveju, logiškiau yra persiųsti išimtį kvietėjui, jei kviečiamasis negali apdoroti išimties. Tarkim, funkcija turi atidaryti konfigūracijos failą ir grąžinti jo turinį. Jei dėl kokios nors priežastis ji negali surasti failo, tai ji gaus FileNotFoundException (išimtis, kai failas nerastas). Dabar ji gali padaryti du dalykus: jei kvietėjui nerūpi sužinoti apie klaidą, ji gali grąžinti tuščią eilutę. Arba ji gali persiųsti išimtį kvietėjui, iškviesdama ją atvirai.

2. Išimčių apvilkimas. Kaip ir išimčių persiuntimo atveju, tačiau tam tikrų išimčių kiekis (kurios yra Exception poklasiai) gali būti įvilktas į nuo taikomosios programos priklausančią specialią Exception klasę. Pvz., vietoj to, kad būtų persiunčiamos FileNotFoundException arba IOException į iškvietusįjį metodą, jos gali būti įvelkamos į specialią išimčių klasę ir tada persiunčiamos kvietėjui.

2.3.3 Dokumentavimas

Appfuse projektas yra aktyvus ir šiuo metu, turi plačią dokumentaciją. Jis yra atvirai prieinamas, bei tobulinamas aktyvistų ir viso pasaulio. Bendruomenė yra plati, nes Appfuse naudojamas tiek verslo, tiek kituose sektoriuose. Pats projekto įkūrėjas Matt Raible teigia [5], kad bendruomenė yra draugiška ir prisideda prie produkto vystymo. Daug populiarų sistemų yra sukurtų su šiuo karkasu (2 lentelė).

Dokumentacija galima rasti net šešiomis kalbomis: kinų, vokiečių, anglų, korėjiečių, portugalų ir ispanų. O technologijos kurios įeina į Appfuse įrankį taip pat turi savo nepriklausomas bendruomenes ir dokumentacijas.

2 lentelė. Projektai sukurti su Appfuse įrankiu

Pavadinimas	Adresas	Aprašymas
Booking Booster	www.bookingbooster.com	Sistema skirta rezervuoti viešbučiams
Crystal Hotels	www.crystalhotels.net	Londono viešbučių tinklo svetainė
Dive Worldwide	www.diveworldwide.com	Nardymo paslaugų svetainė
ExpressPlanner	www.expressplanner.com	Pirma nemokama pasaulyje įvykių registravimo sistema
forGetaway.com	forgetaway.com	Weather.com puslapis
Go2Nice France	www.go2nice.net	Sistema skirta rezervuoti viešbučiams
greendimes.com	greendimes.com	Kovos su nepageidaujamais laiškais projektas
HTTP Scoop	tuffcode.com	Programinės įrangos svetainė
Internet Polyglot	www.internetpolyglot.com	Nemokami kalbos kursai
LA Lakers Courtside Connection	my.lakers.com	LA Lakers komandos fanų klubas
myJazz Connection	my.utahjazz.com	Utah Jazz komandos fanų klubas
Rivercats Fan Loyalty Portal	getinthegame.rivercats.com	Sacramento Rivercats komandos fanų klubas
SourceBeat	sourcebeat.com	Skaitmeninių ir spausdintų knygų publikavimas
USA Soccer Stars.com	usasoccerstars.com	Puslapis remiantis JAV futbolo turnyrą
Walks Worldwide	www.walksworldwide.com	Pramogų tiekėjo sistema
Wildlife Worldwide	www.wildlifeworldwide.com	Atostogų planuotojo sistema

2.4 Programinės įrangos kūrimo metodologija

2.4.1 Iteracinis modelis

Iteracinis modelis yra programinės įrangos kūrimo modelis, kuris eliminuoja tradicinio Krioklio modelio trūkumus [14]. Pagrindinė Iteracinio modelio idėja yra sistemos kūrimas priauginant. Tai leidžia kūrėjams naudotis tuo ką jie išmoko kurdami ankstesnes sistemos versijas.

Iteracinio modelio projektas nebandomas pradėti su išbaigta reikalavimų specifikacija [15]. Todėl projektas pradedamas kurti su turima dalimi reikalavimų. Iš gautos sistemos galima išgauti naujus reikalavimus. Tada toks procesas kartojamas didinant funkcionalumą ir gaunama nauja sistemos versija.

Nagrinėjant Iteracinį modelį, kuris susideda iš pasikartojančių (3 pav.) keturių fazių:

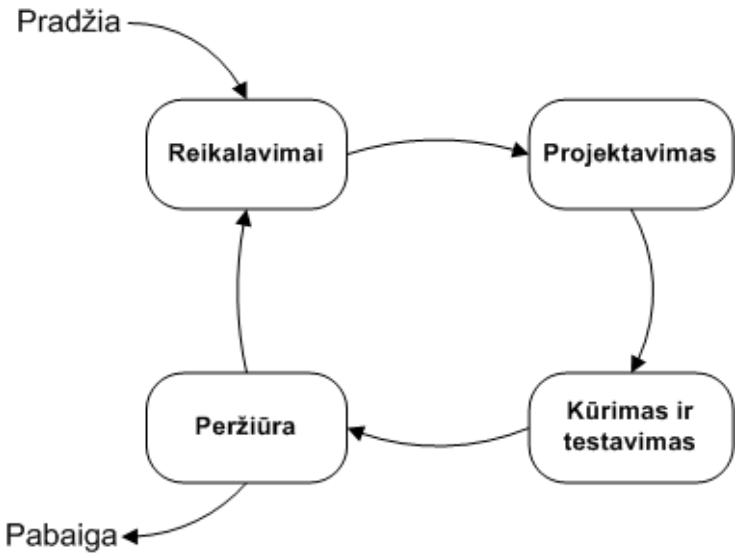
Reikalavimų fazė, kurios metu reikalavimai programinei įrangai yra renkami ir analizuojami. Iteracijos rezultatas lemia reikalavimų fazę, kuri gauna išbaigtą ir galutinę

reikalavimų specifikaciją.

Projektavimo fazė, kurioje turi būti suprojektuotas sistemos sprendimas tenkinantis reikalavimus.

Kūrimo ir testavimo fazėje programinė įranga yra programuojama, integruojama ir testuojama.

Peržiūros fazės programinė įranga yra įvertinama, peržiūrimi esami reikalavimai. Pasikeitimai ir nauji reikalavimai yra pridedami.



3 pav. Iteracinio modelio procesai

2.4.2 Iteracinio modelio argumentai

Vartotojas turi būti įtraukas į projektą ir aktyviai veikti. Šis įtraukimas yra pozityvus, bet reikalaujantis laiko, o tai gali įtakoti projekto užsitęsimą. Komunikavimo ir koordinavimo praktika yra centrinė ašis projekto kūrimo procese. Neformalūs poreikiai pakeitimams gali sukelti painiavą, todėl tokių poreikių atsiradimas turi būti kontroliuojamas ir apdorojamas. Iteracinis modelis gali sukelti apimties išaugimą, nes vartotojas kiekvienos fazės metu pateikia vis naujus reikalavimus. Vartotojas kūrimo metu gali suvokti ir kitų sistemų galimybių naudą jo darbo pagerinimui.

2.4.3 Agile modelis

Šiuo metu vis daugiau kalbama apie šiuolaikiškus ar netgi ekstremalius modelius. Vienas iš tokių būtų Agile metodai. Agile metodai teigia [16], kad kuriant programinę įrangą dalinai veikiančios programos yra efektyvesnė reikalavimų išsiaiškinimui nei išsami dokumentacija, gyvas bendravimas su vartotoju yra geriausia komunikavimo priemonė, o pasikeitimai yra neišvengimai ir juos įvertinti reikia vystant programinę įrangą trumpomis iteracijomis.

Naujas Agile požiūris [17], skatinantis nevengti pasikeitimų ir juos įvertinti kuriant programinę įrangą mažomis iteracijos, kurių metu įgyvendinamas funkcionalumas iškart aptariamas su vartotojais.

Grupė žymių programinės įrangos metodologų pasirašė Agile manifestą [18], kuriame paskelbė, kad labiau vertina:

- žmones ir bendravimą nei procesus ir įrankius,
- veikiančią programinę įrangą nei išsamią dokumentaciją,
- bendradarbiavimą su užsakovais nei kontrakto pasirašymą,
- reagavimą į pakeitimus nei plano vykdymą.

Agile metodų vertybės[19] :

Drąsumas, narsa: pasirinkus Agile modelį reikalaujama drąsos ir narsumo. Nes tu esi privestas pasitikėti savimi ir savo patirtimi, negu griežtai laikytis standartų.

Komunikavimas: modelyje svarbiausia komunikavimas tarp užsakovo ir kūrėjo.

Atsiliepimai: kiek įmanoma ankščiau ir dažniau gauti atsiliepimus.

Nuolankumas: kartais gali tekti atsisakyti gerų dalykų tik dėl to kad užsakovas jų nesupranta.

Paprastumas: stengiamasi naudoti kuo paprastesnius dalykus.

2.4.4 Agile modelio argumentai

Agile procesai siūlo kuo mažiau dėmesio skirti dokumentavimui. Programinė įrangą kuriama trumpomis iteracijomis, o vartotojo poreikiai išsiaiškinami bendraujant tiesiogiai. Tarpinių programinės įrangos versijų vertinimas atliekamas kiekvienos iteracijos pabaigoje. Naujas Agile požiūris daugiau tinkamas nedideliems projektams, kurie gali būti vykdomi ir finansuojami pagal iteracinį gyvavimo ciklą [20].

Nors Agile metodai beveik idealiai tinka produktams, vidiniams projektams ir atviro kodo sistemoms kurti ir jie reikalauja visiškai kitokio programinės įrangos kūrimo proceso, jo planavimo, valdymo ir finansavimo, tačiau užsakomiems projektams jų tiesiogiai taikyti neverta, nes nėra tenkinamos dauguma potencialios sėkmės sąlygų.

2.4.5 Neiteracinių modelių kritika

Trumpai apžvelgsime neiteracinių modelių (tokių kaip Krioklio) kriticizmą. Daugelis kylančių problemų yra susijusių su praėjusiais kūrimo etapais. ([21] „Many problems in software development can be traced back to the uncertainty in the planning phase“).

Projektai turi būti atviri pakeitimams, kadangi užsakovas gali pakeisti reikalavimus. Specifikuojant kartais žmonės nežino ko nori kol neprasideda kiti etapai. Kartais užsakovai tik

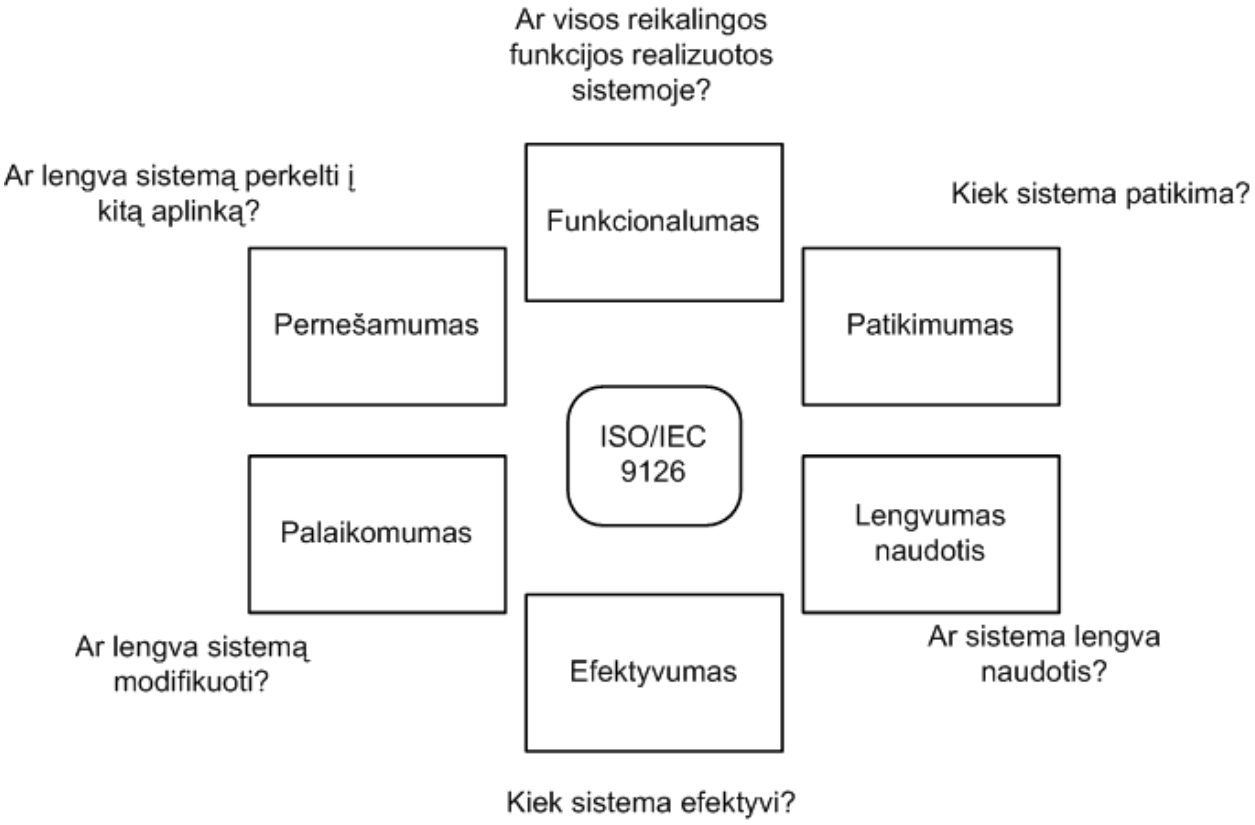
vėlesniuose etapuose supranta, kad ne to norėjo ir kūrėjams tenka sugrįžti. Pavyzdžiui reikalingas diegimo fazės rezultatas tam, kad būtų identifikuotos projektavimo fazės reikalavimai. Sistemos kūrimas ir pastangų dėjimas projektams tikint, kad jų reikalavimai nesikeis yra mažai realus.

Pastovus testavimas kiekvienoje fazėje yra reikalingas tam, kad šios fazės būtų patvirtinamos. Prototipų kūrimas yra reikalingas tam, kad būtų identifikuoti prieštaringi reikalavimai ir galimybė juos įvykdyti. Vartotojai naudojantys Krioklio metodą gali teigti, kad jeigu projektuotojai (arba kiti) nepadarys klaidų projektuojant tuomet nereikalingas žingsniuotas kiekvienos fazės patvirtinimas.

Sunku numatyti laiko sąnaudos kurių pareikalaus vieni ar kiti procesai kai nėra daromi jokie šios fazės darbeliai, nebent specialistas turi didelę patirtį ir žino kiek koks darbas gali pareikalauti laiko.

2.5 Kokybė vertinimas

Programinės įrangos kokybės vertinimas dažnai atliekamas naudojant ISO/IEC 9126 standartą [22]. Šis modelis pritaikomas betkokiai PĮ. Standartas turi šešis pagrindinius kriterijus, o jie turi subkriterijus (4 pav.).



4 pav. ISO/IEC 9126 standartas

Toliau lentelėje (3 lentelė) pateikiame subkriterijus šiam standartui.

3 lentelė. ISO standarto subkriterijų aprašymai

Kriterijus	Subkriterijus	Apibrėžimas
Funkcionalumas	Tinkamumas	Programinės įrangos atributai nurodantys esančias ir teisingas funkcijas reikalingas tam tikrai užduočiai.
	Tikslumas	Programinės įrangos atributai nurodantys grąžinamus teisingus ar sutartus rezultatus, ar efektus.
	Bendradarbiavimas	Programinės įrangos atributai nurodantys galimybes sistemai sąveikauti su specifikuotoms sistemom.
	Atitikimas	Programinės įrangos atributai kurie padaro programinę įrangą atitinkančią standartus, konvencijas ar įstatymus ir panašius aprašus.
	Apsauga	Programinės įrangos atributai saugantys nuo neteisėto panaudojimo ir priėjimo, nesvarbu tyčinio ar ne, prie programos ar duomenų.
Patikimumas	Išbaigtumas	Programinės įrangos atributai nurodantys klaidų dažnį programinėje įrangoje
	Pakantumas klaidoms	Programinės įrangos atributai nurodantys sistemos galimybes išlaikyti tam tikrą veikimo lygį įvykus klaidoms.
	Atstatomumas	Programinės įrangos atributai nurodantys galimumą atstatyti normalų veikimo darbą ir atgaminti duomenis tiesiogiai paveiktus klaidų bei laiką ir pastangas reikalingas tai atlikti.
Lengvumas naudoti	Suprantamumas	Programinės įrangos atributai nurodantys vartotojui reikalingas pastangas, kad suprastų loginius principus ir jų pritaikymą.
	Išmokstamumas	Programinės įrangos atributai nurodantys vartotojui reikalingas pastangas, kad išmokti naudotis programa.
	Darbingumas	Programinės įrangos atributai nurodantys vartotojo pastangas atliekant operacijas ir operacijų kontrolę.
Efektyvumas	Laiko naudojimas	Programinės įrangos atributai nurodantys sistemos veikimo laikus ir atliekamų operacijų skaičius per tam tikrą laiką.
	Resursų naudojimas	Programinės įrangos atributai nurodantys sunaudojamų resursų kiekį ir kiek laiko jie yra naudojami.
Palaikomumas	Analizuojamumas	Programinės įrangos atributai nurodantys reikalingą pastangų kieki siekiant nustatyti netikslumus, klaidų priežastis ar identifikuoti dalis kurias reikia modifikuoti.
	Keičiamumas	Programinės įrangos atributai nurodantys pastangų kiekį reikalingą modifikavimui, klaidų pašalinimui ar aplinkos pakeitimams.
	Stabilumas	Programinės įrangos atributai nurodantys riziką apie nenumatytus efektus atliekant modifikavimą.

	Testuojamumas	Programinės įrangos atributai nurodantys pastangas reikalingas ištestuoti programinę įrangą.
Pernešamumas	Pritaikomumas	Programinės įrangos atributai nurodantys galimybes pritaikyti skirtingoms aplinkoms nenaudojant jokių kitokių priemonių kaip tos kurios priklauso tai programinei įrangai.
	Instaliuojamumas	Programinės įrangos atributai nurodantys pastangas reikalingas instaliuoti programinę įrangą specifikuotoje aplinkoje.
	Prisitaikymas	Programinės įrangos atributai kurie leidžia programinei įrangai laikytis standartų ir konvencijų susijusių su portabilumu.
	Pakeičiamumas	Programinės įrangos atributai nurodantys galimybes naudoti šią programinę įrangą vietoj kitos programinės įrangos tos programinės įrangos aplinkoje.

2.6 Analitinės dalies išvados

Šiame skyriuje mes apžvelgėme internetines technologijas. Aprašėme tris populiariausias, tai .NET, J2EE ir PHP. Jas analizavome ir palyginome. Išskyrėme projektų kūrimą su J2EE technologija ir aprašėme jos problematiką ir ypatumus. Šioje dalyje trumpai apžvelgėme šių projektų kūrimą su Appfuse karkasu. Šis įrankis suteikia kūrėjui daug galimybių palengvindamas jo darbą. Tolesni skyriai aprašys projekto sukūrimą su Appfuse ir teikiamų galimybių tyrimą. Taip pat pateikėme ISO 9126 standarto aprašymą, pagal kurį apliekamas programinės įrangos kokybės vertinimas.

3. PROJEKTINĖ DALIS

3.1 Bendrabučių informacinė sistema

3.1.1 Sistemos paskirtis

Šiuo metu bendrabučiuose naudojama primityvi sistema. Visi praėjimai registruojami žurnaluose. Resursų (skalbyklos, sporto kambariai ir kt.) naudojimas registruojamas taip pat žurnaluose. Visos kitos paslaugos taip pat prieinamos rašant įvairius prašymus. Nėra galimybės patogiai gauti ataskaitas. Gyventojai neturi galimybės savarankiškai valdyti (rezervuoti resursą, valdyti savo svečių sąrašus ir kt.) teikiamų paslaugų.

Bendrabučių informacinė sistema naudojama efektyvesniam jų darbo valdymui. IS skirta svarbiausioms veiklos

funkcijoms: gyventojų ir svečių duomenų valdymas, gyventojų-valdytojų-budėtojų paslaugų valdymas.

Bendrabučių informacinė sistema yra užsakomasis projektas. Projektas kuriamas pagal užsakovo pateikiamus reikalavimus. Ekonominiu atžvilgiu šis projektas yra gera investicija į ateitį. Be to, jis gali būti plėtojamas arba integruojamas su kitais esančiais arba ateityje pasirodysiančiais projektais (pvz.: Bendrabučių apskaitos sistema, nekilnojamo turto apsaugos sistema ir pan.).

3.1.2 Projekto kūrimo pagrindimas

Informacinių technologijų amžiuje sistemos kuriamos darbo efektyvumui padidinti ir valdymo kaštams sumažinti. Kuriama sistema tam ir yra skirta. Ji supaprastins bendrabučių valdytojų, budėtojų, gyventojų kasdienines procedūras eliminuojant popierinių dokumentų poreikį, bei sistema bendrabučių paslaugas perkels į elektroninę erdvę.

Suinteresuotos šalys visas procedūras atliks vienoje vietoje, elektroninėmis priemonėmis, nuotoliniu būdu naudojantis Bendrabučių IS teikiamomis funkcijomis.

3.1.3 Sistemos panaudojimo atvejai

Sistemos panaudojimo atvejų aktoriai:

- Valdytojas
- Budėtojas
- IS administratorius
- Gyventojas

Pagrindinių panaudojimo atvejų sąrašas pagal aktorius:

Valdytojas (5 pav.)

- Valdyti gyventojų duomenis
- Valdyti svečių duomenis
- Peržiūrėti praėjimų registrą
- Valdyti pranešimus
- Peržiūrėti resursų naudojimąsi

Budėtojas (6 pav.)

- Peržiūrėti praėjimų registrą
- Registruoti praėjimą
- Peržiūrėti gyventojų sąrašą
- Valdyti svečių sąrašą
- Valdyti pranešimus
- Valdyti resursų naudojimąsi

Gyventojas (7 pav.)

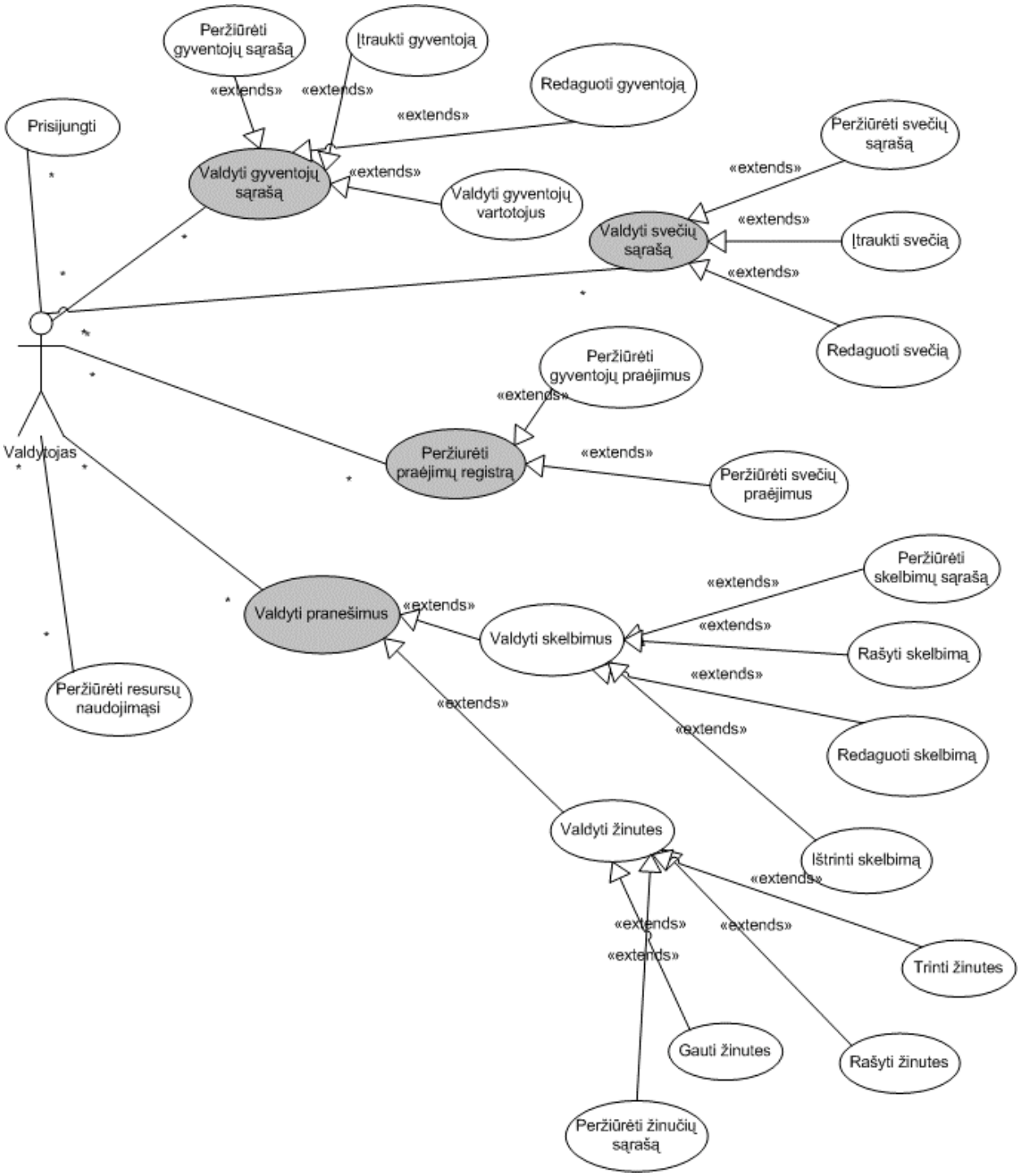
- Valdyti žinutes
- Peržiūrėti skelbimų sąrašą
- Valdyti savo duomenis
- Peržiūrėti resursų naudojimąsi
- Registruoti rezervaciją resursui
- Peržiūrėti lankytojus
- Valdyti svečių leidimus

IS administratorius (8 pav.)

- Valdyti bendrabučių sąrašą
- Valdyti bendrabučių resursų sąrašus
- Valdyti sistemos vartotojus

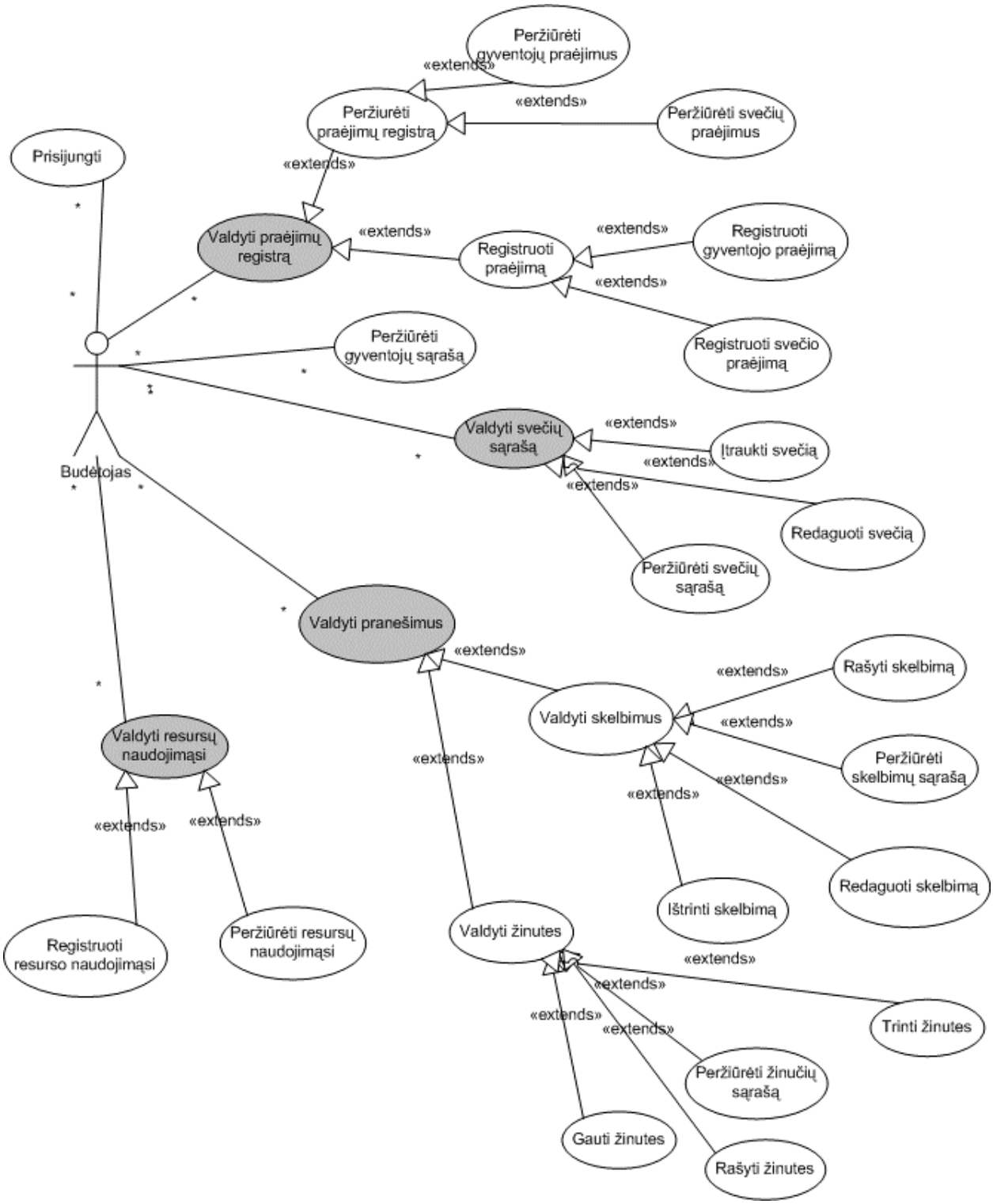
3.1.4 Panaudojimo atvejų diagramos

3.1.4.1 Valdytojas



5 pav. Valdytojo PA

3.1.4.2 Budėtojas



6 pav. Budėtojo PA

3.1.4.3 Gyventojas

7 pav. Gyventojo PA

3.1.4.4 IS administratorius

8 pav. IS administratoriaus PA

3.2 Pagrindiniai architektūros aspektai

Programinė įranga buvo kuriama naudojant kliento-serverio architektūros modelį. Serverinė dalis veikia bet kokios OS serveryje Java pagrindu, o klientinė dalis – naršyklė palaikanti priimtus internetinius standartus. Bendradarbiaujančių sistemų kūrimo stadijoje nebuvo numatyta, jos gali atsirasti plėtojimo stadijoje.

Sistemoje naudojami karkasai:

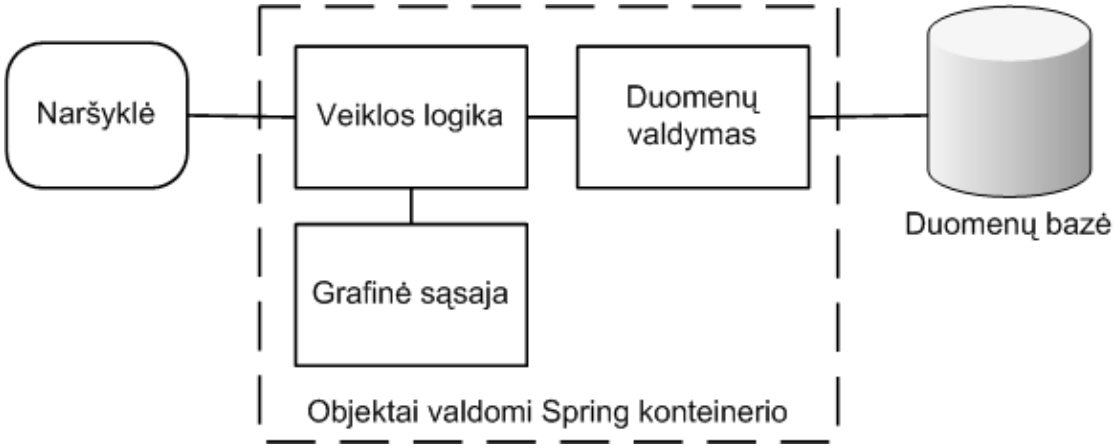
- Apache Struts2
- Hibernate
- Spring Framework

Sistemos platforma: Java 1.5.0

Kūrimo įrankiai:

- NetBeans IDE 6
- Appfuse 2.0 (Maven2)
- JBoss
- Oracle Database 10g Express Edition

Sistema sukurta pagal 3-tier(MVC, Model2) architektūrinį modelį(9 pav.).



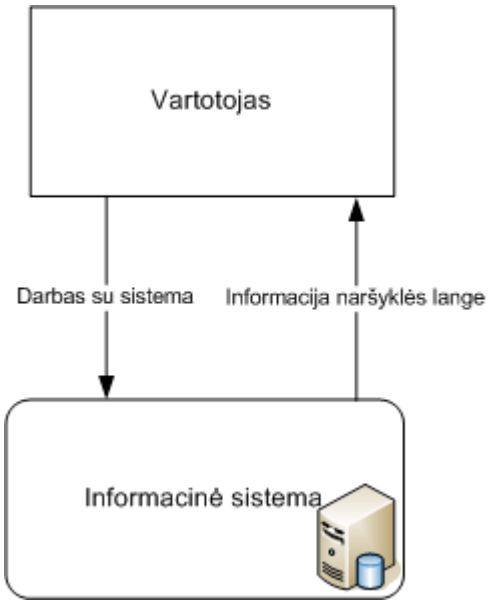
9 pav. MVC, Model2 architektūros modelis

MVC paprastai turi daug atmainų, tačiau esminė ir pagal kurią sukurta sistema, procesas vyksta sekančiai:

- Vartotojas atlieka bet kokį veiksmą susijusį su vartotojo sąsaja (pvz.: paspaudžia mygtuką);
- Kontroleris apdoroja šį veiksmą pagal nurodytą įvykių apdorojimo elementą (angl. event handler);
- Kontroleris iškviečia modelį, kuris, atlieka įvairius veiksmus su duomenimis (pvz.: atnaujina registracijos informaciją ir pan.);
- Atitinkamas „vaizdas“ naudojasi modeliu, kad sugeneruotu atitinkamai naują vartotojo sąsają (atvaizduoja naują registracijos informaciją, išmeta pranešimą apie užregistruotą praėjimą);
- Vartotojo sąsaja laukia tolimesnių veiksmų iš vartotojo.

Sistema buvo kuriama palaipsniui po vieną pasirinktą komponentą. Tai buvo daroma iteraciniu metodu pagal Agile principus.

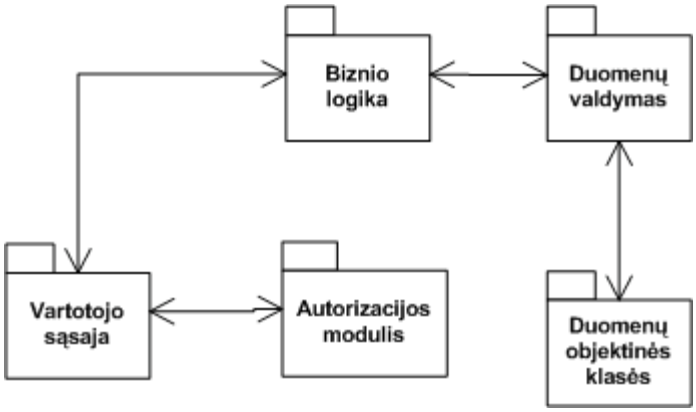
3.2.1 Veiklos kontekstas



10 pav. Sistemos veiklos konteksto diagrama

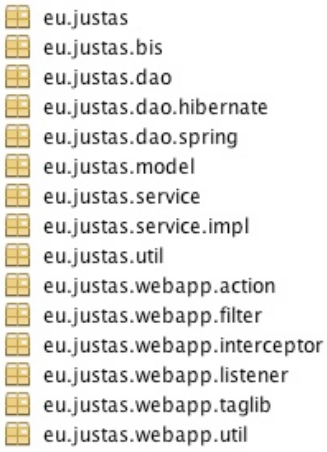
3.2.2 Sistemos komponentai

Sistemos paketai pateikiami UML komponentų diagrama, kuri aprašo sistemoje naudojamus komponentus (11 pav.).



11 pav. Komponentų diagrama

Sukurti sistemos Java paketai:

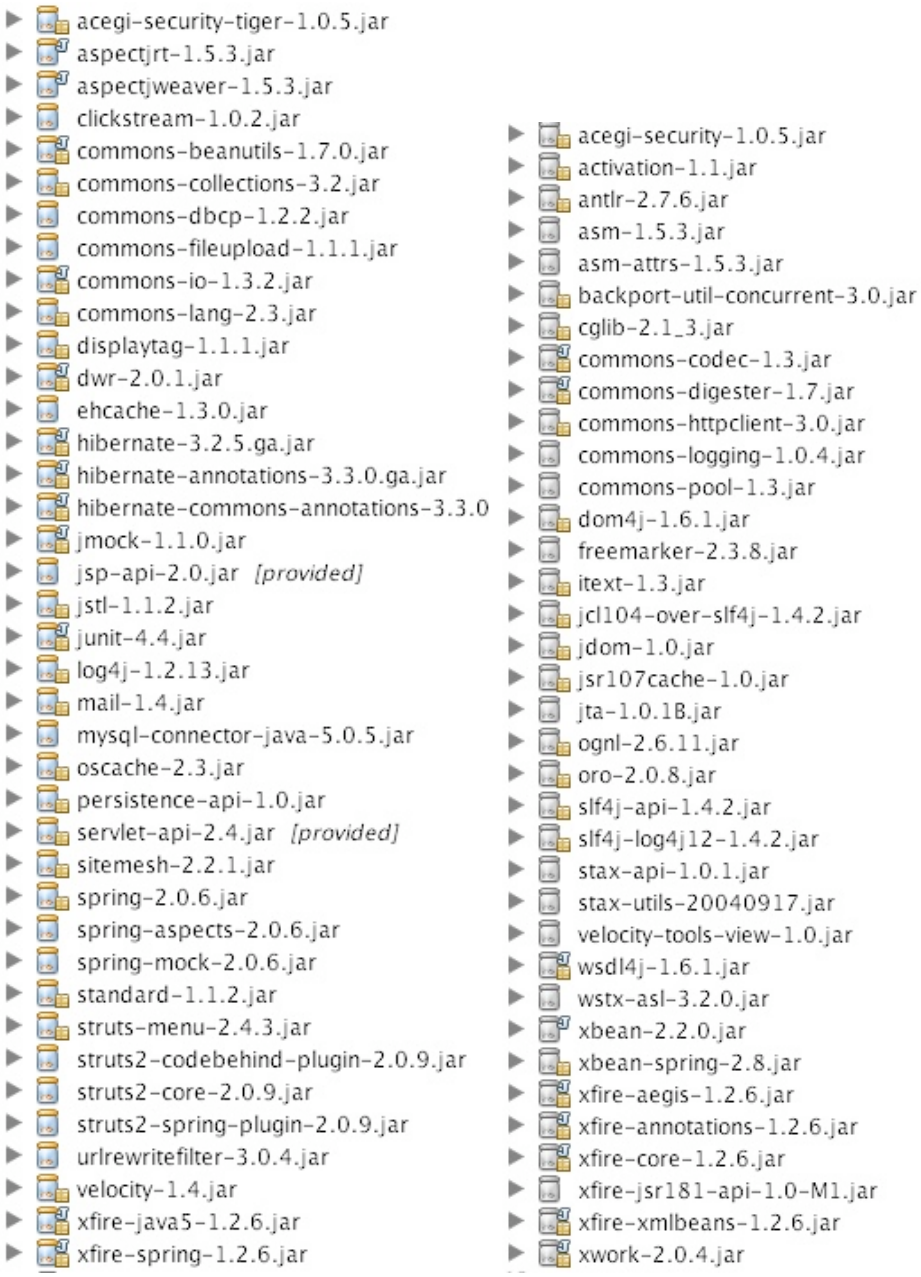


12 pav. J2EE projekto paketai

4 lentelė. Projekto paketų apibrėžimas

Paketas	**Apibrėžimas**
`eu.justas.bis`	Bendrinis paketas
`eu.justas.dao`	Duomenų valdymo paketas
`eu.justas.dao.hibernate`	Duomenų valdymo paketas skirtas Hibernate klasėms
`eu.justas.dao.spring`	Duomenų valdymo paketas skirtas Hibernate klasėms
`eu.justas.model`	Sistemos objektinių klasių paketas
`eu.justas.service`	Sistemos servisų paketas (pvz.: el.paštas, saugumas ir kt.)
`eu.justas.service.impl`	Sistemos servisų realizacijų klasės
`eu.justas.util`	Sistemos papildomų įrankių paketas (pvz.: datos konverteriai ir pnš.)
`eu.justas.webapp.*`	Sistemos internetinės sąsajos klasių paketas

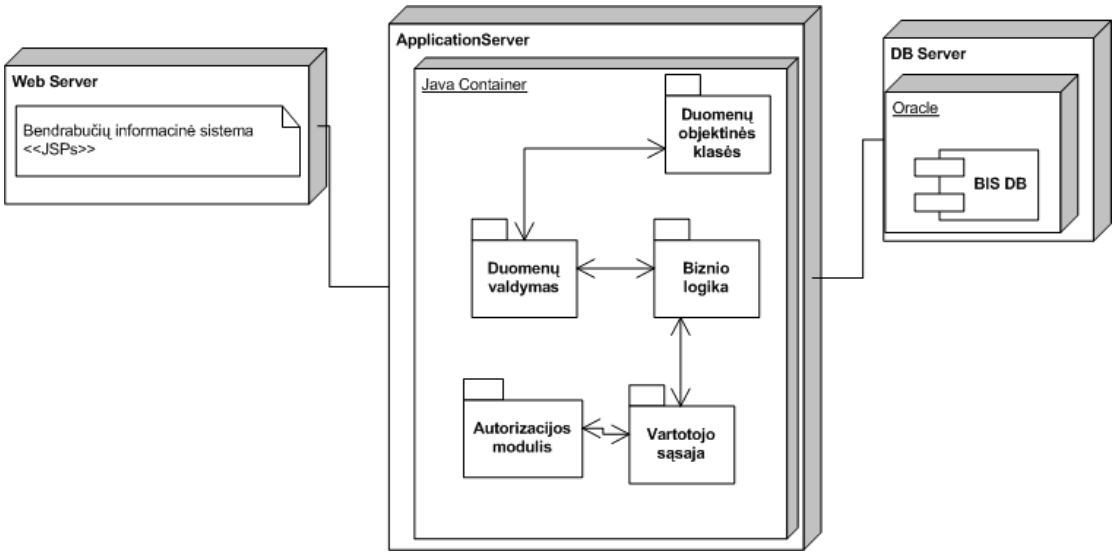
Sistemoje naudojamos bibliotekos:



13 pav. Bibliotekų failų medis

3.2.3 Sistemos išdėstymas

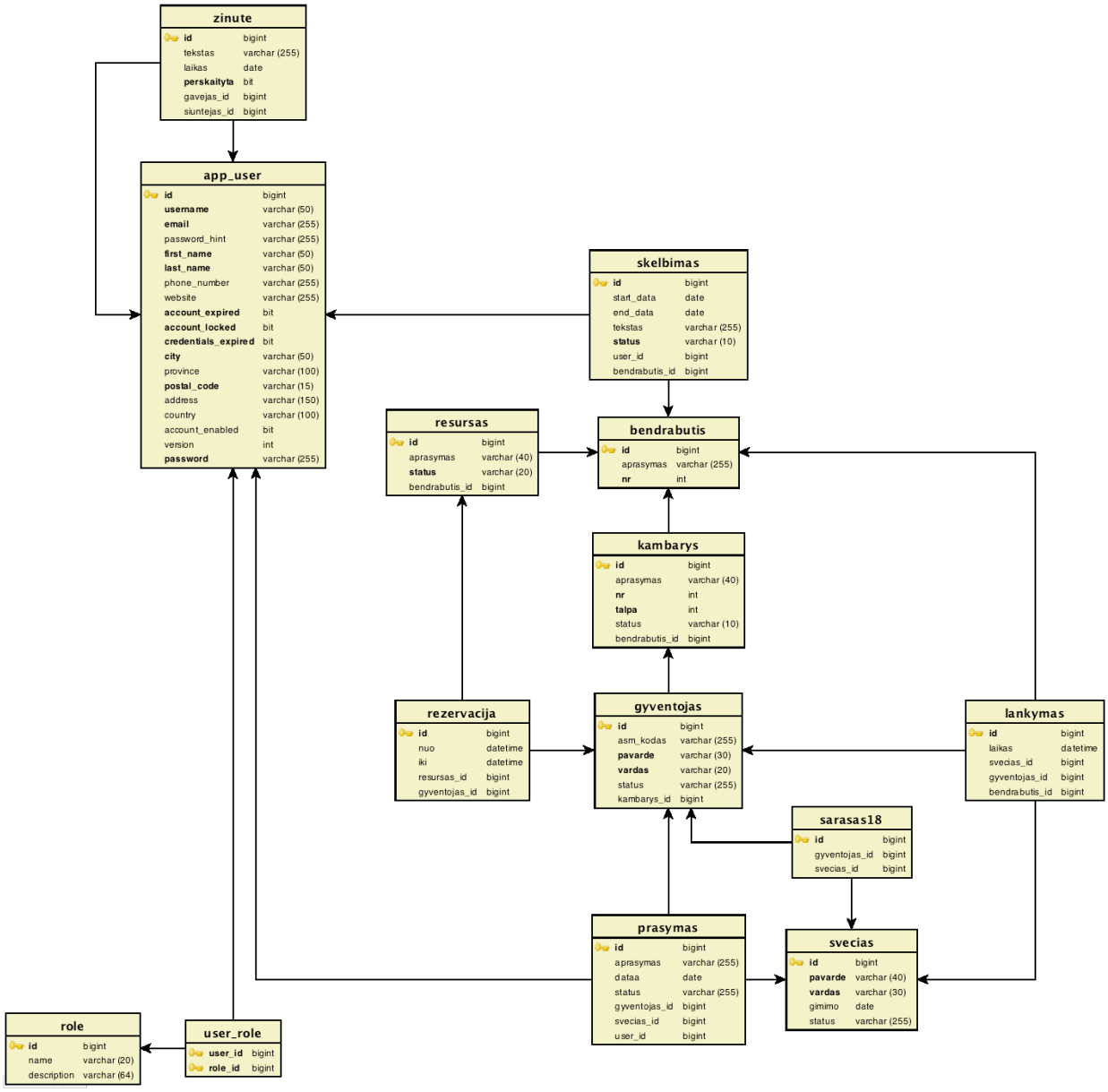
Sistemos išdėstymas pateikiamas UML išdėstymo (deployment) diagrama:



14 pav. Išdėstymo diagrama

Duomenų bazių serveris ir aplikacijų serveriai realizuojami atskiruose techniniuose taškuose. Vartotojai gali dirbti su savo technine įranga nuotoliniu būdu per internetą.

3.2.4 Duomenų bazės vaizdas



15 pav. Duomenų bazės fizinė schema

3.3 Sistemos apimties matavimas

Norėdami pademonstruoti kokia projekto apimtis, atlikome statinius kodo matavimus (5 lentelė). Į skaičiavimus neįtraukėme failų, kurie nebūna keičiami (pvz.: stiliai ir Javascript kodas).

5 lentelė. Kodo eilučių skaičiavimas

Paketas/Direktorija	**Failų tipas**	**Kodo eilučių skaičius**

eu.justas.model	.java	2111
eu.justas.dao	.java	290
eu.justas.service	.java	996
eu.justas.util	.java	632
eu.justas.webapp	.java	3289
main/recourses	.xml, .properties	4705
main/wepapp	.jsp	1876
eu.justas.dao (testai)	.java	313
eu.justas.service (testai)	.java	596
eu.justas.util (testai)	.java	172
eu.justas.webapp	.java	1213
test/resources	.xml, .properties	5229
	Suma:	**21422**

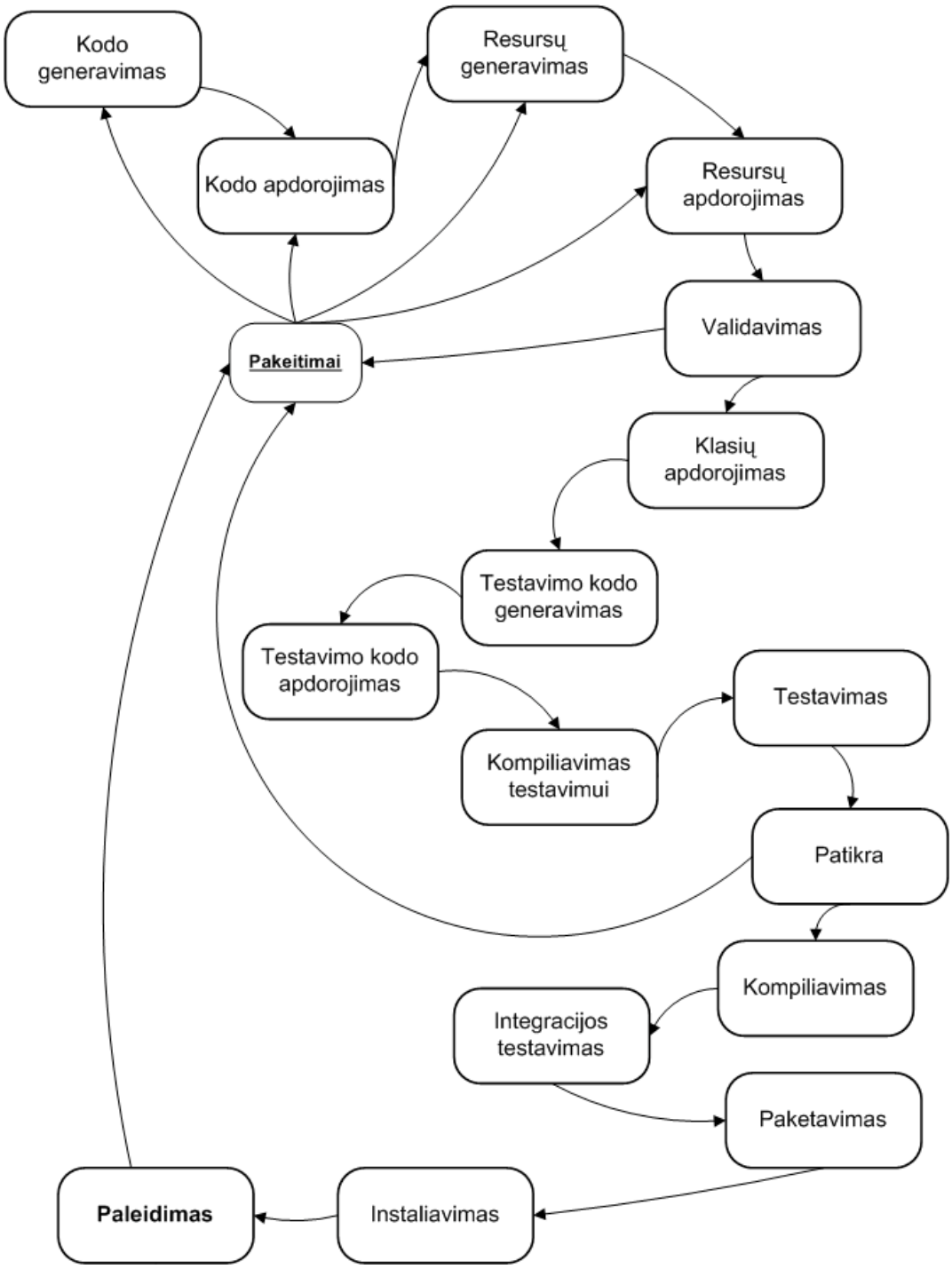
4. TYRIMO IR EKSPERIMENTINĖ DALIS

Šioje dalyje mes tyriame Appfuse karkasą sukurdami Bendrabučių informacinę sistemą. Tiriame įvairias Appfuse dalis, kritiškai įvertindami jų naudą. Taip pat pateikiame tyrimo pavyzdžius ir išvadas. Appfuse galima tirti kaip metodologiją, architektūros panaudojimą ir jos atskirus komponentus.

4.1 Projekto kūrimo procesai

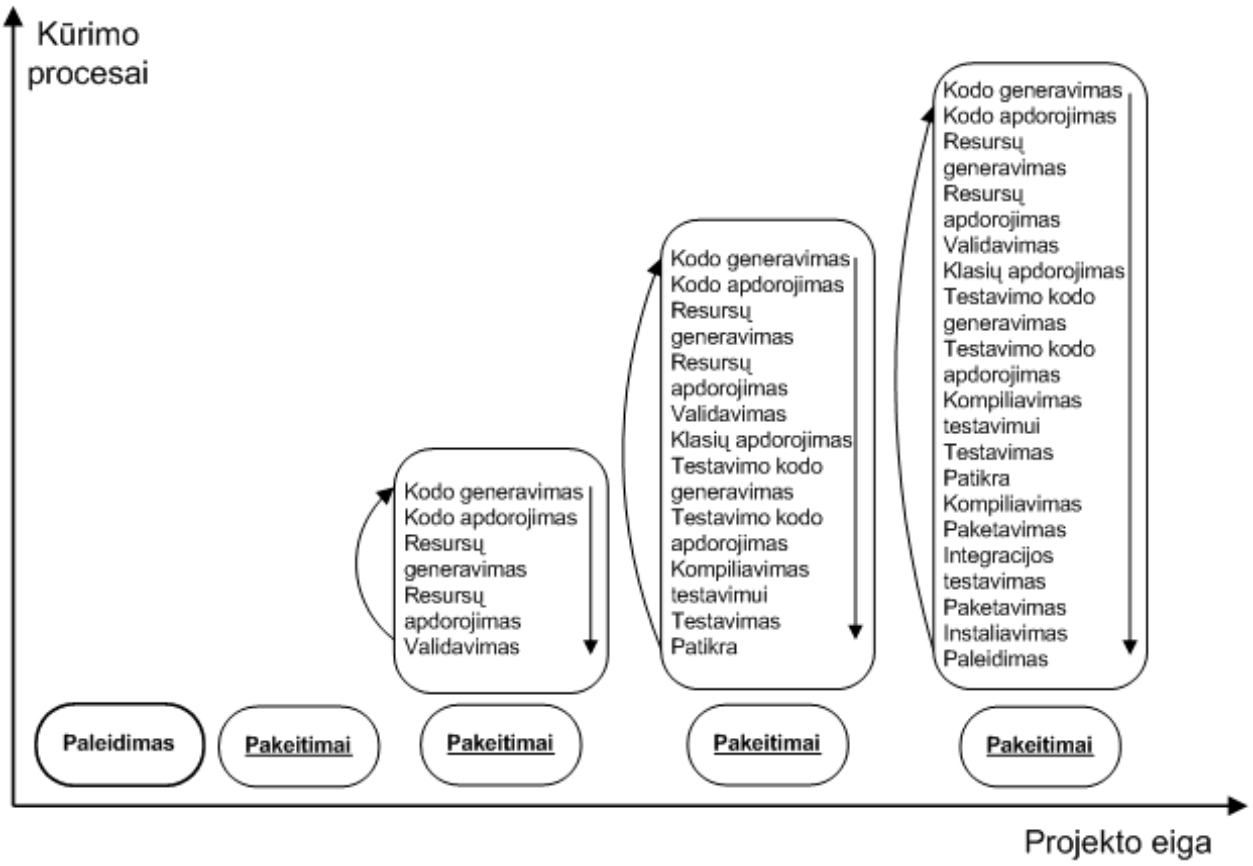
Agile programinės įrangos kūrimas yra sugebėjimas kurti aukštos kokybės programinę įrangą greitai ir kartu reaguojant į dažnus reikalavimų pakeitimus. Tam puikiai tinka įrankis Appfuse, nes jis gali būti naudojamas inkrementiniam projekto kūrimui su iteracijomis. Appfuse taip pat orientuota į aukštos kokybės programinės įrangos kūrimą naudojant naujausius įrankius ir metodologijas. Mes savo eksperimente identifikavome šiuos programinės įrangos kūrimo procesus (16 pav.) :

- Validavimas
- Kodo generavimas
- Kodo apdorojimas
- Resursų generavimas
- Resursų apdorojimas
- Kompiliavimas
- Klasių apdorojimas
- Testavimo kodo generavimas
- Testavimo kodo apdorojimas
- Kompiliavimas testavimui
- Testavimas
- Paketavimas
- Integracijos testavimas
- Patikra
- Instaliavimas
- Paleidimas



16 pav. Procesų diagrama

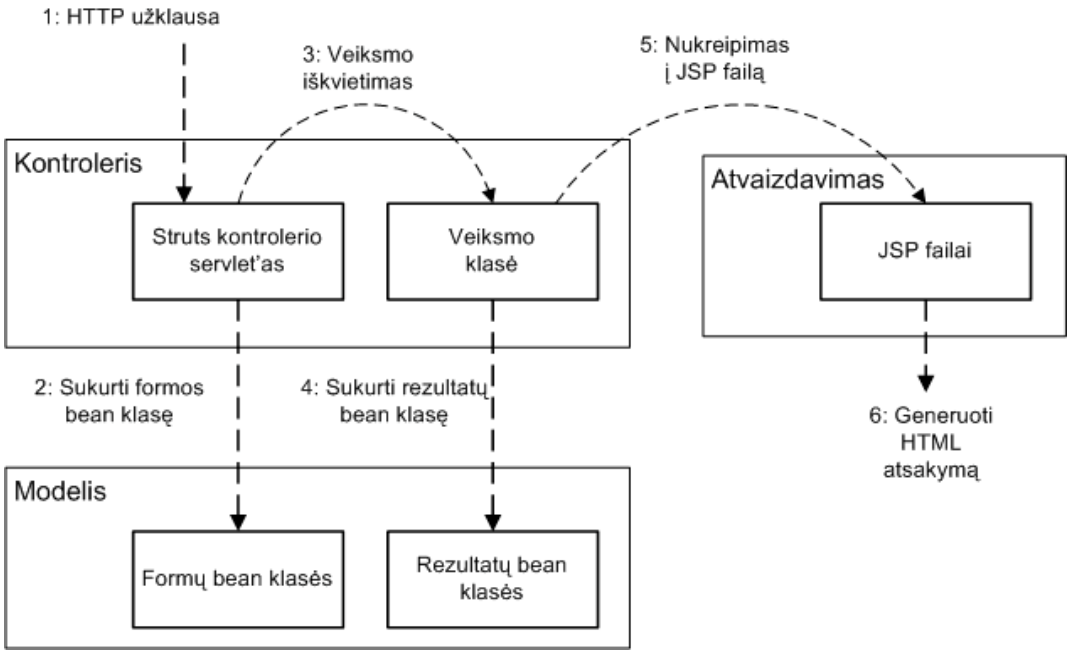
Projekto pradžioje buvo vykdomos mažos iteracijos, vėliau jas didinant. Tai atsispindi žemiau esančiame paveiksle (17 pav.).



17 pav. Procesai kūrimo eigoje

4.2 Architektūrinis modelis

Sudėtingėjant programinei įrangai, jos priežiūra tampa vis sudėtingesnė ir brangesnė. MVC modelio pagrindinė idėja yra suskaidyti programinę įrangą į loginius blokus ir juos kurti atskirai (18 pav.). Įvykdžius pakeitimus viename iš blokų, nereikia keisti kitų blogų. Taip yra, nes blokai turi nustatytus standartus bendravimui. Vieniems blokams nėra jokio skirtumo kaip veikia kiti blokai, jam svarbu kad jis gautų reikiamą informaciją ir bendradarbiavimas vyktų sklandžiai. Mūsų sukurtoje sistemoje yra laikomasi MVC modelio aprašymu. Tai padeda pasiekti Appfuse karkasas, kuris iš generuoja programinį kodą skirstydamas į paketus, skirtingiems sistemos lygiams.



18 pav. MVC architektūros projekto veikimo schema

Savo projekte mes pritaikėme modelio-peržiūros-valdiklio (model-view-controller (MVC)) modelį (6 lentelė). Pagrindinis programinis kodas yra išskaidomas į šias dalis: jsp, JavaBeans klasės, valdytojai (angl. manager) ir servletai (angl. servlets). Akivaizdu, kad jsp puslapiai MVC modelyje sudaro „Peržiūros“ dalį, kadangi jie yra sistemos bendravimo sąsaja su išoriniu pasauliu. JavaBeans klasės saugo duomenis ir jų tikrinimo logiką, todėl jos yra „Modelis“. Valdytojai paprastai atlieka CRUD (angl. create, retrieve, update, delete) operacijas su vartotojo duomenimis įrašydami, paimdami, atnaujindami ir trindami juos iš duomenų bazės. Tam pasinaudojamos JavaBeans klasėmis kaip saugyklos. Servletai gauna vartotojo užklausas, kai išsiunčiami jsp puslapių formų duomenys, ir jas apdoroja. O valdytojai yra naudojami bendrauti su duomenų baze ir valdyti sukurtą sesiją. Taigi valdytojai ir servletai sudaro „Valdiklį“.

6 lentelė. Paketai ir MVC lygmuo

Paketas/Direktorija	**MVC architektūros lygis**
`eu.justas.model`	Modelis
`eu.justas.dao`	Valdiklis
`eu.justas.service`	Valdiklis
`eu.justas.util`	Valdiklis
`eu.justas.webapp`	Valdiklis
`main/wepapp`	Atvaizdavimas

4.3 Sistemos kūrimas su Appfuse

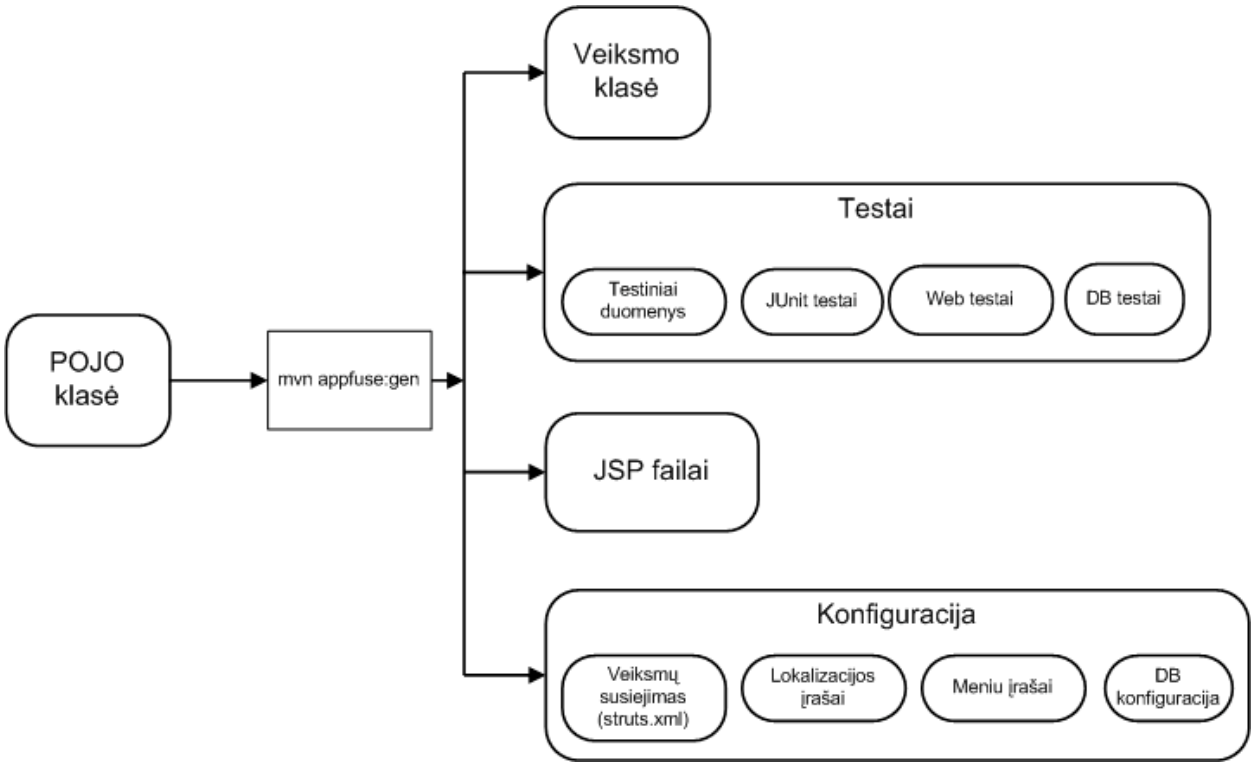
Appfuse karkasas buvo pritaikytas Bendrabučių informacinei sistemai kurti. Sistema pasižymi tokiomis ypatybėmis kaip daugiakalbiškumas, vartotojų grupės, saugumas, multiplatformiškumas ir kt.. Projektą pradėjome pasirinkdami technologijų masyvą pagal architektūras iš Appfuse. Pasirinkome Struts 2 Basic į kurį įeina technologijos: Struts2, Spring, Hibernate, Maven2, JUnit ir kt.. Sukurtas projektas naudojantis viena komandinės eilutės komandą (mvn archetype:create -DarchetypeGroupId= org.appfuse.archetypes -DarchetypeArtifactId= appfuse-basic-struts -DremoteRepositories= http://static.appfuse.org/releases -DarchetypeVersion=2.0.1 -DgroupId=eu.justas.bis -DartifactId=bis). Sukurta duomenų bazė ir prisijungimo konfigūracija įrašyta į projekto pom.xml failą. Gavome veikiančios sistemos rėmus. Jį galima pamatyti įvykdžius komandinės eilutės komandą (mvn jetty:run). Paleidimo metu galima nurodyti kokio lygio pranešimai bus rodomi komandinėje eilutėje.

Tolimesnis sistemos kūrimas vyko sukuriant objektines domeno klases. Anotacijomis nurodėme, kad klasės bus saugomos duomenų bazėje ir kokios atributų savybės. Kiekvienam objektui vykdomas generavimas verslo logikos veiksmų, prezentacijos, testų failai, bei testiniai duomenų failams (mvn appfuse:gen -Dentity=Objektas). Turint sugeneruotus logikos ir prezentacijos failus, juos keitėme pagal mūsų reikalavimus. Taip plėtėme sistemą programuodami visus panaudojimo atvejus.

Eksperimento metu su kuriama programine įranga norėjome pamatyti Appfuse privalumus ir juos išbandyti. Programinės įrangos kūrimas atitiko Agile principus, nes buvo kuriama su iteracijomis, papildant funkcionalumą, bei viskam rašomi testai.

4.3.1 Automatizavimas ir generavimas

Appfuse naudojasi Maven2 ir gali atlikti daug automatizavimo veiksmų: kompiliavimas, testavimas, diegimas ir t.t.. Taip pat Appfuse gali sugeneruoti biznio logikos klases, vartotojo sąsają, duomenų bazės struktūrą, testinius duomenis, testus, ir objektinės klasės (19 pav.). Projekto failai, reikalingi integruotoms programavimo aplinkoms, gali būti taip pat sugeneruoti naudojant vieną Appfuse komandą. Visi veiksmai gali būti atlikti iš komandinės eilutės arba naudojantis integruotos programavimo aplinkos funkcijomis. Kiekvienas vykdomas veiksmas yra lydimas detaliais pranešimais. Appfuse padeda efektyviai tvarkytis su dideliu kodo kiekiu, nes Java kodą, neskaitant jsp puslapių, suskaido į valdomus tokius vienetus: objektines klases, valdytojus, veiksmų klases ir įrankių klases, kurie talpinami į atskirus katalogus, paketus.



19 pav. Appfuse kodo generavimo schema

Appfuse iš objektinės klasės gali sugeneruoti begalę kitų reikalingų ir naudingų failų.

Appfuse sukuria projektui griežtą ir aiškią failų struktūra. Yra sukuriamos dvi pagrindinės direktorijos „src“ ir „target“ ir projekto failas pom.xml.

SRC - direktorija skirta išeities failams saugoti.

TARGET - direktorija skirta darbiniams vykdymo failams.

SRC direktorija turi tris subdirektorijas: main, site, test. Main yra saugomi java išeities teksto failai, site yra saugomi jsp išeities failais, test yra saugomi testiniai projekto resursai.

Formos puslapis gali būti įsivaizduojamas, kaip sąsaja įterpti ar atnaujinti lentelės eilutę ar jos dalį. Daugeliu atvejų formos laukai tiksliai atitinka laukus duomenų bazės lentelėje. Be to, jsp formų puslapiai iš esmės yra kelių parametrų funkcija, pvz., lentelė vardo, laukų, formos įvykdymo servleto ir t.t.

Formos puslapius Appfuse generuoja remiantis objektinių klasių atributais ir jų tipais. Daugiausiai sunkumų kelia laukų tipai ir tinkama sąsaja duomenims atvaizduoti. Tipinis pavyzdys yra pasirinkimas, naudojant vieno pasirinkimo mygtukų grupę (angl. radio buttons) ar keleto pasirinkimų laukelių grupę (angl. checkbox). Su šia užduotimi susidoroti tenka naudoti rankinį programavimą. Appfuse to padaryti kol kas negali ir nėra numatytą tokiems laukams specifikavimo priemonių.

4.3.2 Klaidų ir išimčių apdorojimas

Appfuse naudoja log4j. Tai vienas iš naudingiausių Java paketų, skirtų žurnalizavimui API. Šis paketas aktyvuojamas viena programinio kodo eilute Java klasės pradžioje, o jo paprastas konfigūracijos failas išsaugomas į esamą katalogą, taigi nėra jokios priežasties nepasinaudoti šios priemonės galimybėmis, kuriant bet kokio tipo programinę įrangą. log4j naudoja keletą esminių koncepcijų: pranešimų rimtumo lygiai arba prioritetai, prijungiamos paskirties šaltinių vietos arba prijungikliai ir hierarchiniai žurnalizavimo šaltiniai arba kategorijos. Žurnalo pranešimų išvedimo formą galima keisti formatavimo eilute. Viską, kas aprašyta aukščiau, galima sukonfigūruoti, pasinaudojant failu.

Prioritetai yra keli pranešimų lygiai, pvz., SEKIMAS, DERINIMAS, INFORMACIJA, ĮSPĖJIMAS, KLAIDA, KRITINĖ KLAIDA, vardijami didėjančia tvarka pagal rimtumą. Pagrindinis dalykas yra tai, kad slenkstinę reikšmę galima nustatyti taip, kad mus domintų tik aukščiau nurodyti pranešimai, išvedami tam tikra tvarka.

Prijungikliai yra pridedamos paskirties vietos. Pranešimą galima nusiųsti į kelias paskirties vietas. Pvz., tam tikrai pranešimui kategorijai išvesti galime pridėti komandinę eilutę arba failą. Kiekvienam prijungikliui galima nustatyti jo paties slenkstines reikšmes taip, kad komandinėje eilutėje būtų išvedami visi pranešimai iki DERINIMO, o į failą būtų įrašomi pranešimai su lygiu iki INFORMACIJA. Kiekvienas prijungiklis taip pat gali turėti skirtingą išvedimo formatą.

Įrankis log4j mūsų projekte konfigūruojamas log4j.xml faile. Demonstruojame dalį konfiguracijos:

```
<logger name="org.acegisecurity">
        <level value="ERROR"/>
    </logger>

    <logger name="org.apache">
        <level value="WARN"/>
    </logger>


    <logger name="org.hibernate">
        <level value="WARN"/>
    </logger>

    <logger name="org.springframework">
        <level value="WARN"/>
    </logger>

    <logger name="eu.justas">
        <level value="DEBUG"/>
    </logger>
```

Šiame pavyzdyje matome, kad paketo eu.justas yra nustatytas lygis „WARN“. Tai reiškia, kad bus išvedami pranešimai „WARN“ ir aukštesnio sekimo lygio t.y. „ERROR“ ir „FATAL ERROR“.

Kiekviename Java faile, kuriame naudojome log4j sekimo mechanizmą įtraukėme šią eilutę:

```
protected final Log log = LogFactory.getLog(getClass());
```

O sekimo pranešimai gali būti išvedami pasirinkto lygio. Lentelėje žemiau pateikiame metodus (7 lentelė).

7 lentelė. Sekimo metodai ir lygiai

Metodas	**Lygis**
`log.trace(log)`	Sekimo (TRACE)
`log.debug(log);`	Derinimo (DEBUGING)
`log.info(log);`	Informacijos (INFORMATION)
`log.warn(log);`	Įspėjimo (WARNING)
`log.error(log);`	Klaidos (ERROR)
`log.fatal(log);`	Fatalios klaidos (FATAL ERROR)

Informacijos išvedimo pavyzdys:

```
[INFO] [talledLocalContainer] DEBUG [http-8081-14] DateUtil.convertStringToDate(179) | converting date with pattern: yyyy-MM-dd
[INFO] [talledLocalContainer] DEBUG [http-8081-14] DateUtil.convertStringToDate(87) | converting '1983-05-12' to date with mask 'yyyy-MM-dd'
WARN - UserSecurityAdvice.before(71) | Access Denied: 'user' tried to modify 'admin'!
```

4.3.3 Testavimas

Testavimui kartais skiriama mažai dėmesio. Tai ne reiškia, kad testavimui nėra skiriama dėmesio visai. Publikacijose daug rašoma apie principą „testuok prieš programuodamas“, bet realybėje į testavimą žiūrima kaip į projekto užbaigimo prailginimą. Bet taip manyti yra neteisinga, nes jeigu yra rašomi testai prieš programavimą, galima rasti tai, kas paspartins projekto užbaigimą. Appfuse yra pilnai į testavimą orientuotas kūrimo įrankis. Jis nesugeneruoja instaliuojamo projekto, jeigu jame yra randamos klaidos. Testavimui naudojamos technologija JUnit, atlieka funkcinių vienetų testavimą, Canoo WebTest – atlieka internetinės sąsajos testavimą. DbUnit testuoja duomenų bazę. Naudojant kodo generavimą, pilnai sugeneruojami veikiantys testai visiems funkciniams vienetams ir internetinės sąsajos užduotims.

8 lentelė. Testavimo skaičiavimai

Testai	**Testų kiekis**
Vienetų (JUnit)	107
Duomenų bazės (DbUnit)	13
Sąsajos (Canoo WebTest)	80

Testuojant mūsų projektą (8 lentelė) testavimo rezultatai buvo išvedami į terminalo langą. Pakeikiame pranešimų pavyzdžius:

DbUnit testavimas:

```
DEBUG
 HibernateConfigurationTest.testColumnMapping(25) | Trying select * from:
eu.justas.model.Gyventojas
DEBUG
 HibernateConfigurationTest.testColumnMapping(28) | ok: eu.justas.model.Gyventojas
DEBUG
 HibernateConfigurationTest.testColumnMapping(25) | Trying select * from:
eu.justas.model.Lankymas
DEBUG
 HibernateConfigurationTest.testColumnMapping(28) | ok: eu.justas.model.Lankymas
```

JUnit:

```
Running eu.justas.webapp.action.BendrabutisActionTest

DEBUG - BendrabutisActionTest.testEdit(35) | testing edit...
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.131 sec
```

WebTest:

```
SearchGyventojass:

EditGyventojas:

SaveGyventojas:

AddGyventojas:

DeleteGyventojas:

GyventojasTests:
     [echo] Successfully ran all Gyventojas UI tests!
```

Aptikus klaidą testavimas stabdomas ir išvedamas klaidos pranešimas:

```
Embedded error: The following error occurred while executing this line:
/Users/justas/Documents/workspace/bis/src/test/resources/web-tests.xml:312: Wrong
document title found!. Expected value ".*Bendrabutis List.*" but got "Data Access
Error | KTU BIS"
```

4.4 Duomenų valdymo lygmuo

Appfuse karkasas sukurtai objekto klasei (angl. POJO) sugeneruoja: duomenų bazės lentelę, veiklos logikos klasę (įterpimui, sąrašui, ištrynimui, redagavimui), prezentacijos lygio failus jsp, testavimo duomenis, testavimo atvejus. Generavimas vykdomas objekto sąrašo valdymui, įterpimui, ištrynimui, redagavimui.

AppFuse karkasas gali dirbi su šiomis duomenų bazėmis

• Derby

• H2

• HSQLDB

• MySQL

• Oracle

• PostgreSQL

• SQL Server

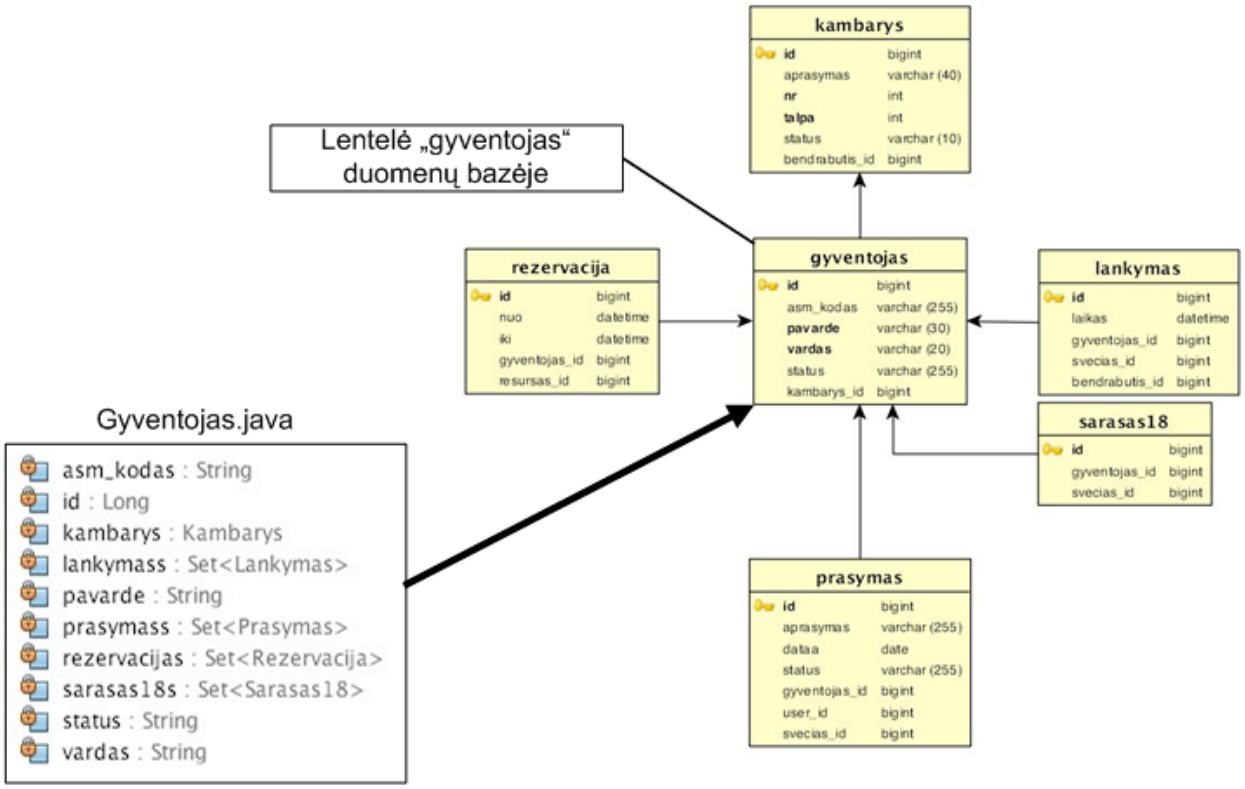
Populiariausiai naudojamas DB lygio karkasas yra Hibernate. Jis išlaisvina nuo rašymo rankomis JDBC kodą. Vietoj to, kad rašyti SQL ir JDBC mes galime naudotis objektų sąrašais. Naudojant Java klases su anotacijomis duomenys turi sąryšį su reliatyvia duomenų baze. Tai palengvina programavimą. Appfuse gali naudotis trimis šio lygio karkasais:

• Hibernate

• iBATIS

• JPA

Anotacijos naudojamos tam, kad būtų nurodomi ryšiai tarp objekto atributų ir duomenų bazės lentelės stulpelių. Hibernate turi galingą užklausų kalbą HQL. Ši kalba leidžia rašyti SQL užklausas, bet taip pat naudoti objektiškai orientuotą semantiką. Naudojant String ir Hibernate nereikia valdyti sesijų – tai tiesiog veikia.



20 pav. POJO klasės sąryšis duomenų baze

Paveiksle aukščiau (20 pav.) matome tiesioginį sąryšį tarp objektinės klasės ir duomenų lentelės. Mūsų projekte buvo kuriami tik objektų klasės. Appfuse su Hibernate pagalba automatiškai sugeneruodavo duomenų bazės lenteles naudojant JPersistence anotacijas, kurios atrodo taip:

```
@Id @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }

@ManyToOne()
    public Kambarys getKambarys() {
        return kambarys;
    }


@OneToMany(targetEntity=eu.justas.model.Rezervacija.class,
    cascade=CascadeType.ALL, mappedBy="gyventojas")
    public Set<Rezervacija> getRezervacijas() {
        return rezervacijas;
    }
```

4.4.1 SQL užklausų analizė

Pagal automatiškai sugeneruotus duomenų valdiklius SQL užklausos būna perteklinės. SQL užklausos perduoda daugiau duomenų negu reikia sistemos funkcijoms. Taip yra, nes

naudojami apibendrinti kontroleriai. Šiuos trūkumus galima eliminuoti sukuriant atskirus objektų kontrolerius kiekvienai objektinei klasei.

Pavyzdžiui prisijungimo metu kviečiama užklausa yra:

```
select
  user.id,
  user.account_expired,
  user.account_locked,
  user.address,
  user.city,
  user.country,
  user.postal_code,
  user.province,
  user.credentials_expired,
  user.email,
  user.account_enabled,
  user.first_name,
  user.last_name,
  user.password,
  user.password_hint,
  user.phone_number,
  user.username,
  user.version,
  user.website
from app_user user
where
user.username='admin'

select
  roles.user_id,
  roles.role_id,
  role1_.id,
  role1_.descriptio,
  role1_.name
from user_role roles
  left outer join role role1_ on roles.role_id=role1_.id
where roles.user_id=-2
```

Šioje užklausoje yra ištraukiami visi duomenys esantys lentelėse „user“ ir „role“. Akivaizdu, kad vartotojo autorizacijai nereikalingi detalūs jo duomenys.

4.5 Vartotojo sąsajos lygmuo

4.5.1 Vartotojo sąsajos dalies dekoravimas

Appfuse taip pat yra numatytas patogus internetinio puslapio dekoravimas. Meniu ir kitos nekintančios dalys neturi būti kartojamas kiekviename jsp faile. XML faile konfigūruojamas puslapio vaizdas ir kintančios dalys aprašomos atskiruose jsp failuose. Jsp failai nėra maišomi kartu tarp statinių ir dinaminių.

Meniu konstravimui naudojamas Velocity įrankis. Meniu punktai gali turėti poskyrius. Tai konfigūruojama faile menu-config.xml. Failo dalies pavyzdys pateikiamas žemiau.

```
<Menu name="MainMenu" title="mainMenu.title" page="/mainMenu.html"
roles="ROLE_ADMIN,ROLE_USER,ROLE_BUDET,ROLE_VALDYT"/>

<Menu name="UserMenu" title="menu.user" description="User Menu"
page="/editProfile.html" roles="ROLE_ADMIN,ROLE_USER,ROLE_BUDET,ROLE_VALDYT"/>

<Menu name="AdminMenu" title="menu.admin" description="Admin Menu" roles="ROLE_ADMIN"
width="120" page="/admin/users.html">
            <Item name="ViewUsers" title="menu.admin.users" page="/admin/users.html"/>
```

```
            <Item name="ActiveUsers" title="mainMenu.activeUsers"
page="/admin/activeUsers.html"/>
            <Item name="ReloadContext" title="menu.admin.reload"
page="/admin/reload.html"/>
            <Item name="Clickstream" title="menu.clickstream"
page="/admin/clickstreams.jsp"/>
</Menu>

<!--Bendrabutis-START-->
        <Menu name="BendrabutisMenu" title="bendrabutisList.title"
page="/bendrabutiss.html" roles="ROLE_ADMIN,ROLE_BUDET,ROLE_VALDYT"/>
        <!--Bendrabutis-END-->
<!--Kambarys-START-->
        <Menu name="KambarysMenu" title="kambarysList.title" page="/kambaryss.html"
roles="ROLE_ADMIN,ROLE_USER,ROLE_BUDET,ROLE_VALDYT"/>
        <!--Kambarys-END-->
```

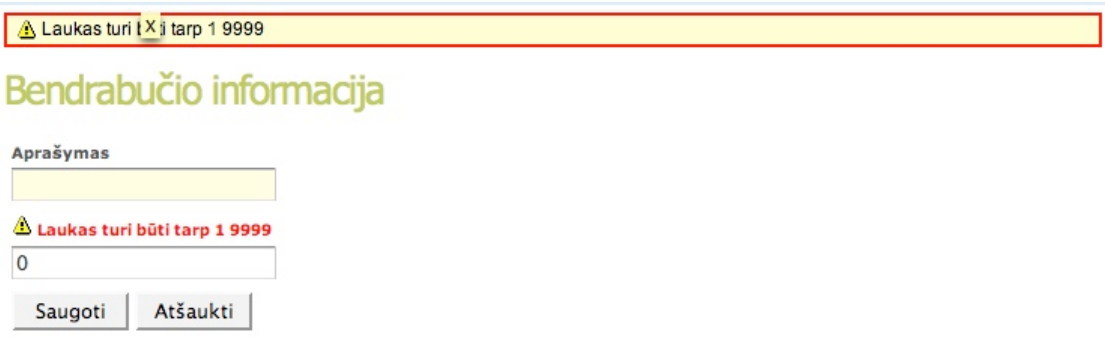
Pavyzdyje matome keletą meniu punktų su papunkčiais. Meniu skyrius žymimas <Menu> anotacija, o poskyris <Item>. Su anotacija „roles“ apibrėžiami vartotojų grupės, kurios matys šiuos meniu punktus.

4.5.2 Duomenų tikrinimas ir validavimas

Kadangi mūsų kuriamas projektas yra internetinis ir turi daug įvedimo formų, dėl to reikalingas duomenų tikrinimas ir validavimas. Appfuse turi šiai sričiai specialius įrankius. Formų validavimui naudojamas XWork karkasas.

```
<validators>
    <field name="bendrabutis.nr">
        <field-validator type="int">
            <param name="min">1</param>
            <param name="max">9999</param>
            <message key="invalid.count">
               ${min} ${max}
            </message>
        </field-validator>
    </field>
</validators>
```

Pavyzdyje sukonfigūruotas atributo bendrabutis.nr tikrinimas. Tikrinimo tipas yra „int“ sveikas skaičius. Taip pat papildomai nurodytos galimų reikšmių ribos nuo 1 iki 9999. Sukonfigūruotas pranešimas, kuris bus rodomas pažeidus šias taisykles. Pranešimas yra rakto pobūdžio. Taip yra dėl lokalizavimo. Tekstas nėra tiesiai koduojamas. Vartojami raktai, pagal kuriuos įstatomas tekstas naudojamai sąsajos kalbai. Jei laukas neteisingas užpildytas, bandant išsaugoti prie jo atsiranda paaiškinimas (21 pav.).



21 pav. Klaidos pranešimo dialogas

4.5.3 Lokalizavimas

Tai veiksmas, atvirkščias internacionalizavimui. Programa pritaikoma darbui konkrečioje kultūrinėje terpėje. Didžiausią lokalizavimo darbų dalį sudaro dialogo tekstų ir elektroninių žinynų vertimas. Todėl lokalizavimas dažnai tapatinamas su vertimu.

Projektams kuriamiems su Appfuse, lokalizavimo mechanizmas yra įdiegiamas. Trumpai išanalizuosime lokalizavimo mechanizmą.

Visame programos kode yra nenaudojami tiesioginiai kalbos pranešimai. O bet koks pranešimas, kuris bus matomas vartotojui yra žymimas raktu. Vartotojas darbo metu su programa gali pasirinkti naudojamą kalbą. Tai realizuojama jsp faile:

```
<c:if test="${pageContext.request.locale.language != 'en'}">
    <div id="switchLocale"><a href="<c:url value='/?locale=en'/>">
<fmt:message key="webapp.name"/> in English</a></div>
</c:if>
<c:if test="${pageContext.request.locale.language != 'lt'}">
    <div id="switchLocale"><a href="<c:url value='/?locale=lt'/>">
<fmt:message key="webapp.name"/> Lietuviškai</a></div>
</c:if>
```

Tada bet koks tekstas išvedamas jsp faile rakto pagalba:

```
<fmt:message key="bendrabutis.deleted"/>
```

Arba java faile:

```
getText("bendrabutis.deleted")
```

Sąsajos pavyzdžiai pateikiami žemiau (22 pav.) lietuvių ir anglų kalbomis.

22 pav. Grafinės sąsajos lokalizavimo pavyzdys

Raktų reikšmės yra nurodomos .properties failuose, pavyzdžiui lietuviu kalbos tekstas saugomas ApplicationResources_lt.properties faile.

```
# -- login --
login.title=Prisjungimas
login.heading=Prisijungimas
login.rememberMe=Atsiminti mane
login.signup=Neregistruotiems. <a href="{0}">Sukurti</a> prisijungimą.
hint sent to you." onclick="passwordHint(); return false"> Priminimas </a>.
login.passwordHint.sent=Slaptažodžio priminimas {0} išsiųstas {1}.
login.passwordHint.error=Vartotojo vardas {0} nerastas.
```

4.6 Verslo logikos lygmuo

4.6.1 Saugumas

Autorizavimo ir autentifikavimo, vartotojų rolių, prisijungimo išsisaugojimo funkcionalumas yra naudojamas daugelyje internetinių projektų. Todėl iš naujo kurti šio lygio funkcionalumo neverta, kai yra profesionalų sukurta sistema Acegi Security. Ši sistema yra puikiai išdirbta ir atitinka šiuolaikinius internetinių projektų saugumo reikalavimus. Todėl saugumas ir vartotojų autorizavimas bei autentifikavimas yra įtraukiami į visus Appfuse sugeneruotus projektus. SSL funkcionalumas yra taip pat įdiegiamas pagal nutylėjimą.

Paminėtas funkcionalumas atitinka reikalavimus Bendrabučių informacinei sistemai. Todėl nereikėjo kurti patiems ir šios dalies funkcionalumo. Sistemoje naudojome keturias vartotojų rolės (kaip ir minėta panaudojimo atvejuose projektinėje dalyje). Administratorius turi išskirtines teises ir gali valdyti visą sistemą. Jis gali redaguoti vartotojus. Specialus funkcionalumo failai sudėti į specialias direktorijas, kurių negali pasiekti kitų grupių vartotojai. O failų prieinamumas redaguojamas specialiame saugumo faile (security.xml), kuris yra Acegi Security karkaso dalis. Šiame faile yra aprašomi sisteminiai vienetai ir jų prieinamumo teisės. Failo security.xml dalies pavyzdys:

```
<bean id="methodSecurityInterceptor"
class="org.acegisecurity.intercept.method.aopalliance.MethodSecurityInterceptor">
        <property name="authenticationManager" ref="authenticationManager"/>
        <property name="accessDecisionManager" ref="accessDecisionManager"/>
        <property name="objectDefinitionSource">
```

```
            <value>
                eu.justas.service.UserManager.getUsers=ROLE_ADMIN
                eu.justas.service.UserManager.removeUser=ROLE_ADMIN
            </value>
        </property>
    </bean>
```

Šiame kode matome, kad metodai getUsers ir removeUser gali būti atliekami tik turint rolę ROLE_ADMIN. Todėl bandant panaudoti šiuos metodus kitų rolių vartotojų veiksmuose, bus išmetamas saugumo pažeidimo pranešimas. Kitų rolių vartotojų savo sesijose šių metodų iškviesti negalės.

4.6.2 El. pašto priemonės

El. pašto įrankiai taip pat įtraukiami į sugeneruojamus Appfuse projektus. Laišką išsiusti labai paprasta:

```
mailMessage.setTo(adresas);
mailMessage.setSubject(tema);
mailMessage.setText(tekstas);
mailEngine.send(mailMessage);
```

SMTP serveris ir prisijungimo detalės konfigūruojamos mail.properties faile.

4.7 Kokybės vertinimas

Bendrabučių informacinės sistemos kokybės vertinimas buvo atliekamas pagal 2.5 skyriuje apibrėžtus ISO 9126 standarto kriterijus. Kokybės subkriterijų vertinimo parametrai apibrėžti remiantis sistemai iškeltais reikalavimais, o vertinimas atliekamas stebint potencialių vartotojų darbą su sistema, juos apklausiant ir analizuojant įvairią statistinę sistemos registruojamą informaciją.

Kiekvienas kokybės subkriterijus susideda iš vieno ar daugiau jį nusakančių parametrų. Kiekvienas parametras įvertintas dešimtbalėje skalėje: 1 – kokybė labai bloga arba toks parametras išvis nerealizuotas, 10 – kokybė labai gera.

Žemiau pateiktoje lentelėje (9 lentelė) pateikiami sistemos kokybės vertinimo rezultatai. Kiekvienas kokybės kriterijus įvertinamas pagal visus jį sudarančius subkriterijus paskui apskaičiuojant bendrą kriterijaus įvertinimą.

9 lentelė. Sistemos kokybės vertinimas pagal ISO 9126 standartą

Funkcionalumas			
	Subkriterijus	*Subkriterijaus parametrų aprašymai*	*Įvertinimas*
	Tinkamumas	Realizuoti visi funkciniai reikalavimai	7
		Realizuoti visi nefunkciniai reikalavimai	6
	Tikslumas	Sistemos atliekami veiksmai yra tikslūs	8

	Bendradarbiavimas	Sistema gali būti sukonfigūruota bendradarbiavimui su kitomis sistemomis	9
	Atitikimas	Sistemos atliekamos funkcijos atitinka organizacijos nuostatus	9
		Sistema neprieštarauja įstatymams	10
	Apsauga	Sistemos duomenys nėra pasiekiami tretiesiems asmenims	10
		Vartotojai negali pasiekti ne jiems skirtos informacijos	10
		Sistema yra apsaugota nuo įsilaužimų	10
		Prisijungimui naudojamas koduotas kanalas SSL	10
		Bendras	8,9
Patikimumas			
	Subkriterijus	*Subkriterijaus parametrų aprašymai*	*Įvertinimas*
	Išbaigtumas	Sistemoje nėra kritinių klaidų	7
	Pakantumas klaidoms	Atsiradus klaidai sistema nenustoja veikti	8
		Sistema apdoroja klaidas aukštame lygyje	7
	Atstatomumas	Sistema, įvykus klaidai, nesugadina duomenų	9
		Bendras	7,8
Lengvumas naudoti			
	Subkriterijus	*Subkriterijaus parametrų aprašymai*	*Įvertinimas*
	Suprantamumas	Manipuliavimas sistema galimas įprastais metodais	8
		Sistemoje daug paaiškinimų	6
	Išmokstamumas	Sistema turi visų vartotojų grupių dokumentacijas	5
		Su sistema galima išmokti dirbi ir be dokumentacijos	9
	Darbingumas	Su sistema paprasta dirbti	9
		Sistemos stilius neerzinantis	8
		Bendras	7,5
Efektyvumas			
	Subkriterijus	*Subkriterijaus parametrų aprašymai*	*Įvertinimas*
	Laiko naudojimas	Nedideli pakeitimai nereikalauja perkrauti lango	3
		Vartotojo sąsaja užkrauname per mažiau nei 1s	6
	Resursų naudojimas	Sistema naudoja pilną sistemos pajėgumą tik 1% laiko	8
		Duomenų bazės dydis neauga eksponentiškai	9
		Bendras	6,5
Palaikomumas			
	Subkriterijus	*Subkriterijaus parametrų aprašymai*	*Įvertinimas*
	Analizuojamumas	Sistemos kodą lengva analizuoti	8
		Klaidų pranešimai yra informatyvus	9

	Keičiamumas	Sistema pilnai dokumentuota ir lengva daryti pakeitimus naudojantis ja	5
		Galima keisti sistemą neturint sistemos sukūrėjo žinių	8
	Stabilumas	Sistema veikia be sutrikimų	8
	Testuojamumas	Sistemo turi visų lygių testus	10
		Testai pereina visą sistemos kodą	6
		Bendras	7,7
Pernešamumas			
	Subkriterijus	*Subkriterijaus parametrų aprašymai*	*Įvertinimas*
	Pritaikomumas	Sistema gali dirbti bet kokios operacinės sistemos aplinkoje	10
	Instaliuojamumas	Paprastas sistemos įdiegimas	10
	Prisitaikymas	Naudojami plačiai taikomi standartai	10
	Pakeičiamumas	Sistemos komponentus galima pakeisti kitais įrankiais	10
		Bendras	10

4.8 Tolimesnė sistemos tobulinimo, plėtojimo ir tyrimo analizė

Šiame skyrelyje pateikiame dalis, kurios galėtų būti analizuojamos ateities darbuose. Ateityje būtų galima ištirti sistemos našumą. Bandyti sistemą, kurti su dubliavimo technologija. Taip pat Appfuse turi puikias priemones sistemoms naudoti AJAX technologiją. Tai nebuvo panaudota ir ištirta mūsų darbe. Darbas buvo atliekamas vieno žmogaus, todėl lygiagretaus programavimo galimybės liko neištirtos. Kaip Appfuse patobulinimą būtų galima pasiūlyti perteklinio kodo eliminavimą.

Ateityje taip pat būtų galima gilintis į Appfuse trūkumus. Kuriant projektą buvo pastebėta, kad SQL užklausos yra perteklinės, sugeneruotame kode sunkiau orientuotis, daug konfigūravimo atliekama XML failuose. Automatiškai generuoto kodo yra akivaizdžiai daugiau negu parašytų programuotojas. Sugeneruoti testai atlieka tik bazinių funkcijų veikimo patikrinimą.

4.8.1 Našumo tyrimas

Sukūrus bet kokią programinę įrangą, yra įprasta įvertinti jos našumą. Dažnai naudojami standartiniai metodai kompiuterio programų našumui patikrinti bei palyginti jas su kitomis programomis (angl. benchmarking). Regresijos testai įvertina našumą blogiausiu atveju. Žiniatinklio taikomosios programos testavimas skiriasi nuo autonominės programos. Autonominėms programoms galima sukurti testavimo paketus, kurie greitai ir pakartotinai išbandot atskirus metodus arba veiksmų seką, kurios galima tikėtis įprastinio naudojimosi metu.

Žiniatinklio taikomosiose programose 0.1 sekundės yra maždaug tokia riba, kai vartotojas jaučia, jog sistemos atsakas įvyksta akimirksniu, o tai reiškia, kad nereikia jokio specialaus grįžtamojo ryšio, išskyrus parodomus rezultatus.

1.0 sekundė yra maždaug tokia riba, kai vartotojo minčių tėkmė išliks nepertraukiama, nors vartotojas pastebės uždelsimą. Paprastai nereikia jokio specialaus grįžtamojo ryšio, jei uždelsimas yra tarp 0,1 ir 1,0 sekundės, tačiau tuo metu vartotojas jau nebesijaučia tiesiogiai dirbantis su duomenimis. 10,0 sekundžių yra maždaug tokia riba, kai vartotojo dėmesys yra išlaikomas ties dialogu. Ilgesnio uždelsimo atveju vartotojai, laukdami kol kompiuteris baigs darbą, norės atlikti kitas užduotis, todėl turėtų parodoma informacija, kada galima tikėtis laukiamo rezultato.

Grįžtamasis ryšys uždelsimų metu yra ypač svarbus, jei nusimato, kad atsako laikas bus ilgas, kadangi, priešingu atveju, vartotojai nežinos, ko tikėtis.

Appfuse nepateikia laikinių charakteristikų internetinės sąsajos testavime. Galima tik vizualiai stebėti kaip atliekami internetinės sąsajos veiksmai. Todėl tai galėtų būti patobulinimas šio karkaso.

4.8.2 Perteklinio kodo sumažinimas

Kodo pertekliškumas iki šiol yra vienas iš labiausiai nepageidaujamų aspektų bet kokiame stambiame programinio kodo projekte. Dažnai kūrėjai renkasi kokį nors greitą tam tikros kodo dalies pataisymą, pasiskolindami programinį kodą iš kitos dalies, kuri iš esmės yra tiksli kodo fragmento kopija. Toks greito kodo pataisymo tipas yra nekas kitas, kaip palikta pažeidžiama vieta klaidai. Jei vienoje iš tų kopijų klaida yra randama ir ištaisoma, o apie kitą vietą pamirštama.

Norint išspręsti šią problemą, kodo pakartotinis panaudojimas yra viena iš svarbiausių priemonių, remiantis programinės įrangos kūrimo praktika. Pakartojamo kodo dalis gali būti patalpinama į funkciją, kuri savo ruožtu būtų patalpinama kokioje nors patogioje vietoje, pvz., bibliotekoje, bei kviečiama iš daugelio vietų, nerizikuojant, kad naudojamas netinkamas kodas. Aišku, kad pakartotinis panaudojimas yra teisingas būdas, jei mums reikia pasinaudoti tuo pačiu programiniu kodu daugiau nei vienoje vietoje, tačiau tai netaikoma, kai naudojamas panašus, o ne identiškas kodas. Dažnai susiduriame su problema, kai turime nukopijuoti kodo gabalą iš kitos vietos, pakeisdami tik kelis kintamųjų vardus, o logiką palikdami tą pačią. Tai dar viena svarbi išvengiamo pertekliškumo forma, nes jei logika vėliau keičiasi vienoje vietoje, tai ji rankiniu būdu turi būti pakeičiama ir kitose vietose su skirtingais kintamaisiais.

4.8.3 CVS naudojimas ir lygiagretus plėtojimas

Tokios apimties projekto atveju turėtų būtų privaloma versijų kontrolė. Tai ne tik užtikrina gerą bendradarbiavimą tarp keleto kūrėjų, tačiau turint saugyklas gerai veikiančiuose serveriuose, tokiu būdu sumažinamos saugumo spragos, atsirandančios nešiojantis programinį kodą asmeniniuose nešiojamuosiuose kompiuteriuose. Be to, pasidaro ypač lengva dirbti esant bet kurioje vietoje ir nereikia jaudintis, ar dirbama su naujausia kodo versija.

Lygiagretus skirtingų mūsų projekto dalių plėtojimas buvo įmanomas, be to, labai naudingas dėl mūsų pritaikyto MVC modelio. CVS naudojimas leidžia greitai sujungti įvairių dalių programinio kodo atnaujinimus. Tai smarkiai sumažina įprastinį el. pašto bei kitų priemonių naudojimą netgi dalinantis laikinu bandomuoju programiniu kodu.

Appfuse sugeneruotas kodas tinka daugeliui integruotų programavimo aplinkų. Kiekvienas programuotojas gali pasirinkti jam mėgstamą įrankį.

4.8.4 AJAX technologijos

Žymi šių dienų žiniatinklio taikomųjų programų dalis vis plačiau naudoja naują technologiją - Asynchronous Javascript and XML (AJAX) (asinchroniškas JavaScript ir XML). Kol mes žinojome tik apie tokį perdavimo metodą, kai visas žiniatinklio puslapis yra persiunčiamas iš serverio į klientui, netgi ir tais atvejais, kai atliekami smulkūs pakeitimai, o tada atsiunčiamas atgal. Tai sąlygoja nemenką užvėlinimą atsakymo metu; paprastai tuo atveju, kai klientas išsiunčia formos duomenis, o serveris visą formą atsiunčia atgal laukus su neteisingais duomenimis pataisęs į kitą formatą, pvz., nustatęs raudoną spalvą. Naudojant AJAX, šio laiko ir pralaidumo kanalo švaistymo galima išvengti.

Tačiau visi geri dalykai turi bent jau keletą trūkumų. Naudojant AJAX ir negalvojant apie dizainą, atsiranda problemų su naršyklės būkle. Paprastas pavyzdys yra tai, kad naršyklės grįžimo atgal mygtukas praranda savo naudą naršant puslapius, sukurtus su AJAX. Pvz., jei lankomas su AJAX kurtas puslapis ir duomenys jame atnaujinami keletą kartų AJAX priemonėmis, tai grįžimo atgal mygtukas jus nuves į ankstesnį puslapį, o ne į ankstesnius to paties puslapio duomenis. Kadangi grįžimo atgal mygtuką vartotojai naudoja gana intuityviai, turi būti imtasi priemonių, norint sukurti gerai matomą alternatyvų sprendimą to mygtuko funkcionalumui pakeisti; pageidauti tam tik tikras nuorodas, kurios iškviečia kokią JavaScript funkciją, padedančią atgauti ankstesnę to paties puslapio būseną. AJAX taikomąsias programas taip pat sunku derinti (angl. debug), nes apdorojimo logika yra patalpinama ir į klientą ir į serverį. Kliento pusės JavaScript kodą galima pamatyti paprasčiausiai paspaudžiant „View Source“ AJAX naudojančiame HTML puslapyje. Prastai sumodeliuota AJAX grįsta taikomąja programa gali pasinaudoti hakeriai arba plagiatoriai.

Appfuse suteikia galimybe naudotis AJAX priemonėmis. Jis savo architektūrose leidžia prijungti tokius AJAX karkasus kaip DOJO, DWR, Prototype, Scriptaculous. Dėl laiko trūkumo AJAX priemonės nebuvo įtrauktos į mūsų kurtą sistemą, o tai galėtų būti darbas ateičiai.

5. IŠVADOS

Šiame darbe mes:

- Išanalizavome Appfuse įrankį kaip šiuolaikišką būdą kurti J2EE internetinius projektus.
- Naudodamiesi šiuo įrankiu sukūrėme Bendrabučių informacinę sistemą.
- Sistemos architektūra sukurta iš geriausios praktikos technologijų.
- Sistema pradėta greičiau kelis kartus nei įprastu būdu.
- Kūrimas vyko pagal šiuolaikišką metodologiją su naujausiomis technologijomis.
- Dėl automatinio generavimo ir pakartotinio panaudojimo technologijų išvengta klaidų
- Siūlome šį įrankį naudoti kuriant J2EE projektus.

6. LITERATŪRA

[1] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, W. M. Turski. Metrics and Laws of Software Evolution - The Nineties View. 1997, 20-21, .

[2] C. Vawter, E. Roman. J2EE vs. Microsoft.NET A comparison of building XML-based web services. 2001.

[3] PHP dokumentacija. [Žiūrėta 2007.10.21]

Prieiga internete http://www.php.net/manual/en/

[4] B. Saikhan. J2EE vs ASP.NET vs PHP. [Žiūrėta 2008.01.25]

Prieiga internete http://www.plentyofcode.com/2007/07/j2ee-vs-aspnet-vs-php.html

[5] E. Gamma, R. Helm, R. Johnson, J. M. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. 1994.

[6] D. Šilingas. Kaip efektyviau projektuoti programinės įrangos architektūrą. Kaunas, 2005.

[7] Java persistence dokumentacija. [Žiūrėta 2008.02.09]

Prieiga internete http://java-source.net/open-source/persistence

[8] Hibernate dokumentacija. [Žiūrėta 2008.02.09]

Prieiga internete http://www.hibernate.org/hib_docs/reference/en/html/index.html

[9] Struts alternatyvos. [Žiūrėta 2008.03.18]

Prieiga internete http://www.roseindia.net/struts/struts-alternative.shtml

[10] Spring dokumentacija. [Žiūrėta 2007.12.16]

Prieiga internete http://www.springframework.org/documentation

[11] R. Johnson. J2EE Development Frameworks. *Computer*, vol. 38, no. 1, pp. 107-110, Jan., 2005

[12] H. Sheil. J2EE project dangers. JavaWorld.com, 2001. [Žiūrėta 2007.10.06]

Prieiga internete http://www.javaworld.com/javaworld/jw-03-2001/jw-0330-ten.html?page=5

[13] Ant įrankio aprašymas. [Žiūrėta 2007.11.12]

Prieiga internete http://ant.apache.org/manual/

[14] Vikipedijos straipsnis „Iterative and incremental development“ [Žiūrėta 2006 11 12]
Prieiga internete http://en.wikipedia.org/wiki/Iterative_and_incremental_development

[15] Straipsnis „SDLC Model: Iterative Model“[Žiūrėta 2006 11 12]

Prieiga internete http://qualityvista.blogspot.com/2005/11/sdlc-model-iterative-model.html

[16] R. C. Martin. Agile Software Developement: Priciples, Patterns, and Practices. Prentice Hall, 2003.

[17] IT konferencijos straipsnis [Žiūrėta 2006 11 12]

Prieiga internete http://www.ktu.edu/lt/mokslas/konf06/konf_02/IT2005/Sekc09.pdf

[18] Agile manifesto tinklapis [Žiūrėta 2006 11 12]

Prieiga internete http://agilemanifesto.org

[19] S. Baird. Agile Modeling. JAV [Žiūrėta 2006 11 12]

Prieiga internete http://www.informit.com/articles/article.asp?p=30428&rl=1

[20] BPI straipsnis. Reikalavimų valdymas. [Žiūrėta 2006 11 12]

Prieiga internete http://www.bpi.lt/text.php?lang=1&item=161&arg=119

[21] Korsaa, Morten. Iterative Software Development - A Practical View, p. 4. 2002.
[22] ISO 9126: The Standard of Reference. [Žiūrėta 2007.05.19]

Prieiga internete http://www.cse.dcu.ie/essiscope/sm2/9126ref.html

7. TERMINŲ IR SANTRUMPŲ ŽODYNAS

IS – Informacinė sistema

PĮ – Programinė įranga

Agile – Programinės įrangos kūrėjų susivienijimas

API – Aplikacijų Programavimo Sąsaja

HTTP – Hiperteksto perdavimo protokolas (angl. akronimas HyperText Transfer Protocol)

HTML – Hiperteksto žymėjimo kalba (angl. akronimas Hypertext Markup Language)

XML – Kalba skirta aprašyti duomenis (angl. akronimias eXtensible Markup Language)

AJAX – Asinchroninis JavaScript ir XML programavimas

CVS – Bylų versijų kontrolės sistema

Java – Programavimo kalba

ASP.NET – .NET platformos programavimo kalba

JavaScript – Internetinių interaktyvių puslapių kūrimo programavimo kalba

Flash – Daugialypis naršyklės priedas

.NET Framework – Microsoft programavimo technologija

RDBMS - Reliacinė duomenų bazių valdymo sistema

ODBC – Duomenų bazių prisijungimo technologija (angl. akronimas Open Database Connectivity)

J2EE – Java 2 verslo lygio technologija (angl. akronimas Java 2 Platform, Enterprise Edition)

JDBC - Duomenų bazių prisijungimo technologija (angl. akronimas Java Database Connectivity)

Oracle – Duomenų bazių produkto prekinis vardas

SQL – Struktūrizuota užklausų kalba (angl. akronimas Structured Query Language)

DB2 – Duomenų bazių valdymo sistema sukurta IBM kompanijos

Servlet – Technologija dinaminių puslapių generavimui

JSP – Technologija, leidžianti dinamiškai generuoti HTML, XML, ar kito tipo puslapius

CGI – Web serverio bendravimo protokolas (angl. akronimas Common Gateway Interface)

PHP – Plačiai paplitusi dinaminė interpretuojama programavimo kalba

mod_perl - Integruotas skriptų kalbų interpretatorius į Web serverius

Enterprise Java Beans - Serverinė duomenų modulių technologija

Model 1 – Internetinių sistemų architektūrinis modelis

Model 2 – Internetinių sistemų architektūrinis modelis

Struts – Karkasinis projektas skirtas Java internetinės sistemoms kurti

Sping – Karkasinis projektas skirtas Java internetinės sistemoms kurti

MVC – Programų architektūrinis modelis (angl.akronimas Model-View-Controller)

8. PRIEDAI

8.1 Publikacija

APPFUSE KARKASO TYRIMAS IR TAIKYMAS

Justas Sperauskas

Informatikos fakultetas, KTU, www.ktu.lt

Šiame straipsnyje mes įvardiname J2EE projekto kūrimo problematiką. Aprašome įrankį Appfuse, kuris yra skirtas programuotojo darbui palengvinti. Appfuse atlieka J2EE internetinio projekto infrastruktūros generavimą, kompiliavimą, veiksmų automatizavimą ir kitas naudingas užduotis. Šį įrankį panaudojome Bendrabučių informacinei sistemai sukurti ir taip įsitikinti Appfuse nauda. Projektas buvo pradėtas greitai ir kodo rėmai leido programuoti šiuolaikiškai pagal trijų lygių architektūros schemą. Nereikėjo gilintis į sudėtingų technologijų, tokių kaip Spring ir Hibernate integravimą. Dokumente aprašome naudojamos priemonės suteikiamus privalumus.

1 Įvadas

Pradedant kurti programinės įrangos produktą sistemos architektas turi svarbią užduotį - parinkti technologijas. Architektai dažnai parenką technologijos tas, kurias geriausiai išmano. Kuriant daugiasluoksnes sistemas naudojami įvairūs jau esami komerciniai arba atviro kodo karkasai, technologijos. Kad ir kokių technologijų derinį jis pasirinktų - priemones tenka apjungti ir nuolat konfigūruoti jų tarpusavio sąryšius. Kuriant didelius, daugiasluoksnius, internetinius projektus reikalingas masyvas įvairiausių programinių servisų: nuo dinaminio puslapio HTML generavimo iki verslo domeno objektų saugojimo. Vietoj to, kad kurtų visus šiuos servisus nuo pradžios, programuotojai naudojasi jau sukurtais ir laisvai prieinamais sprendimais.

Kompanijos Sun J2EE platforma teikia specifikaciją ir bendras sąsajas tiems servisams Java programavimo kalbai. J2EE leidžia programinės įrangos kūrėjams kurti konkurencingus sprendimus. Visiems programinės įrangos kūrimo etapams naudojamos įvairios technologijos. Mes analizuosime etapus nuo programinės įrangos infrastruktūros sukūrimo iki jo galutinio pristatymo. J2EE projekto kūrimo pradžioje daugelis programuotojų susiduria su komponentų ir įvairių modulių integracijos problema. Analizuojama technologija Appfuse yra skirta palengvinti programuotojų ir architektų darbą. Šis karkasas yra junginys daugelio technologijų ir atsakingas už jų sąveiką, darbo automatizavimą, lengvą programinės įrangos kūrimą ant sukurto maketo. Mes sukūrėme internetinę Bendrabučių informacinė sistemą. Naudojome Appfuse architektūros maketą su Struts2, AJAX, Spring, Hibernate kaip pagrindinėmis technologijomis. Straipsnyje [1] aprašoma dalis iš minėtų karkasų ir jų panaudojimo privalumai kuriant J2EE projektus.

Įvairiuose šaltiniuose teigiama, kad pradedant naują J2EE projektą būtina naudoti karkasus ir taip pat akcentuojama, jog juos sukonfigūruoti yra sudėtinga [2], bei kaip sprendimas nurodomas Appfuse panaudojimas. Šiame straipsnyje nėra minimos Appfuse alternatyvos, nes jų nebuvo rasta. Todėl ir nėra palyginimo su panašiais karkasais. Straipsnyje nebuvo bandoma testuoti analizuojamo karkaso. Straipsnyje pateikiami panaudojimo įspūdžiai ir įrankio suteikiami privalumai.

1.1 Appfuse karkasas

Appfuse yra atviro kodo projektas ir programa, kuri naudoja atviro kodo laisvai prieinamus jau sukurtus karkasus bei įrankius, ir leidžia internetinių projektų greitą ir efektyvų kūrimą Java platformai. Appfuse naudodamas Maven2 įrankį gali sukurti projekto skeletą, kompiliuoti, testuoti, generuoti kodą, įdiegti ir atlikti begalę kitų užduočių. J2EE projekto problematikos yra viešai keliamos ir nagrinėjamos [3]. Programuotojams, kuriems yra tekę patiems kurti visą J2EE internetinio projekto infrastruktūra, žino, kad tai yra varginantis ir didelio tikslumo darbas. Šis darbas yra beveik panašus visiems internetiniams projektas, kur tenka apjungti tokias technologijas kaip JavaServer Faces (JSF), Spring MVC, Struts, Tapestry, arba WebWork. Dėl to sukurtas Appfuse, kuris paremtas geriausios praktikos sprendimais. Appfuse gali sutaupyti apie dvi savaites programuotojo darbo. Nereikia rūpintis atviro kodo karkasų konfigūravimu, nes tai padaroma už mus. Projektas yra sukonfigūruojamas darbui su duomenų baze, įdiegiamas į aplikacijų serverį ir leidžia vartotojo prisijungimą. Tinkamų technologijų parinkimas tai tas pats kaip reikiamų įrankių pasirinkimas darbui. Šis karkasas pasirinktas analizei, nes yra unikalus J2EE pasaulyje.

Appfuse technologijos gali generuoti: projekto infrastruktūros veikianti maketą, internetinio puslapio failus, duomenų bazės lenteles, biznio logikos failus, konfigūracijas, testinius duomenis, testus. Taip pat gali atlikti šiuos veiksmus: kompiliuoti, parsiųsti bibliotekas, užkrauti duomenų bazės duomenimis, testuoti, diegti, paleisti veikiantį internetinį projektą, rodyti sisteminius pranešimus ir testavimo ataskaitas.

Su Maven galima sukurti projekto failą naudojamai kūrimo aplinkai. Tinkamos kūrimo aplinkos Appfuse projektams yra:

- Eclipse,
- IDEA,
- MyEclipse,
- NetBeans.

AppFuse integruotų programavimo aplinkų suderinamumas gali būti dviejų tipų:

• Galima sugeneruoti projekto failus daugumai kūrimo įrankių,

• Integruotos kūrimo aplinkos turi priedėlius leidžiančius dirbti su Maven sistema ir taip vykdyti Appfuse užduotis.

2 Metodo detalizacija

2.1 Infrastruktūros sugeneravimas

Appfuse siūlo [4] pasirinkti vieną iš keturių architektūrų ir sugeneruoti visą projekto infrastruktūrą. Mūsų eksperimente naudota architektūra yra Struts2 pagrindo. Visą architektūrą sudaro begalė šiuolaikiškų įrankių: Spring, Hibernate, JUnit, Canoo WebTest, DbUnit ir kt.. Projektas sugeneruojamas su jau veikiančiomis šiomis dalimis: autentifikacija ir autorizacija, vartotojų valdymu, vartotojo prisiminimu, slaptažodžio priminimu, vartotojų užsiregistravimu, SSL saugumu, el. pašto galimybėmis, išvaizdos, puslapių dalių dekoravimu, failų įkėlimu. Appfuse architektūrų maketai atitinka šiuolaikišką MVC modelį.

2.2 Automatizavimas

Appfuse naudojasi Maven2 ir gali atlikti daug automatizavimo veiksmų: kompiliavimas, testavimas, diegimas. Taip pat Appfuse gali sugeneruoti biznio logikos klases, vartotojo sąsają, duomenų bazės struktūra, testinius duomenis ir testus, ir objektinės klasės. Projekto failai, reikalingi integruotoms programavimo aplinkoms, gali būti taip pat sugeneruoti naudojant vieną Appfuse komandą. Visi veiksmai gali būti atlikti iš komandinės eilutės arba naudojantis integruotos programavimo aplinkos funkcijomis. Kiekvienas vykdomas veiksmas yra lydimas detaliais pranešimais.

2.3 Saugumas

Autorizavimo ir autentifikavimo, vartotojų rolių, prisijungimo išsisaugojimo funkcionalumas yra naudojamas daugelyje internetinių projektų. Todėl iš naujo kurti šio lygio funkcionalumo neverta, kai yra profesionalų sukurta sistema Acegi Security. Ši sistema yra puikiai išdirbta ir atitinka šiuolaikinius internetinių projektų saugumo reikalavimus. Todėl saugumas ir vartotojų autorizavimas bei autentifikavimas yra įtraukiami į visus Appfuse sugeneruotus projektus. SSL funkcionalumas yra taip pat įdiegiamas pagal nutylėjimą.

2.4 Testavimas

Testavimui kartais skiriama mažai dėmesio. Tai ne reiškia, kad testavimui nėra skiriama dėmesio visai. Publikacijose daug rašoma apie principą „testuok prieš programuodamas“, bet realybėje į testavimą žiūrima kaip į projekto užbaigimo prailginimą. Bet taip manyti yra neteisinga, nes jeigu yra rašomi testai prieš programavimą, galima rasti tai, kas paspartins projekto užbaigimą. Appfuse yra pilnai į testavimą orientuotas kūrimo įrankis. Jis nesugeneruoja instaliuojamo projekto, jeigu jame yra randamos klaidos. Testavimui naudojamos technologija jUnit, atlieka funkcinių vienetų testavimą, o Canoo WebTest – atlieka internetinės sąsajos testavimą. Naudojant kodo generavimą, pilnai sugeneruojami veikiantys testai visiems funkciniams vienetams ir internetinės sąsajos užduotims.

2.5 Pagalba

Appfuse projektas yra aktyvus ir šiuo metu, turi plačią dokumentaciją. Jis yra atvirai prieinamas, bei redaguojamas aktyvistų ir viso pasaulio. Bendruomenė yra plati, nes Appfuse naudojamas tiek verslo, tiek kituose sektoriuose. Pats projekto įkūrėjas Matt Raible teigia [5], kad bendruomenė yra draugiška ir prisideda prie produkto vystymo.

3 Eksperimentas

Appfuse karkasas buvo pritaikytas Bendrabučių informacinei sistemai kurti. Programa pasižymi tokiomis ypatybėmis kaip daugiakalbiškumas, vartotojų grupės, saugumas. Pradėjome nuo pasirinkimo technologijų masyvo pagal architektūras iš Appfuse. Pasirinkome Struts 2 Basic į kurį įeina technologijos: Struts2, Spring, Hibernate, Maven2, JUnit ir kt.. Architektūra yra trijų lygių sistema. Hibernate naudojamas kaip duomenų lygio karkasas [6]. Spring yra vidurinio sluoksnio karkasas ir naudojamas kaip alternatyva [7] EJB. Struts2 yra patogus [8] prezentacijos sluoksnio karkasas. Sukurtas projektas naudojantis viena komandinės eilutės komandą (mvn archetype:create -DarchetypeGroupId=org.appfuse.archetypes -DarchetypeArtifactId= appfuse-basic-struts -DremoteRepositories=http://static.appfuse.org/releases -DarchetypeVersion=2.0.1 -DgroupId= eu.justas.bis -DartifactId=bis). Sukuriama duomenų bazė ir jos vartotojas. Prisijungimo konfigūracija įrašomą į projekto pom.xml failą. Turime veikianti projektą. Jį galima pamatyti įvykdžius komandinės eilutės komandą (mvn jetty:run). Paleidimo metu galima nurodyti kokio lygio pranešimai bus rodomi komandinėje eilutėje.

Tolimesnis sistemos kūrimas vyko sukuriant objektines domeno klases. Anotacijomis nurodėme, kad klasės bus saugomos duomenų bazėje ir kokios atributų savybės. Kiekvienam objektui vykdomas verslo logikos veiksmų, prezentacijos, testų failai, bei testiniai duomenys (mvn appfuse:gen -Dentity=Objektas). Turint sugeneruotus logikos ir prezentacijos failus, juos keitėme pagal mūsų reikalavimus. Taip gavome veikiančią sistemą.

Eksperimento metu su kuriama programine įranga norėjome pamatyti Appfuse privalumus ir juos išbandyti. Programinės įrangos kūrimas atitiko Agile principus, nes buvo kuriama su iteracijomis, papildant funkcionalumą, bei viskam rašomi testai.

4 Rezultatai

Akivaizdu, kad Appfuse yra naudingas ir paspartina programinės įrangos kūrimo startą ir tolesnį vystymą. Programuotojui nereikia vargi pradedant ir jungiant skirtingų lygių karkasus, tokius kaip Hibernate ir Spring. Jis gali kurti vartotojo aplinką nežinodamas vidinės Spring ir Hibernate logikos. Sukurta programinė įranga turi gerą struktūrą ir gali būti lengvai plečiama. Kuriant projektą buvo pastebėta, kad SQL užklausos yra perteklinės, sugeneruotame kode sunkiau orientuotis, daug konfigūravimo atliekama XML failuose. Automatiškai generuoto kodo yra akivaizdžiai daugiau negu parašytų programuotojas. Sugeneruoti testai atlieka tik bazinių funkcijų veikimo patikrinimą. Appfuse neapsaugo nuo technologijų pasenimo, tik pateikia rekomendacijas kaip pereiti prie naujesnių.

5 Išvados

Šiame straipsnyje mes:

• Išanalizavome Appfuse įrankį kaip šiuolaikišką būdą kurti J2EE internetinius projektus.

• Naudodamiesi šiuo įrankiu sukūrėme Bendrabučių informacinę sistemą.

• Nustatėme, kad projektą galėjome pradėti greičiau su Appfuse ir gauti architektūrą iš geriausios praktikos technologijų.

• Straipsnyje aprašėme kūrimo įspūdžius ir įrankio teikiamus privalumus.

• Siūlome šį įrankį naudoti kuriant J2EE internetinius projektus.

• Ateityje būtų galima ištirti įrankio trūkumus ir pasiūlyti sprendimus jiems.

Literatūros sąrašas

[1] R. Johnson, J2EE Development Frameworks. Computer, vol. 38, no. 1, pp. 107-110, Sausis, 2005

[2] J. Dutta, P. Fodor, A Systematic Approach to Web-Application Development. 2007.

[3] S. Demuth, Bringing J2EE into your Established (Slightly Hidebound) Organization. CTO, Artemis Alliance.

[4] Appfuse: http://appfuse.org/display/APF/AppFuse+QuickStart

[5] M. Raible, Seven simple reasons to use Appfuse. 2006

[6] Hibernate: Hibernate Reference Documentation version 3.2.2.

[7] J. Arthur, S. Azadegan, Spring Framework for rapid open source J2EE Web Application Development: A case study. 2005.

[8] C. Cavaness, Programming Jakarta Struts. Knyga. 2004.

Appfuse framework

In this paper we tried to look at J2EE project start and development problem. We propose the solution Appfuse. The Appfuse is open source project and program which help fast and efficient software development. The principles are illustrated by creating the Information System for Hostels.