T A R T U Ü L I K O O L
MATEMAATIKA - INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Tarkvarasüsteemide õppetool
Informaatika eriala

Dmitri Afanasjev

Küsimustiku läbimise süsteem

Magistritöö (40 AP)

Juhendaja: Indrek Sander, MSc

Autor: ..... „.....“ juuli 2010
Juhendaja: ..... „.....“ juuli 2010
Lubada kaitsmisele
Professor: .... „.....“ juuli 2010

TARTU 2010

Sisukord

Sissejuhatus

Tänapäevane tarkvaraarendus (infosüsteemide arendus) on eelkõige suunatud tööprotsesside toetamiseks. Tööprotsessidele suunatus nõuab eelkõige arusaamist tööprotsessidest enestest ja selleks on loodud mehhanismid ja lähenemised, mis kõik arenevad. Hetkel on fundamentaalseks aluseks töövoo (*workflow*) mõiste.

Töövoog on dokumentide ja ülesannete liikumine läbi tööprotsessi. Täpsemalt, töövoog kirjeldab tööprotsessi tegevusepoolseid vaatenurki:

- kuidas tööülesanded on struktureeritud
- kes tööülesandeid teostab
- kuidas tööülesandeid sünkroniseeritakse
- kuidas informatsioon liigub, et tööülesannete täitmist toetada ning nende täidetust kontrollida [1]

Töövoosüsteem on süsteem, mis aitab organisatsioonil täpsustada, täita, kontrollida ja koordineerida töörutiine [2]. Käesoleva töö raames on rõhk tarkvaralistel töövoosüsteemidel. Tarkvaraline lahendus on mõeldud aitama organisatsiooni, millel on organisatoorsel tasemel töörutiinid süstematiseeritud ning sellest tulenevalt on neid võimalik tarkvaraliselt toetada. Tavaliselt koosneb tarkvaraline lahendus töövoomootorist (*workflow engine*), tarbimis-liidesest (*user interface*) ning modelleerimisliidesest (*workflow designer*). Modelleerimis-liidese abil tehakse organisatoorselt reglementeeritud tööprotsess arusaadavaks töövoo mootorile. Tarbimisliides on organisatsiooni igapäevase töö tegemiseks mõeldud töövahend. Klassikalises klient/server süsteemis tarbimisliides (klient) teab, millised tegevused ja mis järjekorras tuleb teostada, või olles piisavalt paindlik jätab ta vastutuse töörutiinide järgimisest kasutajale. Klient/server süsteemis tegeleb tarbimisliides kõigega välja arvatud püsiandmete hoidmine. Lihtsustatult võib öelda, et töövoosüsteemis tegeleb tarbimisliides üksikute ekraanipiltidega (andmete näitamine, uute andmete kogumine) ning teadmine, mida tuleb iga ekraani järel edasi teha (sõltuvalt andmesisestusest või hoopis välistest olekutest), saadakse töövoomootorilt.

Püsiandmeid võivad hoida kõik komponendid, seetõttu ei ole välja toodud andmehoidega tegelevat komponenti (andmebaasi) kui ühte iseseisvat üksust, vaid eeldatakse, et

andmekogud on põhikomponentide alamkomponendid või asuvad “eemal” ja kõigile piisavalt kättesaadavad – organisatsiooni olemasolevad andmed (*legacy databases*).

Tarkvaralise töövoosüsteemi kasutamine eeldab süstematiseeritud tööprotsessi olemasolu ja mahtu. Viimast ei saa üheselt defineerida, kuid see tähendab, et tööprotsess peab korduma piisavalt tihti, et seda oleks majanduslikult mõistlik arvutiseerida. Võrrelda võib näiteks 3 töötajaga firmat, kes saab 2 kaebust aastas, 200 töötajaga firmaga, kes saab 2 kaebust päevas. Viimase jaoks omab mõtet kaebuste register luua ja veelgi enam – omab ka mõtet süstematiseerida kaebuste töötlemise firmasisene protsess. Viimane võimaldab juba firma (või tootmise) kasvu korral lihtsa registripõhise tarkvara asemele luua töövoosüsteemi lahendus, mis annab selles tööosas võimaluse kasutada vähemkvalifitseeritud ja seega odavamat tööjõudu.

Nagu eelmisest lõigust nähtub, tuleb arvutiseerimisel arvestada kasuga. Ja kui on olemas risk, et tööprotsess muutub (ja see on alati olemas), siis tähendab see seda, et tuleb arvestada ka tööprotsesside muutumisest tuleneva lisatööga (st lisakuluga). Kui töövoosüsteem on programmeeritud monoliitsena ja ilma modelleerimisliideseta, siis tähendab tööprotsessi muutumine suuri lisakulusid tarkvara ümberprogrammeerimiseks. Sellest riskist lähtuvalt ongi tarkvaralised töövoosüsteemid tavaliselt üles ehitatud varem mainitud moodulitena. Niimoodi saavutatakse paindlikkus ja ka teatud universaalsus.

Maailma juhtivad tarkvaratootjad on seda mõtet edasi arendanud ja leidnud, et kui investeerida piisavalt palju, siis on võimalik luua väga universaalne töövoosüsteem, mis sobib väga suure hulga asutuste tööprotsesside arvutiseerimiseks. Suured investeeringud universaalsuse jaoks tähendavad aga seda, et sellise valmisprodukti soetuskulud on üsna kõrged ja ei tarvitse olla majanduslikult mõistlikud väiksemate asutuste jaoks. Veelgi teravamalt tekib majandusliku mõistlikkuse küsimus üles juhul, kui arvutiseerimist vajav tööprotsess on valmistoodete kasutusala äärtel või kui ka omab olulisi kitsendusi ja lihtsustusi.

Käesoleva magistritöö eesmärgiks oli leida efektiivne ja odav lahendus valikvastustega küsimustiku arvutiseeritud protsessi jaoks, mille eesmärk on reaalajas optimaalse otsuseni jõudmine piiratud ressursi kasutamise kohta. Töös on kasutatud näitena Häirekeskuse kutsetöötlust, kus dialoogi vormis suhtluse käigus tuleb saada vastus, kas saata välja kiirabibrigaad või mitte saata. Tööprotsess on lahti kirjutatud peatükis 7. Süsteemi visiooniks

võeti Helmer Aaviksoo pool aastat enne projekti algust kaitstud bakalaureusetöö „Kasutaja vastustel põhineva töökulgemise diagrammistiku läbimise kasutajaliides" [3]. Tööprotsess on selline, et tema arvutiseerimiseks sobib töövoosüsteem. Kas selleks on aga parim mõni universaalne valmistoode või on majanduslikult mõistlikum luua uus kitsendatud süsteem, polnud esialgu selge. Seetõttu otsustati 2007. aastal asuda uut süsteemi looma. Subjektiivseteks põhjusteks olid universaaltoodete kasutamise keerukus (sh keelest tulenev keerukus) ja hind. Viimast tuleb lugeda tol hetkel subjektiivseks näitajaks, sest polnud veel teada uue süsteemi loomise hind. Töö käigus, juba analüüsi ajal hakkas tekkima arusaamine, et viimane saab olla piisavalt madal, et olla taskukohane ka Eesti asutustele ja organisatsioonidele. Süsteemi realiseerimiseks pandi kokku tudengite töögrupp, kus magistritöö autoril oli kaks rolli. Projektijuhina oli tema ülesanne töögrupi juhtimine visioonist töötava terviksüsteemini. Selles rollis tuli tal ka juhtida komponentide vahelise suhtlusprotokolli väljatöötamist. Teine roll oli olla programmeerija ning luua algusest lõpuni üks osa süsteemist – modelleerimisliides.

Sellise oma süsteemi puudus on loomulikult sobivus ainult teatud ülesannete klassile, halvemal juhul ainult üksikjuhule. Siiski julgeb autor projekti lõpus väita, et spetsiifiliste korralikult granuleeritud süsteemide loomine on täiesti konkurentsivõimeline organisatsioonide jaoks, kes on oma tööprotsesside süstematiseerimisega alles algusjärgus.

Töös antakse ülevaade töövoosüsteemidest (peatükk 1), nende arvutipõhiste lahenduste komponentidest (peatükk 1.3.1) ja tuuakse võrdluse jaoks näiteid ka valmis tarkvaratoodetest, mis toimivad töövoo terviksüsteemina või mõne komponendina (peatükk 4).

Peatükk 6 kirjeldab modelleerimisliidese loomise tarkvaratehnilisi aspekte. See komponent on graafiline rakendus, mis võimaldab kasutajal konstrueerida spetsiifilise töövoosüsteemi poolt kasutatavaid töövoogusid visuaalselt. Antud lähenemine annab võimaluse palju mugavamalt ja efektiivsemalt luua ning modifitseerida süsteemi poolt kasutatavat andmebaasi, kus hoitakse andmeid töövoode kohta. Liides võimaldab lisada uusi ning modifitseerida andmebaasi salvestatud töövooge.

Töös sisalduvad joonised on loodud programmidega Sybase PowerDesigner, MircoOLAP Database Designer for MySQL 1.9.10, NetBeans IDE 6.8, Adobe Photoshop CS5 ja Gliffy Online Diagramm Designer. Rakenduse realiseerimisel on kasutatud programmeerimis-

keskkonda Eclipse GMF (Graphical Modelling Framework) Edition 3.5 Apache Batik SVG Toolkit 1.7. Andmebaasi halduspaketiks on kasutatud MySQL Workbench 5.2.25. Rakendus kastutab MySQL Connector/J 5.1 andmebaasiga ühenduse loomiseks.

Rakenduse loomisel kasutati järgmisi tehnilisi allikaid: Batik SVG Toolkit dokumentatsioon [11], Java 2 EE käsiraamat [12], Java Design Patterns ressursi [13], informatsiooni MySQL andmebaasiga ühendamiseks [10][14], rakenduse kasutajaliidese ja mootori projekteerimiseks ja programmeerimiseks kasutati mitmeid ressursse [4][5][6][7][8][9][15][39].

1. Töövoosüsteem

1.1 Terminid

Organisatsioon on sotsiaalne mehhanism, mis järgib kollektiivseid eesmärke, kontrollib oma efektiivsust, ja omab piiri, mis eraldab teda välisest keskkonnast. See on grupeeritud inimeste ja teiste ressursside struktuur, mis koostoimega jõuavad ühise eesmärgini [41].

(Äri)Protsess (*business process*) on eristuv, terviklik, ajaline ja loogiline tegevuste kogum organisatsiooni mingi konkreetse eesmärgi saavutamiseks [41].

Tegevus (*activity*) on üks eristatav osa tööprotsessist. Tegevus võib jaguneda samalaadselt kirjeldatavateks alamtegevusteks.

Töövoog on tegevuste järjestatud kogum, mis on ühendatud omavahel jäigalt või tingimuslikult. See kirjeldab tegevuste järjekorda, millest koosneb ühe inimese töö, gruppitöö, töötajate organisatsiooni ettevõttes, masinate tööde organisatsiooni ettevõttes. Töövoog on reaalse töö abstraktsioon või virtuaalne esindaja, mis on konstrueeritud konkreetsest vaatenurgast tööle. Näiteks, toidu kaupuste kettides töövoona saab kirjeldata teed kuidas kala tarnitakse laost läbi veoauto, läbi laaduri, läbi letti, läbi müüjat meie kodulauale [30].

Nagu näha on töövoo ja protsessi definitsioonid üsna sarnased. Konkreetne töövoog on üks võimalus protsessi teostamiseks. Ümbritseva keskkonna mõjul või optimeerimisprotsessi tagajärjel võidakse protsessi teostamise töövoogu muuta – lisades või eemaldades tegevusi, muutes tegevuste järjekorda või tingimusi.

Protsesside haldamine (*Business Process Management*, BPM) asutuses on protsess, millega juhitakse asutuse teisi protsesse. Selle alla kuuluvad nii protsesside eesmärkide, sisendi ja väljundi kirjeldamine, kui ka protsesside realisatsioonide (töövoogude) kirjeldamine ning ka haldusmeetmed protsesside monitooringuks ja arendamiseks.

Töövoogude haldamine (*Workflow Management*, WfM) on protsesside haldamise tehnilise iseloomuga alamosa, mis tegeleb tehnilise koordineerimistega protsessi käivitamisel. See on keerulise BPM strateegia komponent [31].

1.2 CTI näide

Üldiselt peaaegu kõike tööprotsesse ja ülesandeid on võimalik arvutiseerida ning paljudel juhtudel see on majanduslikult mõistlik. Töövoogude arvutiseeritud haldamised leiduvad, näiteks, e-bussi-, e-trammi-, e-trolli-, e-rongipiletite (elektrooniline pilet) ostmisel. Internetipoe süsteemid, jaeladu süsteemid, panga süsteemid (internetipank), logistika ettevõte süsteemid, süsteemid dokumentide haldamiseks on majanduslikult kasulikud arvutilahendused. Siiski, valdkondi, kus on rahaliselt mõistlik töövoo haldamissüsteeme kasutada, on rohkem.

Süsteeme, mis aitavad hallata töövooge, kohtab ka kaasaegsetes kõnekeskustes. Need kasutavad CTI (*Computer Telephony Integration*) tehnoloogiat selleks, et automaatselt leida lahendust triviaalsete ja tihtiküsitavate küsimuste jaoks, mis võivad vähendada inim-dispetšeri töö efektiivsust. Lihtsustatud võimalik töövoog on selline: klient helistab kõnekeskusesse ja teda algusest ühendatakse CTI süsteemiga, mis küsib küsimusi ja pakub eeldefineerituid nummerdatud vastusi, mille põhjal helistaja liigub läbi eeldefineeritud töövoo. Lõppude lõppuks, kas helistaja probleem sai lahendatuks CTI poolt pakutud variandiga või kui see on keeruline või ebastandardne probleem, siis klient ühendatakse klienditeenindajaga. Sel juhul kogutakse vajalik informatsioon probleemi kohta ja proovitakse helistajat aidata individuaalselt. Seega CTI tehnoloogia kasutab töövoo tehnoloogiat küsimuste organiseerimiseks ja helistaja juhtimiseks.

Mõnedel juhtudel on vaja suheldes klienditeenindajaga edastada informatsiooni, näiteks, isikukoodi või aadressi. Sellel momendil tihti tekkivad raskused, sest mõnikord on päris keeruline suuliselt korrektselt edastada mingit informatsiooni. Antud juhul informatsiooni, mida tahetakse edastada nimetatakse kontekstiks. Konteksti edastamiseks saab kasutada abivahendid, mis ei ole otseselt kõnekeskuse tarkvaraga seotud: e-mail, kiirsõnumite rakendused (*Microsoft Messenger, ICQ, IRC, Skype*). Antud edastamisviis ei ole eriti mugav,

kuna tuleb mitu rakendust korraga käima panna. Intelligentne *smart-call-center* rakendus lahendab selle probleemi ühendades kokku mehhanismid ja vahendid, mis aitavad kiiresti konteksti, sh ekraanipilti, saata teenindajale kasutades muuhulgas kiirsõnumite ja VoIP funktsionaalsust. Antud lähenemine parandab kõnekeskuse kvantitatiivsed (rohkem kliente teenindatakse ajaühikus) ja kvalitatiivsed parameetrid (teenindaja, konteksti vahetamise abil, saab paremini ja kiiremini probleemist aru, seega lahendatakse probleemi kiiremini) [22].

Samal ajal täiesti automaatsed kõnekeskuse vastamissüsteemid ei ole efektiivsed ja tavaliselt väheneb klientide rahuolu tase navigeerimisloogika ebamugavuse ja mitteolulise teksti kuulamisele kuluva ajakulu tõttu. Selle probleemi lahendus on kombineeritud algatuse (*mixed-initiative*) süsteemid. Neid süsteeme kasutatakse tavaliselt laialt tehnilise toe kõnekeskustes. Süsteem, mis aitab teenindajal lahendada probleemi kliendilt küsimuste küsimisega ja võimalikke vastusvariante pakkudes. Küsimustiku läbimisel probleemi lahendus on käes. Küsimustikud on genereeritud vastava valdkonna spetsialistide abil teadmusbaasi andmete põhjal, sest tavalised teenindajad tihtipeale ei ole kompetentsed probleemi lahendama oma kogemuse abil. Teenindajate põhieesmärk on kiiresti ja õigesti leida probleemile lahendus ning sellega tõsta kliendi rahuolu. Kõik küsimustikud moodustavad süsteemi teadmusbaasi [26]. Selline segatud algatusega variant aitab klienditeenindajal filtreerida ainult need probleemid, mis nõuavad personaalset lahendamist ja automatiseerib lahenduste leidmist triviaalsete, teadaolevate ja tihtiküsitavate probleemide ja küsimuste jaoks.

Ansaplus on kõnekeskuse haldamissüsteem, pakub unikaalse kõnedele vastamissüsteemi, mis näitab helistaja kohta kogu vajaliku informatsiooni teenindajale, annab võimalust kiiresti saata sõnumi SMS, e-mail ja faksi kaudu. Pakutakse vahendeid kõnevoogude haldamiseks ja CRM (*Customer Relationship Maagement*) funktsionaalsust. Viimane aitab organiseerida, automatiseerida ja sünkroniseerida äriprotsessi, ehk. tegevusi, mis on seotud müügiga, turundusega, tehnilise ja klienditoega [40].

Samuti, CTI tehnoloogiat kasutavad nn. *smart-call-center* süsteemid, mis mängivad tähtsa rolli *online* mängude valdkonnas, kasutades automaatse kõne navigatsiooni, informatsiooni ühendamist ühes kohas ja dispetšeriga suhtlemist selleks, et pakkuda klientidele interaktiivseid teenuseid [23].

1.3 Töövoo haldamissüsteemid

Töövoo haldamissüsteem (THS) on arvutipõhine süsteem, mis haldab ja defineerib ülesannete kogumi organisatsiooni piires lõpp-produkti tootmiseks. THS annab võimaluse defineerida erinevaid töövooge erinevat tüüpi tööde või protsesside jaoks. Näiteks, tootmises, disaini dokument peab olema automaatselt saadetud disaineri käest tehnilisele direktorile ja tootmisinsenerile. Igas töövoo etapis on üks töötaja või töötajate grupp vastutav ülesande lahendamise eest. Siis, kui ülesanne on lahendatud, tagab töövoo tarkvara, et töötajad, kes on seotud järgmise ülesandega, on teavitatud ja nendele on saadetud kõik vajalikud materjalid ja andmed selleks, et oma osa tööprotsessis teostada. THS samuti automatiseerib koondatuid ülesandeid ning tagab, et lõpetamata ülesanded on kontrolli all. THS võib kontrollida automatiseeritud protsesse ja lisaks pakub tihti dokumentidevahetust elektroonsel kujul, mis omakorda väldib paberkuju dokumentide liikumist. Näiteks, kui eelmist tegevuse käigus valmistatud dokument on *MS Word* kujul aga järgmise tegevuse sisend nõuab dokumendi *Adobe PDF* kujul, siis automaatne protsess konverteerib antud dokumenti ja alles siis teavitab järgmise töövoo etapiga seotuid isikuid. Seda nimetatakse sõltuvuste kontseptsiooniks. THS peegeldab sõltuvusi, mis on vajalikud iga ülesande edukaks lahendamiseks[30].

Seega, töövoo haldamissüsteem on selline süsteem, mis pakub äriprotsesside menetluslikku automatiseerimist. Ta haldab töövoo tegevuste jada ja kasutab vastavaid ressursse, mis on vajalikud töövoo sammu lõpetamiseks.

Erinevad protsessid omavad erinevat eluiga, mis on mõõdetav minutitega, päevadega ja mõnikord kuudega. See sõltub protsessi keerukusest ja protsessi sammudega seotud tegevuste kestvusest.

Kõik töövoo haldamissüsteemid pakuvad tuge kolmes funktsionaalses alas:

1. Loomise ajal: süsteem omab funktsionaalsust modelleerimaks töövoo protsessi ja sellega seotuid tegevusi, määramaks iga tegevuse eest vastutavaid isikuid

2. Käivitamise ajal: süsteem omab mehhanisme selleks, et juhtida töövooprotsessi eeldefineeritud loogika järgi, pannes järjekorda tegevusi, mis oli paika pandud töövoo modelleerimise ajal
3. Käivitamise ajal: süsteem omab mehhanisme väliskeskkonnaga (inimene, välisrakendus) interaktiivseks suhtlemiseks, andmete vahetamiseks või valikute tegemiseks

Töövoo haldamissüsteemide kasutamispiirid on väga laiad: alustades kalendri rakendustest, mis kasutavad e-maili automaatsete meeldetuletuste saatmiseks kuni süsteemideni, mis juhtivad kogu ettevõtte äriprotsesse.

Töövoo haldamissüsteemid tulid turule 1990. aastate alguses. Nad aitasid arvutispetsialistel automatiseerida kogu äriprotsessi, aga mitte toetada või automatiseerida konkreetset ülesannet. Piltide töötlemine, tõlked, büroode ja kontorite süsteemid andsid hoogu THS süsteemide arendamiseks. Need süsteemid olid järk-järgult integreeritud kontoritöö süsteemidega (MS Office jt) ja kasutasid e-maili tarkvara kommunikatsiooni jaoks, saates teadmisi ja dokumente inimestele ja teistele programmidele.

Ajalooliselt on eristatavad neli THS süsteemide põlvkonda, alates programmeerija poolt kodeeritud süsteemidest, kuni universaalsete ja täielikult konfigureeritavate süsteemideni:

1. Esimene põlvkond on probleemispetsiifiline, kus töövoogude definitsioonid on sisseprogrammeeritud ja rakenduse arhitektuur on suletud. Muudatused süsteemis on võimalikud ainult programmikoodi muutmisel.
2. Teise põlvkonna süsteemides on töövoo osa eraldatud rakenduse osast, seega töövoo haldamine on eraldiseisev moodul. Töövoogude definitsioonid on ühendatavad süsteemiga skriptkeele abil. Antud põlvkonna süsteemid saavad olla sisseehitatud piiratud hulka rakendustesse.
3. Kolmanda põlvkonna THS rakenduse arhitektuur on ehitatud osaliselt kasutades avatud standardeid ja on ühendatav kõikide teiste töövoogude rakendustega. Antud põlvkonna töövoogude definitsioonid on võimalik luua kasutades graafilist liidest, mis on suur edasiminek. Rakenduse liidesed ja formaadid on tihti patenteeritud ja tootja-spetsiifilised.
4. Neljanda põlvkonna rakendused on täiesti integreeritavad teenustega nagu e-mail, töölaua haldamine, kaustade haldamine ja teised. Liidesed ja formaadid on

standardiseeritud. Töövoo mehhanismid ja keeruline arhitektuur on kasutaja eest peidetud.

Aegade jooksul on tarkvareturule tulnud erinevaid töövoo haldamissüsteeme, mis on mõeldud erinevate ülesannete lahendamiseks. Need süsteemid erinevad omavahel avatuse, standardite toe, võimsuse, lihtsuse ja spetsiifilisuse poolest. On võimalik eristada kolme tüüpi töövoosüsteemide tehnoloogilist lahendust. Need on eristatavad töövoo loomise paindlikkuse ning töövoomootori avatuse poolest:

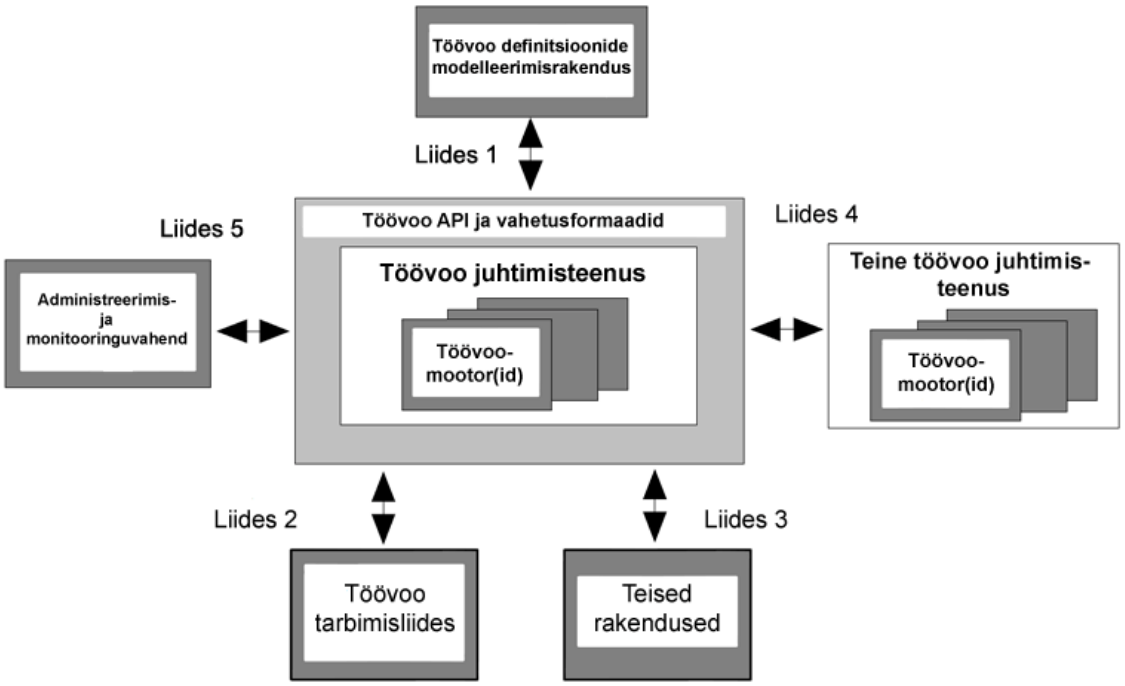
1. Püsiprogrammeeritud (*hard-coded*) süsteemid – programmeerija poolt töövoomootorisse sissehitatud töövoo loogika ja definitsioonid. Kinnine rakendus, muudatused tehakse ainult programmi koodi muutmise abil.
2. Paindlikud süsteemid – süsteemid, mis annavad võimalust lõppkasutajale vabalt luua, modelleerida, käivitada ja hallata töövoo protsesse. Universaalsed süsteemis, mis ei ole spetsialiseeritud ja ei oma eeldefineeritud tegevusi. Kasutajal on võimalik luua oma tegevuste hulka ja nende abil koostada vajalikke töövoo diagramme. Tavaliselt, neid süsteeme pakkuvad graafilist toimetajat töövoo definitsioonide loomiseks ja muutmiseks.
3. Teised süsteemid – süsteemid, mis omava juba sisseprogrammeeritud lailevinud standardseid töövoo tegevusi, skeeme ja malle. Spetsialiseeritud süsteemid, konkreetsete ülesannete lahendamise jaoks.

Iga THS-i tüüp omab väärtusi ja puudusi, millega tuleb arvestada, siis kui valitakse rakendust, mille abil hakatakse töövoo looma, käivitama ja haldama. Järgmistes peatükkides on kirjeldatud THS süsteemide komponentides ja nende rollides sellistes süsteemides.

1.3.1 THS komponendid

Kuna tööprotsesside erinevus on päris suur, eksisteerivad erinevad töövoo haldamissüsteemid. Vaatamata sellele omavad kõik süsteemid sarnaseid omadusi, mis on aluseks nende programmide integratsiooni ja koostalitusvõimsuse arendamiseks erinevate produktide vahel.

WfMC (Workflow Managemen Coalition) arendas välja etalonmudeli [28, 30], mis kirjeldab ühtlustatud töövoosüsteemide baasmudeli:



Joonis 1: WfMC etaloonmudel

Joonis 1 illustreerib töövoo etalonmudeli põhikomponente ja liideseid. WAPI (*Workflow API and Interchange formats*) on liideste kogum, mille abil kõik töövoosüsteemi komponendid on kättesaadavad ja mille abil reguleeritakse koostoime töövoo juhtimiskeskusega (töövoomootor(id)) ja ülejäänud süsteemi komponentidega. Antud kihis on realiseeritud toetus kõikide töövoo juhtimiskomponendi liideste jaoks (Liidesed 1...5). Liidesed ühendavad töövoo juhtimiskomponendi ülejäänud süsteemi komponentidega:

- *Töövoo definitsioonide modelleerimisrakendus* – rakendus töövoode defineerimiseks.
- *Administreerimis- ja monitooringuvahend* – rakendus administreerimiseks ja monitooringuks.
- *Töövoo tarbimisliides* – klient-rakendused, mille abil kliendid (rakendused, kasutajad) suhtlevad töövoomootoriga.
- *Teised rakendused* – rakendused, mida võidakse kasutada töövoo käivitamise ajal. Selle liidese abil kutsutakse välja selliseid rakendusi nagu näiteks dokumendi haldusrakendus, piltide töötlemissüsteem, ettevõtte juhtimissüsteem ja teised.

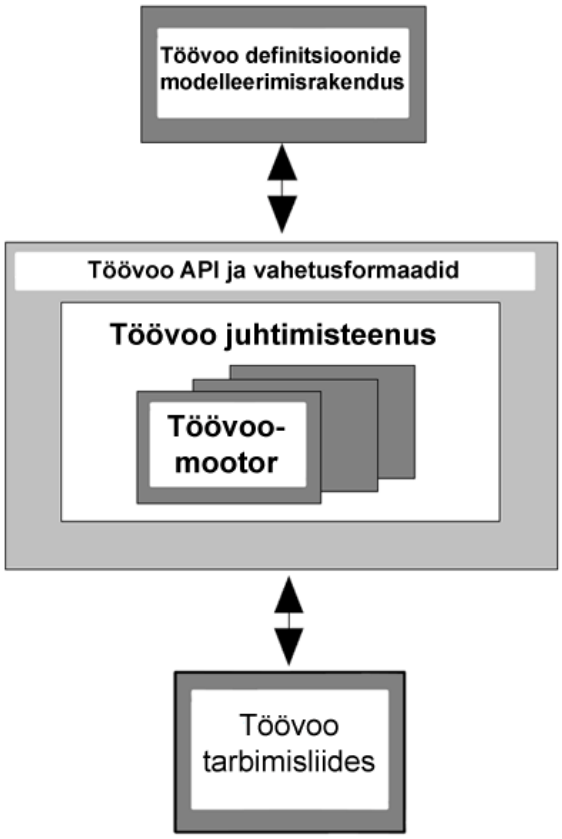
- *Teine töövoo juhtimisteenus* – teiste THS juhtimiskomponendid. Antud liidese abil ühendatakse sõltumatu töövoosüsteemid ettevõtte sees või erinevate ettevõtte vahel. Näiteks, autotehase automaatne detailide tellimise süsteem on ühendatud varuosade tehases oleva töövoosüsteemiga selleks, et vajaduse tekkimisel automaatselt tellida puuduvaid varuosi.

Seoses sellega, et töövoode automatiseerimistehnoloogia on muutunud keeruliseks, muutub töövoosüsteemide liideste standardiseerimine väga tähtsaks faktoriks. Rakenduse edasiarenguks on hädavajalik, et ta on ehitatud kasutades avatud arhitektuuri selleks, et rakendust saaks ühendada olemasoleva ja areneva keskkonnaga ning vajadusel ühendada teiste tootjate rakendustega. See annab paindliku võimaluse ehitada organisatsioonile sobiva kooskõlaliselt töötavate tarkvarasüsteemide komplekti organisatsiooni eesmärkide täitmiseks.

Joonisel 1 toodud etalonmudel on koormav ja keeruline paljude ülesannete jaoks. Näiteks, liides teiste töövoosüsteemidega ühendamiseks ja välisrakenduste suhtlemisliides ei ole alati vajalikud. Saab konstrueerida lihtsustatud etalonmudeli, mis koosneb järgmistest komponentidest:

- Töövoomootor
- Tarbimisliides
- Modelleerimisrakendus ehk. modelleerimisliides

Lihtsustatud etalonmudeli skeem on toodud joonisel 2.



Joonis 2: Lihtsustatud etalonmudel

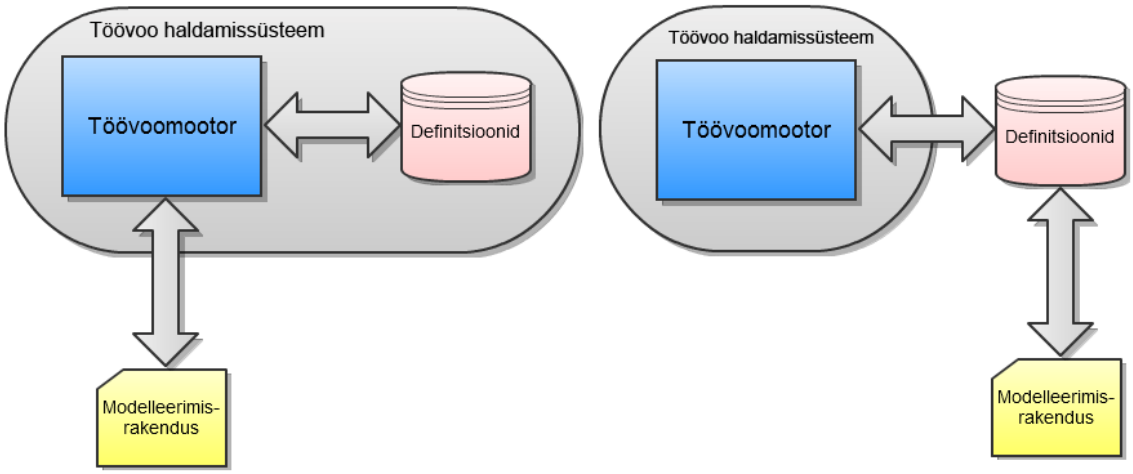
Sellel kujul on etalonmudel sarnane Model-View-Controller (*MVC*) arhitektuuri põhimõttel loodud süsteemiga. Töövoo definitsioonid ehk *Model* komponent paikneb töövoo juhtimisteenuse komponendi sees. Töövoo modelleerimisliides on töövoo definitsioonide ehk *Model* komponendi haldusrakendus. Töövoomootor on *Controller* komponendi rollis ja töövoo tarbimisliides on *View* komponendi rollis.

Järgmiseks on kirjeldatud ülalpool loetletud lihtsustatud etalonmudeli komponendid.

1.3.2 Töövoo modelleerimisrakendus

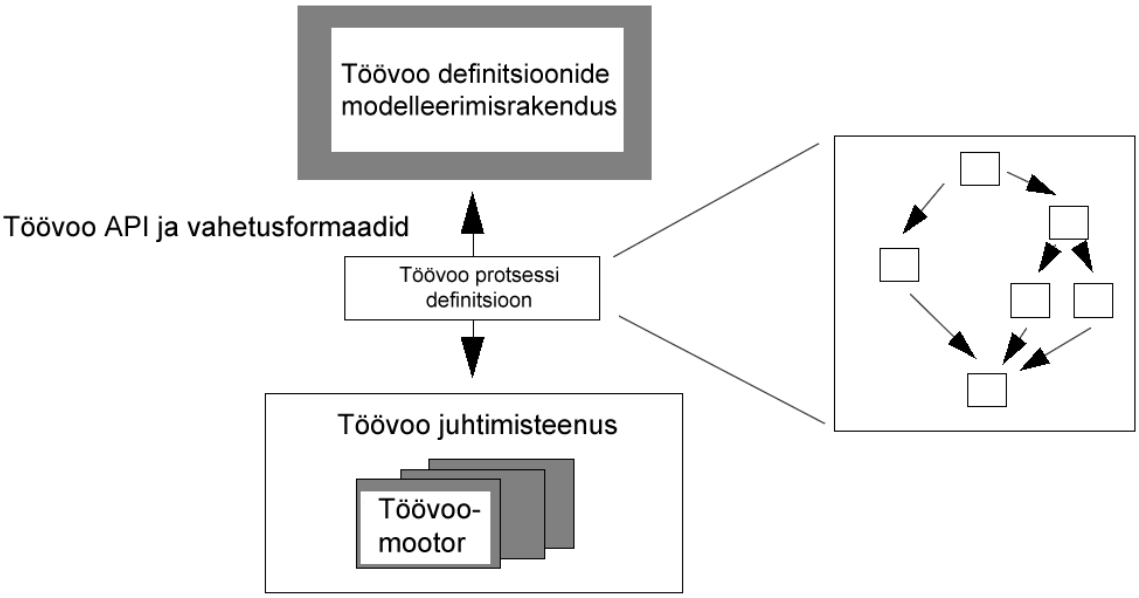
On olemas palju erinevaid rakendusi, mida võib kasutada, et analüüsida, modelleerida, kirjeldada ja dokumenteerida äriprotsesse ja töövoogusid. See võib olla kõige triviaalsem „pliiats ja paber“ või väga komplitseeritud graafiline toimetaja. Antud rakendus võib olla eraldiseisev või olla töövoo haldamissüsteem osa. Kui kasutatakse eraldiseisvat rakendust, siis

tavaliselt hoitakse töövoo definitsioone andmebaasis, mis asub töövoo juhtimisteenusest väljaspool. Seda selleks, et modelleerimisliides ei segaks töövoomootori tööd. Kui aga definitsioonide haldamisrakendus on töövoosüsteemi osa, siis paikneb andmebaas töövoosüsteemi piiri sees (joonis 3).



Joonis 3: Erinevad definitsioonide andmebaasi asukohad

Modelleerimisrakenduse väljundiks on töövoo definitsioon, mida töövoomootor loeb sisse ja käivitab omas keskkonnas. Liides modelleerimisrakenduse ja töövoomootori vahel (vt. joonis 1) peab olema paindlik ja võimeline kooskõlaliselt transportima/salvestama definitsioone andmebaasi. Antud liidest nimetatakse „Definitsioonide import/eksport liides“. Viimase THS põlvkonda süsteemides on see liides realiseeritud kasutades avalikke standardiseerituid tehnoloogiaid ja API’sid. Seega töövoosüsteem on ühildatav teiste modelleerimisliidestega, mis toetavad samu tehnoloogiaid.



Joonis 4: Töövoo definitsiooni vahetus

Joonisel 4 on illustreeritud töövoo definitsiooni vahetamismehhanism modelleerimis-rakenduse ja töövoo juhtimiskomponendi vahel. Töövoo defineerimisel kasutatav standardiseeritud formaat loob piiri defineerimis- ja käivitamiskeskkonna vahele, see annab lõppkasutajale võimaluse valida erinevate modelleerimisrakenduste ja töövoo käivitamissüsteemide vahel. Lisaks annab töövoo definitsioonide standardiseeritud formaat võimaluse käivitada antud töövoogu erinevates töövoosüsteemides.

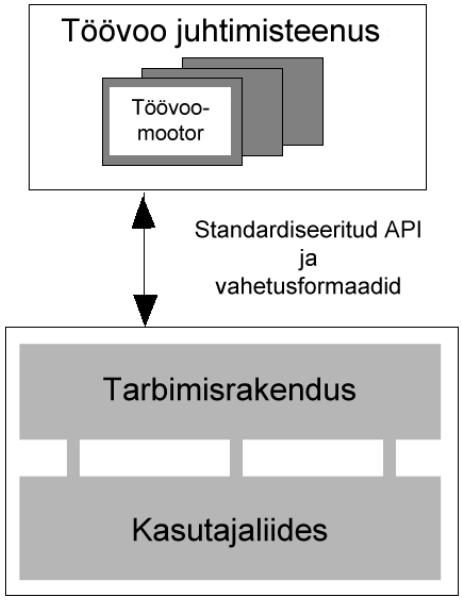
Antud tüüpi komponendi realiseerimisel peab silmas pidama, et rakendus peab täielikult oskama salvestada (*serialiseerida*) koostatud definitsiooni visuaalset esitust (diagrammi) standardiseeritud formaati ja sellest formaadist tagasi.

1.3.3 Tarbimisliides

Töövoo tarbimisrakendus ehk tarbimisliides on programm, mida töövoomootor kasutab kasutajaga suhtlemiseks, siis kui ta nõuab viimase tähelepanu. Näiteks, valiku tegemiseks: kas saata kliendile sõnum SMS või e-maili teel. Antud rakendus võib olla kas töövoosüsteemi paketi sees või olla esitatud eraldiseisva rakendusena. Lisaks, antud rakendus võib olla integreeritud mõne kontorisüsteemiga nagu e-maili klient, *office* rakendused, selleks, et lõppkasutajal oleks kogu tööülesannete teostamiseks vajalik ühest kohast kättesaadav. Väliste

rakenduste käivitamine töövoo mootori poolt võib olla kas otsene ja automaatne või lõpp-kasutaja nõusolekul.

Standardiseeritud liides töövoomootori ja tarbimisliidese vahel peab olema väga paindlik ja peab võimaldama ilma piiranguteta edastada andmeid nende komponentide vahel (vt. Joonis 5).



Joonis 5: Tarbimisliides ja töövoomootor

Mõnedel süsteemidel eksisteerib mitu erinevat tarbimisrakendust, mida võib kasutada erinevates kohtades ja erinevates situatsioonides. Erinevus võib olla sisuline, aga võib olla ka tehnoloogiline. Tehnoloogiliselt on võimalikud järgmised lahendused:

- Õhuke klient – veebilehitsejal põhinev klient, töötab lehitseja kontekstis, võib-olla ka ka *applet* vormis (nt. Java) või interaktiivne keskkond (nt. *Adobe Flash*, *MS Silverlight*). Antud tüüpi rakendust on mugav kasutada siis, kui lõppkasutaja ei istu oma arvuti taga, kus olemas *paks klient*.

  Plussid: platvormisõltumatus, lihtsus, ei nõua (spetsiifilist) paigaldamist

  Miinused: suhteliselt ebamugav, suhteliselt aeglane ja kitsendatud võimalustega
- Paks klient – iseseisev rakendus, mis töötab konkreetse operatsioonisüsteemi peal. Välja töötatud lokaalseks kasutamiseks. Seda tüüpi rakendust kasutatakse siis, kui lõppkasutajal on juurdepääs arvutile, kuhu rakendus on paigaldatud. Antud tüüpi klient võib olla ühendatud arvutitöökeskkonnaga. See annab võimaluse kasutada

klienti integreerituna e-mail kliendiga, *office* tarkvaraga ja teiste igapäevaste rakendustega

Plussid: kiire, mugav, integreerimisvõimalus

Miinused: platvormisõltuvus, keeruline, nõuab paigaldamist

Ülalpool toodud tarbimisrakenduste tüüpidel võivad eksisteerida edasiarendused ja versioonid erinevate platvormide jaoks, näiteks *Symbian OS*'i jaoks, *IPhone OS*'i jaoks, *Andriod OS*, *MIDP* versioon ja teised.

Tarbimisliidese realiseerimisel tuleb silmas pidada, et rakenduse kasutajaliides peab olema lihtne, arusaadav ja selge. Lisaks peab rakendus oskama interpreteerida kõiki tegevusi ja näidata kasutajale vastavaid andmeid, mida talle saadab töövoo mootor. Seega peab rakendus toetama töövoosüsteemis kasutatavat protokolli.

1.3.4 Töövoomootor

Kõige tähtsam töövoosüsteemi komponent on töövoomootor. See on **tarkvaraline teenus**, mis pakub käivitamiskeskkonda töövoogude jaoks ning ta on vastutav selle keskkonna juhtimismehhanismide eest. Tavaliselt pakub töövoomootor vahendeid selleks, et:

- Interpreteerida töövoo definitsiooni – lugemis-, parsimis- ja veaotsimismehhanismid aitavad laadida definitsioonide hoidlast töövoo definitsiooni ja konstrueerida sellest virtuaalsete objektide struktuuri edasiseks töötlemiseks ja käivitamiseks.
- Hallata töövooge – vahendid loomiseks, aktiveerimiseks, peatamiseks, hävitamiseks jne.
- Identifitseerida töövoo elemente, mis vajavad lõppkasutaja tähelepanu, ning liideseid, mis toetavad kasutajaga koostoimet. Vastutab töövoo ja kliendi rakenduse vahelise suhtlemise eest.
- Hallata töövoo kontrollimisandmeid ja teisi andmeid, mis on seotud konkreetse töövooga. Töövoo andmete saatmine ja vastuvõtmine kasutajale/kasutajalt ja välisrakendustele/välisrakendustelt.

- Luua liidest välisrakenduste käivitamiseks ja töövoo protsessiga seotud andmete vahetamiseks. Töövoo mootor võib käivitada e-mail kliendi ja anda talle käsu e-maili saatmiseks.
- Tagada kogu süsteemi turvalisus – administreerimis- ja kontrollimismehhanismid, andmete revisjon.

Töövoomootor kontrollib kõikide tegevuste ja alamtegevuste käivitamisi, mis on määratud töövoo definitsioonis. Töövoo juhtimiskomponent võib koosneda mitte ainult ühest töövoomootorist. Suurte keeruliste süsteemide puhul on töövoo juhtimine jagatud mitmete töövoomootorite vahel, mis on tihedalt omavahel seotud ning igaüks vastutab konkreetsete ülesannete eest: kas iga töövoomootor vastutab ühe konkreetse või mitme temaga seotud protsessi eest või on ülesannete jagamine töövoomootorite vahel realiseeritud tehnilise funktsionaalsuse alusel: üks töövoomootor tegeleb lõppkasutaja suhtlemisega, teine – ainult ressursside hõivamisega, kolmas – protsesside loomisega, hävitamisega ja peatamisega ja nii edasi.

Töövoo juhtimiskomponent pakub käivitamiskeskkonda töövoogude jaoks. Mehhanismid, mis on realiseeritud kahe ja rohkema töövoomootori suhtlemiseks ja andmete vahetamiseks ei ole standardiseeritud ja on tootespetsiifilised, kuna töövoo juhtimismehhanism (isegi mitme töövoomootoriline) on üks ja terve loogiline komponent, mille tööd ei ole võimalik tükkideks lõigata.

Seoses maailma industriaalse arenguga ja arvutusvõimsuste suurenemisega on turule tulnud suured kommerts töövoosüsteemid. Iga süsteem kasutab eripärast tehnikat töövoogude haldamiseks. Näiteks, *Intalio|BPMS* töövoosüsteemi põhiidee on pisut erinev *IBM WebSphere* tööpõhimõttest. Erinevad funktsioonid, erinevad sihtgrupid ja sihtülesanded annavad hoogu sellele, et töövoosüsteeme saab klassifitseerida ja jagada süsteemide töövoomootori võimsuse ja avatuse järgi.

Töövoomootori võimsus, see on mootori võimalus töötada erinevate töövoogude tüüpidega, käivitada paralleelselt mitu töövooprotsessi ning toetada erinevaid töövoo defineerimisvõimalusi. Mida võimsam mootor on, seda keerulisemad on töövoomootori sisemised mehhanismid.

Mootori avatus on vabadus definitsioonide loomisel. Töövoomootori paindlikus on otseselt seotud töövoomootori avatusega. Mida paindlikum on töövoomootor, seda ta on avatum, seega töövoode definitsioonide loomisel on rohkem vabadust ja vähem kitsendusi. Samas, mida rohkem vabadust töövoode defineerimisel pakub süsteem, seda vähem mugavust tavakasutajale. See on tingitud sellega, et vabadus defineerimisel, spetsialiseeritavus ja kasutajasõbralikus ei saa koos elada. Mida rohkem vabadust, seda vähem spetsialiseeritavust ja seda ebamugavam tavakasutajale.

	Plussid	**Miinused**
API - .NET klassid, Java klassid	• Suur vabadus • Universaalsus	• Programmeerimisoskus on vajalik • Süsteemi kohandamine
Raamistik/valmis toode	• Eeldefineeritud tegevused	• Defineerimisvabadus ainult toode raamistiku piires • Ei sobi spetsiifiliste ülesannete jaoks
Spetsiifiline kitsendatud süsteem defineerimisvõimalusega	• Mugav ja kasutajasõbralik defineerimisliides	• Ainult spetsiifiliste ülesannete klassi jaoks
Püsiprogrammeeritud süsteem	• Kiire ja lihtne	• Väga spetsiifiline • Töövoog on eeldefineeritud • Puudub lihtne võimalus töövoode uuendamiseks

Tabel 1. Töövoo defineerimistehnoloogiad

Tabel 1 toob välja erinevad tehnoloogiad töövoo defineerimiseks ning nende miinused ja plussid. Kõige rohkem vabadust töövoode defineerimiseks annab API kasutamine, näiteks, *.NET* või *Java* klassid, mille abil programmeerijal on võimlaus luua suvaline töövoo objekt, kirjeldada selle iseloom ja võimalused ning loodud objektide abil luua töövoo definitsiooni ja käivitada töövoomootori abil. Antud juhul on töövoomootor väga võimas, kuna ta oskab

käivitada neid töövoo protsesse, mis on loodud objektide abil, mis omakorda on programmeeritud kasutades etteantud klasse ehk API-t. Seega on see lähenemine väga mugav spetsiifilise töövoo loomiseks. Negatiivseks momendiks on nõue programmeerimisoskuste omandamine etteantud API kasutamiseks.

Järgmine võimalus on kasutada olemasolevat toodet või raamistikku töövoogude defineerimiseks. Kui kasutusele on võetud valmistoode, siis lõppkasutajal on töövoogude defineerimisvabadus piiratud toote eeldefineeritud funktsioonidega. Näidiseks on ERP süsteemid, kus on juba kogum eeldefineeritud standardseid töövooge, mida võetakse kasutusele ettevõtte tööprotsesside haldamiseks. Kui kasutusele on võetud raamistik, näiteks WWF (*Windows Workflow Foundation*), siis kasutajal on võimalik defineerida töövooge kasutades eeldefineerituid tegevusi või kirjutada uusi kasutades pakutud raamistiku. See annab programmeerimisvabadust antud raamistiku raames. Nii toote kui ka raamistiku töövoomootor on võimeline käivitama töövooge, mis on defineeritud kasutades eeldefineerituid funktsioone ja raamistiku. Positiivne moment on see, et suur osa standardseid objekte on eeldefineeritud, jääb üle vaid nende abil töövoog luua ja käivitada. Negatiivne on kitsendatus: defineerida ja programmeerida on võimalik kasutades etteantud funktsioone ja raamistiku, sest nende abil on tagatud töövoo toetus töövoomootori poolt. Antud tehnoloogiate kasutades on töövoomootori võimsus teisel kohal.

Kolmas võimalus on kasutada spetsiifilist süsteemi konkreetse ülesande klassi jaoks. Kitsendustega süsteem pakub kasutajale eeldefineeritud objektide kogumi, mille abil lõppkasutaja defineerib töövoo. Puudub võimalus uute objektide loomiseks. Töövoomootor ei ole väga võimas, sest ta toetab ainult eeldefineeritud objekte koos kitsendustega, mis on seotud ülesande klassi erilisusega. Töövoomootori võimsuse ja avatuse tõstmiseks pakuvad sellised süsteemid vahendeid töövoo defineerimiseks. Positiivseks aspektiks on see, et süsteem töötab optimaalselt ja kasutaja jaoks mugavalt terve klassi ülesannete jaoks, ning omab töövoode toimetajat. Negatiivseks momendiks on see, et teistsuguste ülesannete jaoks on kasutajal vaja teist süsteemi.

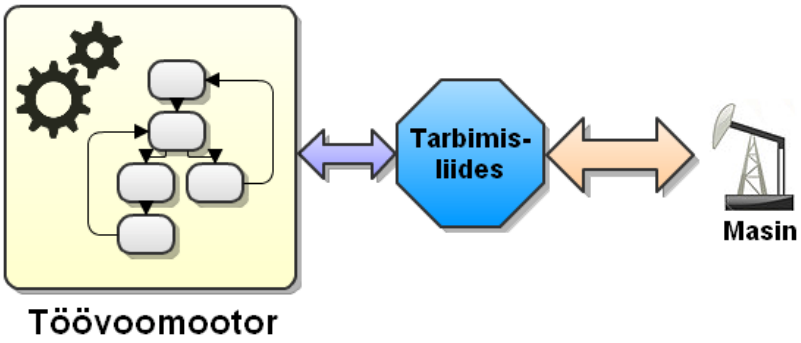
Neljas ja viimane tehnoloogia klass on püsiprogrammeeritud (*hard-coded*) süsteem. See on ühe probleemi lahendus. Töövoomootor on väga kinnine ja võimsuse poolest nõrk. Töövoo loogika ja reeglid on sisseehitatud töövoomootorisse. Positiivseks momendiks on see, et kuna töövoomootor ei ole võimas, seega ta on lihtne ja seega kiire (piisavalt kiire ka väiksema

riistvararessurssi korral). Miinuseks on see, et antud süsteemi on võimalik kasutada ainult konkreetses kontekstis. Töövoo uuendamiseks tuleb muuta rakenduse lähtekoodi ja uuesti kompileerida. Antud lähenemist on mugav kasutada siis, kui on tähtis süsteemi stabiilsus ja töövoogusid ei ole vajalik tihti muuta. Näiteks süsteem, mis automatiseerib masina tööd tehase tooteliinil.

Kõikidel defineerimisviisidel on omad positiivsed küljed ja omad nõrkused. See tähendab seda, et enne ülesande lahenduse leidmist, tuleb hoolikalt analüüsida eesolevat ülesannet, proovida seda klassifitseerida ja valida tehnoloogia, mis on maksimaalselt sobiv ja majanduslikult kõige kasulikum antud ülesande lahendamiseks.

1.4 Püsiprogrammeeritud süsteemid

Süsteemid, kus töövoo loogika on kirjeldatud koodi tasemel, nimetatakse püsiprogrammeeritud (*hard-coded*) süsteemideks. Antud töövoo haldamissüsteemid koosnevad töövoomootorist ja tarbimisliidesest. Definitsioonide hoidla (näiteks, andmebaas) ja definitsioonide modelleerimisliides puuduvad. Seega antud tüüpi süsteemid ei ole nii paindlikud nagu teised süsteemid. See tähendab, et puudub võimalus püsiprogrammeeritud süsteemi loogikat odavalt muuta. Sellepärast antud süsteeme kasutatakse siis, kui ülesanne või tööprotsess, mida kirjeldab süsteemi töövoog, ei muutu mitte kunagi (või muutub väga harva). Antud süsteemi abil ei ole võimalik luua ega muuta olemasolevat töövoo definitsiooni, sest see on juba tehtud süsteemi programmeerimise ajal. Antud süsteemid on esimese põlvkonna töövoo haldamissüsteemid. Näidiseks on rakendus, mis omab sisseehitatud töövoo loogikat ja töötab klaastaara tehases. Süsteem automatiseerib masinate tööd, mis valmistavad ja kontrollivad klaaspudeleid. On selge, et pudelite valmistamise ja kontrollimisloogika ei muutu tihti. Seega, selle töövoo käivitamise jaoks ei ole vaja keerulist ja universaalset töövoo haldamissüsteemi.



Joonis 6: Püsiprogrammeeritud süsteem tehases

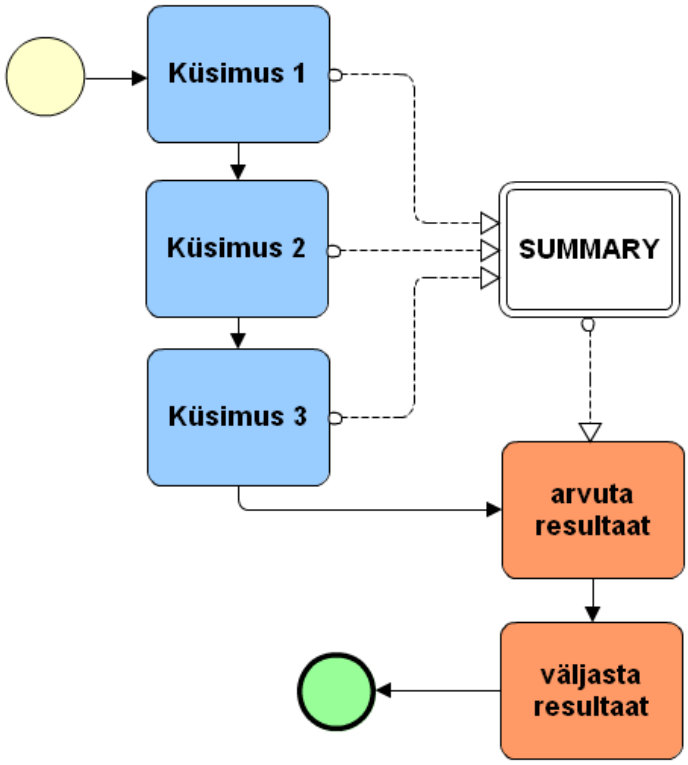
Joonisel 6 on illustreeritud lihtsustatud püsiprogrammeeritud süsteem, kus on nõutud stabiilne, kiire, ilma vigadeta tehasetöö.

1.5 Kitsendustega töövoo haldamissüsteemid

Püsiprogrammeeritud süsteemist avatum süsteem on kerge paindlik töövoo haldamissüsteem koos töövoode modelleerimisvõimalusega. Süsteemi tehniline ülesehitus on standardne: olemas töövoomootor, definitsioonide andmebaas, modelleerimis- ja tarbimisliidesed. Süsteem võimaldab defineerida, muuta ja käivitada töövoogusid. Graafiline toimetaja on mugav abivahend töövoode loomiseks ja redigeerimiseks. See annab võimaluse kasutada graafilist keskkonda, mis võimaldab intuitiivselt manipuleerida töövoo kui diagrammi objektidega. Peale töövoo definitsiooni loomist või muutmist on toimetajal võimalus salvestada diagramm töövoo definitsiooni andmebaasi. Sel momendil hakkab tööle konverter, mis serialiseerib virtuaalse graafilise diagrammi sellisele kujule, kuidas seda hoitakse definitsioonide andmebaasis. THS tarbimisliidese ja töövoomootori abil interpreteeritakse ja käivitatakse graafilise toimetaja abil loodud töövoo definitsiooni.

Näiteks on autokooli eksami testide süsteem. Iga küsimus on üks pilet, kus on toodud ka vastuste variandid. Inimene, kes vastutab eksami testide eest, loob iga pileti eraldi, lisades teksti, küsimuse, vastused ja vajadusel pildi või skeemi. Iga küsimus on üks töövoo samm ning sammu tegevus on küsimuse esitamine, millele vastamisel eksamineerijat suunatakse teisele küsimusele. Pileti järjestamisega on lihtne tegeleda, kuna graafilise toimetaja abil on lihtne siduda kahte töövoo objekti. Antud seos näitab, et peale ühe küsimuse vastamist,

eksamineeritav näeb ekraanil järgmist küsimust vastavalt töövoo diagrammile. Peale kogu küsimustiku loomist salvestab vastutav isik selle töövoo definitsioonina. Süsteem serialiseerib testi ja salvestab selle andmebaasi hilisemaks käivitamiseks. Eksamiküsimuste süsteem sisaldab tarbimisliidest, mis kooskõlas töövoomootoriga näitab eksamineeritavale küsimusi ja võtab vastuseid vastu, salvestades neid hilisemaks kontrollimiseks. Süsteemil on olemas ka õiged vastuseid küsimustele, selleks, et peale eksamit võrrelda neid vastustega, mis saadi eksamineeritava käest. Hinne sõltub eksamineeritava poolt tehtud vigade arvust. Antud vastuste kontrolli teeb süsteem automaatselt peale kõigi küsimuste vastamist, see tähendab et antud töövoo tegevus, mis tegeleb vastuste kontrolliga, on täisautomatiseeritud ja ei nõua inimeste sekkumist. Joonis 7 illustreerib lihtsustatud autokooli eksami töövoo diagrammi.



Joonis 7: Lihtsustatud autokooli eksami töövoog

1.6 Teised süsteemid

Ülejäänud süsteemid on kolmanda ja neljanda põlvkonna süsteemid, mis on keerulised universaalsed produktid, mille omadusteks on lai funktsioonide ulatuses, paindlikkus,

integreeritavus, avatud standardid ja liidesed, keerulisus. Neid võib nimetada kõik-ühes („All-in-one“) süsteemideks, sest süsteem koosneb rakendustest, mis on tihedalt seotud omavahel ning katavad kõiki kõige levinumaid ülesandeid, alustades piltide töötlemisest ja dokumentide haldamisest ja lõpetades ettevõtte töötajate tööülesannete loomisega, muutmisega ja optimeerimisega. Näidetena on vaatluse all *ERP* komplekssüsteemid ja *Microsoft WWF* raamistik.

1.6.1 ERP süsteemid

Eksisteerivad suuremahulised süsteemid, mida kasutatakse suurtes ettevõtetes sise- ja välisressursside haldamiseks. Antud tüüpi süsteemid keskenduvad universaalsete äriülesannete lahendamisele (tootmine, müük). Need on paindlikud süsteemid piiratud arvu ülesande klasside jaoks. Antud süsteemide hulka kuuluvad *ERP* süsteemid (*Enterprise Resource Planning* – Ettevõtte Ressursside Paneerimine), mis on integreeritud arvutipõhine süsteem, mida kasutakse sise- ja välisressursside haldamiseks sh. ka rahalised vahendid, materjalid ja inimressursid. See on tarkvaraline lahendus, mille eesmärk on ühtlustada informatsioon erinevate ettevõtte osakondade ja divisjonide vahel ja hallata ühendusi välispartneritega. Üheks reegliks on see, et antud süsteemi kõik rakendused kasutavad ühist andmebaasi selleks, et vältida tarbetute andmete olemasolu ja mitmekordset andmete kirjeldust. Tavaliselt ERP-rakendused kasutavad ühist arvutiplatvormi, ning koondavad kõiki äritegevusi ühtseks ja laiaks süsteemi keskkonnaks. ERP süsteemid võivad paikneda ühes tsentraliseeritud serveris või olla jaotatud modulaarsete riist- ja tarkvara üksuste vahel, mis pakkuvad teenuseid ja suhtlevad omavahel lokaalvõrgu abil. ERP süsteemid omavad juba eeldefineeritud mehhanisme ja töövooge ettevõtte ülesannete jaoks nagu: tootmine, ajakava, töövoo juhtimine, kvaliteedi kontroll, kulude juhtimine, tellimused, arveldus, kaebuste töötlemine, maksmine, efektiivsuse kontroll, inimressursid, koolitused, palgafondid, töökorraldus, vahendustasud, teenindus jne.

ERP süsteemi paigaldamine ja rakendamine nõuab suuri oskusi ja tavakasutaja ei suuda juurutamise käigus tekkivaid probleeme lahendada. Süsteemi paigaldus peab olema tehtud väljaõpetatud spetsialisti pideva kontrolli all. Antud paigaldamisviis suurendab veelgi niigi kõrget süsteemi hinda. Juurutamine võtab suhteliselt palju aega, mis sõltub moodulite arvust

ja ettevõtte suurusest. Sellise keerulisuse tõttu võib kogu juurutamisprotsess võtta väikestes ettevõtetes 3-9 kuud ja suurtes rahvusvahelistes ettevõtetes mitu aastat aega.

ERP süsteemid on standardiseeritud enamlevinud äriprotsesside osas. Seega ettevõtted proovivad viia omad tööprotsessid ja töövood vastavusse ERP poolt pakutavale, mis võib viia selleni, et ettevõtte tööprotsessid kaotavad osa oma efektiivsusest, sest tihti spetsiifiline tööprotsess annab rohkem kasumit kui standardiseeritud. Selle tõttu, et juurutamisprotsess on väga keeruline ja et süsteem maksab väga palju, peab ettevõtte tegema kõigi oma protsesside jaoks suure analüüsi, mille alusel valitakse ERP süsteemi tootja ja vajalikke moodulite konfiguratsioon.

ERP süsteemi mugavus seisneb selles, et ta pakub lahendust luua, käivitada ja hallata kõik standardseid majanduslikke protsesse, töövooge ja ülesandeid, mis võivad ettevõttes tekkida, alates toodete disaini loomisest kuni kulude, tulude ja kasumi jälgimist. ERP süsteemi andmete tsentraliseeritavus annab omakorda mugavusi juurde:

- Puudub vajadus sünkroniseerida andmeid erinevate süsteemide vahel
- Kogu vajalik ettevõtte informatsioon ja andmed on kättesaadavad ühest kohast
- Vähendab andmete kaotamisriske
- Võimaldab jälgida kõike protsesse ühest kohast

Suur ettevõtte, mis ei kasuta ERP tarkvara peab kasutama suurt hulka eraldiseisvaid programme, mis ei oska suhelda omavahel – see on ebamugav. ERP süsteemi rakendused on disainitud maksimaalselt efektiivseks omavaheliseks suhtlemiseks ja andmete omavaheliseks vahetamiseks.

Nagu kõik süsteemid omab ka ERP süsteem ebamugavusi:

- ERP-s on eeldefineeritud kõik töövood standardselt. See ei anna võimalust väga täpseks häälestamiseks, mis omakorda ei anna võimalust luua väga spetsiifilist tööprotsessi, mis on hädavajalik ettevõtte äri jaoks.
- ERP süsteemid on väga kallid.
- ERP süsteeme peetakse tihti liiga jäigaks ja need on liiga raske adopteerida olemasolevate töövoode ja äriprotsesside jaoks.
- Turvalisuse aspekt on väga tähtis, seoses andmete hoidmisega ühes kohas.

Suured ERP süsteemide arendajad ja tootjad on *SAP, Oracle, OpenERP, Syspro, Microsoft* ja teised.

1.6.2 Microsoft WWF

Eksisteerivad süsteemid, mis ei ole nii keerulised nagu ERP süsteemid ja annavad rohkem vabadust töövoogude loomisel ja käivitamisel. Töövoo reeglid on paika pandud, aga mida iga töövoo element teeb, see programmeeritakse ise. Sellise tüüpi süsteemide hulka kuulub *Microsoft*'i poolt arendatav *WWF* (*Windows Workflow Foundation*), mis on .NET raamistiku osa. See on üldotstarbeline tehnoloogiline programmeerimisraamistik, mis võimaldab luua „reaktiivseid" (*reactive*) programme, mis reageerivad vastuseks tegevustele väljast. „Reaktiivsete" programmide põhikarakteristik on see, et nad oskavad käivitamise ajal peatuda ja oodata sisendit, mistahes kaua. Sisendiks saab kas inimese või välisprogrammi poolt tulnud tegevus.

Tegevuste käivitamine on loomupäraselt episoodiline: tegevus võib peatuda oodates sisendit ja jätkuda, kui sisend on kätte saadud. Samamoodi nagu lugedes pesumasina juhendit, pannakse see kinni ja kasutatakse järjehoidjat. Hiljem, kui on vaja, avatakse juhend sel lehel, kus järjehoidja parasjagu on ja loetakse teksti edasi. Raamat „ootas" välispoolset sisendit, mis on juhendi taasavamine. Kuna WWF programmi iseloom on peatuda ja käivituda sisendi kättesaamisel, siis WWF programmi loogika on kirjeldatud töövoogude abil. See tähendab, et iga saamu järel WWF programm ootab sisendit, millest sõltub järgmise programmi samm. See annab võimalust programmi juhtida kasutades andmeid, mis saadetakse sisendisse. Termin „reaktivne" programm on laialt kasutatav meie reaalses elus: neid kasutatakse ühiste dokumentide muutmistel, veebipõhise kauplemise puhul, toormaterjalide hankimisel ja teistel tegevustel, mis nõuab kas inimeste või teiste programmide tähelepanu ja valiku tegemist. Summeerides, WWF pakub deklaratiivse programmeerimismudeli kirjutamiseks ja käivitamiseks „reaktiivseid" programme, kasutades eeldefineeritud kokku ühendatud tegevusi. Eristatakse WWF programme, mis kasutavad kolme erinevat töövoo tüüpi:

1. Järjestikune töövoog – käib algusest lõpuni, kus töövoomootor alustab iga sammu automaatselt.

2. Interaktiivne töövoog on juhitav väliste tegevuste ja kasutajaga suhtlemise abil
3. Reeglitepõhine töövoog eelmiste kahte töövoo tüüpide konstruktsioon. Töövoog käib eeldefineeritud reeglite järgi, vahepeal oodates välispoolseid suhtlemisi

Kõik töövood võivad kasutada hargnemist ja tsükleid. Tehniliselt WWF programmi töövood on esitatud *XAML* (*Extensible Application Markup Language*) keele abil ning annab .NET arendajatele võimalust eraldada töövoo loogika rakenduse käivitamiskomponentidest. See annab mugavust programmi töövoo loogika programmeerimisel ja analüüsimisel. Iga töövoog sisaldab ühte või rohkemat tegevust, kus iga tegevus täidab eraldatud ja diskreetset loogika osa. Arendajatel on võimalus kirjutada oma tegevusi ja hiljem kasutada neid töövoo koostamisel ehitusplokkidena. WWF raamistik pakub juba eeldefineerituid standardseid üldotstarbelisi tegevusi, mis käivitavad mõnda voogu kontrollimiskonstruktsioonist: if...else loogika, paralleel käivitamisloogika, tsüklite loogika ja nii edasi.

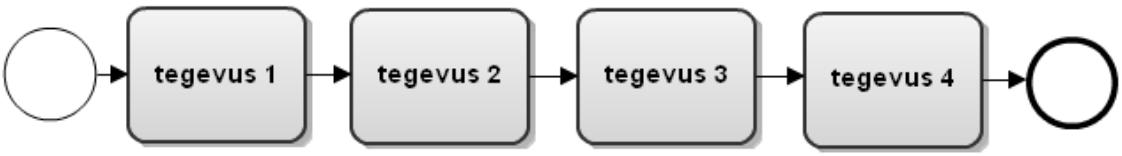
2. Ülesannete klassifitseerimine

2.1 Rollide arvu järgi

Ülesandeid ehk probleemvaldkondi, mida on võimalik automatiseerida ja lihtsustada kasutades töövoo tehnoloogiat on palju. Ülesandeid on võimalik klassifitseerida mitmete parameetrite järgi. Näiteks, ülesandega seotud rollide arvu järgi. Selle järgi saame kaks jaotust. Ülesannete klass, kus ülesandega on seotud üks inimene või masin ehk seotud üks roll. Näiteks, ravimi retsepti väljakirjutamine nõuab vastava vormi täitmist ühe arsti poolt. Teine oleks ülesannete klass, kus ülesandega on seotud mitu rolli: köögielektroonika valmistamise tooteliin.

2.2 Diagrammi tüübi järgi

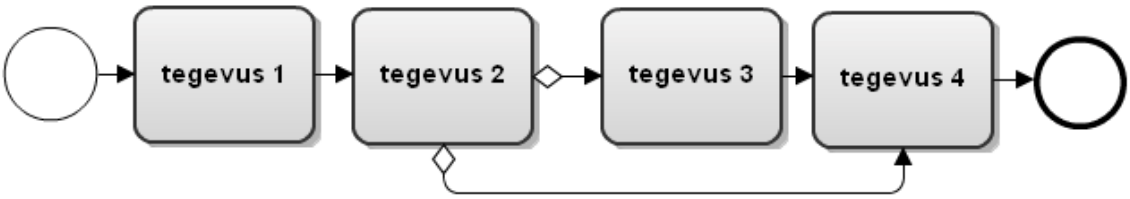
Teine viis klassifitseerimiseks on jagada ülesandeid vastava töövoo diagrammi tüübi järgi. Näiteks, automatiseeritud ülesanded, mille töövoog on järjestikune, kus ülesande lahendamiseks tehtavaid tegevusi läbitakse järk-järgult (vt. Joonis 8). Antud klassi ülesanded on tingimusteta järjestatud tegevuste komplekt, näiteks, dokumenti konverteerimine ühest formaadist (*MS Word*) teisse (*Adobe PDF*).



Joonis 8: Järjestikune töövoog

Eksisteerivad ülesanded, mille lahendus on järjestikune tingimuslik töövoog. Tähtsad on tegevuste resultaadid, aga mõni osa tegevustest võib sõltuda ühest eelmistest tegevuse väljundist (vt. Joonis 9). Näiteks, statistilise rahuolu uuringu kohta küsimusi küsitakse järjest, ning vastuseid salvestatakse, samas mõne küsimuse vastus võib viia selleni, et osa

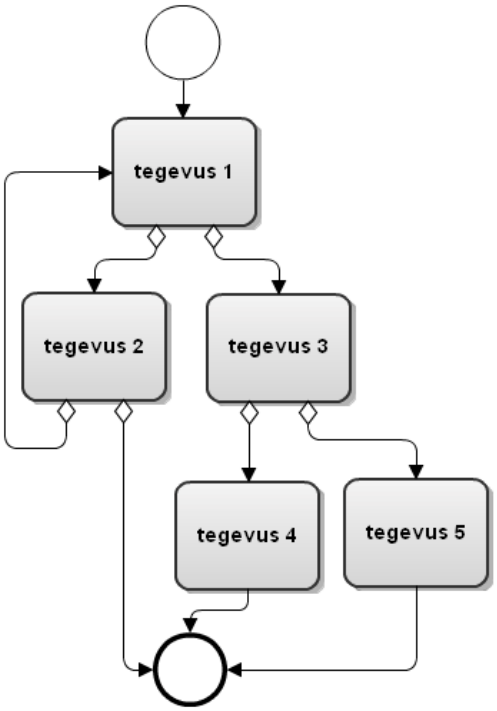
küsimustest jäetakse küsimata. Näiteks, küsimuse „Kas teil on televiisor“ negatiivne vastus viib selleni, et küsimus „Millist kaabeltelevisiooni operaatorit te kasutate?“ jäetakse küsimata.



Joonis 9: Tingimuslik järjestikune töövoog

Teiseks näiteks siin on *ant*-skript [44], mis on automatiseeritud tarkvara kompileerimiseks ja juurutamiseks juhtimiskäskude järjestikune kogum.

Teine klass on selline, kus ülesannete töövoo diagrammid on graafi- või puukujulised. Tingimuslikult järjestatud tegevused, hargnemine, tsüklid – need on antud klassi ülesande diagrammi omadused. Antud juhul iga tegevus sõltub eelpool olevate tegevuste resultaatidest (vt. Joonis 10). Tihtipeale antud klassi ülesanded omavad interaktiivset tüüpi lahendusi: leidub tegevusi, mille kohta süsteem küsib vastust kasutajalt või välisrakenduserakenduse käest.



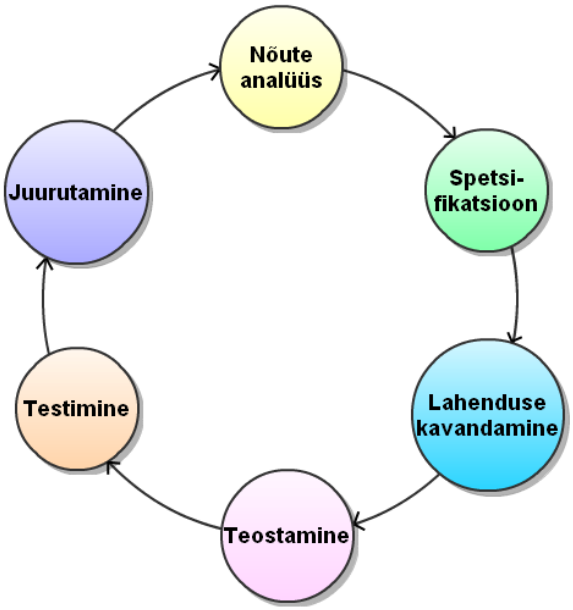
Joonis 10: Graafikujuline töövoo diagramm

2.3 Töövoo kestuse järgi

Kolmas viis on jagada ülesandeid töövoo eluea pikkuse põhjal. Sel juhul ülesanded jagunevad pikaajalisteks ning lühiajalisteks. Pikaajalised protsessid toimuvad tunde, päevi või kuid.

Definitsioon. Kui ülesande lahenduse töövoo eluea pikkuse mõõteühik on tund, päev, kuu või suurem ühik, siis antud ülesannet nimetame **pikaajalisteks** ülesandeks.

Pikkajaliste ülesannete hulka kuulub näiteks dokumenti haldamissüsteem, autode valmistamine tehases või suure internetipoe tellimussüsteem. Materjal, energia, informatsioon paber- ja elektroonkujul liigub läbi erinevaid samme transformeerides vastavalt loogikale ja reeglitele, lõppude lõppuks muutub lõppdokumendiks, lõpp-produktiks ja lõppteenuseks. Tarkvaraarendamise puhul antud pikaajaline informatsiooni töövoog on eriti tuntud skeem (vt. Joonis 11).



Joonis 11: Tarkvara arendusprotsess

On selge, et tarkvara arendusprotsess hõlmab endas palju informatsiooni, energiat, inimressursse. Erinevate rollide sisaldavus on kenasti nähtav: Nõuete analüüsiga,

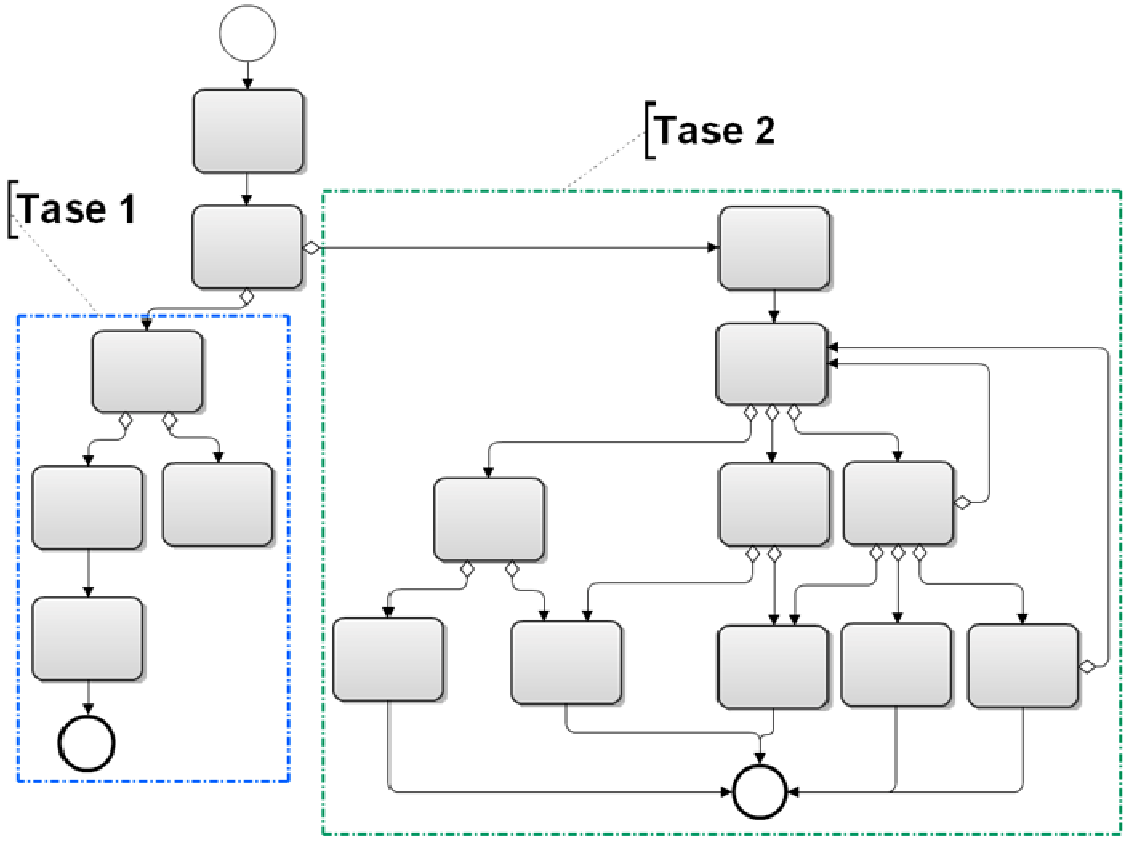
spetsifikatsiooni koostamisega tegelevad ühed inimesed, lahenduse kavandamisega – teised, kolmandad programmeerivad süsteemi, neljandad testivad seda ja viiendad juurutavad valminud süsteemi. Igal inimesel on oma roll antud töövoos ning programmeerimine ei saa olla alustatud enne rakenduse lahenduse kavandamist. Võib juhtuda ka see, et kõikide nende töödega tegeleb üks inimene. Sel juhul omab tema erinevaid rolle erinevates tarkvaraarenduse töövoos.

Internetipoe tellimussüsteem koosneb paljudest etappidest, mis on seotud omavahel ning iga etapiga on seotud konkreetne roll. Lihtsustatud töövoo protsessi kirjeldus: Inimene valib toote internetipoest, peale seda paneb selle korvi, siis maksab selle eest. Tellimus salvestatakse lokaalsesse andmebaasi, peale seda haldur vaatab, mis sorti tellimust tehti. Kui toode on rõivas, siis saadetakse tellimus edasi osakonda, mis tegeleb riietega, kui tegemist on elektroonikaga, siis saadetakse osakonda, mis tegeleb elektroonikaga. Igal osakonnal on oma ladu. Ladutöötaja leiab laos tellitud kauba, vormistab paberid ja saadab pakkimisele. Pakkija võtab kauba vastu ja pakib. Logistikaettevõtte, mis töötab koos internetipoega, võtab pakkijalt pakitud kauba ja tarnib selle füüsiliselt ostjale. Ostja võtab kauba vastu ja annab allkirja selle kättesaamise kohta.

Patsiendi ravimissüsteem samamoodi kuulub pikkajaliste ülesannete klassi.

Antud klassifikatsiooni järgi või tekkida ka erandeid. Ülesande lahenduse töövoog võib olla pikaajaline ning teatud juhtudel lühiajaline. Näiteks on kaabeltelevisiooni ettevõtte klienditoe (*help-desk*) ülesanne. Iseloomu poolest, antud ülesande töövoog on pikkajaline: kaebuse/probleemi kirjapanemine, selgitamine, kes sellega tegelema hakkab, probleemiga tegelemine ja lahendamine, probleemi lahendamise kirjeldamine, kliendi probleemi lahendusest teavitamine on pikkajaline protsess, kus tegevuste pikkused kestavad tunde ja päevi. Seda nimetatakse „Klienditoe teine tase“ (*help-desk level 2*).

Samas aga antud ülesannet on võimalik teatud juhtudel lahendada minutitega, kliendi halduri abiga. Näiteks, kui klient helistab ja ütleb, et temal ei näita mõnda kaabelkanali ja operaatoril on teada see, et sel momendil toimuvad hooldustööd, siis ta võib vastata kliendile (ilma mingite tehniliste spetsialistide häirimist), milles on probleem ja millal on oodata muutust. Antud vestlus võib kesta mõne minuti ja selle ajaga saadakse probleemile lahendus. Seda nimetatakse „Klienditoe esimene tase“ (*help-desk level 1, front line support*).



Joonis 12: Tase 1 ja Tase 2 klienditugi

Joonisel 12 on näha ülesande töövoo diagrammi graafi kujul, mis koosneb kahest harust, parempoolne haru on pikaajaline protsess ehk klienditoe teine tase, nii öelda, ülesande standardne lahendus, mis nõuab mingi ressursi kasutamist (spetsialistide tööaeg, seadmete parandus jne) ja vasakpoolne haru on lühiajaline protsess ehk klienditoe esimene tase, kus operaator lahendab antud ülesannet üksinda, seega ei nõua lisaressursi kasutust.

Pikkajaliste töövoo protsesside puhul on rollide kasutamine tähtis, kuna iga rolliga on seotud hulk ülesandeid ja prioriteete – see annab efektiivsust ja mugavust töövoo koostamisel.

Definitsioon. Kui ülesande lahenduse töövoo eluea mõõtühik on minut või sekund, siis antud ülesannet nimetame **lühiajalisteks** ülesandeks. Lühiajaliste ülesannete lahenduse töövoo alamtegevused ei katke ja teostatakse esiplaanil.

Definitsiooni kohapealt protsessi katkestamine ja tahaplaanile panemine on situatsioon, kus käesolev protsess pannakse ootele, kuna üks protsessi tegevus lõpetas oma töö ja järgmine

pole veel alustanud oma tööd, sest ootab mingi ressursi taga. Samas, kui üks protsess pannakse ootele selleks, et sel ajal lahendada teist sarnast lühiajalist protsessi, siis seda nimetame protsesside paralleelseks lahenduseks. Antud situatsioon ei ole protsessi katkestamine ega ootele panemine ja tagaplaanile saatmine.

Näiteks on panga ülekanne ühelt arvelt teisele. Rahasaatja tuleb telleri juurde, seletab oma soovi, annab vajadusel raha, teller trükib kviitungi, klient allkirjastab ja selle dokumendi põhjal teller juba teeb ülekande ühelt arvelt teisele. Antud protsess tavaliselt ei võta rohkem kui 5 minutit aega standardsete tingimuste korral. Juhtub aga nii, et teller ei saa ülekannet teha mingite probleemide tõttu: kas paberrahaga on probleemid, saaja arvel ei ole piisavalt raha või tekkis tehniline tõrge. See on hoopis ülesande lahenduse erand, mis ei mõju ülesande klassifitseerimisprotsessile, sest oletame, et ülesande lahenduse ajal kehtivad antud tegevusele standardsed tingimused ja ei teki mittestandardseid (*force majeure*) situatsioone.

Lühiajaliste töövoogude hulka võime lisada ka kiirinfo saamise telefoni teel, kaabeltelevisiooni tehnilise toe kõnetöötlust, küsitluse läbiviimist. Kuna lühiajalise töövoo protsessi elutsükkel on päris lühike, näiteks 10-15 minut, siis antud töövooge kasutatakse siis, kui on vaja kiiret reageerimist. Näiteks, häirekeskuse dispetšer peab kiiresti reageerima, vastama kõnele ja lahendama probleemi.

Info hankimine telefoni abil nõuab ka kiireid töid. Infoliini teenindajale on abiks programm, mis kiiresti leiab vajaliku informatsiooni, mida nõuab klient. Sarnase tööga tegeleb kaabeltelevisiooni tehnilise toe spetsialist, kes võtab vastu kõnesid, siis määrab kindlaks, millise valdkonna probleemiga tegu on ja suunab helistaja pädeva spetsialisti juurde.

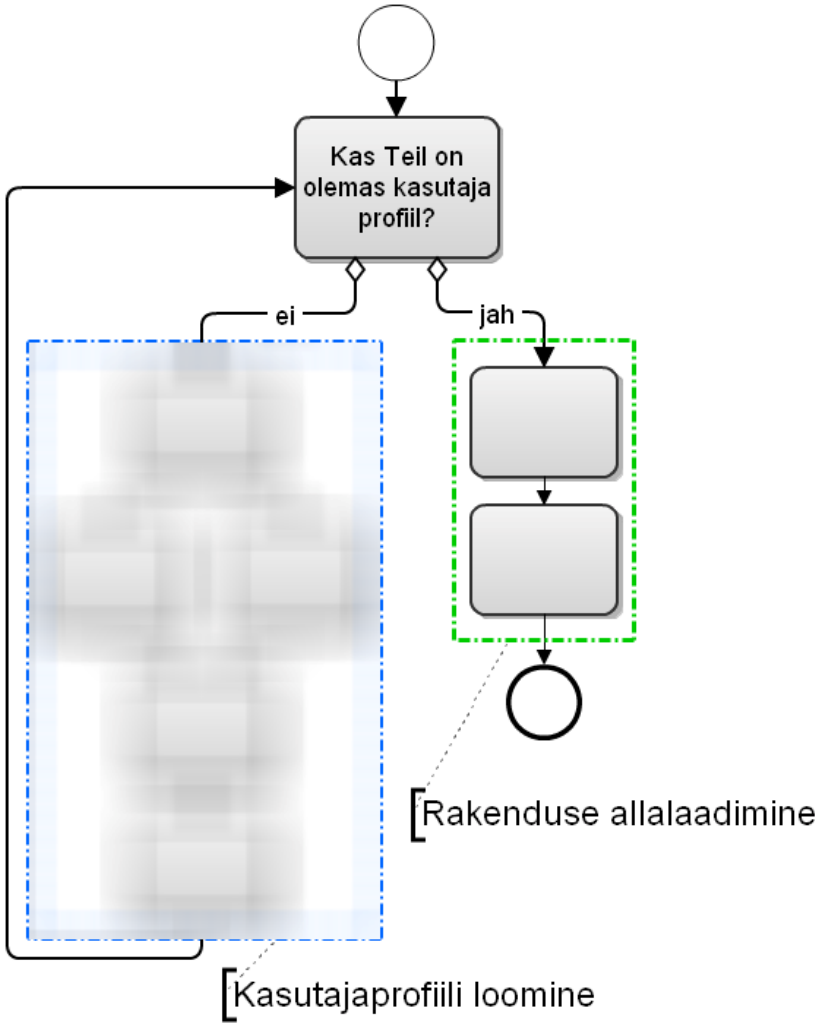
Eksisteerivad ka küsitlused, mis kuulivad ka lühiajaliste ülesannete klassi. Küsitlus võib olla mitmetes erinevates vormides:

- inimene istub koos küsitlejaga
- küsitleja helistab inimesele ja küsib küsimusi

Küsimustele eeldefineeritud vastusvariantidest sõltub järgmine küsimus. Tavaliselt küsitlust saab ette kujutada orienteeritud graafi abil. Iga tipp on küsimus, iga kaar ühendab kahte küsimust. Ühest küsimusest teisse saab liikuda kasutades ainult ühte suunda. Eksisteerib algküsimus ja lõppvastus. Graafi mõiste on kasutatud selle poolest, et ühest tippust on

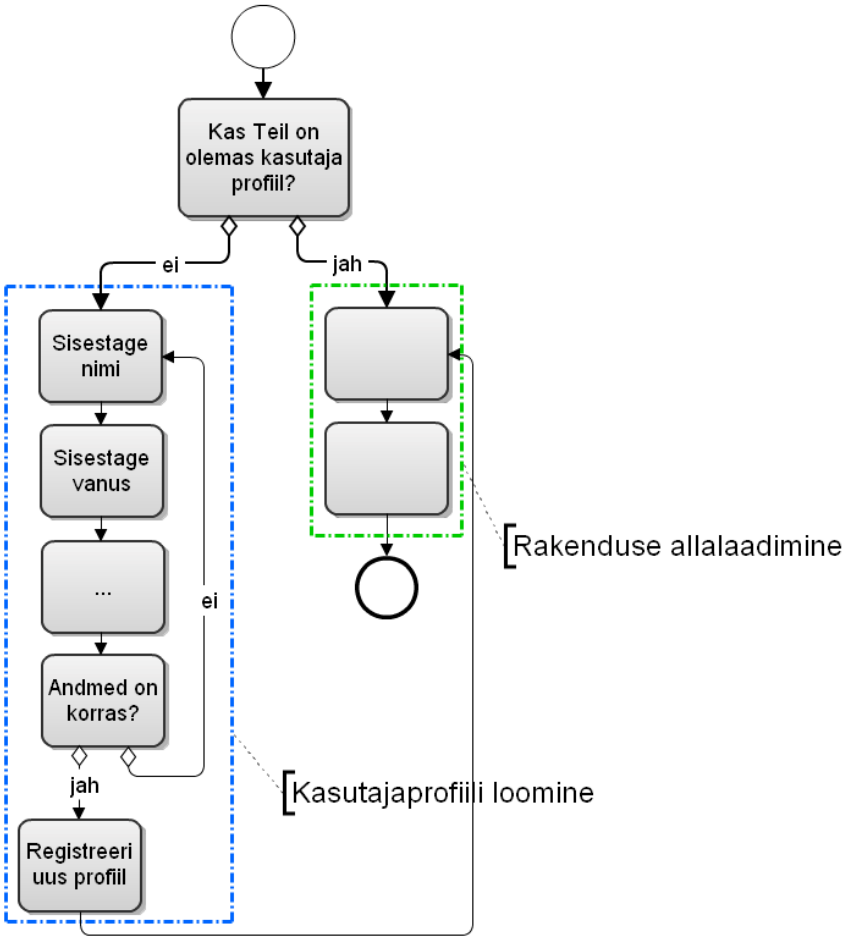
võimalik liikuda teise tippu, mis oli juba varem vastatud. Selle mehhanismi abil on tagatud tegevuste tsüklilisus, mis on võimalik kasutada probleemi lahendamisel.

Näiteks on situatsioon, kus tekkib vajadus alla laadida rakenduse uus versiooni veebisaidist, mis nõuab kasutaja profiili olemasolu antud saidis. Süsteem – küsimustik, mis juhib kasutajat rakenduse allalaadimisel, küsib kasutajalt “Kas teil olemas kasutajaprofiil?”. Kui kasutaja vastab, et “Jah, on olemas”, juhendab süsteem kasutajat, millisele nupule tuleb vajutada, et edukalt rakendus alla laadida. Kui kasutaja vastab “Ei”, siis süsteem juhib kasutajat läbi kasutaja registreerimisprotsessi, mille tulemusena on loodud kasutaja profiil. Selleks, et veenduda, et kõik on korras, süsteem küsib veel kord “Kas teil olemas kasutajaprofiil?”. Juhul kui registreerimine õnnestus, kasutaja valib “Jah, on olemas”, juhul kui registreerimine ebaõnnestus mingite probleemide tõttu, näiteks serveri ülekoormuse tõttu, süsteem juhib kasutajat läbi registreerimisprotsessi veel kord. Situatsiooni kirjeldav töövoo diagrammi osa on toodud joonisel 13.



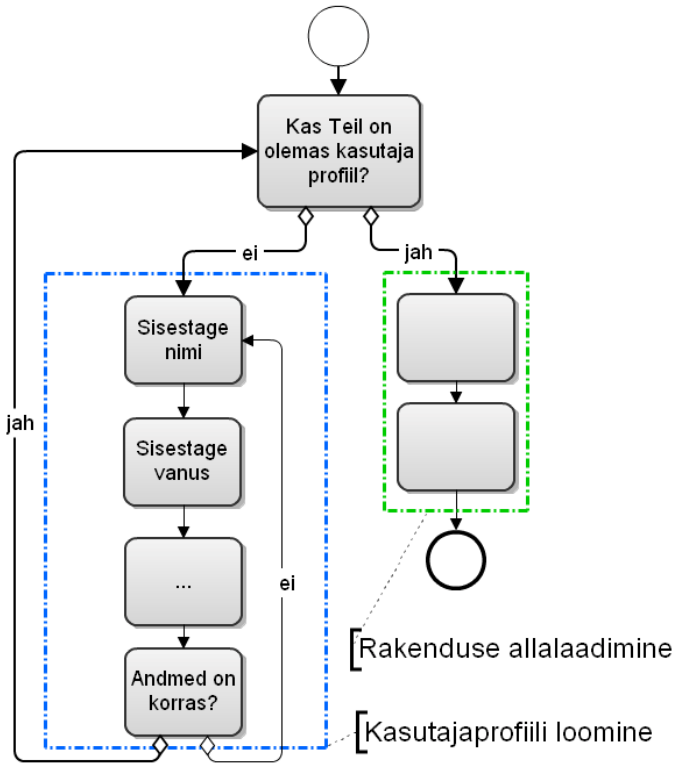
Joonis 13: Rakenduse allalaadimine

Juhul kui kirjeldatud süsteem uue kasutaja profiili registreerimisel küsib kasutajalt tema andmeid ja lõppude lõppuks ise registreerib kasutajat antud saidis (omab aktiivset interneti ühendust), siis see süsteem lahendab komplitseeritud ülesannet, mille põhieesmärk on otsuseni jõudmine, aga samas ta on võimeline koguda ka vajalikke andmeid, töödelda ja kasutada automatiseeritud tegevusi ja mehhanisme (vt. Joonis 14).



Joonis 14: Rakenduse allalaadimine koos automaatse profiili loomisega

Teine variant on selline, et ülalpool kirjeldatud süsteem on ainult abivahend ja mõeldud selleks, et juhtida kasutajat läbi dialoogivormis tegevuste konkreetse eesmärgini (vt. Joonis 15).



Joonis 15: Rakenduse allalaadimine koos profiili loomisjuhendiga

Mõlematel juhtudel (Joonis 14 ja Joonis 15) lahendused omavad teatud tsüklilisust selleks, et võimalikult kvaliteetselt lahendada uue kasutajaprofiili loomisülesannet.

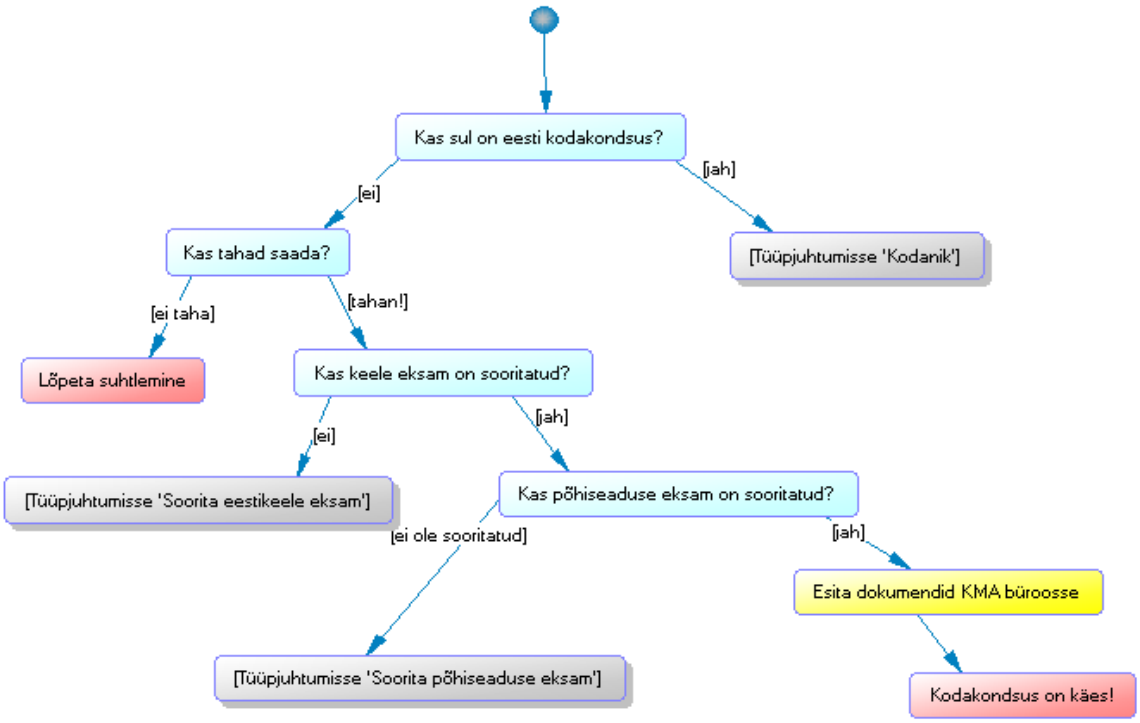
Valikvastustega küsimustiku lühiajaliste ülesannete klassi kuuluvad ka autokooli testid, mis toimuvad tavaliselt 45 minutit (vt. Joonis 7) ja häirekeskuse kõnetöötlus. Sellise klassi ülesandeid nõuavad kiirreageerimist ja suhteliselt kiiret lahendusaega.

Ülesanded klassi sees omavad ühiseid omadusi ja erilisusi. Antud lühiajaliste ülesannete klassifitseerimise järgi tundub, et mitte kõik ülesanded pole sarnased. Võtame, näiteks, statistilist arvamusuuringu, mida tihti tehakse telefoni teel, ning häirekeskuse kutsetöötluse. On ilmne, et neid kaks ülesannet kuuluvad lühiajaliste ülesannete klassi aga eesmärgi ja ülesehituse järgi on nad erinevad.

Esimene ülesande eesmärk on info kogumine. Küsimusi küsitatakse järjest ja vastuseid pannakse kirja edaspidiseks analüüsimiseks. Küsimuste järjekord ei ole eriti oluline, kui vastaja ei oska/ei taha küsimusele vastata, siis küsija võib küsida antud küsimust ka hiljem,

sest suurim osa küsimustest on iseseisvad ja ei sõltu teistest küsimustest, näiteks: “Kui vana te olete?” ja “Millist mobiilside operaatori teenust kasutate?” (vt. Joonis 7).

Häirekeskuse ülesande puhul eesmärk on täiesti erinev: dispetšer küsib helistajalt küsimusi selleks, et jõuda konkreetse otsuseni. Näiteks, helistaja helistab häirekeskusesse ja küsib abi. Dispetšer küsib küsimusi ja selliste küsimuste vastuste abil tema teeb otsuse, kas on vaja kiirabi saata või mitte. Otsus on hädavajalik selleks, et kiirabi saatmine peab toimuma ainult teatud juhtudel, sest valede otsuste tegemine ei ole rahaliselt optimaalne. Lisaks, otsus annab ka inforatsiooni kasutatava ressursi prioriteedi kohta. Näiteks, kui kaks paralleelset kõnet, mõlemad ootavad kiirabi aga konkreetsel juhul ainult üks kiirabi on vaba, siis otsuse abil pannakse prioriteedid paika: kellele kiirabi antud juhul on kiiremalt vajalik ja kes jääb ootele. Seoses sellega, antud lühiajaliste ülesannete klassi saab omakorda veel alamklassideks jagada: need, mis tegelevad andmete kogumisega korduva iseloomuga tegevuste läbi (autokooli eksam, statistiline arvamusuuring) ja sellisteks, mis tegelevad otsusele jõudmisega (häirekeskuse dispetšer, klienditoe haldur), kus on tähtis küsimuste läbimisjärjekord ja otsus, mitte tegevuste tulemused iseseisvalt (vt. Joonis 16).



Joonis 16: Lühiajaline ülesanne otsuseni jõudmise eesmärgiga

Samas eksisteerivad ka erandid ehk komplekssed ülesanded, mis omavad nii esimese ja ka teise lühiajaliste ülesannete alamklassi omadusi: ülesande lahendamisel korjatakse informatsioon kokku ja mõnedes juhtudes jõuakse otsuseni läbi teiste tegevuste. Antud klassi kuuluvad näiteks tarkvara esmaseadistused (*wizard*), mis võimaldavad kiiresti ja lihtsalt lahendada ülesandeid läbi valdkonnaspetsiifiliste küsimuste. Selle abil on võimalik nii vajalikku informatsiooni koguda ja kasutajale interaktiivses vormis valikvastustega küsimus esitada. Eristatakse kahte tüüpi esmaseadistusi: lihtne (*plain*) ja juhtiv (*guided*). Esimene on esitatav järjestikuste tegevuste jadana (vt. Joonis 8 ja 9), seega töövoo läbimistee on eeldefineeritud viitetarga loomise ajal. Teist tüüpi esmaseadistus on rohkem sarnane ekspertsüsteemiga, sest vastava töövoo läbimisetee ei ole defineeritud ja selgub alles viitetarga käivitamise ajal baseerudes kasutaja valikute peal. Antud juhul toimub nii andmete kogumine, kui ka interaktiivses vormis kasutaja poolt valiku tegemine, millest sõltub järgmine kasutajale näidatud lehekülg [21].

Esimese lühiajaliste ülesannete klassi (andmete kogumine) korral on töövoosüsteemi kasutamine piiripealne otsus. Teise klassi ülesandeid (otsuseni jõudmine) on võimalik lahendada kasutades töövoo tehnoloogiat, mis suures mahus lihtsustab tööd. Häirekeskuse kutsetöötlus, mis on võimalik esitada valikvastustega küsimustiku abil, kuulub lühiajalise ülesannete klassi, kus on tähtis otsuseni jõudmine piiratud ressursi kasutamise kohta. Baseerudes sellel, et antud ülesannete klassi on võimalik lahendada kasutades töövoo tehnoloogiat, võib järeldada, et häirekeskuse kutsetöötluse ülesannet on ka võimalik töövoogude abil lahendada.

3. Valikvastustega küsimustiku läbimise süsteem

Süsteem, mis lahendab valikvastustega küsimustiku ülesannet klassifitseerub ekspertsüsteemi erijuhiks, sest ta annab võimaluse leida probleemi lahenduse ja vajadusel selgitab ebakindlusi juhtudel, kus on vajalik ühe või rohkema valdkonnaeksperdi konsultatsioon [22]. Teadmiste hoidla ehk töövoo definitsioonid, mis sisaldavad reegleid küsimustiku läbimiseks aitavad töövoomootoril esitada tarbimisliidese kasutajale õiget informatsiooni. Neid süsteeme nimetatakse kõne-voog süsteemideks (*call-flow system*), sest dialoogivormis süsteeme tihti kasutatakse kõnekeskustes. Kolmas nimetus on graafil-baseeruv süsteem (*graph-based system*), sest informatsioon, mida kasutatakse küsimustiku läbiviimisel on struktureeritud graafkujul. Neid süsteeme ilmselt defineerivad dialoogi kõik võimalikud seisundid ja nende seisundite vahelised võimalikud üleminekud. Selle abil kompleksne probleem on jagatud piiratud arvuks lihtülesanneteks. Olekute vahelise ülemineku valimine toimub väliskeskkonnaga suhtluse baasil, näiteks kasutaja poolt valiku tegemisel [25].

Lisaks sellele, valikvastustega küsimustiku süsteem on *agenda*-baseeruv süsteem, sest otsuseni jõudmiseni küsimustikkude kogum on esitatud graafi abil, kus küsimused on struktureeritud naturaalse hierarhia ja järgnevuse põhimõttel. Agenda-baseeruv süsteem spetsifitseerib kõiki võimalikke otsuseni jõudmise teid [25]. Hierarhia põhimõttel, küsimus, mis seisab juurele (algküsimusele) lähemal on prioriteetsem ja üldisem küsimus, sest tema defineerib alamvaldkonna alluvate küsimuste jaoks. Samas juurest leheni jõudmisel suurendatakse küsimuse spetsiifilisust selle momendini, kus otsus on käes.

Valdkonnad, kus on võimalik valikvastuste küsimustiku kasutada, ei ole ainult kõnekeskus, häirekeskus, tehniline tugi ja sarnased. Allika [24] autorid pakkuvad võimalust lihtsustada etalon-protsessmudeli konfigureerimist (*Reference Process Model*) [28] valikvastuste küsimustiku põhimõtte abil. Seoses sellega, et etalon protsessmudeli konfigureerimine nõuab selle mudeli modelleerimiskeele ja valdkonna aspektide konfiguratsiooni kohta teadmust, muutub protsess iseeneselt päris raskeks. Autorid kirjeldavad süsteemi, mis inimkeeles küsimuste küsides loob etalonmudeli konfiguratsiooni mugavalt ja ilma vajaliku programmeerimiskeele oskusteta.

4. Ülesande lahenduse viisid

Valikvastustega küsimustik kuulub lühiajaliste ülesannete klassi, mille abil ei koguta andmeid – selle eesmärk on dialoogivormis vestluse läbi kasutaja juhtimine selleks, et saavutada konkreetne eesmärk, saada otsus piiratud ressursi kasutamise kohta. Sellisel kujul püstitatud ülesanne klassifitseerib ülesandeks, mida on võimalik lahendada töövoosüsteemide abil.

Analüüsi käigus selgus, et häirekeskuse kutsetöötlemise ülesanne on lahendatav vähemalt kahel viisil:

- Olemasoleva universaalse THS kasutamine
- Ülesande lahendamiseks käsitsi töövoosüsteemi kirjutamine algusest peale

Valmisolev universaalne toode pakub eeldefineeritud tegevusi ja vahendeid standardsete ülesannete lahendamiseks. Toodete hulka, mis pakuvad standardiseeritud töövoogude tuge, võime lugeda suured süsteemid, näiteks *IBM Lotus Notes, GoPro* ja *Microsoft Dynamics MX (Axapta), Jboss jBPM, Intalio|BPMS* ja teised.

Kuna töövoo haldamissüsteeme on väga palju, selleks, et leida meie ülesande jaoks sobiv lahendus, on vajalik kindlaks määrata, millisesse töövoo tüüpi meie ülesanne kuulub. See on vajalik selleks, et valida ainult neid lahendusi, mis oskavad tegeleda vajaliku töövoo tüübiga ja võtta arvesse kõiki töövoo omadusi, selleks et olla lahenduse kandidaadiks. Neid, mis ei oma funktsionaalset toetust lühiajalise ülesande klassilahenduse jaoks, ei ole mõtet isegi vaatluse alla võtta, kuna nende rakenduste abil ei ole võimalik või on üpris rakse püstitatud ülesannet lahendada.

Erinevate töövoo tüüpide konstrueerimisel kasutatakse erinevaid tehnikaid ehk töövoo malle. Eksisteerib erinevaid malle [35, 36, 29], mis on jagatud mitmeteks gruppideks, kus iga grupp kirjeldab töövooge erineva perspektiiviga:

- Voo-kontrollimisperspektiiv – kirjeldab sõltuvusaspekte erinevate ülesannete vahel (paralleelsus, valik, sünkroniseerimine)
- Andmete perspektiiv – tegeleb informatsiooni edastamisega
- Ressursi perspektiiv – tegeleb ressursside jaotusega ülesannete vahel

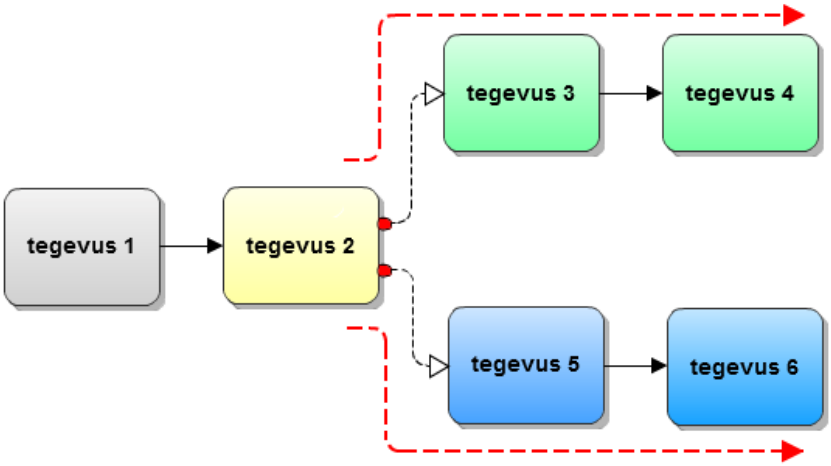
- Eranditega tegelemise perspektiiv – tegeleb erinevate erandite põhjustega ja tegevustega, mis peavad olema käivitatud juhul, kui erand on toimunud

Mallid voo-kontrollimisgrupist kirjeldavad erinevaid viise tegevuste ja voogude organiseerimiseks töövoo modelleerimisel. Andme-mallid kirjeldavad aspekte, kuidas on võimalik andmeid edastada ja milline on andmete ulatuspiirkond töövoos. Ressurss-mallid kirjeldavad selle, kuidas ressursid töövoos on esitatud ja tarbitud. Nagu oli mainitud erand-mallid tegelevad erandvoo haldamisega.

Püstitatud ülesannet sügavamalt uurides on ilmne, et lahenduse töövoo konstrueerimisel ei ole vaja kasutada malle, mis tegelevad andmetega, kuna valikvastustega küsimustiku ülesande põhieesmärk on otsuseni jõudmine ja ei ole vajalik andmeid koguda ja edastada. Samal põhjusel puudub vajadus ressurss-malle kasutada. Ülesande lahenduse töövoog samuti ei tegele eranditega, seega puudub vajadus kasutada malle, mis organiseerivad eranditega tegelemist. Ainuke mallide grupp, mis on vajalik ülesande lahenduse töövoo modelleerimisel on voo-kontrollimismallid. Antud gruppi mallide abil on võimalik modelleerida voogude kontrollimisloogikat erinevate tegevuste vahel.

Püstitatud ülesande lahenduse töövoog voo-kontrollimisperspektiivi poolt kasutab valikmalli (*choice pattern*). Kuna meie ülesande lahenduse tüüp on valikvastustega küsimustik, siis sellest järeldub, et igal küsimusel on mitu vastusevarianti. Sellest järeldub, et võimalikud kandidaadid antud gruppi hulgast on:

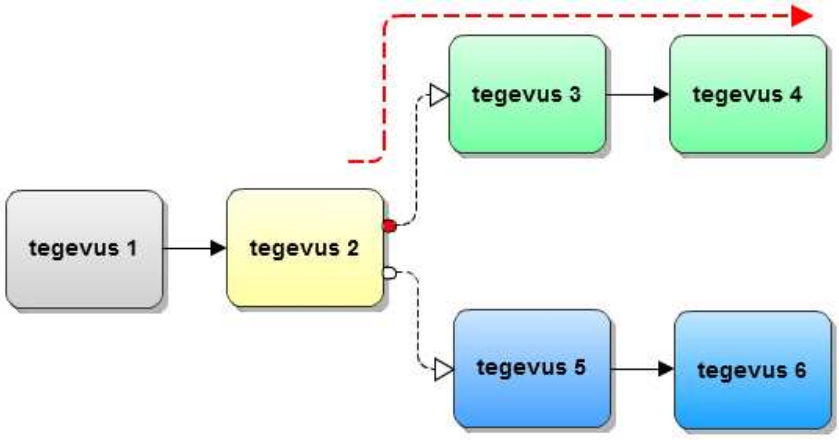
- **Multi-Choice (WCP6)** [36] – mitmevaliku mall, kus protsessi haru on jagatud mitmeteks harudeks, ning tegevuse käigus mehhanismide abil valitakse ja antakse juhtimine üle ühele või mitmele harule.



Joonis 17: Mitmevaliku mall

Joonis 17 illustreerib, et peale tegevuse 2 käivitamist, juhtimine antakse üle kas tegevusele 3 või tegevusele 5 või mõlematele.

- **Exclusive Choice (WCP4)** [36] – üksikvaliku mall, kus protsessi haru on jagatud mitmeteks harudeks, ning tegevuse käigus mehhanismide abil valitakse ja antakse juhtimine üle ainult ühele harule.

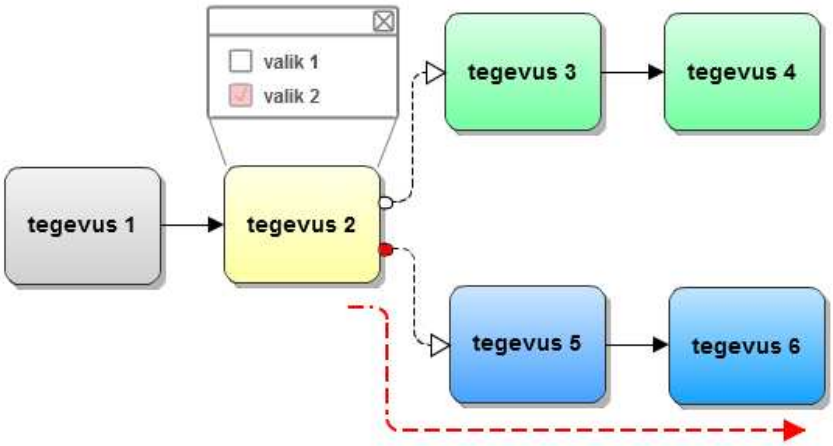


Joonis 18: Üksikvaliku mall

Joonis 18 illustreerib, et peale tegevuse 2 käivitamist, juhtimine antakse üle kas tegevusele 3 või tegevusele 5.

- **Deferred Choice (WCP16)** [36] – edasilükatud valik. Üks haru on jagatud mitmeteks harudeks, millest valitakse ainult üks, kuhu edasi minna. Valiku tegemine

edasilükatud viimase momendini, siis kui valikust sõltub järgmise tegevuse valimine. Valiku valimine toimub suhtlemise käigus väliskeskkonnaga. Kõik võimalikud harud need on võimalikud teed, kuhu on võimalik edasi minna.



Joonis 19: Edasilükatud valiku mall

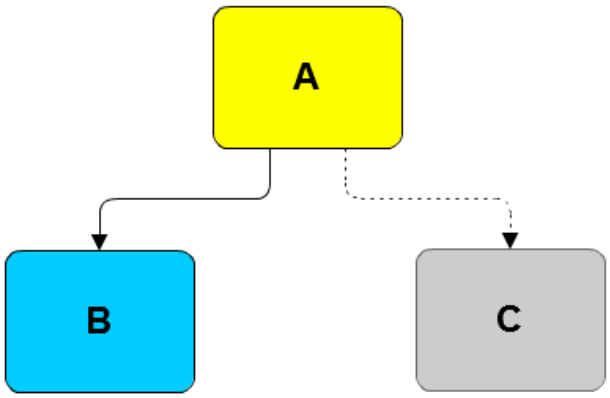
Joonis 19 illustreerib, et peale tegevuse 2 käivitamist, juhtimine antakse üle kas tegevusele 3 või tegevusele 5, baseerudes otsusele, mis on saadud väliskeskkonnast. Väliskeskkonna all antud juhul mõeldakse inimest, masina või rakendust, mis on võimeline otsust tegema.

Mitmevaliku malli (WCP6) kasutatakse lahenduste töövoogude konstrueerimisel ülesannete jaoks, kus on võimalik mitme ressursi kasutamine. Näiteks, vastavalt häirekeskuse kõne informatsioonile, kohale võib olla saadetud kiirabi, politsei, tuletõrje või kõik korraga. Püstitatud ülesande puhul on selle malli kasutamine mittevajalik, kuna ülesande püstitusest järeldub, et iga tegevuse sooritamisel peab olema valitud ainult üks järgmine tegevus. Seega ülesanne ei nõua mitmevaliku funktsionaalsust.

Üksikvaliku mall (WCP4) ei ole juba nii üleliigne nagu eelmine mall. Antud malli järgi konstrueeritakse töövooge, kus ühe tegevuse käigus valitakse ainult üks järgmine tegevus. Antud malli on võimalik kasutusele võtta selliste ülesande lahendamiseks, kus on vajalik välistatud valik. Sellest aspektist, antud mall sobib lahendamiseks valikvastuste küsimuste ülesandele. Teistest küljest, malli kirjelduses on märgitud, et valik tehakse mehhanismide abil ehk tarkvaraliselt ehk automaatselt, baseerudes olemasolevatel andmetel. Seega lõppude

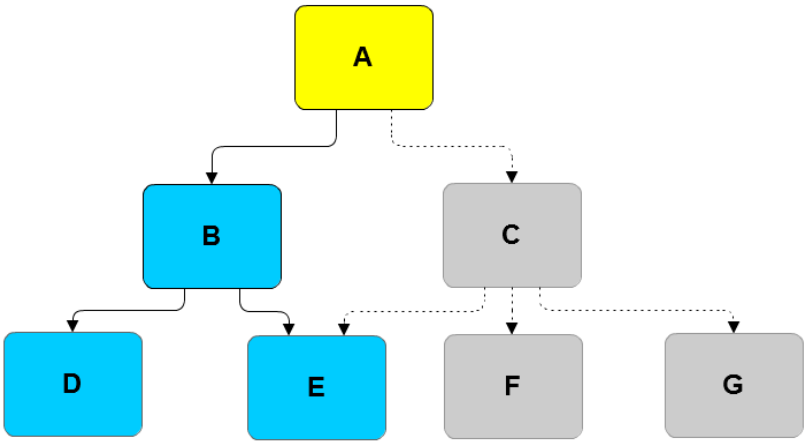
lõpuks antud mall ei sobi meie ülesande lahenduse konstrueerimiseks, kuna siin puudub interaktiivne kasutajaga suhtlemise moment.

Viimane kandidaat kirjeldab edasilükatud valiku mall (WCP16). Antud malli abil valitakse ainult üks tegevus ja selle tegevuse haru, baseerudes suhtlemisel väliskeskkonnaga. Seega antud malli kasutades eksisteerib interaktiivsuse moment: toimub väliskeskkonna mõju valikule ja seega ka teedele, kuhu liigub protsess. Iga võimalik haru, see on võimalik tee liikumiseks. Järgmise tegevuse valimisel loobutakse teistest tegevustest kui võimalikest teedest (vt. Joonis 20).



Joonis 20: Tee valimine

Tegevuse A (kollane) sooritamisel valitakse minek tegevusele B (sinine), seega ülejäänutest tegevustest (C, hall) loobutakse. Tegevusest loobumisel loobutakse ka kõikidest tegevustest, mis asuvad samas harus loobuva tegevuse all, ning mis on kättesaadavad ainult loobutud tegevust läbides. Sellest järeldub, et loobutakse ainult nendest tegevustest, millisteni ei ole võimalik jõuda peale vastava valiku valimist:



Joonis 21: Tegevustest loobumine

Joonisel 21 on näidatud, et tegevuse A (kollane) käigus valitakse tegevus B, seega antud haru on aktiivne, sest eksisteerib võimalus peale B tegevuse sooritamist jõuda nii D, kui ka E tegevusteni (sinised tegevused). C tegevus ja kõigi selle all olevate tegevusteni ei ole võimalik enam jõuda B tegevuse valimisel A tegevuse käigus, seega tegevustest C, F ja G (hallid tegevused) loobutakse. Tegevusest E ei loobuta, sest antud tegevuseni on võimalik jõuda läbi tegevuse B. Antud malli on võimalik kasutada olukordades, kus on vajalik valida ainult üks valik valikute hulgast ja teha antud valikut interaktiivselt ehk mitteautomaatselt. Näiteks inimene valib koha, kus ta tahab puhkuse ajal olla. Et kohale jõuda, valib ta rongi, auto või lennuki. Sellest valikust sõltuvad tema järgmised tegevused. Sama nagu joonisel 21 esitatud skeemi puhul, erinevate harude „ülejärgmised“ tegevused, näiteks, puhkusekindlustuse tegemine (E tegevuse puhul), võivad kattuda.

Seega, püstitatud ülesande lahenduse töövoo konstrueerimiseks on valitud mall, mis omab interaktiivset iseloomu valiku tegemiseks, mis on hädavajalik omadus valikvastustega küsimustiku konstrueerimisel.

Sellega, püstitatud ülesande analüüsimisel järeldub, et võimalikud lahenduse kandidaadid peavad toetama järgmist malli täielikult:

- **Deferred Choice (WCP16)** – edasilükatud valik.

Koostame kandidaatide loetelu ja selgitame välja, millised neist toetavad vajalikku malli. Kui kandidaat toetab antud malli ainult osaliselt, siis sellest kandidaadist automaatselt loobutakse,

sest meie eesmärk leida lahendust, mis on kiire, odav ja mugav. Mittetäiuslik nõuete toetus töövoo modelleerimisel lisab keerukust juurde.

Deffered Choice (WCP16) toetus	Kirjeldus
FLOWer (3.51)	Otseselt ei toeta antud malli, aga eksisteerivad viisid, kuidas vajadusel realiseerida.
COSA (5.1)	Otsene toetus tegevusest mitmete väljuvate kaarte abil
WS-BPEL (1.1)	Toetus <pick> konstruktsiooni abil
Websphere Integration Developer (6.0)	Toetus <pick> tegevuse abil
Oracle BPEL Process Manager (10.1.2)	Toetus <pick> konstruktsiooni abil
BPMN (1.1)	Otsene toetus tingimuslikke kaarte abil
XPDL (2.0)	Toetus XOREVENT-split konstruktsooni abil
jBPM (3.1.4)	Eksisteerib võimalus defineerida mitu ülemineku elementi, mis väljuvad *Task*-sõlmest. Käivitamise ajal üks nendest üleminekust on valitav kasutaja poolt

Tabel 2. Süsteemid ja tehnoloogiad, mis toetavad WCP16 malli (sulgudes on versiooni number)

Seoses sellega, et *FLOWer* süsteem ei toeta otseselt WCP16 malli (eksisteerib võimalus objektide kombineerimisel samasugust tulemust saada), siis antud süsteem ei sobi võimalikke kandidaatide hulka oma lahenduse konstrueerimise keerulisuse tõttu.

Kandidaadid, mis esmavaatlusel on võimelised püstitatud ülesande jaoks lahenduse töövoogu modelleerima, on:

- *BPMN* – ehk *Business Process Modeling Notation* pakub standardiseeritud graafilist tähistust äriprotsesside töövoogude modelleerimisel [34].
- *XPDL* - formaat, mis on standardiseeritud Workflow Management Coalition (WfMC) poolt, selleks, et vahetada töövoo protsesside definitsioone erinevate töövoosüsteemide vahel. Antud formaat baseerub XML keelel [45].
- *WS-BPEL* (*Web Services Business Process Execution Language*) või lihtsalt *BPEL* - on standardiseeritud käivitamiskeel veebiteenustega suhtlemiseks, mille abil on

võimalik kirjeldada ja käivitada äriprotsesse. Protsessid, mis on kirjeldatud BPEL keele abil, suhtlevad välisressurssidega (programm, inimene) kasutades veebiteenuseid.

- *COSA* – universaalne äriprotsesside haldamissüsteem.
- *WID* (*Websphere Integration Developer*) – IBM'i poolt arendatav BPEL tehnoloogial baseeruv rakendus, mille abil on võimalik modelleerida töövooge ning genereerida nendest BPEL formaadis töövoo definitsioonide käivitamisfaile. Rakendus baseerub SOA'l (*Service-Oriented Architecture*)
- *Oracle BPEL Process Manager* – rakendus, mis võimaldab mugavalt ja kiiresti luua, juurutada ja hallata äriprotsesse, mis on realiseeritud BPEL keele abil.
- *jBPM* – on täielik töövoo haldamissüsteem, mille abil on võimalik nii luua töövoo definitsioone, kui ka neid käivitada ja hallata. Kasutamisele võetud uus BPMN 2.0 notatsiooni standard.

Kandidaatide hulgas on kommertsproduktid-täissüsteemid, kui ka hulk standardeid ja tehnoloogiaid (BPMN, XPDL, WS-BPEL), mille abil on võimalik püstitatud ülesande jaoks lahenduse töövoo diagrammi konstrueerida. Antud kandidaatide kogum pole kindlasti täielik: siia võib lisada süsteeme ja üksikuid rakendusi (nii kommerts- kui ka vabavaralisi), mis täielikult toetavad toodud tehnoloogiaid, seega on võimelised looma sobilikku lahendust.

Muidugi, kõige lihtsam viis lahendada püstitatud häirekeskuse ülesannet, on luua algusest peale püsiprogrammeeritud (*hard-coded*) süsteem. Selle süsteemi tüübi kirjeldus on toodud peatükis „1.4 Püsiprogrammeeritud süsteemid".

Püstitatud ülesande lahenduse kandidaadid on võimalik jagada kaheks grupiks:

1. Kommerts/vabavaralised täissüsteemid, mis töötavad eespool mainitud tehnoloogiate baasil (BPMN, XPDL, WS-BPEL). Antud universaalsed süsteemid ei ole spetsialiseerutud püstitatud ülesande lahenduse konstrueerimiseks
2. Nullist kirjutatud süsteem, kus ülesande lahenduse töövoo loogika on sisseehitatud nö. püsiprogrammeeritud süsteem.

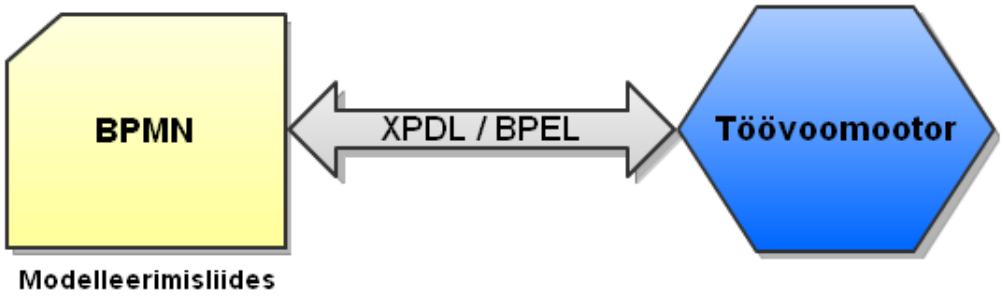
Esimesse gruppi kuuluvad kandidaadid, mis toetavad BPMN, XPDL, BPEL standardeid, seega kandidaatideks võib olla kaks süsteemi: BPMN + XPDL ja BPMN + BPEL toetusega.

Selliste kombinatsioonide katsetamine annab ülevaate, kuidas need süsteemid (ja ülejäänud rakendused, mis samu standardeid toetavad) võivad abiks olla valikvastuste ülesande lahendamisel. Seega vaatluse all on *COSA BPM Suite*, *Enhydra Workflow System*[33] ja *Intalio|BPMS*. Süsteemid, näiteks, *Oracle BPEL Process Manager*, *IBM Websphere*, *ActiveVOS, jBPM* süsteem kasutavad äriprotsesside modelleerimiseks ja käivitamiseks BPMN (või oma notatsioon, mis on ühildatav BPEL-iga) ja BPEL tehnoloogiaid, seega ei ole mõtet neid lisada kandidaatide listi, kuna püstitatud ülesande lahenduse idee on nendel süsteemidel sarnane näiteks *Intalio|BPMS* süsteemiga, mis on juba kandidaatide listis olemas. Peale esimese kandidaatide grupi analüüsi tulevad teise gruppi püsiprogrammeeritud süsteemi analüüs.

Enne kandidaatide proovimist, peab testimismetoodika olema paikapandud ja kandidaatide analüüsimisel kasutusele võetud. Testimisreeglid on kirjeldatud peatükis „4.2 Testimismetoodika“. See annab kõigile kandidaatidele võrdsed tingimused.

4.1 Tehnoloogiad: BPMN, XPDL, BPEL

Üldiselt kandidaadid vabavara- ja kommertssüsteemide hulgast toetavad kas BPMN [34] ja XPDL [45] või BPMN ja BPEL selleks, et salvestada töövoo definitsioone edasiseks käivitamiseks töövoomootoris, mis toetab sisendit XPDL või BPEL kujul (vt. Joonis 32).



Joonis 22: Standardid

Komponendid ei pea olema ühe tarkvara paketi omad või ühe tootja omad: Modelleerimisrakendus ja töövoomootor võivad olla erinevatest pakettidest või üldse iseseisvad rakendused. Standardid ühendavad erinevaid töövoosüsteemi komponente.

Modelleerimisliidese jaoks on BPMN (või muu notatsioon, mis otseselt toetab WCP16 malli) ja protsessi käivitamiseks on XPDL või BPEL keeled.

WS-BPEL (*Web Services Business Process Execution Language* või lühidalt BPEL) on SOA (*Service-oriented Architecture*) standardiseeritud XML keel äriprotsesside ja töövoo diagrammide defineerimiseks, käivitamiseks ja suhtlemiseks väliskeskkonnaga. Keel ei toeta graafilist notatsiooni töövoo diagrammide jaoks. BPEL abil töövoo protsessi modelleerimisel seotakse erinevad veebiteenuseid eesmärgi saavutamiseks.

Paljude modelleerimisliideste väljund võib olla nii BPEL kui ka XPDL formaadis. Mis vahe nende kahte standarte vahel on? XPDL on loodud äriprotsesside definitsioonide vahetamiseks, formaat kirjeldab ära nii graafilist kui ka töövoo protsessi semantilist perspektiivi. XPDL formaat on loodud spetsiaalselt selleks, et toetada kõiki BPMN aspekte. XPDL omab elemente diagrammi sõlmede graafilise informatsiooni hoidmiseks, mille hulgas nii sõlme x-positsioon kui ka y-positsioon. Samas formaat salvestab käivitamisaspekte, mis on kasutatavad defineeritud tööprotsessi käivitamiseks. Antud omadus eristab XPDL formaati BPEL formaadist, mis on fokuseeritud äriprotsessi käivitamisaspektide peale. BPEL ei oma elemente, mis hoiaksid enda sees graafilist informatsiooni sõlmede kohta.

Kuna püstitatud ülesande jaoks põhiline asi ongi protsessi modelleerimine ja siis käivitamine, ei oma tähtsust XPDL graafiliste elementide toetuse olemasolu. Tähtis aspekt: kui täpselt XPDL ja BPEL formaadid kirjeldavad modelleeritud töövoo diagrammi käivitamisaspekte. Kuna XPDL on välja töötatud selleks, et toetada BPMN ja kuna viimane toetab püstitatud ülesande lahenduseks vajaliku malli (WCP16), siis antud tehnoloogiate paar on võimeline antud ülesannet lahendada. BPEL ei toeta otseselt mõnda BPMN konstruktsooni, näiteks tema otseselt ei toeta, WCP43[36] malli: BPEL'i seisukohast protsess lõpetab oma töö siis, kui ei ole rohkem tegevusi tegemiseks. Lisaks BPEL ei toeta kasutaja teavitamisoperatsioone. Vaatamata sellele eksisteerivad viisid, kuidas transformeerida BPMN notatsiooni abil tehtud diagramme BPEL formaati [37].

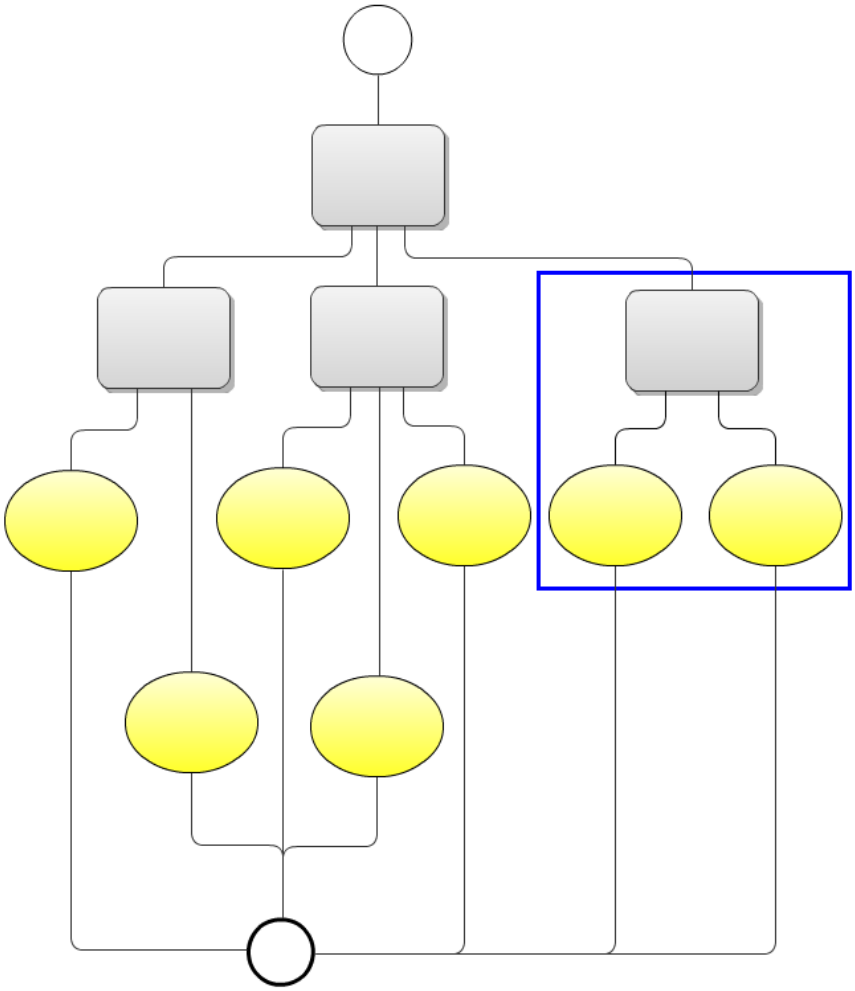
Kokkuvõtteks, BPEL keel defineerib äriprotsesside käitumist, kus iga tegevus on veebiteenus. Inimesega koostoime toetust ei eksisteeri. Vaatamata sellele, et veebiteenuse kasutamine on väga lai levinud, inimesega koostoime toetuse puudumine on suur puudus paljude kaasaegsete äriprotsesside jaoks. Selle probleemi parandamiseks loodi WS-BPEL 2.0 jaoks laiendusi

nimega *BPEL4People*[32] ja *WS-HumanTask* [38]. Antud laiendused aitavad BPEL'il juhendada mitte ainult veebiteenuseid aga ka rollipõhiseid inimtegevusi. Seoses sellega töövoomootor, mis toetab BPEL keeles töövoo definitsioone peab toetama ka BPEL4People ja WS-HumanTask laiendust (nt. *Intalio|BPMS 6* ja *ActiveVOS 7*), sest valikvastustega küsimustiku ülesanne nõuab lõppkasutajaga interaktiivset suhtlemist.

4.2 Testimismetoodika

Selleks, et teha kindlaks, kas konkreetne kandidaat sobib püstitatud ülesande lahendamiseks ja kui mugavalt ta lahendab, tuleb teda katsetada. Iga kandidaati testitakse, kas ta saab olla püstitatud valikvastuste küsimustiku ülesande lahenduseks või mitte. Lisaks analüüsitakse lahenduse töövoogu loomise raskust ja küsimustiku läbimise lihtsust ja mugavust.

Valikvastustega küsimustik on iseenesest triviaalne ülesanne. See on korduv küsimuste esitamine, kus küsimus omab piiratud arv vastusi, ning küsimuse vastusest sõltub järgmise küsimuse esitamine. See tähendab seda, et küsimustiku töövoog on graafikujuline, kus algküsimus, millest küsitlus algab, on juur, lehed on lõppotsused ja iga küsimuse alluv on järgmine võimalik küsimus, mille esitamine sõltub eelmise küsimuse vastuse valikust vastuste hulgast.



Joonis 23: Valikvastustega küsimustiku diagramm

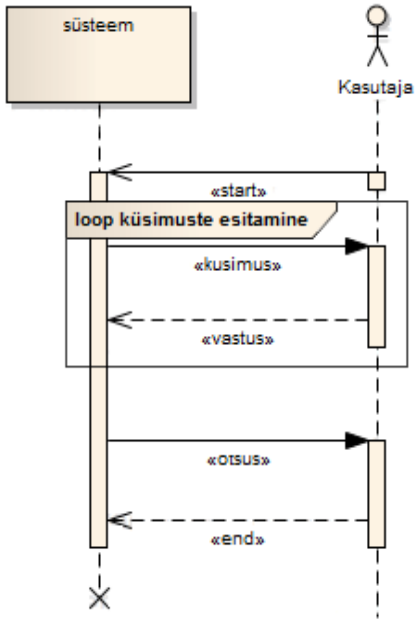
Joonisel 23 on illustreeritud suvalise küsimustiku töövoo diagramm, kus hallid ristkülikud tähendavad tegevusi, antud juhul küsimuse esitamise tegevusi, kollased ovaalid tähendavad otsuse kättesaamise tegevusi, ülemine ring tähendab küsimustiku protsessi algust ja alumine ring tähendab küsimustiku protsessi lõppu. Küsimuste esitamise tegevused koos otsuse kättesaamisega moodustavad küsimustiku **keha**. Seetõttu küsimustiku diagramm koosneb kolmest osast: algus, keha ja lõpp.

Küsimuste esitamise korduva iseloomu tõttu on võimalus eraldada antud diagrammist neid atomaarseid elemente, millest küsimustiku töövoo diagramm koosneb. Joonisel 23 on märgistatud antud atomaarsete elementide kogum sinise ristkülikuga. Need diagrammi elemendid on tegevus (küsimuse esitamise tegevus ja otsuse kättesaamise tegevus) ja tegevusest väljuvad kaared (küsimuse vastused). Sellise atomaarsete elementide

konstrueerimise abil on võimalik koostada küsimustiku keha. Atomaarsete elementide hulka on tarvis lisada ka protsessi alg- ja lõppelement. Sellega küsimustiku töövoo diagrammi on võimalik konstrueerida kasutades järgmisi atomaarseid elemente:

- Algelement – töövoodiagrammi algtegevus
- Küsimuse esitamine – töövoodiagrammi tegevus
- Küsimuse vastus – töövoodiagrammi kaar, mis ühendab kahte tegevust
- Otsuse kättesaamine – töövoodiagrammi tegevus
- Lõppelement – töövoodiagrammi lõpptegevus

Joonis 24 illustreerib küsimustiku läbimise protsessi käigus andmevahetust kasutaja ja süsteemi vahel.

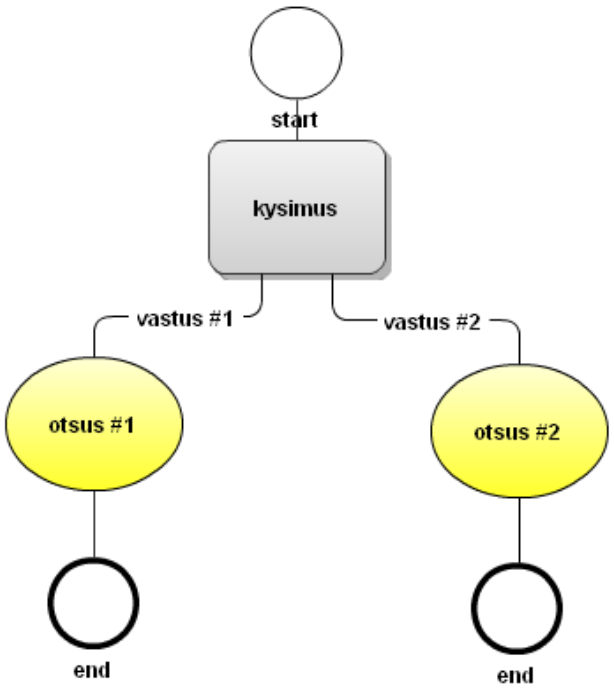


Joonis 24: Andmevahetus kasutaja ja süsteemi vahel

Andmevahetust on võimalik jagada neljaks osaks: küsimustiku alustamine, korduv küsimuste esitamine ja vastamine, mille käigus läbitakse eeldefineeritud töövoog, otsuse esitamine ja küsimustiku lõpp. On selge, et kuna korduv osa ehk. küsimuste esitamine ja vastuste põhjal järgmise küsimuse valimine, on triviaalne ja rutiine tegevuste kogum, on piisav sellest, et töövoog, mis on võetud kandidaatide testimiseks, sisaldab ühte küsimust, kahte vastust ja otsust. Ülejäänud tegevused jäävad alles nii nagu on, sest sõltumata küsimustukus olevate

küsimuste arvule, kogu protsessi alustamismehhanism ja protsessi lõpetamismehhanism on kõikidel võimalikel küsimustikel sarnane.

Kandidaatide testimiseks on vajalik luua test-töövoo diagramm, mida proovitakse koostada ja käivitada kasutades kandidaatide poolt pakutavaid vahendeid. Test-töövoog peab olema maksimaalselt väike, samal ajal peab sisaldama kõik atomaarseid elementide (vt. Joonis 25).



Joonis 25: Test-töövoo diagramm

Joonisel 25 on illustreeritud test-diagramm, mida proovitakse moodustada ja käivitada kasutades kandidaatide vahendeid. Antud diagramm sisaldab kõiki atomaarseid elemente ning antud töövoog võib olla reaalse lihtsa valikvastustega küsimustiku ülesande lahenduseks. Näiteks ülesanne: saada informatsiooni, ka inimene on taimetoitlane või mitte. Võimalik lahendus on selline: küsimusele „Kas Te sööte liha?“ vastuste variandid on kas „Jah“ või „Ei“, vastavad otsused on „Te olete taimetoitlane“ ja „Te ei ole taimetoitlane“. Töövoo protsess on seotud ainult ühe rolliga, sest küsimuste vastamistega tegeleb korraga üks inimene, isegi siis, kui tegu on häirekeskusega. Dispetšer telefoni kaudu edastab küsimusi, saab helistajalt vastuse kätte ning fikseerib vastuse ja kinnitab otsuse koos vajalikke tegevustega, mida kinnitatud otsus nõuab.

4.3 Kandidaat: COSA BPM Suite

COSA BPM (Business Process Management) Suite on suur kommertssüsteem, mis pakub vahendeid äriprotsesside modelleerimiseks, käivitamiseks ja optimeerimiseks. Üks süsteemi komponentidest on *COSA Process Designer* – see on modelleerimisvahend äriprotsessi definitsioonide jaoks. Antud rakendus kasutab diagrammide jaoks BPMN notatsiooni ning XPDL standardit modelleeritud töövoogude salvestamiseks.

Alates 1990-ndate algusest oli COSA üks esimestest, kes hakkas tegelema intelligentsete töövoosüsteemidega, äriprotsesside haldamissüsteemidega ja dokumendi haldussüsteemidega.

COSA BPM Suite on täielik äriprotsesside haldamisrakenduste komplekt, mis tegeleb kõikide äriprotsesside haldamisfaasidega:

- Protsesside modelleerimine ja dokumenteerimine
- Protsesside simuleerimine
- Protsesside käivitamine ja integreerimine
- Protsesside monitooring ja kontroll
- Protsesside optimeerimine
- Täielik dokumendihaldussüsteem on süsteemi integreeritud komponent

COSA Process Designer baseerub Petri võrkudel (*Petri Nets*) on üks süsteemi komponent, mille abil kasutaja ilma sügavate tehniliste teadmistega saab konstrueerida töövooge, siduda rolle ja vastavusi. Protsesside graafiliseks modelleerimiseks kasutatakse standardiseeritud BPMN notatsiooni. Modelleeritud protsess salvestatakse standardiseeritud XPDL formaadis. Antud formaat toetab BPMN elemente täielikult. Seega, selle formaadi kasutamine annab võimaluse vahetada modelleeritud töövoo protsesse erinevate töövoosüsteemide vahel. standardiseeritud formaatide kasutamine annab võimaluse kasutada protsessi

modelleerimisvahendi ühest süsteemist ja töövoomootorit antud protsesside käivitamiseks – teisest täissüsteemist.

COSA Protsess Simulator annab võimalust simuleerida defineeritud protsessi enne reaalset käivitamist selleks, et leida ja parandada vigu ja ebaoptimaalseid kohti.

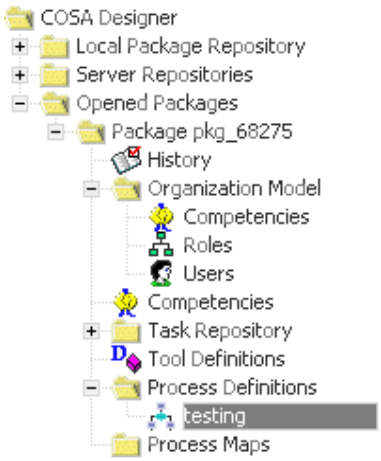
COSA Engine on töövoomootor, mille abil on võimalik käivitada COSA Protsess Designeri (või muu modelleerimisrakenduse) abil defineeritud töövoogu XPDL formaadis. Mootor pakub käivitamiskeskkonda protsesside jaoks ning on vastutav selle keskkonna juhtimismehhanismide eest.

Kuna COSA on üks WfMC'i (*Workflow Management Coalition*) rajajast, siis antud süsteem vastab täielikult etalonmudelile, mis on kirjeldatud WfMC poolt koostatud dokumendis [30].

Püstitatud ülesande nõuete analüüsi käigus on COSA BPM Suite üks kandidaatidest, mille abil on võimalik lahendust luua. COSA töövoode toimetaja on üks vähestest, kes otseselt ja täielikult toetab põhilisi voogude kontrollimismalle, seal hulgas edasilükatud valiku mall (*Deffered Choice* (WCP16)), mis on vajalik valikvastuste küsimustiku modelleerimiseks [46]. COSA toetab malli WCP16 mitmete tegevusest väljuvate kaarte abil. Antud malli kasutatakse situatsioonides, kus otsust tehakse mitte automaatselt süsteemi poolt, vaid lõppkasutaja poole pöördudes. Antud omadus ongi vajalik valikvastuste küsimustiku ülesannet lahendatava töövoo modelleerimisel.

COSA on kommertstarkvara ja tootja ei paku tasuta versiooni allalaadimiseks. Selle asemel COSA ametlikul veebilehel olemas interaktiivne *COSA Suite Demo* [47] rakendus, mille abil on võimalik töövoo definitsioone defineerida kasutades BPMN notatsiooni ja töövoo tööd simuleerida. Seoses sellega on võimalik eelmises peatükis moodustatud test-töövoogu modelleerida kasutades ainult COSA testimisversiooni. COSA BPM Process Designer on *Java* keele abil kirjutatud *applet* vormis töövoogude modelleerimisrakendus.

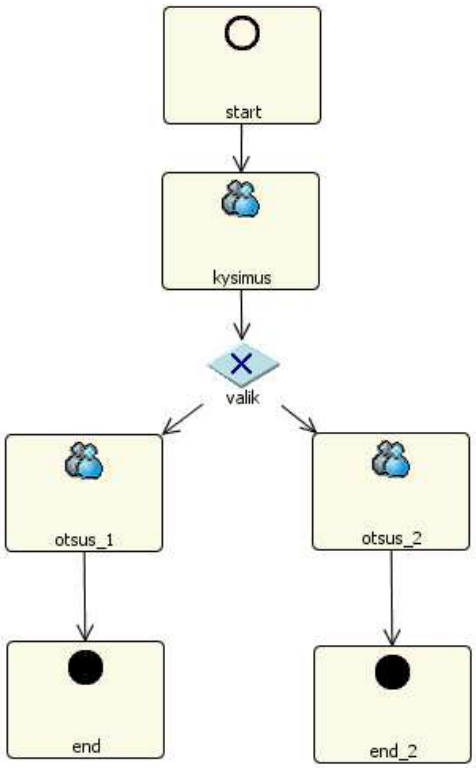
Seoses sellega, et antud universaalne rakendus võimaldab töövooga ja töövoo tegevustega siduda konkreetset rolli, kasutajat ja sätestada pädevusi – rakenduse vasakpoolne paneel sisaldab esmapilgul päris keerulist struktuuri (vt. Joonis 26).



Joonis 26: COSA Process Designer'i paneel

Lähemalt uurides, kõik on hästi struktureeritud ja organiseeritud. Valikvastuste küsimustiku koostamise jaoks nii rollide, kui ka pädevuste ja teiste süsteemi kohandamised ja sätted on üleliigsed.

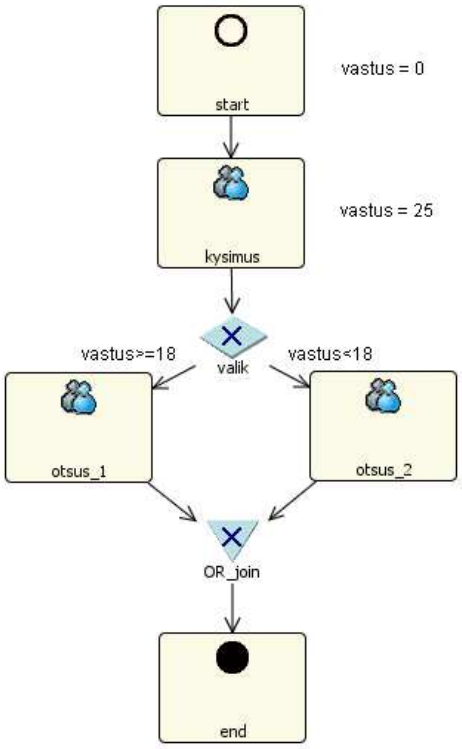
Modelleerimisrakenduse peapaneel sisaldab tausta diagrammide joonistamiseks, BPMN objekte ja objektide parameetrite paneeli. Modelleeritud töövoog on esitatud järgmisel joonisel:



Joonis 27: COSA toimetaja abil modelleeritud test-töövoo diagramm

Joonisel 27 oleva diagrammi koostamiseks pakkus rakendus kõiki vajalikke elemente, seega nõutud töövoo diagramm on sarnane test-töövoo diagrammiga. Loodud diagramm koosneb algtegevusest, millest töövoo protsess algab, tegevusest nimega „kysimus“, mille käigus esitatakse küsimust ja saadakse vastus, „valik“ marsruutimistegevus ehk värav, mille käigus baseerudes kasutajalt saadud vastusele valitakse, millist haru on vaja aktiveerida ja millist töövoo osa ignoreerida. Tegevused „otsus_1“ ja „otsus_2“ on võimalikud töövoo protsessi käigus saadavad otsused ehk küsimustiku resultaadid. Seoses sellega, et „valik“ värav on *OR*-tüüpi valiku mehhanism, ainult üks kahest otsusest on aktiivne töövoo läbimise lõpus. Saadud aktiivne otsus fikseeritakse ja näidatakse kasutajale.

Töövoo tegevuste loogika koostamisel (küsimused, vastused jne.) selgus, et iga tingimusliku ülemineku (mis väljub OR-väravast) jaoks on vajalik tingimusliku avaldise koostamine. Antud avaldis on *boolean* tüüpi, mille arvutades töövoomootor otsustab, kas aktiveerida haru, kuhu suunab antud üleminek või mitte. Küsimuse esitamisel, mida tehakse kohe enne tingimusliku väravat, salvestatakse vastus muutujasse, mis on kättesaadav ka järgmistes tegevustes. Tingimusliku ülemineku avaldis kontrollib antud muutuja väärtust ja saadud *boolean* vastus näitab, kas saab antud haru kasutada või mitte. Näiteks, küsimusele „Kui vana sa oled?“ pakutud vastused on „>=18“ ja „<18“. Pealse küsimuse esitamist kasutajalt saadud vastust salvestatakse muutujasse nimega `„vastus“`. Tänu sellele üks haru jagatakse *OR* tingimusliku värava abil kaheks haruks, mille tingimuslikud üleminekud omavad vastavaid tingimusikke väljendeid: `(vastus >= 18)` ja `(vastus<18)`. Antud väljendute arvutades töövoomootor valib, millist haru tuleb aktiveerida ja millest loobuda.



Joonis 28: COSA toimetaja abil modelleeritud edasi arendatud test-töövoo diagramm

Joonisel 28 esitatud diagramm on varustatud muutuja väärtustamis- ja kontrollimisloogikaga, mida tehakse töövoo protsessi käigus. *OR-join* marsruutimistegevus ühendab kahte haru kokku *OR* tingimuse abil, mis tähendab seda, et piisab ainult ühest aktiivsest sisendharust, et protsessi edasi suunata lõpptegevusele.

Antud viis töövoo protsessi modelleerimiseks, kus iga küsimuse ja vastuse jaoks on vajalik luua tingimuslik avaldis, ei ole kõige efektiivsem viis küsimustiku loomiseks, sest palju lihtsam oleks see, kui süsteem genereeriks neid tingimusi ise, baseerudes eeldefineeritud vastustel. See nõuab, et inimene on tuttav avaldis-keelega ning inimesel on piisav tehniline kogemus. Seoses sellega, püstitatud ülesande lahendamiseks vajaliku töövoo diagrammi koostamine võtab päris palju aega.

Valikvastuste küsimustik nõuab interaktiivset suhtlemist vastajaga. Töövoo simuleerimine toimub *COSA Simulator* rakenduse abil. Loodud töövoo diagrammi käivitamine ei olnud võimalik, kuna rakenduse demoversiooni tõttu ei ole võimalust COSA töövoomootorit katsetada.

Kokkuvõtteks, *COSA BPM Suite* pakub võimsat modelleerimisrakendust, mis kasutab BPMN ja XPDL standardeid töövoo diagrammide koostamiseks ja salvestamiseks. Aga kuna antud universaalne rakendus on mõeldud paljude erinevate ülesannete lahendamiseks, tundub et triviaalse valikvastuste küsimustiku koostamine ei ole nii lihtne ja efektiivne, sest nõuab küsimustiku loojalt tehniliste oskuste omandamist. Lisaks võtab töövoo diagrammi konstrueerimine suhteliselt palju aega. Seoses sellega, et otsitav lahendus peab olema lihtne ja odav, antud universaalne süsteem tundub koormav ja seega mitte sobilik.

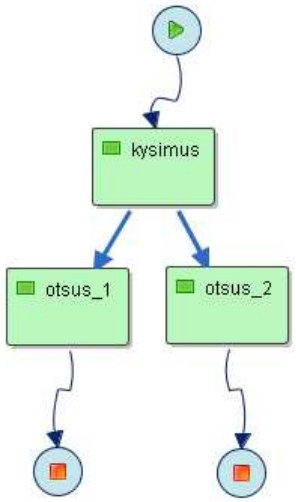
4.4 Kandidaat: Enhydra Workflow System

Vabavaraline avaliku koodiga (GPL V3 litsents) töövoomootori ja modelleerimisliidese raamistik sisaldab standardse töövoosüsteemi rakendust, mis täielikult baseerub WfMC ja OMG spetsifikatsioonidel [30, 34, 47] ja kasutab XPDL keelt töövoo protsesside defineerimiseks.

Antud süsteem koosneb kahest komponendist:

- *Together XPDL Workflow Server* – teine nimetus on *Endyra Shark.* Antud rakendus on paindlik ja laiendatav WfMC XPDL ja OMG spetsifikatsioonidel põhinev töövoo sõltumatu Java haldamisrakendus. See sisaldab administreerimisliidest, töövoo definitsioonide haldamiseks. XPDL formaadis töövoo definitsioone on võimalik luua kasutades *Endyra JaWE* modelleerimisliidest.
- *Together XPDL Workflow Editor* – teine nimetus on *Endyra JaWE,* on graafiline diagrammidel põhinev töövoo definitsioonide modelleerimisliides. Täielikult rakendatud WfMC XPDL spetsifikatsioon.

Töövoo modelleerimisel pakub rakendus hulga vahendeid diagrammi loomiseks, kaasa arvatud vahendeid rollide ja pädevuste sätestamiseks. Negatiivseks momendiks on see, et modelleerimisliides ei toeta BPMN standardit täielikult: puuduvad marsruutimistegevuste elemendid. WCP16 mall ehk interaktiivne otsuse tegemine on realiseeritav tegevusest kahe ja rohkema väljuva ülemineku (*kaare*) abil.



Joonis 29: Endyra JaWE abil modelleeritud test-töövoo diagramm

Joonisel 29 on illustreeritud test-töövoo diagrammi põhjal realiseeritud töövoo diagramm, kasutades Endyra JaWE modelleerimisliidest. Nagu näha töövoog on konstrueeritud kasutamata OR-väravaid (mis on BPMN standardi element). Antud „puudus“ ei sega, kuna töövoo defineerimisprotsess on loogiline ja ei häiri ega aeglusta tööd.

Tagaplaanil töötab sarnane muutujate abil informatsioonide vahetus nagu see oli COSA modelleerimisliidesel. Muutujaid defineeritakse XPDL laiendatud atribuutide abil. Samasugune ebamugavus on seotud üleminekute loogiliste väljendite defineerimisega, mis juba ilmus COSA Process Designer’iga töötamisel – iga üleminek tuleb käsitsi varustada tingimusliku avaldisega selleks, et töövoomootor oskaks otsustada, millist haru edasiseks tööks valida. Neid loogilisi avaldisi on võimalik konstrueerida kolmes erinevas keeles: Java, JavaScritpt ja Python. XPDL faili alguses olemas märgend nimega `script`, mis võib omada üks järgmistest väärtustest:

- `text/java` – tingimuslike üleminekute keel on Java
- `text/javascript` - tingimuslike üleminekute keel on JavaScript
- `text/pythonscript` - tingimuslike üleminekute keel on Python

Ja niimoodi antud märgend näeb välja XPDL-töövoo failis:

```
<xpdl:Script Type="text/javascript"/>
```

Antud omadus tuleneb rakenduse laiast vaatenurgast. Spetsiifiliste ülesannete puhul on süsteemi kohandamine päris tülikas ja mittetehniliste oskustega inimestele on see raske. Suure (20 küsimust) küsimustiku töövoo diagrammi koostamine võtab päris palju aega. Ja isegi lihtsa küsimustiku loomiseks või muutmiseks on nõutud tehnilisi oskusi vähemalt üleminekute tingimuste avaldiste koostamiseks.

Loodud töövoo käivitamine on loogiline ja mugav. On võimalik eristada kuut etappi:

1. Töövoo definitsiooni paketi üleslaadimine töövoomootori serveri repositooriumi
2. Serveril uue kasutaja profiili tegemine, kus kasutaja näeb temale määratud ülesandeid. Vastasel juhul töövoomootor paneb kõik ülesandeid töövoomootori administraatori tööloetellu (*Work List*)
3. Loodud töövoomootori kasutaja sidumine osavõtjaga, mis on määratud diagrammis. Kuna iga töövoo tegevus on seotud konkreetse osavõtjaga, siis töövoomootori registris peab olema kasutaja, kellega on seotud antud töövoo osavõtja. Sel juhul töövoomootor töövoo protsessi käigus teab kellele saata antud tegevus. Püstitatud ülesande puhul on kõikide interaktiivsete tegevustega seotud ainult üks osavõtja.
4. Töövoo definitsiooni protsessi isendi loomine ja käivitamine (on võimalik luua ja käivitada mitu iseseisvat isendit ühest töövoo definitsioonist)
5. Protsessi käivitamise ajal esitatakse küsimusi sel moel, et töövoog paneb tegevusega seotud kasutaja tööloetellu ülesandeid, mis on iseenesest küsimused, millele tuleb vastata.
6. Kasutaja näeb oma tööloetelus uut ülesannet, avab selle, ja valib esitatud küsimuse vastust. Salvestades, vorm saadab töövoomootorile tagasi esitatud küsimuse vastuse. Pealse seda, töövoomootor salvestab vastuse ja vastavalt sellele leiab uue tegevuse (küsimuse esitamise tegevus) töövoo diagrammist ja saadab kasutajale.

Antud süsteemi tehnilise aspekti uurides ilmus, et valikvastused on realiseeritavad kasutades XPDL `EnumerationType` andmetüübi abil. Töövoo protsessi tagaplaanil salvestatakse kasutaja valik muutujasse (test-töövoo muutujanimi on „`whereToGo_1`“), mis edastatakse järgmisele tegevusele. Töövoomootor teeb selle muutuja väärtuse põhjal otsuse, kuhu edasi liikuda ja milline on järgmine tegevus. Antud juhul muutuja `whereToGo_1` on `EnumerationType` tüüpi muutja ja hoiab endas kahte väärtust: „Jah“ ja „Ei“. Need on valikvastused, mida pakutakse kasutajale. Töövoo XPDL faili osa välja näeb selliselt:

```
<xpdl:DataField Id="whereToGo_1" IsArray="FALSE" Name="Kas saata kiirabi
kohale? (jah/ei)">
    <xpdl:DataType>
        <xpdl:EnumerationType>
            <xpdl:EnumerationValue Name="jah"/>
            <xpdl:EnumerationValue Name="ei"/>
        </xpdl:EnumerationType>
    </xpdl:DataType>
    <xpdl:InitialValue>jah</xpdl:InitialValue>
</xpdl:DataField>
```

Peale küsimusele vastamist hoiab antud muutuja endas valitud vastust. Aga selgus, et *Endyra* töövoomootor kahjuks ei toeta andud XPDL andmetüüpi, töövoo definitsiooni üleslaadimisel töövoomootorisse informeeritakse, et toetus eksisteerib ainult mõnedele XPDL andmetüüpidele (*BOOLEAN*, *STRING*, *INTEGER*, *FLOAT*, *DATETIME*):

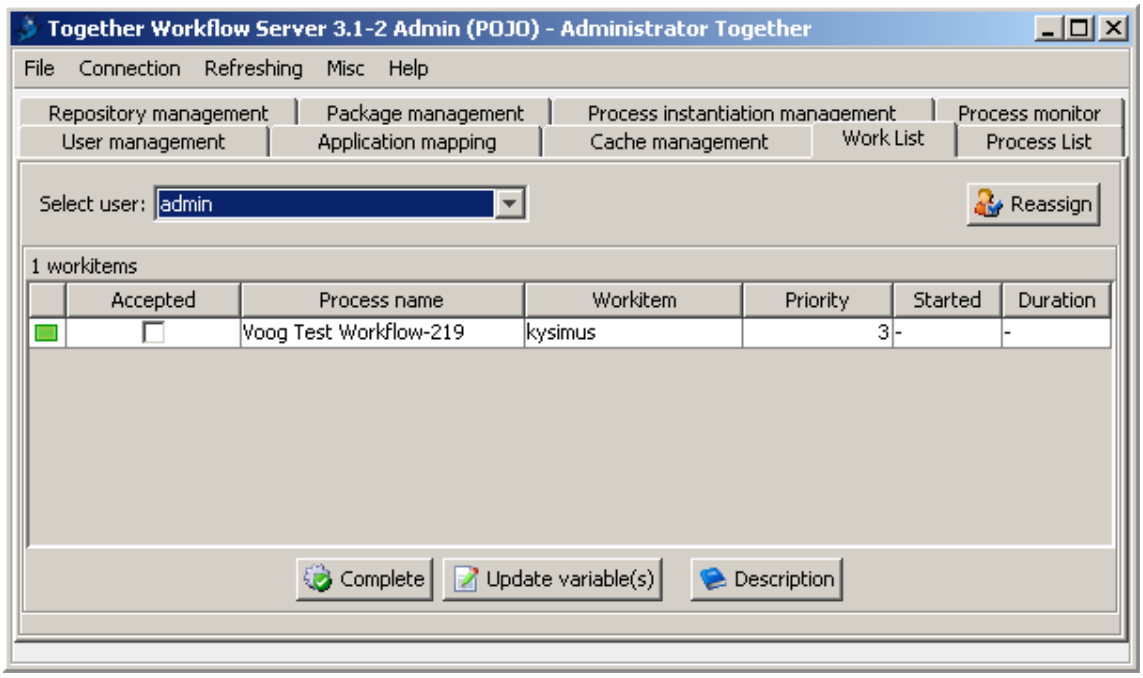
„*Type=ERROR, Sub-Type=LOGIC, Location=Package 'voog_pkg', DataField 'whereToGo_1' -> EnumerationType:*
Data type is not supported (currently supported types are ExternalReference and basic types:BOOLEAN, STRING, INTEGER, FLOAT, DATETIME, as well as their type declaration versions): EnumerationType“

Seega, ainuke võimalus valikute realiseerimiseks on, kas teha muutuja tüüpi *boolean*, kus kasutajale pakutakse valida kas *true* või *false* või *String* tüüpi, kus kasutajale pakutakse sisestada vastust vabatekstina. Järgmine koodilõik on osa XPDL failist:

```
<xpdl:DataField Id="whereToGo_1" IsArray="FALSE" Name="Kas saata kiirabi
kohale? (jah/ei)">
    <xpdl:DataType>
        <xpdl:BasicType Type="BOOLEAN"/>
    </xpdl:DataType>
    <xpdl:InitialValue>true</xpdl:InitialValue>
</xpdl:DataField>
```

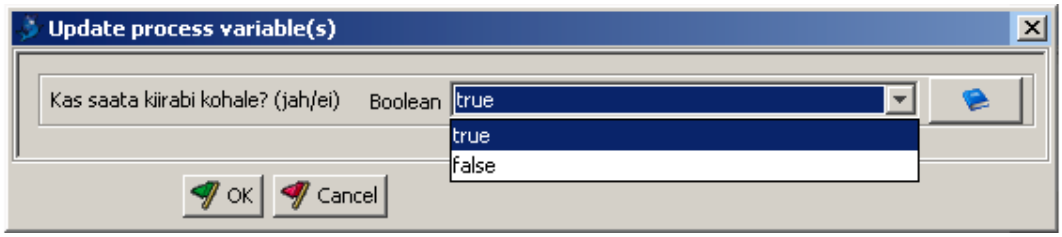
Viimane variant ei ole hea, sest töövoo loogika peab sisaldama palju vabateksti sisendi kontrollimist (*data validation*), et vältida töövoo protsessi ebakorrektset tööd [19]. Kasutades muutujat tüüpi *boolean*, eksisteerib suur ebamugavus, sest antud andmetüüpi abil on võimalik

defineerida ainult kahte väärtust: *true* ja *false*. Seega antud tüüp sobib ainult nendele küsimustele, mis omavad kahte eeldefineeritud vastust „Jah“ ja „Ei“ (või muud vastused, mida on võimalik siduda `true` ja `false` väärtustega). Kui küsimus nõuab kolme või rohkemat vastusevarianti, siis antud andmetüüp ei sobi. Seega on võimalik luua küsimustiku, kus iga küsimus omab ainult kahte vastust ehk küsimustikule vastab kahendpuu (bi-puu) struktuur. Vaatamata sellistele ebamugavustele on lihtsustatud küsimustiku võimalik käivitada.



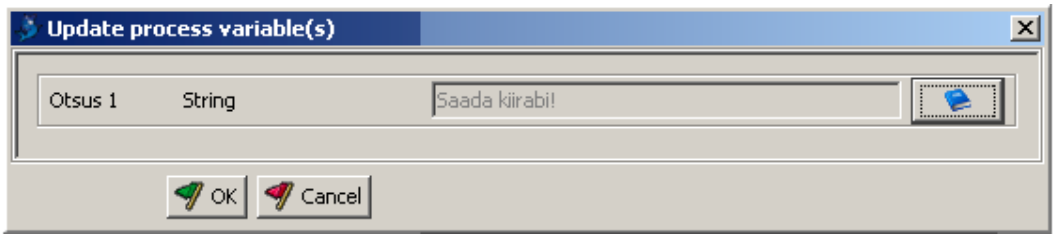
Joonis 30: Kasutaja töölist ja esitatud küsimus

Joonisel 30 on esitatud kasutaja tööloetelu, kus on näha kasutajale määratud ülesanded.



Joonis 31: Esitatud küsimusele vastamine

Joonisel 31 on esitatud dialoog, mille abil töövoomootor suhtleb kasutajaga, pakkudes valida vastust esitatud küsimusele.



Joonis 32: Küsimustiku otsus

Joonisel 32 on esitatud rakenduse poolt otsuse dialoogi aken.

Mugavust annab juurde töövoomootori poolt pakutud käivitatud protsesside visuaalne monitooring. Selle abil on võimalik näha momendis käivitatud protsesse ja selles protsessis momendis aktiivset tegevust. On võimalik näha tegevuste täitmise ajalugu (sellega on tagatud veel üks püstitatud ülesande nõuetest: süsteem peab salvestama graafi läbimisteed) ja hallata protsesse ehk käivitada, peatada ja hävitada.

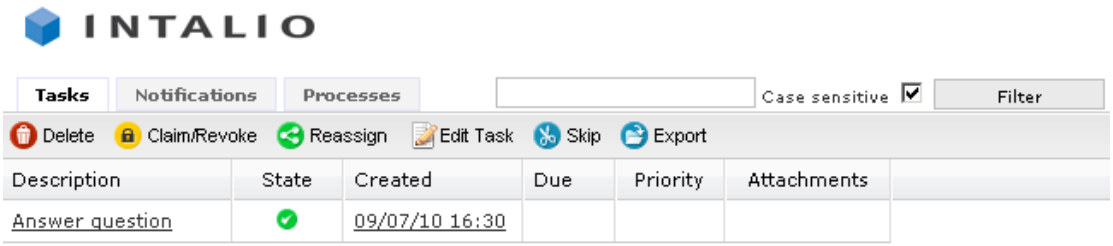
Kokkuvõtteks: professionaalne süsteem suure hulga erinevate ülesannete lahendamiseks. Sellest tulenevalt tehniliselt väga komplitseeritud ja valikvastuste küsimustiku ülesande jaoks üleliia keerukas ja ebamugav süsteem, mis nõuab kasutajalt tehniliste oskuste omandamist ja aega tööriistadega kasutamaõppimist. Oma kohmakuse ja erinevate vastuste defineerimisvõimaluste puudumise tõttu antud kandidaat ei ole sobilik püstitatud ülesande lahendamiseks.

4.5 Kandidaat: Intalio|BPMS

Intalio pakub vabavaralist äriprotsesside modelleerimis- ja käivitamiskeskkonda, mis koosneb modelleerimisliidest, mis toetab BPMN 2.0, XForms 1.1 ja XPath 2.0 tehnoloogiaid ja Apache ODE BPEL töövoomootorist WS-BPEL 2.0 ja **BPEL4People** toetusega. Kasutusele võetud Intalio poolt arendatud *Tempo WS-HumanTask* teenus.

Töövoode toimetaja on realiseeritud pluginana Eclipse[7] arendamiskeskkonna jaoks. Võrreldes *Endyra* toimetajaga pakub *Intalio Designer* diagrammide koostamiseks sümboleid ja reegleid sel kujul, nagu nad on esitatud BPMN spetsifikatsioonis.

Töövoomootori käivitamine on väga lihtne ja mugav – mootor töötab koos *Apache* veebiserveriga. *Intalio* pakub sarnast *Endyra*-ga töövoo käivitamis- ja haldamislähenemist: eksisteerib protsesside monitor, mille abil on võimalik käivitada, peatada ja kustutada töövooge ning iga töövooga seotud kasutaja kasutab oma ülesannete saamiseks süsteemi tööloetelu (vt. Joonis 33). Test-töövoo diagrammi modelleerimisel loodud töövoo lahenduse juurutamine on imelihtne.



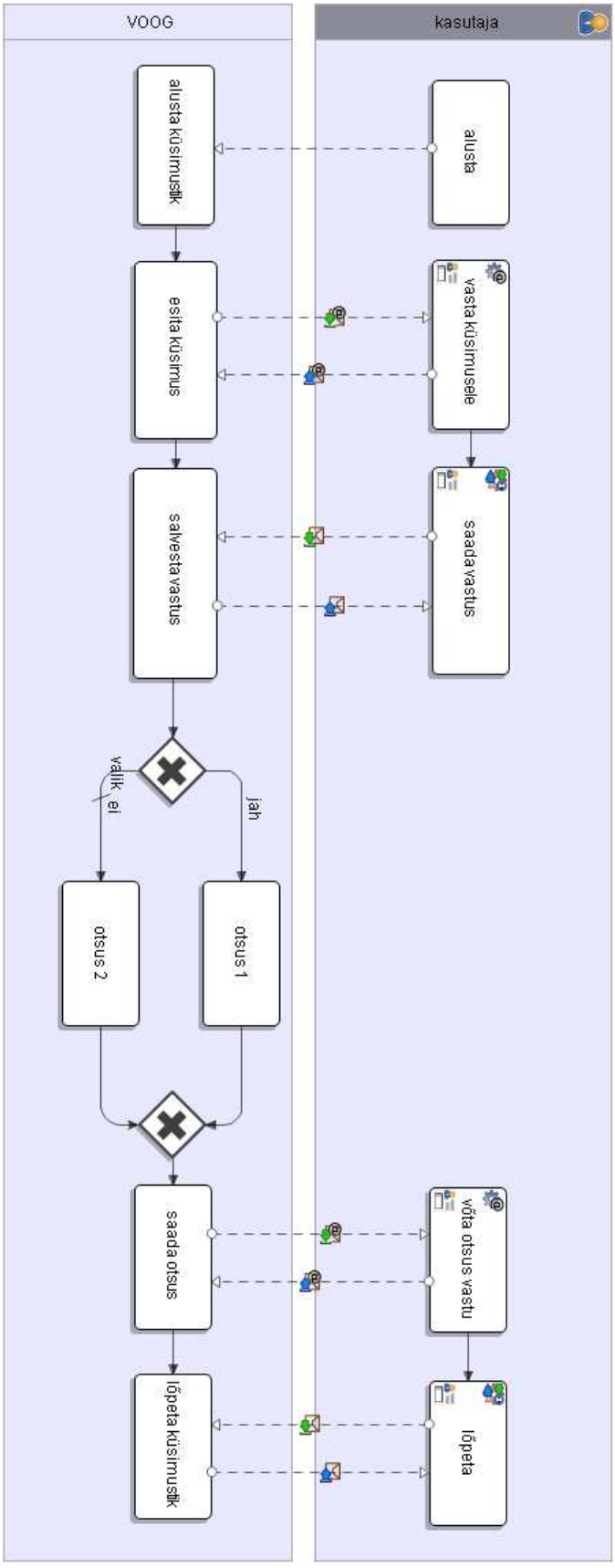
Joonis 33: Intalio süsteemi kasutaja tööloetelu

Tõelisi raskusi annab juurde BPEL tehnoloogia kasutamine. Esiteks BPEL ei salvesta enda sees objektide graafilisi parameetreid, seega teoreetiliselt peale diagrammi salvestamist, hiljem selle faili avamisel graafilises toimetajas pole võimalik saada esialgset seisu. *Intalio* lahendab selle probleemi lisafailide lisamisel projektile. Lisafailides hoitakse töövoo diagrammi objektide graafilist info selleks, et projekti avamisel töövoo diagramm näeks välja samasugune, nagu ta oli salvestamisel. XPDL keelt toetavad toimetajad salvestavad kogu töövoo diagrammi info ühte faili.

Teine tõsine ebamugavus on otseselt seotud BPEL keele otstarbega. Seoses sellega, et WS-BPEL fokuseerib äriprotsessidele, mis suhtlevad veebiteenustega, ei ole BPEL sees realiseeritud funktsionaalust, mis toetaks interaktiivset suhtlemist kasutajaga ja muu väliskeskkonnaga. Selleks on BPEL keele jaoks loodud BPEL4People ja WS-HumanTask laiendused. Seetõttu aga töövoode modelleerimine ja diagrammi koostamine ei ole nii lihtsad ja mugavad, nagu need on XPDL tehnoloogia kasutamisel.

Töövoomootor lõpetab töövooprotsessi ainult siis, kui ta ei leia rohkem tegevusi tegemiseks. Vaatamata sellele, et BPEL ei toeta otseselt interaktiivset suhtlemist kasutajaga, Intalio töövoomootor toetab BPEL4People ja WS-HumanTask laiendusi. See annab võimalus modelleerida test-töövoogu. Selle abil saab aru, kas antud rakendus on nii võimas, et luua

lahendust nii spetsiifilisele ülesandele nagu valikvastuste küsimustik, mille tulemus on otsuseni jõudmine. Lahenduse konstrueerimine algab töövoo diagrammi modelleerimisega.

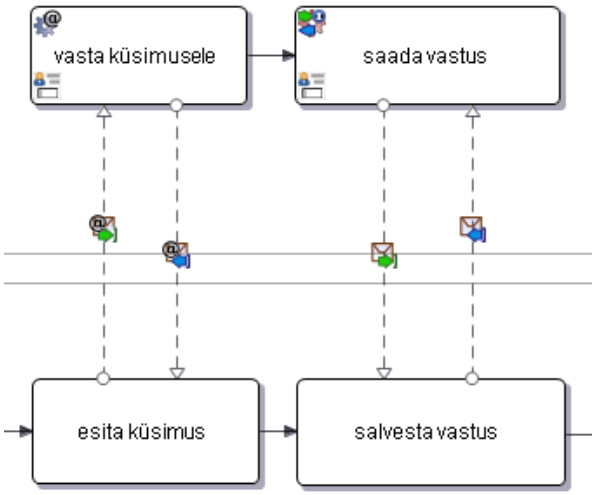

VOOG
kasutaja
alusta küsimustik
alusta
esita küsimus
vasta küsimusele
salvesta vastus
saada vastus
valik
ei
jah
otsus 2
otsus 1
saada otsus
võta otsus vastu
lõpeta küsimustik
lõpeta

Joonis 34: Intalio toimetaja abil loodud test-töövoo diagramm

Joonisel 34 on näha, et diagrammi modelleerimisel kasutatakse *Swimline* diagrammide kuju, kus on selgelt eraldatud inimese tegevused (*People Activity*) ja töövoo automaatsed tegevused. Tegevusi grupeeritakse tegijagruppidesse (*pool*), mis tähendab, et eksisteerib kasutajate tegevusgrupp ja põhi tegevusgrupp. *Endyra* toimetaja abil loodud diagrammi peal ei saa eristada automaatseid tegevusi nagu küsimuse esitamine ja kasutaja tegevusi, nagu küsimusele vastamine (vt. Joonis 29). Tänu sellele, *Intalio* toimetaja poolt moodustatud diagramm sisaldab rohkem infot töövoo protsessist. Aga teiselt vaatenurgalt vaadates, see lisab keerukust ja segadust diagrammi modelleerimisel.

Joonis 35: Küsimustiku alustamissõnum

Joonisel 35 on näidatud, et kasutaja tegevusgrupp on käivitatav (*executable*) ja töövoo põhi tegevusgrupp ei ole otseselt käivitatav: esimese tegevuse („alusta küsimustik“) käivitamine nõuab sõnumi kättesaamist kasutaja tegevusgrupi tegevusest („alusta“), mis on otseselt käivitatav.



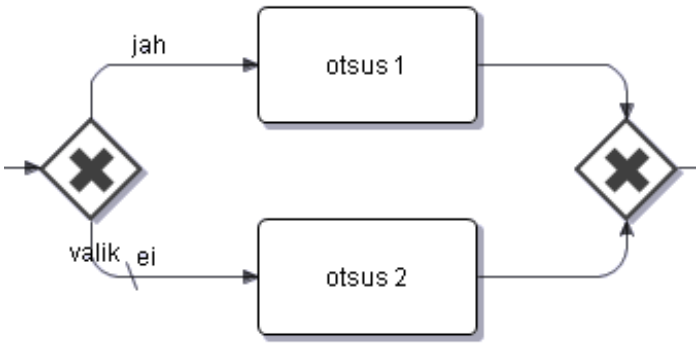
Joonis 36: Sõnumite vahetus põhipuuli ja kasutajapuuli vahel

Joonis 36 illustreerib, et iga põhi tegevusgrupi tegevus, mis nõuab inimese tähelepanu, on seotud kasutaja tegevusgrupis oleva vastava tegevusega. Põhi tegevusgrupi tegevuse alustamisel, saadetakse päringusõnum kasutaja tegevusgrupis olevale seotud tegevusele ja esitatakse küsimus kasutajale. Küsimus on varustatud valikvastustega. Antud versioonis on modelleerimise ajal sisseehitatud kaks vastust: „Jah“ ja „Ei“. Esitamine toimub tööloetellu uue kirje lisamise moel. Seega antud mehhanism kasutaja tähelepanu köitmiseks ei ole väga efektiivne võrreldes, näiteks, *pop-up* dialoogiaknaga. Tegevus („esita küsimus“), mis saadab päringu, peatab kogu protsessi ja ootab vastust kasutaja tegevusgrupis olevast tegevusest („vasta küsimus“), seega WCP16 mall on täielikult rakendatud. Kasutaja saab esitatud küsimuse kätte (avades uue kirje oma tööloetelus) ja vastav sõnum saadetakse tagasi põhi tegevusgrupis olevale tegevusele („esita küsimus“). Sellest momendist kasutaja tegevusgrupi töövoo osa ja põhi tegevusgrupi töövoog töötavad paralleelselt. Järgmisel põhi tegevusgrupis olev tegevuse („salvesta vastus“) ootab kasutaja vastust. Kasutaja vastab küsimusele (vt. Joonis 37) ja paneb dialoogiakna kinni (vajutades „Save“ ja „Complete“ nuppudele), sel momendil kasutaja tegevusgrupi tegevus („saada vastus“) saadab sõnumi koos küsimuse vastusega põhi tegevusgrupis olevale tegevusele „salvesta küsimus“ (antud tegevus ootab kasutaja tegevuse poolt sõnumit, et teha tööd edasi).

Joonis 37: Kasutajale esitatud küsimus

Viimane tegevus salvestab vastuse ja saadab sõnumi tegevusele „saada vastus“, et kõik on korras ja kasutaja tegevus võib oma töö lõpetada. Sel momendil kasutaja tööloetelus olev ülesanne pannakse lõppstaatusesse ja kustutatakse kasutaja tööloetelust.

Järgmised tegevused enne otsuse saatmist süsteem teeb automaatselt.



Joonis 38: Kasutaja vastuse töötlemine ja otsuse kinnitamine

Joonisel 38 on esitatud automaatne põhipuuli tegevuste järjend. See on kasutaja vastuse töötlemine ja otsuse valimine. Otsust tehakse „valik“ *OR*-marsruutimistegevuse käigus, kontrollides kas vastus on „Jah“ või „Ei“. Antud kontroll on realiseeritud sel moel, et kui vastus on „Jah“, siis saadetakse protsess edasi „otsus 1“ tegevusele, vastasel juhul „otsus 2“ tegevusele. Järgmisena on esitatud otsuse tegemise kirjeldatav BPEL kood.

```
<bpel:condition>$question1NotifyTaskCompletionRequestMsg.root/question1:tas
kOutput/question1:output/question1:vastused=1</bpel:condition>
```

Väljend tüüpi *boolean* `vastused=1` kontrollib kas vastus on „Jah“ (*mapping* on 1) või „Ei“ (*mapping* on 0).

Peale vastuse analüüsimist antakse juhtimine üle kas „otsus 1“ või „otsus 2“ tegevusele. Need tegevused väärtustavad sisemise muutuja nimega *msg* vastava otsusega. Järgmised koodilõigud näitavad, kuidas see on realiseeritud BPEL keele abil:

„Otsus 1“ tegevuse puhul:

```
<bpel:copy>
  <bpel:from>"Saada kiirabi!"</bpel:from>
  <bpel:to>$msg</bpel:to>
</bpel:copy>
```

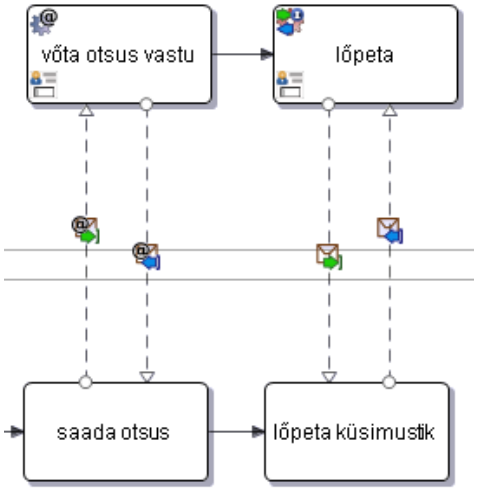
„Otsus 2“ tegevuse puhul:

```
<bpel:copy>
  <bpel:from>"Ara saada kiirabi!"</bpel:from>
  <bpel:to>$msg</bpel:to>
</bpel:copy>
```

Antud sisemine muutuja *msg* väärtust kasutatakse kasutajale otsuse konstrueerimisel. Peale otsuse kinnitamist saadetakse kasutajale vastav otsus (vt. Joonis 39).

Joonis 39: Otsuse kättesaamine

Antud mehhanism töötab samal põhimõttel nagu töötab küsimuse esitamismehhanism: saadetakse otsus kasutajale, kasutaja kinnitab, et otsus on saadud ja dialoogiaken pannakse kinni.



Joonis 40: Otsuse esitamisega seotud tegevused

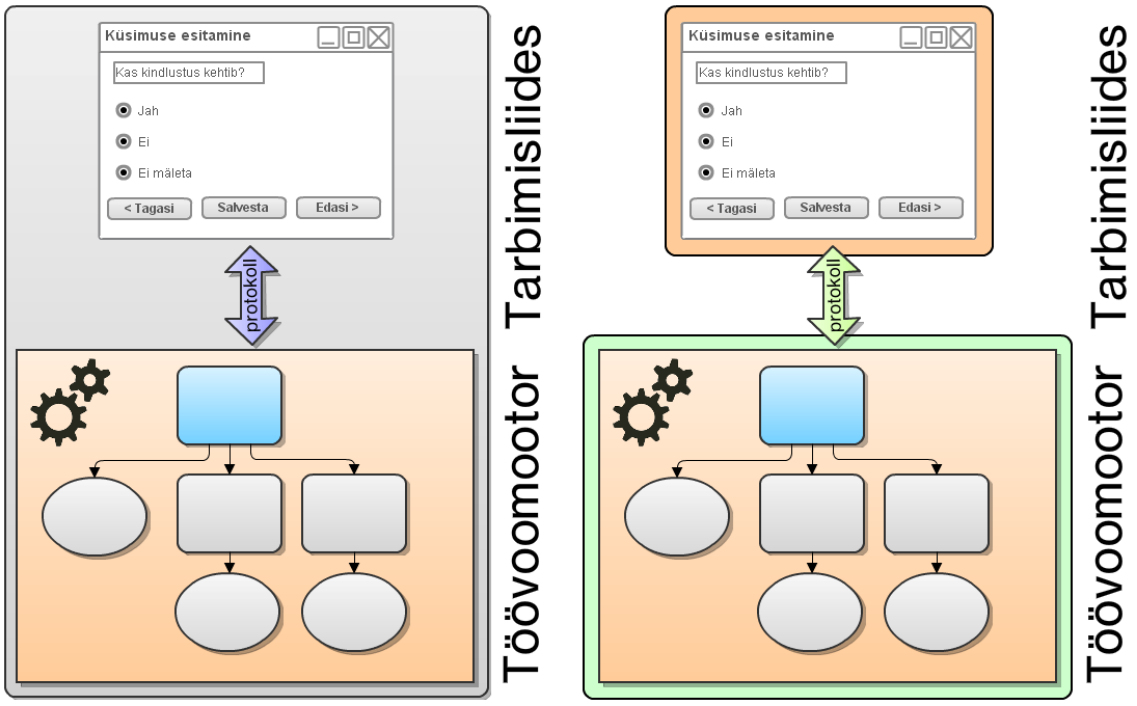
Joonisel 40 on illustreeritud kasutajale otsuse esitamisega seotud tegevused. Tegevus „lõpeta küsimustik“ on viimane tegevus töövoo protsessist, seega BPEL ideoloogia järgi on kogu protsess lõppenud.

Intalio|BPMS on väga võimas süsteem. Aga modelleerimisprotsess näitab, et valikvastuse küsimustiku ülesannet lahendava töövoo diagrammi modelleerimine nõuab tehniliste oskuste olemasolu. Lisaks, diagrammi modelleerimine nõuab palju aega, kui konstrueerida selle sel kujul, nagu on esitatud ülalpool. Tundub, et antud test-töövoo diagrammi on võimalik lahendada ka odavamalt ja lihtsalt, aga selleks on nõutud BPEL tehnoloogia ja Intalio toimetaja sügav kasutamaõppimine. See kokkuvõtteks tunnistab, et antud toode on kohmakas ja keeruline spetsiifilise triviaalse ülesande jaoks.

4.6 Kandidaat: püsiprogrammeeritud süsteem

Teine viis püstitatud ülesannet lahendada on luua süsteem nullist, kus ülesande lahendamisloogika on sisseehitatud. See tähendab, et töövoo loogika ja reeglid on defineeritud programmi koodis otseselt, mitte aga võetud kuskilt väljast, näiteks konfiguratsiooni failist, töövoo protsessi definitsioonidest, andmebaasist jne. Antud tüüpi rakendus on väga kiire, kuna töövoomootor ei pea hoolitsema sellest, kuidas töövoo definitsioone on vaja sisse lugeda, parsida ja interpreteerida – kogu loogika reeglid on realiseeritud kasutades

programmeerimiskeele juhtumiskonstruktsioone: `if…else, while, for, block` jne või selliste konstruktsioonide alternatiive. Rakenduse konstrueerimine on lihtne, kuna teoreetiliselt on vajalik realiseerida töövoosüsteemi kaks komponenti: töövoomootor ja tarbimisliides, mille abil läbitakse töövoomootoris eeldefineeritud küsimustik. Veelgi lihtsam programmeerimisvõimalus, kui töövoomootor ja küsimustiku läbimise liides on üks monoliitne rakendus, näiteks, iseseisev *Java*-rakendus, mis on pakitud *jar*-arhiiviks. Protokoll, mida sel juhul kasutatakse töövoomootori ja tarbimisliidese vahel võib olla täiesti unikaalne sisemine mittestandardiseeritud protokoll (vt. Joonis 41, vasakpoolne skeem).



Joonis 41: Töövoosüsteemi võimalikud ülesehitused

Juhul kui töövoomootor ja tarbimisliides on sõltumatud rakendused, mis suhtlevad omavahel üle võrgu (vt. Joonis 41, parempoolne skeem), siis on andmevahetusprotokolli jaoks mõned kitsendused. Näiteks, et protokoll oleks võimalikult kergekaaluv ja oleks võimeline endas kandma kõiki andmeid, mis on vajalikud tarbimisliidesele.

Antud tüüpi süsteem võib olla lahenduseks püstitatud ülesande jaoks, kuna programmeerimis-keele struktuuride abil on võimalik realiseerida mistahes keerulisi reegleid, seega valikvastuste küsimustiku ülesanne on samuti lahendatav sel viisil. Seoses töövoo reeglite

sisseehitusele, puudub vajadus luua modelleerimisliidest, mille abil modelleeritakse töövoo definitsioone. Seega rakenduse realiseerimine on lihtne ja kiire.

Samas kõik ülesloetud „plussid“ on teisest küljest ka selle lahenduse viisi miinused: töövoogude defineerimisvõimaluse puudus teeb käesoleva lahenduse tüüpi liiga kinniseks ja seega vähepaindlikumaks. Seoses töövoomootori sisseehitatud loogikale, viimane on väga mittepaindlik ja kinnine. Antud süsteemi tüüp ei ole paindlik olukorra või reaalse töö muudatuste suhtes, mida sisseehitatud töövoog kirjeldab. Seoses sellega, et ühe probleemi lahendus võib ajas muutuda, näiteks lahendus tagasiside abil saab olla muudetud lihtsamaks ja kergemaks, seega muutub selle lahendusele vastav töövoog. Selleks, et antud süsteem kasutaks töövoo uut versiooni, on vajalik programmeerija poolt koogi muutmist ja süsteemi uuesti vajadusel kompileerimist.

Samas juba paarikümne töövoo programmi tasemel sissekirjutamine on ebamugav rutiinne töö programmeerijale, kes teadmata valdkonda detailideni võib kergesti teha vigu, mille olemasolu on raske avastada (ja puudumist raske tõestada).

Need aspektid ei anna võimalust nimetada antud lähenemisviisi püstitatud ülesande jaoks efektiivseks lahenduseks.

5. Hüpotees

Püstitatud ülesannet on võimalik lahendada mitmel äärmisel viisil: kasutada olemasolevat universaalset süsteemi või luua uus. Idee on leida lahendus, mis on võimalikult efektiivne, lihtne ja odav, selle alusel kandidaatideks on valitud valmissüsteemid, mis on loodud üldiste ja tihti ilmuvate probleemide lahendamiseks. Neid süsteeme on vaja kohandada selleks, et nad lahendaksid püstitatud valikvastustega küsimustiku ülesannet. Suured kommerts- ja vabavaralised süsteemid kasutavad standardiseeritud tehnoloogiad töövoode modelleerimiseks, haldamiseks ja käivitamiseks. Analüüsi käigus ilmnes, et ainult mõned töövoo tehnoloogiad on võimelised pakkuma lihtsaid vahendeid püstitatud ülesande jaoks lahenduse ehitamiseks, need on BPMN, BPEL + BPEL4People + WS-HumanTask ja XPDL.

Süsteeme, mis endas rakendavad loetletud tehnoloogiaid on palju, seega selleks, et kontrollida, kuidas antud tehnoloogiad suudavad lahendada püstitatud ülesannet, on vajalik läbiproovida mitmeid erinevaid süsteeme ehk läbi proovida tehnoloogiate paare BPMN + XPDL ja BPMN + BPEL. Testimise käigus tundus, et süsteemid, mis kasutavad XPDL töövoode käivitamiseks lahendavad ülesannet efektiivsemalt (vt. Tabel 3).

Töövoo protsessi faas	XPDL	BPEL
Modelleerimine	töövoo diagrammi modelleerimine tundus lihtsam ja läbipaistvam	töövoo diagrammi modelleerimine tundus keerulisem
Definitsiooni salvestamine	töövoo definitsioon ja sellele vastav diagramm on esitatud ühe failiga XPDL formaadis	Selleks, et salvestada diagrami graafilisi aspekte on projektile lisatud lisafaile
Protsessi käivitamine	XPDL omab sisseehitatud mehhanisme töövoo ja kasutaja omavahelise suhtlemise rakendamiseks	BPEL töövoomootor peab toetama ka BPEL4People ja WS-HumanTask laiendit. Antud aspekt kitsendab võimalikke kandidaatide hulka

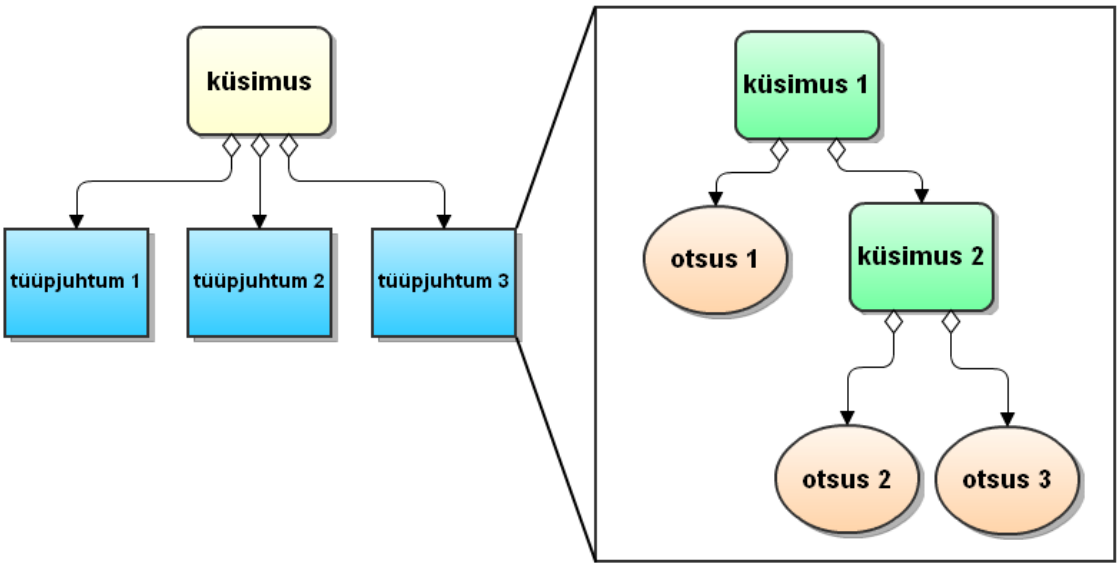
Tabel 3: BPEL ja XPDL tehnoloogiate lühivõrdlus

Vaatamata sellele, et universaalsete süsteemide abil on võimalik lahendada püstitatud ülesannet, selle kohandamine ja diagrammide koostamise keerukus teevad küsimustiku koostamise väga kohmakaks, ebamugavaks ja aeganõutavaks protsessiks. Kui siia lisada veel süsteemide kasutamaõppimisega seotud ajakulu ja kommertssüsteemide puhul litsentside ja uuendustasude kulud, siis antud viis on ebaefektiivseks lahenduseks antud ülesandele ja kogu ülesannete klassile. Olemasolevad universaalsed tooted tundusid kohmakad antud ülesande jaoks. Alternatiiviks oleks olnud programmeerida kogu lahendus algusest peale.

Algusest peale lahenduse programmeerimine on keerulisem variant võrreldes olemasoleva süsteemi kasutamisega. Kuigi see annab suure vabaduse valimisel, mida süsteem teeb ja kuidas süsteem töötab. On võimalik luua ülesande jaoks maksimaalselt sobiva süsteemi, mis otseselt toetab ja lahendab kõiki ülesandega seotud aspekte ja eripärasid (ja ei tee midagi üleliigset). Süsteemi suureks puudus on kinnised töövoo reeglid – kui tuleb vajadus neid uuendada, programmeerijal tuleb muuta koodi ja uuesti süsteemi kompileerida ja testida. Antud protsess on väga ajakulukas, seega ei ole üldse efektiivne.

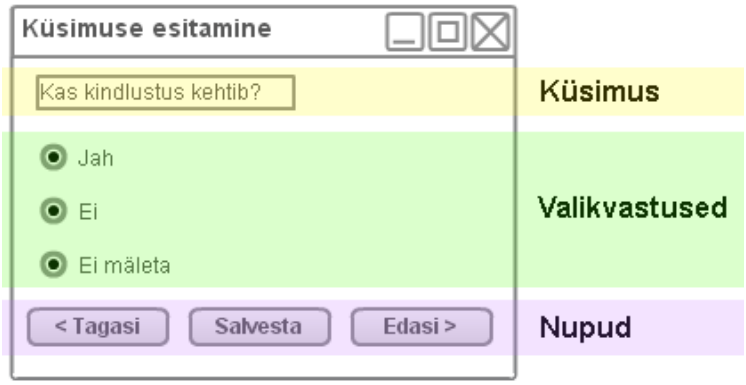
Selleks, et leida sobiv lahendus, mis on odavam, kui esimest kaks alternatiivi, on vajalik leida omadused, mis eristavad valikvastustega küsimustiku ülesannet teistest lühiajalistest ülesannetest.

Küsimusi on võimalik grupeerida valdkonna järgi. See annab võimaluse jagada küsimustikku sektsioonideks ehk tüüpjuhtumiteks, mis on seotud spetsiifiliste küsimustega ja vastustega ja kasutada antud küsimuste kogumit teiste küsimustike koostamisel[19]. Joonis 42 illustreerib tüüpjuhtumite kasutamist.



Joonis 42: Küsimuste tüüpjuhtum

Nagu oli mainitud, püstitatud ülesande lahendatava töövoogu on võimalik dekomponeerida neljaks osaks: töövoo protsessi algus, töövoo protsessi lõpp, otsuse kätte saamine ja küsimuse esitamine koos vastuse saamisega. Küsimuse esitamise ja vastuste saamise protsess on korduv tegevus küsimustikus: töövoomootor valib küsimuse (küsimustiku alguses, see on algküsimus) baseerudes eelmise küsimuse vastusel. Küsimuse esitamine ja eeldefineeritud vastuste hulgast vastuse valimine on realiseeritud kasutades tarbimisliidest. Kasutaja näeb dialoogivormi akent, kus on esitatud küsimus ja valikvastuste variandid. Kasutaja valib vastust ja salvestab, vastust saadetakse töövoomootorile, mille põhjal viimane valib järgmist küsimust ja kasutaja näeb selle sama dialoogiakna, kus on juba teine küsimus ja teised vastusevariandid ja nii edasi kuni töövoomootor jõuab otsuseni. Sel momendil mootor saadab otsuse kasutajale ja protsess lõpetab oma töö. Kasutaja vaatenurgast dialoogiakna vorm on kogu aeg üks ja sama. Muutub ainult küsimuse tekst ja pakutavad vastused.



Joonis 43: Küsimuse esitamisvorm

Joonisel 43 on illustreeritud tarbimisliidese dialoogiaken, mille abil toimub küsimuse esitamine ja vastuse valimine. Kollase värviga märgistatud ala, kus paikneb küsimuse tekst, rohelise alaga – valikvastused ja lilla värviga – nuppud. Kollase ja rohelise ala elemendid on dünaamilised, lilla ala – staatiliste elementide kogum.

Valikvastuste küsimustiku ülesande puhul, sõltumata küsimuste tekstist ja küsimuse vastustest on dialoogiakna vormid sarnased. Seega kogu antud ülesannete klassi, kuhu kuulub ka püstitatud ülesanne, eristab teistest ülesannetest see, et nende eesmärk on otsuseni jõudmine ja küsimuste esitamisel ekraanivormid on sarnased. Lisaks küsimus ei kogu täiendavat informatsiooni: küsimuse esitamise eesmärk on ainult järgmise küsimuse esitamiseks valimine. Neid aspekte on võimalik kasutada selleks, et lahendada püstitatud ülesannet efektiivsemalt.

Baseerudes sellele, tekib hüpotees, et antud ülesanne on lahenduv odavamalt, kui algselt välja mõeldud kaks alternatiivi: luua piisavate kitsendustega töövoosüsteem, mis võtab parimaid omadusi eelmistest kahest alternatiividest. Nimelt:

- Universaalsetest toodetest võtab omaduse modelleerida töövoo definitsioone ja tagada nii töövoomootori sõltumatus definitsioonidest. Töövoomootor on võimas ja avatud.
- Püsiprogrammeeritud süsteem annab ideed rakendada uues süsteemis ainult neid omadusi, mis on nõutud püstitatud ülesande efektiivseks lahenduseks – töövoomootor on võimas ja avatud ainult lühiajaliste ülesande klassis piires, mille eesmärk on otsuseni jõudmine.

Uus süsteem ei ole koormatud liigsete omadustega püstitatud ülesande klassi jaoks, samas ta on maksimaalselt piisav selle lahendamiseks. Lisaks töökulu, võrreldes teiste alternatiividega, on väga väike.

Oma hüpoteesi kinnitamiseks pandi kokku tudengite töörühm (magistritöö autor ja kolm bakalaureusetöö kirjutajat) ning loodi piisavate kitsendustega töövoosüsteem. Järgmises peatükis selle süsteemi realisatsioon, sellega seotud probleemid ja nende lahendused.

6. Süsteemi loomine

Loodud süsteem on tehtud Model-View-Controller [12] põhimõttel. See tähendab, et ta on jaotatud loogilisteks osadeks, kus iga osa ehk komponent tegeleb oma ülesannetega. Komponent on eraldiseisev programm, mis kasutab teiste komponentide tööd oma eesmärkide ja ülesannete täitmiseks. Osad on järgmised:

- Kasutajaliides – moodul, mille ülesandeks on abistada küsijat/teenindajat vahetult.
- Töövoomootor – moodul, mis on vahekiht kasutajaliidese ja andmebaasi (töövoode definitsioonide) vahel.
- Töövoode haldamise ja defineerimise liides ehk modelleerimisliides

Loodav süsteem on „spetsiifiline“ teatud kitsendustega töövoo süsteem, mis on väga piisav efektiivne lahendus püstitatud ülesande jaoks.

Töövoosüsteemi **tarbimisliides** on visuaalne rakendus, mis näitab kasutajale andmeid (küsimused ja vastused) ning pakub vahendeid andmete kasutamiseks ja navigeerimiseks. Kasutajaliidese abil küsija näeb küsimuse teksti, vastuseid, seni teostatud küsimuste piires navigeerimise võimalust ja muud informatsiooni, mis on vajalik küsitluse edukaks läbiviimiseks. Süsteemi **töövoomootor** on nähtamatu rakendus, mis toetab kasutajaliidest vajalikke andmetega ning haldab kasutajaliideselt tulnud käske. Süsteemi **modelleerimisliides** on visuaalne rakendus, mis aitab küsitluste koostajal mugavalt koostada töövoogusid. Lisaks pakub rakendus vahendeid olemasolevate andmete muutmiseks ja kustutamiseks, seega antud rakendus on töövoosüsteemi andmete toimetaja.

Aastal 2008 valmis magistritöö autori juhtimise all töövoomootor, kaks erinevat kasutajaliidest, vajalik andmebaasi struktuur ning kasutajaliidese ja töövoomootori vahelise suhtluse protokoll. Dokumendid kasutajaliideste kohta valmistasid bakalaureusetöö raames Tartu Ülikooli tudengid Allar Tammik ja Roman Norman. Töövoomootori loojaks oli Tartu Ülikooli tudeng Hillar Petersen.

6.1 Süsteemi spetsiifika

Termin „spetsiifiline“ tähendab süvenemist konkreetsele piiratud alale. „Spetsiifiline süsteem“ on süsteem, mis on spetsialiseerunud konkreetse piiratud arvu tegevuste sooritamiseks. Käesolevas peatükis on toodud spetsiifilisele töövoosüsteemile esitatud kitsendused.

Töö objektiks oleva spetsiifilise töövoosüsteemi protsessiks on enamasti küsimuse esitamine. Seetõttu tähistatakse töös ja tulemuses pideva korduse vältimiseks küsimuse esitamise protsessi objektile viitava sõnaga „küsimus“ ning protsessi nimeks kasutatakse küsimuse sõnastust. Küsimus omab piiratud arvu eeldefineeritud vastuseid, mis on eeldefineeritud selleks, et süsteemi saaks kasutada kiiresti ka valdkonna osas vähese väljaõppega isikud. Vastuse eesmärgiks antud töövoosüsteemis on ainult viimine lõppotsuseni (läbi järgmiste küsimuste) ning küsimus ei kogu täiendavat informatsiooni. Küsimused ja vastused peavad tagama jõudmise õigete otsusteni. Komplitseeritud töövoode korral saab kogu töövoo jagada tüüpjuhtumiteks (iseseisvateks tüüpjuhtumiteks). Selle läbi muutub suurte töövoogude defineerimine selgemaks ning vastus võib viia lisaks vahetu lõppotsuseni või järgmise küsimuseni ka väljalõigatud tüüpjuhtumi käsitlemisele.

Töös kasutatakse vastustele järgnevaid tegevusi lühemate väljenditega:

- lõpp – teatud tüüpi otsuse fikseerimine
- järgmine küsimus – järgmise küsimuse esitamine
- viide (teisele) tüüpjuhtumile – teise tüüpjuhtumi alguskü simuse esitamine

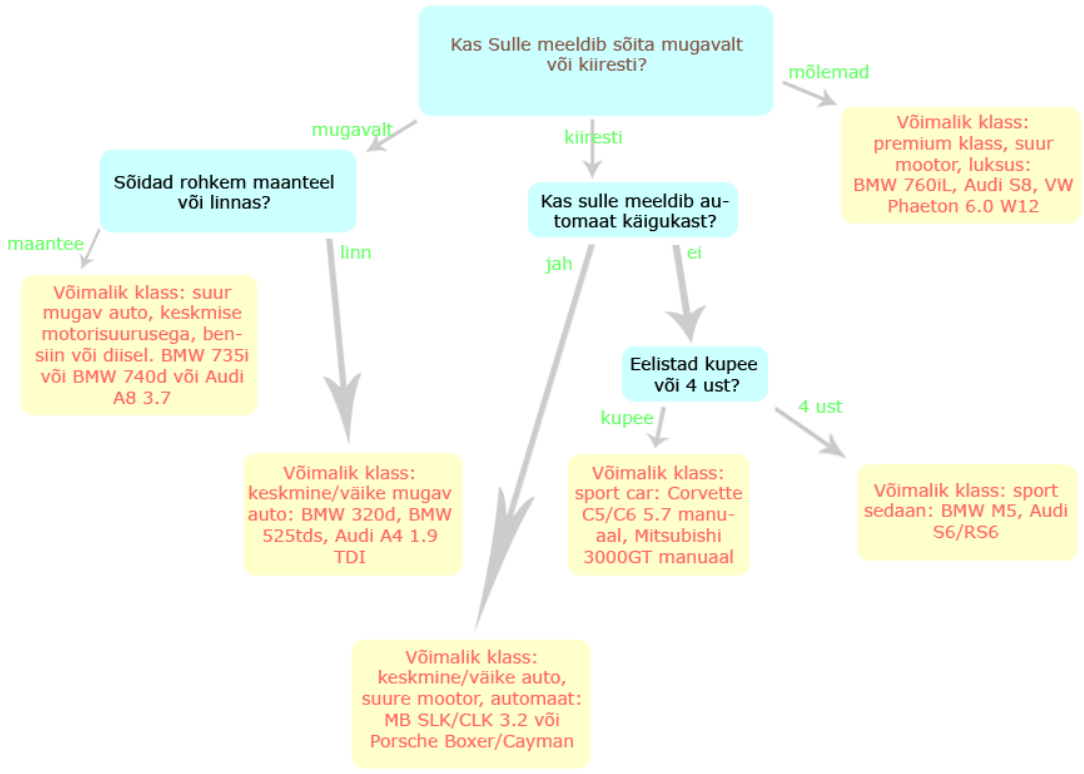
Enne uue küsimusega tegelemist võib süsteem pakkuda juhendeid küsimuste vahetegevuste läbimiseks (näiteks, *„Lugege boileri kasutusjuhend läbi enne uuele küsimusele vastamist“*). Kui vastus viib üle teise küsimustikku, siis see tähendab, et peale küsimusele vastamist vastaja suunatakse teise tüüpjuhtumi esimesele küsimusele. Kui vastus viib lõppu, siis see tähendab seda, et küsimustik on edukalt läbitud ning jõutud otsuseni. Järgmises jaotises on toodud kirjeldus loodava süsteemi projektist.

6.2 Projekt

Järgmises peatükis on kogu projekt kirjeldatud üldisest aspektist: projekti visiooni ja vajaduste analüüs, valdkonna kirjeldus, süsteemi füüsiline ning loogiline struktuur ja funktsionaalsed ning andmenõuded.

6.2.1 Visioon ja vajaduste analüüs

Olgu tegemist rutiinküsitlustega, mis on üks töövoosüsteemidest, kus on eeldefineeritud küsimus-vastuse käitumisloogika ning kogu küsitluse läbides jõutakse konkreetse tulemuseni. Antud küsitluse tüüp on väga efektiivne juhul kui vastaja ei suuda vastata küsimusele konkreetselt, siis talle pakutatakse hulk suunavaid küsimusi ning küsitluse lõppedes ongi vastus käes. Seega süsteemi abil jagatakse probleem osadeks, millele on palju lihtsam vastuseid leida kui esialgsele küsimusele. Näiteks küsimusele “Milline auto sulle kõige rohkem meeldib?” on kohe vastata üsna raske, sest inimesel on väga palju kriteeriumeid mille abil ta hakkab autot ostmiseks valima. Ja on väga tõenäoline, et mitte ükski auto ei vasta kõikidele nõuetele, mis ostja esitab. Üks nendest nõuetest on näiteks, et kütusekulu oleks võimalikult väike ja võimsus oleks võimalikult suur. On arusaadav, et need nõuded on teineteist välistavad, sest kui mootor on piisavalt suur ja võimas siis on seda ka kütusekulu. Aga kuna autole hakatakse esitama väga palju nõudeid, siis küsitluste läbiviimine võib vähemalt piirata autode klassi, kus kõige rohkem nõudeid on rahuldatud. Võimalik küsitluse stsenaarium võiks olla selline (Joonis 44):

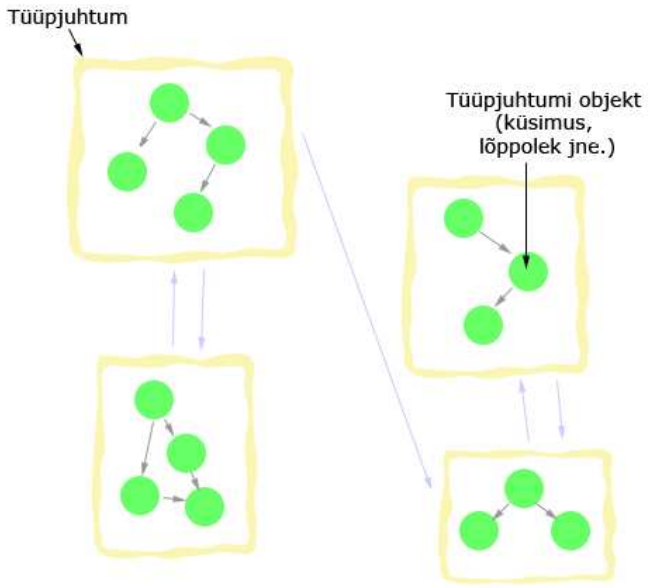


Joonis 44: Auto valimine

Küsitlusel peab olema kindel alguspunkt ehk **alguskü**simus, millest küsitlus algab. Igal küsimusel on teatud piiratud hulk sobivad eeldefineeritud vastuseid, millede hulgast vastaja saab valida ühe. Iga kord valitakse vastus, mis on kõige rohkem sobib. Valikust sõltub see, millisele järgmisele küsimusele palutakse vastajal vastata ja. Et kindlaks teha, kas vastaja vastab õigesti ja täpselt, võib küsija sõnastada küsimust teistmoodi ning pakkuda vastajal sellele vastata. Näiteks küsimus: “Kas te õpite hästi?” on udune ja ebatäpne, seega vastaja võib valida “Ei tea” vastuse, seega järgmine küsimus tuleb natuke täpsem, et suunata vastajat rohkem teemasse ja näidata talle, mis on saadakse aru sõna “hästi” all: “Milline on teie keskmine hinne viimase kolme kuu jooksul?”. Kui küsimusele vastatakse näiteks “3.5” on selge et õpilasel on õpinguga probleemid seega ka esimesele küsimusele saab kindlalt panna vastuse “Ei”. Lisaks on küsitlejal enne uue küsimuse pakkumist olemas võimalus pakkuda vastajale juhendeid lisategevuseks või lisainformatsiooni järgmisest küsimusest aru saamiseks. Näiteks viimase küsimuse eeltegevuseks oleks see, et küsija seletab vastajale kuidas arvutada kolme kuu jooksul keskmist hinnet. Küsimuste esitamisega ja vahetegevuste pakkumisega jätkatakse kuni lõpptulemuse selgumiseni.

Projekti eesmärgiks oli luua spetsiifiline töövoosüsteem, mille spetsiifika seisneb selles, et süsteem aitab autonoomselt koostada ja hallata rutiinküsitlusi, kasutades eeldefineeritud objekte. Objektid on süsteemi põhielemendid, mille abil kostja koostab küsitlusi ja küsimusi ning vastaja vastab nendele. Eksisteerib piiratud arv töövoo *põhiobjekte* (küsimus, lõppseis, viide teisse juhtumisse) ja piiratud arv töövoo *lisaobjekte* (küsimuste ühendus, vastus, tegevus). Igal objektil on oma notatsioon, mis on kirjeldatud osises "6.5.1 Notatsioon". Vastaja vastuse valikud salvestatakse, et hiljem teha statistikat ja analüüsi näiteks, et milline küsimus ei too lisaselgust küsitlusele või milline vastusevariant ei sobi pikka aega mitte kellelegi ja seda on parem üldse mitte pakkuda.

Järgmine samm andmete hoidmise efektiivsuse poole minekuks on grupeerida küsimused teemade järgi, kui tegevusvaldkond on üsna lai. Lõppude lõpuks on selge, et süsteemi manipuleerimisobjektiks on küsimuste kogum ehk tüüpjuhtum. Iga tüüpjuhtum sisaldab endas küsimuse-vastuse järgnevusloogikat, mis on seotud konkreetse teemaga või valdkonnaga. Näiteks tüüpjuhtum, mis teeb selgeks, milline värv on vastaja jaoks sobivam või tüüpjuhtum, mis sisaldab endas küsimusi ja juhendeid, kuidas täpsustada milline haigus patsiendil on. Igal tüüpjuhtumil olemas küsimus, millest alustatakse küsimist, samas ka tüüpjuhtumite kogumil on olemas alustamise tüüpjuhtum, millest küsitlus peale hakkab. Objekt-orienteeritud programmeerimise paradigmas on selline struktuur on hästi tuntud: iga tüüpjuhtum on klass, iga küsimus on protseduur, mis kutsub välja kas sama klassi teist protseduuri või teise klassi esimest protseduuri. Iga tüüpjuhtumi alguskü simus on nagu algus protseduur klassis, ning algustüüpjuhtum juhtumite kogumis on nagu reguleerija, mis suunab vastajat sobivale tüüpjuhtumile sõltuvalt sellest millise vastuse kasutaja valib. Näiteks, algustüüpjuhtum päästeametis võib sisaldada küsimust, mis küsib: "Millise valdkonnaga teie probleem on seotud?" ja vastused: "Politsei", "Päästeamet", "Kiirabi". Sobiva vastuse valides, süsteem suunab teid järgmisele tüüpjuhtumile. Teiste sõnadega küsitlus on tüüpjuhtumite kogum (vt. Joonis 45 ja Joonis 42).



Joonis 45: Tüüpjuhtumite kogum

Iga küsimustik viib alati konkreetse tulemuseni ehk lõppolekuni, kus süsteem salvestab vastaja vastused edaspidiseks analüüsimiseks.

6.2.2 Küsimuste ja vastuste sarnasus

Kuna tüüpjuhtumid on teineteisest sõltumatud elemendid, ning defineeritakse neid ka eraldi, siis küsitluse käigus võib juhtuda, et kasutajale esitatakse sama või teise sõnastusega, aga sama sisuga küsimus kui see, millele vastaja on juba vastanud. Sel juhul süsteem valib ja märgistab eelneva vastuse, mis on sarnane varem valitud vastusega. Vastajal on olemas võimalus selle vastuse suhtes ümber mõelda ja vajadusel muuta. Küsimuste ja vastuste sarnasus defineeritakse küsitleja poolt küsitluste ja tüüpjuhtumite koostamisel modelleerimisliidese abil.

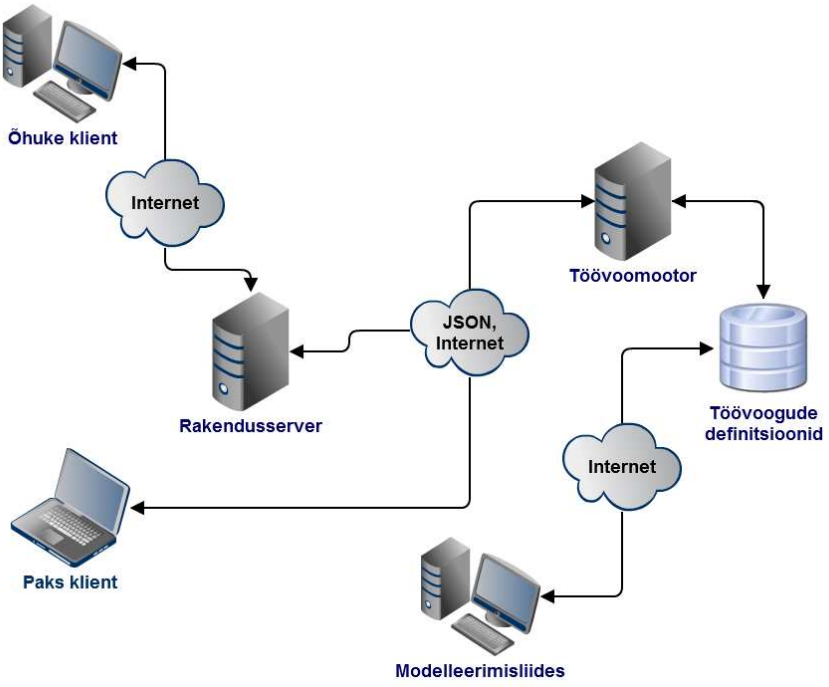
6.2.3 Süsteemi füüsiline struktuur

Terve süsteem omab palju funktsionaalsust, seega on jaotatud elementideks. Iga element omab konkreetseid eesmärke, ülesandeid ja vastutab nende täitmise eest.

Seega süsteem on nagu üks tervik kus iga element töötab kooskõlas teiste elementidega antud süsteemis. Elemente saab omakorda grupeerida eesmärgi järgi:

- Tarbimisliidesed – interaktiivne suhtlemine vastajaga, vastaja abistamine informatsiooniga (küsimuste kirjeldused, tegevuste juhendid, vastused jne), koguvad vastaja poolt valitud vastuseid ja saadavad süsteemi tuumale
- Süsteemi tuum – koht, kus on kontsentreeritud kogu süsteemi äriloogika. Selle grupi elemendid tegelevad andmete manipuleerimisega, andmete hoidmisega ja vastaja vastuste salvestamisega.
- Modelleerimisliides – liides, mis võimaldab küsitlejal luua ja hallata andmeid mida kogu süsteem kasutama hakkab. Teiste sõnadega on antud liides küsimustike redaktor.

Tarbimisliideste gruppi kuuluvad juba realiseeritud *õhuke klient* (veebiklient koos rakendusserveriga) ning *paks klient* (allalaaditav Win32 rakendus). Süsteemi tuuma kuuluvad *töövoomootor*, mille ülesandeks on andmetega manipuleerimine ja vastaja vastuste salvestamine, ning *töövooandmebaas*, mille ülesandeks on andmete hoidmine struktureeritud kujul. Andmebaasi struktuuri kirjeldus on toodud käesolevas peatükis. Modelleerimisliidese gruppi kuulub ainus modelleerimisliides, mille loomine on autori üks magistritöö osadest. Joonis 56 illustreerib ülaltoodut teksti graafilisel kujul:



Joonis 46: Süsteemi struktuur

Kuna tarbimisliidese elemendid ja süsteemi töövoomootor suhtlevad omavahel kindlat protokolli kasutades, siis antud süsteem on laiendatav: tuleb realiseerida tarbimisklient mis hakkab 100% antud suhtlusprotokolli toetama.

6.3 Rakenduse funktsionaalsed nõuded

Antud peatükis on kirjeldatud süsteemi funktsionaalsed nõuded, mis peavad olema tagatud süsteemi efektiivseks ja mugavaks tööks. Rakenduse funktsionaalsus on jagatud kahte ossa:

- põhifunktsionaalsus – funktsionaalsus, mis on tagatud rakenduse põhielementidest
- lisafunktsionaalsus – vahendid, mis teevad kasutaja tööd mugavamaks ja lihtsamaks

Põhifunktsionaalsust tagavad rakenduse lõuend ja visuaalsed objektid, lisafunktsionaalsust toovad rakenduse instrumentaalne paneel, objekti info paneel ja üldinfopaneel.

Arvesse on võetud aspekt, et süsteem koosneb hajutatud komponentidest, seega ühest funktsionaalsete nõuete punktidest on efektiivne ja kompaktne osade vaheline suhtlusprotokoll. Kuna süsteemi arendamisest võttis osa 4 inimest, siis iga osa arendati eraldi, kasutades paikapandud nõudmisi ja süsteemi spetsifikatsiooni. Iga süsteemi elemendi funktsionaalsed nõuded on leitavad vastava autori poolt koostatud dokumendis:

- “Töövoode läbimise kasutajaliides – aknapõhine rakendus”, *pt. 2. Funktsionaalsed nõuded*, Roman Normann, 2007 [43]
- “Töövoo läbimise süsteemi õhuke kasutajaliides”, *pt. 1.4. Funktsionaalsed nõuded*, Allar Tammik, 2007 [42]
- Töövoomootor – dokument on hetkel arendamisel

Käesolevas peatükis on kirjeldatud modelleerimisliidese funktsionaalsed nõuded.

6.3.1 Andmevahetus andmebaasiga

Programm peab olema võimeline laadima ja salvestama tüüpjuhtumeid andmebaasist. Selle jaoks on programmi paketis olemas klass ühenduse tagamiseks programmi ja andmebaasiliidese vahel. Rakendus peab pakkuma kasutajale liidest andmebaasiga ühenduse loomiseks kasutades andmebaasi serveri IP-aadressi, porti, kasutajanime, parooli ja andmebaasi nimetust. Lisaks peab eksisteerima lisafunktsioon failist ühenduse andmete lugemise jaoks. Andmete salvestamisel ja lugemisel peab programm informeerima kasutajat tegevuste lõpetamisest ja tekkinud vigadest.

6.3.2 Visuaalsed elemendid

Visuaalseteks elementideks nimetame elemente, mille abil kasutaja koostab tippjuhtmeid ja küsimustikke. Programm peab pakkuma 4 visuaalset elementi, mis omavad erinevat tähtsust andmete loomisel:

- **Küsimus** (*question*) – kõige kasutatavam visuaalne objekt, mille peal peab kirjas olemas tema nimetus ja kirjeldus. Küsimuse visuaalne objekt on süsteemi küsimuse visuaalne interpretatsioon modelleerimisliideses. Visuaalse objekti nimetus on küsimuse sõnastus ja kirjeldus on küsimuse kirjeldus. Küsimuse objekt võib olla kas aktiivne või mitte, mitteaktiivsuse ajal küsimus ei oma ühendusi teiste küsimustega. Küsimuse objekt võib olla kas tüüpjuhtumi algküsimus või mitte. Kui küsimus on algküsimus, siis peale tüüpjuhtumi läbimist alustatakse just sellest küsimusest.
- **Lõppolek** (*end state*) – visuaalne objekt mis interpreteerib küsimustikku või tüüpjuhtumi lõppu. Objekt omab nimetust ja kirjeldust vaatavalt lõpu tekstiga ja kirjeldusega. Käesolev visuaalne objekt saab olla kahes olekus, kas aktiivne või mitte. Kui objekti olek pole aktiivne, siis temal ei ole ühendusi teiste objektidega.
- **Viide teistele tüüpjuhtumile** (*link to occasion*) **–** objekt interpreteerib viidet teisele tüüpjuhtumile. Selle objekti abil on realiseeritud liikumine ühelt tüüpjuhtumilt teisele ühe küsimustiku raames. Visuaalne objekt omab nimetust ja kirjeldust, mis on vastavalt viidatud tüüpjuhtumi nimetus ja kirjeldus. Nagu ka teised objektid siis käesolev objekt saab olla kas aktiivne või mitteaktiivne. Mitteaktiivsuse puhul, objekt ei ole ühendatud teiste objektidega

- **Ühendus, vastus ja tegevus** (*connection, answer, activity*) – kõik kolm objekti on interpreteeritud ühe visuaalse objektina. Ühendus illustreerib järgnevusloogika liikumist ühest objektist teise vastavalt vastaja poolt valitud vastustest. Objektil on olemas algus ja lõpp, kus alguspunkt on ühendatud objektiga millest ühendus väljub (allikasobjekt) ja lõpppunkt on ühendatud objektiga, millele ühendus viitab (sihtobjekt). Ühenduse lõpp on ka visuaalselt märgistatud, selleks et visualiseerida ühenduse orientatsiooni. Igal ühendusega on seotud täpselt üks vastuse objekt. Visuaalne vastuse objekt esitab allikaküsimuse vastust. Seega ühel küsimusel on samapalju väljuvaid ühendusi, kui palju vastuseid antud küsimusel on. Igal ühendusega võib olla soetud null või üks tegevuse objekti. Tegevuse visuaalne objekt esitab tegevust, mida pakutatakse küsimustiku vastajale täitmiseks enne käesoleva ühenduse lõpp-küsimusega tegelemist. Seega antud objektide kogum peab töötama järgnevalt: iga küsimus omab vastusvariante, igal vastusel on olemas ühendus, mis ühendab kaks objekti. Kui vastaja valib antud vastuse küsimusele vastamisel, siis suunatakse vastaja objektile mis on ühenduse lõpppunkt. Kui eksisteerib tegevus teel allikaobjektist lõpp-objektini, siis vastajale pakutatakse juhendid selle täitmiseks.

6.3.3 Tüüpjuhtumi salvestamine

Juhtumi koostamise käigus koostaja loob graafi visuaalsete objektide abil. Graafi tippudeks on kolm esimest objekti, mis oli kirjeldatud eelmises peatükis. Graafi kaarteks on viimane objektide grupp, mida kasutatakse selleks, et luua objektide omavahelisi seoseid ja luua järgnevusloogikat objektide vahel. Peale visuaalse graafi koostamise lõpetamist, on loojal võimalus see salvestada andmebaasi konkreetse tüüpjuhtumi alla. Kuna liides lubab salvestada ka lõpetamatute objektide struktuure ja on olemas võimalus, et klient hakkab kohe selle tüüpjuhtumi struktuuri kasutama siis eksisteerib tingimus, millele loodav struktuur peab vastama, et teda saaks salvestada andmebaasi:

- Graafis peab olema üks ja ainult üks objekt algusobjektiks. Kui loodav struktuur on tüüpjuhtum siis iga tüüpjuhtumit alustatakse algusobjektist, seega tüüpjuhtumil peab kindlasti olema määratud algusobjekt. Kuna küsimus on ainuke põhiobjekt süsteemis, siis algusobjektiks võib olla ainult küsimuse visuaalne objekt.

Kuna süsteem võimaldab salvestada ka tüüpjuhtumit, mis on veel arendamisel, siis on täiesti lubatav jätta mõni objekt mitteaktiivseks: peale salvestamist ja uuesti üleslaadimist, saab antud mitteaktiivset objekti redigeerida ja lisada struktuuri sisse ehk aktiveerida ja ühendada olemasolevate objektidega. Tüüpjuhtumi struktuuri salvestamisel koostaja peab meeles pidama, et klient hakkab läbima ainult neid objekte, milleni saab leida teed algusobjektist selle objektini. See tähendab, et loodavas graafis peab eksisteerima tee algustipust vaadeldava objektini. Need objektid, mille kohta ei ole teed algusobjektist, ei osale küsimustikus, mida hakatakse saatma kliendile edaspidiseks näitamiseks vastajale. Sama reegel töötab ka mitteaktiivsete objektidega, kuna need objektid definitsiooni järgi pole ühendatud teiste objektidega.

6.3.4 Tüüpjuhtumi laadimine

Tüüpjuhtumi laadimise protsess on salvestuse protsessi vastane, see tähendab, et programm loeb andmebaasist andmeid ja konstrueerib nende põhjal täiesti samasuguse visuaalse mudeli, mille pealt need andmed olid genereeritud salvestamise ajal. Salvestamise ja laadimise mehhanismi peamiseks reegliks on see, et andmeid ei tohi kaotada. Peale tüüpjuhtumi struktuuri üleslaadimist peab iga visuaalne element omama samu andmeid, mis oli külge pandud enne salvestamist ja iga element peab omama samapalju ühendusi teiste elementidega ja iga element peab paiknema visuaalselt samal kohal.

6.3.5 Visuaalsete objektide loomine

Käesolevas peatükis on toodud funktsionaalsed nõuded modelleerimisliidesele visuaalsete objektide loomisel. Visuaalsete objektide grupiks loeme neid nelja gruppi, mis oli kirjeldatud peatükis “Visuaalsed elemendid”. Kuna vaatluse all on elementide visualiseerimine, siis objektid peavad paiknema taustal või teise sõnadega lõuendil.

6.3.5.1 Küsimuse objekti loomine

Programm peab pakkuma kasutajale mugavaid lahendusi ja vahendeid visuaalse küsimuse objekti loomiseks. Esiteks valib kasutaja elementide valikust sobiva elemendi, antud juhul on element *Question*, ning vasaku hiireklahviga vajutab lõuendile. Programm avab uue akna, kus asub vorm küsimuse objekti parameetrite sisestamiseks. Kasutaja peab täitma:

- Küsimuse nimetus (*Name*) – kõige tähtsam parameeter, sisuliselt see on tekst, mida hakatakse kasutajale näitama küsitluse käigus. Antud väli on kohustuslik.
- Küsimuse kirjeldus (*Description*) – parameeter, mis seletab lahti küsimuse nimetuse. Teiste sõnadega, küsimuse kirjeldus on küsimuse nimetuse seletav tekst. Väli pole kohustuslik.
- Küsimuse aktiivsus (*Active*) – määrab ära küsimuse objekti aktiivsuse lõuendil ja ka salvestataval andmemudelil.
- Alguskü simus (*Is initial*) – parameeter määrab, kas küsimus on tüüpjuhtumi alguselement, millest alustatakse antud tüüpjuhtumi näitamist.

Nupule “OK” vajutamisel programm kontrollib andmed. Kui andmed on vigased, siis programm teavitab sellest kasutajat. Kui kontrollimine õnnestunud, siis programm sulgeb akna ja joonistab lõuendile uue küsimuse objekti enne sisestatud parameetritega.

6.3.5.2 Lõppoleku objekti loomine

Lõppoleku kui visuaalse objekti loomine on sarnane küsimuse objekti loomisega. Esiteks kasutaja valib elementide valikust sobiva elemendi, antud juhul on element *End state*, ning vasaku hiireklahviga vajutab lõuendile. Programm avab uue akna, kus on toodud vorm lõppoleku objekti parameetrite sisestamiseks. Kasutaja peab täitma:

- Lõppoleku nimetus (*Name*) - tekst, mida hakatakse kasutajale näitama küsitluse edukal lõpul. Antud väli on kohustuslik.
- Lõppoleku kirjeldus (*Description*) – parameeter, mis seletab lahti lõppoleku nimetuse. Väli pole kohustuslik.
- Lõppoleku aktiivsus (*Active*) – määrab ära lõppoleku objekti aktiivsuse lõuendil ja ka salvestataval andmemudelil.

Nupule “OK” vajutamisel programm kontrollib andmed. Kui andmed on vigased, siis programm teavitab sellest kasutajat. Kui kontrollimine on õnnestunud siis programm sulgeb akna ja joonistab lõuendile uue lõppoleku objekti enne sisestatud parameetritega.

6.3.5.3 Teisele tüüpjuhtumile viite objekti loomine

Esiteks kasutaja valib elementide valikust sobiva elemendi, antud juhul on element *Link to occasion/Occasion*, ning vasaku hiireklahviga vajutab lõuendile. Kuna antud objekt illustreerib tüüpjuhtumit, mis on salvestatud andmebaasis, siis kirjeldame selle objekti tüüpjuhtumi peeglina. Arvestades sellega, peale hiireklahvi vajutamist, programm avab vormi kus on toodud olemasolevate tüüpjuhtumite nimekiri kust kasutaja valib ühe ja kinnitab valiku “OK” nupu vajutamisega. Peale seda programm sulgeb avatud dialoogiakna ja joonistab lõuendile vastava objekti, mille peale on kirjutatud vastav tüüpjuhtumi nimetus. Viide tüüpjuhtumile on kogu aeg aktiivne, see tähendab, et kui antud objekt on lõuendil ja ühendatud teiste objektidega, siis süsteemimootor hakkab saatma kliendile infot, mis on seotud selle objektiga.

6.3.5.4 Ühenduse, vastuse, tegevuse objektide loomine

Kuna vastus (*answer*) ja tegevus (*activity*) ei saa eksisteerida ilma ühenduseta (*connection*), siis vaatleme neid elemente koos. Esiteks kasutaja valib elementide valikust sobiva elemendi, antud juhul on element *Connection*. Kuna ühendus nõuab kahe objekti olemas olekut, et neid ühendada, siis peab lõuendil eksisteerima vähemalt kaks objekti, millest üks on tüüpi küsimus, sest ainult see objekt võib olla ühenduse algatajaks. Valides kahte sobivat objekti ühendamiseks, on kasutajal võimalik objekte ühendada. Peale viimast operatsiooni toob programm ekraanile uue akna, kus on toodud ühenduse objektidega seotud elementide parameetrid. Need on:

- vastuse nimetus – vastuse tekst, mida näeb kasutaja enda ekraanil küsimustiku täitmisel. Antud parameeter on kohustuslik, kuna ilma vastuseta ei saa ühendust objektide vahel eksisteerida.

- vastuse kirjeldus – vastust kirjeldav ja lahtiseletav tekst. Näidatakse kasutajale vastuse vihjeks (*tooltip*). See parameeter pole kohustuslik.
- tegevuse nimetus – tegevuse tekst, mida näidatakse kasutajale küsimustiku täitmisel. See parameeter pole kohustuslik, kuna peale igat küsimust ei pruugi vahepealne tegevus olemas olla.
- tegevuse kirjeldus – tegevust lahtiseletav tekst

Ühenduse objektide loomisel ja parameetrite sisestamise käigus, peab koostajal eksisteerima võimalus kasutada juba vanu, olemasolevaid tegevusi, mis paiknevad andmebaasis. Kui kohustuslik parameeter pole sisestatud, peab programm sellest kasutajat teavitama.

Kuna ühendus on ainuke objekt, mis omab orientatsiooni, ehk suunda, siis visuaalselt peab antud objekt näitama kus on algus ja lõpp. Lisaks, elemendid vastus ja tegevus peavad olema visualiseeritud lõuendil ka loodud ühenduse visuaalse objekti juures.

6.3.6 Visuaalsete objektide muutmine

Kuna vaadeldav modelleerimisliideses on käesoleva süsteemi andmestruktuuride redaktor, siis antud rakendus peab võimaldama tüüpjuhtumi loojal objekte ka muuta.

6.3.6.1 Küsimuse objekti muutmine

Küsimuse objekti muutmiseks peab kasutaja vajutama parema hiireklahviga objekti peal, programm peab tooma ekraanile menüü, kust kasutaja valib punkti objekti muutmiseks (*Edit*). Peale seda programm toob ekraanile akna, kus on toodud samasugune vorm, mis oli kirjeldatud antud peatükis punktis “Küsimuse objekti loomine”. Kuna vaatluse all on juba eksisteeriv objekt, siis ekraanile toodud vorm peab olema ära täidetud. Peale muutmiste tegemist, salvestab kasutaja andmed ära, programm paneb akna kinni ja uuendab tüüpjuhtumi visuaalmudeli.

6.3.6.2 Lõppoleku objekti muutmine

Lõppoleku objekti muutmine peab toimuma täiesti samal moel nagu küsimuse objekti muutmine. Toodav vorm on vastavalt kirjeldatud antud peatüki punktis “Lõppoleku objekti loomine”.

6.3.6.3 Teisele tüüpjuhtumile viite objekti muutmine

Antud objekti muutmine peab toimuma täiesti samal moel nagu küsimuse või lõppoleku objekti muutmine. Toodav vorm on vastavalt kirjeldatud antud peatüki punktis “Teisele tüüpjuhtumile viite objekti loomine”.

6.3.6.4 Ühenduse, vastuse, tegevuse objektide muutmine

Antud objektide parameetrite muutmine on sarnane eelmiste visuaalsete objektide muutmisega. Kasutaja valib ühenduse, mida ta kavatseb muuta, programm näitab visuaalselt, et objekt on valitud. Peale seda kasutaja peab vajutama objekti peal parema hiireklahviga objekti menüü kuvamiseks. Valides menüüst punkti objekti muutmiseks toob programm ekraanile vormi mis oli kirjeldatud antud peatüki punktis “Ühenduse, vastuse, tegevuse objektide loomine”. Selle vormi väljad peavad olema täidetud ühenduse objektide vastavate andmetega. Muutmisprotseduuri lõpetamiseks salvestab kasutaja andmed vajutades salvestamisnupule. Programm paneb akna kinni ja uuendab tüüpjuhtumi visuaalmudeli.

6.3.7 Visuaalsete objektide kustutamine

Kuna vaadeldav modelleerimisliideses on käesoleva süsteemi andmestruktuuride redaktor, siis antud rakendus peab võimaldama tüüpjuhtumi loojal objekte ka kustutada. Iga objekti kustutamisel kustutakse teda nii visuaalsest mudelist kui ka andmemudelist.

6.3.7.1 Küsimuse objekti kustutamine

Küsimuse objekti kustutamiseks peab kasutaja vajutama parema hiireklahviga objekti peal, programm peab tooma ekraanile menüü, kust saab leida valiku objekti kustutamiseks (*Delete*). Valiku valimisel, peab programm kasutajalt üle küsima, kas ta kindlasti soovib antud objekti ära kustutada lõuendilt ja andmemudelist. Kui tuleb eitav vastus siis midagi ei juhtu, vastasel korral kustutakse objekti ja objektiga seotud ühendused, mis on nii väljapoole kui ka sissepoole suunatud. Kustutamine toimub *cascade* moodi. Kuna iga ühendusega on seotud täpselt üks vastuse objekt ja kas null või üks tegevuse objekt, siis need kustutakse ka ära.

6.3.7.2 Lõppoleku objekti kustutamine

Lõppoleku objekti kustutamise protseduur on sarnane küsimuse objekti kustutamisega. Ainuke asi, et kuna lõppolek ei saa olla ühenduste algatajaks, siis kustutakse ainult sissepoole suunatuid ühendusi.

6.3.7.3 Teisele tüüpjuhtumile viite objekti kustutamine

Teisele tüüpjuhtumile viite objekti kustutamise protseduur on sarnane küsimuse objekti kustutamisele. Ainuke asi, et kuna tüüpjuhtumi viite objekt ei saa olla ühenduste algatajaks, siis kustutakse ainult sissepoole suunatuid ühendusi.

6.3.7.4 Ühenduse, vastuse, tegevuse objektide kustutamine

Kuna antud objektide grupp ei ole sarnane eelmiste objektidega visuaalselt, siis kustutamise algfaas toimub samamoodi nagu antud objektide muutmine: kasutaja valib ühenduse objekti ja vajutab parema hiireklahvi. Valides programmi poolt toodud menüüst valiku objekti kustutamise kohta (*Delete*), programm eemaldab visuaalsest mudelist antud ühenduse. Ühenduse kustutamisel eemaldatakse objektide vaheline seos, seega ühenduse sihtobjektini ei saa ligi vana teed kaudu.

6.3.8 Rakenduse abivahendid

Antud jaotis kirjeldab programmi teisi funktsionaalseid osi, mis ei taga rakenduse põhifunktsionaalsust, vaid pakuvad kasutajale mugavaid vahendeid töö tegemiseks.

6.3.8.1 Instrumentaalne paneel

Instrumentaalse paneeli põhifunktsiooniks on visuaalsete objektide valiku pakkumine kasutajale. Valides visuaalse objekti aktiveerib kasutaja kõik funktsioonid süsteemis, mis on seotud antud objektiga.

6.3.8.2 Objekti informatsiooni paneel

Antud paneel koondab ühte vormi kõik valitud objekti parameetrid. Valides visuaalse objekti kajastab antud paneel kogu informatsiooni antud elemendi kohta. Paneeli vormi elementideks on objekti parameetrite väärtused. Igal objektil on erinev parameetrite kogum. Näiteks, küsimusel on olemas parameeter, mis näitab, kas objekt on algusküsimus antud tüüpjuhtumis või mitte. Samas näiteks ühenduse objektil on olemas parameetrid alguspunkt ja lõpppunkt, mis näitavad kasutajale vastavalt ühenduse algusobjekti ja lõppobjekti nimetusi.

Paneel dubleerib ka visuaalsete objektide juhtimise funktsionaalsust, pakkudes kasutajale võimalust nupule vajutades avada objekti parameetrite vormi muutmiseks või kustutada objekti.

6.3.8.3 Üldinfopaneel

Paneeli otstarbeks on käesoleva rakenduse info näitamine. Kajastatud valitud visuaalne objekt, lõuendi lähenemise protsent ja hiire koordinaadid lõuendil.

6.3.8.4 Käesoleva tüüpjuhtumi infopaneel

Paneeli otstarbeks on käesoleva/muudetava tüüpjuhtumi info ning nuppude näitamine. Nuppude abil on kasutajal võimalus välja kutsuda tüüpjuhtumi muutmis- ja eemaldamisfunktsioone.

6.3.8.5 Lõuendi lähenemine

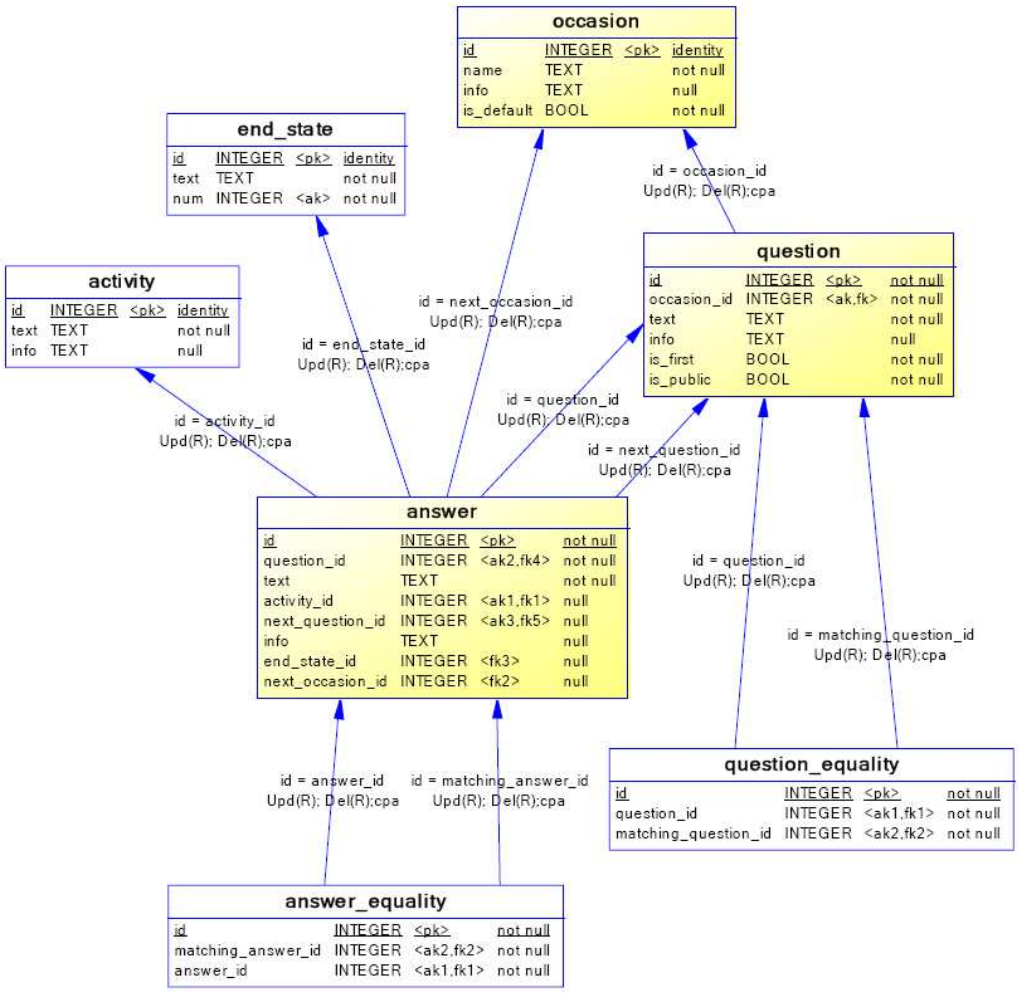
Kuna küsimustikud on tihti väga mahukad, seega ka ehitatav graafiline graaf või visuaalne mudel võib olla päris suur. Selleks, et näha koostatud mudelit üldisemast aspektist peab programm pakkuma võimalust lõuendi suurendamiseks ja vähendamiseks ehk lähenemise võimalust (*scaling*). Antud funktsionaalsus ei pea mõjutama teiste rakenduse osade funktsionaalsust.

6.3.9 Süsteemi teiste osade funktsionaalsed nõuded

Kuna süsteem koosneb paljudest osadest (vt. *Süsteemi füüsiline struktuur*), siis kõik süsteemi ülesanded on jagatud osade vahel. See tähendab, et igal osal on oma funktsionaalsed nõudmised. Kuna süsteemi projekteerimine ja arendamine oli jagatud mitmete inimeste vahel, siis igaüks oli vastutav oma osa eest, ning kirjutas oma rakenduse kohta dokumendi, kus on toodud rakenduse funktsionaalsed nõuded.

6.4 Süsteemi andmestruktuur

Kuna käesolev süsteemi modelleerimisliideses on esialgu andmete redaktor, siis ainuke rakenduse andmenõue on sobiva struktuuriga andmebaasi olemasolu. Kui see nõue on täidetud, siis rakendusel on olemas vahendid andmebaasiga ühenduse võtmiseks, andmete sinna kirjutamiseks, muutmiseks ja kustutamiseks. Andmebaasi struktuuri all on mõeldud andmebaasi objektide kooslust. Objektid on tabelid, kitsendused, välisvõtmed ja teised elemendid, mida sisaldab kooskõlas olev andmebaas. Joonis 47 visualiseerib alguses nõutava süsteemi andmebaasi skeemi.



Joonis 47: Andmebaasi skeem

Toodud andmebaasi skeem oli genereeritud esialgsest andmebaasist, töötava süsteemi andmebaas on natuke teine, sest arendamise käigus seoses funktsionaalsuse lisamisega ja programmi töö optimeerimisega arendati andmebaasi struktuuri edasi. Töötava süsteemi andmebaas on kirjeldatud ja visualiseeritud peatükis 6.6.6.

Andmebaas sisaldab endas objekte, mis on nõutud süsteemi poolt efektiivseks tööks, andmebaasi objektid on seotud omavahel välisvõtmega, andmed on kaitstud kitsenduste abil. Kollasena on märgistatud põhitabelid, kus hoitakse tähtsat informatsiooni ja valged on vähemtähtsad tabelid, kus hoitakse informatsiooni vähemtähtsamate loodavate tüüpjuhtumi elementide kohta. All toodud tabel 3 kirjeldab lühidalt iga tabeli tähtsust süsteemis.

Andmebaasi tabel	Lühikirjeldus
Occasion	Tabel, mille iga kirje kirjeldab ühte tüüpjuhtumit: tema nimetust, kirjeldust ja kas antud juhtum on algne küsimustikus või mitte
Question	Iga tabeli kirje kirjeldab ühte küsimust: tema nimetust, kirjeldust, parameetrit kas antud küsimus on algusküsimus või mitte, parameetrit kas küsimuse objekt on aktiivne või mitte. Lisaks on olemas informatsioon sellest, millisesse tüüpjuhtumisse kuulub antud küsimus
Answer	Iga tabeli kirje on iga vastuse andmed: küsimuse identifikaator, millesse kuulub vastus, vastuse tekst, vastuse kirjeldus, vastusele järgnev tegevus, vastusele järgnev küsimus, lõppoleku identifikaator (kui see on viimane küsimus tüüpjuhtumis), viite teisele juhtumile identifikaator
End_state	Tabel sisaldab küsimustiku kõigi võimalike lõppolekute informatsiooni
Activity	Tabel sisaldab kõiki võimalikke tegevusi, mis olid kasutusel küsimustikus
Answer_equality	Tabel sisaldab informatsiooni vastuste sarnasuste kohta
Question_equality	Tabel sisaldab informatsiooni küsimuste sarnasuste kohta

Tabel 3: Esialgse andmebaasi tabelite kirjeldused

Kuna rakendus nõuab täpselt selle struktuuriga andmebaasi olemasolu, siis enne modelleerimisliidese käivitamist tuleb konfigureerida andmebaas. Tehnilised aspektid on kirjeldatud järgmises peatükis.

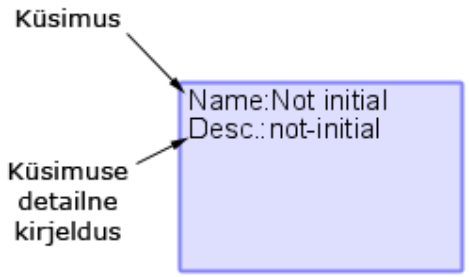
6.5 Rakenduse käitumisloogika

Antud peatükis on toodud modelleerimisliidese objektide notatsioon ning andmehaldus koos andmete hoidmise viisiga andmebaasis, toodud täpsemad reeglid ja stsenaariumid, millele rakenduse käitumine peab vastama. Aluseks on võetud rakenduse funktsionaalsed nõuded, mis on kirjeldatud samanimelises peatükis. Objektide käitumisloogika peatükid kirjeldavad objektide omadusi ning objektide haldamisfunktsioone: objekti loomine, muutmine ja kustutamine.

6.5.1 Notatsioon

Järgmises peatükis on toodud defineerimise liidese graafiliste objektide notatsioon.

- **Küsimus** – objekt on mis visualiseerib küsimuse. Sinine ristkülik sisaldab endas küsimuse nimetust („Name")ja küsimuse kirjeldust („Desc."). Iga väli omab piiranguid ristküliku pikkuse näol (vt. Joonis 58).

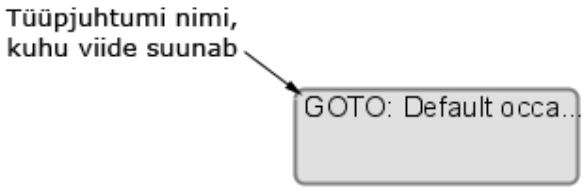


Joonis 48: Visuaalne objekt "Küsimus"

- **End State** – objekt lõppoleku visualiseerimiseks. Roheline ellips sisaldab endas lõppoleku nimetust ja kirjeldust. Samamoodi nagu küsimuse objekt siis ka lõppoleku objekti väljad omavad pikkuse piiranguid (vt. Joonis 59).

Joonis 49: Visuaalne objekt "Lõppolek"

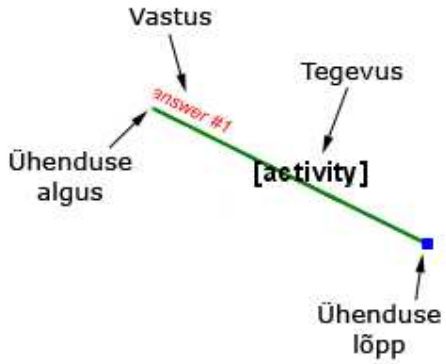
- **Other occasion** – objekt teise juhtumi viite visualiseerimiseks. Helehall ristkülik omab samu välju nagu küsimuse objekt (vt. Joonis 60).



Joonis 50: Visuaalne objekt "Viide teisele tüüpjuhtumile"

- **Answer** – küsimuse vastus. Visualiseeritud tekstina, koos ühenduse objektiga.
- **Activity –** objekt järgmise küsimuse eeltegevuse realiseerimiseks. Visualiseeritud tekstina, koos ühenduse objektiga.
- **Connection –** ühenduse objekt ülalpool toodud objektide sidumiseks. Eksisteerivad seosed:
  - Küsimus –> küsimus
  - Küsimus –> lõppolek
  - Küsimus –> teine tüüpjuhtum

Nagu on välja toodud siis ainult küsimuse objekt võib olla ühenduse allikaks. Ühenduse objekt on visualiseeritud sellisel moel: sinine joon ühendab objektide lähimaid punkte, st joone algoritm töötab lühima tee (*shortest path*) printsiibi abil. Ühenduse objekt on visualiseeritud koos vastuse ja eeltegevuse objektiga (vt. Joonis 61).



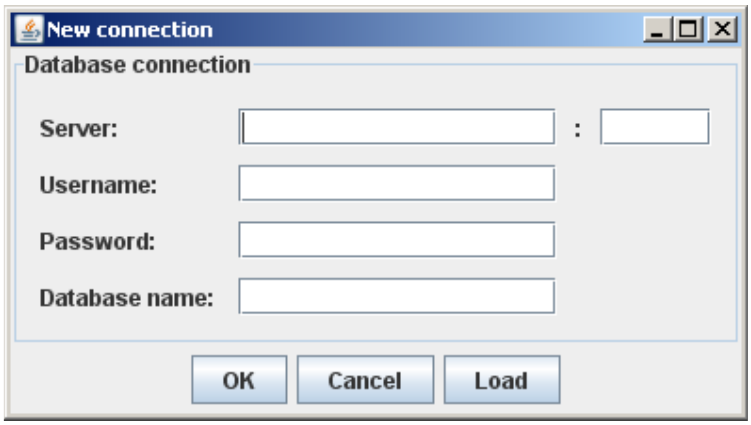
Joonis 51:Visuaalsed objektid "Ühendus, vastus ja tegevus"

6.5.2 Esimesed sammud ehk ühenduse loomine

Programmi käivitamisel, tuleb esiteks luua ühendus andmebaasiga. Selleks on programmi akna peamenüüs kasutajale valitav nupp “File” ja edasi “New Connection…”. Peale vajutamist rakendus toob ekraanile ühenduse parameetrite vormi. Ühenduse parameetrite sisestamiseks, on kasutajal olemas kaks võimalust:

- täita kogu vorm käsitsi
- laadida ühenduse parameetrid XML failist

Käsitsi parameetrite sisestamisel, on iga akna väli on kohustuslik. Kasutajale pakutakse täitmiseks järgmine vorm (vt. Ekraanipilt 1):



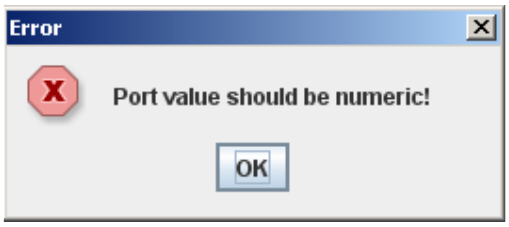
Ekraanipilt 1: Andmebaasiga ühenduse loomise vorm

- **Server** – serveri URL või ip-aadress, peale ‘:’ märki tuleb sisestada andmebaasi pordi numbri
- **Username** – andmebaasi kasutajanimi
- **Password** – kasutaja parool andmebaasis
- **Database name** – andmebaasi nimetus füüsilises serveris, mis paikneb aadressil, mis on sisestatud väljas “Server”

Kui kõik väljad on täidetud, siis nupule “OK” vajutades rakendus kontrollib, kas kõik parameetrid on täidetud ja kas parameetreid on kooskõlas. Vastavalt teavitatakse kasutajat õnnelikku ühenduse loomise kohta (vt. Ekraanipilt 2):

Ekraanipilt 2: Ühenduse loomine õnnestus

Kui pordi sisestamisel kasutaja sisestab midagi, mis pole number siis kasutajale näidatakse vastav teade (vt. Ekraanipilt 3):

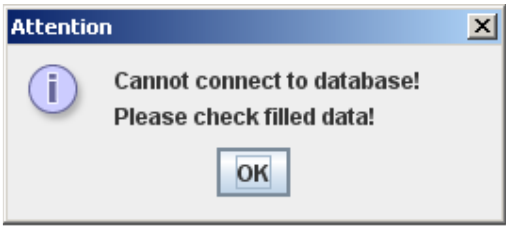


Ekraanipilt 3: Pordi väärtus pole number

Kui vorm ei ole lõpuni täidetud ja kasutaja vajutab nupule “OK”, teavitatakse kasutajat sellest (vt. Ekraanipilt 4):

Ekraanipilt 4: Kõik parameetrid peavad olema täidetud

Kui kõik parameetrid on sisestatud ja pordi numbri väljas on sisestatud number aga ühenduse loomisel tekib tehniline viga (serveri ei vasta, parool või kasutaja nimi on vale vms) teavitatakse kasutajat sellest (vt. Ekraanipilt 5):



Ekraanipilt 5: Ühenduse loomine ebaõnnestus

Kui antud teadet näidati kasutajale, siis on vaja kontrollida veelkord andmeid ja proovida uuesti.

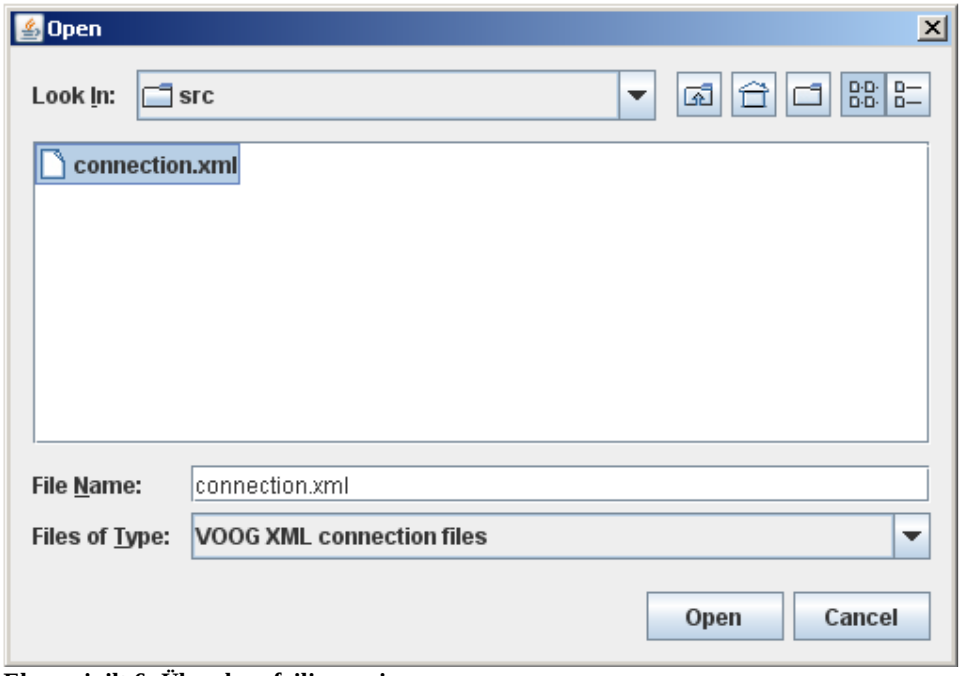
Teine viis sisestada andmebaasi andmeid on laadida neid failist. Fail peab olemas XML formaadis, ning omama konkreetset struktuuri:

```
<connection>
    <name>andmebaasi nimetus xml failis</name>
    <dbname>päris andmebaasi nimetus serveris</dbname>
    <user>kasutajanimi</user>
    <pass>parool</pass>
    <server>server ip</server>
    <port>port</port>
</connection>
```

XML element nimega <name> kirjeldab ära andmebaasi nimetuse, mida rakendus ei loe sisse seega antud parameeter on kasutaja jaoks, andmebaaside ühenduste eristamiseks. Allpool on toodud näidis andmebaasi ühenduse fail:

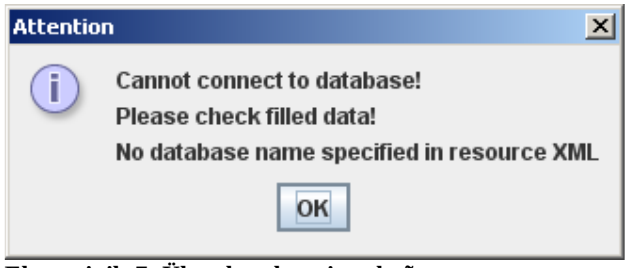
```
<connection>
    <name>VoogDB connection</name>
    <dbname>voogDB</dbname>
    <user>user_voog</user>
    <pass>pass123</pass>
    <server>194.126.111.10</server>
    <port>3306</port>
</connection>
```

Kui vastav fail on loodud ja salvestatud siis kasutajal on võimalus selle faili ja rakenduse liidese abil sisse laadida ühenduse parameetrid. Selleks kasutaja toob ekraanile ühenduse loomise vormi vajutades “File” -> “New Connection…” ja vajutab nupule “Load”. Rakendus avab uue akna, kus on pakutud kasutajale leida ühenduse fail. Peale faili leidmist, kasutaja vajutab nupule “Open” (vt. Ekraanipilt 6):



Ekraanipilt 6: Ühenduse faili avamine

Peale nupule vajutamist, hakkab rakendus kontrollima andmete konsistentsust ja vea korral teavitab sellest kasutajat (vt. Ekraanipilt 7):



Ekraanipilt 7: Ühenduse loomine ebaõnnestus

Peale seda etappi toimub ühenduse loomine samal põhimõttel nagu ühenduse loomine käsitsi vormi täitmisel. Kui ühendus on edukalt loodud, siis on kasutajal võimalus hakata küsimustikku looma. Järgmistes jaotistes on kirjeldatud visuaalsete elementide loomise rakenduse käitumisloogika põhimõtteid. Visuaalsete objektiga on seotud hulk tegevusi, mida nimetame tegevuse komplektiks. Selle komplekti hulka kuuluvad sellised tegevused nagu:

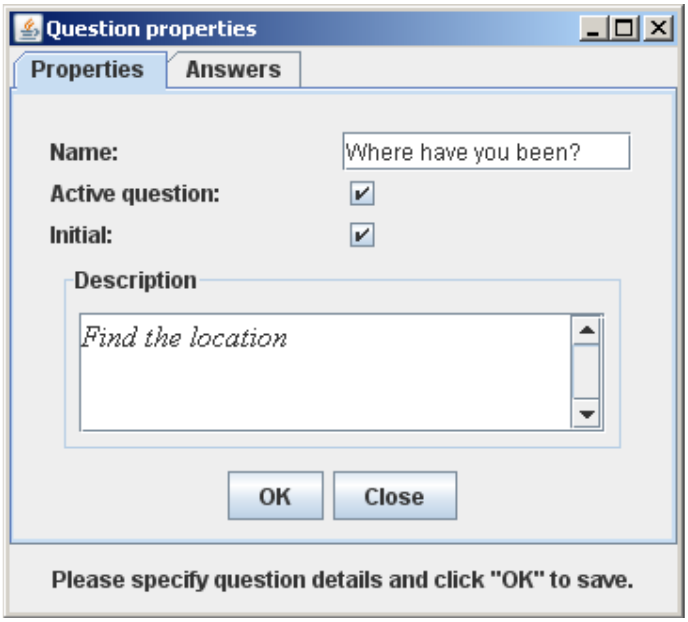
- visuaalse objekti loomine
- visuaalse objekti muutmine
- visuaalse objekti kustutamine

- visuaalse objekti liigutamine lõuendil
- visuaalse objekti mitteaktiivseks tegemine

6.5.3 Küsimuse objekti käitumisloogika

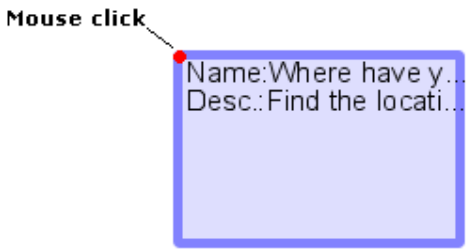
Antud peatükis on kirjeldatud küsimuse visuaalse objekti tegevuste komplekt. Kõige tähtsam tegevus on objekti loomine. Rakenduse käitumisloogika on ehitatud põhimõttel mis on kirjeldatud varem, jaotistes “Küsimuse objekti loomine”, “Küsimuse objekti muutmine” ja “Küsimuse objekti kustutamine”.

Objekti **loomiseks** kasutaja valib instrumentaalse paneeli abil objekti nimega “Question”. Järgmisena vasaku hiireklahviga vajutab ekraanile, süsteem reageerib sellele tuues ekraanile vormi (vt. Ekraanipilt 8), mis on kirjeldatud jaotises “Küsimuse objekti loomine”:



Ekraanipilt 8: Küsimuse objekti parameetrid

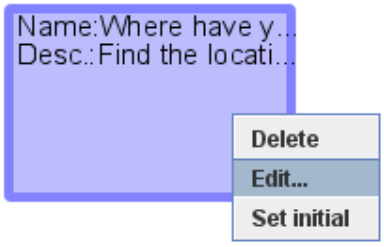
Vajutades nupule “OK” kasutaja kinnitab, et sisestatud andmed on õiged. Süsteem loob ekraanile uue küsimuse objekti niimoodi, et vasak ülemine nurk paikneb sellel kohal, kus oli registreeritud kasutaja vasaku hiireklahvi vajutamine (vt. Ekraanipilt 9).



Ekraanipilt 9: Küsimuse objekti loomise positsioon

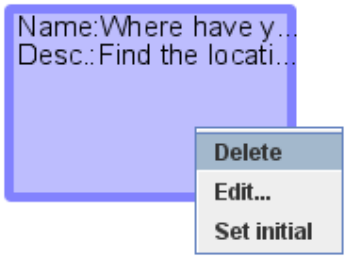
Kui kasutaja ei taha objekti luua, siis ta võib küsimuse parameetrite akna sulgeda vajutades nupule “Cancel”. Kuna küsimuse objekt on visuaalne element, ning näeb välja ristkülikuna kindla laiusega ja pikkusega ning küsimuse nimi ja kirjeldus võivad olla päris pikad, et mitte ära mahtuda objekti peale siis süsteem lõikab ära nimetuse ja kirjelduse teksti objekti joonistamisel lõuendile.

Küsimuse objekti **liigutamine** lõuendil on väga intuitiivne: kasutaja vajutab vasaku hiireklahviga küsimuse objekti ja liigub hiirega, kui objekt omab ühendusi teiste objektidega, siis liigutakse neid objekte koos käesoleva küsimuse objektiga. Seega rakendus toetab “Drag & Drop” objektide manipuleerimisprintsiipi. Küsimuse objekti **muutmise** põhimõte ja loogika on kirjeldatud peatükis “Küsimuse objekti muutmine”, ning vastav vorm on toodud selles peatükis (Ekraanipilt 8). Küsimuse objekti menüü näeb välja niimoodi (vt. Ekraanipilt 10):



Ekraanipilt 10: Küsimuse objekti menüü

Objekti kustutamine on põhjalikult kirjeldatud peatükis nimega “Küsimuse objekti kustutamine” (vt. Ekraanipilt 11):

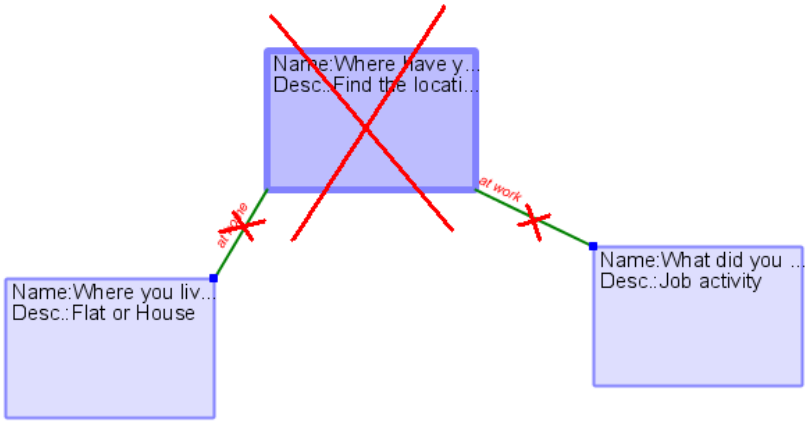


Ekraanipilt 11: Küsimuse objekti menüü

Kuna kustutamise protseduur on päris ohtlik, selles mõttes, et kustutatud andmeid ei saa taastada, küsib süsteem kasutajalt kinnituse objekti eemaldamiseks ((vt. Ekraanipilt 12)):

Ekraanipilt 12: Kinnituse küsimine

Valiku kinnitamiseks peab kasutaja valima “OK”, vastasel juhul “Cancel” nupule vajutades kogu kustutamise protseduur katkestatakse, ning lõuendi seis jääb muutmata. Kui küsimuse objekt omab ühendusi teiste objektidega, siis kõik ühendused kustutatakse ära (vt. Ekraanipilt 13).



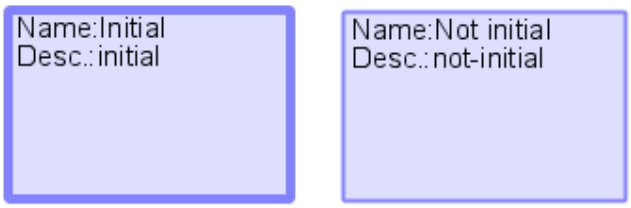
Ekraanipilt 13: Objektide hierarhiline kustutamine

Kui kasutaja tahab ajutiselt võtta küsimuse objekti küsimustikust ära, mitte aga kustutada, siis rakendus pakub funktsionaalsust, et teha objekt mitteaktiivseks. Selleks on vaja avada vorm küsimuse objekti muutmiseks, ning kustutada linnuke nimega “Active question” (vt. Ekraanipilt 8). Küsimuse objekt läheb mitteaktiivseks, ning kõik ühendused on kustutatud.

Küsimuse objekti funktsionaalsus pakub kahes kohas valikut kas küsimus on algküsimus tüüpjuhtumis või mitte. Üks koht on toodud Ekraanipildil 8, linnuke “Initial” ja teine viis määrata objekti on valida rippmenüüs punkt “Set initial” (vt. Ekraanipilt 14):

Ekraanipilt 14: Küsimuse objekti algseks objektiks määramine

Algküsimust saab eristada teistest küsimustest visuaalselt. Paks äär ütleb, et antud küsimus pole tavaline küsimus, ning temast algab tüüpjuhtumi läbivaatamine (vt. Ekraanipilt 15):

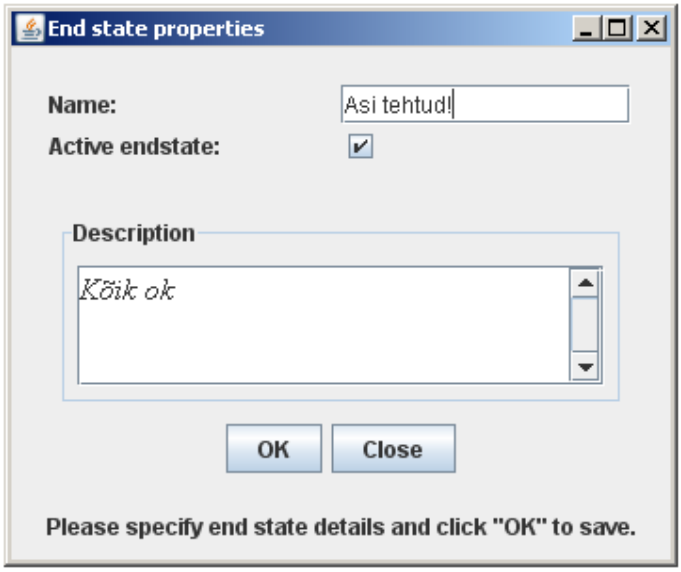


Ekraanipilt 15: Initial ja Non-initial küsimuse objektid

6.5.4 Lõppoleku objekti käitumisloogika

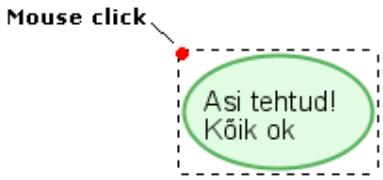
Antud peatükis on kirjeldatud lõppoleku visuaalse objekti tegevuste komplekt. Kõige tähtsam tegevus on objekti loomine. Rakenduse käitumisloogika on samamoodi nagu küsimuse objekti puhul, ehitatud põhimõttel, mis on kirjeldatud varem, jaotistes “Lõppoleku objekti loomine”, “Lõppoleku objekti muutmine” ja “Lõppoleku objekti kustutamine”.

Objekti **loomiseks** kasutaja valib instrumentaalse paneeli abil objekti nimega “End state”. Järgmisena vasaku hiireklahviga kasutaja vajutab ekraanile, süsteem reageerib sellele tuues ekraanile vormi, mis on kirjeldatud peatükis “Lõppoleku objekti loomine”. Avatud vorm on väga sarnane küsimuse objekti loomise vormile, ainult valik “Initial” on puudu, kuna lõppolek pole küsimus ja temast ei saa tüüpjuhtumit alustada (vt. Ekraanipilt 16):



Ekraanipilt 16: Lõppoleku parameetrid

Vajutades nupule “OK” kasutaja kinnitab, et sisestatud andmed on õiged. Süsteem loob ekraanile uue lõppoleku objekti niimoodi, et kui lõppoleku ümber joonistada ristkülik, siis vasak ülemine nurk paikneb sellel kohal, kus oli registreeritud kasutaja vasaku hiireklahvi vajutamine (vt. Ekraanipilt 17).



Ekraanipilt 17: Lõppoleku objekti positsiooni määramine

Kui kasutaja ei taha objekti luua, siis ta võib küsimuse parameetrite akna sulgeda vajutades nupule “Cancel”. Kuna lõppoleku objekt ei sisalda tavaliselt endas pikka nimetust, siis selle objekti jaoks ei ole sisselülitatud teksti lõikamisfunktsiooni, nagu see on tehtud küsimus objekti puhul. Lõppoleku objekti puhul positsiooni muutmine töötab täiesti samal moel nagu ka teiste visuaalsete põhiobjektide puhul, mille hulgas on ka küsimuse objekt, mille kohta on antud loogika ära kirjeldatud: kasutaja valib objektiks “End state” ja kasutab “Drag & Drop” printsiipi objekti **asukoha muutmiseks**.

Lõppoleku objekti **muutmise** põhimõte ja loogika on kirjeldatud jaotises “Lõppoleku objekti muutmine”, ning vastav vorm on toodud selles jaotises Ekraanipildil 16. Lõppoleku objekti menüü on toodud Ekraanipildil 18.

Ekraanipilt 18: Lõppoleku objekti menüü

Objekti kustutamine on põhjalikult kirjeldatud peatükis nimega “Lõppoleku objekti kustutamine” (vt. Ekraanipilt 19):

Ekraanipilt 19: Lõppoleku objekti menüü

Kuna kustutamise protseduur on päris ohtlik, selles mõttes, et kustutatud andmed ei saa taastada, küsib süsteem kasutajalt kinnituse objekti eemaldamiseks (vt. Ekraanipilt 20):

Ekraanipilt 20: Kinnituse küsimine

Valiku kinnitamiseks peab kasutaja peab valima “OK”, vastasel juhul “Cancel” nupule vajutades kogu kustutamise protseduur katkestatakse ning lõuendi seis jääb muutmata.

6.5.5 Teisele tüüpjuhtumile viite objekti käitumisloogika

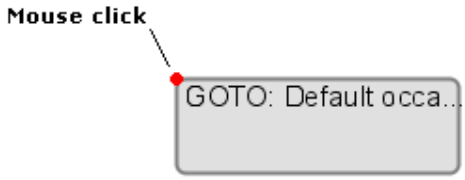
Antud peatükis on kirjeldatud teisele tüüpjuhtumile viite visuaalse objekti tegevuste komplekt. Kõige tähtsam tegevus on objekti loomine. Rakenduse käitumisloogika on ehitatud põhimõttel, mis on kirjeldatud varem, jaotistes “Teisele tüüpjuhtumile viite objekti loomine”, “Teisele tüüpjuhtumile viite objekti muutmine” ja “Teisele tüüpjuhtumile viite objekti kustutamine”.

Objekti **loomiseks** valib kasutaja instrumentaalse paneeli abil objekti nimega “Occasion”. Järgmisena vasaku hiireklahviga vajutab ekraanile, süsteem reageerib sellele tuues ekraanile dialoogiakna, mis on kirjeldatud peatükis “Teisele tüüpjuhtumile viite objekti loomine” (vt. Ekraanipilt 21):

Ekraanipilt 21: Tüüpjuhtumi valimise tabel

Toodud tabeli (vt. Ekraanipilt 21) iga rida sisaldab endas ühe tüüpjuhtumi vajalikku informatsiooni. Veerg “ID” sisaldab tüüpjuhtumi andmebaasi unikaalset identifikaatorit, veerg “Name” – tüüpjuhtumi nimetust, veerg “Description” – tüüpjuhtumi kirjeldust ja veerg “Last modified” näitab tüüpjuhtumi viimase salvestamise aega kui tüüpjuhtumit muudeti ja loomise aega kui tüüpjuhtumit ei ole veel kunagi muudetud. Tabeli veerud on interaktiivsed selles mõttes, et vajutades veeru nimele saab kogu tabeli sorteerida vastavalt selle veeru väärtustele kasvavalt või kahanevalt. Antud funktsionaalsus on väga kasulik juhul kui tüüpjuhtumite arv on väga suur ja on vaja, sorteerida tüüpjuhtumeid viimase muutmise aja järgi.

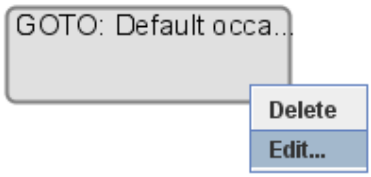
Valides sobiva tüüpjuhtumi rea vajutab kasutaja nupule “OK”, programm peidab valikuakna ära ja joonistab tüüpjuhtumi viite objekti lõuendile. Süsteem loob ekraanile uue küsimuse objekti niimoodi, et vasak ülemine nurk paikneb sellel kohal kus oli registreeritud kasutaja vasaku hiireklahvi vajutamine (vt. Ekraanipilt 22).



Ekraanipilt 22: Objekti positsiooni määramine

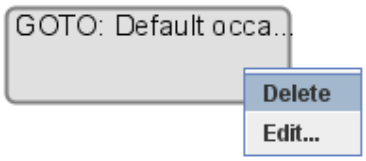
Kuna tüüpjuhtumi viite objekt on visuaalne element ning näeb välja riskülikuna kindla laiusega ja pikkusega ning tüüpjuhtumi nimi võib olla pikem kui objekti peale mahub, sellisel juhul lõikab süsteem objekti lõuendile joonistamisel nimetuse teksti ära.

Objekti **asukoha muutmiseks** peab kasutaja peab kasutama seda sama printsiipi nagu teiste objektide juures: "Drag & Drop" valides enne liikumist objektide hulgast "Occasion" objekti tüübi. Nagu igat visuaalset objekti saab viidet tüüpjuhtumile muuta ja kustutada. **Muutmine** on kirjeldatud peatükis "Teisele tüüpjuhtumile viite objekti muutmine" ja rakenduse poolt avatud dialoogaken on näha Ekraanipildil 21. Viite objekti menüü on toodud Ekraanipildil 23.



Ekraanipilt 23: Objekti menüü

Kustutamine on kirjeldatud peatükis "Teisele tüüpjuhtumile viite objekti kustutamine" vajutades parema hiireklahviga objektil avaneb menüü, kust kasutaja valib punkti "Delete", ja peale kinnituse näitamist ning kasutaja poolt positiivse vastuse saamisest kustutab rakendus antud objekti igaveseks (vt. Ekraanipilt 24 ja 25).



Ekraanipilt 24: Objekti menüü

Ekraanipilt 25: Kinnituse küsimine

Nagu teiste objektide puhulgi, toob objekti kustutamine kaasa objekti ühenduste kustutamise. Lisaks ei oma tüüpjuhtumi objekt parameetrit aktiine/mitteaktiivne, kuna antud objekt on peegeldus olemasolevast tüüpjuhtumist.

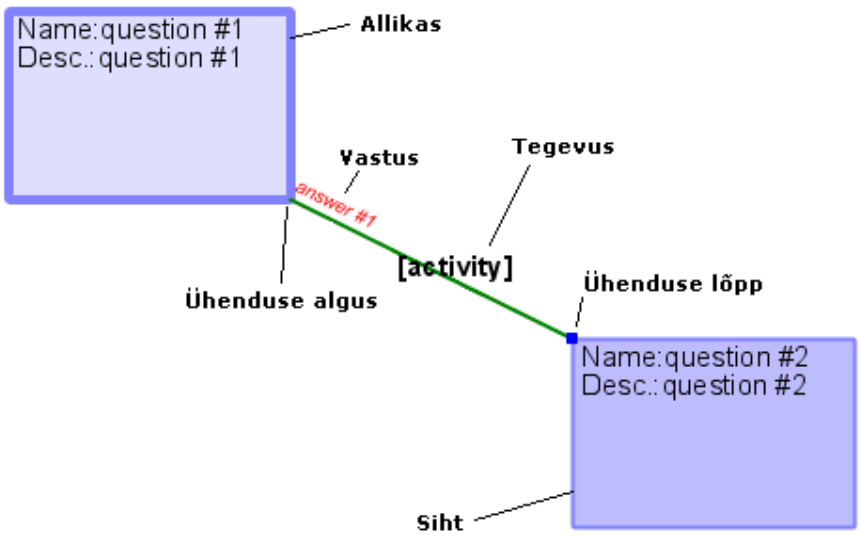
Järgmises peatükis on kirjeldatud teiste visuaalsete objektide käitumisloogika, mis ei ole sarnane ülaltoodud objektide käitumisega, kuna need objektid ei saa eksisteerida ilma põhiobjektideta (küsimus, lõppolek, viide tüüpjuhtumile).

6.5.6 Ühenduse, vastuse, tegevuse objektide käitumisloogika

Antud objektid kuuluvad niiöelda lisaobjektide gruppi, kuna need ei ole põhiobjektid sel põhjusel, et nad ei saa eksisteerida üksinda. Sellesse gruppi kuuluvad:

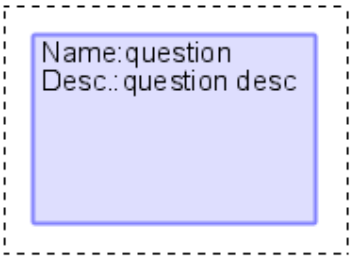
- Küsimuse vastuse objekt
- Tegevuse objekt
- Ühendus ise

Põhiobjekt nendest objektidest on *ühendus*, kuna ühenduse peal paiknevad nii vastuse objekt kui ka tegevuse objekt. Viimane aga pole kohustuslik, kuna ühelt küsimusest teisele minnes ei ole vastajal vaja alati lisategevusi täita. Järgmine Ekraanipilt 26 illustreerib objektide koostööd:



Ekraanipilt 26: Ühenduse objekt

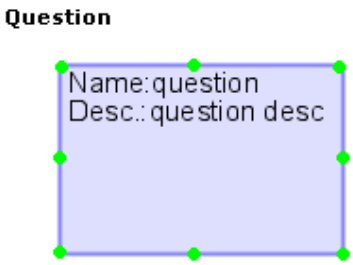
Antud gruppi objektide loomine käib printsiibil, mis on kirjeldatud jaotises “Ühenduse, vastuse, tegevuse objektide loomine”. Nagu oli mainitud on objektide loomiseks vaja vähemalt kahte objekti, kus üks neist on allikas ja teine on sihtobjekt. Kasutaja valib objektide nimekirjast objekti nimega “Connection”, vajutab vasaku hiireklahviga objekti peale ning kasutades “Drag & Drop” tehnikat tõmbab ühenduse sihtobjektini ning vabastab hiireklahvi objekti peal või objekti juures. Kuna rakendus omab “magnetic” funktsionaalsust, mis aitab kasutajat kui ühenduse ots hüppab sihtobjektile, siis antud operatsioon ei ole eriti raske. Järgmine ekraanipilt illustreerib objekti töötsooni, kus hakkab antud “magnetic” funktsioon töötama (vt. Ekraanipilt 27):



Ekraanipilt 27: Küsimuse objekti "magnetic" funktsiooni töötamise ala

Rakendus liimib ühenduse otsa konkreetsele punktile, mis asub visuaalse objekti ääre peal. Igal põhiobjektil on erinev arv ühenduspunkte. Järgmistel joonistel on toodud iga visuaalse põhiobjekti ühenduspunktid:

- Küsimus – omab 8 ühendus punkti (vt. Ekraanipilt 28)

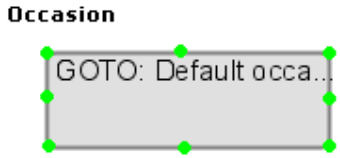


Ekraanipilt 28: Küsimuse objekti ühendus punktid

- Lõppolek – tänu oma kujule omab antud objekt ainult 4 ühenduspunkti (vt. Ekraanipilt 29)

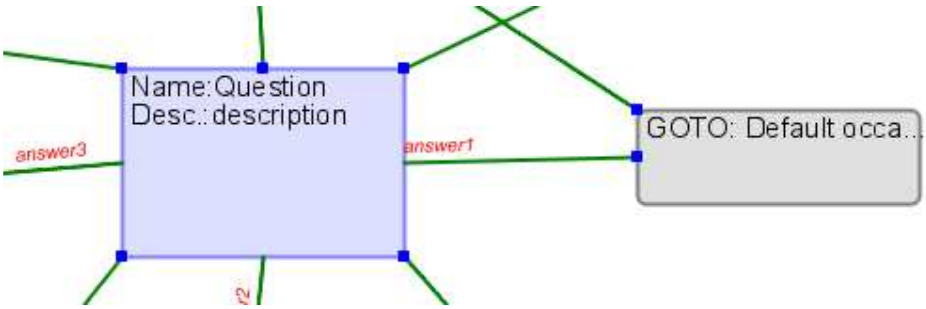
Ekraanipilt 29: Lõppoleku objekti ühendus punktid

- Viide tüüpjuhtumile – kuna antud objekt omab sarnast vormi küsimuse objektiga on tal samamoodi 8 ühenduspunkti (vt. Ekraanipilt 30)



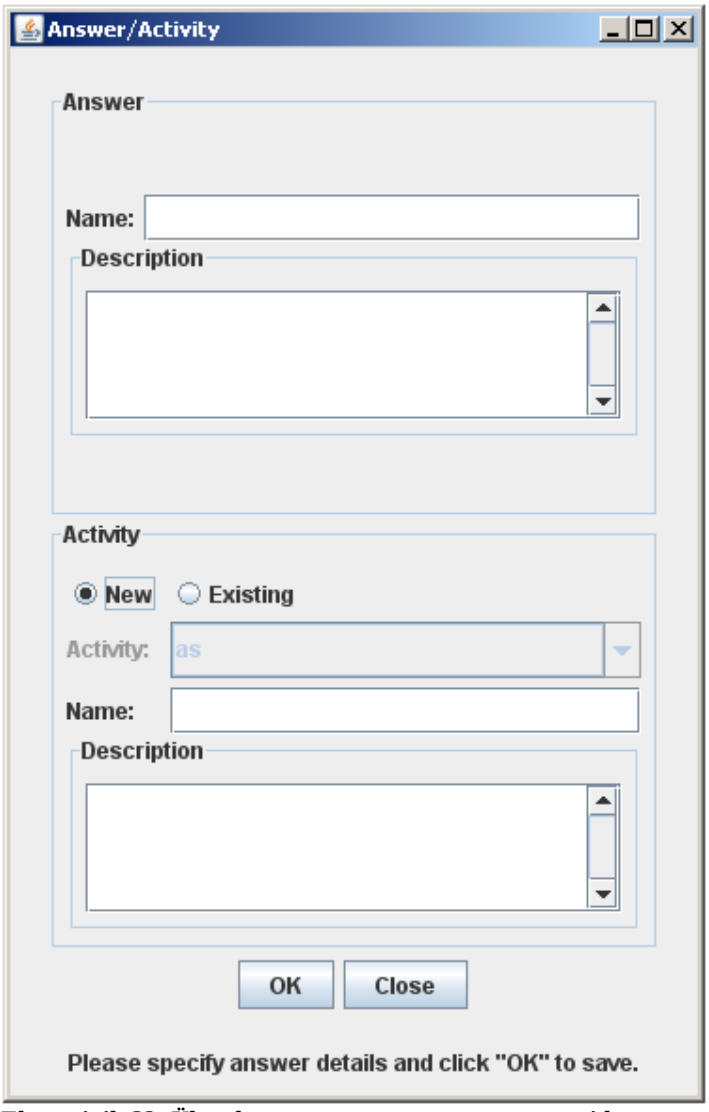
Ekraanipilt 30: Viite objekti ühenduspunktid

Ühenduspunkte kasutatakse nii väljapoole suunatud kui ka sissepoole suunatud ühenduste jaoks. Järgmine ekraanipilt illustreerib küsimuse ja tüüpjuhtumi objekti abil ühenduse võimalusi (vt. Ekraanipilt 31):



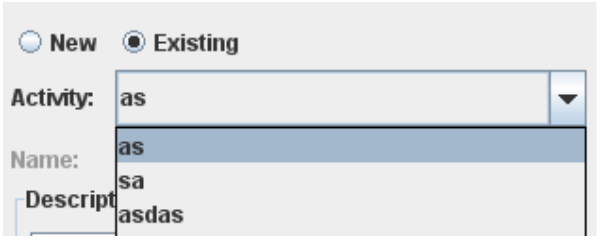
Ekraanipilt 31: Küsimuse ja teise objekti ühenduse võimalused

Ühenduse **loomisel** rakendus kontrollib, kas antud kaks objekt on juba ühendatud või mitte. Kui ühendust veel pole korral, objektid ühendatakse, vastasel juhul teavitatakse kasutajat ühenduse ebaõnnestumisest. Kui kaks objekti on edukalt ühendatud, pakub rakendus kasutajale täitmiseks nii ühenduse kui ka vastuse ja tegevuse objektide parameetrid. Järgmine ekraanipilt illustreerib vormi, mis on kirjeldatud peatükis “Ühenduse, vastuse, tegevuse objektide loomine” (vt. Ekraanipilt 32):



Ekraanipilt 32: Ühenduse, vastuse, tegevuse parameetrid

Vorm on jagatud kahte ossa, üks osa on vastuse (*answer*) ja teine osa on tegevuse (*activity*) parameetrid. Kasutajale pakutakse valida kas kahest viisist, kuidas luua tegevuse objekt, kas luua uus objekti või valida olemasolevate hulgast. Uue objekti puhul valib kasutaja raadionuppudest “New” valiku, mis informeerib, et kasutaja hakkab sisestama uue tegevuse objekti andmeid. Kui on valitud “Existing”, siis rakendus aktiveerib olemasolevate tegevuste valiku nimega ”Activity” (vt. Ekraanipilt 33).

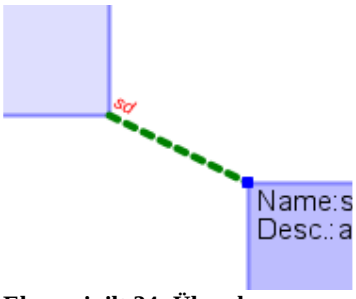


Ekraanipilt 33: Olemas oleva tegevuse valimine

Antud valik (*combo box*) sisaldab endas kõiki tegevusi mis eksisteerib andmebaasis. See tähendab, et mitte ainult käesoleva tüüpjuhtumi omi. Antud lähenemine on realiseeritud sellepärast, et tüüpjuhtumite loomisel on kasutajale tihti vaja nn. üldiseid tegevusi nagu "Puhka natuke", "Loe juhendit" ja nii edasi. Seega on mõistlik kasutajale pakkuda kogu tegevuste komplekti ilma spektri piiranguteta. Vajutades nupule "OK" saab rakendus teada, et kasutaja on lõpetanud parameetrite täitmise ning hakkab andmeid analüüsima. Ainuke kohustuslik väli antud vormil on vastuse nimetus ehk. vastus ise. Kui antud väli on tühi siis süsteem informeerib sellest kasutajat kirjutades vormi lõpus vastava teate. Kui tegevuse tüübiks on valitud:

- "New" (ehk uus) - rakendus loeb sisse andmed mis olid täidetud väljades "Name" ja "Description" *Activity* osas. Kui "Name" väli on tühi siis süsteem loeb, et antud objektide vahelisel ühendusel puudub tegevuse objekt. Kui vähemalt "Name" väli pole tühi siis süsteem loob uue tegevuse objekti ja joonistab selle ekraanile.
- "Existing" (ehk olemasolev) – rakendus loeb andmebaasist andmed selle olemasoleva tegevuse objekti kohta mis oli valitud "Activity" valikus (vt. Ekraanipilt 33).

Peale edukat andmete analüüsi ja kontrolli joonistab rakendus lõuendile visuaalsed objektid: ühenduse, vastuse ja tegevuse (kui eksisteerib). Antud objektide kogumi **muutmiseks** tuleb hiirega minna ühenduse objekti peale ning kui ühenduse objekt muutub katkendjooneks (vt. Ekraanipilt 34) siis see tähendab, et objekt on valitud ning paremat hiireklahvi vajutades ja valiku "Edit" valides toob rakendus ekraanile vormi objektide parameetrite muutmiseks (vt Ekraanipilt 32), kus vormi väljad on täidetud varem sisestatud objekti parameetritega. Peale muutmise lõpetamist on kasutajal võimalik kinnitada andmed vajutades "OK" nupule või panna muutmisvorm kinni ilma muutuste sisse toomiseta.

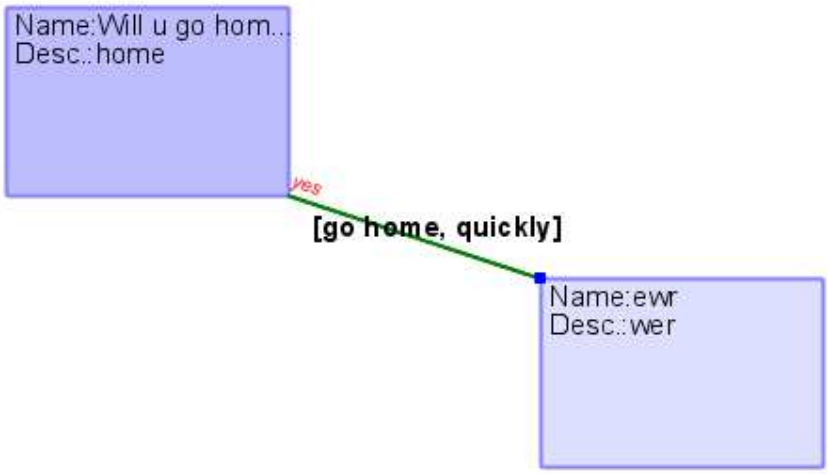


Ekraanipilt 34: Ühenduse objekti valimine

Ühenduse, vastuse ja tegevuse **kustutamiseks** on tarvis valida ühenduse objekt ja hiire paremat klahvi vajutades valida menüüst valik “Delete”. Süsteem küsib kinnitust ning positiivse vastuse korral eemaldab ühenduse, vastuse ja tegevuse objektid lõuendilt. Kuna tegevus ei ole kohustuslik objekt siis rakendus lubab kasutajal selle objekti eraldi kustutada. Selleks tuleb avada ühenduse objekt muutmiseks, valida tegevuse osas tegevuse tüüp nagu “New” ning kustutada välja “Name” sisu. Vajutades “OK” nupule kustutab rakendus tegevuse objekti lõuendilt.

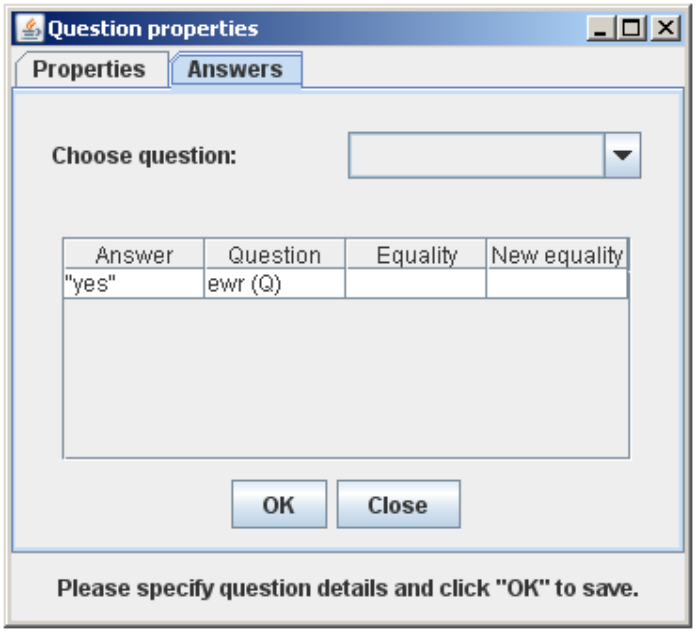
6.5.7 Küsimuste ning vastuste sarnasus

Küsimused võivad olla sarnased põhimõtte poolest. See tähendab seda, et iga küsimust saab mitut viisi. Näiteks küsimuse “Kas oled naine?” saab ümber fraseerida küsimuseks “Kas sa mees ei ole?”. Nagu näha on küsimuse põhimõte sama. Nii nagu küsimuse puhul on küsimuse vastuseid võimalik samamoodi sõnastada ümber. Näiteks vastus “jah” küsimusele “Kas sa juhid autot?” on sarnane vastusega “ei” küsimusele “Kas sa kõnnid iga päev kodust tööle ja tagasi?”. Need vastused on sarnased ning esitatud küsimused on sarnased, see tähendab, et sarnase mõttega kuna mõlemad küsimused küsivad vastaja käest tema auto olemasolu kohta. Süsteemis on küsimuste ja vastuste sarnasuse funktsionaalsus. Modelleerimisliides sarnaste vastuste määramiseks on lihtne. Esiteks on vaja minimaalselt kaks objekti ning ühendust nende vahel. Algusobjektiks peab olema küsimuse objekt, kuna ainult see objekti tüüp võib olla ühenduse algatajaks. Järgmine Ekraanipilt 35 demonstreerib võimalikke seisundeid:



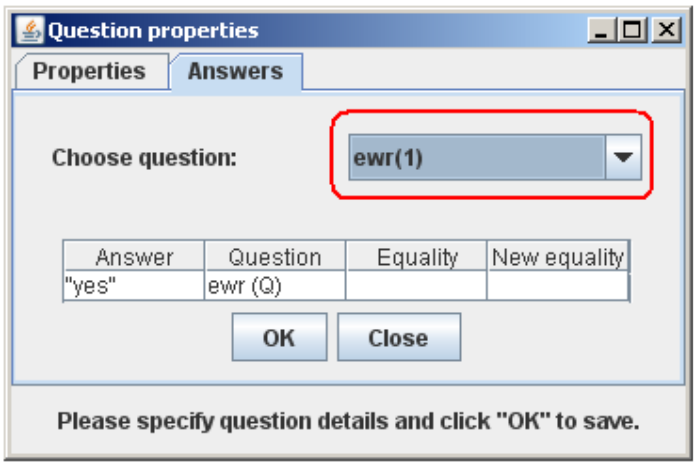
Ekraanipilt 35: Suvaline seisund

Avades ühenduse allikaobjekti parameetrid vajutades objektile parema hiireklahviga ja valides menüüs punkti “Edit” toob rakendus ekraanile küsimuse objekti parameetrite vormi kus sarnaste vastuste määramiseks vajutab kasutaja *taabi* nimega “Answers” (vt. Ekraanipilt 36):



Ekraanipilt 36: Sarnaste vastuse määramine

Esiteks peab kasutaja valima küsimuse mis on antud küsimusega sarnane. See on võimalik valides valikus nimega “Choose question” sobiva küsimuse (vt. Ekraanipilt 37):

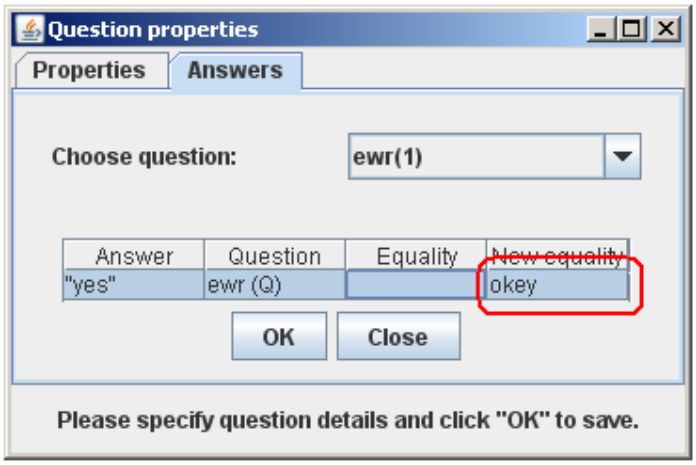


Ekraanipilt 37: Sarnase küsimuse valimine

Antud valikus on iga element esitatud küsimuse nimega ja sulgudes on selle küsimuse vastusevariantide arv. Tabeli veerud on järgmised:

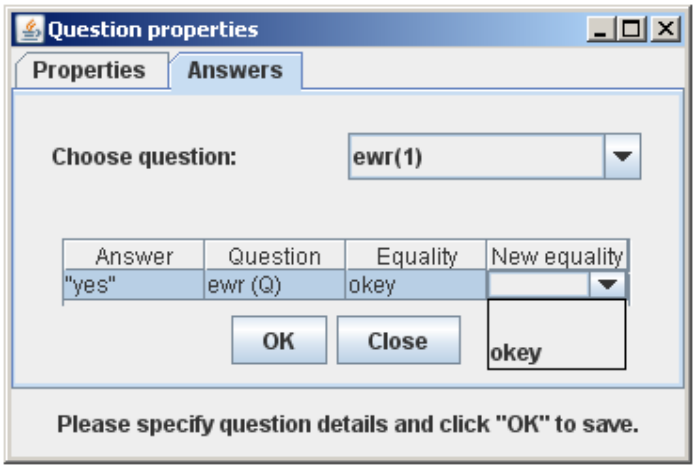
- Answer – küsimuse vastus
- Question – kuhu vastaja suunatakse kui vastaja valib selle valiku
- Equality – vastus, millega käesolev vastus on sarnane
- New equality – lahter/valik, mis sisaldab võimalikke vastuseid

Järgmisena peab kasutaja allolevas tabelis vajutama tühjale lahtrile veerus nimega “New equality” ja valima sobiva vastuse (vt. Ekraanipilt 38):



Ekraanipilt 38: Sarnase vastuse määramine

Vastuse ja küsimuse sarnasuse salvestamiseks peab kasutaja vajutama “OK” nupule. Sarnasuse muutmiseks tuleb veelkord avada küsimuse parameetrite vorm, vajutada “Answers” *taabi*, muuta vajadusel sarnane küsimus (valik “Choose question”), valida “New equality” veerul uus vastus ja vajutada “OK” muudatuste kinnitamiseks (vt. Ekraanipilt 39):



Ekraanipilt 39: Sarnase vastuse muutmine

6.5.8 Rakenduse abikomponentide käitumisloogika

Käesolevas peatükis on kirjeldatud rakenduse abikomponentide elemente ja nende eesmärke. Abikomponentide hulka kuuluvad:

- Instrumentaalne paneel
- Objekti informatsiooni paneel
- Üldinfopaneel
- Peamenüü
- Tüüpjuhtumi infopaneel

Instrumentaalse paneeli (vt. Ekraanipilt 40) eesmärgiks on kasutajale visuaalsete objektide valiku pakkumine. Paneel koosneb neljast nupust, kus iga nupp illustreerib objekti tüüpi. Nupud on järgmised:

- Question (küsimus) – sellele nupule vajutades muutub küsimuse objekt süsteemi käesolevalt valitud objektiks. Nüüd kõik tegevused nii lõuendil kui ka kogu

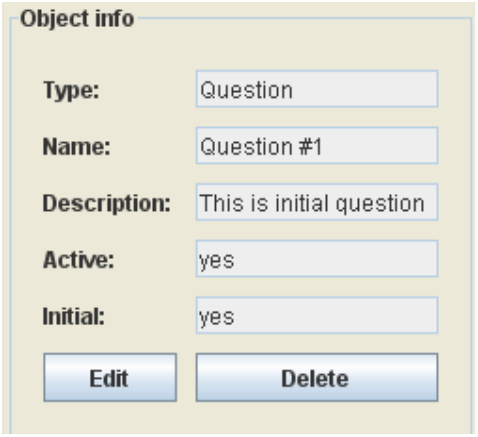
rakendusel on seotud küsimuse objektiga. Näiteks valides küsimuse objekti vajutades vasaku hiireklahviga lõuendil alustab süsteem visuaalse küsimuse objekti loomist.

- End state (lõppolek) - sellele nupule vajutades muutub lõppoleku objekt süsteemis käesolevalt valitud objektiks. Nüüd on kõik tegevused nii lõuendil kui ka kogu rakenduses seotud lõppoleku objektiga. Näiteks saab kasutaja lõppolekut looma hakata.
- Occasion (viide teisele tüüpjuhtumile) - sellele nupule vajutades muutub viite objekt süsteemis käesolevalt valitud objektiks. Nüüd on kõik tegevused nii lõuendil kui ka kogu rakenduses seotud teisele tüüpjuhtumile viite objektiga. Kasutajal aktiveeruvad võimalused teisele tüüpjuhtumile viite loomiseks ja haldamiseks.
- Connection (ühendus) - sellele nupule vajutades muutub ühenduse objekt süsteemis käesolevalt valitud objektiks. Nüüd on kõik tegevused nii lõuendil kui ka kogu rakenduses seotud ühenduse objektiga. Näiteks on kasutajal aktiveerunud võimalus ühendada kahte objekti ja luua tegevuse objekt käesoleva ühenduse objekti külge.

Käesoleva paneeli visuaalne skeem on toodud ekraanipildil 40. Mitteaktiivne nupp informeerib meid sellest, et see objekt on parajasti valitud ja süsteemis on aktiveerunud funktsioonid, mis on seotud selle objektiga:

Ekraanipilt 40: Instrumentaalne paneel

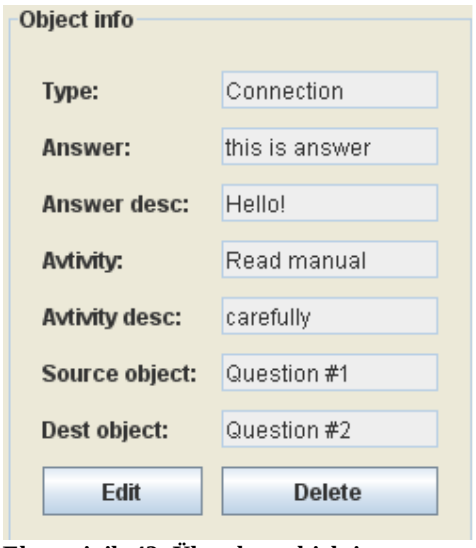
Objekti informatsiooni paneeli (vt. Ekraanipilt 41) rolliks on valitud visuaalse objekti info näitamine. Teiste sõnadega kui kasutaja vajutab vasaku hiireklahviga visuaalse objekti peale siis kogu objekti info on toodud antud paneelil ja informatsioon muutub alles siis, kui kasutaja valib teise objekti. Järgmine ekraanipilt illustreerib suvalise küsimuse objekti informatsiooni paneeli sisu:



Ekraanipilt 41: Objekti informatsiooni paneel

Paneel näitab infot kõikidest parameetritest, mis on toodud objekti loomise ja muutmise vormil. Alumised nupud dubleerivad seda funktsionaalsust, mis on kirjeldatud iga objekti muutmise ja kustutamise jaotistes: nupp “Edit” on samaväärne sellega, kui kasutaja vajutab objekti peale parema hiireklahviga ja valib menüüs valiku “Edit…” ehk toob ekraanile vormi objekti parameetrite muutmiseks; samamoodi nupp “Delete” dubleerib objekti kustutamise funktsionaalsust.

Kuna erinevad objektid omavad erinevaid parameetrid, siis antud paneel näitab erinevaid välju iga objekti kohta ning lisaks ka seda infot mida kasutaja otseselt muuta ei saa, näiteks ühenduse algus- ja lõppobjektide nimetused (vt. Ekraanipilt 42):



Ekraanipilt 42: Ühenduse objekti informatsiooni paneel

Üldinfopaneeli ülesandeks on näidata rakenduses käesolevalt valitud objekti, mille kohta on rakenduses aktiveeritud vastavad funktsioonid: hiire kursori koordinaadid lõuendil ja lõuendi *skaleerimisprotsent* (vt. Ekraanipilt 43):

Ekraanipilt 43: Üldinfopaneel

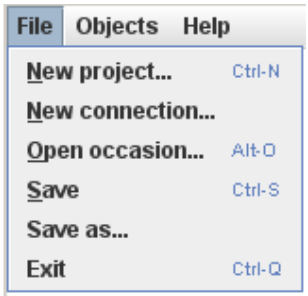
Peamenüü pakub (vt. Ekraanipilt 44) kasutajale üldiseid rakenduse halduse funktsioone. Need on küsimustiku manipuleerimisfunktsioonid, objektide valik ja üldinformatsioon. Menüü koosneb kolmest põhivalikust:

- "File" – siin on toodud funktsioonid tüüpjuhtumi salvestamiseks ja üleslaadimiseks ning uue projekti loomiseks
- "Objects" – antud menüüpunkt on duplikaat rakenduse instrumentaalsest paneelist, mis pakub kasutajale visuaalsete objektide tüüpi valikut
- "Help" – rakenduse informatsioon

Menüü "File" koosneb järgmistest komponentidest:

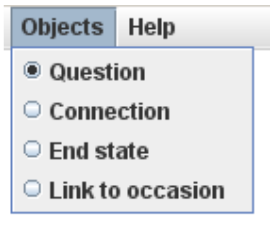
- "New project…" – uus küsimustik - algväärtustab kogu rakenduse parameetrid ning teeb lõuendi puhtaks

- “New connection…” – aktiveerib uue andmebaasiühenduse loomise protsessi (vt. pt. 5.2.1 Esimesed sammud ehk. Ühenduse loomine)
- “Open occasion” – aktiveerib tüüpjuhtumi andmebaasist üleslaadimisprotsessi (vt. pt. 3.4 Tüüpjuhtumi üleslaadimine ning pt. 5.2.8 Tüüpjuhtumi käitumisloogika)
- “Save” – salvestab muudetud tüüpjuhtumi. Kui tüüpjuhtum pole veel loodud siis alustab tüüpjuhtumi salvestamist (vt. pt. 3.3 Tüüpjuhtumi salvestamine ning pt. 5.2.8 Tüüpjuhtumi käitumisloogika)
- “Save as…” – alustab kohe uue tüüpjuhtumi salvestamisprotseduuri. Erinevus “Save” funktsiooniga on selles, et antud funktsioon alati küsib kasutajalt uue tüüpjuhtumi parameetrid aga “Save” funktsioon uuendab olemasolevat tüüpjuhtumit, mis enne oli kas üles laetud või salvestatud andmebaasi.
- “Exit” – rakenduse töö lõpetamine



Ekraanipilt 44: Rakenduse peamenüü

Menüü “Objects” (vt. Ekraanipilt 45) koosneb raadionuppude objektide valikust. Antud menüüpunkt kopeerib rakenduse instrumentaalse paneeli funktsionaalsust, selle abil saab määrata lõuendi põhiobjekti:

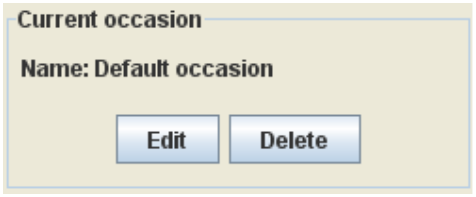


Ekraanipilt 45: "Objects" menüüpunkti sisu

Menüü “Help” sisaldab endas menüü punkti “About” mida valides rakendus näitab programmi ja autori informatsiooni (vt. Ekraanipilt 46):

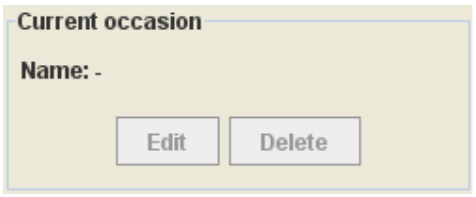
Ekraanipilt 46: Rakenduse ja autori informatsioon

Tüüpjuhtumi infopaneel (vt. Ekraanipilt 47) on mõeldud tüüpjuhtumiga seotud info ja funktsionaalsuse hoidmiseks: käesoleva tüüpjuhtumi nimetus, kirjeldus ja funktsionaalsus käesoleva tüüpjuhtumi muutmiseks ja kustutamiseks. Järgmine ekraanipilt illustreerib tüüpjuhtumi infopaneeli:



Ekraanipilt 47: Täidetud tüüpjuhtumi infopaneel

Nuppud “Edit” ja “Delete” vastavalt avavad vormi tüüpjuhtumi muutmiseks või kustutavad tüüpjuhtumi andmebaasist. Juhtub ka nii, et antud paneel ei näita tüüpjuhtumi nime ja nupud on mitteaktiivsed (vt. Ekraanipilt 48):



Ekraanipilt 48: Tühi tüüpjuhtumi infopaneel

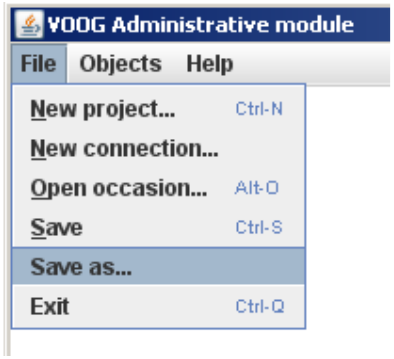
See tähendab seda, et tüüpjuhtum pole loetud andmebaasist või antud visuaalsete objektide struktuur, mis paikneb lõuendil pole veel salvestatud andmebaasi.

Tüüpjuhtumi salvestamise, muutmise ja üleslaadimise protseduurid on kirjeldatud järgmises peatükis.

6.5.9 Tüüpjuhtumi käitumisloogika

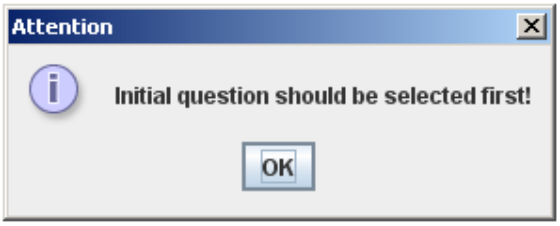
Loodav süsteem opereerib tüüpjuhtumitega. Iga tüüpjuhtum sisaldab endas objekte, mis on omavahel seotud loogiliste ühendustega. Nagu kõiki objektide saab tüüpjuhtumeid luua, muuta ja kustutada.

Tüüpjuhtumi **loomine** toimub siis, kui kasutaja lõpetab visuaalsete objektide puu tegemise ja üritab salvestada kogu struktuuri andmebaasi. Valides peamenüüs “File” -> “Save as…” (vt. Ekraanipilt 49) initsialiseerib kasutaja tüüpjuhtumi loomise ja andmebaasi salvestamise protseduuri:



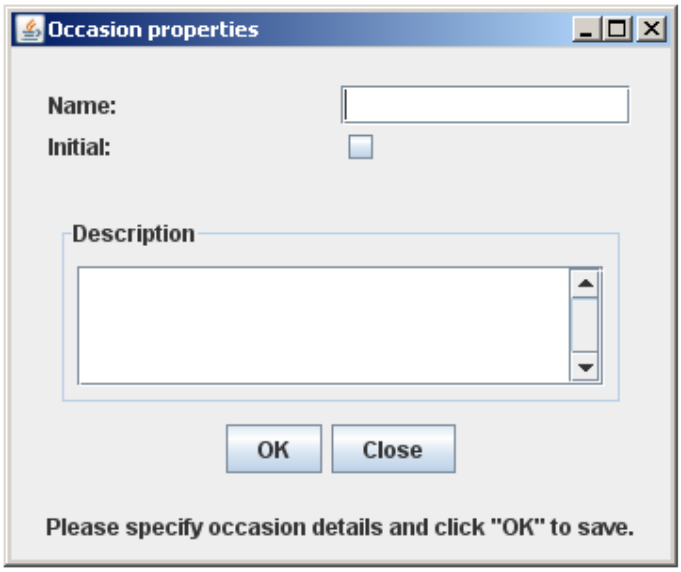
Ekraanipilt 49: Peamenüü "Save as..." punkt

Esiteks kontrollib rakendus, kas visuaalsete objektide hulgas on olemas küsimus, mis on märgitud algküsimuseks. Kui sellist ei leita katkestab rakendus salvestamisprotseduuri ja teavitab kasutajat veast (vt. Ekraanipilt 50):



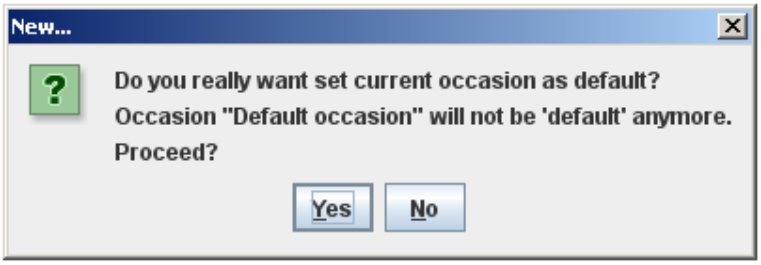
Ekraanipilt 50: Alguskü simus pole valitud

Vea korral toob rakendus ekraanile tüüpjuhtumi loomise vormi, kus on toodud tüüpjuhtumi parameetrid. Parameeter “Name” kirjeldab käesoleva tüüpjuhtumi nime ning parameeter “Initial”, mis annab võimaluse märkida tüüpjuhtumit algtüüpjuhtumiks kõigi tüüpjuhtumite seast. See tähendab, et antud tüüpjuhtumiga alustatakse küsimustiku täitmist. Parameeter “Description” annab võimaluse salvestada tüüpjuhtumi kirjeldus. Järgmine Ekraanipilt 51 illustreerib antud vormi:



Ekraanipilt 51: Tüüpjuhtumi loomise vorm

Vajutades nupule “OK” hakkab rakendus hakkab olevat struktuuri andmebaasi salvestama. Kui parameeter “Initial” on valitud, siis kontrollib rakendus, kas andmebaasis eksisteerib juba tüüpjuhtum, mis on märgitud algtüüpjuhtumiks. Kui vastus on positiivne, siis rakendus küsib kasutajalt kinnitust, et muuta käesolevat tüüpjuhtumi algtüüpjuhtumiks tühistades eelmise tüüpjuhtumi algtüüpjuhtumi staatus. Järgmine Ekraanipilt 52 illustreerib dialoogi:



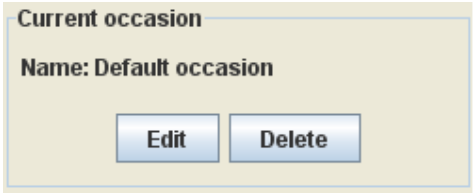
Ekraanipilt 52: Vaikimisi tüüpjuhtumi määramine

Vajutades nupule “Yes”, märgib rakendus loodava tüüpjuhtumi algtüüpjuhtumiks, vastasel korral loodud tüüpjuhtum ei saa algtüüpjuhtumi staatust.

Peale seda hakkab rakendus tegelema otseselt visuaalsete objektide struktuuri salvestamisega andmebaasi. Järgmises jaotuses on kirjeldatud konverteerimisprotsess visuaalsetest objektidest andmebaasi objektideks, mis hiljem salvestatakse andmebaasi. Vea korral informeerib süsteem kasutajat ning eduka salvestamisprotseduuri lõpust informeerib teade (Ekraanipilt 53):

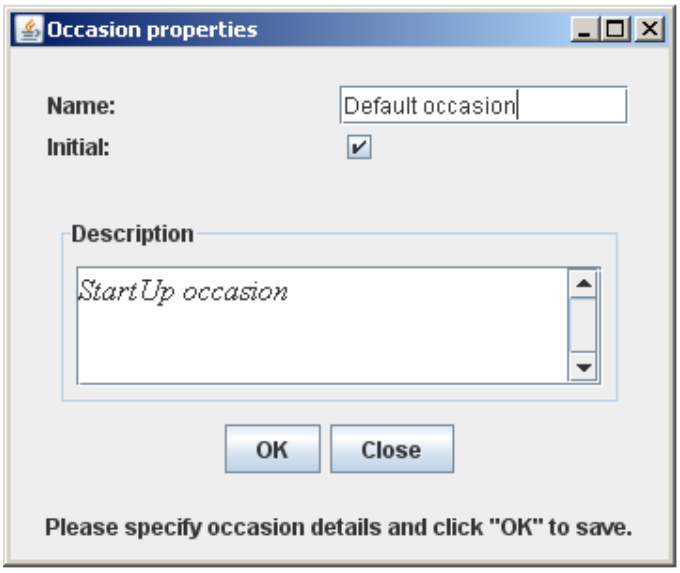
Ekraanipilt 53: Salvestamise õnnestus

Seda fakti et salvestamisprotseduur lõppes edukalt näitab ka see, et loodud tüüpjuhtumi nimi on kirjutatud tüüpjuhtumi infopaneelile ning funktsionaalsus tüüpjuhtumi muutmiseks ja kustutamiseks on aktiveeritud (vt. Ekraanipilt 54):



Ekraanipilt 54: Salvestatud tüüpjuhtumi info

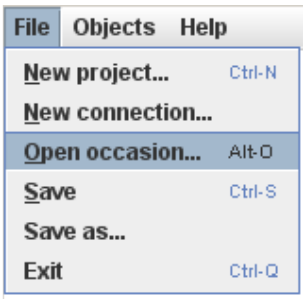
Tüüpjuhtumi parameetri **muutmiseks** peab kasutaja vajutama nupule “Edit”. Rakendus toob ekraanile parameetrite muutmise vormi, mis on illustreeritud järgmisel Ekraanipildil 55:



Ekraanipilt 55: Tüüpjuhtumi muutmisvorm

Nupp “Delete” käivitab käesoleva tüüpjuhtumi **kustutamisprotseduuri**. Rakendus kustutab kõik tüüpjuhtumi andmed ning antud tüüpjuhtumisse kuuluvate objektide andmed (küsimused, vastused, tegevused). Kuna lõppolek on rohkem üldine objekt kui küsimus ja teda saab kasutada paljudest tüüpjuhtumites korraga, siis kustutamise protseduur ei eemalda lõppolekuid. Samuti ei eemalda protseduur lõuendilt mitte ühtegi visuaalset objekti jättes kogu visuaalse mudeli muutmata, selleks et kasutaja töö ei kaoks ära ning eksisteeriks võimalus salvestada antud struktuuri.

Salvestatud tüüpjuhtumeid on võimalus **üles laadida** edaspidiseks redigeerimiseks. Selleks kasutaja valib peamenüüs “File->Open occasion…” (vt. Ekraanipilt 56):



Ekraanipilt 56: Peamenüü punkt "Open occasion..."

Rakendus reageerib sellele niimoodi, et toob ekraanile tüüpjuhtumite vormi, mis on sarnane ekraanipildiga 21 peatükis 5.2.4 „Teisele tüüpjuhtumile viite objekti käitumisloogika" (vt. Ekraanipilt 57):

Ekraanipilt 57: Tüüpjuhtumi valimise vorm

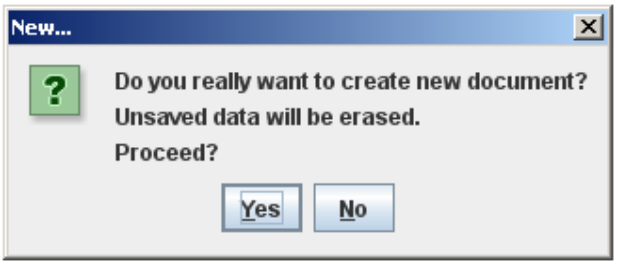
Et tüüpjuhtumit laadida on kasutajal võimalus vormis olevat tabelit sorteerida veergude abil vajutades veeru nimetusele. Näiteks, kasutaja saab sorteerida "Last modified" veeru järgi, et teada saada milline tüüpjuhtum on muudetud viimasena. Vajutades sobivale reale on kasutajal võimalus käima panna üleslaadimisprotsess vajutades nuppu "Load" või sulgeda vorm ilma tüüpjuhtumit üles laadimata, vajutades "OK". Kasutajal on ka võimalik **kustutada** tüüpjuhtumeid märkides neid viimasel veerul olevate linnukeste abil ning vajutades punast "Delete" nuppu (vt. Ekraanipilt 58):

Ekraanipilt 58: Tüüpjuhtumi määramine kustutamiseks

Peale edukat üleslaadimisprotsessi lõpetamist on lõuendil täiesti sama visuaalne mudel, mis oli salvestatud. Edasi on kasutajal võimalus mudelit redigeerida ja salvestada kas vana tüüpjuhtumi asemele valides peamenüüs valiku „Save...” või uueks tüüpjuhtumiks valides peamenüüs valiku „Save As...”.

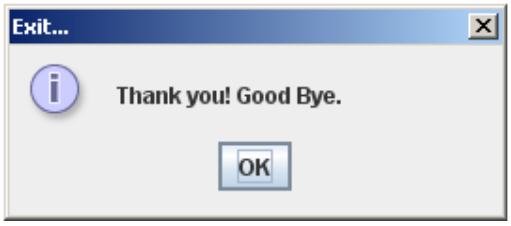
6.5.10 Küsimustiku projekti haldamisloogika

Rakendus omab ka funktsionaalsust, mis on mõeldud rakenduse haldamiseks. Need on uue projekti loomine ning rakenduse sulgemine. Peamenüü punkt „New project...” loob uue tüüpjuhtumi projekti tühjendades lõuendi ja väärtustades kõik rakenduse parameetrid vaikimisi. Alguses programm kontrollib kas lõuendil olev visuaalne mudel on salvestatud andmebaasi, kui ei, siis rakendus küsib kasutaja käest kinnitust andmete kustutamiseks (vt. Ekraanipilt 59):



Ekraanipilt 59: Uue tüüpjuhtumi projekti loomine

Kui kasutaja on nõus sellega, et andmed lähevad kaduma, siis vajutatakse „Yes”. Vastasel korral, vajutades „No” nuppu, katkestab kasutaja andmete kustutamise lõuendilt. Rakenduse sulgemisprotseduuri saab käivitada valides peamenüüst „File - > Exit”. Peale seda kontrollib rakendus visuaalset mudelit sama loogika abil, mis on kirjeldatud uue projekti/lõuendi tühistamisel. Kui mudel on salvestatud või kui kasutaja on nõus sellega, et sulgemise käigus kaotab ta loodud mudeli, siis programmi töö lõpetatakse ning näidetakse kasutajale järgmist akent (vt. Ekraanipilt 60):



Ekraanipilt 60: Rakenduse kinnipanemine

6.6 Rakenduse tehniline lahendus

Rakenduse disaini käigus leitakse lahendus nii andme- kui ka funktsionaalnõuetes määratletud tingimustele. Peatükk kirjeldab rakenduse tehnoloogiad, arhitektuuri, graafilist disaini, andmebaasi disaini ja tekkinud probleemide lahendusi.

6.6.1 Rakenduse arhitektuur

Kuna antud rakendus on andmebaasi andmete redaktor, siis oli plaanitud kasutada Java programmeerimiskeelt rakenduse ehitamiseks. Kuna rakendus pole tavaline tabeli tüüpi redaktor ning kasutab palju käsitsi tehtud graafilisi elemente andmete struktureerimiseks, siis on kasutusele võetud Batik SVG Toolkit 1.7 [8], mis on Java baasil tehtud raamistik rakenduste ja *applettide* jaoks. Seal on kasutusele võetud joonised SVG (*Scalable Vector Graphics*) formaadis, mida saab erinevalt vormistada. Neid saab näidata, genereerida ja manipuleerida. Graafilise disaini loomiseks oli kasutatud Java SE 1.6 versiooni [16] ning andmebaasiks oli valitud MySQL Server 5.1 versioon [17]. Selleks, et rakendus saaks suhelda andmebaasi mootoriga võeti kasutusele MySQL JDBC Connector/J 5.1 klass [9], mis pakub arendajale mugavaid vahendeid andmebaasi transaktsioonide loomiseks ja haldamiseks ning andmebaasi objektidega manipuleerimiseks. Kuna rakendused, mis on konstrueeritud Java baasil ei oma mingeid käivitamispiire operatsioonisüsteemide suhtes ning andmebaas on ka sobiv paljude operatsioonisüsteemide jaoks siis antud raamistike ja tehnoloogiate valik on sobiv.

6.6.2 Klasside grupid

Rakenduse programmeerimisel loodud klasse saab jagada neljaks grupiks:

- **visuaalseid objekte kirjeldavad ja haldamisklassid** – klassid, mis kasutavad otseselt Batik SVG raamistiku baasklasse. Nende abil on konstrueeritud klassid, mis kirjeldavad rakenduses kasutatavaid visuaalseid objekte, nagu küsimus, lõppolek, viide teisele tüüpjuhtumile, ühendus, vastus ja tegevus.
- **ärikihi objekte kirjeldavad ja haldamisklassid** – antud klasside kogumi abil konstrueerib rakendus äristruktuuri, mis peegeldab visuaalset struktuuri lõuendil. Siia kuuluvad ka kõik klassid, kus on kirjeldatud objektide käitumisloogika ning üldised rakenduse funktsioonid.
- **andmebaasi funktsionaalsust sisaldavad klassid** – antud gruppi kuuluvad klassid, mis suhtlevad andmebaasiga, sisaldavad andmebaasi päringuid ning funktsionaalsust, mis transformeerib ärimudeli, mis on tehtud ärikihi klasside abil, andmebaasi päringuteks. Lisaks kuuluvad siia klassid, mis tegelevad andmebaasi ühenduse loomisega ja haldamisega.
- **abiklassid**, mis sisaldavad üldisemaid funktsioone, mida kasutavad klassid teistest klassigruppidest.

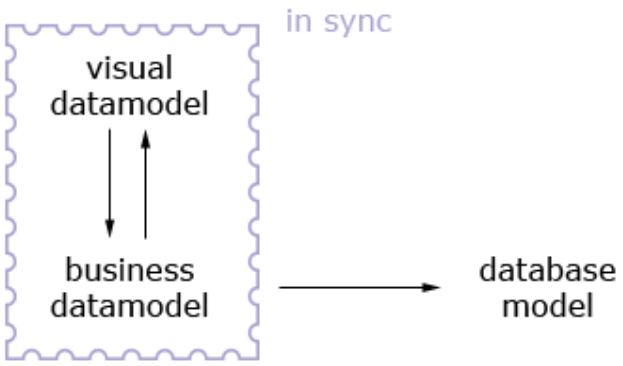
Klassigruppide kirjeldus mainib kõigi nelja klassigrupi omavahelist suhet. Iga grupp omakorda tegeleb konkreetsete ülesannetega kolmes rakenduse kihis: kasutajaliides ja visualiseerimine, äriloogika ja objektide haldamine ning andmebaasi liides.

Rakendus manipuleerib mälus kolme sorti andmestruktuuridega, kus kõik struktuurid on omavahel identsed aga omavad erinevat formaati, kuna on välja mõeldud erinevateks ülesanneteks:

- **visuaalne andmestruktuur** – visuaalsete objektide struktuur, mida kasutatakse visuaalsel joonistamisel lõuendile. Struktuuri luuakse XML DOM formaadis ning sellega tegelevad sissehitatud Batik SVG klassid, mida juhtivad autori poolt loodud klassid
- **äri andmestruktuur** – antud andmestruktuuri ehitatakse koos visuaalsete andmestruktuuriga ning seal hoitakse neid kahte struktuuri kogu aeg kooskõlas. Struktuur on loodud *bean* tüüpi klasside abil, kus iga *bean*-klass kirjeldab konkreetset visuaalset objekti ning omab funktsioone antud objekti haldamiseks.

- **andmebaasi struktuur** – antud struktuur ehitatakse sel hetkel kui kasutaja nõuab oma visuaalse andmestruktuuri andmebaasi salvestamist. Aluseks võetakse äristruktuur, mis on täiuslik koopia visuaalsest andmestruktuurist ning selle alusel tehakse andmebaaside päringutest koosnev struktuur. Antud struktuuri kasutatakse andmebaasi uuendamiseks.

Andmemudelite sünkroniseerimine on skemaatiliselt toodud joonisel 62.



Joonis 52: Andmemudelite sünkroniseerimine

6.6.3 Klassid

Eelmises peatükis jagasime klassid kolmeks grupiks täidetavate ülesannete järgi. Selles peatükis on lühidalt kirjeldatud igasse klassi gruppi autori poolt loodud klassid, nende otstarbed ja põhifunktsionaalsus.

Visuaalsete objektide klassi kuuluvad niinimetatud visuaalsete objektide tehased (*Factory*). Need on klassid, mille kaudu saab luua uue objekti isendi, hallata seda ning igal ajal käesolev objekti kätte saada. Kuna tehasklassid peavad meeles iga tema abil tehtud objekti (vt. Tabel 5).

Klassi tüüp	Klassi nimetus	Lühikirjeldus
interface	VoogObjectFactory.java	Antud liides defineerib meetodid, mida iga teda realiseeriv klass peab teostama.
public class	QuestionFactory.java	Tehasklass, mis realiseerib VoogObjectFactory liidest ning sisaldab meetodeid visuaalse

		küsimuse objekti loomiseks, kustutamiseks, objekti asukoha muutmiseks, kahe küsimuse objekti ühendamiseks ja objekti otsimiseks identifikaatori järgi.
public class	LineFactory.java	Tehasklass, mis realiseerib VoogObjectFactory liidest ning sisaldab meetodeid visuaalse ühenduse objekti loomiseks, kustutamiseks, objekti otsimiseks identifikaatori järgi ja lühima tee arvutamise funktsiooni kahe objekti vahel
public class	EndStateFactory.java	Tehasklass, mis realiseerib VoogObjectFactory liidest ning sisaldab meetodeid visuaalse lõppoleku objekti loomiseks, kustutamiseks, objekti asukoha muutmiseks ja objekti otsimiseks identifikaatori järgi.

Tabel 5: Visuaalsete objektide tehasklassid

Sellega on näha, et antud klassigrupi klassid tegelevad ainult visuaalsete objektide haldamisega. Järgmisena vaatleme kõige suuremat klassigruppi – **äriklasside gruppi**. Need on klassid, mis realiseerivad kogu rakenduse funktsionaalsuse (vt. Tabel 6).

Klassi tüüp	Klassi nimetus	Lühikirjeldus
public class/form	About.java	Klass, mis realiseerib akent, mida rakendus näitab kui kasutaja vajutab peamenüüs “Help->About”
interface	VoogObject.java	Liides määrab ära kõik meetodid, mida peavad kasutama klassid, mis teda realiseeruvad. Antud liidest realiseerivad klassid, mis tegelevad visuaalsete objektide visualiseerimisega (Question.java, EndState.java)
public class/bea	Question.java	*Bean*-klass, mis hoiab infot küsimuse objekti kohta. Iga selle klassi isend sisaldab infot nii

n		visualiseerimise jaoks kui ka teisi tähtsamaid objekti parameetreid nagu nimetus, kirjeldus jne. Lisaks on klassis kirjeldatud abimeetodid objekti kõigi ühenduste kätte saamiseks, uue ühenduse lisamiseks ning ühenduse objektist eemaldamiseks.
public class/bean	EndState.java	*Bean*-klass, mis hoiab infot lõppoleku objekti kohta. Iga selle klassi isend sisaldab infot nii visualiseerimise jaoks, kui ka teisi tähtsamaid objekti parameetrid nagu nimetus, kirjeldus jne.
public class/bean	Line.java	*Bean*-klass, mis hoiab infot ühenduse ja tegevuse objektide kohta. Iga selle klassi isend sisaldab infot nii visualiseerimise jaoks kui ka teisi tähtsamad objekti parameetrid nagu nimetus, kirjeldus, algusobjekt, lõppobjekt, tegevuse nimetus ja tegevuse kirjeldus. Lisaks mõned ühenduse abimeetodid
bean	Answer.java	*Bean*-klass, mis hoiab infot vastuse objekti kohta. Iga selle klassi isend sisaldab infot nii visualiseerimise jaoks kui ka teisi tähtsamaid objekti parameetreid nagu nimetus, kirjeldus jne.
public class/form	Activity_AnswerForm.java	Klass kirjeldab ühenduse parameetrite vormi, mis sisaldab tegevuse ja vastuse parameetrid. Rakendus toob ühenduse loomisel antud vormi ekraanile
public class/form	EndStateForm.java	Klass kirjeldab lõppoleku parameetrite vormi, kus on toodud väljad lõppoleku nime ja kirjelduse jaoks. Rakendus toob lõppoleku objekti loomisel antud vormi ekraanile
public	newOccasionForm.java	Klass kirjeldab vormi uue tüüpjuhtumi

class/form		loomiseks. Peatükis "Tüüpjuhtumi käitumisloogika" on kirjeldatud tüüpjuhtumi loomise protsess, kus on toodud selle vormi joonis. Vormi parameetriteks on loodava tüüpjuhtumi nimi, loodava tüüpjuhtumi kirjeldus ja parameeter, kas antud tüüpjuhtum on vaikimisi tüüpjuhtum, millest alustatakse küsimustikku või mitte.
public class/form	OccasionForm.java	Klass mis kirjeldab tabeli vormi tüüpjuhtumi valimiseks. Kasutatakse teisele tüüpjuhtumile viite loomiseks lõuendile. Vormi loogika ja käitumine on kirjeldatud peatükis "Teisele tüüpjuhtumile viite objekti käitumisloogika".
public class/form	QuestionForm.java	Klass kirjeldab küsimuse objekti parameetrite vormi. Kasutatakse antud objekti loomisel lõuendile. Vormi parameetrite nimekiri: küsimuse nimi, küsimuse kirjeldus, parameeter, mis kirjeldab, kas küsimus on algküsimus tüüpjuhtumis, parameeter, mis määrab objekti aktiivsust. Vormi käitumisloogika on kirjeldatud peatükis "Küsimuse objekti käitumisloogika"
public class/form	ConnectionCreator.java	Klass, mis kirjeldab andmebaasi ühenduse loomise vormi. Täpne vormi käitumisloogika koos joonistega on toodud peatükis „Esimesed sammud ehk ühenduse loomine"
interface	ObjectProperties.java	Liides, mis määrab ära meetodid, mida peavad sisaldama klassid selle liidese realiseerimisel.
public class/bean	EndStateObjectProperties.java	*Bean* klass teostab ObjectProperties liidest. Sisaldab lõppoleku objekti parameetreid, mis on kasutusele võetud objekti lõuendile joonistamisel ja matemaatiliste arvutuste tegemisel.

public class/bean	QuestionObjectProperties.java	*Bean* klass teostab ObjectProperties liidest. Sisaldab küsimuse ja tüüpjuhtumi objektide parameetreid, mis on kasutusele võetud objekti lõuendile joonistamisel ja matemaatiliste arvutuste tegemisel.
public class/bean	ObjectDataBean.java	*Bean* klass, mis sisaldab kõiki visuaalsete objektide (va ühendus) parameetreid, mis võivad tulla antud objektide loomise vormidelt.
public class/bean	MainOccasionBean.java	*Bean* klass, mis sisaldab käesoleva/muudetava tüüpjuhtumi parameetreid, mis tulevad tüüpjuhtumi loomise vormilt.
public class/panel	CurrentOccasionPanel.java	Klass, mis kirjeldab käesoleva tüüpjuhtumi infopaneeli. Antud kasutajaliidese osa käitumisloogika ja põhimõtted on kirjeldatud peatükis "Rakenduse põhikomponentide käitumisloogika" ja "Tüüpjuhtumi käitumisloogika"
public class/panel	InstrumentPanel.java	Klass, mis kirjeldab kasutajaliidese instrumentaalset paneeli, kus on toodud visuaalsete objektide nuppude valik. Jaotises 5.8 "Rakenduse abikomponentide käitumisloogika" on kirjeldatud antud paneeli omapärasid ja funktsionaalsust.
public class/panel	StatusInfoPanel.java	Klass, mis kirjeldab objektide informatsiooni paneeli, kus on toodud valitud objekti tähtsad parameetrid mugavas vormis. Peatükis "Rakenduse abikomponentide käitumisloogika" on kirjeldatud antud paneeli omapärasid ja funktsionaalsust.

public class/panel	StatusPanel.java	Üldine rakenduse paneel, mis asub rakenduse akna paremal pool. Sisaldab endas *käesoleva tüüpjuhtumi infopaneeli, instrumentaalset paneeli ja objektide informatsiooni paneeli*.
public class/panel	JStatusBar.java	Klass, mis kirjeldab **üldinfopaneeli**. Jaotis "Rakenduse abikomponentide käitumisloogika" sisaldab informatsiooni antud paneeli kohta.
public class/listener	LineMenuActionListener.java	Klass, mis kuulab sündmusi, mis on seotud ühenduste muutmisega ja eemaldamisega
public class/listener	MenuActionListener.java	Klass, mis kuulab sündmusi, mis on seotud kõikide teiste visuaalsete objektidega ja rakenduse menüüdega
public class/listener	SvgObjListener.java	Klass, mis kuulab lõuendi visuaalsete objektide ja hiire vahelisi tegevusi ning tegeleb nendega. Näiteks, hiir on objekti peal või mitte.
public class	NonEditableCellEditor.java	Antud klassi kasutatakse küsimuse loomise vormil, sarnase vastuse määramisel, "Answers" taab. Klass on ühendatud vastuste tabeliga ning annab võimalus määrata tabeli lahter mittemuudetavaks.
public class	OccasionLoader.java	Klass, mis kirjeldab tüüpjuhtumi üleslaadimise vormi. Rakendus toob ekraanile vormi, kui kasutaja valib põhimenüüs "File->Open occasion…". Vormi käitumisloogika on kirjeldatud peatükis "Tüüpjuhtumi käitumisloogika".

Main public class	SVGDrawingTest.java	Rakenduse käivitamisklass, mis sisaldab endas kõikide põhiobjektide isendeid, rakenduse käitumisloogikat, lõuendi käitumisloogikat ja menüü käitumisloogikat. Rakenduse tuuma klass.

Tabel 6: Äriklasside gruppi kuuluvad klassid

Nagu on näha ülaltoodud tabelist, võib äriklasside gruppi jagada mitmeks alamgrupiks. Näiteks *bean*-klasside grupp või klasside grupp, mis tegeleb vormide loomisega ja parameetrite korjamisega ning kasutajaliidese paneeliklasside grupp. Järgmisena on vaatluse all **andmebaasi klasside grupp** (vt. Tabel 7).

Klassi tüüp	Klassi nimetus	Lühikirjeldus
public class	DBConnector.java	Põhiklass, mis tegeleb andmebaasi ühenduse tagamisega ning päringute käivitamisega. Süsteemi andmebaasi niiöelda *driver*-klass. Klassi ülesannete hulgas on ka ühenduse loomine ning hävitamine.
public class	DBLoader.java	Klassi põhiülesanne on tüüpjuhtumi üleslaadimine andmebaasist. Nimelt, tüüpjuhtumi andmebaasi struktuuri põhjal äri andmestruktuuri ja visuaalse struktuuri loomine. Protsess on vastupidine tüüpjuhtumi andmebaasi salvestamisega.
enum	SQLStatements.java	Rakenduse SQL päringute kogum.
public class	XMLConnectionFilter.java	Klass filtreerib .xml laiendiga faile, ning kasutatakse andmebaasi ühenduse loomisel, kui info võetakse failist. Faili struktuur, ning tööpõhimõtted on kirjeldatud peatükis „Esimesed sammud ehk ühenduse loomine”.

Tabel 7: Andmebaasi klassidegrupile kuuluvad klassid

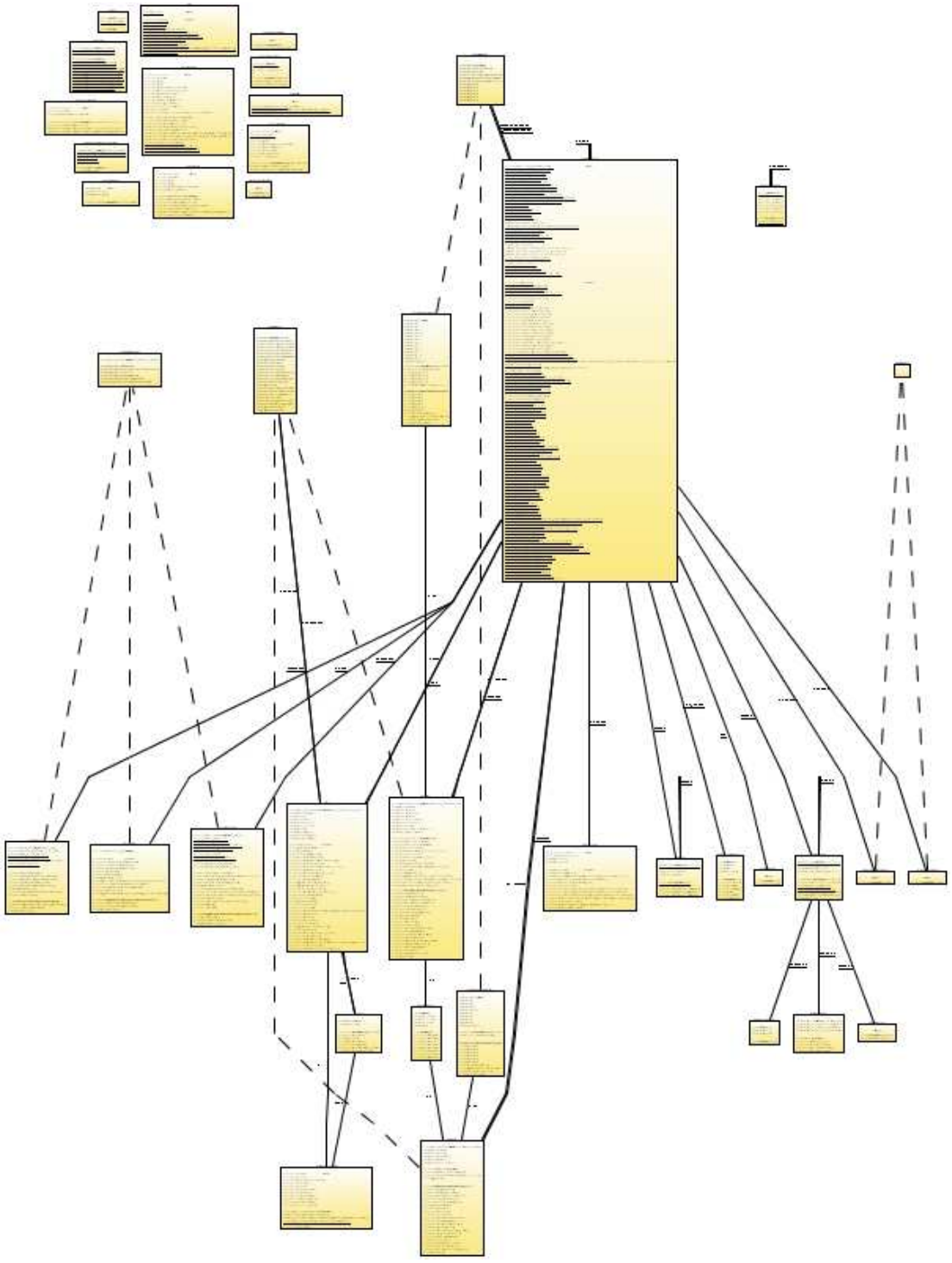
Järgmine klasside grupp on **abiklassid** (vt. Tabel 8). Sinna hulka kuuluvad klassid, mis ei oma konkreetset ülesannet, vaid sisaldavad endas funktsionaalsust, mida kasutavad rakenduse teised klassid.

Klassi tüüp	Klassi nimetus	Lühikirjeldus
public class	SpringUtilities.java	Klass, mis võimaldab kasutada SpringLayout funktsionaalsust Java rakenduses. Kastutatakse rakenduses korduvalt.
public class	AngledLinesWindowsCornerIcon.java	Klass, mis teeb ilusaks rakenduse paneelide äärt, lisades sinna pehmet varju.
public class	Utils.java	Abiklass, mis sisaldab universaalsed ühisfunktsioone. Näiteks, funktsioonid, mis on seotud sõnede töötlemisega ja nii edasi.

Tabel 8: Abiklassid

6.6.4 Klassidiagramm

Rakenduse funktsionaalsus on jagatud klassigruppide vahel nii, et igas grupis iga klass tegeleb oma ülesannetega. Seoses sellega rakenduse klasside arv on suur. Mõned klassid pärivad teisi, mõned kasutavad teiste klasside meetodeid, mõned on niiöelda eraldiseisvad klassid. Nende klasside eksemplarid tehakse iga kord uuesti ja hävitakse, kui antud eksemplar pole kasutusel. Kõik klassid, mis tegelevad vormi kokkupanemisega (eelmises peatükis olevates tabelites klassid, mille klassi tüüp on *public class/vorm*) kuuluvad sellesse gruppi. Järgmise joonise (Joonis 53) otstarve on näidata klasside omavahelisi seoseid (suurema resolutsiooniga joonise saab leida dokumendile lisatud CD plaadil):



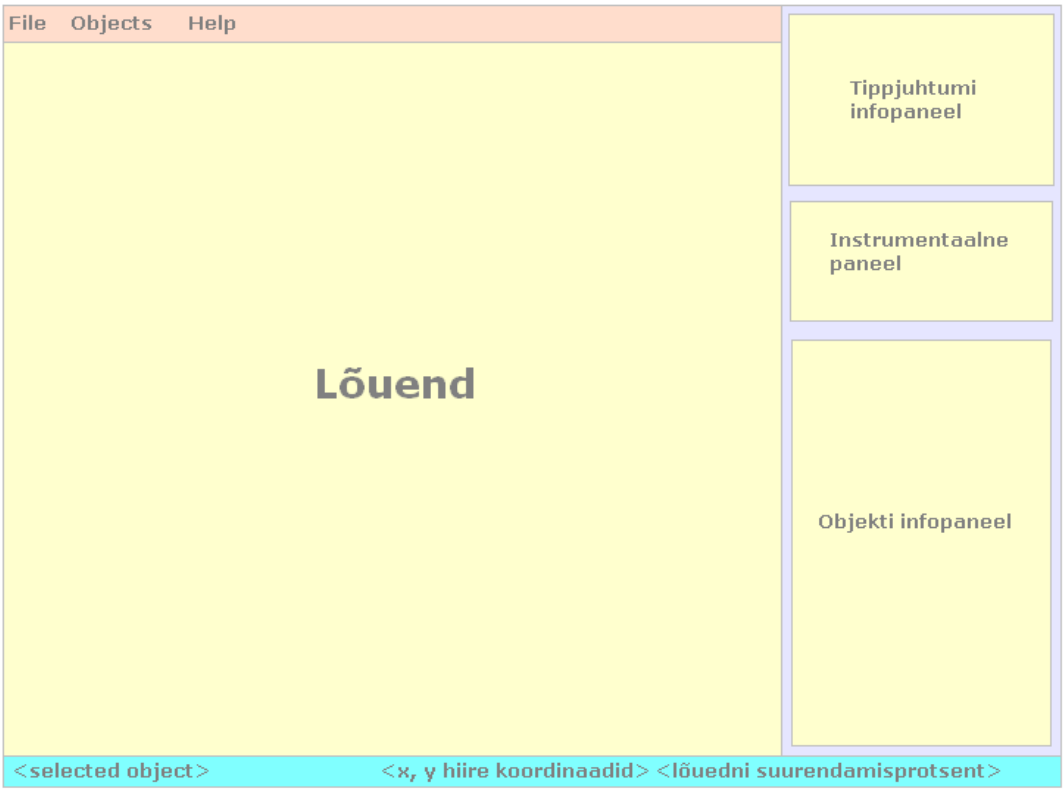
Joonis 53: Rakenduse klassidiagramm

6.6.5 Rakenduse graafilised elemendid

Rakenduse graafilisteks elementideks loeme programmi visuaalseid osi, mis koostöös annavad meile täiusliku rakenduse visuaalse disaini. Nende komponentide hulka kuuluvad kõik visuaalsed paneelid, lõuend ja menüü. Visuaalsete objektide hierarhia on toodud järgmisena:

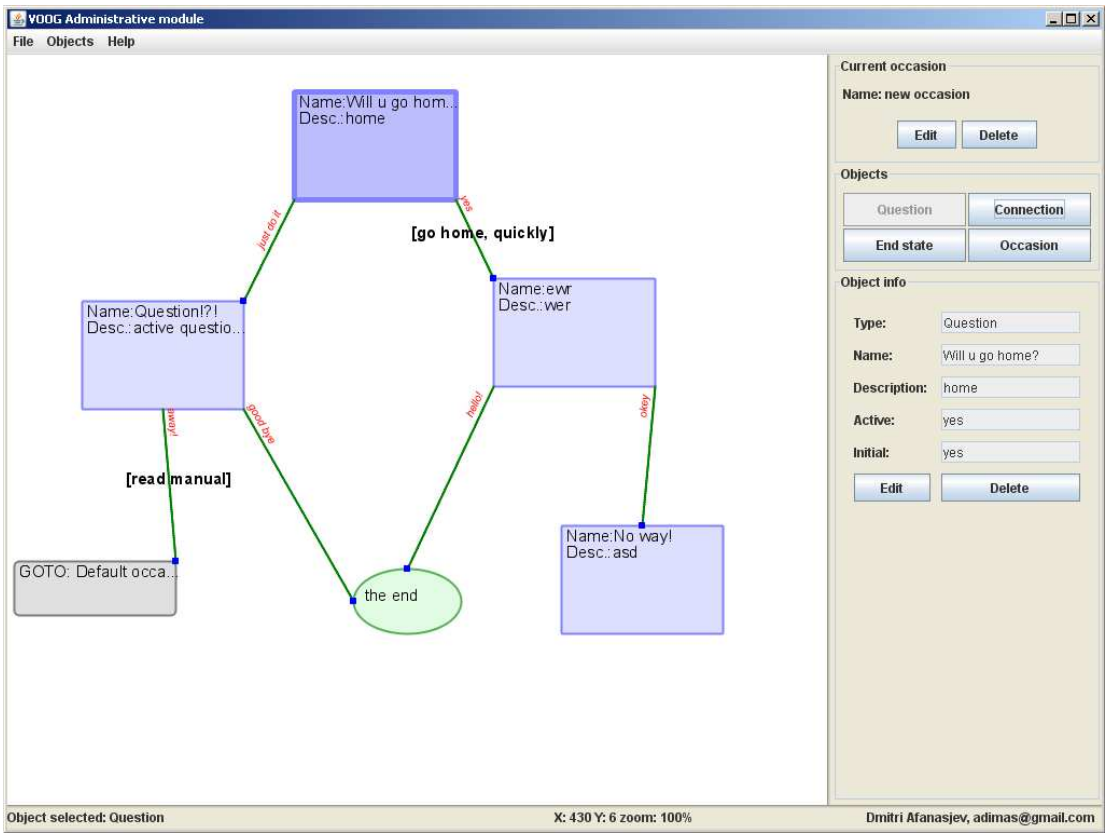
- Rakenduse visuaalne ala
    - Peamenüü
        - „File”
            - „New project...”
            - „New connection...”
            - „Open occasion...”
            - „Save”
            - „Save As...”
            - „Exit”
        - „Objects”
            - Question
            - Connection
            - End state
            - Link to occasion
        - „Help”
            - „About”
    - Lõuend
    - Peapaneel
        - Tüüpjuhtumi infopaneel
        - Instrumentaalne paneel
        - Objekti informatsiooni paneel
    - Üldinfopaneel

Kuna kõik visuaalsed komponendid ja nende töö on kirjeldatud eelmises peatükis, siis kasutame ülaltoodud hierarhiat rakenduse kasutajaliidese kavandamiseks. Järgmine joonis 54 näitab esialgset rakenduse kasutajaliidese disaini:



Joonis 54: Rakenduse paneelide paigaldus

Antud komponentide asetus on väga sarnane tuntud olemasolevate programmidega: töölaud on eespool ja vajalikud materjalid on “parema käe all”, rakenduse informatsioon on teostatud ribana, mis asub rakenduse akna alumises osas ning peamenüü on traditsioonilisel kohal: rakenduse akna ülemises osas. Rakenduse graafiline liides on ingliskeelne. Kuna rakenduse programmeerimiskeeleks on Java, mille jaoks pole oluline millise platvormi peal töötada (Windows, *NIX ja teised süsteemid, kus on realiseeritud Java Virtuaalne Masin), siis oli plaanitud teha rakenduse disain Windowsi rakenduse moodi. Järgmine Ekraanipilt 61 näitab, kuidas näeb välja reaalne rakenduse aken:



Ekraanipilt 61: Töötava rakenduse ekraanipilt

6.6.6 Andmebaas, tekkinud probleemid ja nende lahendused

Andmebaas, kui andmehoidla peab toetama süsteemi osi vajalike andmetega, ning pakkuda nendele liidest ja teenuseid informatsiooni salvestamiseks, muutmiseks ja otsimiseks. Nende nõuete tagamiseks peavad andmed olema salvestatud efektiivselt, see tähendab, et andmed peavad olema jagatud tabeliteks. Lisaks peavad andmebaasi tabelid ühiselt konstrueerima andmebaasi struktuuri niimoodi, et vältida andmete liiasust. Sellepärast, peab andmebaas olema vähemalt kolmandal normaalkujul.

Kavandatav andmebaasi disain tehti enne seda kui süsteemi teiste osade autorid valmistasid oma rakendusi ning see oli kirjeldatud peatükis “Andmenõuded”. Kuid modelleerimisliidese loomisel leidis autor mõned probleemid ja tegi muudatusi niimoodi, et kogu süsteem jääks töötama nagu alguses. Osa nendest probleemidest ning probleemidega seotud muudatustest

puudutavad ainult modelleerimisliidest. Üks reaalne probleem, mis nõudis osade tabelite muutmist tekkis siis, kui rakenduse arendusetapp jõudis selleni, et realiseerida tüüpjuhtumi struktuuri salvestamis- ja allalaadimisemehhanismi. Nimelt, kuna rakenduse lõuend on tasand, mis omab koordinaadistikku, siis rakendus kasutab just seda koordinaadistikku visuaalsete objektide manipuleerimiseks. Igal visuaalsel objektil peavad olemas olema väljad, mis määravad *x* ja *y* koordinaadi lõuendil. Kuna tüüpjuhtumi salvestamisel lõuendi algseis ja tüüpjuhtumi üleslaadimisel lõuendi lõppseis peavad olema võrdsed, siis andmebaasi tabelid peavad pakkuma rakendusele võimalust hoida iga visuaalse objekti jaoks tema lõuendi koordinaate. Selleks on tabelitesse “Question” ja “End_state” lisati veergude paarid (*x* ja y). Sama protseduuriga on seotud veel üks probleem, mida polnud kajastatud esialgses andmebaasi disainis: rakendus peab eristama aktiivseid ja mitteaktiivseid visuaalseid objekte. Samasuguse eristuse tagamiseks lisati tabelitesse “Question” ja “End_State” lisaparameeter nimega “is_public”, mis määrab ära kas antud objekt peab osalema küsimustikus või on tema veel kinnitamata ja vajab muutmist.

Rakenduse disainimisel oli paika pandud poliitika, et loodud klasse tuleb kasutada maksimaalselt efektiivselt. Sel põhjusel, kirjeldab üks *java* klass kahte visuaalset objekti, kuna nende objektide käitumine ja visuaalne kuju on sarnane. Täpsemalt öeldes, klass nimega Question.java kirjeldab ja tegeleb *Question* ja *Link to occasion* visuaalsete objektidega. Nende objektide eristamiseks omab klass järgmist parameetrit:

```
private boolean isOccasion;
```

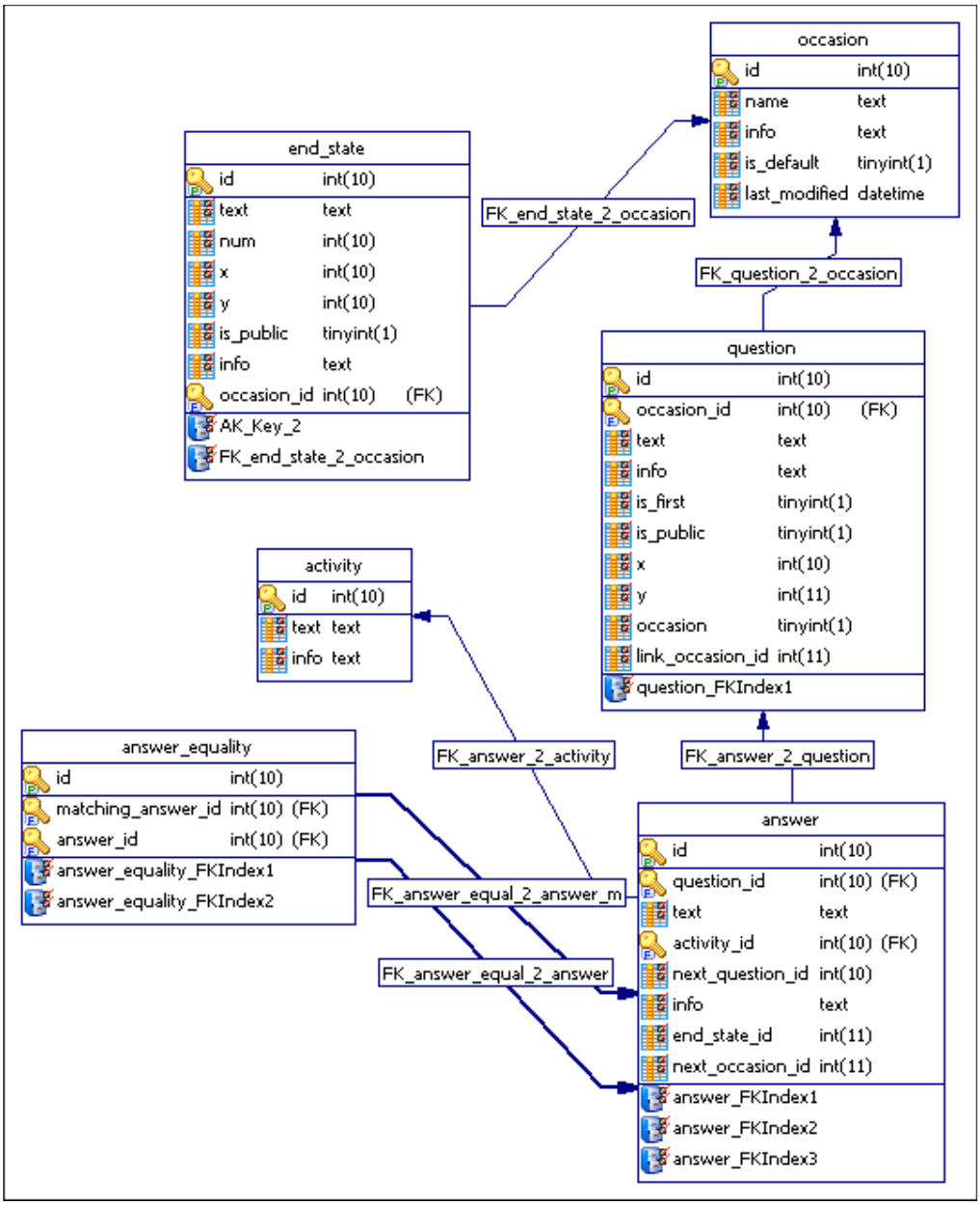
Kui antud parameeter on tõene, siis on vaatluse all viide teisele tüüpjuhtumile, vastasel juhul on vaatluse all küsimus. Neid kahte objekti hoitakse erinevates *HashMap* objektides küsimuste objektide tehases (klass QuestionFactory.java). Kuna äriandmestruktuurilt andmebaasi sünkroniseerimisel tehakse identne andmebaasi struktuur, siis viimane ka peab toetama antud atribuuti. Selleks on tabelisse “Question” lisatud kaks parameetrit *occasion ja link_occasion_id*. Esimene neist määrab ära kas objekt on küsimus või viide teisele tüüpjuhtumile. Juhul kui objekt on viide teisele tüüpjuhtumile, siis teine parameeter on tüüpjuhtumi identifikaator, millele viidab antud objekt.

Järgmine probleem on selline, et kuna definitsioon järgi tüüpjuhtumil peab olema kindlasti üks algküsimus, millest tüüpjuhtumi küsimustikku alustatakse, siis ka andmebaas peab seda

peegeldama. Selleks lisati tabelisse “Question” lisaväli nimega “is_first”, mis määrab ära, kas objekt on algobjekt või mitte.

Tabelisse “Occasion” oli lisatud lisaveerg nimega “last_modified”. Antud veergu kasutatakse tüüpjuhtumi üleslaadimisprotseduuril ning viite teisele tüüpjuhtumile visuaalse objekti loomisel. Rakendus avab akna, kus on toodud tüüpjuhtumite nimekiri koos viimase muutmisajaga. Selle aja järgi on lihtne aru saada, milline tüüpjuhtum on vanem ja milline on uuem.
Nagu enne öeldud, lisati tabelile “End_state” veerud nimedega “x”, “y” ja “is_public”. Lisaks sellele lisati veerud “info” ja “occasion_id”. Esimene väli hoiab lõppoleku kirjeldust ja teine parameeter määrab ära, millisele tüüpjuhtumile kuulub antud objekt. Selle parameetri järgi on võimalik kiiresti filtreerida objekte selleks, et joonistada neid lõuendile üleslaadimise protsessi käigus.

Rakenduse arendamise käigus leiti vigu, mis olid tekkinud süsteemi disainimisel. Kõige suurem viga oli seotud andmete liiasusega. Nimelt oli esialgsel andmebaasil skeemil olemas kaks tabelit “Answer_equality” ja “Question_equality”. Esimene neist sisaldab endas sarnaste vastuste identifikaatorite paare, teine aga sarnaste küsimuste identifikaatorite paare. Kuna üks vastus võib olla seotud ainult ühe küsimusega ja ühes küsimuses võib olla palju vastuseid (1-N seos), siis vastuse identifikaatori abil on võimalus üheselt pärida küsimus, kuhu antud vastus kuulub. Sellega “Question_equality” tabel ei oma mingit tähtsust süsteemi jaoks. Järgmine joonis 55 illustreerib reaalset andmebaasi ER diagrammi peale uuendamist:



Joonis 55: Andmebaasi ER diagramm

Järgmised tabelid kirjeldavad ära iga tabeli veeru ülesande, milleks see veerg on tabelis ning süsteemis. Tabelite tehnilised parameetrid on lisatud käesoleva dokumendi lõppu (Lisa 1)

6.6.6.1 Tabel „End_state”

Antud tabel hoiab informatsiooni tüüpjuhtumite lõppolekute objektide kohta (vt. Tabel 9).

Tabeli veerg	Kirjeldus
id	objekti identifikaator
text	objekti nimetus
num	objekti järjekorra number, mida kasutatakse edukate lõppolekute eristamiseks. Mitteedukad lõppolekud on rakendusse sisse programmeeritud
xx	objekti abstsiss koordinaat lõuendi tasandil.
yy	objekti ordinaat koordinaat lõuendi tasandil.
is_public	määrab ära objekti kasutatavuse tüüpjuhtumis, kui „0” siis kliendile antud objekti ei näidata ja vastupidi „1” jaoks.
info	objekti kirjeldus
occasion_id	tüüpjuhtumi identifikaator, kuhu kuulub antud objekt

Tabel 9: Andmebaasi tabeli „End_state“ kirjeldus

6.6.6.2 Tabel „Occasion”

Tabel hoiab informatsiooni süsteemi tüüpjuhtumite kohta (vt. Tabel 10).

Tabeli veerg	Kirjeldus
id	objekti identifikaator
name	objekti nimetus
info	objekti kirjeldus
is_default	määrab ära, kas antud objektiga hakatakse tegelema esmajärjekorras. Niinimetatud vaikimisi tüüpjuhtum
last_modified	objekti viimane muutmisaeg

Tabel 10: Andmebaasi tabeli „Occasion“ kirjeldus

6.6.6.3 Tabel „Question”

Tabel hoiab informatsiooni süsteemi tüüpjuhtumite küsimuste kohta (vt. Tabel 11).

Tabeli veerg	Kirjeldus
id	objekti identifikaator
occasion_id	tüüpjuhtumi identifikaator, kuhu antud objekt kuulub
text	objekti nimetus, küsimuse tekst
info	objekti kirjeldus, küsimuse kirjeldus
is_first	määrab ära, kas objekt on algobjekt tüüpjuhtumis
is_public	määrab ära objekti kasutatavuse tüüpjuhtumis, kui „0” siis kliendile antud objekti ei näidata ja vastupidi „1” jaoks.
xx	objekti abstsiss koordinaat lõuendi tasandil.
yy	objekti ordinaat koordinaat lõuendi tasandil.
occasion	määrab ära kas objekt on küsimus („0”) või viide teisele tüüpjuhtumile („1”)
link_occasion_id	juhul kui *occasion* parameeter on „1”, siis antud parameeter määrab tüüpjuhtumi identifikaatori, kuhu objekt viitab

Tabel 11: Andmebaasi tabeli „Question“ kirjeldus

6.6.6.4 Tabel „Activity”

Tabel hoiab informatsiooni süsteemi tegevuste kohta (vt. Tabel 12).

Tabeli veerg	Kirjeldus
id	objekti identifikaator
text	objekti nimetus, tegevuse tekst
info	objekti kirjeldus, tegevuse kirjeldus

Tabel 12: Andmebaasi tabeli „Activity“ kirjeldus

6.6.6.5 Tabel „Answer_equality”

Tabel hoiab informatsiooni süsteemi sarnaste vastuste kohta (vt. Tabel 13).

Tabeli veerg	Kirjeldus
id	rea identifikaator
answer_id	vastuse identifikaator
matching_answer_id	sarnase vastuse identifikaator

Tabel 13: Andmebaasi tabeli „Answer_equality“ kirjeldus

6.6.6.6 Tabel „Answer”

Tabel 14 hoiab informatsiooni süsteemi küsimuste vastuste kohta.

Tabeli veerg	Kirjeldus
id	objekti identifikaator
question_id	küsimuse objekti identifikaator, kuhu kuulub antud objekt
text	objekti nimetus, vastuse tekst
activity_id	tegevuse identifikaator, mida täidetakse enne edasiminekut järgmisele objektile
next_question_id	järgmise küsimuse/tüüpjuhtumi identifikaator
info	objekti kirjeldus ja vastuse kirjeldus
end_state_id	järgmise lõppoleku identifikaator, parameeter on „0” kui *next_question_id* on väärtustatud ja vastupidi
next_occasion_id	juhul kui *next_question_id* on teisele tüüpjuhtumile viide identifikaator, siis antud parameeter näitab tüüpjuhtumi identifikaatorit, millele antud viide viitab

Tabel 14: Andmebaasi tabeli „Answer“ kirjeldus

7. Häirekeskuse kutsetöötlus

Loodud tarkvaraline töövoo haldamissüsteem on keskendunud küsimustikkude koostamisele. Antud kitsendustega süsteem annab vahendid modelleerimaks töövoo diagramme ja käivitada neid töövoomootori abil. Süsteem on mõeldud mitte ainult häirekeskuse kutsetöötluse ülesande lahendamiseks: toimetaja võimaldab koostada teatud kitsendustega erinevaid küsimustikke ning töövoomootor on piisavalt võimas, et neid käivitada süsteemi keskkonnas ja pakkuda võimalust läbida küsimustikke kasutades ühte kahest erinevast tarbimisliidest. Seega, loodud süsteem võib lahendada kogu teatud kitsendustega lühiajaliste ülesannete klassi, mille läbimise eesmärk on otsuseni jõudmine.

Süsteemi töö näidiseks on häirekeskuse kutsetöötlus, mille käigus telefoni teel dispetšer jõuab otsuseni, kas saata kiirabi kohale või mitte. Selle tõttu, et dispetšeri teadmised pole täielikud (inimene ei saa kõike teada) otsustamiseks on vähe aega vale otsuse tegemine ei ole rahaliselt optimaalne ei saa loota sellele, et dispetšeri poolt “käsitsi” tehtud otsus on õige ja kõige efektiivsem. Parem viis otsuseni jõuda on probleemi dekomponeerimine osadeks ehk probleemiga seotud küsimusteks ning luua järgnevused nende vahel. Antud küsimustiku käivitades, dispetšer küsib helistajalt küsimusi ja pakub eeldefineerituid vastusi (või interpreteerib vastust ühena võimalikest valikutest). Küsimuse vastusest sõltub, millist küsimust küsitakse järgmisena. Lõppude lõppuks jõuakse otsuseni piiratud ressursi (kiirabi) kasutamise kohta ja kui läbitud küsimustik oli kvaliteetselt koostatud ja modelleeritud valdkonna spetsialisti(de) poolt, siis on võimalik väita, et saadud otsus on õige ja kõige optimaalsem variant antud olukorra jaoks.

Näidisena loodud küsimustik ei pretendeeri olema häirekeskuse pärisküsimustik, vaid näitab võimaliku variandi. Selleks, et seda reaalselt kasutada, tuleb eelnevalt paluda kontrollida ja vajadusel seda muuta häirekeskuse spetsialisti poolt. Loodud küsimustikus on 22 küsimust ning ta on jagatud seitsmeks tüüpjuhtumiks. Tüüpjuhtum on alamküsimustik, mis sisaldab valdkonnaga või probleemiga seotud küsimusi. Iga tüüpjuhtum omab algküsimust, millest kogu alamküsimustiku läbimine algab. Tüüpjuhtumite hulgast on võimalik eraldada neid, mis sisaldavad tihtikasutatavaid küsimusi, valdkonnaspetsiifilisi küsimusi ning organisatoorseid küsimusi. Ka tüüpjuhtumite hulgast on olemas üks spetsiaalne tüüpjuhtum, millest kogu küsimustik algab (antud juhul see on tüüpjuhtum nimega „Põhiküsimustik“).

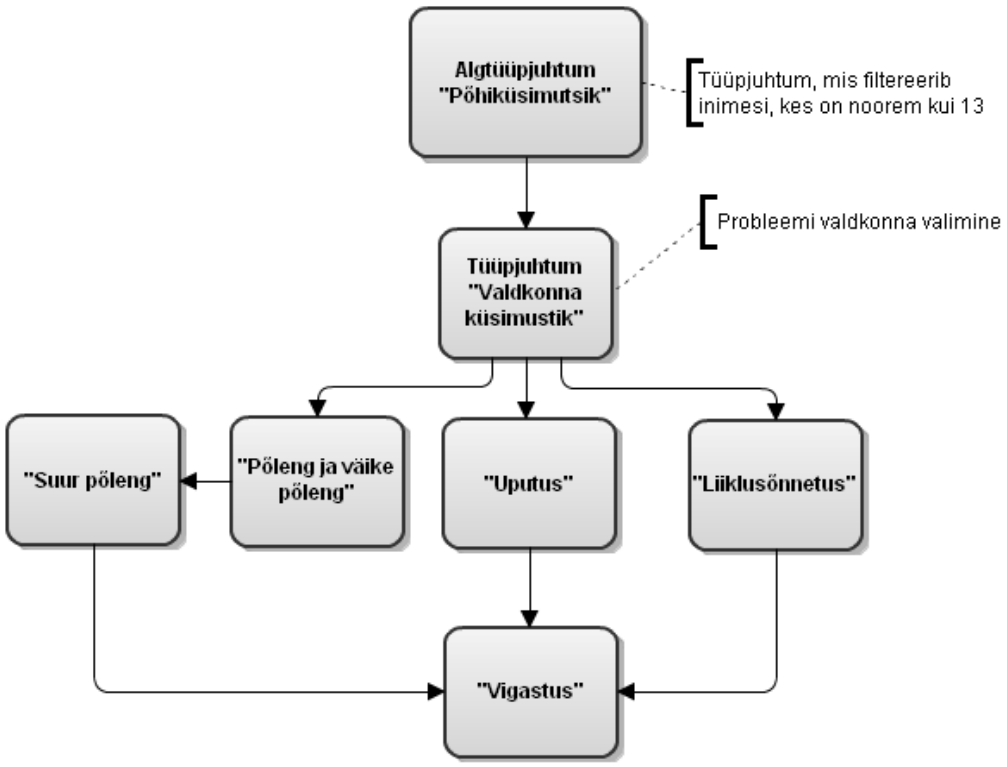
Tihtikasutatav tüüpjuhtum sisaldab küsimusi inimese vigastuse kohta: kui inimesel on vigastusi, siis teeb otsuse, et on vajalik kiirabi kohale saata. Selle tüüpjuhtumi nimi on „Vigastus“.

Valdkonnaspetsiifilisi tüüpjuhtumeid on mitu. Nemad sisaldavad valdkonnaspetsiifilisi küsimusi. Häirekeskuse kõnetöötluse näites on võimalik eraldada kolm käsitletud valdkonda: „Põleng“, „Uputus“ ja „Liiklusõnnetus“. Valdkonna alamküsimustik, mis on seotud põlenguga on omakorda jagatus kaheks tüüpjuhtumiteks: suur ja väike põleng.

Organisatoorsed küsimused on paigutatud kahte tüüpjuhtumisse: esimene tegeleb kontrolliga, et helistaja oleks vanem kui 13 aastat (idee on selline, et noorem inimene alati ei suuda situatsioonist õigesti aru saada ja seetõttu vastab küsimustele valesti, mis võib viia vale otsuse tegemiseni). Juhul kui helistaja on noorem, siis teda palutakse kutsuda lähim täiskasvanu selleks, et see seletaks probleemi ja vastaks dispetšeri poolt esitatud küsimustele. Teine tüüpjuhtum tegeleb probleemi valdkonna valimisega, mille käigus valitakse järgmist tüüpjuhtumi spetsiifiliste tüüpjuhtumite seast („Põleng“, „Uputus“ ja „Liiklusõnnetus“).

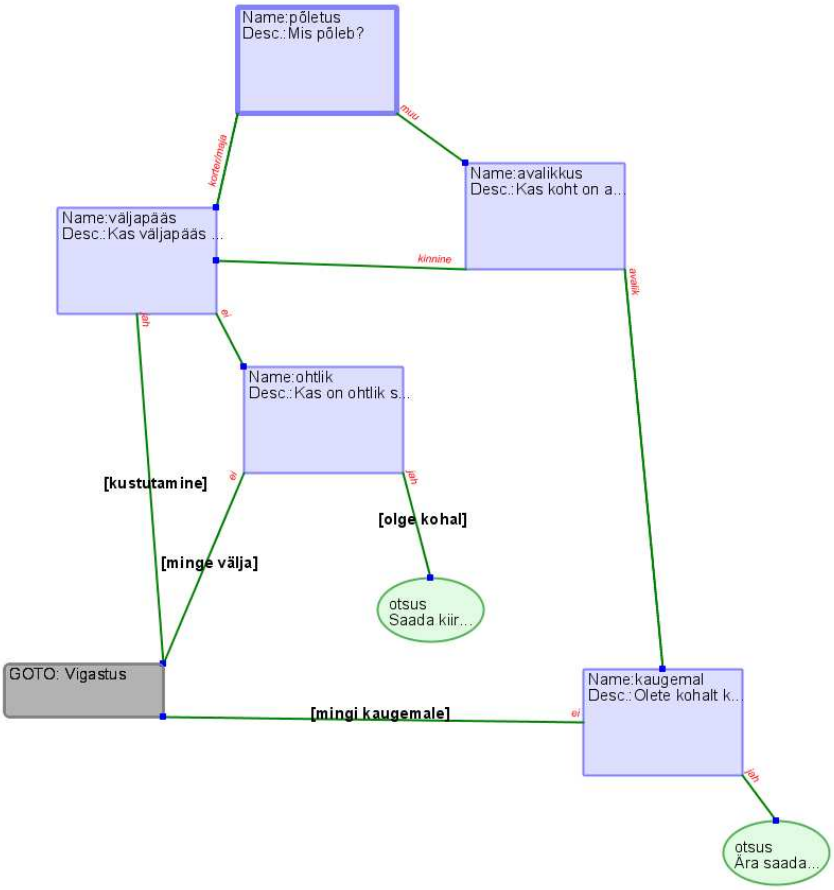
Tüüpjuhtumitest, nagu blokkidest, on ehitatud lõppküsimustik. Selle küsimustiku läbimisel kasutaja otseselt ei näe kuidas küsimustik on ümber lülitatud ühelt tüüpjuhtumilt teisele (antud informatsiooni saab leida tarbimisliidese vastuste ajaloo sektsioonis). Teoreetiliselt on antud küsimustiku võimalik modelleerida ka ühe tüüpjuhtumiga, aga sel juhul diagramm muutub keeruliseks ja halvasti arusaadavaks.

Tüüpjuhtumis „Vigastus“ tehakse lõppotsus, kas saata kiirabi kohale või mitte. Mõned tüüpjuhtumid kirjeldavad situatsioone, mis viivad lõppotsusteni nii kiirabi, kui ka tuletõrje brigaadi saatmise kohta. Näiteks, tüüpjuhtum „Põleng ja väike põleng“ kirjeldab kahte situatsiooni: kui põleng on suur, siis juhtimine antakse üle „Suur põleng“ tüüpjuhtumile; kui põleng on väike ja selle saab käsitsi kustutada, siis jõuakse lõppotsuseni, et vigastuse kohta pole mõtet küsida ning kiirabi ega tuletõrje pole vajalik.



Joonis 56: Häirekeskuse kutsetöötluse küsimustiku struktuur

Joonisel 56 on näidatud, kuidas häirekeskuse kutsetöötluse ülesanne on jagatud tüüpjuhtumiteks ning võimalikud teed ühest tüüpjuhtumist teisse. Iga tüüpjuhtumi sisu on võimalik vaadata loodud süsteemi modelleerimisliidese kasutades, ühendades selle töötava töövoodefinitsioonide andmebaasiga (juhend on toodud peatükis 10).



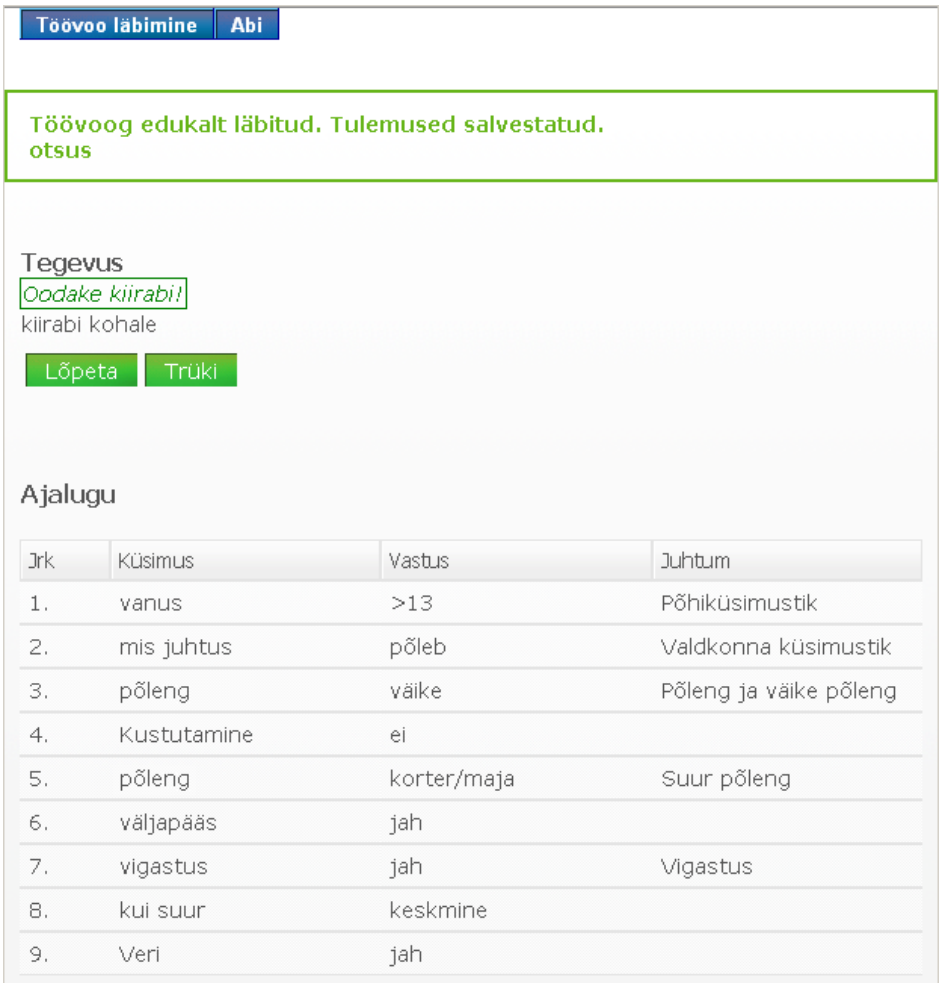
Ekraanipilt 62: "Suur põletus" tüüpjuhtumi diagramm

Ekraanipildil 62 on illustreeritud tüüpjuhtumi „Suur põletus“ küsimuste diagramm. Diagramm kirjeldab mitu situatsiooni, mille tulemuseks on lõppotsus ja üks situatsioon, mille tulemuseks antakse juhtumist „Vigastus“ tüüpjuhtumile, kus küsitakse inimese vigastusega seotud küsimusi.

Loodud illustreeriv küsimustik on salvestatud andmebaasi ning on kättesaadav nii muutmiseks, kui ka käivitamiseks. Muutmine toimub modelleerimisliidese vahendamisel ja küsimustiku käivitamine tarbimisliidese (paks või õhuke) ja töövoomootori abil. Nende komponentide paigaldamis- ja kasutamisjuhend on toodud samuti peatükis 10.

Diagrammi modelleerimisliides toetab kõiki vajalikke elemente küsimustiku koostamiseks, ning kogu selle näidis-küsimustiku modelleerimine võttis mitte rohkem kui 30 minutit aega.

Loodud küsimustiku läbimiseks oli kasutatud õhuke tarbimisliides oma platvormi sõltumatuse tõttu. Läbimine on lihtne ja intuitiivne: süsteem kasutab töövoo definitsiooni, mis on loodud modelleerimisliidese abil, küsib küsimusi ja pakub eeldefineerituid vastusi. Küsimuste vastuste abil süsteem liigub läbi eeldefineerituid küsimusi ja tüüpjuhtumeid. Küsimustiku läbimine lõpeb otsusega.



Ekraanipilt 63: Häirekeskuse kõnetöötluse küsimustiku läbimine kasutades õhukest tarbimisliidest

Ekraanipilt 63 illustreerib näidis-küsimustiku läbimise tulemust ja otsust. Eksisteerib ka võimalus näha küsimustiku läbimise ajalugu.

8. Süsteemi kasutusala ja omadused

Paljudes olukordades on kasutadaolev ressurss piiratud ja seetõttu leiduvad reeglid, kuidas reaalajas jõuda optimaalse otsuseni. Loodud töövoosüsteem on lahendus lühiajaliste ülesannete klassile, mille eesmärk on reaalajas optimaalse otsuseni jõudmine piiratud ressursi kasutamise kohta. Antud töövoosüsteem lahendab valikvastustega küsimustiku ülesannet, mis erineb teistest ülesannetest ja mille iseloomustavad järgmised omadused:

- inimene ei ole kompetentne otsustama, näiteks, vajalikke teadmiste puuduse tõttu või inimese valdkonna kogemus pole täielik
- otsustamiseks on vähe aeg – kriitilistes olukordades on vaja kiirelt reageerida
- vale piiratud ressursi otsuse tegemine ei ole rahaliselt optimaalne

Viimase omaduse näidiseks on kiirabi väljasaatmine, kus inimene helistab 112 ja küsib kiirabi ja dispetšer jõuab küsimuste küsimisega otsuseni, et sellise probleemi korral kiirabi ei ole vajalik (või vastupidi, on vajalik). Samas, kui kaks helistajat nõuavad kiirabi, küsitluse abil on võimalik paika panna prioriteete, kellele antud kiirabi on vajalikum ja kes saab oodata.

Sarnased olukorrad, kus on vaja kiiret reageerimist ja õiget otsust, tekkivad ka politseis, päästeametis, turvafirmades, haiglates, mobiilside operaatori esimese taseme tehnilises toes ja nii edasi.

Artiklis [20] toodud klassifikatsioonile loodud töövoosüsteem on **reeglitel baseeruv** süsteem, mis võib opereerida väga efektiivselt siis, kui töövoogude definitsioonid on hästi tehtud. Artikli autor väidab, et antud tüüpi süsteemid on kiired ja täpsed, sest andmeid hoitakse puustruktuurina. Seoses sellega, et küsimustiku puhul eksisteerib ka küsimuste esitamise tsüklilisus, on magistritöö raames loodud süsteemi töövoo diagrammid omavad graafkuju. Ja kuna puu on graafi erijuht, siis autori väide kehtib ka loodud süsteemi puhul.

Autori sõnul antud süsteemid omavad ka puudusi. Nimelt, definitsioonide loomine võtab päris palju aega. Selle raksuse lihtsustamiseks, süsteem on varustatud graafilise modelleerimisliidesega. Veel üheks tõsiseks probleemiks autor nimetab reeglitele baseeruva süsteemi staatilisust – ise süsteem ei kasva, nagu CBR (*Case-Based-Reasoning*) ja RDR [27] (*Ripple Down Rules*) tehnoloogiatel baseeruvad süsteemid. Selleks, et süsteem oleks abiline uues valdkonnas, tuleb valdkonna spetsialistide abil luua vastava küsimustik.

Nagu oli mainitud, tähtis optimaalne otsus ressursi kasutamise kohta sõltub ka küsimustiku loojatest. Nagu paberis [19] autor väidab, et otsuse kättesaamise õigus ja optimaalsus, samas ka kiirus, sõltub sellest kui kvaliteetselt küsimustik koostatud on. Kui küsimustiku käigus küsitakse sama infomatsiooni kohta kahel erineval viisil, siis see on ajakaotus. Eksisteerivad situatsioonid, kus ühe küsimuse vastuse abil saab automaatselt mingitest küsimustest loobuda, näiteks, küsides „Kas Teil valutab jalg või käsi?“ saadud vastus „Käsi“, kohe annab informatsiooni sellest ei ole mõtet küsida küsimusi, mis on seotud otseselt jalgadega. Seoses sellega, et reeglitele baseeruv süsteem, nagu küsimustiku läbimise süsteem, on andmetega juhitav süsteem, selle süsteemi töö kvaliteet kõvasti sõltub andmetest, mida juhtivad antud süsteemi [20]. Seega eduka süsteemi töö jaoks on nõutud kvaliteetne töövoo definitsioon.

Süsteem ei tööta CRB (*Case-Based-Reasoning*) põhimõttel – teadmiste baas ei laiene eelmiste lahendatud probleemide abil. Käsitsi teadusbaasi uuendamine ja laiendamine ei ole kõige efektiivsem viis, sest süsteemi vaatenurk on piiratud ainult loodud teadusbaasiga. Vastastikune lähenemine kõnekeskuse süsteemi loomisel, mis kasutab tehisintellekti ja andmekaevanduse metoodikat eneseõppimiseks on MCRDR (*Multiple Classification Ripple Down Rules*) süsteem, mis on kirjeldatud dokumendis [27].

9. Hinnanguline võrdlus alternatiivlahendustega

Püstitatud lühiajalise ülesande lahendus peab olema maksimaalselt efektiivne, piisav ja lihtne. See tähendab seda, et süsteem, mis on probleemi lahendus, ei oma funktsionaalsust, mida ei kasutata ülesande lahendamisel, täpsemalt, töövoo diagrammide loomisel, muutmisel ja töövoo protsessi käivitamisel.

Analüüsi käigus selgus, et olemas kolm lahenduse alternatiivi: kaks äärmust ja üks kuldkeskmine. Esimene äärmusvariant on kasutada suurt universaalset süsteemi, mis on väga avatud suurte võimalustega süsteem. Mõned süsteemid ei saanud üldse valikvastustega küsimustiku triviaalset ülesannet lahendada, mõned aga suurte pingutustega. Süsteemi kohandamine, kasutamaõppimine ja funktsionaalne üleliigsus andsid põhjust sellest variandist loobuda. Teine äärmusvariant on minimalistlik lähenemine. Süsteemi luuakse nullist, süsteem omab väga piiratud funktsionaalsust ülesande lahendamiseks. Töövoo reeglid on sisseehitatud. Sel põhjusel, töövoomootor saaks liiga nõrk ja kinnine. Lisaks nõuaks töövoo reeglite uuendamine suurt aja- ja rahakulu, kuna antud töö on programmeerimisoskustega inimese kompetentsuses. Kolmas ja kuldkeskmine tee on võta parimad omadused äärmustelt variantidelt: süsteem peab olema mitte nii universaalne nagu esimene variant ja mitte nii kinnine nagu teine. Spetsialiseeritud piisavate kitsendustega töövoosüsteem lahendab püstitatud ülesannet efektiivselt ja väikeste kuludega.

Töövoosüsteemi üldise efektiivsuse ja sobivuse all mõõdetakse erinevate kulude hulka, mis tuleb kulutada püstitatud ülesande lahenduse konstrueerimiseks ja töötavas seisus hoidmiseks. Mida vähem kulusid, seda efektiivsem ja sobivam konkreetne lahendus on ja vastupidi. Tabel 15 ühendab kokku ja võrdleb alternatiivlahenduste võimalikke kulusid. Võrdluse all on kaks äärmust varianti ja üks keskmine.

	Universaalne toode (Intalio|BPMS)	**Kinnine süsteem**	**Loodud spetsiifiline süsteem**
Süsteemi loomine	4 aastat	400 inimtundi	600 inimtundi

Süsteemi kohandamine	15 min.	10 min.	10 min.
Süsteemi kasutamaõppimine	180 min.	15 min.	30 min.
Töövoo defineerimine nullist (20 küsimust)	120 min.	0 min.	30 min.
Töövoo definitsiooni uuendamine	60 min.	180 min.	15 min.
Litsentsitasud ja paigaldustasud	suured	-	-

Tabel 15: Lahenduste hinnanguline võrdlus

Kõik andmed kinnise süsteemi kohta on saadud analüüsi abil. Analüüs baseerub sellel, et loodud spetsiifiline süsteem on kinnise süsteemi edasiareng ning kinnisel süsteemil puudub graafiline töövoo diagrammide modelleerimisliides.

Süsteemi loomise kriteerium näitab umbkaudselt palju aega võttis olemasoleva süsteemi loomine algusest peale. Intalio|BPMS süsteemi puhul see on tunduvalt pikem aeg võrreldes kahe teise lahenduse alternatiiviga.

Süsteemi kohandamine on vajalik selleks, et rakendus oskaks lahendada spetsiifilist valikvastustega küsimustiku ülesannet. Seoses sellega et Intalio|BPMS on universaalne süsteem, siis selle kohandamine võtab palju rohkem aega, kui süsteemidel, mis on spetsiaalselt loodud konkreetse ülesannete klassi lahendamiseks. Kohandamise all peetakse silmas süsteemi sätestamine, näiteks modelleerimisliidese, tarbimisliidese töövoomootoriga ühendamine loodud süsteemi puhul ja parameetrite paigaldamine, uue projekti loomine Intalio|BPMS puhul. Antud osas on spetsiifilised süsteemid jälle ees.

Süsteemi kasutamaõppimine on tegevus, ilma milleta ei saa süsteemi efektiivselt kasutada. Intalio|BPMS puhul diagrammide koostamine nõuab teadmisi standartide kohta (mis on BPMN ja BPEL). Lisaks on mõistlik läbi lugeda tootja poolt pakutud juhendid töövoo protsessi käivitamiseks. Spetsiifiliste süsteemide puhul on funktsionaalsus piiratud, seega on piiratud ka nõutav teadmiste komplekt selleks, et rakendust kasutada. Lisaks omab loodud süsteemi modelleerimisliides omab nelja graafilist elementi (küsimuse esitamise tegevus, ühendus, lõppolek, viide teisse tüüpjuhtumisse) võrreldes suure hulka BPMN 2.0 standardi poolt pakutud elementidega. Kinnise süsteemi puhul pole vaja üldse õppida töövoogude modelleerimist.

Peale vajalikke kohandamisi on süsteemid valmis töövoogude modelleerimiseks. Eesmärgiks on modelleerida küsimustiku töövoo diagramm, mis koosneb kahekümnest küsimustest. Kinnise süsteemi puhul antud aeg on null, kuna töövoo reeglid on juba sisseprogrammeeritud süsteemi loomise faasi käigus. Seoses sellega, et Intalio|BPMS ei ole otseselt välja töötatud selleks, et lahendada valikvastustega küsimustiku ülesannet, diagrammi konstrueerimine võtab tohutult palju aega. Loodud spetsiifilise süsteemi puhul on see väga lihtne ja kiire tegevus.

Nagu oli eelmistes peatükkides mainitud, töövoo uuendamine on üks tähtsamatest tegevustest. Küsimustikku ennast, küsimusi, vastuseid ja nende järgnevust tuleb aeg-ajalt muuta, selleks, et modelleeritud töövoog kasutaks optimaalse tee konstrueeritud graafi läbimisel ehk süsteem pakuks otsust võimalikult kiiresti ja võimalikult täpselt, sest otsuse täpsus ja otsuse saamise kiirus on tähtsamaid kriteeriumid paljudes alades, kus antud süsteemi kasutada saab. Näiteks, häirekeskuses, kus otsust tehakse piiratud ressursi kohta selle otsuse täpsusest ja kättesaamise kiirusest sõltub väga palju. Nimelt, Intalio|BPMS süsteemi puhul töövoo muutmisel, näiteks, uue küsimuse lisamisel, kasutaja kopeerib ekraanivormi, täidab selle vajalikke andmetega (küsimus ise ja vastusevariandid), ning ühendab selle objekti teise diagrammi objektidega kasutades tingimuslikke üleminekuid. Peale seda sätestab iga ülemineku tingimusi, mida töövoomootor kasutab töövoo protsessi käigus. Kinnise süsteemi puhul töövoo reeglite muutmine on tõsine probleem: kuna reeglid on sisseehitatud töövoomootori loogika sisse, selliste reeglite muutmine toimub programmeerija abil. Peale koodi muutmist, töövoomootori tuleb vajadusel uuesti kompileerida selleks, et kõik tehtud muudatused hakkaksid töötama. Spetsiifiline loodud töövoosüsteem ei oma selliseid probleeme. Ekraanivormid genereeritakse küsimustiku läbimise ajal tarbimisliidese poolt, töövoo defineerimise eest vastutab graafiline töövoomootori modelleerimisliides. Ei ole vaja midagi programmeerida ja omada sügavaid tehnilisi teadmisi.

Litsentsidega ja paigaldustasudega on erilised raskused kommertssüsteemidel. Näiteks, Intalio|BPMS puhul on olemas ka vabavaraline versioon, kuid see on piirangutega. Kui osta täispaketti (*Enterprise Edition)*, siis tuleb maksta selle eest alates 9500 USD või EUR (sõltub regioonist) aastas. Mõnedel süsteemidel eksisteerivad eraldi uuendus- ja hooldustasud. Olemas ka täiesti vabavaralised lahendused, aga siin eksisteerivad ka teised töövoode

modelleerimisega ja käivitamisega seotud puudused. Kinnise süsteemi ja loodud spetsiifilise süsteemi puhul on litsentskulud minimaalsed.

Universaalsete süsteemide rühm Intalio|BPMS süsteemi näol kaotab kõikides kriteeriumites, mis on toodud tabelis 4. Sel põhjusel valikvastustega küsimustiku ülesande lahendamiseks universaalsed täissüsteemid ei ole efektiivsed. Kinnine süsteem oma lihtsuse ja spetsialiseeritavuse tõttu võitis loodud spetsiifilist süsteemi. Aga magistritöö käigus loodud süsteemi tugev koht on töövoode modelleerimisel, kus kinnisel süsteemil on suured raksused. Kahe süsteemi võrdlemiseks modelleerime situatsiooni, kus oletame, et on vaja süsteemides luua kaks töövoogu ja neid uuendada iga kuu aasta jooksul. Arvutatakse ajakulu iga süsteemi puhul. Ajakulude arvutusvalem on selline:

```
sl+sk+sko+tdn*t_arv+kuud*tdu*t_arv = ajakulu kokku, kus
        sl – süsteemi loomisaeg
        sk – süsteemi kohandamisaeg
        sko – süsteemi kasutamaõppimisaeg
        tdn – töövoo nullist defineerimisaeg
        t_arv – töövoode arv uuendamiseks
        kuud – kui palju korda uuendada
        tdu – töövoo definitsiooni uuendamisaeg
```

Kinnise süsteemi puhul ajakulu arvutuskäik on selline:

`400+0.17+0.25+0*2+12*3*2 = 472.42` inimtundi

Magistritöö käigus loodud süsteemi puhul arvutuskäik on selline:

`600+0.17+0.5+0.5*2+12*0.25*2 = 607.67` inimtundi

Antud etapil kinnine süsteem näitas vähemat ajakulu, kuna antud situatsioon ei andnud töö käigus loodud süsteemile võimalust näidata oma tugevat külge ehk lihtsat modelleerimisvõimalust. Edasiseks võrdlemiseks on mugavam jälgida süsteemide ajakulu visuaalselt. Modelleerime kakskümmend situatsiooni, kus kõik parameetrid on staatilised väljaarvatud töövoode arvu (`t_arv`). Seega muudetud valem kinnise süsteemi puhul on selline:

```
400+0.17+0.25+0*t_arv+12*3*t_arv = ajakulu
```

ja loodud süsteemi puhul:

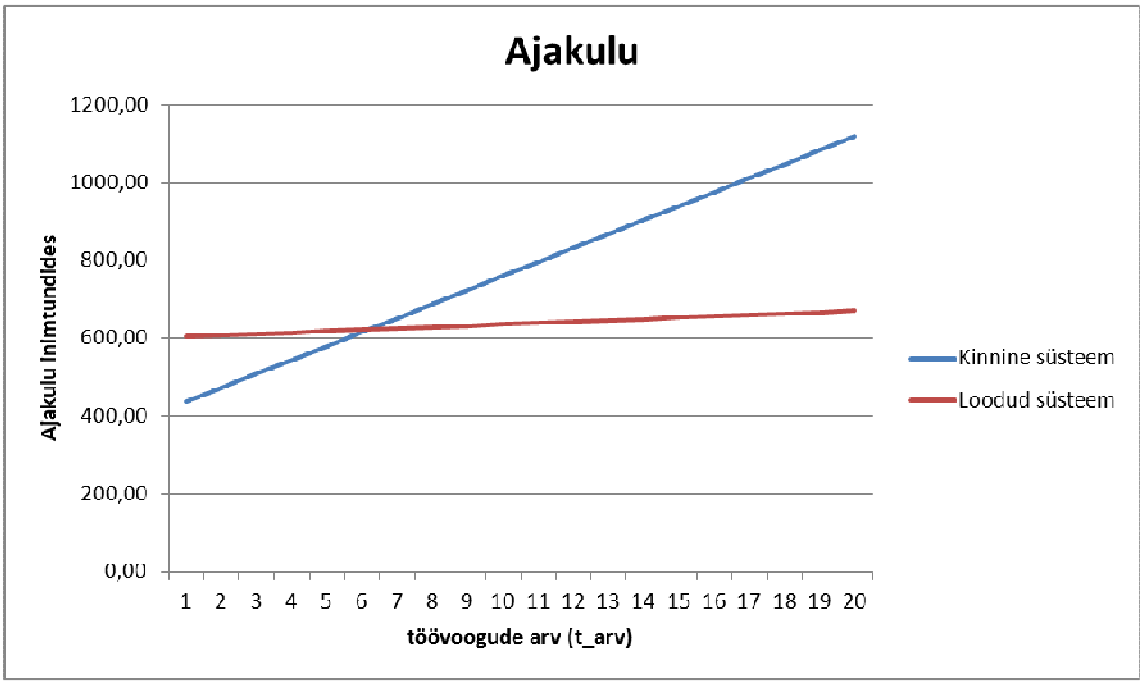
```
600+0.17+0.5+0.5*t_arv+12*0.25*t_arv = ajakulu
```

Saadud tulemused on esitatud järgmisest tabelis 16:

Töövoogude arv	Kinnine süsteem (inimtunde)	Loodud süsteem (inimtunde)
1	436.42	604.17
2	472.42	607.67
3	508.42	611.17
4	544.42	614.67
5	580.42	618.17
6	616.42	621.67
7	652.42	625.17
8	688.42	628.67
9	724.42	632.17
10	760.42	635.67
11	796.42	639.17
12	832.42	642.67
13	868.42	646.17
14	904.42	649.67
15	940.42	653.17
16	976.42	656.67
17	1012.42	660.17
18	1048.42	663.67
19	1084.42	667.17
20	1120.42	670.67

Tabel 16: Modelleeritud tulemused

Ning selle tabeli andmete alusel illustreeriv graafik joonisel 56:



Joonis 57: Ajakulu sõltuvus töövoode arvust

Graafikul on näha, et magistritöö loodud süsteemi ajakulu nõrgalt sõltub uuendatavate töövoode arvust, samas kinnine süsteem oma struktuuri ja töövoomootori kinnisuse ja nõrkuse tõttu suures sõltuvuses uuendatavate töövoode arvust. Alates situatsioonist, kus on vaja uuendada igakuuliselt seitse töövoogu, loodud süsteem võidab kinnisest süsteemist. Seega efektiivne definitsioonide uuendamisvõimaluse väärtus on tunduvalt suurem kogu kinnise süsteemi lihtsusest.

Projekti käigus oli loodud kaks erineva tehnoloogiaga tarbimisliidest, seega tegelik vajalik spetsiifilise süsteemi loomise töömaht on isegi väiksem.

Peatükki kokkuvõtteks: tulemusena loodud süsteem on odavam kui universaalse kommertssüsteemi ostmine või loomine ning kasutamaõppimine ja töövoode defineerimine. Samuti on tulemus pikemas perspektiivis odavam kui puhas programmeerimine, mis nõuab programmeerimisoskustega inimesi ka töövoode muutmiseks. Sellega, töö tõestas autorile, et lihtsustatud ülesannete jaoks võib olla optimaalsem kasutada suurtest universaalsetest töövahenditest lihtsamaid vahendeid, isegi kui need tuleb alles luua.

10. Paigaldamis- ja kasutamisjuhend

Kogu süsteemi on võimlaik proovida järgmiselt:

- luua uus küsimustik andmebaasile või kasutada olemasolevat häirekeskuse näidist
- käivitada tarbimisliides küsimustiku vastamiseks

Esimese ülesande täitmiseks kasutatakse autori poolt realiseeritud modelleerimisliidest. Rakendust on võimlaik käivitada kasutades *Java Web Start* ja JNLP teenust [18].

Selleks, et mitte koormata kasutajat lisategevustega, oli loodud mugav käivitamisprotseduur. Selle kasutamiseks on vaja käivitada JNLP faili, mis asub ülaltoodud aadressil. Peale selle faili käivitamist, kontrollitakse, kas Java Web Start on installeeritud arvutisse või mitte. Vajadusel pakutakse Java Web Start platvormi allalaadida ja installeerida. Peale eduka Java Web Start käivitamist, süsteem hakkab alla laadima käesoleva projekti süsteemi modelleerimisliidese failid ning käivitab rakenduse. Peatükis 7.5 „Rakenduse käitumisloogika" on toodud rakenduse kasutamisjuhendid.

Rakenduse katsetamiseks on loodud näidisküsimustik, mis asub projekti süsteemi andmebaasis. Neid on võimalik avada kasutades rakenduse peamenüü punkti „Open occasion...".

Selleks, et läbida loodud küsimustiku on vajalik töövoomootori olemasolu. Selle käivitamiseks on vajalik server PHP toetusega. Paigaldamiseks on vaja kopeerida töövoomootori failid, mis asuvad CD peal kaustas `/voog/engine/`, anda failidele vajalikud õigused ning ligipääsu saamiseks varustada töövoomootori andmebaasi parameetritega. Andmed töövoomootori andmebaasiga ühendamiseks on võimalik konfigureerida `/voog/engine/config/config.php` failis.

Küsimustiku proovimiseks on kaks varianti: õhukese ja paksu tarbimisliidese kasutamine. Esimese puhul on vajalik käivitada projekti, mis asub CD peal `/voog/thinClient/` kaustas kasutades *ant*-skripti. Täpsemad paigaldamis- ja käivitamisjuhendeid on võimalik leida vastavas dokumendis[42]. See liides on võimalik käivitada mis iganes süsteemi peal, kus on paigutatud Java virtuaalmasin (*JavaVM*) ja *Apache Ant*[44].

Teine võimalus on kasutada jaoks paksu tarbimisliidest *Windows* platvormi. Kasutamiseks on vaja käivitada CD plaadil asuva faili (`/voog/fatClient/FatClient/bin/Release/FatClient.exe`). Täpsemad käivitamisjuhendeid on võimalik leida vastavas dokumendis[43].

Peale tarbimisliidese käivitamist, programm kohe algab tööd algusküsimustikust. Küsimustiku läbimise lõpus salvestatakse küsimustiku läbimistee logifaili.

Tarbimisliideste täpsema käitumisloogika ja omaduste kirjeldusi on võimalik leida vastates dokumentides [44, 45].

Selleks, et proovida loodud süsteemi tööd ja katsetada modelleeritud häirekeskuse kõnetöötluse küsimustiku on paigaldatud testimiskeskkond:

1. Töövoomootor asub aadressil http://voogdb.info/engine/engine.php
2. Modelleerimisliides asub aadressil: http://voogdb.info/
3. Testimiskeskkonna andmebaasi andmed:

    Server: `sql.byethost5.org 3306`

    Kasutajatunnus (username): `voogdbin_voog`

    Parool (password): `voog123`

    Andmebaas (Database name): `voogdbin_voogdb`

Antud andmete abil on võimalik luua ühendus modelleerimisliidese ja andmebaasi vahel. Antud protseduuri lihtsustamiseks dokumendi CD plaadile on lisatud ühenduse fail (connection.xml), mis asub kaustas `/misc/`.

Kokkuvõte

Käesoleva magistritöö eesmärgiks oli lihtsa ja odava lahenduse leidmine valikvastustega küsimustiku ülesande jaoks. Selleks analüüsiti olemasolevaid universaalseid süsteeme ja püsiprogrammeeritud lahenduse varianti. Neid kaks äärmust alternatiivi tundsid mittesobivad oma keerukuse-kohmakuse või kinnisuse tõttu. Tekkis hüpotees, et püstitatud ülesanne on lahenduv odavamalt, kui algselt välja mõeldud kaks alternatiivi.

Oma hüpoteesi kinnitamiseks pandi kokku tudengite töörühm (magistritöö autor ja kolm bakalaureusetöö kirjutajat) ning loodi eksperementaalne piisavate kitsendustega töövoosüsteemi erijuht, küsimus-vastusüsteem, mis koosneb töövoomootorist, tarbimisliidestest ning modelleerimisliidesest.

Autori panus seisnes projektijuhtimises ja modelleerimisliidese loomises. Töös analüüsiti ja kirjeldati kogu projekti ülesehitust, loodava rakenduse funktsionaalseid ja andmenõudeid ning disaini. Lisaks kavandati projekti andmebaas ning graafiline kasutajaliides.

Töö hetkeseisuks on valmis autori juhtimisel valmistatud kõik terviksüsteemi komponendid ja nende hulgas ka autori enda poolt loodud küsimustike modelleerimisliides.

Modelleerimisliidese loomisel kasutati ettemääratud funktsionaalseid nõudeid ning loodi klasside struktuur nende nõudmiste täitmiseks. Rakenduse klassid jagati eri gruppideks, mis täidavad erinevaid ülesandeid.

Iga rakenduse visuaalne osa loodi eraldi moodulina ning lõplik rakenduse kasutajaliides konstrueeriti esialgu skemaatiliselt ning peale seda pandi erinevad funktsionaalsed osad kokku üheks rakenduseks. Prooviti luua kasutajaliidese disain, mis lihtsustaks kasutaja tööd: rakenduses on suurim osa funktsionaalsusest nähtav rakenduse parempoolses osas, kus on toodud nii instrumentaalne (instrumentide), kui ka tüüpjuhtumi ja objektide infopaneelid. Rakenduse hetkeinfo näitamiseks oli lisatud alumine info paneel.

Hinnangulise võrdluse abil on tõestatud, et loodud süsteem on efektiivsem kui kaks alguses väljamõeldud alternatiivi, seega hüpotees on äratõestatud. Loodud eksperimentaalne süsteem nõuab lisakontrolli enne reaalset kasutamist.

Question Flow Management System

Master's thesis (40 pts.)

Dmitri Afanasjev

Abstract

Workflow process is movement of documents and tasks through working process. More precisely, workflow describes actions from the different sides:

- how the task is structured
- who carries out them
- how are they synchronized
- how information moves to carry out performance of the task and to keep up their performance

Workflow system is application which helps the organization to specify, fill, supervise and coordinate working routines.

The master's thesis's main purpose is to find lightweight and effective solution for questionnaire with predefined single-selection answers. Questionnaire purpose is to reach final decision about limited resource in real time. An example of that questionnaire is emergency centre call processing, where the final decision is to be reached as the result of asking questions from caller and receiving answers from them. To solve current problem the existed solutions (existing workflow management systems and hard-coded system) for that problem should be analysed. The existed solutions have a lot of drawbacks; therefore new special workflow-based system is needed. The author's task was to supervise creation of special system of workflow process and to create the graphic module for creation and manipulations with data (modeling tool), which is able to communicate with database and to represent the user convenient visual tools for creation and data processing.

Basic element of specific system of workflow process is the *question* which has the limited amount of the predefined answers. Each question conducts:

- to a following question
- to the reference to other questionnaire
- to the end of questionnaire

Before dealing with a following question, the system can offer training as passages of intermediate actions.

Such process can use the telephone dispatcher for management of work (for example the dispatcher of the center of rescue or the dispatcher of police).

The considered work flow system consists of following components:

- The user interface
- The engine which is an average layer between the user interface and databases. Supplies the user interface with desirable data and saves results of questionnaire.
- The modeling tool of the system

In the past year under direction of the author of current work two different graphical user interfaces, structure of a database and the data transfer protocol between user interface and system engine have been created. In that year, the modeling tool and related document have been created.

Current document is divided into parts:

- Introduction, or a background
- The basic part of the document
- The conclusion
- The used literature
- Possible additions, including a program code of the created module

The final purpose of the given work was to realize workflow system, question – answer system which consists of the engine, clients and workflow modeling tool. The project, functionality and design of the created is analysed and described in current document.

At present, under control of the author, components of developed system, including the modeling tool are ready all. For creation of the graphic program of modeling tool were used

functional requirements, which were created prior to the beginning of programming. The created structure of classes has been divided into groups, where each group is dealing with own problem.

Each visual part of the program is developed as the separate module. The graphic design of the program was developed prior to the beginning of realization. There was a purpose to develop graphic user interface, which would facilitate work of the user: the most part of functionality of the program is located in the right part of program window, where are presented the panel of tools, the information of a questionnaire and the information of selected graphic object. For display of changing information as coordinates of the mouse, the panel which is in the bottom part of program window is created. Further validation of the proposed system is needed.

The current document is provided with CD, which contains additional data, such as modeling tool source code, administrative module class diagram, database technical report and database master script.

Kasutatud kirjandus / Materjalid

Dokumendi koostamisel ja programmi kirjutamisel kasutati järgmiseid infoallikad ja materjale:

[1] L. Fisher. *Workflow Handbook*, 2005. Future Strategies, Inc., 320 lk

[2] S. Jablonski and C. Bussler. *Workflow Management Systems: Modeling, Architecture, and Implementation*, 1996. Thomson Press, 351 lk.

[3] Helmer Aaviksoo. *Kasutaja vastustel põhineva töökulgemise diagrammistiku läbimise kasutajaliides*, Bakalaureusetöö, 2006

[4] *MircoOLAP Database Designer for MySQL 1.9.10*. http://www.microolap.com/products/database/mysql-designer/ (viimati vaadatud: 24.07.2010)

[5] *NetBeans IDE 6.8*. http://www.netbeans.org/ (viimati vaadatud: 20.05.2010)

[6] *Adobe Photoshop CS5*. http://www.adobe.com/products/photoshop/index.html (viimati vaadatud 21.07.2010)

[7] *Eclipse IDE 3.5 GMF Edition*. http://www.eclipse.org/ (viimati vaadatud 10.04.2010)

[8] *Apache Batik SVG Toolkit 1.7*. http://xmlgraphics.apache.org/batik/ (viimati vaadatud 20.07.2010)

[9] *MySQL Workbench 5.2.25*. http://dev.mysql.com/downloads/workbench/5.2.html (viimati vaadatud 20.04.2010)

[10] *MySQL Connector/J 5.1.13*. http://dev.mysql.com/downloads/connector/j/5.1.html (viimati vaadatud 24.07.2010)

[11] *Batik SVG Toolkit for Java overview, principles, documentation*. http://xmlgraphics.apache.org/batik/ (viimati vaadatud 21.05.2010)

[12] Элена Иванова, Максим Вершинин. *Java 2 Enterprise Edition*, Санкт – Петербург, 2003

[13] James W. Cooper, *Java Design Patterns*. http://www.javacamp.org/designPattern/ (viimati vaadatud 23.04.2010)

[14] *MySQL Java Database Connectivity*. http://dev.mysql.com/usingmysql/java/ (viimati vaadatud 20.04.2010)

[15] Marc Loy, Robert Eckstein, Dave Wood, James Elliott, Brian Cole. *Java Swing: Menus and Toolbars, Part 1 & 2*, O'Relly, 2002

[16] *Java SE 1.6*. http://java.sun.com/javase/downloads/index.jsp (viimati vaadatud 15.03.2010)

[17] *MySQL Server 5.1*. http://dev.mysql.com/downloads/mysql/ (viimati vaadatud 25.07.2010)

[18] John Zukowski. *Deploying Software with JNLP and Java Web Start, August 2002* http://java.sun.com/developer/technicalArticles/Programming/jnlp/ (viimati vaadatud 22.07.2010)

[19] K. Morton, C. Carey-Smith, and K. Carey-Smitrh. *The QUEST Questionnaire System*. In *Proceedings of the 2nd ANNES*, pp. 214-217. IEEE Computer Society, 1995.

[20] Mark Kriegsman, Ralph Barletta. *Building a Case-Based Help Desk Application*, pp. 18-26, IEEE Educational Activities Department, 1993

[21] Enis Afgan, Jeff Gray, Purushotham Bangalore. *Using Domain-Specific Modelling to Generate User Interfaces for Wizards.* University of Alabama at Birmingham Department of Computer and Information Sciences

[22] J. Durkin. *Expert Systems: Design and Development*, 1st ed. Englewood Cliffs, NJ: Macmillan Coll Div, 1998.

[23] L.Lou, J.Liu, L.Shao, W.Lu, M. Ye. *A context-aware smart-call-center solution: Improving customer service for online games*, pp. 145-160, IBM Systems Journal, vol. 45, no. 1, 2006

[24] Marcello La Roca, Johannes Lux, Stefan Seidel, Marlon Dumas, Arthur H. M. ter Hofstede. *Questionnaire-driven Configuration of Reference Process Model*, BPM Group, Queensland University of Technology, Australia, 2006

[25] Wei Xu, Alexander I. Rudnicky. *Task-based dialog management using an agenda*, pp. 42-47, Association for Computational Linguistics, 2000

[26] *Appliance call center: a successful mixed-initiative case study*. AI Magazine, 22-07-2007, http://goliath.ecnext.com/coms2/gi_0199-6769969/Appliance-call-center-a-successful.html (viimati vaadatud 21.07.2010)

[27] Megan Vazey, Debbie Richards. *Troubleshooting At The Call Centre: A Knowledge-based Approach*, pp. 721-726, Computing Department Macquarie University

[28] Peter Fettke, Peter Loos, and Jörg Zwicker. *Business Process Reference Models: Survey and Classification*, pp.1-15, Johannes Gutenberg University Mainz, Information Systems and Management, Germany, 2005

[29] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. *Workflow Patterns. Distributed and Parallel Databases*, 14(1): 5–51, 2003.

[30] David Hollingsworth. *Workflow Management Coalition. The Workflow Reference Model*, pp. 6-37, The Workflow Management Coalition, Document Number TC00-1003, 1995

[31] Michael zur Muehlen. *Workflow-based Process Controlling. Foundation, Design, and Application of workflow-driven Process Information Systems*, Logos, Berlin, 2004

[32] Matthias Kloppmann, Dieter Koenig, Frank Leymann, Gerhard Pfau, Alan Rickayzen, Claus von Riegen, Patrick Schmidt, Ivana Trickovic. *WS-BPEL Extension for People – BPEL4People*. IBM and SAP, 2005

[33] Peter Niederlag. *Workflow Management with CMS TYPO3 and WFMS Enhydra Shark*, Niekom Netservice, 2005

[34] *Business Process Modeling Notation (BPMN) White Paper ver. 1.0*, BPMI.org, 2004, http://www.bpmn.org/Documents/BPMN_V1-0_May_3_2004.pdf (viimati vaadatud 11.05.2010)

[35] *Workflow Patterns*. http://is.tm.tue.nl/research/patterns/ (viimati vaadatud 21.07.2010)

[36] *Workflow Patterns*. http://www.workflowpatterns.com/ (viimati vaadatud 21.07.2010)

[37] Stephen A. White. *Using BPMN to Model a BPEL Process*, IBM, 2005

[38] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, G. Pfau, K. Plösser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, M. Zeller. *Web Services Human Task (WS-HumanTask), Version 1.0*. Adobe, BEA, Oracle, Active Endpoints, IBM, SAP. 2007

[39] Gliffy Online Diagramm Designer. www.gliffy.com (viimati vaadatud 21.07.2010)

[40] Rocket Ansaplus Call Center. http://www.rocketuk.com/software/call_centre_software.htm (viimati vaadatud 21.07.2010)

[41] Alec Sharp, Patrick McDermott. *Workflow Modelling. Tools for Process Improvement and Application Development*. pp.13-32, Artech House, 2001

[42] Alar Tammik. *Töövoo läbimise süsteemi õhuke kasutajaliides*, Bakalaureusetöö, Tartu, 2007

[43] Roman Normann. *Töövoogude läbimise kasutajaliides – aknapõhine rakendus*, Bakalaureusetöö, Tartu, 2007

[44] Apache Ant. http://ant.apache.org/ (viimati vaadatud 26.07.2010)

[45] Robert M. Shapiro. *XPDL 2.1 Integrating Process Interchange & BPMN*. The Workflow Management Coalition, 2008

[46] TRANSFLOW Nederland BV. *COSA Workflow Patterns*, 2003, http://www.workflowpatterns.com/vendors/documentation/vc_cosa.pdf (28.07.2010)

[47] COSA Suite Interactive Demo. http://www.cosa.nl/BPM_Demo.html (28.07.2010)

Lisad

Lisa 1. Rakenduse lähtekood

Tööle on lisatud:

- administreerimisliidese lähtekood, mis asub CD plaadi peal, kaustas `/voog/admin/`
- töövoomootori lähtekood, mis asub CD plaadi peal, kaustas `/voog/engine/`
- õhukese tarbimisliidese lähtekood, mis asub CD plaadi peal, kaustas `/voog/thinClient/`
- paksu tarbimisliidese lähtekood, mis asub CD plaadi peal, kaustas `/voog/fatClient/`

Lisa 2. Rakenduse klasside diagramm

Tööle on lisatud administreerimisliidese rakenduse klasside diagramm, mis asub CD plaadil kaustas `/voog/admin/class diagramm/`

Lisa 3. Süsteemi andmebaasi *master-script*

Tööle on lisatud uuendatud andmebaasi *master-script*, mis loob andmebaasi skeemi koos kõikide tabelitega ja muude andmetega. Samuti *script* lisab ka näidisandmeid. Asukoht: CD plaat, kaust `/voog/andmebaas/master_script_w_questionnaire.sql`

Lisa 4. Abifailid

Tööle on lisatud abifailid, mis asuvad CD plaadi peal, kaustas `/misc/`. Abifaili hulka kuulub olemas oleva andmebaasi ühenduse fail (*descriptor*), Helmer Aviksoo, Alar Tammiku ja Roman Normanni bakalaureuse tööd ning käesolev töö pdf-kujul [3, 42, 43].

Lisa 5. Andmebaasi tabelite tehniline raport

Andmebaasi tabelite tehnilised andmed olid koostatud kasutades MircoOLAP Database Designer for MySQL 1.9.6 [4]. Samasugune fail on lisatud CD plaadile ja asub kaustas `/voog/andmebaas/voog_db_report/`