KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Edgaras Dulskis

Dinamiškų užduočių variklio modelis socialinėms kompiuterinėms simuliacijoms

Magistro darbas

Darbo vadovas
doc. dr. A.Ostreika

Kaunas
2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Edgaras Dulskis

Dinamiškų užduočių variklio modelis socialinėms kompiuterinėms simuliacijoms

Magistro darbas

Recenzentas
lekt. dr. G. Mikulėnas
2011-05-

Darbo vadovas
doc. dr. A.Ostreika
2011-05-

Atliko
IFM-9/1 gr. stud.
Edgaras Dulskis
2011-05-23

Kaunas
2011

Turinys

A model of dynamic quest engine for social simulations
Summary

During the past few years a couple innovative projects, aiming at better social integration of certain target groups, has been initiated by public offices in Lithuania. These projects require interactive virtual environments, where users could safely test distinct behaviors and learn from corresponding outcomes, while making their own decisions. The educational content is conveyed as dynamic and text based communication between the user and non-player characters, who resides in a computer game.

This document introduces a possible solution for implementation of such dynamic content for a social simulation. The model of dynamic quest engine, that supports a broad range of non-linear adaptive tasks (quests) and dialogs, is described here. This model has been built upon analysis of similar existing engines, key concepts of non-linearity as well as specific requirements of actual underway social projects.

The usability of the suggested model was tested by applying it in implementation of dynamic quest engine itself. The same engine implementation was later successfully used in two separate social projects. Created engine now powers dynamic educational content, written by expert writers and currently accessible for target groups in Lithuania.

1 . ĮVADAS

Kompiuterinė simuliacija[1] – tai kompiuterio programos panaudojimas, kuriant realaus pasaulio aspektus atspindintį modelį, kurio būsena priklauso nuo laiko ir kitų parametrų. Kompiuterinės simuliacijos tapo reikšminga priemone daugelio realių šakų matematiniam modeliavimui – skaičiuojamosios fizikos, astrofizikos, chemijos, biologijos, taip pat ekonominių, psichologinių ir socialinių sistemų. Simuliacijos naudojamos kuriant ar išbandant naujus modelius, technologijas, siekiant nustatyti jų veikimo tendencijas. Realių procesų simuliacijos kompiuteryje yra pakankamai tikslus, be to dažnai pigesnis, saugesnis, mažiau fizinių pastangų ar kitų resursų naudojantis sprendimas, nei šio modelio išbandymas realiame pasaulyje.

Su šiuo magistro darbu susijusi tik vieno tipo simuliacijos - socialinės edukacinės sistemos, kurių tikslas, bendru atveju, yra galimybė žmonėms saugiai išbandyti įvairius elgesio modelius tariamai realioje gyvenimo situacijoje. Tokiose sistemose dalyviai prisiima tam tikrus vaidmenis, o jų veiksmai įtakoja tolesnę simuliacijos scenarijaus vystymosi kryptį bei pačių dalyvių rezultatus. Tobulėjant kompiuterinėms technologijoms ir sudėtingėjant programinės įrangos produktams, atsiveria galimybės sukurti pažangias mokomąsias sistemas, taip pat didėja ir reikalavimai joms. Sparčiai vystantis įvairioms e-sistemoms nelieka užmirštos ir socialinę atskirtį patiriančios žmonių grupės – valstybiniu mastu ir remiant Europos Sąjungai sudaromi ir vykdomi projektai, kurių tikslas yra tų asmenų socialinės atskirties mažinimas ir integravimas į Lietuvos darbo rinką, pasitelkiant kompiuterines programas.

Natūralu, kad mokomosios simuliacinės priemonės (kitaip – simuliacinio žaidimo) funkcinė kokybė yra tuo geresnė, kuo tikroviškiau vaizduoja realaus gyvenimo aspektus bei kuo daugiau pasirinkimo laisvės vartotojui suteikiama. Socialinės krypties projektų reikalavimuose (reikalavimuose kompiuteriniam produktui) pabrėžiama vartotojo pasirinkimų svarba. T.y., kokiame scenarijaus taške sistemos vartotojas atsiduria turi priklausyti nuo jo paties pasirinkimų. Be to, alternatyvių pasirinkimų turi būti pakankamai daug ir įvairių. Tokiu būdu įgyvendinamas scenarijaus dinamiškumas – kiekvienas pasirinkimas keičia arba gali keisti tolesnę įvykių eigą bei iškviečia atitinkamus rezultatus.

Esame įpratę, kad realiame gyvenime egzistuojantis asmuo (objektas) reaguoja priklausomai nuo jį veikiančios informacijos (veiksnių). Suprantama ir įprasta, kad žmogus į skirtingą informaciją reaguos atitinkamai, įvertindamas tos informacijos turinį ir reikšmingumą

[1] „Kompiuterinė simuliacija – realaus pasaulio vaizdavimo kompiuterine programa būdas [23].

jam. Kiekvienas atliktas veiksmas gali įtakoti aplinkinį pasaulį bei patį veiksmo vykdytoją. Norint sukurti tokias kompiuterines simuliacijas, kurios padėtų socialinę atskirtį jaučiantiems asmenims integruotis į visuomenę, mokintų juos elgesio taisyklių ir padėtų įgyti patirties, praverčiančios tikrame gyvenime, reikia esminę bendravime egzistuojančios dinamikos dalį perkelti į virtualią erdvę. Kadangi projektų dalyviams pateikta medžiaga (scenarijus) turi atitikti tikslinės grupės poreikius, ji turi būti kruopščiai paruošta kompetentingų socialinės ir psichologinės krypties ekspertų. Šis faktas reiškia, kad ieškant efektyvių mokomosios priemonės projektavimo būdų, automatiškai generuojamo turinio ar neapibrėžtai komunikuojančio dirbtinio intelekto sritys nenagrinėjamos.

Pagrindinis šio magistro baigiamojo darbo tikslas yra aprašyti modelį, kuriuo remiantis galima suprogramuoti veiksmo-pasekmės principu veikiantį dinamiškų užduočių variklį ir tinkamai formuoti scenarijų šiam varikliui. Užduotis čia suprantama kaip tikslo ir nurodymų rinkinys, aprašantis ką virtualioje realybėje vartotojas turi ar gali atlikti, kad pasiektų tą tikslą. Siekiama suteikti galimybę adaptyvaus, nelinijinio scenarijaus įgyvendinimui, t.y nuo vartotojo pasirinkimų turi priklausyti, kokios užduotys jam konkrečiu momentu pateikiamos bei kokiu būdu jos bus išspręstos. Scenarijų pakeisti ir papildyti turi būti leidžiama be visos sistemos taisymo, taigi ir po sistemos sukūrimo.

Formuojant idėjas ir formuluojant apibrėžimus buvo atsižvelgta į realius Lietuvoje vykdomų kompiuterizuotų socialinių projektų „Mano Kelias 2“ bei „Renkuosi Aš!“ reikalavimus, specifiką - eksperimentiškai variklio modelis išbandytas būtent minėtuose realius vartotojus turinčiuose visuomeninės krypties projektuose, kurių sudėtingumas lenkia iki šiol kurtus mokomuosius žaidimus Lietuvoje. Šių inovatyvių projektų teminės dalys apima socialinę, psichologinę, aktyvios karjeros planavimo, įsidarbinimo, laisvalaikio planavimo, vertybių ir pan. sritis. Projektų tikslinės grupės yra socialinę atskirtį jaučiančios visuomenės grupės – į laisvę išeisiantys nuteistieji, reabilituoti narkomanai, neįgalūs žmonės, vaikų globos namų auklėtiniai. Tikimasi, kad pagal sudarytą planą naudodamiesi mokomąja socialine priemone šie asmenys vėliau sėkmingai integruosis į realią visuomenę ir darbo rinką, o įvertinę įvairių elgesio modelių pasekmes virtualioje simuliacijoje, realybėje rinksis tuos, kurie neprieštarauja etikos ir moralės normoms.

Būtent dėl inovatyvių „Mano Kelias 2“ ir „Renkuosi Aš“ projektų atsirado poreikis dinamiškų užduočių variklio atsiradimui Lietuvoje. Kita vertus, siekta, kad pasiūlytas modelis būtų universalus, tai yra palaikytų kiek įmanoma įvairesnes užduočių struktūras ir tiktų kuo įvairesniems projektams, nepriklausomai nuo jų prasminio turinio, neužkertant kelio

pakartotiniam panaudojimui ateityje. Pastaroji savybė savaime formuluoja ir dar vieną reikalavimą, kurį bandyta aprašytame modelyje išlaikyti – tai paprastumas, suprantamumas. Sudėtingais principais paremtas modelis apsunkintų tiek jo įgyvendinimą, tiek naudojimąsi sukurta sistema - daugiausiai dėl to, kad mokomųjų priemonių scenarijaus autoriai, kurie bendru atveju nėra IT specialistai, turi gebėti kurti tinkamą šiai sistemai scenarijų, t.y. sistemos principai turi būti suprantami ir žmonėms be programavimo žinių.

Siekiant nustatyto tikslo buvo atliktas kokybinis tyrimas, kurio objektas yra būdai ir priemonės, nurodančios, kaip adaptyvų scenarijų, kurio pagrindas yra tekstiniai dialogai tarp vartotojo ir virtualios erdvės objektų, būtų galima perkelti į tą erdvę. Tyrimo metu nagrinėtos tokius būdus aprašančios publikacijos, moksliniai straipsniai, rasti vykdant detalią paiešką pagal susijusius terminus. Taip pat remtasi konkrečių susijusių produktų bei įrankių aprašais. Produktų kūrimo metodai ir naudoti įrankiai daugeliu atvejų lieka viešai nepaskelbti, ypač kai simuliacines priemones kuriančios įmonės ar organizacijos tikslas yra finansinė nauda. Dėl šios priežasties, siekiant suprasti dinamiško scenarijaus realizacijos principus, teko nagrinėti ir bendras netiesiškumo įgyvendinimo problemas ir galutinius produktus, pasižyminčius dominančiomis savybėmis. Iš didelės kompiuterinių simuliacijų srities aktualūs tik tie produktai, kuriuose vartotojams esminiai nurodymai pateikiami iš anksto paruoštu tekstiniu ar kalbiniu pavidalu, o tų nurodymų vykdymas nėra linijinis ir vienareikšmis - priklauso nuo vartotojo įvesties, t.y. pasirinkimų.

Sudarius modelį atliktas ir eksperimentinis tyrimas, kurio tikslas yra patikrinti modelio tinkamumą (funkcionalumą ir efektyvumą), pritaikant modelį dviejų minėtų realių projektų realizacijose. Šiuo atveju modelio tinkamumą įvertinti gali ne galutiniai sistemos vartotojai - tikslinių grupių asmenys -, o įmonių ar organizacijų darbuotojai, įgyvendinantys tuos projektus. Buvo lyginamas laikas, kurį scenarijaus kūrėjai bei jo talpintojai užtruko scenarijaus formavimui prieš modelio idėjų taikymą ir remiantis pasiūlytomis idėjomis. Sistemos vartotojams principai, kuriais vadovaujantis sistema sukurta ir yra vystoma, nėra aktualūs, nes jie mato tik galutinį produktą. Tačiau, tinkamai parinktas modelis nulemia ir gauto užduočių variklio kokybę, tad yra būtinas ir galutinio produkto sėkmei jo vartotojų akyse.

2. DINAMIŠKO SCENARIJAUS ĮGYVENDINIMO SPRENDIMŲ ANALIZĖ

Per pastaruosius dvidešimt metų kompiuterinių žaidimų, įskaitant ir mokomųjų simuliacijų sferą, kūrimas suklestėjo – įvykdyta ir vykdoma nesuskaičiuojama daugybė projektų, o naujai kuriamose kompiuterizuotose interaktyviose programose taikomos technologijos ir sprendimai nuolat tobulėja. Norint sudaryti realiam naudojimui tinkamą kompiuterinės simuliacijos užduočių variklio modelį, kurio užduotys vykdomos komunikuojant su virtualiais personažais, svarbu susipažinti su jau egzistuojančiais teoriniais ir praktiniais pavyzdžiais. Net jei nėra sukurtas toks modelis, kuris visu šimtu procentų tiktų užsibrėžto tikslo įvykdymui, daugelis bendrų problemų bei jų sprendimo principų išlieka. Analizės metu apžvelgtos susijusios problemos ir jų sprendimai, taip pat projektavimo principai, modeliai ir jų galutiniai produktai, galintys būti naudingi apibrėžiant variklio modelį.

Socialinių mokomųjų priemonių sritis iš techninės pusės nedaug skiriasi nuo komercinių pramoginių žaidimų vaidmenimis. Mokomąjį pobūdį suteikti gali pats prasminis turinys, bei koncentracija į vartotojo profilį, tačiau dinamikos palaikymo problemos iš esmės yra bendros.

Nagrinėjamos kelių tipų scenarijaus dinamikos problemos: pirmasis tipas yra scenarijaus netiesiškumas, arba nelinijinis scenarijus, reiškiantis kompiuterinės sistemos vartotojo pasirinkimo laisvę ir to pasirinkimo įtaką tolesnei įvykių eigai (adaptyvumą), o antroji dinamikos komponentė yra galimybė neribotam ir nesudėtingam scenarijaus vystymui ir taisymui - scenarijaus plečiamumui.

Užduoties ir scenarijaus sąvokos darbe yra sugretintos – laikomasi požiūrio, kad kompiuterinėse simuliacijose scenarijus atskleidžiamas vartotojams vykdant pateiktas užduotis (misijas), kurios kartu suteikia tiems vartotojams tikslą ir kryptį.

2.1 Scenarijaus netiesiškumas ir su juo susijusios problemos

Linijinis scenarijus kuriamas taip, kad įvykius stebinčio ar juose dalyvaujančio asmens (kompiuterinių žaidimų atveju - žaidėjo) progresas judėtų tik per iš anksto nustatytus taškus ir tik vienu būdu. Tokiu būdu sukurtos priemonės galutinio tikslo pasiekimas kartu reiškia, kad vartotojas atsidūrė galutiniame vienareikšmės grandinės, kurios visi taškai galėtų būti vienoje tiesėje, taške. Toje tiesėje parinkus bet kuriuos du scenarijaus taškus A ir B, egzistuos tik vienintelis būdas nusigauti iš A į B [1]. Akivaizdūs tokių scenarijų pavyzdžiai realybėje – filmai

ir knygos.

Kompiuterinės interaktyvios priemonės suteikia galimybę į scenarijų įtraukti ir vartotojo pasirinkimus. Nors vykdoma vis daugiau mokomojo pobūdžio kompiuterinių projektų, paremtų dialogais su KP [8][9], visgi pramoginių žaidimų rinkoje modernių ir sudėtingų tokios srities pavyzdžių rasti yra gerokai paprasčiau (pavyzdžiui, Bioware, Bethesda kompanijų žaidimai vaidmenimis). Šiuolaikiniuose virtualiuose pasauliuose linijinį scenarijų, t.y vienpusę sąveiką dažnai keičia abipusė vartotojų ir kompiuterinių personažų sąveika, kurios galimi variantai [2] yra :

- Iš anksto nustatyti pokalbių medžiai, suteikiantys vartotojui galimybę rinktis norimą dialogo kelią. Tiek personažų tekstas, tiek galimi vartotojo pasirinkimų variantai yra parašyti scenarijaus kūrėjų ir kiti pasirinkimai negalimi; Toks scenarijaus tipas paplitęs tarp nuotykinių žaidimų, pavyzdžiui, sutinkamas populiariame „FunCom“ kompanijos „The Longest Journey“ žaidime;
- Raktiniais žodžiais paremta sąveika. Tokioje sistemoje kompiuteriniai personažai turi prieigą prie bendrų žinių sistemos, o vartotojas kreipiasi tų žinių pagal aktualius raktažodžius. Atsakymai į tokias užklausas yra paruošti scenarijaus kūrėjų ir dažnai pasikartoja tarp skirtingų personažų. Panašias sistemas, pavyzdžiui, naudoja kompanija „Bethesda“ savo „The Elder Scrolls“ serijos vaidmenų žaidimuose;
- Adaptyvūs pokalbių medžiai. Tai labiau įtraukiančios sistemos, naudojančios milžiniškus scenaristų paršuotų dialogų kiekius ir kiekvienu atveju pateikia keletą pasirinkimo variantų. Tokią sąveiką galima laikyti prieš tai nurodytų dviejų junginiu, be to, pokalbių šakos nuolat kinta priklausomai nuo kintančių veiksnių, tokių kaip įvykdytos žaidėjo užduotys.

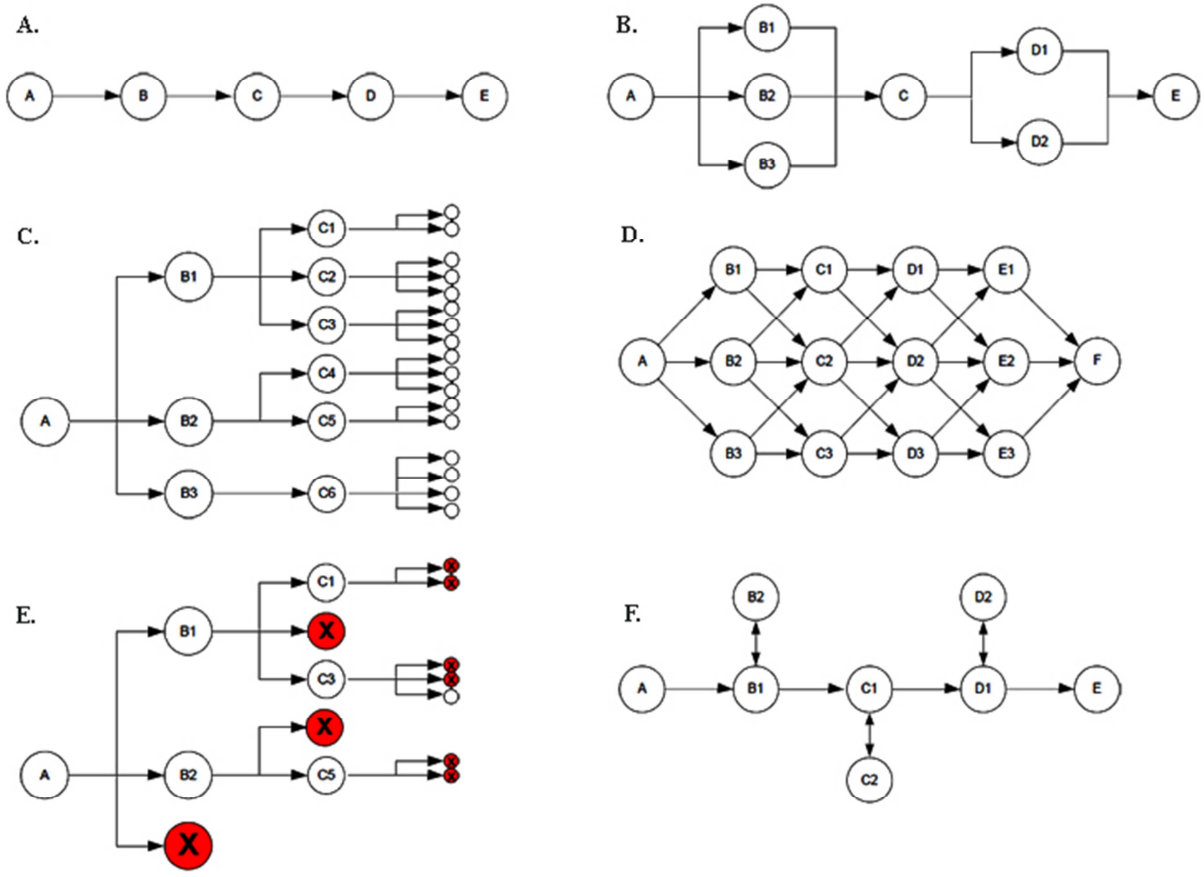
Taigi, nelinijinis scenarijus suteikia pasirinkimo iš tam tikros (statiškos arba kintančios) aibės kelių, laisvę. Dinamiškos sąveikos skirstymas į tris atvejus yra tiktai sąlyginis ir nelinijinį scenarijų galima išskirti į du tipus pagal kitą kriterijų - atskirų jo elementų (misijų) tarpusavio nuoseklumą:

1) Atviri pasauliai (angl. *open words*), kuriuose atskiras bendro siužeto dalis vartotojas gali rasti ar netikėtai su jomis susidurti bet kokia tvarka;
2) Scenarijų keliai, kurie gali susijungti ir vėl išsiskirti – šakotas scenarijus.

Scenarijaus medžių formavimas gali būti labai skirtingas ir reikalauja tarpusavio atskyrimo – galimi atvejai, kur galiausiai šakos veda į tą pačią scenarijaus medžio pabaigą ir priešingai, kitose

sistemose pasirinkimai gali lemti patekimą į visiškai skirtingas pabaigas. 1 pav. vaizduojami galimi scenarijaus šakojimosi tipai [3]. Grafų viršūnės čia žymi svarbius scenarijaus etapus: užduotis, įvykius ar pan. Vienintelė C raide pažymėta schema turi daugiau nei vieną pabaigą, o kiti scenarijų grafai galiausiai vartotoją nukreipia į tą pačią pabaigą.

Mažesnę įvykių plėtrą vaizduojantys atvejai 1 pav. vyrauja ne atsitiktinai - tokia tendencija išlieka ir kuriamuose produktuose. Nepaisant nelinijinio scenarijaus patrauklumo vartotojui, daugelis kompiuterinių žaidimų, įskaitant ir mokomąsias priemones, užduočių dinamikos elementų turi nedaug arba jų visai nėra. Pagrindinė to priežastis – dinamiško scenarijaus sistemų kūrimas yra sudėtingas ir brangus dėl programavimo ir turinio valdymo iššūkių [4]. Kuo daugiau pasirinkimų vartotojui suteikiama, tuo daugiau laiko ir kitų resursų užtrunka kiekvieną galimybę, bei jos įtakotą scenarijaus šaką suprojektuoti. Jeigu kompiuterinis žaidimas, su medžio tipo scenarijumi (1-C pav.) du kartus per visą žaidimą suteikia tik po 3 skirtingus pasirinkimus, reikės jau 8 skirtingų pabaigų. Dėl laipsniško sprendimų medžio augimo projektuotojų komanda gali susidurti su sunkumais valdant didelį kodo ir scenarijaus kintamųjų kiekį.



1 pav. Scenarijaus šakojimosi tipai [3]

Problema šiuo atveju ne tik tai, kad kiekvienai iš pabaigų sukurti reikės papildomų resursų,

bet ir tai, kad kiekvienas žaidėjas patirs tik aštuntadalį žaidimo pabaigos turinio. Panašiai atsitinka, kai sukuriami neprivalomi (1 pav., F tipas) arba alternatyvūs (1 pav., B tipas) žaidėjo pasirinkimai – jeigu iš tašo A į tašką C patenkama atlikus tik vieną iš trijų (B1, B2, B3) veiksmų, tai atlikęs B1 veiksmą vartotojas gali iki žaidimo pabaigos nebesutikti B2 ir B3 pasirinkimų bei juos vaizduojančių scenų. Tai mažina tokių pasirinkimų kūrimo finansinį pagrįstumą bei motyvaciją juos kurti – jeigu tie scenarijaus taškai gali likti nepastebėti, kyla klausimas, ar juos iš viso verta kurti.

Kita vertus, tokia alternatyvų ir skirtingos patirties savybė gali paskatinti vartotoją pakartoti žaidimą ir išbandyti kitus kelius. Dėl žaidimų peržaidimo naudos ginčijamasi ir jis nėra pagrindinis nelinijinio scenarijaus kūrimo tikslas – didelė vartotojų dalis nepasiekia žaidimų pabaigos nė karto [1, 129 p.], tad logiška, jog jie nepradės žaisti dar kartą. Priežastys, dėl kurių žaidėjai nepasiekia tų pabaigų dažnai susijusios su užstrigimu situacijoje: pateiktas galvosūkis per sudėtingas, nepavyksta rasti išėjimo iš aplinkos ir pan. Suteiktas didesnis pasirinkimo laipsnis, individualizuojantis vartotojo patirtį yra ne tik pagrindinis netiesiško scenarijaus tikslas, bet ir sumažina tikimybę vartotojui užstrigti tam tikrame taške, nes bus įveikta kažkuri vartotojui priimtinesnė ir suprantamesnė alternatyva.

Dar viena su alternatyvų kūrimu susijusi kliūtis yra skirtingų kelių lygiavertiškumo balansavimas – jeigu B1 pasirinkimas leistų žymiai lengviau pasiekti C tašką, nei B2 arba B3 keliai, tikėtina, kad niekas nevargs su jais ir visuomet rinksis B1. Todėl prastai paruošti pasirinkimai nedaug skiriasi nuo linijinio scenarijaus įgyvendinimo. Apskritai, net jei sistema techniškai palaiko lygiagrečių pasirinkimų galimybę, dar nereiškia, kad galutinis produktas turės nelinijinį scenarijų, kadangi tokio turinio sukūrimas nėra paprastas procesas.

Šalia paminėto alternatyvų balansavimo sunkumo, reikia pastebėti, kad nėra lengva sukurti pasakojimą, kuris gali būti papasakotas bet kuria (ar įvairia) tvarka ir vis tiek išlaikytų savo prasmę. Scenaristai gali vengti nelinijinio pasakojimo, nes kiekviena sukurta smulkmena, kiekvienas elementas, jų požiūriu, gali būti kritiškai svarbus istorijos vystymui, tad nepamatytų šakų problema aktuali ir iš kūrybinės pusės.

Keblumų dar prideda scenarijaus kūrimo įrankių poreikis – rašytojams scenaristams naudoti standartines teksto rašymo programas (Microsoft Word, Final Draft ar pan.) nepatogu, kadangi scenarijaus elementai tuose įrankiuose siejami linijiškai. Tuo tarpu nelinijinis scenarijus dažnai yra susietas ne realaus laiko, o sąlygų atžvilgiu – tam tikri scenarijaus elementai atskleidžiami tik kai žaidėjas įvykdo atitinkamą veiksmą, nepaisant to, kiek laiko praėjo. Išvardinti tokius neapibrėžtu laiku galinčius vykti įvykius nuosekliai - nuo dokumento viršaus žemyn -, nėra

adekvatu.

Visi minėti sunkumai yra įveikiami ir virtualaus pasaulio dinamika tampa vis svarbesniu kompiuterinių žaidimų sėkmę įtakojančiu parametru. Kuriami modernūs rašytojų darbą lengvinantys įrankiai [5][6][7], supaprastinantys netiesiškų užduočių kūrimo ir užrašymo procesus, pasitelkiant grafines priemones ir padedantys scenaristams įveikti pagrindines kliūtis. Šiame darbe sprendžiama ne tų specializuotų įrankių problema, bet vidinė sistemos logika, techniniai sprendimai, kurie suteikia prielaidas nesudėtingam nelinijinio scenarijaus kūrimui bei tokį scenarijų geba interpretuoti.

2.2 Scenarijaus kalbos

Bendru atveju scenarijaus programavimo kalbos, kitaip - *skripto* kalbos (angl. *script languages*), nuo sisteminių programavimo kalbų skiriasi tuo, kad pirmosios skirtos jau egzistuojančių komponentų sujungimui, komunikacijai, kai sisteminės kalbos yra skirtos tų komponentų – duomenų struktūrų bei algoritmų – sukūrimui [10]. Scenarijaus kalbos yra aukšto lygio ir jomis rašomas kodas yra interpretuojamas (transliuojamas) realiu laiku, t.y nėra kompiliuojamas iš anksto.

Tokia savybė praverčia kompiuterinių žaidimų scenarijui apibrėžti: pasakojamai istorijai, vykdomoms užduotims, įvykių sąlygoms bei kitiems nestatiniams veiksniams. Išorėje parašyti „scenarijai“ (angl. s*cripts*), esantys tiesiog keičiami programinio kodo fragmentai, gali būti įjungti į žaidimą nereikalaujant viso žaidimo perkompiliavimo. Tokiu atveju terminas „scenarijus“ turi kitą prasmę, lyginant su vartotu šiame skyriuje ir žymi tik vieną viso turinio elementą, aprašytą programavimo kalba. Scenarijaus kalbos išryškina skirtumą tarp žaidimo variklio ir paties žaidimo, kodui suteikia moduliškumo ir pakartotinio panaudojimo savybes. Taip pat šių kalbų specifika suteikia galimybę net patiems žaidėjams prisidėti prie žaidimo kūrimo ar modifikavimo.

Taigi, scenarijaus kalba tarnauja kaip sąsaja su žaidimo varikliu: žmogus kuriantis istoriją žino, kodėl tam tikras įvykis turi įvykti ir kas vyks vėliau, o žaidimo variklis žino, kada tas įvykis įvyksta. Scenarijaus kalbos čia aptariamos todėl, kad jos yra ypač dažnas sprendimas įgyvendinant dinamiškus virtualius pasaulius bei jų turinį - užduotis. Dalis kompiuterinių žaidimų naudoja bendros paskirties scenarijų kalbas. Pavyzdžiui, „Sid Meier‘s Civilization“ serijos žaidimai naudoja *Python*, o „Homeworld 2“ ir „World of Warcraft“ naudoja *Lua* [11]. Kita dalis žaidimų turi savo unikalias kalbas, sukurtas būtent tam produktui ar jų serijai. Tokių pavyzdžių

galima rasti daug:

- „Elder Scrolls IV: Oblivion“ naudoja „TES Script“;
- „BioWare Dragon Age“ naudoja į C panašią įvykiais paremtą scenarijaus kalbą;
- „Neverwinter Nights“ naudoja „NWScript“.
- „Realm Crafter“ naudoja to paties pavadinimo kalbą;

Scenarijaus kalbų populiarumo tendencija išlieka, net ir resursų reikalaujantis naujų kalbų kūrimas kompiuteriniams žaidimams yra dažnas reiškinys, kas leidžia daryti prielaidą, jog tokių kalbų privalumai tiems produktams lenkia jų trūkumus. Pateikiami apibendrinti scenarijaus kalbų privalumai, galintys lemti jų, kaip priemonės kompiuterinio žaidimo scenarijaus aprašymui, populiarumą:

- Visos sistemos perkompiliuoti nereikia, kas leidžia greitą vystymą ir paprastą kodo pateikimą, tuo pačiu ir spartų žaidimo turinio balansavimą;
- Scenarijaus kalbos lengvai integruojamos su egzistuojančiomis technologijomis, pavyzdžiui, kitomis programavimo kalbomis [12]. Taip pat lengva jas perkelti į kitas platformas (pavyzdžiui iš asmeninio kompiuterio į DreamCast);
- Tokias kalbas, dėl jų aukšto lygio ir susiaurintos paskirties, paprasta išmokti ir naudoti, t.y nebūtina būti programavimo ekspertu. Be to, kokybiška scenarijaus kalba apsaugo nuo nepatikimo kodo problemų, lyginant su programavimu sistemine kalba, kadangi neteisinga scenarijaus sintaksė gali būti apdorojama tiesiog išvedant tekstinį pranešimą ir sistema nenulūš;
- Paprasta sukurtą elgesį ar dialogą panaudoti pakartotinai, pavyzdžiui, priskirti kitam kompiuteriniam personažui – tai atliekama kopijuojant tą pačią kodo dalį;
- Dinamiškas programavimas – galimybė scenarijus rašyti sukurto sistemos naudojimo metu, net ir paleidus programą; Tokie scenarijai gali būti užkraunami pagal pareikalavimą, t.y. ne visas kodas turi būti patalpintas atmintyje o tik reikalingos tuo metu jo dalys;
- Tinkamai sukurta ir plačias galimybes suteikianti scenarijaus kalba suteiks motyvuotiems vartotojams galimybę patiems kurti naujas sistemos modifikacijas, papildymus bei dalintis tais scenarijais su aplinkiniais;
- Galima sukurti bet kokį teoriškai apibrėžtą scenarijaus nelygiagretumo tipą ar keletą jų palaikančią kalbą;

Scenarijaus kalbų trūkumai, kuriuos pavyko įžvelgti, siekiant tokias kalbas naudoti netiesiško scenarijaus realizacijai, yra:

- Scenarijaus kalbą turi interpretuoti žaidimo variklis, t.y vien scenarijaus kalbos kaip įrankio dinamiškai simuliacijai sukurti neužtenka. To variklio sudėtingumas didės kartu su didėjančiomis kalbos galimybėmis;
- Sudėtingus scenarijus palaikančią kalbą sunkiau ir sukurti - eikvojami laiko resursai;
- Klaidų radimo ir šalinimo (angl. *debugging*) įrankiai bendru atveju nebus tokie patikimi ir išsamūs kaip sisteminių kalbų aplinkose, pavyzdžiui Visual Studio, nes konkrečios sistemos kūrėjai negaiš laiko juos kurdami;
- Kadangi scenarijaus kodo failai kompiliuojami programos vykdymo metu, net kruopščiai optimizuotose kalbose to kodo veikimas bus lėtesnis nei iš anksto C++/C kalbomis sukompiliuotas kodas;
- Reikalingi tam tikri programavimo įgūdžiai;

Taigi, nors scenarijaus kalbos yra plačiai naudojama technologija kuriant kompiuterines priemones su scenarijumi, pati kalba dar nėra užtektina priemonė – reikalingas ją interpretuojantis variklis.

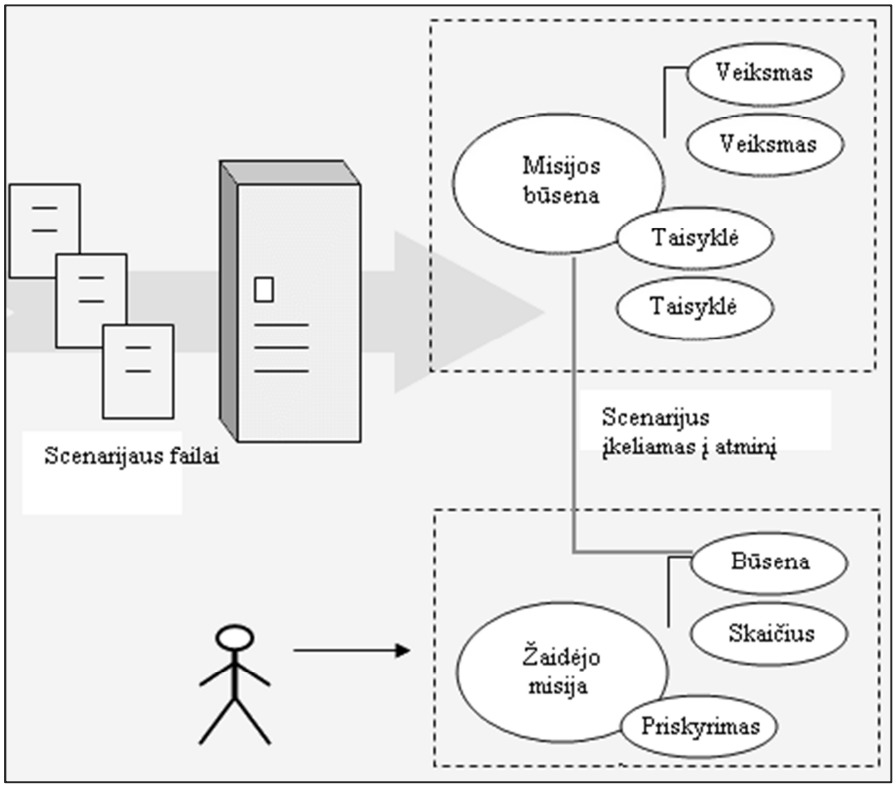
2.3. Užduočių variklis „EO+“

Ieškant konkretaus metodo ar produkto, suteikiančio galimybę sudėtingą scenarijų pateikti kompiuteriniame žaidime (socialinėje simuliacijoje) buvo aptiktas užduočių, dar vadinamų misijomis, variklio (ang. *quest engine*) „EO+“ [13] aprašymas. Šio variklio pagrindą sudaro paprasta scenarijaus programavimo kalba, paremta būsenomis, veiksmais ir taisyklėmis (3 pav.). Iš esmės užduotis - tai tekstinis failas, užkraunamas žaidimo serverio, kuris turi įdiegtą tokio kodo interpretatorių. Šis variklis skirtas daugelio žaidėjų internetiniam žaidimui, „The Endless Online“.

Serverio programos paleidimo metu visi scenarijaus failai, kurie patalpinti reikiamame kataloge, yra nuskaitomi ir paverčiami atitinkamais objektais atmintyje – serveris daugiau nebesikreipia į scenarijaus failus. Žaidėjui prisijungus visos užduotys ir tam skirto katalogo yra užkraunamos į atmintį ir su galinčiais atsitikti įvykiais yra susiejamos specialios funkcijos, iškviečiamos tiems įvykiams įvykus. Jeigu užduoties taisyklės (angl. *rules*) yra tenkinamos, variklis suaktyvins kitą būseną, įvykdys joje nurodytus veiksmus, panaikins senų sąlygų tikrinimą ir prijungs naujas sąlygas, atitinkančias esamą būseną. Įvykdžius užduotį žaidėjui

galima priskirti tam tikrą atlygį – daiktą. Užduočių progresas išsaugomas duomenų bazėje, kad žaidėjas galėtų tęsti ką pradėjęs ir po pakartotinio prisijungimo. Šis užduočių variklis vienu metu (lygiagrečiai) išsaugo ir žaidėjui leidžia vykdyti iki 20 užduočių. Apie sudėtingesnio laipsnio lygiagretumus ir dinamiką nėra užsimenama.

Nors šis variklis yra pritaikytas nedideliame kompiuteriniame žaidime („Endless Online“), tyrimo metu turėjusiame beveik 200 prisijungusių vartotojų, tačiau jo taikymas kitam projektui su plačiais reikalavimais nėra įmanomas dėl keleto priežasčių. Pirmiausia ir svarbiausia, nėra galimybės „EQ+“ modifikuoti – papildyti trūkstamu funkcionalumu. O šis variklis, kartu ir jo modelis, yra akivaizdžiai siauro funkcionalumo, kad galėtų būti naudojamas modernioje šiuolaikinėje simuliacinėje priemonėje. Be to, naudojant užduočių variklį būtų prisirišama ir prie konkrečios profesionalumo neužtikrinančios serverio programos, nėra galimybės pasirinkti įrankius žaidimo gamybai. Galiausiai, nors žaidimo puslapyje kūrėjai trumpai pristato šio variklio techninę informaciją, jis nėra atviro kodo ir juo naudotis gali tik „Endless Online“ vystyme dalyvaujantys asmenys, taigi nėra galimybės net išbandyti scenarijų rašymo šiam varikliui. Tad ši alternatyva neišsprendžia darbo problemos ir neturėtų būti naudojama mokomajai simuliacinei priemonei kurti, tačiau yra vienas iš egzistuojančių sprendimų, kurie naudojami užduotimis paremtai vartotojų ir kompiuterinių personažų komunikacijai kurti.



2 pav. Užduočių variklio „EO+“ koncepcija, pateikiama jo autorių internetinėje svetainėje [13]

2.4. „Adonthell“ variklis

„Adonthell“ – tai projektas, kurio tikslai yra sukurti nemokamą žaidimo variklį, įrankių rinkinį ir tokio paties pavadinimo žaidimą, palaikomą to variklio. Variklį palaiko dauguma operacinių sistemų, įskaitant Linux, Windows, MacOS, Solaris. Naujausios atviro kodo žaidimų vaidmenimis variklio versijos funkcionalumą kūrėjai kol kas pristato dar tik demonstraciniame žaidime „Waste‘s Edge“, pasižyminčiame šiomis savybėmis:

- Iš viršaus matomas spalvotas dvimatis žemėlapis;
- Kompiuteriniai personažai, kurie „gyvena“ užsiimdami tam tikromis veiklomis ir yra nestatiški;
- Dinamiška pokalbių sistema;
- Intriguojantis siužetas;
- Atmosferą sukuriantis muzikinis fonas.

Nors vizualinė šio žaidimo pusė (3 pav.) yra pasenusi – panašesnė į praėjusio šimtmečio pabaigos žaidimų grafiką -, tačiau dominanti savybė yra kompiuterinių personažų ir dialogų su jais dinamikos galimybė.

3 pav. „Adonthell“ variklį naudojančio demonstracinio žaidimo ekranvaizdis. Šaltinis: http://adonthell.linuxgames.com/screenshot/

Kaip pateikiama variklio architektūros apraše [14], Užduočių ir dialogų posistemė sudaro tik nedidelę viso variklio dalį, kuriame žaidimo duomenys ir kodas yra griežtai atskirti: variklis tik apibrėžia būsimų žaidimų funkcionalumą, bet ne pačius žaidimus. Variklio kūrimo tikslais laikytas lankstumas, lengvas plečiamumas, tad pats variklis yra panašesnis į karkasą, kuriame vėliau apibrėžiamos konkrečios taisyklės, objektai, parametrai ir pan, kuriant to variklio plėtinius (angl. *extensions*). Variklio pagrindas realizuotas C++ kalba, tačiau yra integruotas su Python scenarijaus kalba, kuria gali būti aprašomi įvairūs vaidmenų žaidimo elementai.

Vienas iš „Adonthell“ modulių ir yra pavadintas „vaidmenų žaidimo“ moduliu (angl. *RPG*). Jis atsakingas už esminius tokių žaidimų aspektus: personažus, užduotis, daiktus ir kita. Tačiau tai nėra išbaigta realizacija, o tik sąsaja, kuri apibrėžia aspektų formavimo taisykles, tačiau neriboja turinio laisvės – sąsajos turi būti papildytos Phython parašytu kodu, įvedančiu specifines žaidimo taisykles.

Svarbi vaidmenų modulio dalis – užduočių ir jų žurnalo sistema, kuri reikalinga siužeto pateikimui ir vartotojo progreso jame nustatymui. Bendru atveju tokios užduotys nėra tiesiškos ir gali būti išsprendžiamos skirtingais keliais, tad kiekvieną užduotį atitinka medis (grafas), kurio šaknis yra pati užduotis, o lapai vaizduoja atskirus jos etapus. Kiekvienai iš medžio viršūnių priskiriama taisyklė, nusakanti, kaip nustatyti iš jos išeinančių viršūnių įvykdymo būseną. Scenarijai (Python kodo fragmentai), kuriems reikia informacijos tik apie visos užduoties būseną, kreipsis į užduoties medžio šaknį, o detalesnė informacija gali būti gauta tikrinant atskirų medžio lapų būsenas. Variklyje skiriamos trys mazgų (viršūnės ir lapų) būsenos:

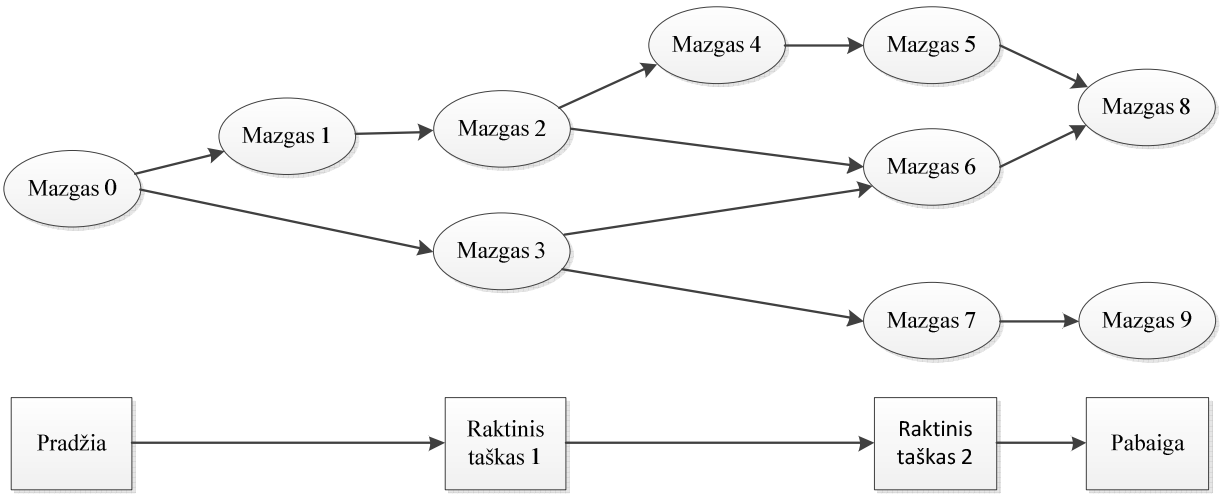
- Pradėta. Užduotis yra pradėta, kai šakninė medžio būsena nustatoma į įvykdytą būseną;
- Vykdoma;
- Įvykdyta.

Sekant tokias būsenas apibrėžiamas žaidimo progresas. Vartotojui šį progresą sekti padeda užduočių žurnalas, kuris yra glaudžiai susijęs su užduočių medžiais – kiekvienam mazgui galima priskirti tekstinį žurnalo įrašą, o kai tas mazgas pasieks įvykdymo būseną, įrašas bus įkopijuotas į vartotojo užduočių žurnalą.

Norintiems naudoti šį variklį savo žaidimo vystymui, kūrėjai pateikia detalią programavimo sąsają (angl. *API*), aprašančią variklio klases ir metodus [15], viešai pateikia pačią variklio architektūrą. Scenarijaus kūrimo procesas šiam varikliui taip pat detaliai ir su pavyzdžiais aprašytas, netgi pateikiama bendrų patarimų nelinijinio scenarijaus kūrimui.

Teigiama, jog scenarijus turėtų adaptuotis prie vartotojo žaidimo stiliaus, tačiau pagrindinis

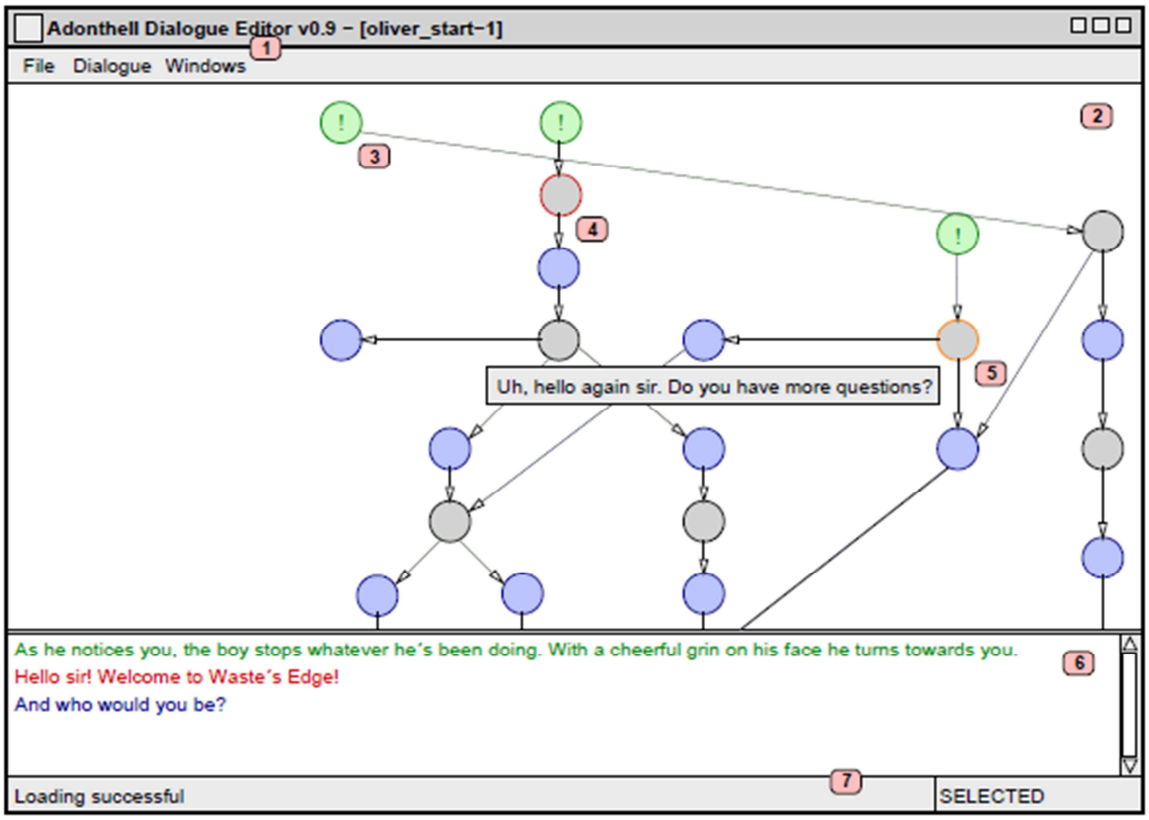
sunkumas yra rasti tinkamą balansą tarp linijinių ir adaptyvių scenarijaus dalių [16]. Variklyje dalis iš žaidimo istorijos yra esminiai (raktiniai) elementai, t.y tuos įvykius ir situacijas kiekvienas žaidėjas turi patirti tam tikra tvarka, kad scenarijus būtų logiškas, o kitos dalys paprasčiausiai jungia tas esmines dalis (4 pav.). Raktiniai taškai šakotame scenarijuje atitinka lygiagrečių siužeto mazgų rinkinį, kurie tarpusavyje yra ekvivalenčios alternatyvos. Dažnai tokie mazgai susiję su tomis pačiomis vietovėmis, aplinkybėmis, įvykiais, tačiau atskiria galimus žaidėjo pasirinkimus ir gautus rezultatus. Tie rezultatai įtakoja toliau pasiekiamus kelius, kurie gali būti prieinami iš keleto mazgų arba galimi tik pasirinkus vieną specifinį kelią.



4 pav. Bendra scenarijaus struktūra – netiesiškumo ir linijiškumo santykis [16]

Šalia teorinių patarimų, kaip formuoti netiesišką scenarijų, siūlomas ir grafinis įrankis - pokalbių su kompiuteriniais personažais redaktorius - lengvam užduočių modeliavimui „Adonthell Dialogue Editor“ (5 pav.) Pagrindinis šio įrankio tikslas – padėti paruošti sudėtingos struktūros dialogus, leidžiant visą dėmesį sutelkti į paties scenarijaus kūrimą. Grafinėmis priemonėmis sukurtos schemos vėliau automatiškai sugeneruojamos į Python kalbos skriptus, tad rašytojams nereiktų rašyti kodo.

„Adonthell“ variklio detalumas ir lankstumas yra neabejotinos šio projekto stiprybės, tačiau platūs užmojai nėra galutinai išbaigti, kalbant tiek apie dokumentaciją, tiek apie patį variklį – prieš septynerius metus pasirodęs projektas nėra baigtas ir jo vystymo darbų sparta (jeigu jie išvis vykdomi) nėra aiški, kadangi kūrėjai nepateikia konkrečių datų. Šiuo metu naujausia variklio versija yra „v.0.4“ ir kartu su dokumentacija gali būti parsisiųsta nemokamai iš projekto internetinės svetainės.



5 pav. Pagrindinis „Adonthell“ dialogų redaktoriaus langas, kuriame matomas modeliuojamas dialogas [17]

2.5. Dirbtinio intelekto vystymo modelis „ScriptEase“

Kaip minėta 2.2 poskyryje, procedūrinės scenarijų kalbos yra įprastas būdas komunikacijai tarp realaus žaidėjo ir virtualių objektų realizuoti. Nelinijiniai scenarijai dažnai yra tokie sudėtingi, kad juos turi rašyti kompiuterių programuotojai, o ne žaidimo istorijos autorius. [18] Šis scenarijaus vystymo trukdis padidina žaidimo kūrimo kainą ir įveda atskirtį tarp rašytojo idėjų ir programavimo rezultatų. Šias problemas bando išspręsti „ScriptEase“ sistema, suteikianti galimybę sparčiai konstruoti sudėtingus kompiuterinių personažų elgesio modelius be tiesioginio programavimo.

„ScriptEase“ paremtas programavimo šablonų (angl. *design patterns*) idėja, kurie yra bendrai efektyvi programavimo priemonė, tik šiuo atveju naudojami generatyviniai šablonai [19] – šablonai, kurie gali būti automatiškai paversti į vykdomąjį programos kodą. Palaikomi keturių tipų šablonai: netikėto susidūrimo su virtualiu objektu, KP elgesio, pokalbio ir užduoties. Žaidimo turinys tuomet nusakomas tiesiog vykdant tris žingsnius:

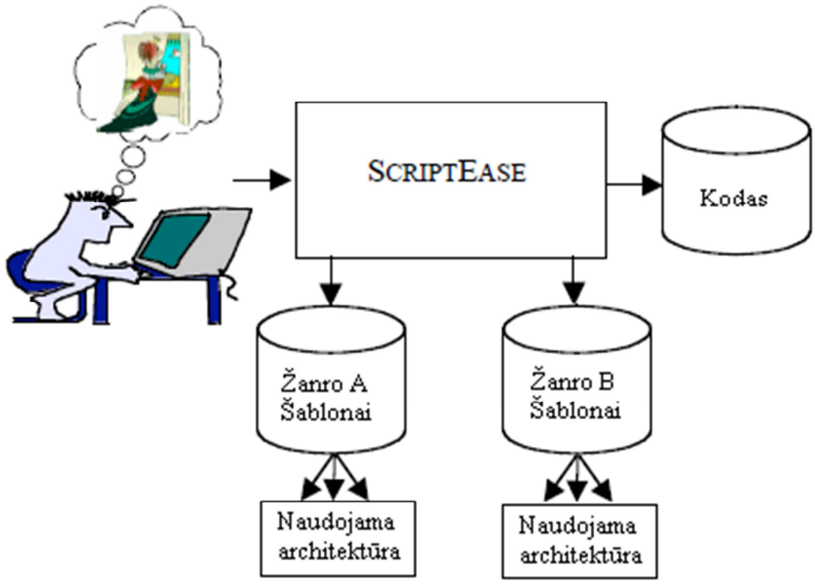
- Pasirenkant konkretų šabloną ir sukuriant egzempliorių (angl. *instance*). „SriptEase“ suteikia pasirinkimą iš plataus žaidimams vaidmenimis skirtų šablonų, kurie, kaip minėta,

gali būti panaudoti sudarant dialogus, personažų sąveiką ar kompiuterinių personažų elgseną;

- Pritaikant šablonui sugalvotą turinį atitinkantį tekstinį aprašymą;
- Sugeneruojant scenarijaus kodą vieno mygtuko paspaudimu.

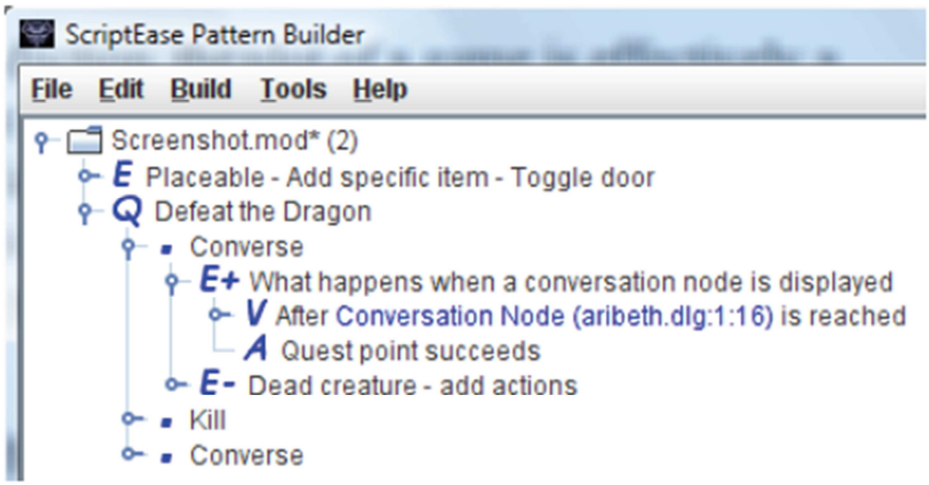
Be numatytų šablonų „ScriptEase“ teikia galimybę autoriams sukurti naujus šablonus savo reikmėms, tad tarnauja ir kaip tokių šablonų katalogas.

Modulinė „ScriptEase“ sistemos architektūra pavaizduota 6 pav. Nors šablonų kūrimas buvo nukreiptas vaidmenų žaidimų žanro linkme (šiais įrankiais kuriamas turinys žaidimui „The Neverwinter Nights“), yra palaikomi ir kitokių kompiuterinių žaidimų žanrų scenarijaus šablonai, o atliktus tam tikras programines modifikacijas šablonus būtų galima taikyti ir kitoms architektūroms, t.y kitų žaidimų varikliams.



6 pav. „ScriptEase“ architektūra [18]

Žaidimo misijų, įskaitant ir nelinijines, atskiri elementai vedami ir matomi kaip hierarchinė teksto elementų struktūra. „ScriptEase Pattern Builder“ įrankio (7 pav.) ekranvaizdyje matomas šablonų panaudojimas ir kita misijų sudarymo specifika. Čia *Q* žymi misiją, o punktais (▪) žymimos tos misijos dalys. Kiekviena misijos dalis įgyvendina bent vieną sąveikos su virtualiu objektu šabloną, nurodantį, kada ta dalis įvykdoma (*E+*); taip pat ta dalis gali turėti tokių šablonų, nurodančių, kada įvykdyti dalies nepavyksta (E-).



7 pav. Misijos formavimas „ScriptEase“ įrankiu [11]

Įrankis numato misijų tarpusavio susiejimą, vartotojo alternatyvas, t.y lygiagretumą, tiek misijų pasirinkimo lygyje, tiek jų atskirų dalių lygyje. Tačiau, sudarytas šakotos misijos medis nėra matomas tiesiogiai, nes įrankis remiasi nuosekliu teksto eilučių išdėstymo principu.

Iš apibrėžtos hierarchinės struktūros galima sugeneruoti NWScript scenarijaus kalbos kodą, kuris skirtas minėto „Neverwinter Nights“ žaidimo scenarijui apibrėžti. Nors šio eksperimentinio projekto apraše teigiama, kad galės būti palaikomos ir kitos architektūros, tačiau galimybių tą išbandyti ir pritaikyti įrankius kitoms reikmėms nėra – parsisiųsti galima tik su šiuo žaidimu susietą įrankį, kuris netgi neveikia be paties žaidimo įdiegimo.

2.6. Mokomoji simuliacinė priemonė „Mano Kelias“

Lietuvos rinkai skirtų ir Lietuvos kompanijų sukurtų kompiuterinių žaidimų kiekis ir įvairovė akivaizdžiai nusileidžia didžiųjų pasaulio valstybių rezultatams. Tikėtis, kad šioje šalyje būtų sukurta „Mass Effect“ ar „Dragon Age“ serijų lygio žaidimai vaidmenimis (juos sukūrė „Bioware“ kompanija), yra per drąsu – sudėtingiems ir milžiniškiems projektams reikalingas ir atitinkamas resursų kiekis. Visgi, net ir Lietuvos rinkoje galima aptikti projektų, kuriuose scenarijus turėtų dinamikos požymių, davusių pradžią modernesnių sprendimų paieškai, tad ir šio darbo problemos aktualumui.

Vienas iš tokių - simuliacinis mokomasis žaidimas „Mano Kelias“ (8 pav.), kuris yra Kauno teritorinės darbo biržos inicijuoto projekto dalis. Žaidimas skirtas nuteistiems asmenims, kurie netrukus išeis į laisvę. Pagrindinis žaidimo tikslas – supažindinti žaidėją su realiu gyvenimu: suteikiama galimybė įvairiose tariamai tikrose gyvenimo situacijose išbandyti įvairius sprendimus. Svarbu tai, kad nuteistasis gali išbandyti naują elgesį ir apsisprendimo pasekmes

saugioje aplinkoje. Jis patenka į įvairias gyvenimiškas situacijas ir perima įvairaus pobūdžio informaciją, reikalingą kasdieniniame gyvenime ar profesinėje veikloje. Žaidėjas-veikėjas skatinamas įsidarbinti, siekti karjeros, planuoti gyvenimą, užmegzti ir gerinti santykius su aplinkiniais bei artimaisiais. Taip mokoma galimų elgesio modelių. Žaidėjas skatinamas pakartotinai nepatekti į įkalinimo įstaigą, bet uždirbti pinigų, susikurti kuo geresnes gyvenimo sąlygas: susirasti ar įsigyti gyvenamąjį plotą, pagerinti materialinę padėtį, palaikyti teigiamus santykius su šeima, užmegzti naujus socialinius santykius ir kt., o kaip jam tai pavyks, priklauso nuo pasirinktų atsakymų dialoguose.

8 pav. Simuliacinio žaidimo „Mano kelias“ scena

Šis žaidimas sukurtas naudojant Macromedia Flash 8 technologiją, o jo scenarijus parengtas tikslinių grupių auditorijos ir dalykinės srities ekspertų. 2007 metais visuose projektą įgyvendinančiose įstaigose tikslinės grupės dalyvių (nuteistųjų), pradėjusių žaisti simuliacinį žaidimą „Mano kelias” buvo apie 570 [20]. Vėliau įvykdyta projekto analizė ir nuteistųjų apklausa sąlygojo projekto plėtrą, t.y. modernesnės, didesnės apimties ir dinamikos žaidimo dalies idėją. To priežastys – nepakankamai aukšta simuliacijos kokybė techniniu požiūriu ir nepakankamai išsamus turinys. Analizės rezultatai mini ir tokius „Mano kelias“ trūkumus:

- Žaidimo techninis sprendimas nenumato situacijų, dialogų plėtros galimybių. Tad norint turėti galimybę papildyti žaidimą naujais elementais, reikia perdaryti patį žaidimo variklį iš naujo.

- Nėra galimybės žaidėjui suformuoti savo realistišką aplinką, įkelti objektus. Tobulintinos funkcijos suteikiančios daugiau statistinės medžiagos apie žaidėjo atliktų užduočių efektyvumą, laiką, klaidų kiekį, mokymosi progresą.
- Žaidimas neturi nei tinklinės nei internetinės versijos, nėra grupinio žaidimo ar žaidimo su realiu oponentu galimybės.
- Nėra užduočių ir aplinkų susijusių su asmenybės vystymu ir religiniais poreikiais. Nėra pilietiškumo ugdymui, bendruomeniniam gyvenimui, politiniam išprusimui skirtų įgūdžių lavinimo užduočių.

Taigi, nors inovatyvios idėjos buvo paverstos žaidimu, tačiau tyrimas parodė, kad jam trūksta detalumo, dinamikos, ir turinio apimties, kurios padidinti galimybės nėra. Trūkumams pašalinti reikalingas modernesnis, sudėtingesnis ir daug lankstesnis techninis sprendimas, galintis didesnės apimties scenarijus pateikti tikslinės grupės asmenims patrauklia forma. Dėl šios priežasties pateiktas pavyzdys yra daugiau problemos aktualumo pagrindimas, nei jos sprendimas. Nors „Mano kelias“ sprendimo tematika ir tikslas yra panašūs į magistrinio darbo problemą, tačiau sprendimo kokybė – nepakankama, struktūra – neaiški ir nedokumentuota. Būtent šio projekto vykdymas įtakojo vieno iš darbo eksperimentu pasirinkto projekto atsiradimą, kartu ir dinamiško užduočių variklio poreikį.

2.7. Analizės apibendrinimas

Susipažinus su nelinijinio scenarijaus kūrimo problemomis paaiškėjo, jog tokio scenarijaus realizavimas kompiuteriniuose žaidimuose nėra triviali užduotis – ji sudėtinga tiek iš techninės, tiek iš turinio sudarymo pusės. Daugybė įvairių sprendimų naudota dinamiškų užduočių ir dialogų palaikymui. Šiame darbe analizuotų „EO+“, „Adonthell“ ir „ScriptEase“ tarpusavio palyginimas, pateiktas 1 lentelėje. Į palyginimo kriterijus įeina nelinijinių užduočių palaikymas, galimybė plėsti scenarijų be kompiliavimo, viešai pateikiamos architektūros (techniniu požiūriu) išsamumas, galimybė pritaikyti su įvairiomis technologijomis bei pastebėti trūkumai, galintys užkirsti kelią variklio ar įrankių naudojimui.

Žinoma, tai nėra vieninteliai egzistuojantys sprendimai – analizės metu buvo aptikta ir nagrinėta daugiau produktų, kurie apibendrinant buvo arba iš esmės panašūs į pateiktus, arba jų aprašymui trūko oficialios informacijos. Pavyzdžiui, Bioware‘s „Infinity“ žaidimo variklio, kuris taikytas tokiuose izometrinių žaidimų vaidmenimis granduose kaip „Baldur‘s Gate“ ir „Icewind Dale“, detalės nėra viešai atskleidžiamos, tačiau jo geriausias savybes bando nukopijuoti atviro

kodo projektas „GemRB“ [21], kurio techniniai pagrindai nedaug skiriasi nuo „Adonthell“ projekto. Vyraujančių produktų, kurie būtų naudojami problemos sprendimui daugelyje projektų, nepastebėta – dažnu atveju vienam dinamiškam žaidimui sukuriamas ir atskiras variklis.

1 lentelė. Egzistuojančių dinamiško scenarijaus sprendimų palyginimas

	EO+	**Adonthell**	**ScriptEase**
Tipas	Variklis	Variklis	Strategija ir įrankiai
Palaikomos nelinijinės užduotys	Neapibrėžta	Palaiko šakotą scenarijų ir jo santykį su vartotojo profiliu	Palaiko šakotą scenarijų ir jo santykį su vartotojo profiliu
Paremtas scenarijaus kalba	Taip, specifinė kalba	Taip, Python kalba	Taip, strategija galėtų būti taikoma įvairioms kalboms.
Galima plėsti scenarijų be kompiliavimo	Taip	Taip	Taip, ypač patogiai, nes toks yra sistemos tikslas
Sistemos architektūros aprašymas	Pateikta neišsami schema	Detali dokumentacija	Paaiškinta koncepcija
Pritaikymas įtakotų technologijų pasirinkimą	Taip	Taip	Teoriškai – ne
Atviro kodo	Ne	Taip	Ne
Esminiai trūkumai problemos sprendimui	Siauras funkcionalumas; nėra variklio vystymo galimybės	Neišbaigtas projektas; pasenusi grafinė sąsaja, kaip nekintanti variklio dalis	Specializuoti įrankiai, kurių pritaikymas kitokioms architektūroms tik teorinis – darbiniai failai nepateikiami

Apskirtai, dauguma variklių orientuoti į visų žaidimo modulių realizaciją, ir užduočių modulis užima tik nedidelę sistemos dalį. Šiuolaikiniai komerciniai žaidimai ir jų varikliai, rinkos poreikiams patenkinti vis labiau koncentruojasi į įspūdingą grafinę sąsają, tad santykinai mažėja dėmesys, skiriamas tekstinio scenarijaus sudėtingumui, kuris didžiausią pasisekimą turėjo senesnės kartos žaidimuose. Tiesa, mokomiesiems žaidimas tekstinio turinio svarba išlieka, o nelinijiškumas kiekviename kompiuteriniame žaidime yra teigiamai vertinama savybė, nepriklausomai nuo jo žanro. Neliniškumui modernizuoti teoriniame lygmenyje šnekama ir apie automatinį turinio generavimą [2][22], kurio pritaikymas mokomosioms priemonėms yra mažiau tikėtinas, dėl kiekvienos turinio detalės svarbos ir prasmės jose.

Dalis scenarijaus nelijiniškumo problemas sprendžiančių straipsnių koncentruojasi į to scenarijaus sudarymo iš autoriaus pusės problemas, tačiau į sistemų, kurios palaiko tą kuriamą scenarijų detales nesigilinama. Iš techninės pusės, pastebėta, kad scenarijaus kalbos iki šiol yra faktiškai populiariausias sprendimas dinamiško scenarijaus formavimui, nors ir turi savų trūkumų – visos trys šioje dalyje aprašytos sistemos taip pat paremtos scenarijaus kalbų idėja.

Tačiau yra ir išimčių – nors apie Riot žaidimų variklio [1, 389 p.], kurį naudoja „The Lord of the Rings: The Fellowship of the Ring", bei keletas kitų žaidimų, architektūrą detalios informacijos rasti nepavyko, tačiau yra žinoma, kad trimačių aplinkų lygių kūrimo įrankis scenarijaus kūrėjams leidžia tiesiogiai keisti objektų elgesio nustatymus, kai patys elgesio modeliai yra suprogramuoti C++ kalba – kartu išlaikomi scenarijaus kalbų teikiami balansavimo privalumai ir C++ kalbos kompiliatorių efektyvumas.

Ši idėja – galimybė apibrėžti scenarijų be specializuotos scenarijaus programavimo kalbos, išlaikant tų kalbų teikiamus privalumus – yra viena iš priežasčių, kodėl nuspręsta sudaryti naują variklį ir jo modelį, Lietuvoje vykdomų socialinių projektų priemonėms sukurti. Tokiu būdu nereiktų eikvoti resursų nei kalbos kūrimui, nei jos sintaksės mokymuisi kuriant scenarijų. Tačiau tai nebuvo vienintelė prielaida naujo modelio sudarymui. Pagrindinės priežastys, dėl kurių problemos sprendimui nebuvo pasirinkta nei viena iš minėtų egzistuojančių technologijų, yra šios:

- „EQ+" variklis neskirtas kitų sistemų scenarijui palaikyti – tai konkretaus žaidimo variklis, be perpanaudojimo ir vystymo galimybės, be to – riboto funkcionalumo.
- „Adonthell" variklio projekto įgyvendinimas dar nėra baigtas, jis vyksta lėtai. Su užduočių varikliu surištos grafinės priemonės jau pasenusios, tačiau nėra galimybės paprastai jų pakeisti modernesnėmis technologijomis.
- „ScriptEase" nėra naudojimui paruoštas variklis. Tai strategija ir įrankiai, tačiau viešai prieinama tik konkrečiam žaidimui/architektūrai skirta jų versija.
- Kitų tinkamų egzistuojančių sprendimų, kuriuos būtų patogu ir paprasta naudoti vykdomuose socialiniuose projektuose, aptikti nepavyko.
- Pritaikant svetimą technologiją atsiranda rizika, kad tam tikro reikiamo funkcionalumo išgauti nepavyks, arba bus aptikta kitų nesklandumų, kurių sutvarkymas priklauso tik nuo variklio kūrėjų. Kuriant savo variklį naujų funkcijų sukūrimui trukdžių nėra.
- Kadangi variklis bus pirmiausia taikomas mokomojo pobūdžio simuliacijoms, reikalinga lankstesnė profiliavimo ir įvertinimo sistema, nei tą siūlo pramoginių žaidimų varikliai;

- Pastebėta, kad scenarijaus kalbos naudojimas dinamiškų užduočių varikliui sukurti bei jo scenarijui suformuoti nėra būtinas, tad naudoti tokias kalbas, kaip papildomą technologiją vien scenarijaus formavimo reikmėms, nėra tikslinga, jei scenarijaus kalbų teikiami privalumai liks išlaikyti.
- Kadangi vienu metu vykdomas daugiau nei vienas panašus projektas, o perspektyvoje jų gali ir padaugėti, resursų, reikalingų naujos sistemos sukūrimui kiekis, sąlyginai mažėja, nes tas pats naujas variklis galės būti panaudotas pakartotinai. Tai padidins galimybę, kad patogios specifinėms reikmėms pritaikytos sistemos kūrimas atsipirks.

Taigi, pasirinkta kurti naują variklį, kuris realizuotas remiantis tolesniame skyriuje aprašomu modeliu.

3. DINAMIŠKŲ UŽDUOČIŲ VARIKLIO MODELIO APRAŠYMAS

Nei vienas iš surastų egzistuojančių modelių ar galutinių produktų nebuvo pasirinktas problemos – socialinės kompiuterinės simuliacijos kūrimo – sprendimui, tad sudarytas ir pateikiamas specializuotas ir ankščiau nepublikuotas sprendimas. Šio naujo sprendimo atsiradimą labiausiai įtakojo atsiradusių socialinių projektų specifiniai dinamikos reikalavimai, kuriuos įgyvendinti naudojant egzistuojančius modelius pasirodė neparanku.

Šioje dalyje yra aprašytas modelis, sudarytas remiantis atliktu susijusios literatūros ir panašių produktų tyrimu, egzistuojančiuose sprendimuose pastebėtomis bazinėmis idėjomis, konkrečių projektų Lietuvoje reikalavimų specifikacijomis, srities ekspertų sudarytais scenarijaus pavyzdžiais, taip pat ir darbo autoriaus asmenine patirtimi. Dinamiško scenarijaus variklio modelis apibūdina interaktyvioms kompiuterizuotoms priemonėms skirto tekstinio scenarijaus realizacijos principus.

Pagrindinis variklio funkcionalumo reikalavimas – jis turi palaikyti nelinijinį adaptyvų scenarijų, formuojamą atskiromis užduotimis ar pokalbiais. Adaptyvumo rezultatai priklauso nuo konkrečių vartotojo pasirinkimų, kurių kiekvienas gali būti vertinamas skirtingai. Tokio scenarijaus plėtra neturi būti ribojama.

Esminis skirtumas, tarp kitų panašių sprendimų, ir siūlomojo, yra tai, kad scenarijus čia ne tik vaizduojamas, bet ir apibrėžiamas duomenų struktūrų pavidalu, t.y. jokio lygio programavimas užduočių formavimui nėra reikalingas, kadangi variklis interpretuoja objektus bei jų ryšius, o ne kodą.

Realizuojant tokį variklį, rekomenduojama remtis pateikta struktūra, į kurią įeina: virtualaus pasaulio objektų tipai ir sandara, tekstinio scenarijaus skaidymas, scenarijaus elementų jungimo taisyklės, scenarijaus vienetų pasiekimo sąlygos ir atlygis už įvykdytas užduotis.

3.1 Virtualaus pasaulio objektai

Kuriant virtualią socialinę simuliaciją siekiama atkartoti visuomenės funkcines sąsajas, tad svarbiausi virtualaus pasaulio objektai šiuo atveju yra tokios virtualios visuomenės dalyviai. Esminis skirtumas tarp simuliacijoje egzistuojančių bei komunikuojančių individų yra:

a) tai realaus žmogaus valdomas personažas, kurio kompiuterinis įvaizdis (įsikūnijimas) ir elgsena yra to žmogaus sprendimų projekcija;
b) tai kompiuterinis personažas, kurio elgsena ir sprendimai yra užprogramuoti algoritmais;

Šis modelis neapibrėžia, ar tame pačiame kompiuterizuotame pasaulyje egzistuos tik vienas žmogaus valdomas personažas, ar jų bus didesnis kiekis – kitaip tariant, tolesni principai gali būti sėkmingai taikomi tiek vieno žaidėjo (anlg. *single player*) mokomuosiuose žaidimuose, tiek virtualiuose pasauliuose su daugeliu vartotojų. Antruoju atveju tereikia įvertinti papildomas funkcijas, kurias gali turėti dviejų ar daugiau žmonių sąveika, lyginant su vartotojo ir kompiuterinių personažų komunikacija.

Laikomasi principo, kad abiejų tipų personažai vizualiai yra tame pačiame hierarchiniame lygyje - stengiamasi kompiuterinius personažus savo savybėmis priartinti prie realių vartotojų, kad bendravimas tarp jų būtų kuo labiau įtikinamas.

3.1.1 Realaus asmens įsikūnijimas

Virtualaus pasaulio centre – simuliacijos tikslinės grupės asmuo. Jo persikėlimas į kompiuterizuotą erdvę siejamas su tą asmenį ar jo pasirinktą charakterį reprezentuojančiu grafiniu elementu – įsikūnijimu (angl. *avatar*). Tai gali būti dvimatis ar trimatis, animuotas arba statiškas sprendimas. Logiška manyti, kad vizualinis pasaulio, tad ir įsikūnijimo, realumas bei kokybė tam tikru laipsniu lems ir simuliacijos sėkmę, tačiau tai nėra susiję su sprendžiama problema, tad yra nagrinėjamos vartotojo įsikūnijimo savybės, susijusios su veiksmo-pasekmės principo (dinamikos) įgyvendinimu, o ne vaizdo parametrais, nors jie taip pat būtini. Vartotojo įsikūnijimą užduočių variklyje apibrėžia:

- Identifikatorius, pagal kurį užduočių variklyje yra atpažįstamas konkretus vartotojo

įsikūnijimas;

- Užduočių, kurias tas vartotojas vykdo, įvykdė bei gali įvykdyti, aibės;
- Vartotojo profilį nusakantys parametrai, kurie kinta vykdant veiksmus, priklausomai nuo užduočių vykdymo rezultatų. Parametrų pavadinimai ir konkretus kiekis nustatomas atsižvelgiant į kuriamos simuliacijos tikslus ir sukurtą scenarijų. Maksimalus parametrų kiekis turi likti neapribotas, kad tam tikrame scenarijaus kūrimo taške prireikus specifinės paskirties parametro, jį būtų galima įtraukti be programavimo. Galimi vartotojo įsikūnijimo parametrų pavyzdžiai – įvykdytų užduočių skaičius, simuliacijoje praleistas laikas, gyvenamoji vieta (virtuali), savarankiškumas, draugiškumas ir pan. Kokiu principu tokie parametrai kinta, aprašyta 3.5 poskyryje.
- Virtualūs tam vartotojui priklausantys daiktai, susiję su užduočių vykdymu (pavyzdžiui, įsidarbinimui reikalingi dokumentai) arba su virtualios realybės socialiniu gyvenimu (virtualūs pinigai, butas, maistas, rūbai ir t.t.).

Toks objektas virtualiame pasaulyje egzistuoja tik tuo metu, kai vartotojas naudojasi kompiuterizuota sistema. Šiam atsijungus, visos savybės turi likti išsaugomos ir panaudotos kuriant vartotojo įvaizdį kitos sesijos metu.

3.1.2 Kompiuterinis personažas

Kompiuterinis personažas (KP) yra dirbtinis, jo elgsena nustatyta programinėmis priemonėmis. Tokie personažai sudaro supaprastintą virtualią visuomenę, kurioje tikri asmenys gali saugiai išbandyti įvairius elgsenos modelius. Su kiekvienu kompiuteriniu personažu vartotojas ar vartotojai gali komunikuoti, nes virtualiame pasaulyje tokie personažai pateikiami kaip vizualūs interaktyvūs objektai. Savybės, priskiriamos kiekvienam KP, yra:

- Identifikatorius, pagal kurį užduočių variklyje yra atpažįstamas konkretus KP.
- Vardas ar pasirinktinai kiti vizualūs parametrai, kurie padeda galutiniam vartotojui atskirti vieną KP nuo kito.
- Scenarijaus tekstinių elementų (žr. 3.2. skyrių) aibė, priskirta šiam objektui. Ši simuliacijos metu nekintanti aibė apibrėžia visus įmanomus dialogus su šiuo personažu ir KP vaidmenį vykdant užduotis.
- Padėtis. Nesvarbu, ar kompiuterinis personažas juda, ar stovi vietoje, jis priskiriamas tam tikrai vietovei, kurias patogu vadinti kambariais. Žinoma, veiksmas gali vykti tiesiog vienoje bendroje e-terpėje, t.y. kambaryje, tačiau bandant atkartoti realią visuomenę

paprastai bus reikalingas didesnis kiekis įvairių vietų – pavyzdžiui, nakvynės namai, bankas, universitetas, tėvų namai ir t.t. Svarbu, kad techniškai ne kompiuteriniam personažui yra nurodoma aplinka, o atvirkščiai, konkrečiai aplinkai yra priskiriami reikiami KP.

- Kompiuterinio personažo migracijai tarp kambarių išspręsti įvedami du parametrai – atsiradimo misijos ID ir dingimo misijos ID. Jei KP aplinkai yra pastovus, šie parametrai nenurodomi. Jeigu vartotojas dar nevykdo atsiradimo misijos – KP aplinkoje dar nevaizduojamas, o jeigu vartotojas jau įvykdęs dingimo misiją – KP jau nebevaizduojamas. Personažo persikėlimas iš vieno kambario į kitą vykdomas jį priskiriant dviem ar daugiau aplinkų, o atsiradimo/dingimo parametrai nurodomi taip, kad viename scenarijaus taške būtų matoma tik viena iš jo kopijų.

Kompiuteriniu personažu laikomi ir interaktyvūs aplinkose esantys objektai, nepanašūs į žmones. Nors tie objektai nėra personifikuojami, jų paskirtis ir principas yra toks pats – tam tikroje vietoje esantis objektas turi vartotojui prieinamų užduočių, dialogų sąrašą. Jų dialogai susiję su galimais atlikti veiksmais, pačių objektų aprašu, o ne dviejų asmenų pokalbiu. Nežmogiško tipo kompiuterinių personažų pavyzdžiai – skelbimų lenta, kurioje rašoma aktuali informacija, virtualus kompiuteris, kuriuo vartotojo įsikūnijimas gali naudotis, gatvėje stovintis automobilis ir pan.

3.1.3 Įsigyjami objektai

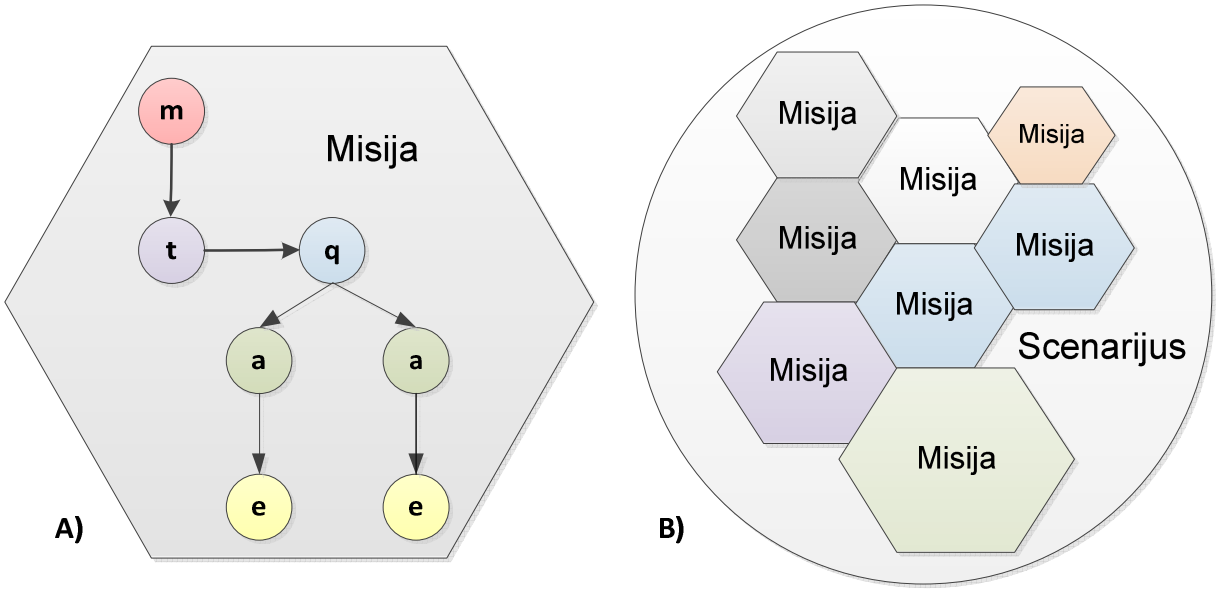
Trečia su užduočių vykdymu susijusi virtualios aplinkos objektų rūšis – realaus pasaulio daiktus vaizduojantys virtualūs daiktai, kuriuos vartotojai gali tam tikrame scenarijaus taške gauti (bei prarasti) arba už virtualius pinigus nusipirkti. Tokie daiktai naudojami ir kaip reikalavimai užduoties vykdymui, ir kaip užduoties įvykdymo atlygis. Pavyzdžiui, virtualus universiteto baigimo diplomas, vartotojui įteikiamas sėkmingai įvykdžius užduotis, susijusias su studijomis universitete, o tokio objekto turėjimas yra privaloma sąlyga kvalifikuoto darbo (virtualaus) radimui.

Žinoma, ne visa tokių objektų imtis privalo būti susijusi su užduotimis. Virtualų simuliacijos pasaulį galima praturtinti prekybos ir laisvalaikio aplinkose parduodamais ar randamais daiktais, kurie, pavyzdžiui, keičia žaidėjo įsikūnijimo išvaizdą (kirpykloje – nauja šukuosena, parduotuvėje – kitokie rūbai, baldai virtualiems namams ir t.t.).

3.2 Scenarijaus skaidymas

Nagrinėjamas scenarijus pasižymi tuo, kad pagrindinė informacija pateikiama tekstu – ne abstrakčiais vaizdais, garsais ar formulėmis, o rašytinėmis kalbos formomis. Užduotys (angl. *quests*) formuluojamos tekstu, o užduočių vykdymas veda prie tolesnių užduočių. Tokio tipo scenarijaus panaudojimas kompiuterinėse mokomosiose ar pramoginėse priemonėse (žaidimuose) yra nebrangus, lengvai valdomas ir redaguojamas, greitai įsisavinamas (lyginant, pavyzdžiui, su įgarsintu dialogu) ir vis dar populiarus. Akivaizdu, kad ne visa tekstinė informacija yra lygiavertė. Pavyzdžiui, žaidime pateiktos frazės „susirask darbą“ ir „darbo susirasti nepavyko“ savo sritimi yra susijusios, tačiau neša savyje kitokią paskirtį. Dėl to įvedamas empiriniu būdu nustatytas tekstinio scenarijaus skaidymas į 5 tipų scenarijaus vienetus – misijos pradžią (*M* tipas), užduotį (*T* tipas), klausimą (*Q* tipas), atsakymą (*A* tipas) ir pabaigą (*E* tipas).

Šiuo atveju priklausomybės simbolis (ϵ) rodo priklausomybę konkrečiam tipui, bet ne aibei. Taikant tokį skaidymą galima pilnai padengti bet kurios žaidybinės situacijos scenarijų, kur žaidėjui pateikiama tekstinė informacija. Gali prireikti rašytojų pateikto scenarijaus teksto modifikacijų, tačiau to nereiks jeigu scenarijus kuriamas laikantis siūlomų principų.



9 pav. Primityvi misija ir iš misijų sudarytas scenarijus

Tekstinė informacija yra visus tipus siejantis parametras, tačiau apibrėžti elementai nėra vien tekstas – scenarijaus elementams priskiriama ir savybių aibė *P*, egzistuoja ryšiai su kitais scenarijaus elementais, tad šiuos scenarijaus vienetus yra teisingiau vadinti objektais. Tokių objektų rinkinys (sudarytas laikantis tvarkos) vadinamas *misija* (9 pav., a). O visas mokomosios

priemonės scenarijus ir yra sudarytas iš skirtingų misijų aibės (9 pav., b).

Jungaus orientuoto ir galimai ciklinio grafo [24] struktūra, kur grafo viršūnes atitinka minėti teksto objektai, o lankai atitinka tų objektų vaizdavimo alternatyvas ir eiliškumą, sudaro kiekvienos misijos dinamiškos struktūros pagrindą. Schemoje pavaizduoti orientuoti grafo lankai žymi tai, kad iš konkrečios viršūnės vartotojui judėti galima tik tolyn (gilyn), tačiau ne atgal, išskyrus tuos atvejus, kada lankas nukreipia atgal, sudarydamas kilpą. Pseudografas nebus gaunamas, kadangi viršūnė niekuomet nenukreipia į save pačią.

Ryšiai nėra trivialūs dėl to, kad siekiama įgyvendinti dinamišką scenarijų, t.y kelias, atitinkantis vartotojo pasirinkimus, skiriasi skirtingiems vartotojams, nes į vieną viršūnę gali nukreipti keli lankai bei galima patekti į kilpas. Akivaizdu, kad medžio tipo grafo neužtenka, tad šių ryšių sekimui variklyje naudojami du teksto objektų parametrai: ankstesnės viršūnės identifikatorius (ID) ir tolesnės viršūnės ID. Dėl to svarbu, kad kiekviena viršūnė, nepaisant koks jos tipas, turėtų to paties formato identifikatorių. Patogumo ir trumpumo dėlei šie ryšiai žymimi atitinkamai *tėvoID* ir *vaikoID*, tačiau nereiškia tarpusavio priklausomybės ir yra tiesiog grafo struktūros sudarymą nurodantys parametrai.

Toliau pateiktas smulkus teksto objektų aprašymas apibrėžia logiką ir tvarką, kuria remiantis yra formuojami misijų grafai, taip pat ir minėtų ryšių savybių naudojimą.

3.2.1 Misijos pradžia (M)

Galutiniam sistemos vartotojui misijos pradžia yra vizualinės paskirties vienetas - tai tekstu pateiktas pavadinimas, galintis apibendrinti arba nurodyti, ką žaidėjas turi atlikti. Misijos pradžios tipo tekstinio vieneto pavyzdžiai – „Studijos universitete“ arba „Rask vietą nakvynei“. Techniškai, misijos pradžia yra grafo viršūnė, kartu ir pačios misijos identifikatorius, pagal kurį galima atskirti vieną grafą (misiją) nuo kito. Tad kalbant apie misijos pradžią kartu kalbama ir apie pačią misiją. Šiam teksto objektui priskiriamos tokios savybės:

- Unikalus sveikasis skaičius - identifikatorius. Jis išsprendžia problemą, kai dvi misijos turi tokį patį tekstinį pavadinimą, be to, skaitinis žymėjimas kompiuterinėje sistemoje yra efektyvesnis (išrinkimo, palyginimo prasme) nei tekstinė informacija;
- Tekstinis pavadinimas, kuris bus matomas vartotojui – vartotojas supras kurią misiją vykdo ne pagal jo identifikatorių, bet būtent pagal tekstinę informaciją;
- Kompiuterinio personažo, kuriam priskirta ši misija, ID arba virtualios aplinkos (kambario), kuriam priskirta ši misija, ID. Misijos priskyrimas kompiuteriniam

personažui reiškia, kad misiją vartotojas pradės susitikęs ir „prakalbinęs“ šį personažą, o priskyrimas aplinkai reiškia, kad misija prasidės vartotojui patekus į nurodytą aplinką. Naudojamas vienas ir tik vienas iš šių dviejų parametrų, kadangi žinant KP identifikatorių, apibrėžta tampa ir aplinka, kuriai priklauso tas KP;

- Kartojamumas. Šio parametro reikšmė nurodo kiek kartų ši misija gali būti vykdoma pakartotinai. Numatytoji reikšmė turėtų būti 1, kas reikš jog vieną kartą pasiekus misijos pabaigą daugiau vartotojas jos vykdyti nebegalės, tačiau ši savybė leidžia realizuoti ir N kartų kartojamas misijas, bei tokias, kurios gali būti daromos neribotą kiekį kartų (nurodoma 0).
- Vykdomumas, loginio tipo parametras, apibrėžiantis ar šiuo elementu identifikuojama misija yra matoma vartotojo misijų sąraše (žr. 3.3 sk.), ar yra tiesiog pokalbis virtualioje erdvėje, nesusijęs su naujais nurodymais, užduotimis vartotojui. Tokio tipo misijomis talpinama tekstinė informacija, nereikalaujanti kokių nors veiksmų vykdymo. Taigi, jei prakalbintas KP turi matomą (vartotojui) tekstinį dialogą, tačiau tas dialogas nenukreipia į pokalbius su kitais KP ar kitas aplinkas, bei nėra susietas ryšiais su jau vykdomomis misijomis, tai dialogo grafo šakninė viršūnė turės vykdomumo parametrą nustatytą į 0. Pavyzdžiui, virtualūs informacijos darbuotojai gali suteikti vartotojui aktualios informacijos, atsakyti į klausimus, tačiau tai nebus susiję su kažkokių užduočių vykdymu, tad vykdomumo parametras bus 0.

Misijos pradžiai reikalingas tik *vaikoID* ryšys, kadangi tai yra šakninė misijos grafo viršūnė. Šiam tipui *vaikoID* turi nukreipti tik į antrojo tipo teksto objektą – pirmąją misijai priklausančią *užduotį*.

3.2.2 Užduotis (T)

Užduotis yra misijos žingsnis, nurodantis, ką tuo metu vartotojas gali ar turi atlikti. Misija turi neribotą maksimalų užduočių kiekį, tačiau privalo turėti bent vieną. Be to, visos užduotys misijos pradžios (*M)* atžvilgiu yra tame pačiame hierarchiniame lygyje.

Tarus, kad sistemoje egzistuoja misija „Studijos universitete“, pirmosios užduoties tekstas galėtų būti „pateikti stojimo paraišką“, o paskutinės – „atsiimti diplomą“. Tarp jų, galėtų būti tokios užduotys kaip „lankyk paskaitas“, „išlaikyk egzaminą“ bei „susimokėk už mokslus“ tiems, kurių rezultatai nepakankamai geri kompensuojamoms studijoms. Žinoma, užduoties tekstas gali būti ir daug detalesnis ir nebūtinai turi būti trumpas – tą spręsti turi simuliacijos scenarijaus

autoriai. Kaip ir misijos pradžios (*M)* atveju, šis misijos elementas turi sąrašą reikalingų savybių:

- Unikalus identifikatorius. Vėlgi, efektyviau ir patogiau scenarijaus elementus kompiuteriu valdyti atpažįstant juos pagal unikalų kodą.
- Tekstinė informacija, formuluojanti veiksmą ar veiksmus, kurie turi įvykti, kad vartotojas pasiektų tolesnę užduotį arba misijos pabaigą. Paprasčiausiu atveju nurodytus veiksmus turės atlikti pats vartotojas.
- Virtualios aplinkos (kambario), kuriam priskirta ši misija, ID, kuri gali būti ir nenurodoma. Aplinkos nurodymo atveju, ta užduotis (žingsnis) bus pateikta, vartotojui nukeliavus į tą aplinką. Kompiuterinio personažo nurodymas šiam tipui nėra reikalingas.

Ryšyje *tėvoID* visuomet nurodomas misijos pradžios ID, nepriklausomai ar tai pirma ar n-toji tos misijos užduotis. Kodėl to užtenka ir kokiu būdu išaiškinamas užduočių tarpusavio eiliškumas, paaiškinta 3.5 skyriuje. Parametras *vaikoID* gali būti įvairus ir yra privalomas:

- Nuoroda į klausimą (*Q*). Tokiu atveju užduoties vykdymas susijęs su konkrečiu tekstiniu dialogu.
- Nuoroda į kitą užduotį (*T*) – tarp dviejų užduočių nėra tarpinių viršūnių. Šiuo atveju tolesnės užduoties pasiekimas įvyks, kai vartotojas tenkins tai tolesnei užduočiai priskirtas sąlygas (žr. 3.3 sk.).
- Nuoroda į misijos pabaigą (*E*); Tai paskutinis misijos žingsnis, kurio įvykdymas taip pat reikalauja nurodytų sąlygų.

3.2.3 Klausimas (Q)

Klausimo tipo objektas talpina tekstą, kurį „sako“ virtualus kompiuterinis personažas tam tikru momentu. Tas tekstas nebūtinai turi būti klausiamojo pobūdžio sakinys su klaustuku. Toliau tekste minimas žodis „klausimas“ reiškia būtent šio tipo objektą, o ne klausiamąjį sakinį. Toks tipo pavadinimas parinktas dėl to, kad šiame modelyje į kiekvieną KP tekstą – ar tai būtų teiginys, ar klausimas, ar bet kokia informacija – bendru atveju egzistuoja mažiausiai vienas tekstu išreikštas vartotojo pasirinkimas (kitaip - atsakymas). Klausimas, kartu su jam priskirtais atsakymais, sudaro dialogo tarp virtualių personažų ir žaidėjo, pagrindą.

Kaip minėta 3.1 skyriuje, kompiuteriniu personažu gali būti ir į žmones nepanašūs virtualaus pasaulio objektai, pavyzdžiui, skelbimų lenta. Taigi, ir tokioje lentoje rastas pranešimas variklyje yra traktuojamas kaip klausimo tipo objektas. Tokiu atveju klausimą galima panaudoti ir kaip interaktyvumo šakojimosi aklavietę, jeigu nei vienas atsakymas lieka

nenurodytas: objektui *Q* priskirta informacija bus pateikta, tačiau jokių pasirinkimų ir tolesnių šakų nebus – teliks tik grafinės sąsajos numatytais būdais pašalinti tą informaciją iš ekrano – pavyzdžiui, uždaryti skelbimo lentą vaizduojantį langą.

Kiekvienam klausimui turi būti apibrėžti tokie parametrai:

- Unikalus identifikatorius.
- Tekstinė informacija, kuri bus vaizduojama ekrane, tarsi ją sakytų kompiuterinis personažas.
- Kompiuterinio personažo, kuriam priklauso šis tekstas, ID.
- Maksimalus atsitiktinių atsakymų kiekis. Šis skaičius nurodo, kiek daugiausiai atsakymų (*A*), kurie turi savybę *atsitiktinumas*, bus vaizduojama vartotojui. Prasmę ši savybė įgyja tada, kai klausimas sutinkamas pakartotinai – patirties įvairovei ir kintamumui (dinamikai) sukurti. Jei visi atsakymai yra rodomi kiekvieną kartą, šio parametro reikšmė nurodoma 0.

Klausimo ryšiai su kitų tipų tekstiniais objektais yra įdomūs tuo, kad turint konkretų atskirą klausimą, negalima iš anksto nustatyti iš kur į jį bus atkeliauta (nes keli skirtingi keliai misijos grafe gali vesti į tą patį klausimą). Šiam teksto objekto tipui *tėvoID* nurodyti nereikia. Toliau, visos iš klausimo išeinančios vaikinės viršūnės, t.y atsakymai (*A*), negali būti sutalpinti į parametrą *vaikoID*, nes parametras *vaikoID* yra tik vienas, o atsakymų kiekis – neribojamas, tad *vaikoID* klausimams taip pat nėra nurodomas. Dėl to, klausimo (*Q)* vieta misijos grafe nustatoma pagal į tą klausimą vedantį elementą, o išeinančios viršūnės – pagal atsakymus, turinčius klausimo ID savo ryšyje *tėvoID*.

3.2.4 Atsakymas (A)

Kaip teigia pats tipo pavadinimas, atsakymo paskirtis yra išspręsti užduotą klausimą ar suformuluotą problemą. Atsakymas šiame modelyje – pagrindinis vartotojo interaktyvumo šaltinis, dinamikos pagrindas, reiškiantis vieną konkretų pasirinkimą tam tikroje situacijoje ir galintis nukreipti į skirtingas scenarijaus šakas. Atsakymai priskiriami konkrečiam klausimui, taip sukuriant virtualų dialogą tarp kompiuterinių personažų ir realaus asmens, kuris pats sprendžia kaip reaguoti, rinkdamasis vieną iš numatytų, t.y scenaristų sukurtų atsakymų.

Tekstinė atsakymo informacija kiekvienam klausimui yra iš anksto paruošta ir turi atspindėti to pasirinkimo paskirtį, pobūdį, iš dalies ir pasekmes. Pavyzdžiui, jei *Q* tipo objekto tekstas yra „Gal nori lengvai ir greitai užsidirbti? Žinau vieną slidų bet pelningą darbelį“,

vartotojo pasirinkimai galėtų būti:

a) „Papasakok daugiau, mane tai domina“.
b) „Ne, ačiū, čia kvepia nelegalia veikla, man tai nepatinka“.
c) „(Patraukyti pečiais)“.

Iš esmės galima išskirti dviejų tipų tekstinę atsakymo informaciją: žodžiai, kuriuos vartotojas pasirenka „atsakyti“ dialogo metu (a, b), bei tekstas, apibūdinantis vartotojo, ar jo įsikūnijimo, veiksmus (c). Techniškai tokie atsakymai niekuo nesiskiria – skirtumą pasirinkimo prasmėje pajus tik vartotojas. Kaip ir kitiems tekstiniams scenarijaus elementams, atsakymui priskiriama parametrų aibė:

- Unikalus identifikatorius.
- Tekstinė informacija, vartotojui apibūdinanti šio pasirinkimo prasmę.
- Eiliškumas – atsakymo numeris, lyginamas su kitais to paties klausimo atsakymų numeriais. Leidžia nustatyti tarpusavio vaizdavimo tvarką.
- Atsitiktinumas. Jei šios savybės reikšmė – 0, tai atsakymas bus rodomas kiekvieną kartą, kai bus rodomas ir klausimas, kuriam atsakymas priklauso. Jei atsitiktinumas lygus 1, tai bus rodomas tik klausimui nustatytas maksimalus atsitiktinių atsakymų kiekis, atsitiktinai parinkus juos iš visų, turinčių reikšmę 1. Pavyzdžiui, jei vartotojas ateina į virtualią valgyklą, klausimas „ko pageidausite pietums?“ galėtų turėti vieną pastovų (neatsitiktinį) atsakymą „ačiū, nieko“, o dar 5 atsakymai galėtų būti atrenkami atsitiktinai iš didelio meniu atsakymų sąrašo, taigi kiekvieną kart apsilankius būtų pateikiamas vis kitoks meniu.
- Nuorodų į kitas misijas sąrašas. Atsakymo pasirinkimas gali reikšti vienos arba daugiau misijų pradžią, nors misija, kurios metu (kurios grafe) sutiktas atsakymas dar nesibaigia. Šis, dažnai tuščias masyvas, gali talpinti neribotą (labai didelį) misijų pradžių *M* kiekį.

Kiekvienas atsakymas priklauso konkrečiam klausimo objektui, taigi *tėvoID* nurodomas to klausimo ID. Pagal tai yra išrenkami kiekvienam klausimui priskirti atsakymai – ieškoma, atsakymų, *tėvoID* ryšyje turinčių to klausimo ID.

Atsakymas gali nukreipti į įvairaus tipo teksto objektus – kitą klausimą (*Q)*, tolesnę užduotį (*T)*, arba misijos pabaigą (*E)*, priklausomai nuo pasirinkimo paskirties ir misijos struktūros. Ryšys *vaikoID* nurodo į atitinkamą objektą, o jeigu paliekamas tuščias, tiesiog nutraukia dialogą.

3.2.5 Pabaiga (E)

Paskutinis tekstinio scenarijaus elemento tipas yra misijos pabaiga. Misijos grafui pasiekus šią viršūnę, vartotojui pateikiama jai priklausanti tekstinė informacija, apibendrinanti pasiektą rezultatą, įvykdytus žingsnius. Pabaigą turi tik visa misija, bet ne atskira jos dalis. Kiekviena misija, net ir turinti vykdomumo parametrą lygų 0, turi turėti pabaigą, kuri varikliui praneš, kad su šia misija daugiau veiksmų nebus atlikta. Be to, šį modelį taikant nelinijinėms užduotims, misijos turės daugiau nei vieną, tarpusavyje skirtingos prasmės pabaigą. Tokio tipo objektą apibrėžiančios savybės yra:

- Unikalus identifikatorius.
- Tekstinė informacija, kuri pasirinktinai gali būti tiesiog pranešimas, jog misija baigta vykdyti, arba išsamesnis su vykdyta misija susijęs komentaras.
- Įvykdymo rezultatas. Tai skaičiumi, nelygiu nuliui, išreikštas įvykdymo įvertis. Rekomenduojama laikytis principo, kad misija laikoma sėkmingai įvykdyta tuomet, kai pasiekta pabaiga turi teigiamą rezultatą, ir atitinkamai laikyti jog misijos įvykdyti nepavyko, jei pasiektos pabaigos rezultatas yra neigiamas.
- Kartojimo draudimas. Šią savybę turinčios pabaigos nurodo, kad įvykdyta misija nebegalės būti daugiau vykdoma, ignoruojant ar misijos pradžios (*M)* savybės kartojamumas reikšmę. Tokia pabaiga naudinga įgyvendinant scenarijų, kur vartotojas gali neribotą kartų bandyti teisingai atlikti užduotis, tačiau kai jam tai pavyksta, daugiau kartų bandyti (vykdyti misiją) nebelieka tikslo.
- Nuorodų į kitas misijas sąrašas, kurio principas identiškas atsakymo nuorodų sąrašui.

Pabaiga priskiriama vienai konkrečiai misijai, tad *tėvoID* reikšmė – misijos pradžios (*M)* identifikatorius.

Nors dažnai vienos misijos įvykdymas gali vesti prie vienos misijos pradžios, ir niekur daugiau, *vaikoID* šis ryšys nenurodomas. Tokios naujos misijos pradžios (*M*) ID tuomet įtraukiamas į „nuorodų į kitas misijas sąrašą“ ir yra vienintelis elementas sąraše. Papildomai, modelyje numatyta galimybė, kad misijos pabaiga (*E)* aktyvuoja tolesnį kitos misijos įvykį, o konkrečiai – naują užduotį (*T)* arba jos pabaigą (*E)*. Tad *vaikoID* gali nukreipti tik į kitai misijai priklausančią užduotį ar pabaigą.

Kaip misija sujungiama į jungų grafą, kaip apibrėžiamas tekstinių objektų priklausomumas konkrečiai misijai bei kaip misijos susiejamos tarpusavyje, aprašoma 3.2.6 poskyryje

3.2.6 Scenarijaus elementų sujungimas– misijų formavimas

Apibrėžus misijos elementus galima suformuoti tą misiją atitinkantį grafą. Į vieną misiją būtinai įeina:

- Vienas ir tik vienas misijos pavadinimas, kartu identifikuojantis ir pačią misiją;
- Viena arba daugiau pabaigų (*E*);
- Viena arba daugiau užduočių (*T*);

Klausimų (*Q*) ir jų atsakymų (*A*) skaičius nėra ribojamas. Šio tipo vienetų gali net ir nebūti. Jei misiją sudarys tik viena užduotis, o klausimų ir atsakymų (dialogų) nebus, tuomet tai bus mažiausia įmanoma taisyklinga misija, kurios struktūra pavaizduota 10 pav. (a) schemoje.

Visa grafo struktūra nėra papildomai saugoma, kadangi vykdant misijas ji nėra reikalinga - įtakos turi tik konkrečiu metu prieita viršūnė iš jos išeinančios viršūnės, tad užtenka žinoti kaip parinkti tolesnę viršūnę. Visgi, bendras misijos vaizdas reikalingas tas misijas kuriantiems ar redaguojantiems žmonėms, bei paties modelio supratimui. Scenarijaus elementų (tekstinių objektų) tarpusavio ryšių įvairovė nurodyta 2 lentelėje. Lentelėje kableliu atskiriamos alternatyvos, o brūkšnelis simbolizuoja ryšio nebuvimą. Bet kuriuo momentu iš teisingai apibrėžtų *M, T, Q, A, E* tipų elementų, galima nubraižyti misijos struktūrą atitinkantį grafą, remiantis tokia veiksmų seka:

1. Misijos pradžią atitinkanti viršūnė – tai *M* tipo objektas.
2. *M* tipo objektas veda į pirmąją užduotį, t.y *vaikoID* atitinka *T* objekto identifikatorių, tad šios dvi viršūnės sujungiamos lanku (lankas visada nukreiptas iš „tėvo“ į „vaiką“).
3. Užduotis nukreipia į *Q, T* arba *E*; bet kokiu atveju tas objektas lanku sujungiamas su *T* objektu.
4. Jei prijungta viršūnė yra *T* tipo, kartojamas trečias žingsnis.
5. Jeigu šis objektas yra klausimas *Q*, tai ieškoma A tipo atsakymų, kurių *tėvoID* lygus to klausimo ID, ir kiekvienas surastas atsakymas prijungiamas lanku, išeinančiu iš klausimo.
6. Kiekvienam atsakymui *A* prijungiamas *vaikoID* nurodytas tekstinis objektas, o tolesnis žingsnis gali būti 3 - jei *vaikoID* atitinka užduotį *T*, 5 – jei nukreipiama į kitą klausimą *Q* ir 7 – jei vaikoID rodo į pabaigą *E*. Jei atsakymui vaikoID nenurodytas, tai ta šaka daugiau nebepildoma.
7. Nesvarbu iš kur patenkama į *E* tipo viršūnę, ta viršūnė prijungiama prie grafo. Tuomet grafe ieškoma *T, Q* ir *A* tipo viršūnių, iš kurių nėra išeinančių lankų ir remiamasi ankstesniais žingsniais (atitinkamai 3, 5 ir 6). Kai neapžiūrėtų viršūnių nebelieka – grafas

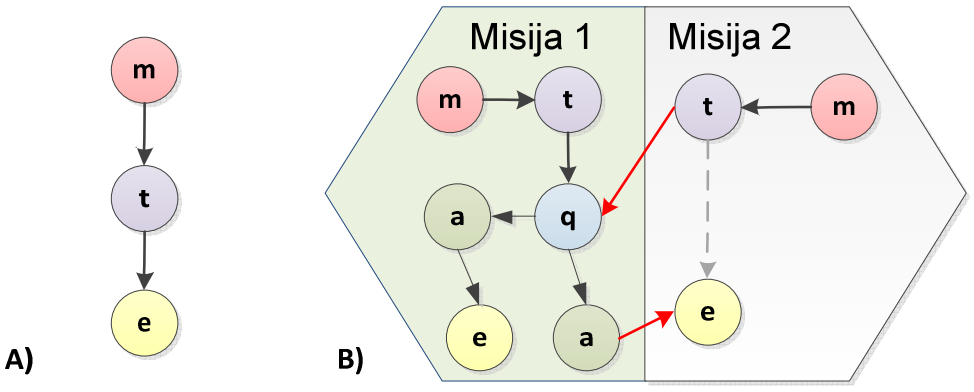
pilnai sudarytas.

Kiekvienas taisyklingai suformuotos misijos kelias baigiasi *E* viršūne, reiškiančia vieną iš misijos pabaigų arba atsakymu (*A*), neturinčiu *vaikoID*. Pabaigoms gali būti nurodomi *vaikoID*, bet jie nurodo dviejų misijų tarpusavio ryšį.

2 lentelė. Tekstinių scenarijaus elementų tarpusavio ryšiai

	M	***T***	***Q***	***A***	***E***
tėvoID	-	*M*	-	*Q*	*M*
vaikoID	*T*	*Q, T, E*	-	*-, Q, T, E*	*-,T, E*
misijosID	*M*	*M*	*M*	*M*	*M*

Prieš aptariant misijų tarpusavio sujungimą, reikia atkreipti dėmesį į 2 lentelėje nurodytą ir iki šiol neaptartą ryšį *misijosID*. Dėl ryšių *tėvoID* ir *vaikoID* nevienareikšmiškumo sudėtingėja vieno tekstinio objekto priklausomybės konkrečiai misijai nustatymas. Toks nustatymas tampa neįmanomas, jei *Q* arba *A* tipo tekstinis objektas yra izoliuojamas. Šiai problemai išspręsti ir įvedamas *misijosID*, laukas, o jo reikšmė visuomet atitinka misijos, kurios grafe turi būti randamas scenarijaus elementas, pavadinimo ID. Nors užduoties ir pabaigos tipams šis laukas dubliuojasi su *tėvoID*, tačiau *misijosID* vienareikšmiškumas įveda aiškumą ir paprastumą nustatant konkrečios misijos ribas. Be to, turint šį papildomą ryšį, lengviau išspręsti įvairias problemas, pavyzdžiui, paprasčiau programiškai uždrausti galimybę iš vienos misijos grafo nukreipti į kitai misijai priklausantį klausimą.



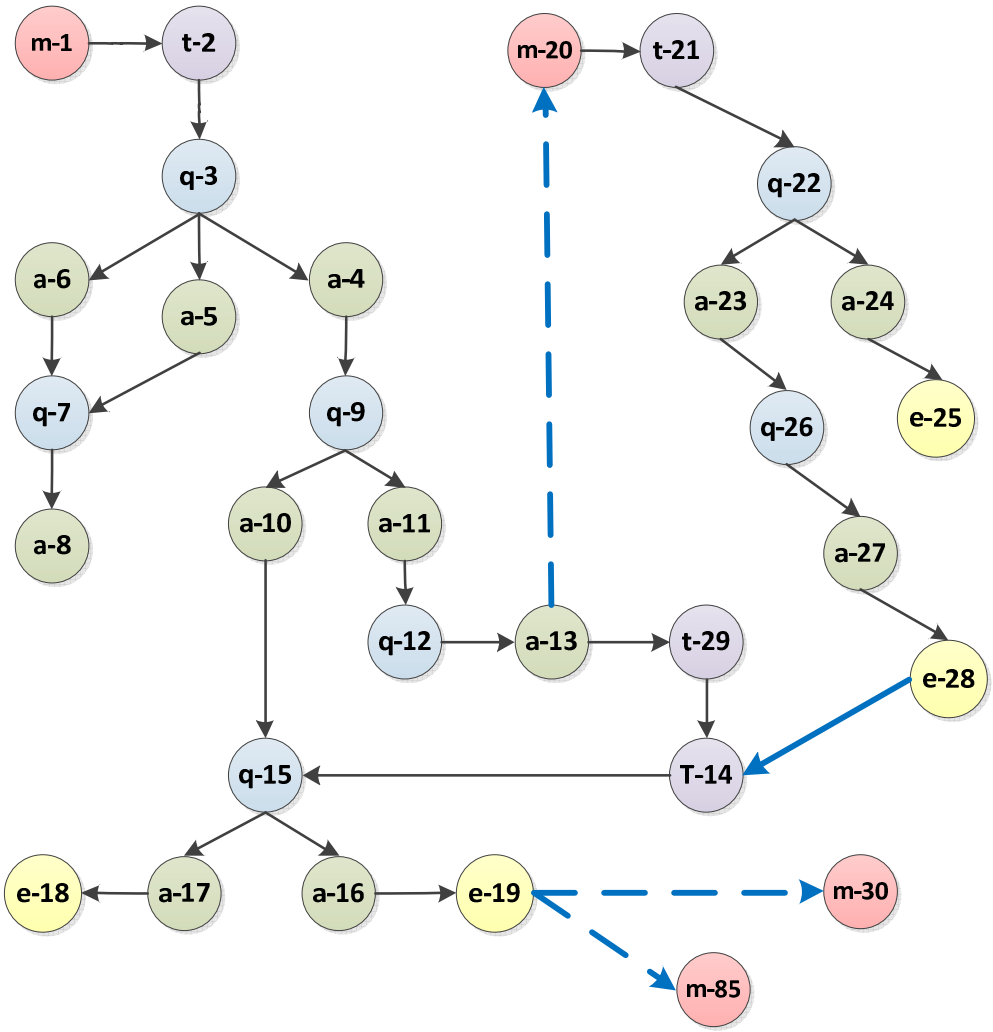
10 pav. Trumpiausia galima misija (a) ir neteisingai sujungtos viršūnės (b)

Apskritai, formuojant misiją, ryšiuose *tėvoID* ir *vaikoID* draudžiama nurodyti kitą *misijosID* turinčius scenarijaus elementus, išskyrus *E* objekto *vaikoID* atvejį. Taigi, tik misijos pabaigos gali nukreipti į kitą misiją ryšiuose *tėvoID/vaikoID*. Visgi, net ir šiuo atveju rekomenduojama išorinį ryšį perkelti į tam skirtą naujų misijų sąrašą, kad minėti du ryšiai visuomet atitiktų tik

vidinius misijos ryšius. Lankai, 10 pav. (b) schemoje pavaizduoti raudonomis rodyklėmis yra neleistini – čia antrosios misijos užduotis nukreipia į pirmajai priklausantį klausimą, o to klausimo atsakymas nukreipia atgal į antrosios misijos pabaigą. Net jeigu abejoms misijoms to klausimo tekstas yra identiškas, reikia sukurti po vieną *Q* tipo objektą kiekvienai misijai, kad misijos grafo viduje nebūtų lankų, nukreipiančių į išorines viršūnes.

3 lentelė. Dviejų misijų sąryšio pavyzdys duomenų pavidalu

ID	Tipas	Tekstas (pavyzdys)	*tėvoID*	*vaikoID*	*misijosID*	Naujos M
1	**M**	Pateik darbdaviui reikalingą informaciją	-	2	1	
2	**T**	Pakalbėk su buhaltere.	1	3	1	
3	**Q**	Sveiki, kuo galėčiau padėti?	-	-	1	
4	**A**	Direktorius prašė pateikti informaciją.	3	9	1	{}
5	**A**	Oi, nepasiėmiau dokumentų. Grįšiu vėliau.	3	7	1	{}
6	**A**	Niekuo, ačiū.	3	7	1	{}
7	**Q**	Na gerai, kai reikės - kreipkitės.	-	-	1	
8	**A**	Būtinai, iki.	7	-	1	{}
9	**Q**	Man reikės Jūsų soc. draudimo numerio.	-	-	1	
10	**A**	Čia šitam pažymėjime turbūt parašytas?	9	15	1	{}
11	**A**	Tai kad neturiu aš tokio...	9	12	1	{}
12	**Q**	Teks nuvykti į Sodrą ir įsigyti.	-	-	1	
13	**A**	Gerai.	12	29	1	{20}
14	**T**	Grįžk pas buhalterę.	1	15	1	
15	**Q**	Matau turite pažymėjimą, parodykit prašau.	-	-	1	
16	**A**	Prašom. (Paduoti)	15	19	1	{}
17	**A**	Ai, atsibodo man tas popierizmas, išeinu!	15	18	1	{}
18	**E**	Atsisakei pateikti informaciją.	1	-	1	{}
19	**E**	Sėkmingai pateikei informaciją.	1	-	1	{30, 85}
20	**M**	Įsigyk socialinio draudimo pažymėjimą.	-	21	20	
21	**T**	Nuvyk į Sodrą.	20	22	20	
22	**Q**	Laba diena. Ko norėsite?	-	-	20	
23	**A**	Atėjau soc. draudimo pažymėjimo.	22	26	20	{}
24	**A**	Nieko, einu jau.	22	25	20	{}
25	**E**	Negavai pažymėjimo.	20	-	20	{}
26	**Q**	Tuoj atnešiu jau pagamintas. Štai, prašom.	-	-	20	
27	**A**	Ačiū, viso gero.	26	28	20	{}
28	**E**	Įsigijai socialinio draudimo pažymėjimą.	20	14	20	{}
29	**T**	Gauk reikiamą dokumentą.	1	14	1	
30	**M**	Dirbk iki dienos pabaigos	-	30	30	
	...		...	...	30	
85	**M**	Paprašyk didesnio atlyginimo	-	86	85	
	...		...	...	85	



11 pav. pagal 3 lentelės duomenis sudarytas misijų grafas

Kaip ir ankstesnėse schemose, juodais lankais pavaizduoti tos pačios misijos elementų ryšiai. Grafe mėlynu lanku parodyta, kad iš misijai *m-20* (čia skaičius 20 tiesiog nurodo to tekstinio objekto identifikatoriaus reikšmę) priklausančios pabaigos *e-28*, nukreipiama į misijai *m-1* priklausančią užduotį *t-29*, taip susiejant misijų vykdymą. Punktyriniais mėlynais lankais pažymėti nukreipimai į naujas misijas, kurių masyvai yra išskirtinai atsakymų (*A*) ir pabaigų (*E*) savybė. 3 lentelėje stulpelis „Naujos M“ nurodo tų masyvų reikšmes šiame pavyzdyje.

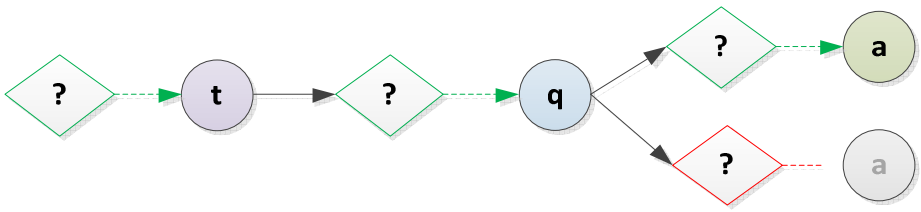
Žvelgiant į lentelę lengva pastebėti, jog vienas jos tekstinis objektas, *t-29*, yra ne prie tos pačios misijos įrašų. Tokie duomenys parinkti norint pabrėžti, jog misiją apibrėžiančių duomenų tęstinumas nėra reikalingas, ir ID dydis nėra naudojamas nustatymui „kas toliau“. Dėl tokios savybės į misiją įterpti naujas viršūnes (užduotis, klausimus ir atsakymus, pabaigas) visuomet paprasta – naujam elementui priskyrus unikalų skaičių, tereikia pakoreguoti šalimai atsidursiančių tos misijos elementų ryšius.

Pademonstruotais dviem būdais – naujų misijų masyvais ir pabaigų *vaikoID* ryšiais - misijos jungiamos tarpusavyje, tad visą socialinės simuliacijos scenarijų galima įsivaizduoji kaip vieną didelį grafą. Tiesa, nejungų, nes labai tikėtina, kad dalis misijų bus visiškai nepriklausomos ir nesusijusios su kitomis. Tačiau, vien grafo struktūros pilnam variklio funkcionalumui apibrėžti neužtenka.

3.3 Sąlygos

Vien tik pažvelgus į 3 lentelėje pateiktus atsakymus *a-10* ir *a-11*, iškyla klausimas, kodėl vartotojas turėtų rinktis variantą, jog pažymėjimo neturi, jeigu gali pasirinkti priešingai ir pagreitinti misijos įvykdymą. Atsakymas – todėl, kad vartotojui vienu metu bus pasiekiamas tik vienas iš tų pasirinkimų, atsižvelgiant į tai, ar savo daiktuose (žr 3.1.1 poskyrį) jis iš tiesų turi reikiamą pažymėjimą imituojantį objektą.

Kitaip tariant, ne visas misijos grafas tam tikru momentu bus prieinamas konkrečiam žaidėjui. Aplinkybės, nurodančios, kada vartotojas gali pasiekti tekstinį objektą, vadinamos sąlygomis (12 pav.). Jos padeda sukurti virtualų pasaulį, kur kiekvieno individo, priklausomai nuo jam priskirtų parametrų, patirtis skiriasi nuo kitų individų, turinčių kitokius parametrus.



12 pav. Sąlygos, įtakojančios misijos grafo viršūnių prieinamumą vartotojui

Prieš tai nagrinėtame pavyzdyje (11 pav.), sąlygos neegzistavo, tiksliau nebuvo nurodytos. Kuriant misiją galima priskirti tam tikras sąlygas bet kuriam iš scenarijaus elementų: misijos pradžiai, užduočiai, klausimui, atsakymui bei pabaigai. Pagal jų prasmę galima išskirti 6 modelyje numatomus sąlygų tipus. Kad vartotojas pasiektų scenarijaus elementą, turi būti tenkinamos sąlygos, nusakančios:

1) Reikiamas konkrečių vartotojo profilio parametrų reikšmes.
2) Reikiamą konkrečių vartotojo turimų virtualių daiktų kiekį.
3) Misijų vykdymo rezultatus, pavyzdžiui, turi būti įvykdyta nurodyta misija ar prieita jos užduotis.
4) Laiko intervalą. Turi būti jau praėjęs (arba dar nepraėjęs) tam tikras realaus arba virtualaus laiko tarpas. Tie įvykiai gali būti ir misijų vykdymo rezultatai, skirtumas

tas, kad 3 tipo sąlygoms neturi įtakos, kada tie rezultatai pasiekti.

5) Specifinius simuliacijoje turinčius įvykti įvykius, kurie gali būti susiję tiek su vartotojo parametrais, tiek su daiktais. Pavyzdžiui, kambaryje, kuriame tuo metu yra vartotojas, turi būti prisijungęs tam tikras skaičius kitų vartotojų. Arba sąlyga, reikalaujanti nusipirkti bet kuriuos 3 skirtingus maisto produktus; Tokių sąlygų kiekis variklio realizacijoje priklauso nuo reikalavimų kuriamam produktui, bei scenarijaus sąsajų su vartotojo veiksmų įvairove.
6) Vietą – virtualų kambarį. Šio tipo sąlygų tenkinimą išsprendžia *M* ir *Q* tipo scenarijaus elementų priskyrimas aplinkai ar kompiuteriniam personažui. Kadangi tokia priklausomybė apibrėžta kaip scenarijaus elementų savybė, jokių vietos tipo sąlygų papildomai nurodyti nereikia.

Kiekvienam scenarijaus elementui galima priskirti neribotą kiekį 1-5 tipo sąlygų ir jei nors viena iš tų sąlygų nebus tenkinama, priklausomai nuo tekstinio objekto tipo, vartotojas:

- nepradės naujos misijos vykdymo, jei tai misijos pradžia (*M)*;
- nepajudės į tolesnę užduotį ir liks dabartiniame žingsnyje, jeigu tai užduotis (*T)*;
- nematys dialogo (pašnekesio su kompiuteriniu personažu), kurio pradžią atitinka sąlygų netenkinantis klausimas (*Q)*;
- nematys pasirinkimo, kurį atitinka atsakymas (*A)*;
- nebaigs vykdyti misijos, nors visi galimi veiksmai jau atlikti, jei tai pabaiga (*E)*.

Kiekviena sąlyga formuluojama priskiriant jai reikalingas savybes ir susiejant su scenarijaus elemento, kuriam ta sąlyga taikoma, ID. Sąlygos neįtraukiamos į pačių teksto objektų aprašus todėl, kad sistema ir vartotojas neturi sužinoti dalies to scenarijaus elemento savybių, įskaitant tekstinę informaciją, jeigu sąlygos nėra tenkinamos. Taigi, elemento sąlygos ir pats elementas yra logiškai ir fiziškai atskirti ir sąlygos, priklausomai nuo tipo, yra įtraukiamos į vieną iš keturių sąlygų matricų.

3.3.1 Vartotojo parametrų sąlygų matrica

Kiekviena eilutė dvimatėje parametrų matricoje apibrėžia vartotojo parametrų aibės elementams keliamus reikalavimus. Vieną 1 tipo sąlygą aprašančioje eilutė nurodoma:

- Scenarijaus elemento, kuriam taikoma ši sąlyga, identifikatorius.
- Reikiamą vartotojo parametrą identifikuojanti klasifikatoriaus reikšmė – parametro kodas. Aiškumui užtikrinti gali būti naudojamas parametro pavadinimas, tačiau variklio

realizacijoje rekomenduojama naudoti sveikojo skaičiaus tipo klasifikatorių, kurio reikšmės atitinka konkretų vartotojo parametrą.

- Reikšmė, šiuo atveju reiškianti reikiamą nurodyto parametro reikšmę.
- Reikšmės palyginimo tipas. Pagal šią sąlygos savybę nustatoma koks aritmetinis operatorius taikomas vykdant palyginimą su realia vartotojo parametro reikšme ir nusprendžiant, ar ta sąlyga tenkinama. Galimi palyginimo tipai pateikti 4 lentelėje. Parametrų sąlygoms negalimas palyginimo tipas „nelyginamas“.

4 lentelė. Reikšmių palyginimo būdų klasifikatorius

Klasifikatorius	Operatorius	Aprašymas
0	nelyginamas	Naudojamas sąlygose, kur reiškmės palyginimas nereikalingas
1	=	Reali reikšmė privalo būti lygi sąlygoje nurodytai reikšmei
2	>	Reali reikšmė privalo būti didesnė už sąlygoje nurodytą reikšmei
3	<	Reali reikšmė privalo būti lygi sąlygoje nurodytai reikšmei
4	>=	Sąlygą tenkins didesnė arba lygi reali reikšmė
5	<=	Sąlyga tenkins mažesnė arba lygi reali reikšmė

Norint apriboti parametro reikšmę tam tikrame intervale [*r1, r2*], (*r1, r2*), (*r1, r2*] ar [*r1, r2*), čia *r2, r1*$\in \mathbb{R}$ ir *r2 > r1*, reikia vienam parametrui sukurti dvi sąlygas – vieną, kurios reikšmė *r1*, o palyginimo operatorius > arba >=, ir kitą, kurios reikšmė *r2*, o palyginimo operatorius yra < arba <=.

3.3.2 Vartotojo turimų daiktų sąlygų matrica

Ši sąlygų matrica panaši į parametrų matricą, tačiau vietoj vartotojo parametrų čia nurodomos 2 tipo sąlygos – reikalaujami vartotojo daiktai. Tokios matricos stulpelių reikšmės:

- Scenarijaus elemento, kuriam taikoma ši sąlyga, identifikatorius.
- Virtualaus daikto, kurį simuliacijos metu gali kaip įsikūnijimo nuosavybę turėti vartotojas, unikalus identifikatorius.
- Reikiamas daiktų kiekis – 1 arba daugiau.
- Daiktų kiekio palyginimo tipas, apibrėžiantis palyginimo operatorių, kaip parodyta 3 lentelėje. Šios savybės reikšmė taip pat negali būti „nelyginamas“.

Įmanoma suformuluoti ir sąlygą, kad vartotojas privalo neturėti konkretaus daikto. Tereikia daiktų kiekį nurodyti lygų 0, o palyginimo operatorių – 1;

3.3.3 Užduočių vykdymo sąlygų matrica

Dinamiškas misijų vykdymas priklauso ne tik nuo vartotojo profilį apibrėžiančių parametrų ir daiktų, bet ir nuo iki dabartinio momento priimtų sprendimų, atliktų arba neatliktų užduočių. Užduočių vykdymas apima ir misijų vykdymą ir atskirų jų žingsnių (*T*) aibę. Į užduočių vykdymo sąlygų matricą vedamos 3 ir 4 tipo sąlygos. Jų savybės:

- Scenarijaus elemento, kuriam taikoma ši sąlyga, identifikatorius.
- Scenarijaus elemento, kuris yra šios sąlygos taikinys, ID. Jeigu misijai *m-1* gauti, būtina įvykdyti misiją *m-2*, tai čia bus nurodomas misijos *m-2* identifikatorius.
- Užduočių vykdymo sąlygos tipą nurodanti klasifikatoriaus reikšmė. Rekomenduojama sukurti tokius sąlygų tipus:
  - užduotis (misija) baigta vykdyti;
  - vartotojas vykdo šią užduotį (misiją), tačiau nėra jos užbaigęs;
  - vartotojo užduotyse nėra įrašo apie tokią užduotį/misiją;
  - laiko kiekis nuo užduoties/misijos vykdymo pradžios;
  - laiko kiekis nuo misijos įvykdymo pabaigos;

  Bei palikti galimybę klasifikatorių papildyti naujomis sąlygomis projekto realizacijos eigoje.
- Reikšmė, kurios prasmė interpretuojama skirtingai, priklausomai nuo sąlygos tipo. Laiko sąlygoms – tai laiko kiekis sekundėmis (ar kitais laiko vienetais), o užduoties įvykdymo sąlygai tai misijos pabaigos (*E)* savybė „įvykdymo rezultatas“. Kitoms sąlygoms reikšmė išviso nereikalinga, tad lieka nenurodyta.
- Reikšmės palyginimo tipas (4 lentelė), reikalingas toms sąlygoms, kurioms nurodoma reikšmė.

3.3.4 Specialiosios sąlygos

Specialiosios sąlygos yra visos sąlygos, pagal prasmę netinkančios į prieš tai nurodytas matricas. Tokių sąlygų simuliacijoje gali išvis nebūti, arba būti labai daug – priklausomai nuo kuriamo produkto. Kaip ir užduočių sąlygų atveju kuriamas klasifikatorius kuriame nurodyti tokių sąlygų tipai. Tokių tipų pavyzdžiai galėtų būti tokie:

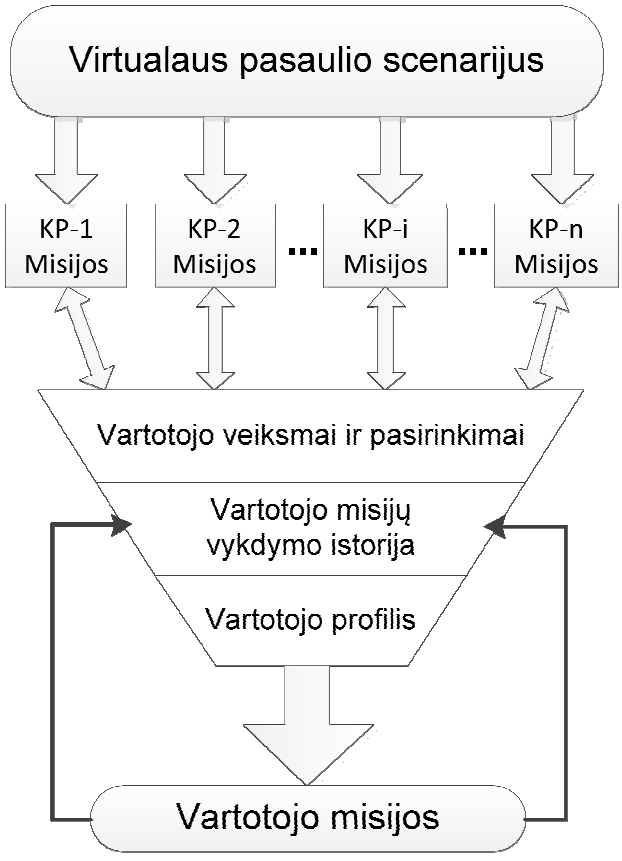
- Atsitiktinumo sąlyga. Bus tenkinama arba ne priklausomai ar jos tikrinimo metu sugeneruotas atsitiktinis skaičius pateks į pasirinkto dydžio intervalą.

- Neapibrėžto (bet kurio) daikto ar daiktų įsigijimas.
- Virtualaus pasaulio parametrai, nepriskirtini vienam vartotojui. Pavyzdžiui prisijungusių vartotojų kiekis.

Matricos stulpelių reikšmės tokioms sąlygos beveik atitiks užduočių vykdymo sąlygų matricos stulpelius – nereikalinga tik sąlygos taikinio savybė. Pasirinktinai šias matricas ir sąlygų klasifikatorius sujungti į vieną.

3.4 Vartotojo ir kompiuterinių personažų misijos

Vienas misijų sąlygų tipas – vietos sąlygos – nėra vedamas kaip ir kitos, kadangi užduočių sprendimas ir virtualus bendravimas vyksta tik susitikus su konkrečiais kompiuteriniais personažais, kurie randami nurodytose jiems aplinkose. Kai vietos sąlygos tenkinamos, t.y vartotojas yra toje aplinkoje, kurioje yra ir KP, o simuliacijos grafikos modulis nenumato kliūčių jų komunikacijai, vartotojas gali „pasiekti“ tas misijas ir klausimus, kurių savybė „kompiuterinio personažo ID“, atitinka sutiktą personažą.



13 pav. Scenarijaus projekcija į vartotojo misijas

Žinoma, vienu metu personažas gali turėti daugiau nei vieną dialogą ar misiją, tad pirmiausia vartotojui turi būti leista pasirinkti, su kuria misija susijusį dialogą nori matyti, ar kurią naują misiją pradėti.

Pasirinkimų skaičius priklauso nuo vartotojo profilio (parametrų, turimų daiktų) bei iki to meto vykdytų misijų. Konkretaus vartotojo misijų vykdymas registruojamas saugant informaciją tik apie dviejų tipų prieitus scenarijaus elementus – misijos pradžias (*M)*, bei užduotis (*T)*.

Kiek vienu metu vartotojas vykdo misijų nėra ribojama (nebent to reikalaus kuriamas produktas). Simuliacijos pradžioje vartotojas gali iš karto turėti numatytąsias misijas – jos įrašomos kuriant žaidėjo įsikūnijimą. Gauti naują misiją vartotojas gali dviem būdais –

„kalbėdamas“ su kompiuteriniais personažais ir pasirinkdamas dar nevykdomą misiją iš KP misijų sąrašo arba gaudamas nukreipimus į naujas misijas turimų vykdymo metu, kaip pademonstruota 3.2.6 poskyriuje. Įvykus šiam veiksmui (ir žinoma, jei tenkinamos misijos pradžiai priskirtos sąlygos) vartotojui įrašoma tokia informacija:

- Prieitos misijos pradžios ID.
- Tikslus laikas, kada tai įvyko (data milisekundžių tikslumu).
- Tikslus laikas, kada misija įvykdyta – pradžioje reikšmė neegzistuoja.
- Įvykdymo rezultatas, kurio reikšmė 0 reiškia, kad misija yra vykdoma.
- Skaičius, reiškiantis kiek kartų ši misija jau buvo įvykdyta (reikalinga pasikartojančioms misijoms), pradžioje įrašoma 0.

Kadangi kiekviena misija privalo turėti ir užduotį (*T*), į kurią nukreipia misijos pradžios elementas, tai automatiškai (iš karto) įrašomas ir dar vienas įrašas su tos pirmosios užduoties ID ir tiksliu laiku, kada ši informacija išsaugota.

Kai bendraudamas su kompiuteriniais personažais vartotojas sutinka nukreipimą į tolesnę vykdomai misijai priklausiančią užduotį, atsiranda ir atitinkamas naujas įrašas. Pasiekus misijos pabaigą atnaujinamas tos misijos įrašas ją užbaigusiam vartotojui, įrašant misijos įvykdymo laiką, pabaigai *e* priskirtą įvykdymo rezultatą ir padidinant įvykdymo kiekį vienetu (jei pabaigai nepriskirtas kartojimo draudimas). Taigi, kad misija įvykdyta sistema sprendžia pagal tai, kad egzistuoja jos užbaigimo data bei įvykdymo rezultatas nelygus 0;

Kompiuterinio personažo siūlomų (matomų) misijų sąrašas iš visų tam personažui priskirtų scenarijaus elementų išrenkamas taip:

a) Rodyti tas misijas (jų pavadinimus), kurių vartotojas šiuo metu nevykdo, jei misijos pradžiai priskirtas sąlygos tenkinamos. Be to, vartotojas dar neturi būti įvykdęs tos misijos kartojamumo parametru nustatytą kiekį kartų; Tokios misijos vadinamos pasirenkamomis, kadangi vartotojas gali pats spręsti ar pradėti tokią misiją ar ne. Apie misijos turinį sprendžiama iš jos pavadinimo teksto;
b) Rodyti misijos pavadinimą kiekvienam KP priskirtam klausimui (*Q*), į kurį nukreipianti užduotis T yra užfiksuota vartotojo užduočių sąraše, bet tik tuo atveju, jei ta užduotis (*T*) yra pati naujausia (pagal įrašymo laiką) iš visų tai pačiai misijai priklausančių vartotojo įrašų. Kitaip tariant, nors ir KP turi su misija susijusį klausimą, jis nebus išrinktas jeigu vartotojas jau yra tolesniame žingsnyje. Į sąrašą neįtraukiami klausimai, kuriems priskirtos sąlygos nėra tenkinamos.

Taigi, dialogai, nesusiję su tolesniu žingsniu vartotojų vykdomų misijų grafe, kaip ir maksimalų kartų kiekį įvykdytos misijos yra atmetamos automatiškai.

Žinoma, dažnu atveju misijos grafe seks keli klausimai iš eilės, nes pašnekesys su virtualiu asmeniu bendru atveju nebus tik vienas klausimas su keletu atsakymų. Tokio pokalbio viduryje išjungus visą simuliaciją (nutraukus sesiją išjungiant programą) ir vėl grįžus pas tą patį KP, bus vėl rodomas pirmasis klausimas, tarsi į jį dar nebūtų atsakyta. Toks sprendimas variklių realizacijose visuomet gali būti pakeistas, ir saugoti galima ir po vieną patį naujausią matytą klausimą kiekvienai misijai, tačiau klausimų neregistravimas turi ir savo privalumų, priklausomai nuo to kaip kuriami misijų medžiai. Kuriant dinamišką scenarijų kiekvienoje misijose ir net klausime leidžiama rinktis, o dalis tų atsakymų veda į aklavietes, kurių prasmė – tuo momentu vartotojas nenori spręsti užduoties taip, kad patektų į tolesnį scenarijaus tašką. Kadangi modelyje nėra saugomas paskutinis prieitas klausimas, tai kitą kartą pakartojęs pokalbį vartotojas galės pasirinkti kelią, nevedantį į aklavietę. Klausimų registravimo atveju, reikėtų kiekvieną aklavietę sieti su misijos pabaiga, o iš tos pabaigos nukreipti atgal į misijos pradžią, taip ne tik prarandant tos misijos progresą, bet ir ženkliai padidinant scenarijaus talpinimo į sistemą darbus (įrašų skaičių).

Dar vienas svarbus su vartotojo užduotimis susijęs principas yra sąlygų tikrinimo laikas. Šiame modelyje nuolatinis sąlygų tikrinimas, kuris reikalautų daug skaičiavimo resursų, nevyksta. Kai vartotojas nukreipiamas į konkrečią viršūnę, patikrinamos jos sąlygos, ir jei jos netenkinamos, nieko neįvyksta. Šiokia tokia išimtis yra specialiosios sąlygos, kadangi tokias sąlygas turinčias viršūnes sesijos metu reikia kaupti, o įvykus tas sąlygas atitinkantiems įvykiams patikrinti, ar sukauptos viršūnės jau gali aktyvuotis. Pavyzdžiui, jei tolesnis misijos žingsnis yra pasiekiamas kai į aplinką virtualiai ateina bent 3 realūs vartotojai, kiekvieną kartą naujam vartotojui atėjus į kambarį, reikia patikrinti ar nėra tokių sukauptų žingsnių, kurių sąlyga atitinka įvykio tipą. Jei yra – patikrini sąlygas pakartotinai.

3.5 Atlygis už užduočių vykdymą

Ne vien įrašai apie vartotojo vykdomas misijas keičiasi simuliacijos metu vykdant užduotis. Pagrindinis stimulas tas užduotis vykdyti – už jų vykdymą gaunamas virtualus atlygis, pasirinkimų vertinimas. Socialinės krypties projektuose pasirinkimų vertinimas yra pagrindinė priemonė tikslinės grupės asmenų (vartotojų) elgesio tendencijoms bei pokyčiams fiksuoti. Tad vartotojo pasirinkimų įtakotiems scenarijaus elementams – atsakymams (A) ir misijų pabaigoms

(*E*) – galima priskirti virtualų atlygį, kuris keičia arba to vartotojų parametrus arba turimus daiktus.

Simuliacijoje egzistuojantys vartotojo parametrai pasirenkami laisvai, ir kiekvienas iš tų numatytų parametrų gali kisti (kinta jo reikšmė). Priskirti scenarijaus elementui parametro pokytį, reiškia suformuluoti įrašą, turintį laukus:

- Scenarijaus elemento (tik *A* arba *E* tipo) ID.
- Parametro, kurio reikšmė keičiama, identifikatorius.
- Reikšmė, kuri vartotojui bus pridėta prie nurodyto parametro reikšmės. Gali būti ir neigiama, tokiu atveju galutinė reikšmė sumažės.
- Misijos įvykdymo kartas (tik *E* tipo elementams), skaičius skirtas pasikartojančių misijų atlygiui diferencijuoti. Nurodo, kelintą kartą įvykdžius misiją šis atlygis aktyvuosis. Nulinė reikšmė naudojama nepasikartojančioms misijoms, o pasikartojančių atveju reikš, kad šis atlygis suteikiamas kiekvieną kartą.
- Perrašymas, kuris nurodo, kad nurodytą parametro reikšmę reikia ne pridėti, o užrašyti vietoj buvusios vartotojo parametro reikšmės.

Toks parametro pokytis įsigalioja tada, kai vartotojas pasirenka atlygį turintį atsakymą, arba kai prieina E tipo scenarijaus elementą – užbaigia užduotį. Atsakymui ar pabaigai priskiriamų parametrų kiekis nėra ribojamas – galima nepriskirti nieko, o galima pakoreguoti ir keletą skirtingų parametrų. Paprasčiausio parametro atlygio pavyzdys – virtualių pinigų suma, gauta už misijos įvykdymą, kai ta misija, pavyzdžiui, vaizduoja darbo mėnesį virtualioje darbovietėje. Atlygio su perrašymu pavyzdys: jeigu nelegalaus darbo metu įsikūnijimas „susižaloja“, tuomet parametro „darbingumas“ reikšmė perrašoma į priešingą, o ne pridedama.

Atlygio principu vykdomas ir vartotojų profiliavimas - už etiniu ar teisiniu požiūriu teigiamus pasirinkimus vartotojui būtų galima duoti „draugiškumo“ ar „savarankiškumo“ taškų, už neigiamai vertinamus – „konfliktiškumo“, „aplaidumo“ taškus ir pan. Jeigu simuliacijos scenarijus yra ilgas, laikui bėgant išaiškės šių parametrų pasiskirstymo reikšmės. Tas reikšmes, ar išvestines jų kombinacijas, vaizduoti galima ir pačiam vartotojui ir, mokomųjų simuliacijų atveju, prižiūrėtojams, analitikams ar konsultantams, vertinantiems žmogaus elgesį ir jo pasekmes simuliacijoje.

Kitas galimas atlygis už vartotojo pasirinkimus, kuris gali būti teikiamas nepriklausomai nuo to, ar elementui jau yra priskirtas parametrų atlygis, yra gaunami arba prarandami virtualūs daiktai. Atsakymui (*A)* arba pabaigai (*E)* virtualių daiktų priskirs kiekvienas įrašas su tokia

informacija:

- Scenarijaus elemento (tik *A* arba *E* tipo) ID.
- Daikto, kuris suteikiamas arba atimamas, identifikatorius.
- Duodamas nurodyto daikto kiekis. Gali būti ir neigiamas, tai reikš daiktų atėmimą. Nulinė reikšmė žymi visų tokių daiktų atėmimą, kiek tik jų tuo momentu turi vartotojas, kadangi dėl scenarijaus netiesiškumo kartais neįmanoma žinoti, kiek tokių daiktų vartotojas surinkęs.
- Misijos įvykdymo kartas, identiškas tokiai pat parametrų atlygio savybei.

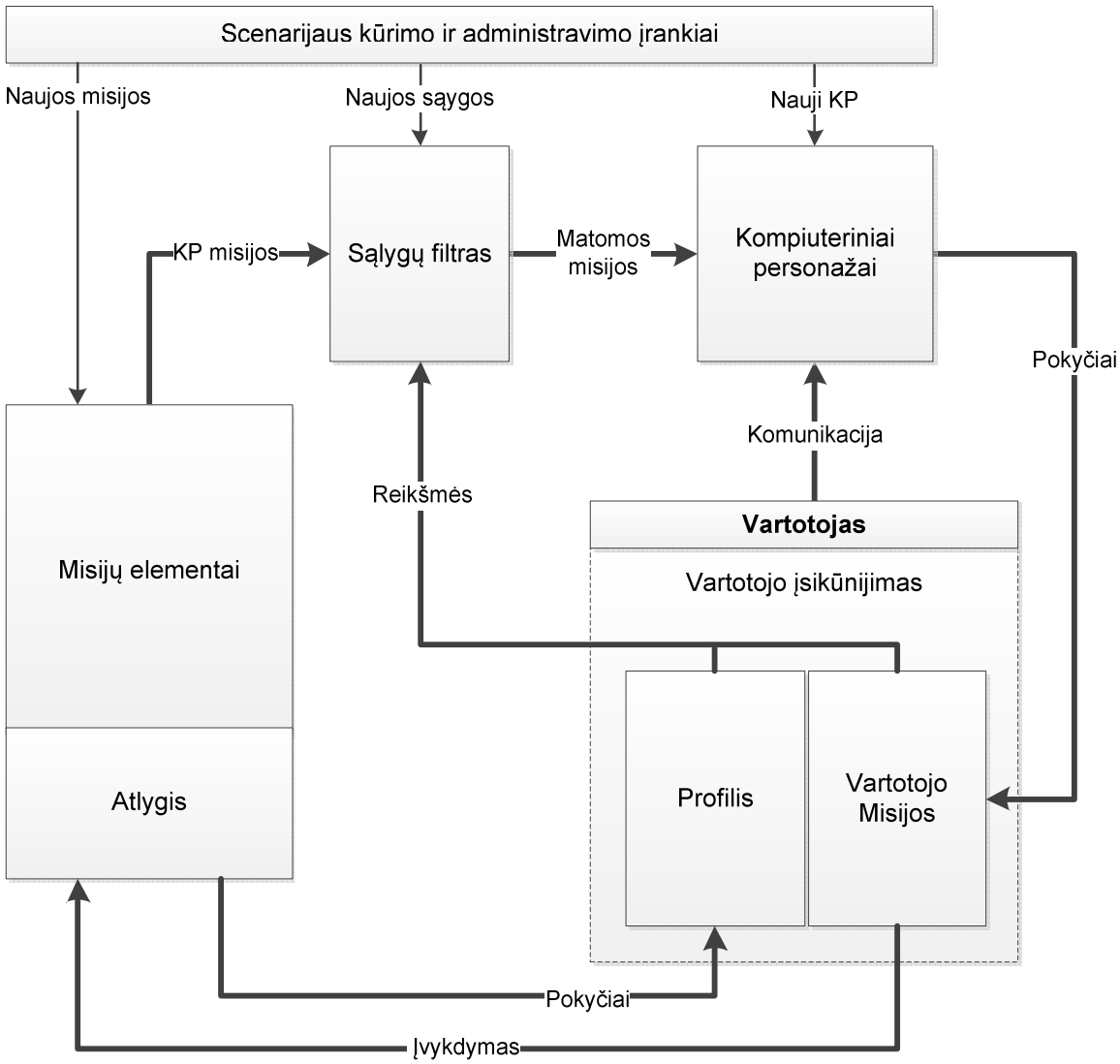
Daiktų atlygio pavyzdys: misijos „studijos universitete“ pabaigoje, vartotojui duodamas 1 aukštojo mokslo diplomas ir -1 (atimamas) studento pažymėjimas.

3.6 Apibendrintas dinamiško scenarijaus variklio modelis

3.1 - 3.5 skyriuose aprašytus principus galima apibendrinti kaip vieną dinamiško scenarijaus variklio modelį, kai minimą scenarijų sudaro tekstiniais dialogais tarp vartotojo ir kompiuterinių personažų paremtos užduotys. Esminius šio modelio komponentus, kurių aprašymai pateikti ankstesnėse šio skyriaus dalyse, bei ryšius tarp tų komponentų, vaizduoja 14 pav.

Modelis paremtas komunikacija tarp vartotojo (vartotojų) bei kompiuterinių personažų, imituojančių visuomenę virtualioje erdvėje. Vartotojo profilis, kurį sudaro vykdytų misijų istorija, parametrai bei virtualūs daiktai, kinta priimant sprendimus, darant pasirinkimus iš galimų alternatyvų. Kitimą užtikrina scenarijaus elementams priskiriamo atlygio reikšmės. Konkrečiu momentu prieinamų misijų sąrašą taip pat įtakoja vartotojo sprendimai, priimti bendraujant su virtualiais kompiuteriniais personažais, nes dauguma misijų sąlygų susiejamos su vartotojo profilio reikšmėmis. Kompiuterinių personažų turimi dialogai yra iš anksto paruošti, tačiau ne visus scenarijaus elementus juose gali pasiekti vartotojas, net jei misiją ir vykdo – dalis scenarijaus elementų liks nematomi, nepasiekiami, kadangi neatitiks užbrėžtų sąlygų, keliančių reikalavimus vartotojo profiliui.

Į modelio schemą įeina ir tiesiogine variklio dalimi nesantys scenarijaus kūrimo ir administravimo įrankiai, kadangi jų egzistavimas galutinėje sistemoje glaudžiai susijęs su paties modelio idėjomis. Šie įrankiai, nepriklausomai nuo jų specifikos, vaizduoja galimybę plėsti, redaguoti adaptyvų ir netiesišką scenarijų, keisti vertinimo kriterijus (atlygį), sukurti naujus kompiuterinius personažus bei kitus virtualaus pasaulio objektus.



14 pav. Esminiai užduočių variklio moduliai ir jų ryšiai

Modelio ypatumai sąlygoja, kad modelio esmę atitinkančio variklio realizacija, kartu su visa kompiuterine sistema, pasižymės tokiomis savybėmis:

- Vartotojo pasirinkimo svarba tolesniam scenarijaus vystymuisi. Kiekvieno skirtingus sprendimus priimančio žmogaus patirtis simuliacijoje bus skirtinga, jei tik sistemoje patalpintos misijos išnaudos scenarijaus netiesiškumo ir adaptyvumo galimybę. Kiekvienas pasirinkimas gali pakeisti, ar vėliau sutikta sąlyga bus tenkinama ar netenkinama.
- Didelė kūrybinė laisvė scenarijaus autoriams. Misijų struktūra turi savo formavimo taisykles, tačiau nei misijos grafo apimtys, nei skirtingų pasirinkimų ar pabaigų kiekiai nėra ribojami, be to, leidžiami ciklai, misijų tarpusavio perpynimas, tad galima teigti, jog

ne scenarijaus autoriai turi taikytis prie misijų struktūros, o misijų struktūra yra pritaikoma pagal tų autorių idėjas.

- Galimybė pildyti scenarijų. Turinio patalpinimas išorinėje duomenų saugykloje (programos kodo atžvilgiu) leis neperkompiliuojant programos ir net jos eksploatacijos periodu kurti ir į sistemą įtraukti naujas misijas, jų elementus, sąlygas ar atlygį. Tokie darbai nereikalaus programavimo žinių ir priemonių, nes yra vedami kaip duomenų struktūros.
- Paprastumas, suprantamumas. Modelis remiasi paprastais matematiniais ir objektiniais principais, tad jo esmė turėtų būti nesunkiai suprantama ir ne informatikos specialistams.
- Efektyvumas. Sąlygų tikrinimas tik įvykus įvykiui, vietoj pastovaus jų tikrinimo, sveikojo tipo klasifikatoriai, minimalus reikalingų duomenų kiekis ir kt., kartu su teisinga modelio realizacija užtikrins spartų veikimą ir taupų resursų naudojimą.
- Lankstumas, plečiamumas. Požiūris į modelio sudedamąsias dalis kaip į savybes turinčius objektus, leidžia patį modelį pildyti, plėsti. Jeigu kuriami produktai to reikalaus galima nesunkiai įvesti:
    - naujo tipo sąlygas. Kiekvienai į klasifikatorių įtrauktai sąlygai reikėtų sukurti programuojamą veikimo logiką (sąlygos prasmę);
    - vartotojo profilio savybes, galinčias kisti atlygio būdu. Pavyzdžiui, vartotojai be parametrų ir daiktų galėtų turėti pasiekimus (angl. *achievements*), kurie būtų „atrakinami“ vykdant užduotis;
    - vartotojo interaktyvumo pakeitimai, pavyzdžiui, vietoj to, kad sąlygų netenkinantys atsakymai būtų slepiami, galima juos rodyti tačiau įvesti tikimybę, kad ta šaka „praeiti“ nepavyks;
    - naujus tekstinių objektų tipus, pavyzdžiui patarimas (angl. *hint*) galėtų būti sudėtingų misijų sprendimo palengvinimas, rodomas reikalaujant žaidėjui. Šis pakeitimas būtų gana sudėtingas, kadangi reiktų apibrėžti ryšius bei peržvelgti daugelį scenarijaus konstravimo taisyklių, tačiau pati modelio ir misijų struktūra nedraudžia naujų tipų atsiradimo.

Visos išvardintos savybės ir apskirtai, sudaryto modelio pritaikymo galimybės išbandomos realiuose inovaciniuose socialinės krypties projektuose. Šis eksperimentas aprašomas 4 skyriuje.

3.6.1 Matematinis modelio apibrėžimas

Tegul bet kuris R={M⊗T⊗Q⊗A⊗E} tipo scenarijaus elementas yra žymimas l^R:

$$l^R = \{P, \Gamma, \dot{\Gamma}, C, W\}.$$

Čia P – elementui priskiriama parametrų aibė, įskaitant ir jo identifikatorių. Ryšiai su tos pačios misijos elementais apibrėžiami identifikatorių rinkiniu Γ= {*tėvoID, vaikoID, misijosID*}, o su visais išorinių misijų elementais, kai tokie egzistuoja, rinkiniu $\dot{\Gamma} = \{\gamma_1, \gamma_2, \dots, \gamma_n\}$. $C = \{C_p, C_d, C_m, C_s\}$– tam elementui priskiriamų parametrų sąlygų C_p, daiktų sąlygų C_d, užduočių sąlygų C_m, ir specialiųjų sąlygų C_s aibės, o $W = \{W_p, W_d\}$ atitinkamai yra už šį elementą skiriamų parametrų ir daiktų atlygių rinkinys. C ir W gali būti tuščios aibės.

Tuomet viena misija (užduotis ar pokalbis) $\widehat{M}_i$, kur i – tos misijos pradžios elemento l_i^M unikalus identifikatorius, yra apibrėžiama taip:

$$\widehat{M}_i = l_i^M \cup \{l_{i1}^E, \dots, l_{in}^E\} \cup \{l_{i1}^T, \dots, l_{ik}^T\} \cup (\{l_{i1}^Q, \dots, l_{ij}^Q\} \vee \{\}) \cup (\{l_{i1}^A, \dots, l_{iu}^A\} \vee \{\}),$$

$$\text{čia } n, k, j, u \in \mathbb{N} \text{ ir } \forall l_i: \exists! \textit{misijosID} \in \Gamma_i, \text{ kur } \Gamma_i \in l_i$$

Kitaip tariant, misiją sudaro viena misijos pradžia *M*, viena arba daugiau pabaigų *E*, viena arba daugiau užduočių *T*, ir galimai tuščios klausimų *Q* ir atsakymų *A* aibės. Elemento l_i priklausomybė misijai nustatoma pagal to elemento ryšį *misijosID*. Visas tekstinis scenarijus S susideda iš apibrėžto misijų kiekio:

$$S = \{\widehat{M}_1, \widehat{M}_2, \dots, \widehat{M}_n\}, n \in \mathbb{N}.$$

Tam tikru diskrečiu laiko momentu *t,* vartotojo z atliktas pasirinkimas ar veiksmas $v_t \in V_z$ atitinka konkretų pasirinktą ar pasiektą elementą l_t. Šio elemento tipas gali būti bet kuris iš penkių – misijos pradžia, pabaiga, užduotis, klausimas arba atsakymas. Simuliacijos pradžioje vartotojo z turimos misijos, t.y scenarijaus projekcija į to žaidėjo užduočių sąrašą, yra S_z^0. Į jas įeina numatytosios pradinės misijos. Kiekvienu tolesniu diskrečiu laiko momentu, atitinkančiu prieitą naują scenarijaus elementą, vartotojo z misijos yra funkcija, priklausanti nuo to scenarijaus elemento savybių ir ankstesniu laiko momentu egzistavusio vartotojo profilio:

$$(S_z^{t+1}, P_z^{t+1}, D_z^{t+1}) = \Phi(l_t, S_z^t, P_z^t, D_z^t), t \in \mathbb{N}$$

P_z^t ir D_z^t atitinkamai žymi vartotojo turimus parametrus ir virtualius daiktus laiko momentu *t*. Kitaip tariant, ši sudėtinga funkcija turi tris kintančias reikšmių aibes, kurių kiekviena priklauso nuo ankstesnės visų aibių būsenos ir dabartinio vartotojo pasirinkimo. Tas vartotojo pasirinkimas nėra bet koks – galima jų aibė *V* taip pat priklauso nuo tuo momentu vartotoją apibrėžiančio

profilio:

$$S \setminus F(S_z^t, P_z^t, D_z^t) \Rightarrow V_z^t$$

T.y. galimus vartotojo pasirinkimus apibrėžia scenarijaus dalis, kurios sąlygos yra tenkinamos. Funkcija F čia žymi tų sąlygų filtro funkciją, kurios rezultatas (sąlygų netenkinančios scenarijaus dalys) yra neprieinamas, tad į vartotojo pasirinkimui įeina tik likusių misijų elementai.

4. SUDARYTO MODELIO PRITAIKYMO SOCIALINIUOSE PROJEKTUOSE TYRIMAS

Teorinis modelio principų sudarymas dar neįrodo jo kokybės, tad modelis buvo išbandytas eksperimentiškai – tik po eksperimento galima daryti išvadas, ar užsibrėžtą darbo tikslą pavyko pasiekti. Taip pat siekta sužinoti, ar numatytos teigiamos galutinio produkto savybės pasitvirtins, ar nepaaiškės nenumatyti modelio trūkumai, neiškils praktinio modelio pritaikymo sunkumų.

Pirmiausia, remiantis modeliu buvo sukurta dinamiškų užduočių variklio realizacija, atsižvelgiant ir į tikrų naujausių socialinės krypties projektų Lietuvoje reikalavimus. Tada ši realizacija panaudota kuriant galutines kompiuterines sistemas tiems (dviems) projektams. Projektus, kurių apimtis pareikalavo nemažo žmogiškųjų resursų kieko, vykdantys specialistai, t.y scenarijų kuriantys ekspertai bei sukurtą turinį talpinantys darbuotojai, buvo supažindinti su variklio (modelio) veikimo principais. Būtent šie žmonės, kartu su jų darbo rezultatais, padėjo vertinant variklio veikimo teisingumą, naudojimo paprastumą ir kitas savybes.

Galutiniai sistemų vartotojai – tikslinių grupių asmenys –, neturintys žinių apie produktų realizacijos ypatumus, modelio tinkamumą ir naudą patvirtinti gali tik netiesiogiai – paprasčiausiai dalyvaudami projektuose, t.y. naudodamiesi simuliacinėmis priemonėmis, taip iš savo pusės lemdami projektų sėkmę, kadangi tuos vartotojus pasiekia ir paveikia sukurto scenarijaus kokybė, tačiau ne priemonės ar būdai, kuriais remiantis tas scenarijus buvo sudarytas. Tačiau akivaizdu, kad sėkmingai vartotojų poreikius tenkinti galima tik turint tinkamai parinktą sprendimą, kurio lankstumas prireikus leistų modifikuoti turinį pagal vartotojų pastabas.

4.1 Dinamiško scenarijaus variklio realizacija

4.1.1 Funkciniai reikalavimai

Dinamiškų užduočių variklio realizacijai kelti šie funkciniai reikalavimai:

- Šakoto scenarijaus palaikymas. Variklis turi suteikti galimybę realizuoti tiek linijinį, tiek šakotą scenarijų, su skirtingomis scenarijaus dalių pabaigomis. Kiekvienoje situacijoje vartotojas gali turėti daugiau nei vieną pasirinkimą.
- Scenarijų įvesti, keisti, panaikinti galima be variklio kodo keitimo.
- Scenarijaus adaptyvumo palaikymas. Vartotojo matomos scenarijaus šakos ir jų elementai priklauso nuo to vartotojo profilio – vykdytų užduočių, turimų parametrų ir virtualių daiktų. Taigi, vartotojo pasirinkimai kuria unikalią to vartotojo patirtį – nulemia kada ir kokie scenarijaus elementai bus prieinami.
- Scenarijaus adaptyvumo kriterijus (pavyzdžiui, reikiamo parametro pavadinimą ir kiekį) keisti galima išoriškai, be žaidimo perdarymo, t.y. kodo keitimo.
- Scenarijaus tekstinė informacija pateikiama vartotojui komunikuojant su virtualiose aplinkose esančiais kompiuteriniais personažais.
- Virtualių aplinkų ir kompiuterinių personažų skaičius bei parametrai taip pat gali būti keičiami. Galima sukurti naujus personažus ir patalpinti juos į norimas aplinkas.
- Vartotojas gali matyti savo progresą – įvykdytas ir vykdomas misijas, turimus parametrus, virtualius daiktus.
- Variklis skirtas internetinei (tinklinei) žaidimo versijai – vienoje aplinkoje gali būti daugiau nei vienas vartotojas.
- Scenarijus apibrėžiamas vedant duomenų struktūras, kurių formavimas remiasi 3 skyriuje aprašytomis taisyklėmis.
- Vartotojui išjungus žaidimą, jo pasiekti rezultatai nepradingsta – duomenys yra išsaugomi ir užkraunami jam kitą kartą prisijungus, kad vartotojas galėtų tęsti žaidimą.

4.1.2 Naudoti įrankiai

Atsižvelgiant į vykdomų socialinių projektų (žr. 4.2 ir 4.3 poskyrius) įtakotus funkcinius ir techninius reikalavimus bei norimą pritaikymo terpę, eksperimentiniam dinamiškų užduočių

modelio pritaikymui pasirinkti šie įrankiai:

- Flash technologija (Adobe Flash CS3 programų kūrimo aplinka, ActionScript 3.0 programavimo kalba);
- SmartFoxServer Pro serverinė programa (versija 1.6.9), kuri skirta Flash technologijos kliento programų komunikacijai;
- OpenSpace 2 izometrinės grafikos variklis, suderintas su SmartFoxServer ir Adobe Flash technologijomis; Jis daugiau susijęs su virtualios aplinkos vaizdavimu, nei su paties variklio veikimu;
- Microsoft SQL Server 2008 ir Microsoft SQL 2008 Management Studio, (SQL kalba);
- Eclipse IDE for Java Developers ir Java programavimo kalba.

Flash technologija pasirinkta dėl turimos patirties šioje srityje ir dėl to, kad ši technologija leidžia nesunkiai realizuoti grafinius reikalavimus socialiniam projektui. Pati dinaminių užduočių apdorojimo ir vaizdavimo logika realizuota būtent šioje platformoje, nors modelyje pasiūlyta metodika iš esmės gali būti realizuota ir su daugeliu kitų įrankių ir technologijomis. SmartFoxServer ir OpenSpace technologijos buvo pristatytos bakalauro baigiamajame darbe ir čia yra naudojamos dėl jų paprastumo, kokybės ir patogumo eksperimentui, t.y realiai mokomajai simuliacinei priemonei pagaminti, kuri pasižymės ir galimybe daugeliui žaidėjų vienu metu vykdyti simuliaciją tinkle, bendrauti tarpusavyje realiu laiku. Kadangi SmartFoxServer plėtiniai rašomi Java kalba, tai šiems darbams pasirinktas nemokamas programų kūrimo įrankis Eclipse. Daugelio žaidėjų simuliacijai įgyvendinti reikalingas ir serveris, šios realizacijos atveju ši sistemos dalis atsakinga ne tik už kelių vartotojų tarpusavio komunikaciją, bet ir už duomenų bazės valdymą – reikiamų duomenų išrinkimą, siuntimą klientui, pakeistų duomenų įrašymą.

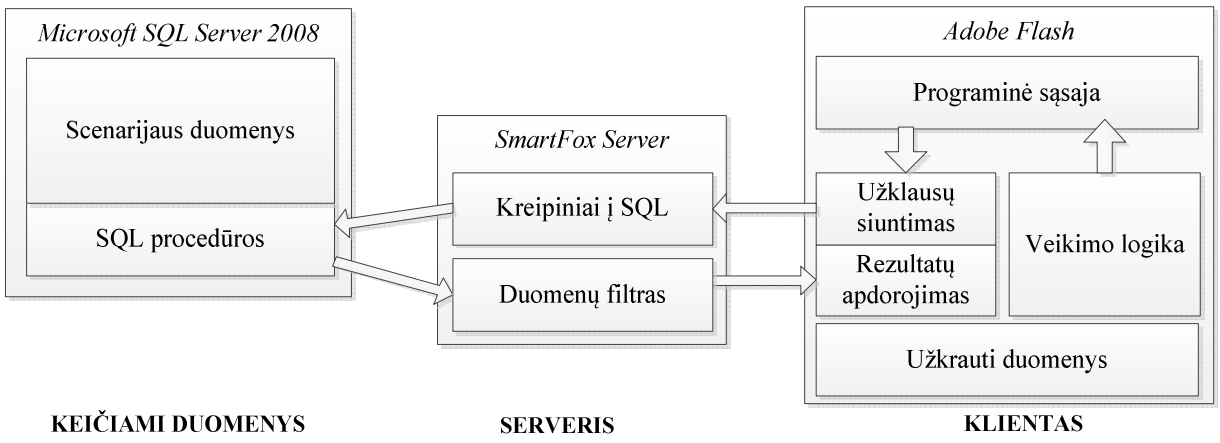
Reliacinė duomenų bazė variklyje naudojama kaip išorinė scenarijaus talpykla, tad jos lentelių turinį galima keisti nepriklausomai nuo ActionScript 3.0 ir Java kodo.

4.1.2 Realizacija

Po maždaug pusę metų trukusių projektavimo ir programavimo darbų, buvo sukurta 3 skyriuje aprašyto modelio realizacija – dinamiškų užduočių variklis, įgyvendinantis visas tam modeliui priskirtas idėjas. Šio variklio apimtis, įskaitant tiktai su užduočių varikliu ir jam reikalingais objektais susijusį kodą (neįskaitant vartotojo sąsajos, duomenų saugos ir pan.):

- 5190 kodo eilučių ActionScript 3.0 kalba. Buvo programuojama remiantis objektinio programavimo pagrindais, tad į nurodytą kodo eilučių skaičių įeina klasių, susijusių su modelio veikimu, eilutės;
- 1934 eilutės Java kalba, atsakingos už duomenų (DB) ir variklio logikos (AS 3.0) komunikaciją, neįskaitant duomenų saugumą (validavimą) užtikrinančio kodo;
- 21 duomenų bazių lentelė ir 39 SQL procedūros (angl. *Stored Procedures*), susijusios su duomenų išrinkimu, įrašymu ar atnaujinimu tose lentelėse.

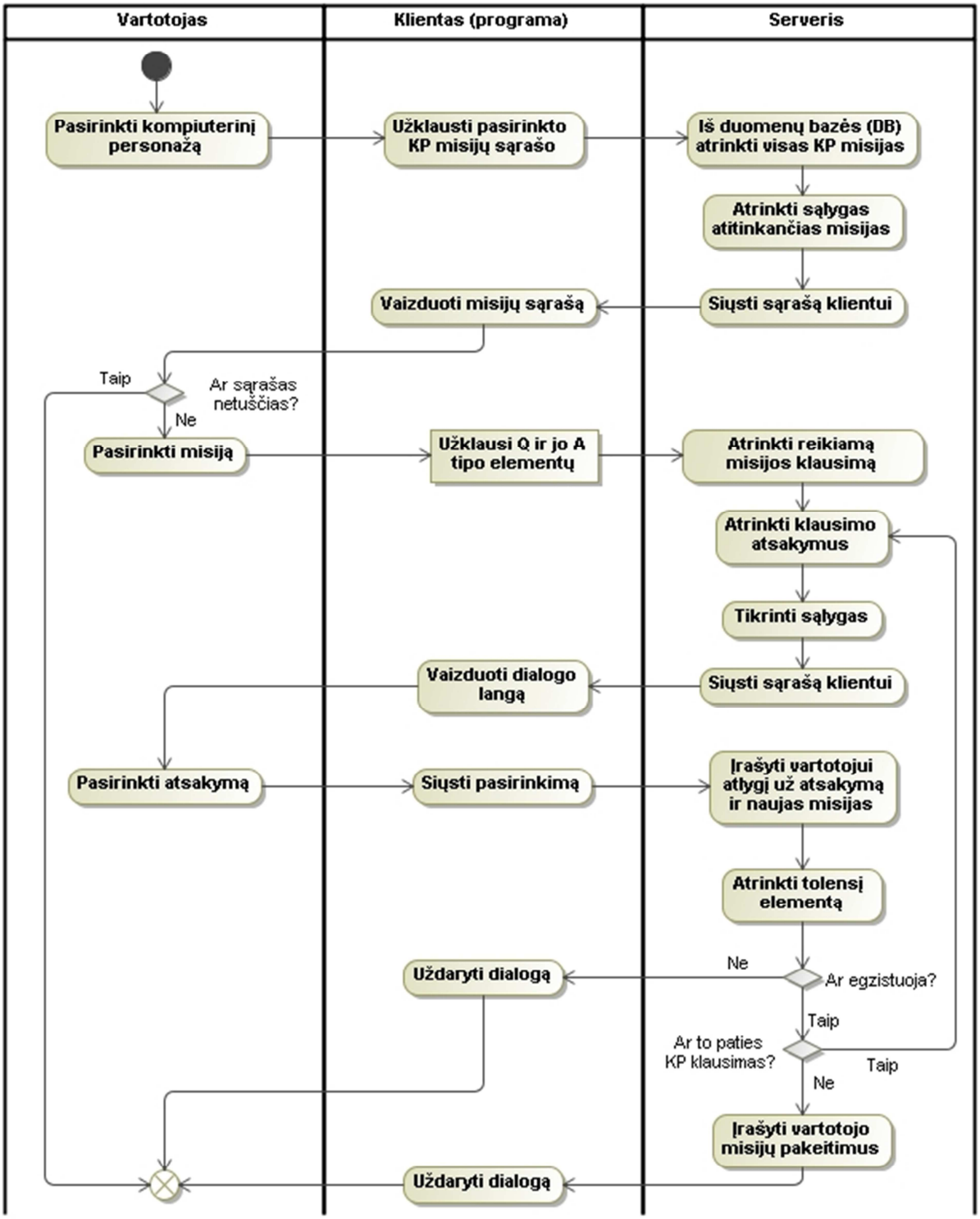
Programos vykdymo metu variklis laukia, kol vartotojas atliks kokį nors veiksmą - tik vartotojo veiksmo metu turi būti iškviečiama jį atitinkanti variklio funkcija. Pavyzdžiui, vartotojui pele paspaudus ties kompiuteriniu personažu, bus kviečiama konkreti funkcija „getNPCTasks“, kuri kreipsis į serverį su reikalavimu pateikti tam personažui priklausančias ir sąlygas atitinkančias misijas (jų pavadinimus); vartotojui pasirinkus atsakymą, kreipiamasi ieškant tolesnio misijos elemento (tekstinio objekto), kadangi jis iš anksto kliento pusėje nėra žinomas ir pan.



15 pav. Modulinė dinamiškų užduočių variklio architektūra

Vidinė variklio architektūra pavaizduota 15 pav. Kiti kliento programos moduliai, pavyzdžiui grafikos variklis, su užduočių varikliu bendrauja per sudarytą programinę sąsają (*API*), t.y. reikia kreiptis į atitinkamus viešus metodus. Tai leidžia pasirinkti bet kokį ActionScript 3.0 kalbos grafikos variklį. Programinės sąsajos metodai pasirūpina kreipiniais į serverį, o serveris – į duomenų bazės procedūras. Duomenų bazėje patalpinti variklio modelį atitinkančios struktūros įrašai, apibrėžiantys scenarijų. Scenarijaus sąlygų filtras patalpintas serveryje, tad klientą pasieks tik jam prieinami, leistini scenarijaus elementai. Šis sukurtas variklis, t.y. programinė realizacija, panaudotas 4.2 ir 4.3 poskyriuose aprašytuose projektuose.

4.1.2 Pokalbio su kompiuteriniu personažu veiklos diagrama



16 pav. Vartotojo dialogo su kompiuteriniu personažu veiklos diagrama

16 pav. pateikta vartotojo komunikacijos su kompiuteriniu personažu veiklos diagrama. Ji vaizduoja vartotojo, kliento programos ir serverio veiklas vieno virtualaus pokalbio metu. Veiklos diagrama sudaryta pagal užduočių variklio realizaciją ir yra svarbi todėl, kad pagrindinis scenarijaus atskleidimo vartotojui būdas kuriamose priemonėse vyksta būtent per vartotojo ir kompiuterinių personažų dialogus.

Į veiklos diagramą neįtraukti pranešimai vartotojui, apie jo misijų pasikeitimus (naujas užduotis, naujas misijas, pasiektas pabaigas). Apie naujus įrašus vartotojas sužino, kai peržiūri savo užduočių žurnalą. Jeigu pokalbis su kompiuteriniu personažu veda į misijos pabaigą, tai po dialogo lango uždarymo seks misijos pabaigos lango atidarymas.

4.2 Projektas „Mano kelias 2“

Kalėjimų departamento prie Lietuvos Respublikos teisingumo ministerijos vykdomas projektas „Mano kelias 2“ – socialinės krypties projektas, kurio pagrindinis tikslas yra „padėti nuteistiesiems integruotis į visuomenę ir darbo rinką, bei palengvinti resocializacijos procesą. Žaidimo pagalba nuteistasis turi gauti žinių bei įgūdžių, padedančių įsidarbinti, pasirinkti profesinę ir darbo veiklą, tobulinti ir lavinti bendravimo įgūdžius su šeima, draugais darbdaviu ir pan.“. Savo pirmtaką „Mano Kelias“ (žr. 2.6 sk.) naujasis žaidimas lenkia dinamiškumu, apimtimi, funkcionalumu, grafine išvaizda.

Esminę projekto dalį sudaro mokomasis simuliacinis žaidimas (toliau - žaidimas), kuriam kelti ir įgyvendinti tokie reikalavimai (pateikiama jų dalis):

- Žaidimas turi būti patrauklus ir interaktyvus.
- Už tinkamą (teisingą, didaktine prasme) elgesį turi būti apdovanojama; Žaidimas turi intelektualiai ir emociškai skatinti nuteistąjį.
- Žaidėjai turi vis kartoti veiksmus, kurių siekiama išmokti.
- Žaidimo variklį formuoti taip, kad būtų nesunkiai įgyvendinama plėtra, naujo turinio įkėlimas, vertinimo bei žaidimo kriterijų keitimas.
- Žaidybinės situacijos turėtų būti dinamiškos, o užduotys ir vertinimas – turėti laipsniškus sudėtingumo lygius.
- Sukurti tinklinę žaidimo versiją, įdiegti grupinio žaidimo ir žaidimo su realiu oponentu funkcijas.
- Suformuoti užduočių, skirtų pilietiškumo, politinio išprusimo, bendruomeninio gyvenimo

įgūdžių ugdymui;

- Nuteistasis privalėtų turėti galimybė pasirinkti ir negatyvias situacijas.
- Integruoti nuteistųjų moterių ir jaunimo poreikius atitinkančias scenas ir aplinkas.
- Į scenarijų įtraukti fizinio aktyvumo žaidimus.
- Žaidėjas pats turėtų formuoti labiausiai jo interesus ir poreikius atitinkančią aplinką, įkelti savo norimus objektus.

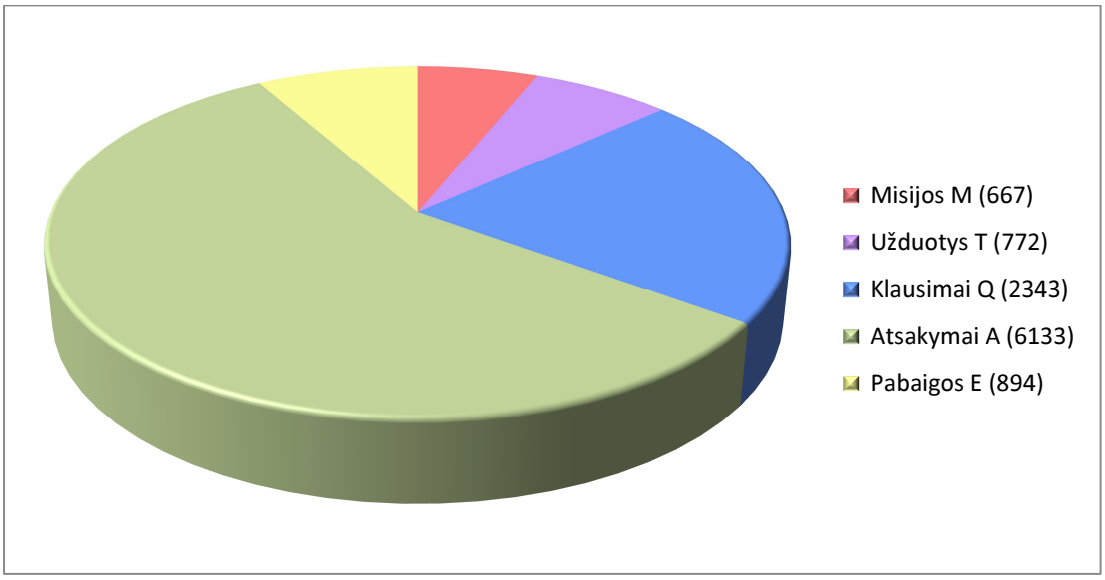
Žaidime nuteistasis gauna bendrą žinių bei įgūdžių kompleksą apie elgesio taisykles su darbdaviu, šeimos nariais, draugais, laiko planavimą. Žaidimas interaktyvus, dinamiškas. Vartotojas spręsdamas užduotis, atsakinėdamas į klausimus, pasirinkdamas aplinkas bei veiklas, lavina darnaus, teisėto, socialaus gyvenimo būdo įgūdžius bei gauna išsamią informaciją apie vieno ar kito pasirinkimo įtaką, teisingumą teisine, socialine, moraline, emocine prasme. Kartu žaidimas išlieka ir pramoginė, laisvalaikio praleidimo priemonė, padedanti resocializacijos darbuotojams turiningiau organizuoti nuteistųjų laisvalaikį, nuteistųjų savęs užimtumo, savitarpio konsultacijų, pagalbos grupes.

Šios mokomosios priemonės turinys formuotas pagal modelyje aprašytas taisykles ir valdomas eksperimento metu sukurto dinamiško užduočių variklio. Scenarijus suskirstytas į teminius lygius, kurių turinys nuteistajam atrakinamas palaipsniui, t.y adaptyviai, atsižvelgiant vykdytas užduotis ir žaidėjo profilį:

1) Susipažinimas su žaidimu (soc. paramos skyrius, pataisos inspekcija, nevyriausybinės org., Caritas ir pan.); Gyvenamosios vietos susiradimas. Tai gali būti šeima, jei žaidėjas tokią turi; gali laikinai apsistoti pas pažįstamus, gali gyventi nakvynės namuose ir pan.;
2) Darbo paieška ir įsitvirtinimas darbe. Minimalių buities sąlygų užsitikrinimas, būtiniausių daiktų įsigijimas. Laisvalaikis: bažnyčia, kazino, baras, sporto klubas ir pan.;
3) Socialiniai santykiai - draugų, pažįstamų susiradimas, bendravimas. Išbandymai: nelegalaus darbo pasiūlymai, kontrabanda, narkotikai, kitokie “lengvi” pinigai, konfliktinės situacijos;
4) Gebėjimų kurti artimesnius santykius ugdymas. Šeimos santykiai (su vyru ar žmona, vaikais), poreikių derinimas, konfliktų sprendimas;
5) Savęs pažinimas, savo profilio nusistatymas; Kvalifikacijos kėlimas, mokymasis;
6) Geresnio, kvalifikuoto darbo paieška. Profilių suderinamumas, realių darbo vietų apžvalga. Galimybė pabūti darbdavio pozicijoje (darbuotojų atranka, priėmimas į darbą ir pan.);

7) Savo būsto įsigijimas. Gerų santykių šeimoje palaikymas. Karjeros perspektyvos (mokymasis, kursai).

Duomenų bazėje šį turinį padengiančių misijų patalpinimui išviso sukurti 10809 įrašai, kiekvienas iš jų atitinka vieną iš tekstinių scenarijaus objektų. Scenarijaus elementų tipų pasiskirstymas pavaizduotas 17 pav. pateiktoje diagramoje, rodančioje, jog daugiau nei pusė visų įrašų yra vartotojų pasirinkimai – klausimai A. Vienam klausimui Q vidutiniškai tenka 2.6 vartotojo pasirinkimų. Misijų pabaigų E ir užduočių T skaičius atitinkamai ~15% ir ~34% didesnis nei pačių misijų, tad scenarijaus nelinijiškumas yra pastebimas – dalis misijų turi daugiau nei vieną skirtingą pabaigą. Sąlyginiai nedidelį užduočių misijose skaičių įtakoja tai, kad didelė dalis scenarijaus yra atskiri pokalbiai su konkrečiu personažu, t.y tai misijos, neturinčios *vykdomumo* parametro, tad po to pokalbio žaidėjas dažnu atveju nėra nukreipiamas tęsti pokalbį su kitu personažu, kas reikalautų naujos užduoties T.



17 pav. Skirtingų misijos elementų kiekiai “Mano Kelias 2” žaidime

Misijų elementų pasiekiamumą įtakojančių vartojo parametrų sąlygų įvesta 272, vartotojo daiktų sąlygų – 8, o misijų vykdymo sąlygų – 1865. Taigi, laikant kad vienam elementui yra priskiriama viena sąlyga, maždaug vienas iš penkių scenarijaus elementų būtų matomas ar pasiekiamas nevisuomet. Tačiau, tokių elementų skaičius yra mažesnis, kadangi dalis elementų turi po kelias sąlygas. Misijų sąlygų tipų klasifikatoriuje iš viso 11 skirtingų sąlygų, kurių įdomiausios yra „arba“ tipo sąlygos – kad tokios sąlygos būtų tenkinamos, užtenka, kad būtų tenkinama nors viena iš tokių pat sąlygų tam elementui. Pavyzdžiui, galima apibrėžti, kad naują pokalbį žaidėjas

matys jei yra įvykdęs bet kurią iš kelių užduočių. Čia išnaudota modelio galimybė plėsti specifinių sąlygų tipus.

Už pasirinkimus simuliacijoje žaidėjas „apdovanojamas“ remiantis variklio modelio atlygio sistema. Vartotojo profilį atitinkančių savarankiškumo, savikontrolės, karjeros, draugiškumo (bei kitų) parametrų reikšmės simuliacijoje gali būti modifikuotos 2771 būdu – būtent tiek parametrų pokyčių numatyta už konkrečius vartotojo veiksmus ir pasiekimus (atsakymo paspaudimą arba misijos pabaigos pasiekimą). Taip pat numatyta 11 virtualių daiktų, kurie žaidėjui duodami už tam tikrų misijų įvykdymą ar atsakymų pasirinkimą.

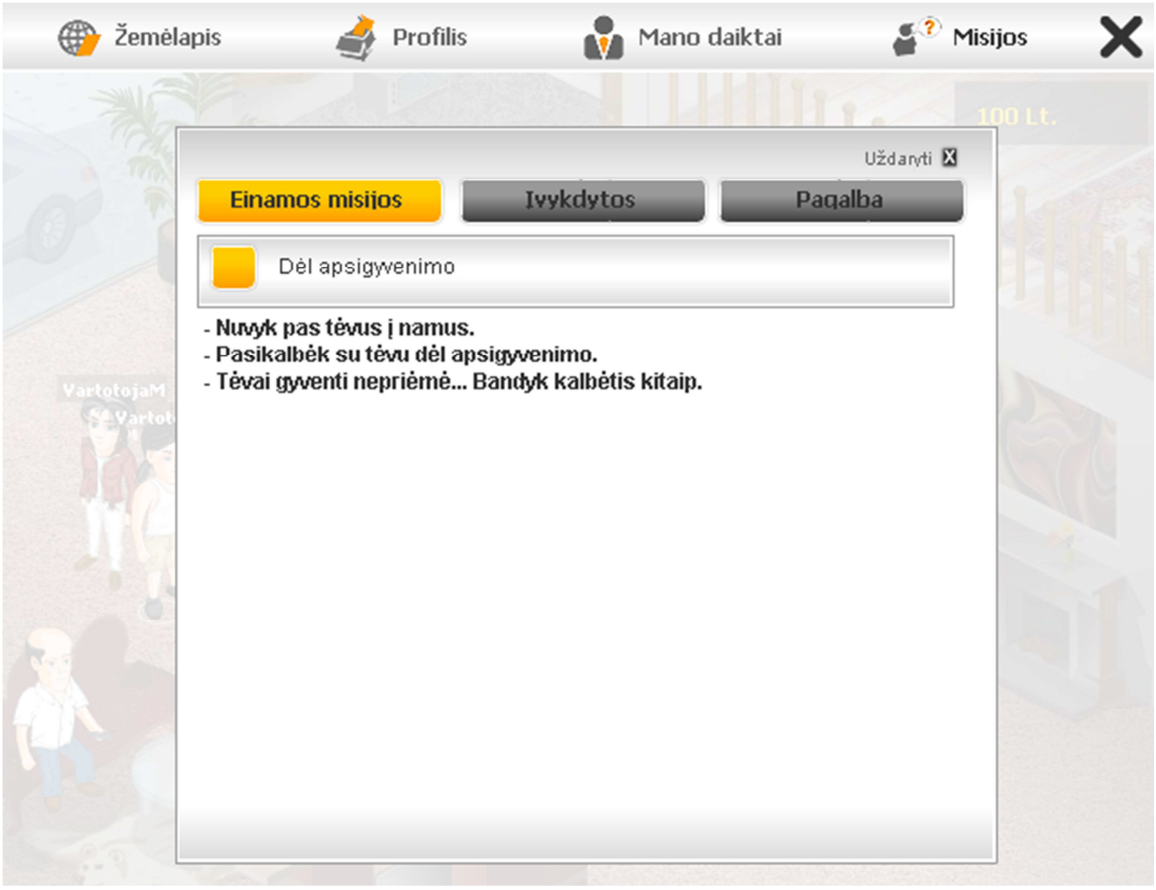
18-20 paveikslėliuose, kurie yra mokomojo žaidimo „Mano Kelias 2“ ekranvaizdžiai ar jų kombinacija, matoma kaip vizualiai užduočių dinamika pasiekia vartotoją. Pirmajame (18 pav.) matomas dialogo tarp kompiuterinio personažo „Juozas“ ir realaus vartotojo įsikūnijimo „VartotojasV“ langas – kairėje pusėje pateikiamas klausimo tekstas, o dešinėje – galimi vartotojo pasirinkimai, kurių šiuo atveju yra 5. Jei žaidėjas pasirenka atsakymą „Pas tėvus“, tolesnė užduotis bus susijusi su pokalbiu tėvų namų aplinkoje (19 pav.).

Kadangi žaidimas tinklinis, tai toje aplinkoje matomi du žaidėjai – VartotojosM matomas vaizdas yra kairėje paveikslėlio pusėje, atitinkamai dešinėje yra VartotojoV matomas vaizdas. Kadangi VartotojasV pasirinko atsakymą „Pas tėvus“, tai pele paspaudęs tėvą vaizduojantį KP, mato galimą pokalbį „Pokalbis su tėvu“. Tuo pat metu šį KP pele spausdama VartotojaM jokio pokalbio nemato, kadangi pasirinko kažkurį kitą atsakymą ankstesniame dialoge, ir šiam pokalbiui sąlygų neatitinka. Jeigu VartotojaM pasirinko, kad gyvena su savo vyru, tuomet šiai vartotojai būtų leidžiama pateikti į daugiabučio aplinką, vaizduojančią jos namus, kurios viduje būtų ir vyrą atitinkantis kompiuterinis personažas. Joks kitas vartotojas į tuos namus įeiti negalės. Jeigu VartotojaM dar ir nurodė turinti vaikų, kartais, grįžus į virtualius namus iššoks su vaikais susieti pokalbiai – tiems pokalbiams nurodytos atsitiktinumo sąlygos, reiškiančios kad egzistuoja konkreti tikimybė, kad tos misijos sąlyga bus tenkinama. Tai tik keli pavyzdžiai iš patalpinto scenarijaus dinamikos ir adaptyvumo.

Užduočių vykdymo rezultatus ir eigą žaidėjai gali pamatyti viršutinėje meniu juostoje pasirinkę „Misijos“. 20 pav. matoma VartotojoV vykdoma misija „Dėl apsigyvenimo“, turinti tris pasiektus (išsaugotus) žingsnius. Vienas žingsnis atitinka vieną T tipo scenarijaus elementą. Naujausias, t.y aktyvus žingsnis rodomas apačioje – jis atsirado po netinkamai vykusio pokalbio su personažu, tad žaidėjui liepiama pokalbį pakartoti ieškant tinkamų atsakymų.

18 pav. Dialogas su kompiuteriniu personažu “Mano Kelias 2” žaidime

19 pav. Skirtingi vartotojai mato skirtingus galimus dialogus



20 pav. Vykdoma misija ir prieiti jos žingsniai - užduotys

Scenarijus gali būti papildytas ar taisomas per paruoštą žaidimo administravimo įrankį, kurioje duomenų formų pavidalu galima rasti ir redaguoti duomenų bazių lentelių duomenis, atitinkančius žaidimo scenarijų. Taigi, scenarijaus vystymas nereikalauja programavimo žinių, nes pats scenarijus saugomas duomenų struktūromis, tad šiuos veiksmus gali atlikti net apmokyti įkalinimo įstaigų darbuotojai. Administravimo įrankis pagamintas naudojant įmonės „SoftArt Solutions“ duomenų bazių administravimo sprendimus.

Svarbu paminėti, kad skirtingai nei 4.1 dalyje aprašytus darbus, visą mokomojo simuliacinio žaidimo realizaciją išplėtojo ne vienas žmogus, o jų komanda, į kurią įeina: 8 scenarijų kuriantys ekspertai, 4 programuotojai (du iš jų Flash), 2 grafikos dizaineriai ir 4 turinio (scenarijaus) talpintojai/testuotojai. Žaidimas šių metų vasario mėnesį įdiegtas trylikos Lietuvos įkalinimo įstaigų vietiniuose tinkluose ir šiuo metu nuteistieji lavina savo įgūdžius naudodamiesi „Mano Kelias 2“ teikiamais privalumais, įskaitant ir dinamiškas užduotis. Įstaigų, kuriose įdiegtas šis žaidimas, sąrašas pateiktas darbo prieduose (1 priedas).

4.3 Projektas „Renkuosi Aš“

Kompleksinių priemonių kūrimas ir taikymas socialinę atskirtį patiriančių asmenų integravimui į darbo rinką - Renkuosi aš!" (toliau - "Renkuosi aš!") – tai inovatyvus Kauno teritorinės darbo biržos vykdomas projektas. Simuliacinė mokymo priemonė (SMP) „Renkuosi aš!“ – kompiuterinė interaktyvi sistema, kurios pagrindinis tikslas yra projekte dalyvaujančių tikslinių grupių asmenų socialinių įgūdžių ugdymas, aktyvių nuostatų savo karjeros planavime formavimas ir įsidarbinimo skatinimas. Projekto tikslinės grupės - darbo rinkoje dalyvaujantys asmenys su fizine negalia, vaikų globos įstaigų auklėtiniai, asmenys, sergantys priklausomybės nuo psichoaktyvių medžiagų ligomis. Sistema parengta remiantis bendra projekto tikslinių grupių situacijos ir poreikio analize.

Sistemą sudaro dvi atskiros, tačiau tiesiogiai susijusios dalys – „Simuliacinis blokas“ ir „Informacinis blokas“. Simuliacinis blokas - vaidmenų-strateginis žaidimas, kuriame dalyviai prisiima tam tikrus vaidmenis ir kuria tolesnį žaidimo scenarijų, sudarant sąlygas pakliūti į tariamai realią gyvenimo situaciją bei saugiai išbandyti skirtingus elgesio modelius.

Savo techniniu pobūdžiu sukurtas kompiuterinis žaidimas yra panašus į “Mano kelias 2”, t.y naudotos tos pačios kūrimo technologijos, tačiau ne tik grafinis priemonės pavidalas, bet ir visas mokomasis tekstinis turinys yra kitas, nepersidengiantis. Svarbu yra tai, kad ir šios priemonės turinio pateikimui buvo panaudotas to paties pasiūlyto modelio idėjos, o pats užduočių variklis (kodas) buvo sėkmingai pakartotinai panaudotas be jokių pakeitimų. Taigi remiantis modeliu tinkamai suprogramuotą užduočių variklį pavyko panaudoti kitame panašaus pobūdžio projekte, kas įrodo variklio lankstumą ir universalumą.

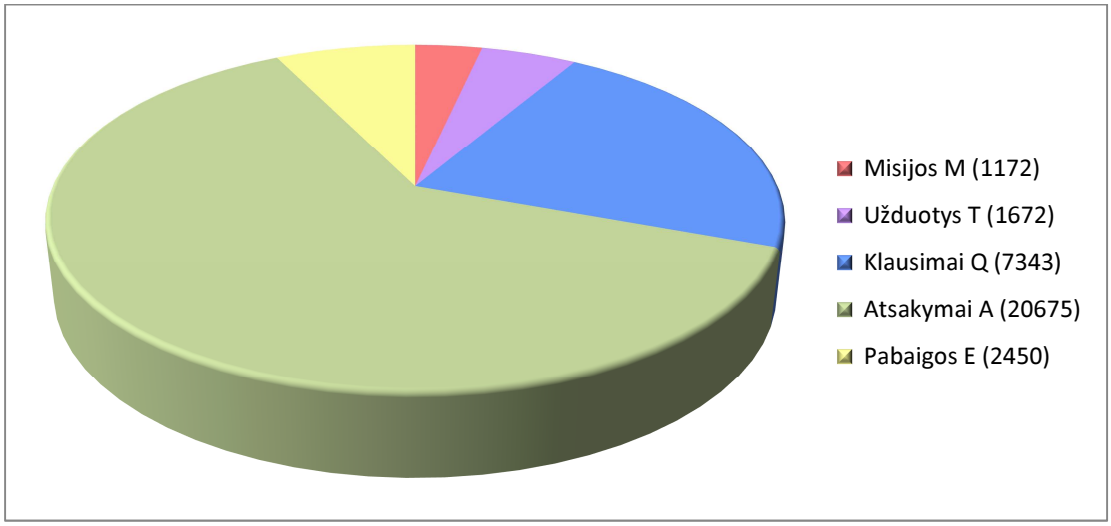
„Renkuosi Aš“ simuliacinės mokomosios priemonės turinys apima apsigyvenimo, darbo, pinigų, laisvalaikio, santykių, mokymosi, kvalifikacijos kėlimo ir kitas temas. Šis scenarijus taip pat formuotas pagal modelyje aprašytas taisykles, kad galėtų būti interpretuojamas ir valdomas minėto dinamiško užduočių variklio ir yra suskirstytas į tokius teminius lygius:

1) Būsto paieškos, maistas, laisvalaikis, draugai;
2) Susirasti darbą pragyvenimui. Užduotis antram lygiui: susirasti darbą ir pasitaupyti pinigų;
3) Įsikurti būste. Užduotis antram lygiui: išlaikyti darbą ir įsikurti būste, numatyti mokymosi galimybes. Pokalbis su šefu dėl papildomo darbo galimybių, dėl viršvalandžių;

4) Draugai, laisvalaikis. Užduotis - pasiekti 1000 taškų už draugiškumą, susirandant draugų. Galimas vairavimo teisių įgijimas;
5) Įgyti kvalifikaciją (mėnesio planas). Apsilankyti profesinio orientavimo tarnyboje;
6) Ieškotis geresnio darbo;
7) Atnaujinti savus namus (banko paskola, remontas).
8) Toliau kelti kvalifikaciją.
9) Gauti kvalifikuotą darbą, kurti savo verslas;
10) Kuruoti keletą pirmo-antro lygio žaidėjų.

Sistemos duomenų bazėje įvesti 33312 misijų elementai (tekstiniai objektai). Scenarijaus elementų tipų pasiskirstymas pavaizduotas 21 pav., kuriame matoma, jog klausimai (A) sudaro dar didesnę tekstinių elementų dalį, nei „Mano Kelias 2“ projekte. Vienam klausimui (Q) vidutiniškai tenka 2.8 vartotojo pasirinkimų. Užduočių (T) skaičius šiame projekte yra ~42% didesnis nei pačių misijų, o kiekvienai misijai vidutiniškai tenka daugiau nei po 2 skirtingas pabaigas (E). Taigi, ir „Renkuosi Aš“ projekto scenarijus formuotas taip, kad vartotojų pasirinkimai ir profilis įtakotų skirtingą patirtį simuliacijoje.

Nurodyta 2418 sąlyga, apibrėžianti misijų elementų matomumą, o didžiausia tų sąlygų dalis – misijų vykdymo sąlygos. Vartotojo parametrams iškeltos 282, turimiems daiktams – 6 sąlygos. Naudojamas tas pats specialiųjų sąlygų klasifikatorius, kaip ir „Mano Kelias 2“ projekte, nes naudojamas tas pats variklio kodas, o kiekviena nauja specifinė sąlyga reikalautų atitinkamų funkcijų sukūrimo variklio realizacijoje.



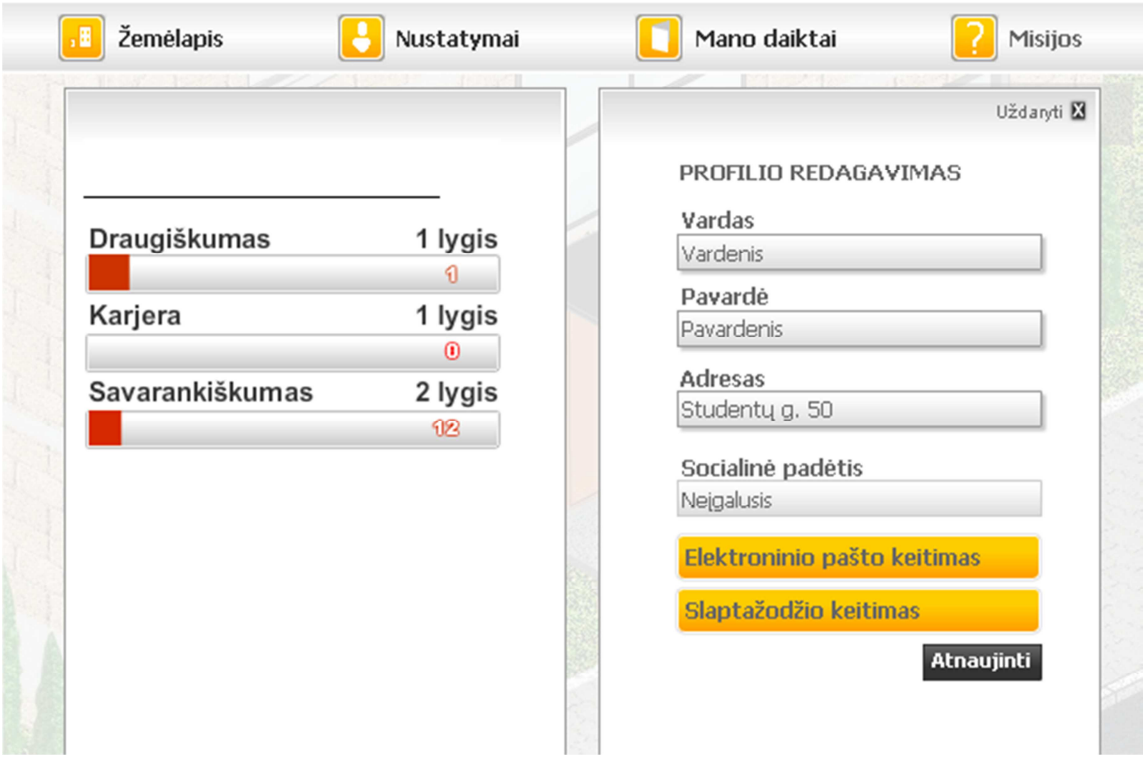
21 pav. Skirtingų misijos elementų kiekiai “Renkuosi Aš” simuliacinėje priemonėje

Maždaug trigubai didesnį misijų elementų kiekį „Renkuosi Aš“ projekte įtakoja ne tik ilgesnė projekto vykdymo trukmė, bet ir tai, kad turinys skirtas net trijų tikslinių grupių asmenims.

Priklausomybė vienai iš tų grupių nustatoma žaidėjo įsikūnijimo kūrimo metu ir sistemoje egzistuoja kaip parametras. Dalis turinio reikalauja konkrečios to parametro reikšmės, tokiu būdu atitinkamos tikslinės grupės asmeniui pateikiamos tik jam sukurtos užduotys bei dialogai. Kiti svarbūs vartotojo parametrai – „draugiškumas“, „karjera“ ir „savarankiškumas“ kaupiami renkantis tam atsakymus, kuriems priskirtas atitinkamo parametro atlygis (23 pav.). Net 12158 atlygio įrašai nurodo galimus parametrų pokyčius šioje SMP, taip pat įvesta 18 daiktų atlygio atvejų.

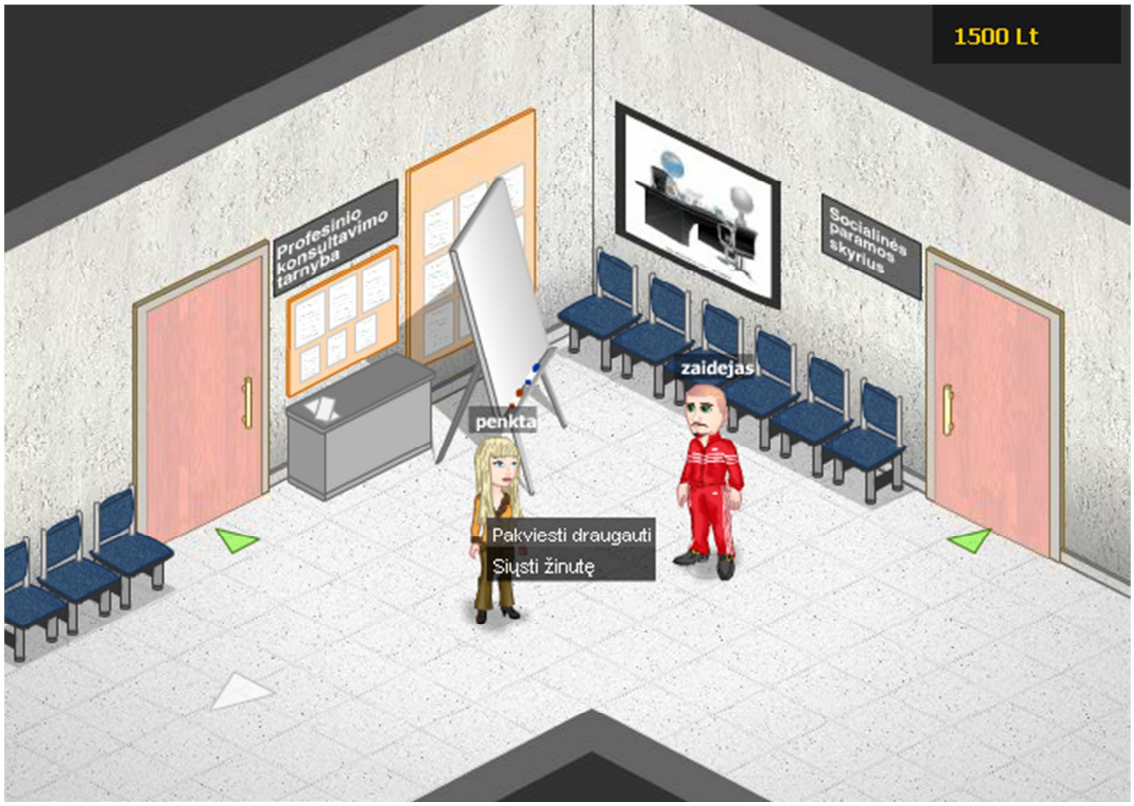
„Mano Kelias 2“ projekto aprašyme pateiktiems ekranvaizdžiams galima surasti iliustracinius atitikmenis ir šioje mokomoje priemonėje. Vietoje to, šioje dalyje ekranvaizdžiais iliustruojamos kitos variklio palaikomo funkcijos. Misijos pabaigos tekstinė informacija simuliacijoje vaizduojama 22 pav. matomu būdu – priėjus misijos pabaigą E, parodoma lentelė, kurioje pateiktas tam scenarijaus elementui priskirtas tekstas. Kaip galima spręsti iš elementų pasiskirstymo diagramos, misijų vykdymo pagrindą čia taip pat sudaro dialogai su kompiuteriniais personažais. 24 pav. pateiktas dialogas su KP yra ypatingas tuo, kad šis pokalbis priskirtas aplinkai – langas iššoka apsilankius aplinkoje, o ne pele paspaudus tam tikrą personažą. Tokiam dialogui taip pat yra priskiriamas personažo ID vien tam, kad grafikos variklis turėtų ką

22 pav. Misijos pabaigos tekstinė informacija SMP „Renkuosi Aš“ pasaulyje



23 pav. Dalies parametrų, kintančių simuliacijos metu, reikšmės yra matomos vartotojui

24 pav. Aplinkai priskirto dialogo pavyzdys (SMP „Renkuosi Aš“)



25 pav. Viena iš projekto „Renkuosi Aš“ aplinkų. Be dialogo langų, su dviem realiais vartotojais

pavaizduoti kairėje, vartotojo pašnekovo, pusėje. Šiuo atveju tai nepersonifikuotas personažas, naudojamas bendriniams pranešimams pateikti.

Simuliacinė mokomoji priemonė „Renkuosi Aš“ yra pilnai baigta ir pasiekiama internetu. Tačiau registruotis ir ja naudotis gali tik tikslinėms grupėms priklausantys asmenys. Per pirmus šešis projekto vykdymo mėnesius simuliacine priemone naudojosi daugiau nei 100 tikslinių grupių asmenų.

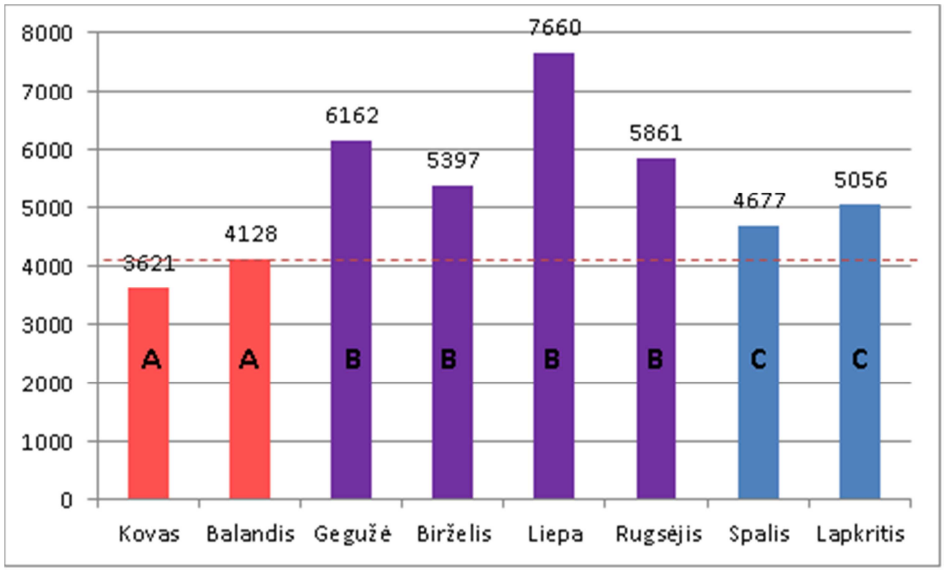
4.4 Modelio efektyvumo įvertinimas

Įgyvendinti socialinės krypties projektai „Mano Kelias 2“ ir „Renkuosi aš“ padėjo pademonstruoti, kad kuriant tokius projektus galima sėkmingai remtis šiame darbe pasiūlytu scenarijaus variklio modeliu. Galutiniai vartotojai tų produktų kokybę galės tinkamai įvertinti tik projektų vykdymui einant į pabaigą, tad magistrinio darbo metu daryta prielaida, jog sklandi simuliacinių priemonių naudojimo pradžia pranašauja teigiamus rezultatus ir ateityje.

Svarbią informaciją apie modelio logikos efektyvumą, pritaikant jį virtualiuose pasauliuose su dinamiškomis užduotimis, teikia to scenarijaus kūrimo ir talpinimo sparta, nes turint universalų variklio kodą, daugiausia laiko resursų reikalauja būtent tos veiklos.

Scenarijų kuriantys sociologijos bei psichologijos ekspertai mokomąjį turinį aprašytiems projektams pradėjo ruošti dar prieš užduočių variklio projektavimą, tad nebuvo laikomasi modelyje siūlyto scenarijaus skaidymo ir sujungimo taisyklių. Kūrybinio proceso rezultatas – įvairiomis formomis aprašytas scenarijus, t.y sudaryti pokalbiai buvo pateikiami ne vienu pavidalu, įskaitant blokines diagramas ,kuriose blokai nesiskyrė nepriklausomai nuo jų patirties, sekų diagramomis, nuosekliu tekstu ar tekstu su žodžiais aprašytais nukreipimais į kitas dokumentų dalis. Turinys buvo suprantamas, tačiau neapibrėžtumo problema egzistavo.

26 pav. pateiktoje diagramoje palygintas užduočių (pokalbių) kūrimo efektyvumas prieš ir po to, kai visi scenarijaus autoriai pradėjo laikytis pagal modelį sudarytos pokalbių vedimo formos – misijų elementų ir jų ryšių surašymo Microsoft Exel skaičiuoklės lentelėse. Pagrindinė forma, kurią pildydavo dialogus kuriantys ekspertai, po supažindinimo su modeliu, iš esmės atitinka 4 skyriuje pateiktos 2 lentelės pavidalą, o sąlygos ir atlygis nurodomi gretimuose langeliuose atitinkamoje eilutėje. Efektyvumas vertintas pagal misijų elementų kiekį, kuriuo galutinėse duomenų bazėse „tapo“ scenaristų paruošti dokumentai per laiko periodą, t.y kokiu greičiu ekspertai pateikė projektuose vėliau išties panaudotą medžiagą.



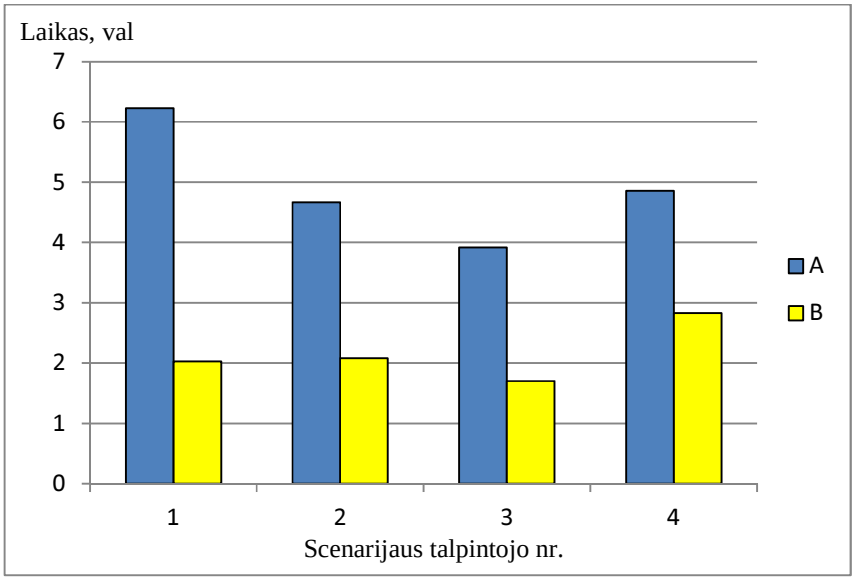
26 pav. Scenarijaus elementų, sukurtų per mėnesį, kiekiai. A – prieš taikant modelį, B – taikant modelį „Renkuosi Aš“ projekte, C – taikant modelį „Mano Kelias 2“ projekte.

Scenarijaus elementų priklausymas tam tikro mėnesio darbui buvo vertintas pagal datą, kada paruoštas scenarijus įkeltas į įmonės failų keitimosi ir versijavimo sistemą. Žinoma, tikslus valandų kiekis, konkretų mėnesį ekspertų praleistas kuriant scenarijų, nebuvo fiksuojamas, tačiau grafike matoma vienareikšmiška tendencija, kad užduočių modelio formos įvedimas ne tik neapsunkino, bet ir akivaizdžiai paspartino ekspertų darbą. Nors, atrodytų, ekspertų darbą turėjo

apsunkinti elementų tarpusavio ryšių identifikatorių surašymas, tačiau tvarkos įvedimas leido greičiau ir intuityviau gaudytis kuriamų užduočių struktūrose, net jei tos užduotys parašytos kito žmogaus. Nei vienas iš ekspertų neturi informatikos krypties aukštojo išsilavinimo, tačiau naudotis Exel skaičiuoklės bazinėmis funkcijomis didelių sunkumų tai nesukėlė. Reikia pastebėti, kad pademonstruotą ekspertų darbo pagreitėjimą po duomenų formos įvedimo iš dalies galėjo lemti ir kiti veiksniai, pavyzdžiui, skaidrėjanti produkto vizija ar įgyti kompiuteriniais įrankiais įgūdžiai, pasikeitusios kitos darbo sąlygos.

Ekspertų pateiktą scenarijų į duomenų bazę talpino 4 darbuotojai. Prieš modelio taikymą sukurtas scenarijus nebuvo perrašomas į Exel lenteles, tad iš tokių dokumentų misijų elementus išrinkti bei įterpti į DB lenteles reikėjo „rankomis“ – po vieną. Vėliau, ekspertams ruošiant misijų struktūrą atitinkančias Exel lenteles buvo vykdomas jų importavimas į duomenų bazę.

Turinio talpinimo greitis buvo patikrintas atskiru eksperimentu, kurio metu visi turinio talpintojai turėjo kuo greičiau suvesti tą patį misijų elementų, jų sąlygų ir atlygių kiekį. Įvedimo laikas buvo fiksuojamas tik tuomet, kai virtualiame pasaulyje suvestas scenarijus veikė teisingai, t.y buvo apdorojamas kaip planuota. Pirmuoju atveju (27pav., a), 4 darbuotojams buvo pateiktas dokumentas, kuris atitiko 286 – 288 misijos elementus, 9-10 sąlygas ir 31 atsakymų atlygį. Elementų ir sąlygų kiekis nežymiai skyrėsi todėl, kad skirtingi dalyviai tą patį nestruktūrizuotą scenarijaus aprašą interpretavo skirtingai (skyrėsi užduočių T kiekis tam tikrai dilemai išspręsti). Antruoju atveju (27 pav., b) tiems patiems darbuotojams buvo pateiktas ekspertų teisingai suformuotas dokumentas su 305 scenarijaus elementais, 5 sąlygomis ir 36 atsakymų atlygiais.



27 pav. Laikas, kurį sugaišo keturi scenarijaus talpintojai, vesdami panašios apimties misijas iš laisvos formos dokumentų (A) ir iš varikliui pritaikytos lentelės (B)

3 ir 4 numeriu pažymėti eksperimento dalyviai turi IT krypties bakalauro laipsnį, o pirmi du yra informatikos krypties bakalauro studijų studentai. Atlikto eksperimento rezultatai (27 pav.) rodo, kad misijų elementus į duomenų bazę importuojant iš su variklio logika suderintos formos, scenarijaus talpinimo laikas pagreitėjo 2-3 kartus ir didžiąją laiko dalį užėmė nebe pačių elementų vedimas, o misijų „surišimas“ sąlygomis, atlygių surašymas. Taigi, modelio logikos pritaikymas, tvarkos nustatymas turėjo teigiamos įtakos projektų vystymui tiek užduočių kūrimo, tiek vedimo atžvilgiu.

Nėra tikslių būdų kokybiškai įvertinti „Mano Kelias 2“ ir „Renkuosi Aš“ scenarijaus realizacijos spartos, lyginant su kitų panašių produktų įgyvendinimo greičiu, nes išnaudoti laiko resursai tik neapibrėžta dalimi priklauso nuo pasirinktos kūrimo strategijos, šiuo atveju – užduočių variklio modelio. Tokį kiekybinį palyginimą būtų galima atlikti tik įgyvendinus tuos pačius projektus, naudojant kitais galimais sprendimais. Šiuo atveju, kai projektų apimtis yra didelė, toks tyrimas būtų labai brangus, kadangi tektų performuoti tūkstančius misijų, suvesti jas į kitokios architektūros sistemą, o tą sistemą dar papildyti reikiamomis funkcijomis. Dėl šios priežasties buvo labai svarbu iš pirmo karto parinkti tinkamą sprendimą. Manyti, kad pasiūlyto dinamiškų užduočių variklio modelio pritaikymas realiuose socialinės krypties projektuose yra efektyvus, leidžia laiku ir pilnai pabaigtos dvi mokomosios priemonės, naudojamos Lietuvoje vykdomuose socialinės krypties projektuose, atsižvelgiant ir į sąlyginai nedidelę komandą ir sąlyginai trumpą tų sistemų vystymo laiką (bendrai apie 14 mėn).

Vykdytų projektų specifika įtakojo kiekybinį palyginimą tik modelio taikymo ir netaikymo aspektu, o 3 skyriuje nagrinėtų sprendimų atžvilgiu atliktas kokybinis modelio palyginimas. Šio palyginimo rezultatai pateikti 5 lentelėje. Lygintas pats modelis, o ne variklis – variklio atveju kriterijaus „pritaikymas įtakotų technologijų pasirinkimą“ reikšmė būtų priešinga, kadangi viena realizacija skirta konkrečioms technologijoms. Eksperimento metu esminių trūkumų, dėl kurių toks modelis neturėtų būti taikomas iškeltai problemai spręsti, nepastebėta. Tiesa, variklio suprogramavimui reikalingi resursai gali būti vertinami kaip trūkumas jau sukurtų variklių atžvilgiu.

Scenarijaus talpintojų nuomone, aprašytų simuliacinių priemonių vystymo metu labiausiai trūko ir darbų spartą būtų pagreitinęs grafinis misijų administravimo įrankis, leidžiantis matyti misijas grafų pavidalu, bei galimybė tokių grafų viršūnes redaguoti. Tačiau ši silpnybė nėra paties modelio trūkumas, kadangi tokio įrankio panaudojimą variklis palaikytų.

5 lentelė. Siūlomo sprendimo ir nagrinėtų egzistuojančių sprendimų palyginimas

	EO+	Adonthell	ScriptEase	Siūlomas sprend.
Tipas	Variklis	Variklis	Strategija ir įrankiai	Modelis
Palaikomos nelinijinės užduotys	Neapibrėžta	Palaiko šakotą scenarijų ir jo santykį su vartotojo profiliu	Palaiko šakotą scenarijų ir jo santykį su vartotojo profiliu	Palaiko šakotą scenarijų ir jo santykį su vartotojo profiliu
Paremtas scenarijaus kalba	Taip, specifinė kalba	Taip, Python kalba	Taip, strategija galėtų būti taikoma įvairioms kalboms.	Ne, paremtas duomenų struktūromis
Galima plėsti scenarijų be kompiliavimo	Taip	Taip	Taip, ypač patogiai, nes toks yra sistemos tikslas	Taip
Sistemos architektūros aprašymas	Pateikta neišsami schema	Detali dokumentacija	Paaiškinta koncepcija	Detalus aprašymas
Pritaikymas įtakotų technologijų pasirinkimą	Taip	Taip	Teoriškai – ne	Ne
Trūkumai	Siauras funkcionalumas; nėra variklio vystymo galimybės	Neišbaigtas projektas; pasenusi grafinė sąsaja, kaip nekintanti variklio dalis	Specializuoti įrankiai, kurių pritaikymas kitokioms architektūroms tik teorinis – darbiniai failai nepateikiami	Reikia resursų jo realizacijai – variklio sukūrimui
Produktai, kuriuose pritaikytas	„Endless online“	„Waste‘s Edge“ demonstracinis žaidimas	„Neverwinter Nights“ žaidimas	„Mano Kelias 2“ ir „Renkuosi Aš“ mokomieji žaidimai
Grafiniai įrankiai scenarijaus sudarymui	Nėra	Yra	Yra	Nėra

5. IŠVADOS

- Netiesiško scenarijaus problemų analizė parodė, kad scenarijaus adaptyvumo ir netiesiškumo įgyvendinimas kompiuteriniuose žaidimuose nėra trivialus uždavinys tiek iš techninės, tiek iš scenarijaus sukūrimo pusės.
- Egzistuojančių dinamiško scenarijaus įgyvendinimo sprendimų analizė parodė, kad vyrauja scenarijaus kalbomis paremtos sistemos.
- Dėl paieškos metu aptiktų variklių panaudojimo trūkumų ir dėl vykdomų socialinių mokomųjų projektų specifikos, problemos sprendimui nebuvo pasirinktas nei vienas iš egzistuojančių sprendimų. Analizės rezultatai pagrindė naujo sprendimo poreikį.
- Sudarytas lankstus dinamiškų užduočių variklio modelis, kuriame scenarijaus programavimo kalba nėra reikalinga, o esminė jo charakteristika – adaptyvių tekstinių užduočių ir dialogų palaikymas, su galimybe tą scenarijų neribotai plėtoti be jokio programavimo.
- Modelis kurtas siekiant jį pritaikyti konkrečiose mokomojo pobūdžio kompiuterinėse socialinėse simuliacijose, tačiau jo konceptų universalumas sudaro prielaidas modelio panaudojimui bet kokio prasminio turinio scenarijams realizuoti ir variklio realizacijai naudojant įvairias programines priemones.
- Remiantis modeliu suprogramuotas dinamiškų užduočių variklis. Realizacija įvykdyta naudojant Adobe Flash, Sun Java ir Microsoft SQL technologijas;
- Tas pats variklis (kodas) sėkmingai panaudotas integracijos į visuomenę ir darbo rinką įgūdžius lavinančiuose “Mano Kelias 2” ir “Renkuosi Aš” mokomuosiuose žaidimuose. Šimtai toms lygiaverčių precedentų Lietuvoje neturinčioms priemonėms priklausančių užduočių ir dialogų pasižymi variklio suteiktomis nelinijiškumo ir adaptyvumo savybėmis;
- Tyrimas, kurio metu buvo fiksuojamas scenarijaus formavimo ir talpinimo sistemose greitis, neatskleidė jokių esminių modelio trūkumų, galinčių paneigti modelio pritaikymo efektyvumą. Scenarijaus kūrimo ir redagavimo procesams palengvinti reikėtų sukurti patogų grafinį užduočių kūrimo įrankį.
- Tolesni darbai galėtų apimti grafinio užduočių kūrimo ir redagavimo įrankio kūrimą, taip pat tolesnį modelio tobulinimą, siekiant didesnio palaikomo lankstumo. Pavyzdžiui, būtų galima suteikti dinamikos ryšiams tarp kompiuterinių personažų ir jiems priskiriamų užduočių; Be to, variklį su galimomis modifikacijomis planuojama panaudoti didesniame mokomųjų bei pramoginių projektų kiekyje.

6. LITERATŪRA

1. Richard Rouse III. *Game Design: Theory & Practice*. Wordware Publishing, Inc, 2001. 125-390p.
2. Luke Bergeron. The Dynamic Conversation Engine [interaktyvus]. 2009 [žiūrėta 2010-05-28]. Prieiga per internetą: < http://www.scribd.com/doc/17682546/Dynamic-Conversation-Engine-Concept>
3. Richard Wages, Non-Linear Scriptwriting. A Graph-Based Authoring Tool for Interactive Scenarios. 2004, gegužė [žiūrėta 2010-10-12]. Prieiga per internetą: < http://www.nomadslab.org/wages/content/wages_sagasnet2004.pdf>
4. Ben McIntosh, Randi Cohn, Lindsay Grace. Nonlinear Narrative in Games: Theory and Practice [interaktyvus]. 2010, rugpjūtis [žiūrėta 2010-10-12]
5. Information Society Technologies. Inscape Project Overview [interaktyvus]. 2008 [žiūrėta 2011-03-24]. Prieiga per internetą: <http://www.inscapers.com/pdfs/INSCAPE.Flyer.A4.project.overview.v1.4.pdf>
6. Stefan Gobel, Luca Salvatore, Robert Konrad. StoryTec: A Digital Storytelling Platform for the Authoring and Experiencing of Interactive and Non-linear Stories [interaktyvus]. International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution. 2008, lapkritis [žiūrėta 2011-03-24]. Prieiga per internetą: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&arnumber=4688056&tag=1>
7. Katherine Howland, Judith Good, Judy Roberson. Script Cards – A Visual Programming Language for Games Authoring by YoungPeople [interaktyvus]. Visual Languages and Human-Centric Computing, 2006 [žiūrėta 2011-04-03]. Prieiga per internetą: <http://www.flipproject.org.uk/wp-content/papers/Script Cards - A Visual Programming Language for Games Authoring by YoungPeople.pdf>
8. Effie Lai-Chong Law, Michael D. Rust-Kickmeier. 80Days: ImmersiveDigital Education Games withAdaptive Storytelling [interaktyvus]. University of Leicester & University of Graz, 2008 [žiūrėta 2011-03-25]. Prieiga per internetą: <http://www.eightydays.eu/uploads/tx_takoscientificpublications/STEG08b.pdf>
9. Serious Games Intercative. About Playinng History [interaktyvus]. 2010 [žiūrėta 2011-05-15]. Prieiga per internetą: <http://www.playinghistory.eu/about-series>

10. John K. Ousterhout. Scripting: Higher Level Programming for the 21st Century. Sun Microsystems Labaratories, 1998.
11. Marcus Trenton, Duane Szafron, Josh Friesen, Curtis Onuczko. Quest Patterns for Story-based Computer Games [interaktyvus]. University of Alberta, Edmonton. 2010 spalis [žiūrėta 2011-05-16]. Prieiga per internetą: <http://webdocs.cs.ualberta.ca/~duane/publications/pdf/2010aiideMT.pdf>
12. Gustavo Lyrio, Roberto Seixas. Using Lua as Script Language in Games Coded in Java [interaktyvus]. IMPA & TECGRAF, Brazil, 2009 [žiūrėta 2011-03-24]. Prieiga per internetą: <http://w3.impa.br/~rbs/pdf/GAMEON-NA07.pdf>
13. Endless Online. EO+ Script engine [interaktyvus]. 2010 [žiūrėta 2011-01-08]. Prieiga per internet: <http://www.endless-online.com/eo+.html>
14. Kai Sterker & The Adonthell Team. Adnothell Engine Architecture [interaktyvus]. 2004 [žiūrėta 2011-05-17]. Prieiga per 76nternet: <http://adonthell.linuxgames.com/files/architecture.pdf>
15. Kai Sterker & The Adonthell Team. Adonthell Documentation [interaktyvus]. 2010, gruodis [žiūrėta 2011-05-17]. Prieiga per 76nternet: <http://adonthell.berlios.de/api/>
16. Kai Sterker & The Adonthell Team. Adonthell Plot Design Guidelines [interaktyvus]. 2004 [žiūrėta 2011-05-17]. Prieiga per 76nternet: <http://adonthell.linuxgames.com/files/plot_guideline.pdf>
17. Kai Sterker & The Adonthell Team. Dlgedit User Manual [interaktyvus]. 2007 [žiūrėta 2011-05-17]. Prieiga per 76nternet: < http://adonthell.linuxgames.com/files/dlgedit.pdf>
18. Maria Cutumisua, Curtis Onuczkoa, Matthew McNaughtona ir kt. ScriptEase: A Generative/Adaptive Programming Paradigm for Game Scripting [interaktyvus]. University of Alberta, Edmonton. 2007 [žiūrėta 2011-05-18]. Prieiga per internetą: <http://webdocs.cs.ualberta.ca/~script/docs/2007SCP.pdf>
19. F. Budinsky, M. A. Finnie, J. M. Vlissides, P. S. Yu, Automatic Code Generation from Design Patterns, IBM Systems Journal, 1996. 151-171 p.
20. Kauno teritorinė darbo birža. Projekto „Nuteistųjų rengimas savarankiškam gyvenimui stiprinant motyvaciją integruotis į visuomenę ir darbo rinką“ ataskaita, 2004.
21. GermRB [interaktyvus]. Ismael Salgado, 2007 [žiūrėta 2010-11-19]. Prieiga per 76nternet: <http://www.gemrb.org>
22. Curtis Onuczko, Duane Szafron, Jonathan Schaeffer, Maria Cutumisu, Jeff Siegel, KevinWaugh, Allan Schumacher. Automatic Story Generation for Computer Role-Playing

Games [interaktyvus]. University of Alberta, Edmonton, Canada. 2006 [žiūrėta 2011-05-19]. Prieiga per internetą: <http://www.aaai.org/Papers/AIIDE/2006/AIIDE06-036.pdf>

23. TheFreeDictionary. Definition of computer simulation [interaktyvus]. Farlex, 2006 [žiūrėta 2010-04-30]. Prieiga per internetą: <http://www.thefreedictionary.com/computer+simulation>
24. Kostas Plukas, Eugenijus Mačikėnas, Birutė Jarašiūnienė, Irena Mikuckienė. Taikomoji diskrečioji matematika. V.: Technologija, 2006. 34-65p.

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

Terminas/santrumpa	Paaiškinimas
ID	Identifikatorius
Įsikūnijimas	Kompiuteriniame žaidime vaizduojamas grafinis veikėjo atitikmuo, kurio veiksmai ir ypatybės priklauso nuo valdančio žaidėjo (angl. *avatar*)
Kompiuterinė simuliacija	Programa, bandanti abstrakčiai atkartoti pasirinkto realaus pasaulio objekto elgseną ar jo savybes
KP	Kompiuterinis personažas (angl. *non-player character*)
Misija	Scenarijaus dalis, apimanti vieno konkretaus tikslo siekimą (angl. *quest*)
Programavimo sąsaja	Taisyklių ir instrukcijų rinkinys, kuriuo remiantis vykdoma skirtingų programų ar jų dalių komunikacija. (angl. *Application Programming Interface; API*)
Programavimo šablonas	Apibendrintas ir pakartotiniam naudojimui tinkamas dažnai pasitaikančių projektavimo problemų sprendimas (angl. *design pattern*)
Scenarijaus kalba	Aukšto lygio programavimo kalba, kuri programos vykdymo metu yra interpretuojama, o jos kodo kompiliuoti nereikia. Tokių kalbų pavyzdžiai – JavaScript, Python, PHP.
SMP	Simuliacinė mokomoji priemonė
Užduotis	Šio darbo kontekste užduotis yra sąvokos „misija“ sinonimas. Pasiūlytame modelyje viena užduotis apibrėžta kaip misijos sudedamoji dalis - žingsnis.
Variklis	Kompiuterinių žaidimų atveju tai sistema, turinti apibrėžtą funkcionalumą bei skirta žaidimo kūrimui ir vystymui

8. PRIEDAI

8.1 Nuorodos į darbe minėtų žaidimų internetinius puslapius

The Longest Journey (FunCom)

http://www.longestjourney.com/

The Elder Scrolls (Bethesda)

http://www.elderscrolls.com/

Sid Meier‘s Civilization

http://www.civilization.com/

Homeworld 2 (Relic Entertainment)

http://www.relic.com/games/homeworld-2/

World of Warcraft (Blizzard)

http://us.blizzard.com/en-us/games/wow/

Dragon Age (Bioware)

http://dragonage.bioware.com

The Neverwinter nights (Bioware)

http://nwn.bioware.com/

Realm Crafter

http://realmcrafter.com/

The Endless Online

http://www.endless-online.com/

Mass Effect (Bioware)

http://masseffect.bioware.com/

Baldur‘s Gate (Bioware)

http://www.bioware.com/games/legacy

8.2 Įstaigų, kuriose įdiegtas „Mano Kelias 2“ simuliacinis žaidimas, sąrašas

Įstaigos pavadinimas	Adresas
Alytaus pataisos namai	Ulonų g. 8a, LT-62505, Alytus
Kauno nepilnamečių tardymo izoliatorius-pataisos namai	Technikos g.12, LT-51334, Kaunas
Kybartų pataisos namai	J.Biliūno g. 14, Kybartai LT-70423, Vilkaviškio r.
Lukiškių tardymo izoliatorius-kalėjimas	Lukiškių skg.6, LT-01108, Vilnius
Marijampolės pataisos namai	Sporto g. 7, LT-68501, Marijampolė
Panevėžio pataisos namai	P.Puzino g. 12, LT-35169, Panevėžys
Pravieniškių 1-ieji pataisos namai	Pravieniškės-2, LT-56551, Kaišiadorių r.
Pravieniškių 2-ieji pataisos namai-atviroji kolonija	Pravieniškės-2, LT-56552, Kaišiadorių r.
Pravieniškių 3-ieji pataisos namai	Pravieniškės-2, LT-56553, Kaišiadorių r.
Pravieniškių gydymo ir pataisos namai	Pravieniškės-2, LT-56554, Kaišiadorių r.
Vilniaus 1-ieji pataisos namai	Sniego g. 2, LT-02109, Vilnius
Vilniaus 2-ieji pataisos namai	Rasų g. 8, LT-11350, Vilnius
Lietuvos Kalėjimų Departamentas	L.Sapiegos g. 1, LT-10312 Vilnius

8.3 Diegimo aktas

AKTAS

2010-05-20

Kaunas

Edgaro Dulskio magistrinio darbo „Dinamiškų užduočių variklis socialinėms kompiuterinėms simuliacijoms“ rezultatai įdiegti plėtojant UAB „Terra IT“ produktus – simuliacinę mokomąją priemonę „Renkuosi Aš!“ ir simuliacinį žaidimą „Mano Kelias 2“.

UAB „Terra IT“ direktorius Egidijus Grigas

A. V. (Parašas) (V.Pavardė)