Tallinna Ülikool
Informaatika osakond

TARKVARARAKENDUSTE INTEGRATSIOONI PRAKTIKAD

Magistritöö

Paavo Peterson
Juhendajad: Ants Sild, Ph.D
Priit Parmakson, M.Sc

Tallinn 2005

Autor: .....”.....”..... 2005. a.

Juhendajad: .....”.....”..... 2005. a.

.....”.....”..... 2005. a.

Osakonna juhataja: .....”.....”..... 2005. a.

1 TÖÖ EESMÄRK

1.1 Tarkvararakenduste integratsiooni aktuaalsus

Tänapäeva organisatsioonides otsitakse lahendusi, kuidas toetada paremini äriprotsesse, tagada suurem infosüsteemi töökindlus ja vähendada kulusid süsteemi ülalhoiule. Üheks selliseks võtmesõnaks on integratsioon, mis tagab eraldiseisvate rakenduste töö ühtses, koostalitlevas süsteemis, mida on kergem hallata ja odavam ülalpidada. Integratsioon pole siiski mingi võlusõna, mille abil kõik asjad kohe maagilisel moel toimima hakkavad. Eri platvormidel ja eri aegadel teostatud rakendusi on seejuures üsnagi keeruline panna omavahel suhtlema ja andmeid vahetama. Kuidas seda teha targalt ja läbimõeldult, et liidestamised oleksid töökindlad ja rahuldaksid ka edaspidi esilekerkivaid vajadusi? Läbimõtlemata integratsiooni tulemusel võib tekkida olukord, kus organisatsioonis on tekkinud nn. "tihedalt" liidetud rakenduste rägastik, mille tõttu rakendused on üksteisest üleliia sõltuvad ja suudavad korraga suhelda ainult ühe, konkreetselt külgeliidetud rakendusega. Sellist keskkonda on aga kulukas muuta ja keeruline hallata. Ka uue funktsionaalsusega rakenduse arendamisel tekib tihti vajadus liidestada seda mõne olemasoleva rakenduse või komponendiga või tarbida juba olemasolevaid andmeid. Olemasolevate funktsioonide ja andmete dubleerimine pole kuigi efektiivne tegevus.

Integratsiooni projektid on suures osas ebaedukad, kuna firmadel puudub strateegiline vaade projektile, selgub *The Butler Group*'i poolt tehtud raportist *(ComputerWire, 2002)*. Uuringufirma *Standish Group* väitel ületavad ligi 90% andmete integratsiooni projektidest plaanitud eelarvet keskmiselt 66% ulatuses või kukuvad üldse läbi *(British Computer Society, 2002)*. Need väited viitavad suurtele probleemidele tarkvarasüsteemide integreerimisel ja jätkuvale teema aktuaalsusele.

Rakenduste integreerimine moodustab tarkvaraarendajate jaoks omaette valdkonna. Seda püüab ka antud töö vaadelda.

Tarkvaraprojekte, kus integreerimise maht on suur või liidestamise olulisus süsteemi jaoks on määrav, võib vaadelda kui infotehnoloogiliste projektide alamliiki. Projekti edukaks õnnestumiseks peavad liidestamised veatult toimima ja seega on nende arendus ka erilise tähelepanu all.

Autori hinnangul on erinevate süsteemide integratsiooni sisaldavad projektid tavaprojektidest keerulisemad nii tehnoloogilise kui juhtimise poole pealt. Reeglina suurendab erinevate liideste teostus riske ning määramatust ja seetõttu võib ootamatuid probleeme projekti kestel tavalisest sagedamini ette tulla. Antud uurimus püüab anda ülevaate integratsioonilahendustest ja probleemidest, ning pakkuda välja ka soovitused integratsioonitööde läbiviimiseks.

Töö eesmärgid. Töö käsitleb tarkvararakenduste integratsiooni probleeme. Eesmärkideks on:

- Selgitada välja oluline integratsiooniteemaline erialane kirjandus ja koostada analüütiline ülevaade aktuaalsetest integratsiooni vormidest ja mõistetest.
- Koguda empiirilisi andmeid ja analüüsida Eestis läbiviidud rakenduste integratsiooni praktikaid ja praktikute arusaamu.
- Läbitöötatud kirjanduse ja kogutud empiirilise andmestiku põhjal tuua välja soovitused rakenduste integratsiooni projektide praktiliseks läbiviimiseks Eesti kontekstist lähtudes.

Töö meetodid. Töö eesmärkide saavutamiseks olen teinud analüütilise kokkuvõtte kirjandusest saadud materjalidest, mis käsitlevad integratsiooni printsiipe, nende erivorme ja juhtimisprobleeme. Eestikeelse kirjanduse osakaal on selles vallas väga väike. Vajalikke materjale tuli otsida nii internetist kui raamatukogudest. Paljude integratsiooni käsitlevate artiklite kõrval olen põhimaterjalina kasutanud järgmisi raamatuid:

- *Microsoft Integration Patterns: patterns & practices (Trowbridge, 2004)*.
- *Enterprise integration patterns: designing, building, and deploying messaging solutions (Hohpe, 2003)*.
- *Next generation application integration: from simple information to Web services (Linthicum, 2003)*.

Nende raamatute autorid on rahvusvaheliselt tunnustatud eksperdid ja omavad ka praktilisi kogemusi integratsiooni vallast.

Tulenevalt tarkvaraarenduse kultuuri erinevustest, rahalistest võimalustest ja muudest asjaoludest, ei pruugi mujal maailmas läbiviidud praktikad ja kogemused üks-ühele siiski sobida Eesti konteksti. Seetõttu pidasin vajalikuks koguda ka empiirilist andmestikku meil läbiviidud integratsiooni praktikatest ja praktikute arusaamadest. Selleks viisin läbi intervjuud seitsme inimesega, kellel on vastav kogemus olemas projektijuhi või analüütikuna. Intervjueeritavad on tegevad tarkvaraarendusega tegelevates firmades nagu Abobase Systems, Cybernetica, TieotoEnator Eesti, IB Krates ja suuremates organisatsioonides, kus on läbi viidud vastavaid projekte nagu Maksu- ja Tolliamet ning Hansapank. Intervjuude vorm küsitlusmeetodina tundus õige valik, kuna vastuste saamine on kindlamini tagatud ja see võimaldab ka küsitluse käiku dünaamiliselt juhtida, vastavalt intervjueeritava kogemustele ja teadmistele.

Töö meetoditena olen analüüsinud erialast kirjandust ja empiirilisi andmeid, tuues välja enim käsitlust leidnud probleemid ja soovitused nende lahendamiseks.

Töö ülesehitus. Töö ülesehitus koosneb neljast suuremast peatükist.

Esimene peatükk annab ülevaate antud teema aktuaalsusest, püstitab magistritöö eesmärgid ja kirjeldab uuringu meetodeid. Lisaks käsitleb integratsiooni mõistet, põhjuseid ja integratsiooni toodete ja teenuste turu arengut.

Teine peatükk käsitleb kasutatud kirjanduse põhjal integratsiooni eri tüüpe, vorme ja väljakujunenud standardlahendusi ehk mustreid. Vaadeldakse, mis on rakenduste integratsioonile iseloomulikud jooned ja milliseid integratsiooni tüüpe ja vorme kasutatakse ning antakse ka hinnang nende tugevatele ja nõrkadele külgedele.

Lisaks kirjeldatakse peatükis põhilisi rakenduste integratsiooni juhtimise probleeme ja soovitusi nende lahendamiseks.

Kolmandas peatükis käsitletakse intervjuudest pärineva materjali põhjal praktilist kogemust Eestis läbiviidud rakenduste integreerimisel. Kirjeldatakse küsitluse meetodeid, analüüsitakse konkreetseid integratsiooni juhtumeid ja praktikute arvamusi ning tuuakse välja põhilised probleemid ja soovitused integratsioonitööde läbiviimiseks.

Neljas peatükk teeb järeldused kogu tööst, võrreldes kirjanduses käsitletud ja kohalikku kogemust rakenduste integratsiooni alal.

Töö lisadena on esitatud praktikutega läbiviidud intevjuude täistekstid.

1.2 Rakenduste integratsiooni mõiste

Integratsiooni on tõlgendatud kui "osade ühendamist tervikuks" *(Võõrsõnade leksikon, 1983)*. Kui tuua siia juurde infosüsteemid, siis on integratsiooni defineeritud kui "infosüsteemi erinevate osade vahelise koostöö loomine terviksüsteemi loomise eesmärgil" *(BCS, 1999)*. Lisaks intergatsioonile on uuringus veel olulisel kohal tarkvararakendused. Tarkvararakenduste kohta on käibel sellised definitsioonid: "ükskõik milline osa tarkvara süsteemist, mis omab lõppkasutaja funktsionaalsust" või siis "arvuti programm, mis on disainitud inimestele abiks teatava töö läbiviimisel"*(Trowbridge, 2004)*.

Rakenduste integratsiooni on David Linthicum defineerinud järgmiselt: "rakenduste integratsioon on strateegiline lähenemisviis siduda mitu infosüsteemi kokku, nii teenuse kui info tasandil, toetades nende võimet vahetada infot ja mõjutada protsesse reaalajas" *(Linthicum, 2003,)*. Antud definitsioon seab integratsioonile üsna kõrged nõudmised ja tõstatab ka küsimuse, milline võiks siis olla õige strateegiline lähenemisviis? Veel on defineeritud rakenduste integratsiooni kui: "protsess kommunikatsiooni loomiseks rakenduste vahel, mis täidavad erinevaid funktsioone, nagu kiendihaldussüsteem, arvetesüsteem, logistika, üldiinfo väljajagamine ja uuendamine nagu kliendiandmete detailid, teenused, tootekataloogi informatsioon jne." *(Informatica, 2005)*.

Töö autorina, tundub mulle üks sobilikumaid definitsioone: "rakenduste integratsioon on protsess informatsiooni ühendamiseks ühest rakendusest teise. Saavutatav rakenduste integratsioon on varieeruv oma raskusastmelt, sõltuvalt tarkvara võimalustest" *(Modern Practice, 2005)*. See definitsioon ei sea mingeid piiranguid, kuidas rakendused ühendatakse, aga samas mainib olulise osana integratsiooni raskusastme, mis sõltub tarkvara võimalustest. Ehk siis antakse mõista, et integratsioone võib olla väga erineval tasemel ja sellest tuleneb ka keerukus.

Rakenduste integratsiooni tähistatakse kirjanduses tihti lühendiga *EAI (Enterprise Application Integration).* See mõiste hõlmab endas kombinatsiooni protsessidest, tarkvarast, standarditest ja riistvara integratsioonist *(Gormly,2001)*.

<u>Integreerimine või liidestamine?</u> Järgnevalt püüan selgitada, mida nende kahe termini all mõistetakse ja mis on nende erinevus.

Liides on "kahe funktsionaalüksuse vaheline ühispiir, kus kehtivad koostöötingimused, mis puudutavad funktsioone, füüsilisi ühendusi, signaalivahetust jms." *(Hanson, Tavast 2003).* Eelnevast definitsioonist tuletades, võiks tähendada liidestamine kahe funktsionaalüksuse vahelist ühendust teostama viisil, kus nende üksuste funktsioonid, füüsilised ühendused, signaalivahetused jms. puudutavad koostöötingimused on rahuldatud.

Küsitledes integtratsiooni kogemusega analüütikuid ja projektijuhte võis täheldada kahe leeri kujunemist nende mõistete tõlgendamisel: üks osa mõistab

integreerimise all seda, kui rakendused vahetavad üksteisega kahepoolselt infot ja funktsioone, on võimelised töötama ühtse süsteemina. Liidestamise puhul on aga tegu nõrgema seosega, näiteks üks rakendus saadab teisele andmeid ja tagasisidet ei saa. Teine osa küsitletavatest peab integreerimist ja liidestamist sünonüümideks või siis vastastikuse kokkuleppe küsimuseks ja ei tee sisulist vahet nendel terminitel.

Autori arvates on keeruline tõmmata konkreetset piiri tarkvaraarenduses integratsiooni ja liidestamise vahele. See sõltub eelkõige kontekstist, millisel tasemel ja millise funktsionaalsusega süsteeme integreeritakse või liidestatakse. Raske on paika panna kindlaid tingimusi, millest alates liidestamine kasvab üle integratsiooniks. Kui näiteks rakendused vahetavad ainult andmeid omavahel andmebaasitasandil, siis võib seda nimetada liidestamiseks- või integratsiooniks andmete tasemel.

Seetõttu käsitlen neid termineid antud töös sarnastena - ka liidestamine on integratsioon, kuid seda võib mõista väiksemas mahus integratsioonina.

Integratsiooni põhjused. *Illuminata Inc.* analüütik Jonathan Eunice on öelnud: "IT võtmeväärtuseks on omavaheline käideldavus ja integratsioon. Firma väärtus sõltub paljuski kooskõlas töötavate süsteemide olemasolust" *(Hall, 2002)* . Kahtlemata võib nõustuda, et kooskõlas toimivad süsteemid tõstavad firma väärtust ja aitavad paremini läbiviia äriprotsesse. See võib olla üks paljudest integratsiooni põhjustest. Selguse huvides toon siinkohal välja mõned rakenduste integreerimise peamised põhjused:

- Äriprotsesside automatiseerimine. Äriprotsessi terviklikuks läbiviimiseks kasutame sageli mitmeid eri rakendusi. Ühendades need rakendused, saame automatiseerida kogu protsessi. Automatiseerides protsessid erinevate rakenduste vahel paraneb teenuse kvaliteet ja vähenevad kulud teenuse pakkumisel. Ettevõttetel tekib vähem vigu andmete ühest rakendusest teise viimisel (näiteks tellimuste telefoni/faxi teel vastuvõtmised, andmete käsitsi sisestamine rakendusse) ja kliendi jaoks paraneb teenuse kvaliteet. Näiteks võib klient jälgida oma tellimuse staatust internetist või saada viimast infot sellekohta klienditeenindajalt.
- Äriprotsesside kiirendamine ja paindlikkus. Protsesside automatiseerimisel paraneb ka protsessi läbiviimise kiirus. Organisatsioonidel on strateegiline vajadus saavutada äriprotsesside suurem paindlikkus ja kiirus. See võimaldab reageerida kiiremini muudatustele ja tõsta konkurentsieelist. Et äriprotsessid oleksid paindlikud, peab ka tehnoloogia olema paindlik. Hea integratsioonilahenduse korral on lihtne muuta äriprotsessi või selle rõhuaetust IT keskkonnas.
- Juhtimisotsuste toetamine. Õiged juhtimisotsused on kriitiliselt tähtsad ja selleks on oluline omada korporatiivinfot erinevatest ettevõtte valdkondadest. Näiteks võib olla süsteem, kus ärijuhid saavad otsuste

tegemisel kasutada ühte keskkonda, kuhu kuvatakse erinevatest ettevõtte süsteemidest pärinev info.

- Organisatsiooni ühendamine. Integratsiooni võib kasutada kui vahendit ettevõtte üksuste ühendamiseks, kaasates uusi kasutajaid väljaspool korporatiivset tulemüüri. Integreerimine võimaldab äriüksustel ja kolmandatel pooltel vastastikku toimida kui ühendatud üksused, lihtsustades olulisi äritegevuse funktsioone. Näiteks liidetakse emafirma ja tütarfirmade infosüsteeme, et vajalikud andmed oleks kõigile operatiivselt kättesaadavad. Või liidetakse firma infosüsteemiga äripartnerite infosüsteeme, nagu tarnijad, logistikafirmad jne.
- Organisatsioonide ühinemine. Suurettevõtted ostavad üles väikeettevõtteid või ühinevad. Olemasolevaid süsteeme pole seejuures tihti võimalik lihtsalt väljavahetada ja need tuleb ühildada omavahel.

Need on mõned põhjused, mis lähtuvad eelkõige organisatsioonilisest vaatest, miks rakendusi liidetakse. Infosüsteemi vaatepunktist võivad aga integratsiooni läbiviimiseks olla järgmised taktikalised ajendid:

- Infosüsteemi terviklikkus. Erinevate rakenduste liitmine sobival moel lihtsustab süsteemi haldamist ja vähendab muudatuste maksumust. Erinevates keskkondades paiknevate ja eri nõudmistega rakenduste asemel on integreeritud süsteemi puhul parem kontroll selle üle. Näiteks on ühtset süsteemi tunduvalt kergem monitoorida kui kõiki eraldipaiknevaid rakendusi.
- Andmete ja funktsioonide dubleerimise vähendamine. Integratsiooni puhul välditakse andmeliiasusest tulenevaid probleeme. Samuti on võimalik korduvkasutada funktsioone, mis mingis rakenduses on juba realiseeritud. Näiteks võivad erinevad rakendused kasutada ühte moodulit, millega hallatakse kliendiandmeid.
- Süsteemi arenengu võimaldamine. Integratsioon pakub paindlikku IT keskkonda, lubades pärandsüsteemidel laieneda uuele funktsionaalsustasemele või migreerida funktsionaalsust, kahjustamata teisi olemasolevaid süsteeme *(Schmelser, 2003)* .
- Olemasolevatele rakendustele lisaväärtuse loomine – integratsioon võimaldab firmadel kasutada paremini olemasolevaid süsteeme, laiendades neid uute rakendustega *(Schmelser, 2003).*

Rakenduste integratsiooni toodete ja teenuste turu areng on käinud käsikäes üldise infosüsteemide arenguga. Rakenduste arenedes tekkis ka vajadus neid omavahel integreerida, et pakkuda kasutajale suuremat funktsionaalsust ja mitmeid sellest tulenevaid hüvesid.

Ettevõtete rakendused olid 1960…-70-ndatel aastatel lihtsa disaini ja funktsionaalsusega ning nende põhiline eesmärk oli dubleerida manuaalseid protseduure arvutil *(Gormly,2001)*

1980-ndatel hakkasid korporatsioonid mõistma rakenduste integratsiooni väärtust ja vajalikkust. Rakendused, mida üritati integreerida olid reeglina juba valmis tehtud ja töötasid oma keskkonnas. Pakuti välja uusi teooriaid integratsiooniprobleemi lahendamiseks, nagu näiteks "tsentraalne andmete sõnaraamat", käibel olid veel mitmed "andmehoidla teooriad" ja andmete modelleerimise teooria. Viimase teooria kohaselt tuli teostada vaid õige andmemudel lahendamaks integratsiooniprobleeme. Kuid kõik need teooriad ei olnud paraku edukad, sest need teooriad oleks olnud rakendatavad ainult siis, kui oleks võimalus rakendused ümberprogrammeerida. Rakendused, aga olid teostatud viisil, kus see polnud lihtsalt võimalik, nad olid nö. "kivisse valatud". Võimetus rakendusi ümberprogrammeerida sai kõigi nende teooriate Achilleuse kannaks *(Inmon, 1999).*

Kuni 90-ndate keskpaigani oli andmete ladustamine populaarne meetod pakkuda andmete tasemel integratsiooni. Selline integratsioon lubab erinevatel rakendustel jagada ühte andmehoidlat, aga ei tee midagi äriprotsesside automatiseerimiseks. Kui andmehoidla lülitub tööst välja, siis rakendused, mis on sellega seotud, kaotavd ligipääsu andmetele, tehes need kasutuks missioonikriitilistele rakendustele *(Thoughtworks, 2002)*.

1990-tel tekkisid tänapäevasemad lähenemised integratsioonile. Kasutusele võeti *EAI (Enterprise Application Integration),* mis hõlmab endas kombinatsiooni protsessidest, tarkvarast, standarditest ja riistvara integratsioonist *(Gormly,2001)*. Sellest ajast hakkasid turule ilmuma ka mitmed integratsioonialased tarkvaratooted ja rakenduste integratsiooni teema muutus ettevõtetes aktuaalsemaks.

Vastavalt *Wintergreen Research* uurimusele *(Thoughtworks, 2002)* oli ülemaailmne integratsioonilahenduste müüjate müügitulu plaanitud ületada 500 miljonit dollarit aastal 1999, ligikaudu 900 miljonit dollarit 2000 aastal ja maksimaalselt 7,3 miljardit 2005 aastal. Turu ennustusi on olnud mitmeid, näiteks on *Meridien Research* ennustanud, et *EAI* tehnoloogiate turg kasvab umbes 22% aastas ulatudes 12,5 miljardi dollarini aastaks 2006 *(InformationWeek, 2002)*.

Hilisemates uuringutes on ka Wintergreen Research täpsustanud prognoose ja pakkunud välja järgmised numbrid integratsioonitoodete turu kasvu kohta:

- 2001 1,265 miljardit dollarit;
- 2002 1,285 miljardit dollarit;
- 2003 1,4 miljardit dollarit;
- 2008 3,1 miljardit dollarit *(WinterGreen Research, 2003)*.

EAI toodete turuliidrite hulka kuuluvad: *BEA Systems, CrossWorlds Software, IONA Technologies, Level 8 Systems, Mercator Software, NEON, SeeBeyond, Software AG, TIBCO, Vitria Technology ja WebMethods (Gormly,2001).*

2 INTEGRATSIOONI PÕHIMÕTTED JA ERINEVAD VORMID

Käesolevas ülevaates toon välja selleala ekspertide poolt käsitletud integratsiooni tüübid ja vormid, erinevad mehhanismid ning integratsiooniga seotud mõisted. Peatüki lõpus annan ülevaate ka aktuaalsetest integratsioonitööde juhtimise probleemidest ja soovitused nende lahendamiseks.

Põhimaterjalina olen kasutanud järgmisi raamatuid:

- *Next generation application integration: from simple information to Web services (Linthicum, 2003)*.
- *Enterprise integration patterns: designing, building, and deploying messaging solutions (Hohpe, 2003)*.
- *Microsoft Integration Patterns: patterns & practices (Trowbridge, 2004)*.

Raamatute autorid on oma ala tunnustatud spetsialistid ja omavad ka praktilisi kogemusi integratsiooni vallast.

David S. Linthicum on *SAGA Software Inc.* peaarhitekt ja pikaajalise kogemusega rakenduste integratsiooni ja e-äri ekspert. David Trowbridge on *Microsoft*'i platvormi arhitektuuri grupi arhitekt ja mitmete integrtsioonilahenduste disainer. Gregor Hohpe juhib integratsioonialast kompetentsi firmas *ThoughtWorks* ja on oma ala tunnustatud lektor IT alastel konverentsidel.

Et mõista paremini integratsiooni toimimist ja edaspidiste punktide all kirjeldatud integratsioonilahendusi, pean vajalikuks tutvustada selliseid mõisteid, nagu lõdvalt seotud- vs. tihedalt seotud rakendused ning sünkroonsus ja asünkroonsus.

Lõdvalt seotud vs tihedalt seotud rakendused (Loosely coupled vs tightly coupled applications). Rakenduste lõdvalt sidumine tähendab rakenduste liitmist viisil, kus nad ei ole teineteisest sõltuvad. Tihedalt sidumine on selle vastandiks, ehk siis tihedalt seotuna on rakendused üksteisest sõltuvad. Lõdvalt seotud rakendused ei pea ideaaljuhul teadma süsteemi kohta enne kasutamist muud lisainfot, peale selle asukoha. Süsteemi funktsioonid ja programmid on ideaaljuhul iseseisvad ja seetõttu sõltumatult väljavahetatavad, nad ei nõua muude komponentide väljavahetamist koos nendega. Rakenduste tihedalt sidumist on seevastu tihtipeale teostuse poole pealt lihtsam realiseerida ja see pakub ka suuremat jõudlust.

Rakenduste lõdvalt sidumine on integratsiooni vaatepunktist kahtlemata eelistatum lähenemine, kuid rakenduste tihedam sidumine annab sageli suurema jõudluse.

Sünkroonsus ja asünkroonsus *(Syncronous/ asynkronous)*. Integratsioonis kasutatakse asünkroonset ja sünkroonset kommunikatsiooni mehhanismi. Asünkroonses liitmises liigutatakse infot ühe või mitme rakenduse vahel

asünkroonsel moel – st. rakendusi siduv liides võib ennast lahtisidestada saatja- või sihtkoha rakendusest. Rakendused ei ole sõltuvad teistest ühendunud rakendustest. Rakendused võivad oma sõnumid panna järjekorda ja tegeleda samal ajal teiste ülesannetega, nad ei pea saama kohe tagasisidet.
Peamine asünkroonse mudeli eelis on, et see ei blokeeri rakenduse töötlusprotsesse.

Kontrastiks on sünkroonne kommunikatsioon tihedalt seotud rakendustega. Rakendused on üksteisest sõltuvad, ühe või mitme funtksiooni väljakutse töötlemisel. Selle tulemusena peab väljakutsuv rakendus seiskama töötluse, et oodata teise rakenduse vastust. Seda tüüpi kommunikatsiooni on nimetatud ka "blokeerivaks" *(Linthicum, 2003)*.

Kuna rakendus on sõltuv nendevahelisest liidesest, siis probleemid nagu võrguühendus või kaugelasuva serveri mahakukkumine, peatavad rakenduse töötlemise. Sünkroonne moodus vähendab ka võrgu läbilaskevõimet, sest mitmed väljakutsed peab tegema üle võrgu, toetades sünkroonset funktsiooni väljakutset.

Selles valguses tundub asünkroonne mudel sünkroonsest selgelt parem, kuid loomulikult sõltub valik eelkõige nõudmisetest ja tehnilistest võimalustest.

Integratsioonitüüpide klassifitseerimiseks nagu ka rakenduste integratsiooni teostamiseks on mitmeid mooduseid. Ühe konkreetse integratsiooniprobleemi lahendamiseks võib kasutada mitut integratsiooni tüüpi ja vormi ning mitmeid tehnoloogiaid. Keerukaks võib osutuda nendest õigete valikute tegemine ja sobiva konfiguratsiooni leidmine lahenduste vahel.

Integratsiooni kirjeldamiseks ja liigitamiseks on selleala spetsialistid kasutanud mustreid või teisiti öeldes skeeme *(Patterns).* Mustri all mõistetakse väljakujunenud standardlahendust, mis kirjeldab rakenduste integreerimise moodust.

Antud peatüki all püüan kokku tuua eelpool mainitud kolme autori poolt kirjeldatud integratsiooni võimalused ja mustrid koos võimalike tugevate ja nõrkade külgedega. Vaatluse alla tulevad intergratsiooni tüübid ja vormid on klassifitseeritud:

- Integratsiooni eesmärgi järgi;
- Interaktsiooni mehhanismi järgi;
- Ühenduse mehhaanika järgi;
- Integratsiooni topoloogia järgi.

Integratsiooni eesmärgi, interaktsiooni mehhanismi ja ühendamise mehaanika järgi klassifitseerimist on kasutanud Gregor Hohpe, *(Hohpe, 2003)*. Ühenduse mehaanikast lähtuvaid järgnevaid integratsiooni mustreid on lisanud D. Trowbridge ja kaasautorid *(Trowbridge, 2004)*. Selles raamatus on toodud välja ka mustrite omavahelised seosed, mis annab integrtasiooni mustritest terviklikuma pildi.

Järgnevate punktide all vaatlen juba konkreetseid integratsiooni võimalusi erinevate tüüpide, vormide ja mustrite näol.

2.1 Integratsiooni eesmärk

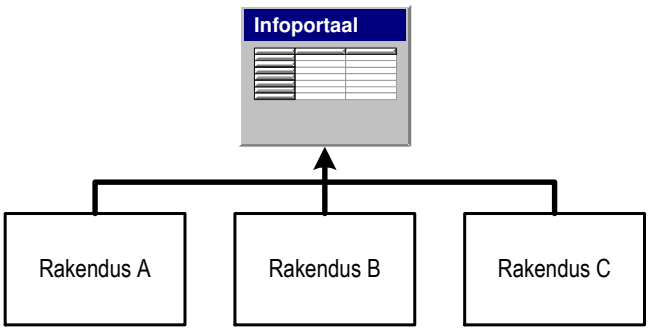
Integratsiooni eesmärgiks on põhiline probleem, mida integratsiooni abil üritatakse lahendada. See võib olla püüe anda kõikidele rakendusetele juurdepääs vajalikele andmetele, anda kasutajatele vaate mitmetest süsteemidest ühel ekraanil või mingi funktsiooni korduvkasutus mitmete rakenduste vahel.

Eesmärgi järgi integratsiooni liigitades ei tehta otseselt ettekirjutusi, milliseid tehnoloogiaid või mehhanisme peaks kasutama integratsiooni teostusel. Võib liigutada faile, käivitada kaugprotseduure, või saata hoopis sõnumeid mööda sõnumi kanalit.

Integratsiooni eesmärke võib olla palju, Hohpe on liigitanud integratsiooni eesmärgi järgi järgmiselt:

- Informatsiooni portaalid *(Information portals)*;
- Andmete replikatsioon *(Data replication)*;
- Jagatud äri funktsioonid *(Shared business functions)*;
- Teenusele orienteeritud arhitektuurid *(Service-oriented architectures)*;
- Hajus äriprotsessid *(Distributed business processes)*;
- Ärilt-ärile suunatud integratsioon *(Business-to-business integration)*.

(Hohpe, 2003).

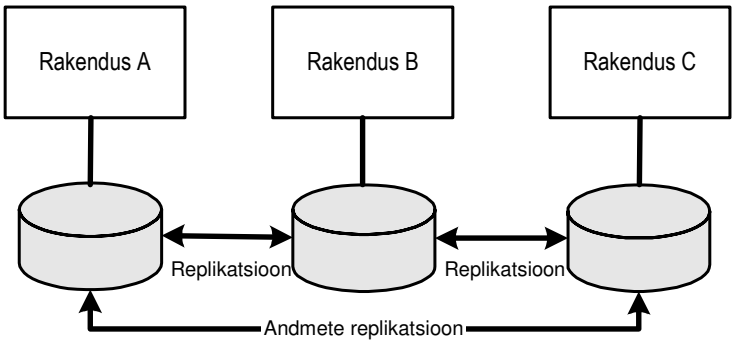


Joonis 1 Informatsiooni portaal

Informatsiooni portaalid *(Information portals).* Ettevõtte infoportaal *(Enterprise Information Portal, EIP)* on rakenduste klass, mis võimaldab kasutajale ühes kohas serveerida mitmetest allikatest pärit vajalikku infot, eesmärgiga toetada äriotsuste tegemist.

Paljud ärikasutajad peavad ligipääsema mitmetele süsteemidele, et vastata spetsiifilistele küsimustele või teostada mõnda ärilist funktsiooni. Näiteks, et kontrollida tellimuse staatust, peab klienditeenindaja omama ligipääsu tellimustehalduse süsteemile, siis veel logima sisse süsteemi, mis haldab veebist sisseantud tellimusi. Informatsiooni portaalid koondavad ühte vaatesse infot mitmetest allikatest, et kasutaja ei peaks seda otsima kokku erinevatest süsteemidest. Lihtsad infoportaalid jaotavad ekraani mitmeteks tsoonideks, igas tsoonis näidatakse infot erinevast süsteemist. Keerukamad infoportaalid pakuvad limiteeritud interaktsioone eri tsoonide vahel; näiteks kui kasutaja selekteerib

mingi elemendi tsooni A nimekirjast, siis tsoonis B värskendatakse andmed ja näidatakse detailset informatsiooni valitud elemendist *(Hohpe, 2003)*.



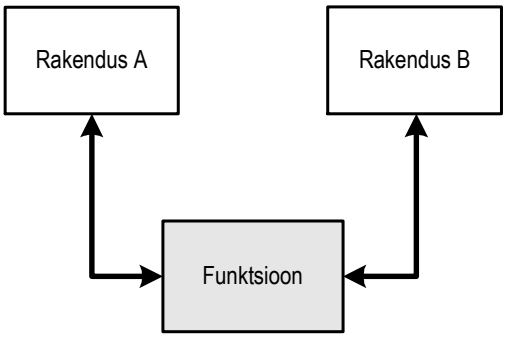
Joonis 2 Andmete replikatsioon

Andmete replikatsioon *(Data replication)*. Andmete replikatsioon on andmete liigutamine kahe või rohkema andmebaasi vahel *(Lithicum, 2003), vt*. Joonis 2. Lisaks andmete liigutamisele, tuleb neid andmeid ka kogu aeg sünkroonsetena hoida.

Paljud ärisüsteemid nõuavad ligipääsu samadele andmetele. Näiteks, kliendi aadressi võib hoida kliendihaldussüsteemis, raamatupidamissüsteemis, transpordisüsteemis või veel kuskil. Paljud nendest süsteemidest omavad enda andmehoidlat, kuhu salvestatakse kliendiga seotud informatsioon. Kui kliendilt peaks laekuma info, et muutunud on tema aadress, siis kõik need süsteemid peavad uuendama oma andmeid selle kliendi kohta. Kui teostatud integratsioon baseerub andmete replikeerimisel on oht, et tekib andmeliiasus *(Hohpe, 2003)*.

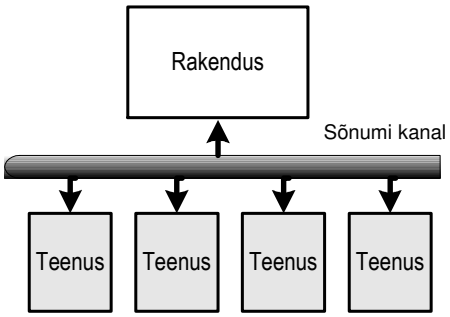
Andmete replikatsiooni teostuseks on mitu erinevat võimalust. Näiteks, osad andmebaasi pakkujad ehitavad replikatsiooni funktsiooni andmebaasi sisse, alternatiivina võib veel eksportida andmed failidesse ja importida need teise süsteemi või kasutada sõnumile orienteeritud vahevara, et transportida andmed sõnumitena.

Kui teostatakse andmebaasi kopeerimist, siis igaljuhul tuleb siin arvestada asjaoluga, et erinevad andmebaasid on vältimatult pisut sünkroonist väljas ja seda tänu latentsusajale, mis tuleneb muudatuste üleviimisest ühest baasist teise baasi *(Trowbridge, 2004)*.



Joonis 3 Jaotatud ärifunktsioonid

Jaotatud ärifunktsioonid *(Shared business functions).* Kui andmete replitseerimine võib põhjustada andmeliiasust, siis samad ohud kehtivad ka funktsionaalsuse puhul. Mitmed süsteemid peavad kontrollima, kas näiteks isikukood on õige, kas aadress või postiindeks sobib või kas valitud ese on laos olemas. Mõistlik on neid funktsioone eksponeerida, kui jaotatud ärifunktsioone, mis on korra teostatud ja saadaval teenusena ka teistele süsteemidele kasutamiseks *(Hohpe, 2003).*



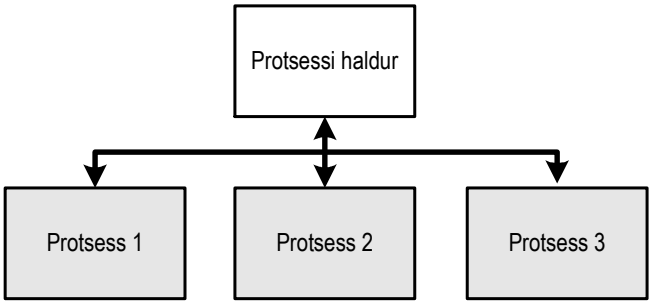
Joonis 4 Teenusele orienteeritud arhitektuur

Teenusele orienteeritud arhitektuur *(Service-oriented architecture).* Teenus on kindlapiiriine funktsioon, mis on universaalselt saadaval ja vastab teenuse tarbija päringutele.

Kui ettevõttele koguneb mingi hulk kasutatavaid teenuseid, muutub kriitiliseks funktsiooniks nende teenuste haldamine. Esiteks, rakendused vajavad mingis vormis teenuste kataloogi, tsentraliseeritud nimekirja saadaolevatest teenustest. Teiseks, iga teenus peab kirjeldama oma liidese sellisel viisil, et rakendus saab selle alusel kommunikeeruda teenusega. Need kaks funktsiooni, teenuse leidmine ja üleantud kirjeldus on võtmeelemendid, mis on aluseks teenusele orienteeritud arhitektuurile.

Kui jätta kõrvale jõudlus, siis teenusele orienteeritud arhitektuur pakub rakendustele vahendid, mis teevad välise teenuse väljakutsumise peaaegu sama lihtsaks kui lokaalse meetodi väljakutsumise *(Hohpe, 2003)*.

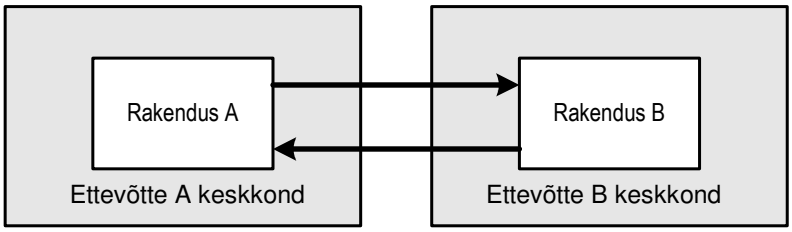
Teenusele orienteeritud integratsioonist on lähemalt juttu veel funktsionaalse integratsiooni all punktis 2.3.2.



Joonis 5 Hajus äriprotsessid

Hajus äriprotsessid *(Distributed business processes).* Üks integratsiooni võtmekäitajatest on asjaolu, et üksik äri transaktsioon on tihti laiali üle mitmete süsteemide. Enamik juhtudel on kõik relevantsed funktsioonid koondatud olemasolevate rakenduste sisse. Puudu on aga koordinatsioon nende rakenduste vahel. Selle olukorra parandamiseks saab lisada äriprotsessi juhtimise komponendi, mis haldab ärifunktsioonide ristkasutamist olemasolevate süsteemide vahel.

Piir teenusele orineteeritud integratsiooni ja hajus äriprotsesside vahel on üsna udune. Näiteks võib eksponeerida kõik relevantsed ärifunktsioonid kui teenused ja siis kodeerida äriprotsessid rakendusse, mis omab ligipääsu kõikidele teenustele üle teenusele orienteeritud integratsioonilahenduse.



Joonis 6 Ärilt-ärile suunatud integratsioon

Ärilt-ärile suunatud integratsioon *(Business-to-business integration).* Siiamaani oleme vaatlenud interaktsiooni rakenduste ja ärifunktsioonide vahel ühe organisatsiooni sees. Paljudel juhtudel on ärifunktsioonid saadaval välistele tarnijatele ja partneritele. Näiteks transpordifirma võib pakkuda oma klientidele teenust saadetiste maksumuse kalkuleerimiseks ja edaspidi saadetiste jälgimiseks. Nendel juhtudel integratsioon toimub äripartnerite vahel. Sedasorti lahendused kasutavad kommunikatsiooniks tihti internetti, või muud laivõrku, mis tõstatab üles sellised teemad nagu transpordi protokoll ja turvalisus.

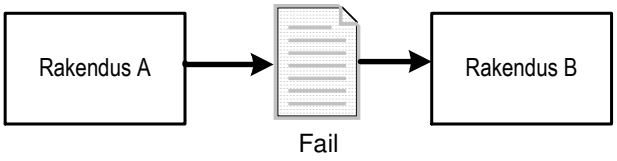
Resümeerides võib öelda, et igal tüübil on oma määratlus ja fokusseeritud probleem, mida integratsioon lahendada püüab. Integratsioonilahendus konkreetse eesmärgi saavutamiseks võib kujutada ka kombinatsiooni neist väljapakutud tüüpidest.

2.2 Interaktsiooni mehaanika

Integratsiooni tüübid lähtudes interaktsiooni mehaanikast süsteemide vahel on Hohpe poolt liigitatud järgmiselt:

- Faili ülekanne *(Fail transfer)*;
- Ühisandmebaas *(Shared database)*;
- Kaugprotseduuride käivitamine *(Remote procedures invocation)*;
- Sõnumivahetus *(Messaging)*.

Iga tüüp defineerib viisi, kuidas viia infot punktist A punkti B. Iga tüüp on erinev ja neil on oma nõudmised ja eelised. Interaktsiooni mehaanika tüübi otsus on rohkem üldise arhitektuuri otsus. Seda klassifikatsiooni võib nimetada ka arhitektuuriliseks stiiliks, mis tuleb otsustada varakult ja mis mõjutab ka seda, kuidas teostatakse kogu süsteemi arhitektuur *(Hohpe, 2003)*.



Joonis 7 Faili ülekanne

Faili ülekanne *(Fail transfer).* Selle mustri puhul produtseerib iga rakendus faile jaotatud andmetest teistele tarbimiseks ja tarbib ise faile, mis on teiste rakenduste poolt produtseeritud. Failid on universaalne salvestuse mehhanism, mis on ehitatud iga ettevõtte toimivasse süsteemi. Failide kasutamine on kõige lihtsam lähenemisviis, et kuidagi rakendused integreerida.

Oluline otsus failide juures on, millist formaati kasutada. Väga harva juhtub nii, et ühe rakenduse väljund on samas formaadis, mida vastuvõttev rakendus kohe tarbib. Reeglina tuleb seal vahel siiski mingit töötlust teha failidega.

Aja jooksul on välja kasvanud ka standardsed failiformaadid. Suurarvutiga süsteemid kasutavad põhiliselt andme-söötmisel baseeruvat *COBOL (Common Business Oriented Language)* failisüsteemi formaati. *UNIX* süsteemid kasutavad tekstil baseeruvaid faile. Põhiliselt käibelolev meetod on *XML (Extensible Markup Language)*.

Teine teema on failide produtseerimine ja tarbimine. Kui infot on palju, siis võib see üsnagi töömahukas protsess olla.

Failide eelisteks on see, et integraatorid ei pea midagi teadma rakenduse siseelust. Lõppkokkuvõttes pole rakendused omavahel seotud ja võivad vabalt lubada muudatusi enda sees, ilma, et nad häiriksid teiste rakenduste elu ja pakuvad endiselt samas formaadis faile samade andmetega.

Faili ülekandmise tehnoloogia teeb lihtsaks asjaolu, et ei vajata spetsiaalseid vahendeid või integratsiooni komplekti, kuid see tähendab ka seda, et arendajad peavad tegema palju tööd mõlema liidestatava osapoole vahel. Rakendused peavad olema teadlikud failinimede kokkulepetes ja kataloogides, kus neid

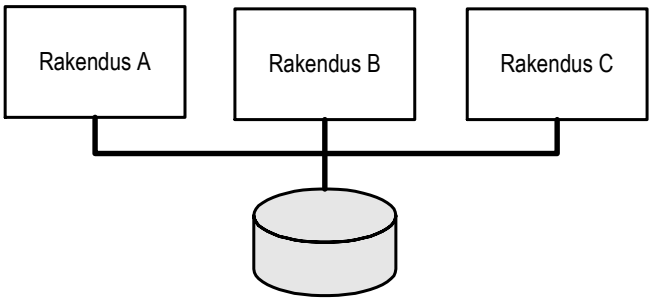
hoitakse. Faili kirjutaja peab teostama strateegia, kuidas hoida faili nimed unikaalsetena. Rakendused peavad kokkuleppima, milline neist kustutab vanad failid ja toimingu eest vastutav rakendus peab teadma, millal fail on vana ja mittevajalik. Rakendused peavad teostama veel lukustusmehhanismi või jälgima ajalist kokkulepet kindlustamaks, et üks rakendus ei ürita lugeda faili kui teine samal ajal alles koostab seda.

Üks probleeme faili ülekandmisel võib veel tekkida kui failiuuendused toimuvad harva, siis pole süsteemid omavahel enam sünkroonis. Kliendijuhtimis-süsteemist viiakse sisse näiteks muudatus kliendi aadressis, andmete uuendamine aga toimub igal õhtul. Samal ajal saadetakse arvete süsteemist teele arve, kus on peal vana aadress.

Faili ülekandmine lubab rakendustel küll jagada andmeid, kuid probleemiks saab siin andmete ülekandmise õigeaegsus, mis on integratsioonis üsna kriitiline tegur. Kaasaegsetes süsteemides on tihti oluline, et andmed oleksid tõepoolest viimased. Mida tihedamini andmeid uuendatakse, seda vähem esineb seal ka ebaühtlusi.

Faili ülekandmine ei sunni peale ka piisavalt andmeformaati. Paljud integratsiooni probleemid tulenevad andmete tähenduse väärast mõistmisest, tekib semantiline dissonants.

Et teha andmed kiiremini kättesaadavaks ja nõuda kokkulepitud andmeformaate on mõistlik kasutada ühisandmebaasi. Kui on vaja integreerida rohkem rakenduste funktsionaalsust kui nende andmeid, on sobiv kasutada kaugprotseduuride käivitamist. Väikse mahuga sagedaseks andmevahetuseks sobib sõnumivahetus *(Hohpe, 2003)*.



Joonis 8 Ühisandmebaas

Ühisandmebaas *(Shared database)*. Selle lahenduse puhul hoiavad rakendused oma andmeid, mida nad soovivad jagada ühtses andmebaasis. Kui integreeritud rakendused kasutavad kõik ühisandmebaasi, võib olla üsna kindel, et andmed on ühtlustatud kogu aeg. Erinevatest allikatest tulevate uuenduste üheaegseks tegemiseks ühele ja samale andmeosale on olemas transaktsiooni haldussüsteem, mis tegeleb nende probleemidega väga edukalt. Kuna uuendustevaheline aeg on väga väike, siis võimalikke tekkivaid vigu on tunduvalt

lihtsam leida ja parandada. Kuna iga rakendus kasutab sama andmebaasi, siis pole ka andmete dissonantsi probleemi.

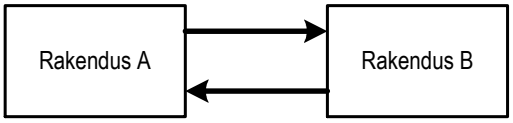
Üks suurimaid keerukusi ühisandmebaasi juures on talle sobiva disaini tegemine. Ühtse andmebaasi struktuuri kirjelduse *(schema)* tegemine, mis vastab erinevate rakenduste vajadustele on keeruline ülesanne. Enamasti lõpeb see skeemiga, millega rakenduste programmeerijatel on raske töötada.

Üks raskendav piirang ühisandmebaasi juures on välised pakitooted. Enamik pakitarkvarasid ei tööta ühegi teise andmebaasi skeemiga, väljaarvatud nende enda oma. Isegi kui seal on mingit ruumi kohandusteks, on tõenäoline, et see on rohkem limiteeritud kui integreerijad seda tahaks. See probleem laieneb integratsioonile ka peale arendusprotsessi. Isegi, kui õnnestub tööle organiseerida kõik rakendused, tekib integratsiooniprobleem tulevikus kui näiteks organisatsioonid ühinevad.

Kui kõik rakendused kasutavad ühisandmebaasi sagedasti samade andmete lugemiseks ja muutmiseks, siis võib andmebaas saada jõudluse pudelikaelaks põhjustades *deadlock'e*, kus rakendused takistavad teineteisel andmetele ligipääsu.

Kui rakendused paiknevad eri asukohtades ja omavad ligipääsu jagatud andmebaasile üle laivõrgu, on see praktikas liiga aeglane. Jagades andmebaasi laiali hajusandmebaasiks, saab ligipääsu korraldada üle lokaalse võrguühenduse. See tekitab jälle küsimuse, millisesse osasse peaks andmed salvestama, kuna võimalusi on nüüd mitu. Hajusandmebaas koos lukustuskonfliktidega võib saada kergesti jõudlusele saatuslikuks.

Faili ülekanne võimaldab hoida rakendused kenasti lahus, aga seda viivise hinnaga – koostöö on liiga aeglane. Ühisandmebaas hoiab andmed koos kiiresti reageerival moel, kuid seda tingimusel, kus kõik rakendused on ühildatud ühe andmebaasiga. Kui on soov integreerida pigem rakenduste funktsionaalsust kui andmeid, sobib kasutada kaugprotseduuride käivitamist. Kui on vaja vahetada tihedalt väiksemahulisi andmeid, mille formaat on andmetüübi järgi, mitte ühe universaalse skeemi järgi, siis sobib rohkem sõnumivahetus *(Hohpe, 2003)*.



Joonis 9 Kaugprotseduuride käivitus

Kaugprotseduuride käivitus *(Remote procedure invocation).* Nagu nimigi ütleb, võimaldavad kaugprotseduurid käivitada mingit tegevust ja vahetada andmeid, mida vastav kaugrakendus on välja eksponeerinud.

Faili ülekanne ja ühisandmebaas võimaldavad rakendustel jagada nende andmeid, mis on rakenduste integratsioonis oluline osa, kuid ainult andmete jagamine pole tihti piisav. Muudatused andmetes nõuavad sageli ka muid toiminguid erinevates rakendustes. Näiteks, aadressi muudatus võib olla lihtne muudatus andmetes, kuid see võib käivitada veel registreerimise ja õiguslikud

protsessid, et viia kliendi kontole selle põhjal sisse erinevad reeglid. Et üks rakendus saaks käivitada otse selliseid protsesse teisetes rakenduses, peab ta päris palju teada teise rakenduse siseehitusest.

Üks võimsamaid struktureerimismehhaniseme rakenduste disainis on kapseldamine *(encapsulation)*, ehk andmetüübi siseehitust varjav liidestamine. Sellisel moel ei pea üks rakendus teadma teise, liidestatava siseehitusest ja saab käivitada vajalikud protseduurid. Ühisandmebaas pakub laiaulatusliku mittekapseldatud andmestruktuuri, mis teeb palju raskemaks selle toimingu. Failiülekanne lubab rakendusel küll reageerida muudatustele kui see töötleb faili, kuid protsess ise on viivisega.

Fakt, et ühisandmebaasis on mittekapseldunud andmed teeb keerulisemaks ka integreeritud rakenduste haldamise. Paljud muudatused rakendustes võivad käivitada muudatuse andmebaasis ja nendel muudatustel on omakorda märkmisväärne mõju teistele rakendustele. Selle tulemusena on ühisandmebaasi kasutavad organisatsioonid tihti väga tõrksad oma andmebaasi muutmise suhtes, millest tulenevalt ei pruugi rakenduste edasine arendus vastata muutuvatele ärinõudmisele.

Vaja on mehhanismi, mis võimaldab ühel rakendusel käivitada teise rakenduse funktsioone, et läbi lasta vajalikud andmed ja käivitada funktsioon, mis ütleb vastuvõtja rakendusele, kuidas töödelda andmeid.

Rakenduse arendamine kapseldatud andmetega objektiks või komponendiks pakub liidese, mis lubab teistel rakendustel vastastikku toimida töötava rakendusega.

Kaugprotseduuri käivitus rakendab kapselduse printsiipi integreeritud rakendustele. Kui rakendus vajab infot, mida omab teine rakendus, siis ta küsib seda rakenduselt otse. Kui üks rakendus vajab teise rakenduse andmete muutmist, siis ta teeb seda saates kutse teisele rakendusele. See võimaldab igal rakendusel säilitada oma andmete terviklikkust. Lisaks võib iga rakendus muuta oma sisemiste andmete formaati, ilma, et see puudutaks teisi rakendusi.

Hulk tehnoloogiaid, nagu *CORBA, COM*, *.NET Remoting*, *Java RMI* võimaldavad teostada kaugprotseduuride väljakutset (tuntud ka nime all *Remote Procedure Call, RPC)*. Need käsitlused varieeruvad, sõltuvalt kui palju süsteem neid toetab ja kasutusmugavuse poolest.

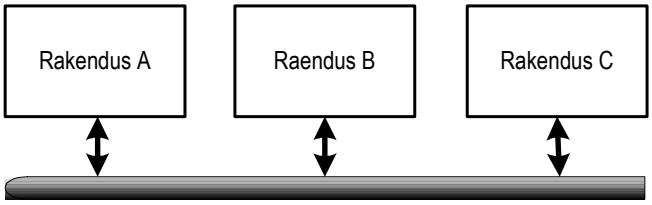
Tavaliselt sellised keskkonnad pakuvad lisavõimalusi, nagu transaktsioonid. Praeguseks favoriidiks on siin veebiteenused, mis kasutavad standardeid nagu *SOAP* ja *XML*. Veebiteenustel on veel väärtuslik omadus lihtsalt töötada *HTTP* protokolliga, mis lihtsustab läbipääsu tulemüüridest.

Siiski on päris suur jõudluse ja töökindluse vahe lokaalse protseduuride väljakutse ja kaugprotseduuride väljakutse vahel. Kaugprotseduuride väljakutse võib siin jääda aeglaseks ja vähem töökindlaks.

Ehkki kapseldamine aitab vähendada rakenduste liitmise probleemistikku, elimineerides suured hajusandmestruktuurid, on siiski rakendused tihedalt seotud üksteisega. Kaugkutsed soodustavad erinevate sõlmpunktide hulga suurenemist,

mida iga süsteem peab ülal erineva süsteemiga liidestumiseks. See teeb keeruliseks muuta süsteemid sõltumatuks.

Et integreerida rakendusi vähem sõltuvatena, lõdvemalt ja asünkroonselt, sobiks kasutada sõnumivahetust *(Hohpe, 2003).*



Joonis 10 Sõnumivahetus

Sõnumivahetus (*Messaging)* Sõnumivahetus võimaldab igal rakendusel kommunikeeruda ühisesse sõnumisüsteemi, vahetada andmeid ja käivitada tegevust, kasutades selleks sõnumeid. Asünkroonne sõnumisaatmine ei nõua mõlemalt süsteemilt valmisolekut töötlemiseks samal ajal. Kuna töötamine hajusrakendustega on aeglasem, siis arendajad disainivad komponente, kus suur osa töötlusest tehakse lokaalselt ja ainult valitud protsessid kaugereziimis. Sõnumivahetus pakub ka rakenduste lahtisidestatust, e. lõdvalt sidumist, mis on omane ka faili ülekandmisel. Sõnumeid saab transformeerida üleviimisel, ilma, et saatja või vastuvõtja üldse teaksid midagi andmete teisendamisest. Lahtisidestatus võimaldab integraatoritel levitada sõnumeid ka mitmete vastuvõtjate vahel, marsruutida sõnumeid ühele või mitmele vastuvõtjale või hoopis teisele topoloogiale. See võimaldab eraldada integratsioonialased otsused rakenduste arendusest.

Kui kasutatakse sõnumi transformeerimist, siis eraldi olevad rakendused võivad omada täiesti erinevaid konseptuaalseid mudeleid. Muidugi võib juhtuda ka siis semantilist dissonantsi. Hohpe väitel pole hajusandmebaasi juures semantilise dissonantsi ärahoidmiseks kasutatavad mõõdikud sõnumivahetuse korral praktikas kasutatavad.

Väikeste sõnumite sage saatmine võimaldab rakendustel teha koostööd ka andmete jagamisel. Kui mingi protsess vajab näiteks käivitumiseks kindlustusnõude vastuvõttu, siis seda saab teha koheselt, saates sõnumi kui kindlustusnõue tuleb sisse. Informatsiooni saab pärida ja päringutele vastata kiiresti. Kaugprotseduuride kutse ei tööta näiteks nii kiiresti. Andmete pärija peab ootama kuni sõnum on töödeldud ja vastus tagasitulnud.

Asünkroonses tehnoloogias on palju erinevaid reegleid ja tehnoloogiaid, millega paljud programmerijad ei pruugi kursis olla. Seega vajab selle teostus mingit õppimisaega. Samuti on asünkroonse keskkona testimine ja silumine raskem.

Sõnumite transormeerimisega kaasneb see hüve, et rakendusi võib üksteisest piisavalt lahtisidestatuna hoida. Andmeid saadetakse sõnumitena mööda sõnumikanalit, mis on saatja ja vastuvõtja vahel. Kui saatja ei tea, kuhu adresseerida andmed, siis võib andmed saata marsruuteri moodulisse, mis suunab sõnumid sobivale vastuvõtjale. Kui saatja ja vastuvõtja ei nõustu sõnumi

formaadiga, siis saatja saab suunata selle otse "sõnumi tõlkija" moodulisse, mis konverdib andmed vastuvõtjale sobivasse formaati ja saadab vastuvõtjale *(Hohpe, 2003).*

2.3 Ühenduse mehhanism

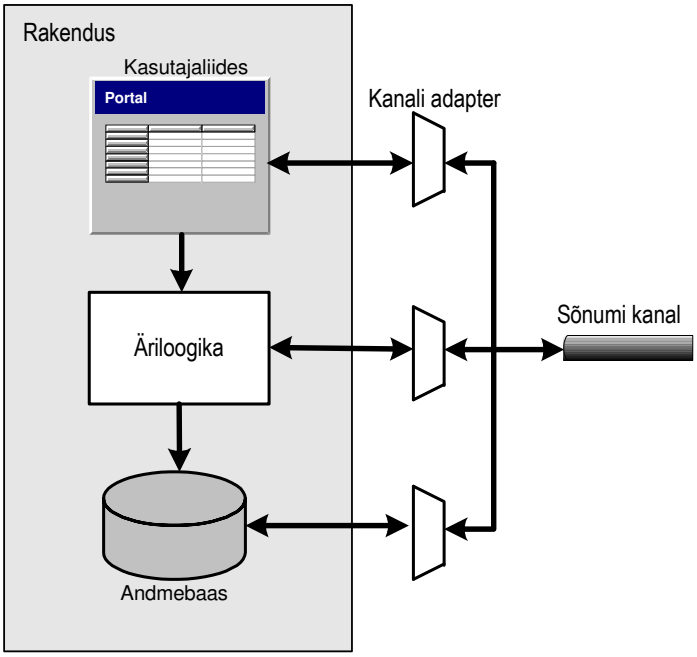
Järgnevalt vaatlen integratsiooni mustreid, mis on liigitatud ühenduse mehhanismi järgi.

Enamik hästitehtud ärirakendusi omab presentatsiooni kihti, äriloogika kihti ja andmebaasikihti. Kõiki neid kihte võib kasutada andmete liigutamiseks rakenduse ja integratsiooni kanali vahel. Ühenduse mehhanismi järgi on G. Hohpe poolt välja toodud järgmised integratsiooni tüübid:

- Presentatsiooni integratsioon *(Presentation integration)*;
- Funktsionaalne integratsioon *(Functional integration)*;
- Andmete integratsioon *(Data integration)*.

(Hohpe, 2003).

Kuidas ühendutakse rakendusega on sageli üsna lokaalne otsus. Kaks asja on siin olulised, millele tuleb tähelepanu pöörata. Kui me võtame vastu sõnumi (eeldades, et momendil kasutame sõnumil baseeruvat integratsiooni), siis on suuresplaanis sekundaarse tähtsusega, kas see sõnum pistetakse andmebaasi (andmebaasi integratsioon) või pöördutakse rakendusliidese *(API)* poole või siis sisestatakse andmed otse ekraanile (presentatsiooni integratsioon), vt. Joonis 11.

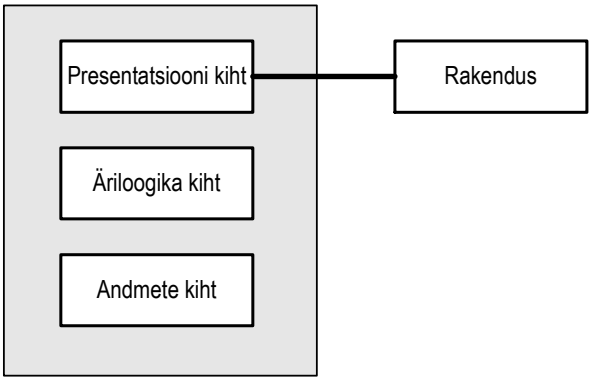


Joonis 11 Integratsioon rakenduse eri kihtidesse

Millist mehhanismi kasutada, on suuresti sõltuv integratsiooni eesmärgist. Kindlasti saab teostada informatsiooni portaali, kasutades andmete integratsiooni, funktsionaalset integratsiooni või presentatsiooni integratsiooni või kasutada kõiki kolme varianti *(Hohpe, 2003)*.

G. Hohpe poolt pakutud ühenduse mehhanismi liigitust on täiendanud veelgi D. Trowbridge ja kaasautorid *(Trowbridge, 2004)*. Ühtsesse skeemi on lahti jagatud funktsionaalne integratsioon, andmete integratsioon ja integratsiooni topoloogiad, mida vaatleme lähemalt järgnevate jaotiste all.

2.3.1 Presentatsiooni integratsioon

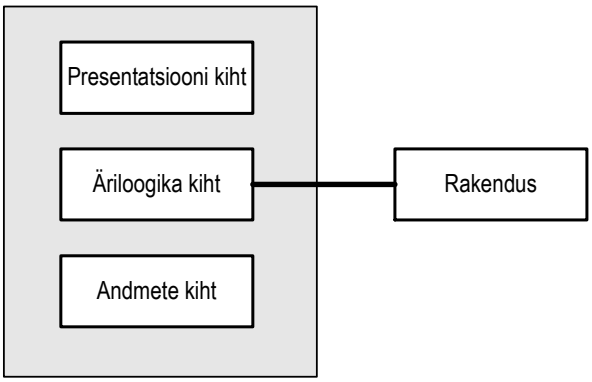


Joonis 12 Presentatsiooni integratsioon

Presentatsiooni integratsiooni korral toimub liidestamine rakenduse presentatsiooni kihti, vt. Joonis 12. Presentatsiooni ühendust võib käsitleda kui sissetungivat moodust rakenduste integratsioonis, kuna ühenduv rakendus jääb peremeesrakenduseks, nagu inimene, kes annab arvutile käske läbi kasutajaliidese. Läbi presentatsiooni kihi ühendudes saab ligipääsu samale funktsionaalsusele, mis on saadaval ka inimkasutajale. Andmed ja funktsioonid pole eksponeeritud piisavalt detailselt, jättes varju võimalusterohkema rakendusliidese *(Application Programming Interface, API)* kasutamise.

Presentatsiooni integratsiooni eeliseks on selle teostamise odavus *(Trowbridge, 2004)*.

2.3.2 Funktsionaalne integratsioon



Joonis 13 Funktsionaalne integratsioon

Funktsionalse integratsiooni korral toimub liidestamine rakenduse loogika kihti, vt. Joonis 13. Funktsionaalne integratsioon võimaldab rakendustel ja teenustel kasutada äriloogikat, mis on teistesse süsteemidesse kapseldatud.

Funktsionaalne integratsioon ühendab rakendused läbi liidese ja spetsifikatsiooni. Kahjuks, mitte kõik rakendused integratsiooni arhitektuuris ei oma liideseid ja spetsifikatsioone. Tihti esineb olukordi, kus rakendused, mis omavad rakendusliidest *(API),* ei eksponeeri oma andmeid ja funktsioone integratsioonile vajalikul detailsel tasemel. Funktsionaalse integratsiooni puhul on valida kolme alternatiivi vahel:

- Hajusobjektide integratsioon *(Distributed object integration)*;
- Sõnumile orienteeritud vahevara integratsioon *(Message-oriented middleware integration)*;
- Teenusele orienteeritud integratsioon *(Service-oriented integration)*.

(Trowbridge, 2004).
Neid valikuid käsitlen ka eraldi järgmistes lõikudes.

Hajusobjektide integratsioon (Distributed object integration). Hajusobjektide integratsioon on tuntud ka kui instantsil baseeruv koostöö, sest ta laiendab objektorienteeritud mudelit hajutatud lahendustele. Rakenduste sees olevad objektid suhtlevad kaugrakendustes olevate objektidega samal viisil, nagu nad suhtleks lokaalselt teiste objektidega. Sellest johtuvalt interaktsioon toimub spetsiifiliste objekti instantsidega ja kliendirakendus haldab tavaliselt ojekti eluiga. Sedasorti koostöö paistab rakenduste arendajatele tavaliselt loomulik, aga võib tulemusena tekitada keerukaid ja tihedalt seotud interaktsiooni mudeleid komponentide vahel. Igaljuhul pole see kõige parem lahendus, kui on vaja integreerida paljusid eraldiseisvaid rakendusi.
Selline integratsioonilahendus töötab hästi kui süsteemid, mida tahetakse ühildada asuvad samas andmekeskuses, on igati töökindlad ja omavad kiiret

ühendust omavahel. Lahendus ei sobi aeglase ja mitteusaldadavate liidestega, kaasaarvatud internetis asuvad liidesed *(Trowbridge, 2004)*.

D. Lithicumi sõnul *(Linthicum, 2003)* on hajusobjektid väiksed rakendusprogrammid, mis kasutavad standardseid liideseid ja protokolle omavahel.

Turul on saadaval kahte tüüpi hajussobjekte: *CORBA* ja *Component Object Model (COM)*. *CORBA*, mis on loodudud *OMG* poolt 1991 on rohkem standard kui tehnoloogia. See pakub spetsifikatsiooni, mis liigendab reeglid, mida arendajad peavd jälgima *CORBA* toetusega hajusobjekti tehes. *CORBA* on heterogeenne ja sobib koos *CORBA*-t toetavate hajusobjektidega enamikele platvormidele.
COM Microsofti poolt propageeritud hajusobjektide standard. Nagu *CORBA*-gi pakub *COM* reegleid arendajatele, kes teostavad *COM*-toega jaosobjekte. Need reeglid sisaldavad liideste standardeid ja kommunikatsiooni protokolle. Kuigi on olemas *COM*-toega objekte ka teistele platvormidele peale Windowsi, tahetakse *COM*-I liita Windowsi keskkonda, mis teeb ta olemuselt homogeenseks *(Linthicum, 2003)*.

Sõnumile orienteeritud vahevara integratsioon *(Message-oriented middleware integration)*. Sõnumile orienteeritud vahevara integratsioon ühendab süsteeme, kasutades asünkroonset sõnumite ootejärjekorda (*message queue*), mis on osa sõnumile orienteeritud vahevarast. Ühilduvad süsteemid kasutavad kommunikatsiooniks sõnumeid, mis on jagatud väikesteks andmepakettideks. Kuna kommunikatsioon on asünkroonne ja püsiv, on olemas väike võimalus, et sõnum läheb kaotsi võrgu või süsteemi rikke tõttu.

Et jagada päringu/vastuse tüüpi funktsionaalsust peab tarbiv süsteem tegema päringu sõnumi ja saatma selle sõnumijärjekorda kasutades süsteemile, mis pakub vajalikku funktsionaalsust. Pakkuja võtab sõnumi järjekorrast, interpreteerib seda kui päringut ja asub vastavalt töötlema. Kui töötlus on lõpetatud, genereerib pakkuja vastuse sõnumi ja saadab selle tagasi läbi sõnumijärjekorra teenuse tarbijale.

Sõnumile orienteeritud vahevara valik on sobiv kui süsteemid ei oma usaldusväärset ühenduskanalit. Kuna sõnumile orienteeritud vahevara saadab sõnumeid asünkroonselt sõnumijärjekorda, on võimalus väike, et sealt midagi kaduma läheb süsteemi- või võrgu vea korral. Saatja ja vastuvõtja raekendus pole omavahel tihedalt seotud, kuna sõnumitel on üldkasutatav formaat. Kui sõnum on saadetud lähtekohast välja, võib kasutada edasi sõnumi jaoturi mustrit *(Message broker pattern)*, sõnumi siini mustrit *(Message bus pattern)*, publitseeri/aboneeri mustrit *(Publish/subscribe pattern)* või lihtsalt sõnumi vastuvõtja serverit, ilma muutmata originaalsõnumi saatjat.

Seda lahendust kasutades tuleb arvestada, et programmerimismudel on siin keerukam kui hajusobjektide integratsiooni puhul. Sõnumite saabumine sihtkohta kujuneb sündmuseks ja programmeerimismudel on ka sündmusepõhine. Kui need sündmused toimuvad üle võrgu, mis võib olla mittetöökindel, tuleb tegeleda

duplikaatsõnumitega, mis võivad põhjustada eraldi probleeme *(Trowbridge, 2004)*.

Teenusele orienteeritud integratsioon *(Service-oriented integration).* Teenusele orienteeritud integratsioon ühildab süsteeme läbi *XML*-il baseeruva veebiteenuste. Nende süsteemide liidesed on kirjeldatud läbi vastava standardi, milleks on *Web Services Definition Language (WDSL).* Süsteemid kasutavad omavaheliseks suhtluseks *SOAP* sõnumeid, mida toimetatakse edasi tavaliselt läbi *HTTP* protokolli, kasutades *XML*-i.

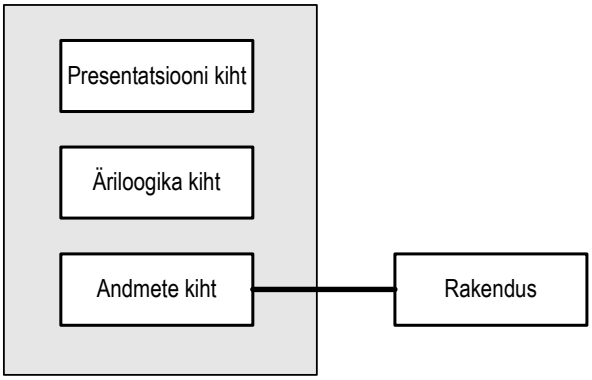
Teenusele orienteeritud integratsiooni kasutamine parandab kindlasti koostalitlusvõimet, kuna *XML* skeemid on kergesti porditavad erinevatele tehnilistele arhitektuuridele.

Kui oluline on paljude süsteemide koostalitlusvõime, siis on teenusele orienteeritud integratsioon sobiv lahendus. Kasutatavad standardid võimaldavad porditava süsteemi ja laiendatava raamistiku, mis ei ole tihedalt liidetud ühegi süsteemi külge.

Teenusele orienteeritud integratsioon võimaldab saata sõnumeid nii sünkroonsel kui asünkroonsel moel.

Nõrga küljena võib välja tuua jõudluse. XML dokumendid on paraku suuremad kui binaarsed ekvivalendid, kuna nad sisaldavad metaandmeid. See kasvatab sõnumite saatmisel andmetöötluskoormust, mida üritatakse korvata järjest võimsamate masinatega *(Trowbridge, 2004)*.

2.3.3 Andmete integratsioon



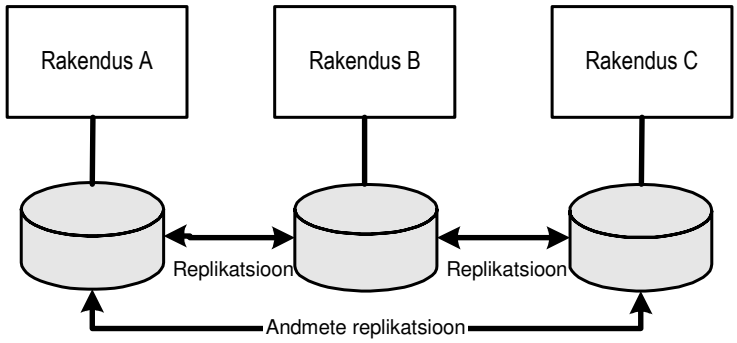
Joonis 14 Andmete integratsioon

Enamik rakendusi talletavad oma andmeid ühes kohas, nagu failides, realtsioonilistes-, hierarhilistes ja objekt-orienteeritud andmebaasides. Rakendused, mis vajavad seda informatsiooni võivad ühenduda otse nendele andmevarudele, vt Joonis 14. Kui on otsus integreerida andmekihti, siis võib valida kolme mustri vahel: faili ülekanne, ühisandmebaas või andmekoopiate ülalhoid.

Faili ülekanne *(Fail transfer).* Selle lahenduse puhul loetakse andmed andmebaasist välja ja salvestatakse faili, mida transporditakse rakenduste vahel. Lähemalt oli juttu faili ülekandest eespool punktis 2.2 interaktsiooni mehaanika all.

Ühisandmebaas *(Shared database).* Ühisandmebaasi korral tarbivad kõik integreeritud rakendused infot otse ühest andmebaasist. Lähemalt oli ühisandmebaasist ka juttu G. Hohpe liigituse all, punktis 2.2 interaktsiooni mehaanika.

Andmekoopiate ülalhoid *(Maintain data copies).* Selle mustri puhul tehakse koopiaid rakenduste andmebaasi, mida iga rakendus saab eraldi tarbida. Selle mustri juures tuleb arvestada andmete pikema latentsusajaga, mis kulub andmete uuendamiseks ühest baasist teise. Latentsusaeg oleneb muidugi sellest, kui tihti andmeid uuendatakse ühest baasist teise. Näitena võib vaadata andmete replikatsiooni vt Joonis 15, mis on andmekoopiate ülalhoiu derivaat.



Joonis 15 Andmete replikatsioon

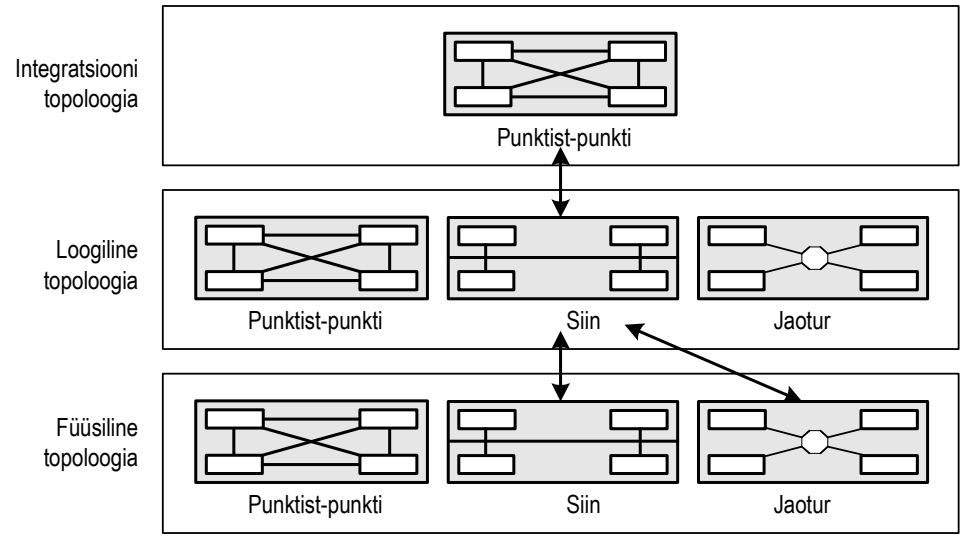
Lähemalt on andmete replikatsioonist kirjutatud eespool punkti all 2.1.
Andmete ülalhoiu mehhanismid võivad olla üsna komplekssed. Igaljuhul pakutakse *Microsofti* mustrite raamatus *(Trowbridge, 2004)* välja veel 12 alammustrit andmete liigutamiseks, mis kuuluvad kõik andmete ülalhoiu mustri alla. Nende kõigi käsitlemine pole autori arvates siinkohal siiski otstarbekas ja viiks magistritöö mahu liigselt suureks.

2.4 Integratsiooni topoloogia

Integratsiooni topoloogia on kontekst, mis kirjeldab integrtasiooni elementide asukohti, struktuure ja kanaleid *(Trowbridge, 2004)*.

Topoloogiad paiknevad erikihtidel ja jagunevad omakorda füüsiliseks, loogiliseks ja integratsiooni topoloogiaks. Füüsiline topoloogia kirjeldab, kuidas iga süsteemi sõlm füüsiliselt ühendub võrku. Loogiline topoloogia mudel kirjeldab

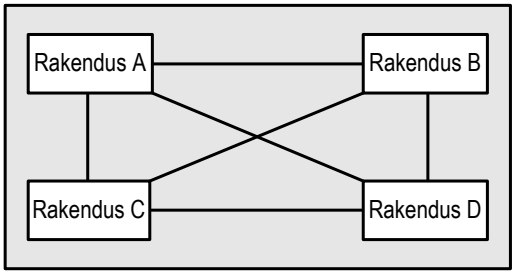
vahendeid, järgnevust ja protokolle, mida füüsilised sõlmed kasutavad kommunikeerumiseks. Mõnedel juhtudel võivad need topoloogia kihid üksteisega ühtida ja mõnedel juhtudel erineda. Integratsiooni topoloogia kiht kirjeldab kanalite ja koostööde korraldust ning mehhanisme, mida erinevad süsteemid ja teenused kasutavad kommunikeerumiseks. Joonis 16 näitab, kuidas punktist-punkti ühenduse korral võivad toploogiad seotud olla.



Joonis 16 Topoloogiate kihid (Trowbridge, 2004)

Integratsiooni topoloogiate kiht jaguneb punktist-punkti, sõnumi siini ja jaotur mustriteks, mida vaatleksin ka lähemalt järgmistes lõikudes.

Punktist-punkti *(Point-to-point).* Punktist-punkti ühenduse realiseerimine on üks lihtsamaid viise ühendada kaks süsteemi omavahel. Punktist-punkti integratsiooni korral kasutatakse lihtsat kanalit, et võimaldada ühel rakendusel liidestuda teise rakendusega: rakendus A liidestub rakendusega B. Kui rakendus A soovib kommunikeeruda rakendusega B, siis ta annab sellest teada, kasutades protseduuriväljakutset või saates sõnumi mööda kanalit. Saatja rakendus peab tavaliselt tõlkima sõnumi sellisesse formaati, mis vastuvõtjale sobib.



Joonis 17 Punktist-punkti topoloogia

Joonis 17 kujutab nelja rakendust, mis on punktist-punkti liidestatud. Kui rakenduses A näiteks muutuvad aadressid või muud detailid, siis tuleb kõiki ülejäänud rakendusi muuta, mis soovivad temaga kommunikeeruda.

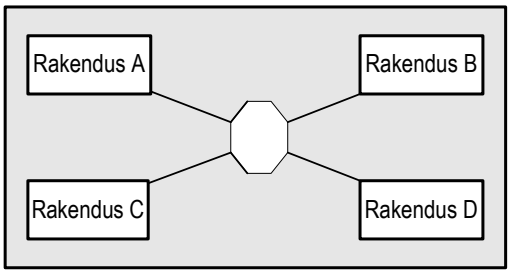
Võrreldes seda teiste mudelitega on see üsna limiteeritud lahendus, kuna korralikult saab omavahel siduda vaid kahte rakendust. Puuduvad veel sellised eelised, nagu vahekihi töötlus, oskus hoida rakenduse loogikat, võime muuta sõnumeid.

Kui on vajadus liita rohkem kui kaks rakendust üksteisega, siis traditsiooniline punktist-punkti mudeli kasutamine pole üldjoontes parim lahendus. Liidestades rohkem kui kaks rakendust point-to-point meetodil nõuab vastavaid liideseid kõikide asjassepuutuvate rakendustega.

Selline lähenemine integratsioonile põhjustab liigjäika ühendamist, mis tähendab, et üks arendajate meeskond peab kontrollima mõlema süsteemi integratsioonikoodi, et neid omavahel suhtlemas hoida. Samuti suureneb järsult ülalpidamine ja muudatuste juhtimine uute süsteemide ühildamisel.

Punktist punkti mudeli eeliseks on jällegi selle lihtsus. Ainult ühe rakenduse liitmine teisega vabastab integratsiooni arhitekti ja arendaja mitmete rakenduste lähte- ja lõpp punktide erinevuste kohandamise probleemidest *(Linthicum, 2003)*.

Jaotur *(Broker)*. Jaotur topoloogiat illustreerib Joonis 18. Jaotur mustrit ja tema alamvariante kasutatakse nii rakenduste- kui integratsiooni disainis.



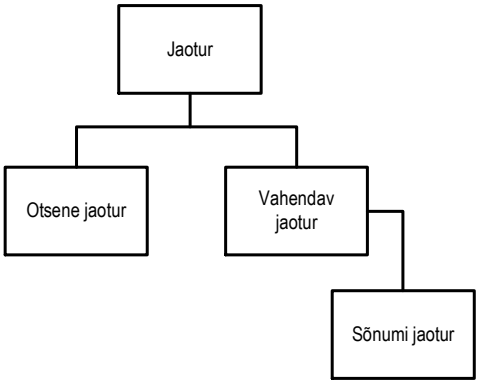
Joonis 18 Jaotur topoloogia

Jaoturi eesmärk on hoida lahus lähte- ja sihtkoha süsteem. Hajusobjektide integratsiooni kontekstis esitab lähtesüsteem sihtsüsteemile päringu meetodi käivitamise kohta saates kokkupandud paketi parameetreid. Teenusele orienteeritud integratsiooni kontekstis saadab lähtesüsteem sõnumi, mis küsib teenust sihtsüsteemilt. Mõlemal juhul kasutatakse jaotur mustrit, mis hoiab lahus lähte- ja sihtkoha süsteemid. Jaotur koordineerib seejuures kommunikatsiooni süsteemi lõpp-punktide vahel. Jaoturi vastutusalasse kuuluvad:

- Marsruutimine – marsruutimisel määratakse kindlaks sihtsüsteemi asukoht;
- Lõpp-punkti registreerimine – mehhanism, mida süsteem saab kasutada, et registreerida ennast ise jaoturis, et teised süsteemid saaksid uurida seda süsteemi;

- Teisendamine – teisendamine on protsess, kus elemendid konverditakse ühest formaadist teise.

Jaoturid jagunevad veel otseseks- ja vahendav jaoturiks, mille alla käib omakorda sõnumi jaotur. Joonis 19 näitab jaoturite jagunemist.



Joonis 19 Jaoturid

Otsene jaotur loob algse kommunikatsiooni lõpp-punktide vahel. Peale algset kommunikatsiooni kommunikeeruvad mõlemad lõpp-punktid otse, kasutades nii kliendi- kui serveripoolseid puhvreid. Vahendav jaotur on seevastu vahemehe rollis, vahendades kogu pooltevahel toimuvat kommunikatsiooni. Vahendava jaoturi kasutamine pakub keskse kontrolli sõnumite voost ja lisaks ei pea saatja teadma midagi sihtkoha detailidest. Otsene jaotur pakub paremat jõudlust, vahendav jaotur pakub paremat kontrolli kommunikatsiooni üle.

Sõnumi jaotur kasutab kommunikatsiooniks sõnumeid ja vahendavat kommunikatsiooni mudelit. Sõnumi jaotur on tihti kasutatav integratsiooni muster ja tuntud ka *hub-and-spoke* arhitektuurina. Jaoturi näidistena võib tuua:

- *Microsoft Distributed Common Object Model (DCOM*);
- *Microsoft .NET Framework Remoting*;
- *Common Oject Request Broker Architecture (CORBA)*;
- *Universal Description Discovery and Integration (UDDI)*;
- *Microsoft BizTalk Server 2004*.

(Trowbridge, 2004).

Sõnumi siin (*Message bus*). Siini topoloogiat illustreerib Joonis 20.

Joonis 20 Siini topoloogia

Sõnumi siin on loogiline komponent, mis spetsialiseerub sõnumite transportimisele rakenduste vahel ja sisaldab järgmisi põhilemente:

- Kogumit kooskõlastatud sõnumite skeeme;
- Kogumit üldisi käsusõnumeid;
- Hajutatud infrastruktuuri.

Kui kasutada sõnumi siini, siis rakendus, mis saadab sõnumi siinile ei oma edaspidi individuaalseid kontakte vastuvõtja rakendusega. Kui sõnum on saadetud siinile, transpordib sõnumi siin selle edasi läbi hajus infrastruktuuri kõigile huvitatud rakendustele.

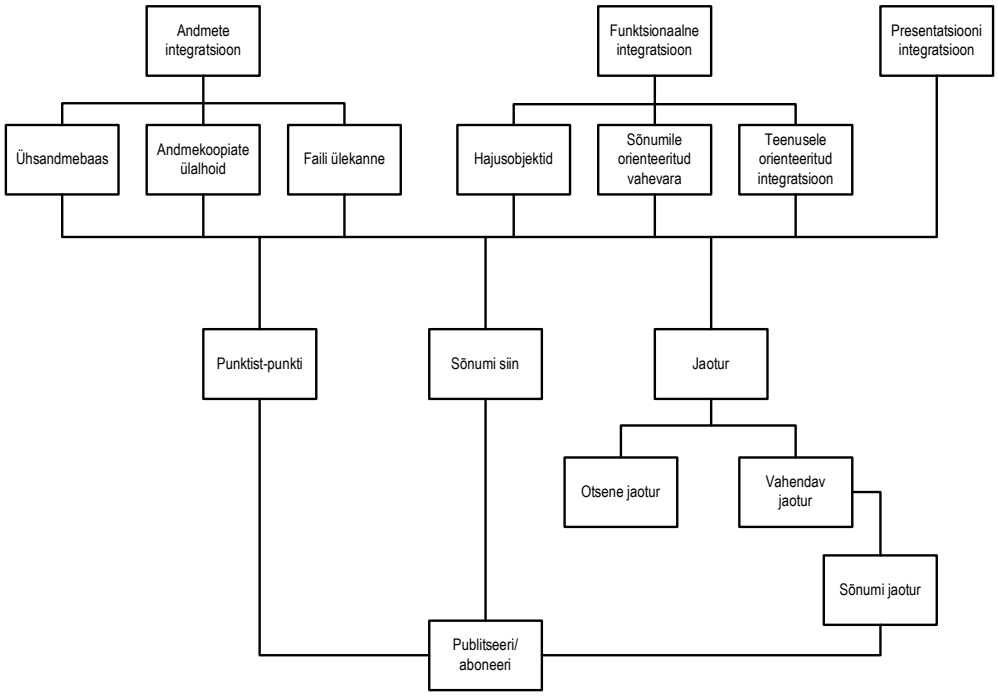
Tavaliselt siin ei hoia sõnumite järjekorda, see nõuab eraldi loogikat. Seda lisaoogikat saavad pakkuda osalevad rakendused. Näiteks, saatev rakendus võib lisada järjekorranumbrid väljaminevatele sõnumitele ja vastuvõtja võib neid sellejärgi järjestada.

Hajusat infrastruktuuri, mis on sõnumi siini ja huvitatud rakenduste vahel saab realiseerida sõnumi marsruuteriga või kasutades publitseer/aboneeri *(publish/subscribe)* mehhanismi.

Publitseeri/aboneeri (*Publish/subscribe*). Publitseeri/aboneeri mehhanism vabastab rakenduse vajadusest teada midagi sihtrakendusest. Vajalik info saadetakse sihtrakenduse poole läbi publitseeri/aboneeri mootori või jaoturi. Vahendaja jagab seejärel info kõikide huvitatud rakenduste vahel. Näiteks, kui finantsrakendus soovib kõikide kasutajakontode informatsiooni teha kättesaadavaks teistele rakendustele, pöördub ta publitseeri/aboneeri mootori poole. Mootor võtab teadmiseks, et selline info on saadaval ja iga rakendus võib tellida omale saadavalolevat infot. Selle stsenaariumi järgi on publitseerija info pakkuja. Publitseerija varustab infoga teemade kohta, aga samas ei pea midagi teadma rakendustest, mis on informatsioonist huvitatud *(Linthicum, 2003)*.

Integratsiooni mustrite seosed. Joonis 21 näitab ühenduse mehhansimi järgi liigitatud D. Trowbridge poolt käsitletud integratsiooni mustrite seoseid. Mustrid on kujutatud kastides ja jooned nende vahel näitavad seoseid.

Integreerides andmete kihti, on võimalik valida ühisandmebaasi-, andmekoopiate ülalhoiu- või faili ülekande mustri vahel. Funktsionaalse integratsiooni puhul võib valida hajusobjektide-, sõnumile orienteeritud vahevara-, või teenusele orienteeritud integratsiooni vahel. Punktist-punkti, sõnumi siin ja jaotur mustrid kuuluvad juba integratsiooni topoloogiate valikute alla.



Joonis 21 Integratsiooni mustrid ja nendevahelised seosed

2.5 Integratsiooni vahevara

Integratsiooni teemat käsitledes kohtab tihti vastavas kirjanduses terminit "vahevara" *(Middleware)*. Järgnevalt püüan anda ülevaate olemasolevatest vahevara tüüpidest, laskumata lähemalt iga vahevara tüübi detailsetesse kirjeldustesse, mis pole antud uurimistöö eesmärk.

Vahevara on mehhanism, mis lubab ühel üksusel (näiteks rakendus või andmebaas) kommunikeerida ühe või teiste üksustega. Teisisõnu: vahevara on igasugune tarkvara, mis aitab kaasa kommunikatsioonile ühe- või mitme tarkvarasüsteemi vahel reaalajas *(Linthicum, 2003,).*

Ka uurimuse praktikate osas käsitletud integratsiooni juhtumite juures tuleb tegmist vahevaraga. Küsitletavate arvamuse kohaselt võib vahevarana mõista spetsiaalselt andmevahetuse hõlbustamiseks arendatud moodulit (tavaliselt realiseeritud andmebaaside vahele), mis võib tegeleda andmete teisendamisega, summeerimisega, ja paljude muude tegevustega, mis võivad nõuda ka äriloogikat.

D. Lithicum on käsitlenud oma rakenduste integratsiooni raamatus järgmisi vahevara tüüpe:

- Kaugprotseduuride väljakutse;
- Sõnumile orienteeritud vahevara;
- Hajusobjektid;
- Andmebaasile orienteeritud vahevara;
- Transaktsioonile orienteeritud vahevara;
- Integratsiooni serverid.

Kuna integratsioonitüüpide all on eelpool mitmed nendest tüüpidest juba kirjeldatud, siis kirjeldaksin lühidalt andmebaasile orienteeritud vahevara, transaktsioonile orienteeeritud vahevara ja integratsiooni servereid.

Andmebaasile orienteeritud vahevara. Andmebaasile orienteeritud vahevara on iga vahevara, mis hõlbustab kommunikatsiooni andmebaasiga, kas rakenduse või teise andmebaasi vahel. Arendajad kasutavad tüüpiliselt andmebaasile orienteeritud vahevara kui mehhanismi ekstraheerida andmed lokaalsest- või kaugandmebaasist. Näiteks, et ekstraheerida informatsiooni, mis asub Oracle andmebaasis, peab arendaja käivitama andmebaasile orienteeritud vahevara, et logida andmebaasi sisse, pärida infot ja töödelda infot, mis on ekstraheeritud andmebaasist *(Linthicum, 2003)*.

Transaktsioonile orienteeritud vahevara. Transaktsiooniline vahevara alla kuuluvad transaktsioonitöötlus monitorid *(Transaction processing monitor, TP monitor)* ja rakendusserverid. Sedatüüpi vahevara ülesanne on koordineerida informatsiooni liikumist ja meetodite jagamist paljude erinevate ressurside vahel. Kuigi transaktsiooniline vahevara pakub suurepärast mehhanismi meetodite jagamiseks, pole see kuigi efektiivne moodus info jagamisel, mis on tihtipeale rakenduste integratsiooni eesmärk. Transaktsioonilisel vahevaral on tendents luua tihedalt seotud rakenduste integratsioonilahendusi

Transaktsioonitöötlus (TP) monitorid on esimese generatsiooni rakenduste serverid, nagu transaktsioonilised vahevara tooted. Nad pakuvad mehhanismi, mis hõlbustab kahe või enama rakenduse vahelist kommunikatsiooni. TP monitorid ja transaktsiooni serverid baseeruvad transaktsiooni kontseptsioonil – ühik tööd, millel on oma algus ja lõpp. Põhjenduseks on siin, et kui rakenduse loogika on kapseldatud transaktsiooni sisse, siis tarnsaktsioon kas lõpetab oma töö täielikult või pöördub tagasi kui tekib probleem. Ka siis, kui transaktsiooni on uuendatud kaugresursside poolt, nagu andmebaasid ja päringud, pöördub transaktsioon vea korral tagasi. TP monitor tagab põhiliste teenustena transaktsiooni terviklikkuse ja pakub lisaks ressursi juhtimist ja käitusaja juhtimise teenust. TP monitoride näitena võib tuua *BEA Systemsi* poolt toodetud *Tuxedo*, *Microsofti* tehtud *MTS*, *IBM-I CICS*.

Enamik rakendusservereid on kasutusel kui veebipõhiseid lahendusi toetavad vahevarad, töödeldes veebirakenduste transaktsioone. Enamik neist kasutab ka kaasaegseid keeli nagu Java. Lihtsalt öeldes rakendusserverid pakuvad rakenduste loogika jagamist ja töötlemist ning ühendusi tagumistele ressurssidele *(back-end recources)*. Need ressursid hõlmavad andmebaase, ettevõtte ressurside planeerimise rakendusi *(Enterprise Resource Planning, ERP)* ja suurarvuti rakendusi. Rakendusserverid pakuvad mehhanisme ka kasutajaliidese arenduseks, rakenduste juurutamiseks veebiplatvormile *(Linthicum, 2003)*.

Integratsiooni Serverid. Integratsiooni serverid, hõlbustavad info liikumist kahe või enama ressursi vahel (lähte- ja sihtrakenduse vahel) ja võimaldavad arvet pidada erinevuste kohta rakenduste platvormis ja semantikas. Integratsiooniserverid võivad liita mitmeid rakendusi, kasutades põhireegleid ja marsruuterit. Neil on võime transformeerida informatsiooni skeeme ja sisu, mis liigub rakenduste ja andmebaaside vahel ja lisaks võivad nad omada veel mitmeid lisafunktsioone, nagu protsessi integreerimis mootor ja liides, juhtimismehhanismid. Lühidalt öeldes on nad info vahendajad rakenduste ja andmebaaside vahel viisil, kus rakendused ei pea teadma üksteise siseelust midagi ja seega täiesti sõltumatud *(Linthicum, 2003)*.

2.6 Soovitused integratsioonilahenduse valikul

Nagu eelmistest jaotistest selgus, on rakenduste integreerimise võimalusi palju: võib kasutada mitut tüüpi integratsiooni ja mitmeid tehnoloogiaid. Võib kasutada keerukat vahevara, mis juhib kogu kommunikatsiooni või sünkroniseerida andmebaaside vahel lihtsate vahenditega andmeid. Integratsiooni seisukohast oleks hea rakendus õhuke presentatsioonikiht, mis tarbib jagatud funktsionaalsust ja andmeid.

Käsitletud integratsioonilahenduste juures on enamik juhtudel välja toodud ka lahenduse tugevad ja nõrgad küljed, mis peaksid aitama otsuste vastuvõtmisel. Millised on seejuures valikud, mis rahuldavd tänased integratsiooniprobleemid ja potentsiaalselt ka homsed? See on üks keerulisemaid küsimusi.

Gartner grupi soovitusel *(Schulte 2002)* peaksid ettevõtted püüdma minimiseerida individuaalselt disainitud punkt-punkti meetodil rakenduste liidestamisest, mis suruvad peale märkimisväärse ülalpidamis koormuse ja seda eriti suures ettevõttes, kus kasutusel palju rakendusi.

Tänapäeva rakenduste integratsiooni märksõnadeks sobiksid protsessikesksus ja sõltumatus. Integreeritud süsteem peaks olema valmis reageerima muutustele, mida võib esile kutsuda äriprotsessi muutumine. Et seda nõudmist täita, peavad rakendused olema võimalikult sõltumatud üksteisest – see tähendab rakenduste lõdvalt sidumist. Lõdvalt seotud rakendused ei pea teadma süsteemi kohta enne kasutamist muud lisainfot, peale selle asukoha. Süsteemi funktsioonid ja programmid on iseseisvad ja seetõttu sõltumatult väljavahetatavad, nad ei nõua muude komponentide väljavahetamist koos nendega. Soovitavalt tuleks kasutada ka standardseid andmeformaate. Enimakasutust leidnud standardite jälgimine aitab rakendusi omavahel ühtselt liita ja pakkuda ka liideseid välistele süsteemidele. Näiteks *XML* kasutamisel andmete edasitoimetamiseks tuleks kasutada kindlasti ühte standardset versiooni, mitte paljusid eri variante.

Kaasaegsed integratsioonilahendused ei vaatle integratsiooni kui ülesannet, kuidas infot erinevate süsteemide vahel liigutada, vaid fookus on suunatud küsimusele, kuidas kasutada süsteemi võimalikult hea teenusepakkujana. See

võimaldab infosüsteemile läheneda kui terviklikule teenuste võrgustikule. Nii mõeldes suudetakse kasutada olemasolevaid funktsioone teistmoodi – ehitada uut süsteemi, laiendada kasutusvaldkondi, komponeerida uusi kasutusvaldkondi.

Õiged tehnoloogilised valikud on kahtlemata kriitilise tähtsusega, kuid sõltuvad eelkõige nõudmistest ja võimalustest. Kui nõuded näevad ette, et lühikese ajaga tuleb andmebaasist suur hulk andmeid ülekanda, siis paraku pole alati võimalik teenusepõhiselt läheneda.

Kas lahendused otsustatakse oma jõududega või kaasatakse ka eksperte mujalt, kuulub rohkem juhtimisalaste otsuste hulka, mida käsitlen järgmises jaotises.

2.7 Integratsiooniprojektide ja tööde juhtimise probleemid

Butler Groupi väitel keskmiselt 20% suurtest integratsiooniprojektidest ebaõnnestuvad *(ComputerWire, 2002)*.

Harva tuleb ette selliseid tarkvararendusprojekte, kus poleks mingeid probleeme. Rakenduste integratsiooni puhul, on autori arvates riskide hulk veelgi suurem, kui liidestamisvajaduseta rakenduse arenduse puhul. Alljärgnevas peatükis on väljatoodud mõned olulisemad juhtimisalased probleemvaldkonnad rakenduste integratsioonis. Need on jagatud kolme jaotise vahel:

- Andmeanalüüs;
- Arenduspartneri valik;
- Sobiv arendusmetoodika;
- Integratsiooni kompetents.

Probleemide ennetamiseks ja vältimiseks on peatüki lõpus ka juhtimisalased soovitused, millele tuleks tähelepanu pöörata rakenduste integratsiooni teostusel.

Andmeanalüüs. Ligi 90% andmete integratsiooni projektidest ületavad plaanitud eelarvet – keskmiselt 66% ulatuses või kukuvad üldse läbi, seda väidab Standish Group, kes arvab, et võtmepõhjuseks on kehva andmeanalüüs. Standish Group väidab, et projektgrupid kulutavad poole oma ajast püüdes andmetest aru saada, kuigi neil napib selles vallas kogemusi ja nad teevad ebakorrektseid eeldusi andmete sisu, struktuuri ja kvaliteedi osas (*British Computer Society 2002*).

Efektiivseks integratsiooniks on kahtlemata oluline omada täpset ülevaadet andmetest ja nende struktuurist. Andmebaasidel on aga kalduvus aja jooksul muutuda - sisestatakse valeandmeid, ebastandardseid andmeid jms. Selletarvis oleks igati mõistlik enne rakenduste integreerimist need üle kontrollida ja korrastada.

Bloor Research analüütiku, Philip Howardi sõnul on analüüsi ja disaini etapp jäänud manuaalseks, mille tulemusena on üldiselt limiteeritud analüüsi ja disaini hulk, millele järgneb seeria iteratsioone ehituse ja testimise tsüklis. Tavaliselt tekib vajadus nii paljude iteratsioonide järele kuna analüüs ja disain ei olnud algusest peale korralikult tehtud (*British Computer Society 2002*).

Andmeanalüüsi puudulikkus on ka töö autori arvates integratsioonis üks suuremaid probleeme, miks andmevahetus kipub vigaseks jääma ja vigade parandamisele kulub hulk lisaaega. Tarkvaraarendajad pakuvad ka automatiseeritud andmeanalüüsi vahendeid, kuid antud juhul ei oska adekvaatselt hinnata nende kasutegurit, kuna puudub kogemus.

Avellinno Tecnologies (andmete integratsioonile spetsialiseerunud firma) tegevdirektor Tony Rodriguez väidab, et enamik projektide tiime alustavad andmete integreerimisega ilma igakülgse arusaamiseta andmetest- ja see on üks peamisi põhjuseid miks projektid ületavad oma planeeritud maksumust. "Paljude projektgruppide valitud ressursid põhinevad sisetundel või õpitud oletamisel

tehtud prognoosidele. Kuid täpne planeerimine enne andmete liigutamist on lõpp-projekti õnnestumise võti- ja sa ei saa planeerida seda, mille kohta sa ei tea. Kui kompaniid ei hinda oma andmeid adekvaatselt, võivad nad eksida andmete integratsiooni strateegia ja disaini valikul". Ta jätkab, et tõenäoliselt suurim probleem selles, kuidas kompaniid oma andmeid hindavad, on eeldused mida tehakse metaandmete kohta: informatsioon mis kirjeldab allikandmete struktuuri. "Ei ole võimalik ehitada andmete integratsiooni liideseid kui sa ei saa täielikult aru andmetest millega sa manipuleerid ja metaandmed ei anna täielikku vastust. See on nii põhiliselt sellepärast et nende aastate jooksul, mil andmebaas on kasutuses, muutuvad andmed olulisel määral. Kliendiandmete välja võib võtta korduvkasutusse kui äriprotsessid muutuvad, salvestamaks terve andmebaasi ümberstruktureerimist. Operaatorid võivad sisestada täiteinformatsiooni nagu näiteks nende oma kompanii aadressi kliendi aadressi asemel et kiirendada oma andmesisestusjõudlust "(*British Computer Society 2002*).

Puudulikust andmeanalüüsist tekkinud probleemidega olen ka isiklikult kokku puutunud konkreetse juhtumi käigus, mida ka töö juhtumite osas kirjeldan. Tegemist on minu hinnangul olulise aspektiga rakenduste integratsioonis, millele tuleks erilist tähelepanu pöörata juba projekti algfaasis.

Arenduspartneri valik. Kui organisatsioonis ei tehta IT projekte ainult majasiseste jõududega, on õige arenduspartneri valik kahtlemata oluline otsus mistahes tarkvaraprojekti läbiviimisel. Kui meil on aga tegu rakenduste integratsiooniga, tuleks seda kriitilisemalt suhtuda partneri seni läbiviidud töödesse ja integratsioonialasesse kompetentsi. Olulised kriteeriumid on siinjuures arenduse kiiruse ja lahenduste kõrge kvaliteedi suhe.

Üks võimalus tagada kõrget kvaliteeti on selekteerida süsteemi integreerijad lahenduste väljatöötamise kogemuse järgi. Kui investeerida integreerija kogemusele, peaks vaatlema mitte ainult teostatud projektide hulka, vaid ka seda, kui paljud neist on lõpetatud planeeritud ajaga. Tihti juhtub, et rakenduste integreerimisperioodil mõtlevad organisatsioonid välja uut funktsionaalsust, mida lisada uuele süsteemile. Sellised soovid mõjuvad laastavalt projekti ajagraafikule. Süsteemi integreerija peab sellisel juhul selgitama koheselt muudatuste mõju ja aitama organisatsioonil prioritiseerida sellised soovid.
Siinkohal toon välja süsteemi integratsiooniga tegeleva firma ThoughtWorks väljapakutud olulisemad momendid, mida arendaja valikul peaks jälgima.

Kui eesmärgiks on ärilt-ärile integratsioon, siis on oluline vaadelda just kogemust integreeritud süsteemi arendamisel, kus kommunikatsioon toimub erinevate firmade vahel. Reeglina peab siin ühenduma läbi Interneti ja seega tõusevad olulisele kohale ka turvalisuse nõuded. Paljud sellised süsteemid nõuavad samat tüüpi integratsiooni arhitektuuri ja XML-i, et töödelda admeid ja saata neid tagasi või edasi üle veebi. Valitud süsteemi integraator peaks olema tuttav e-äri integreerimise viimaste standarditega. Need standardid sisaldavad *XML, HTTP* ja *Secure Sockets*. *XML* on andmete defineerimis keel, mis on

tõusnud standardiks andmete ja dokumentide vahetamisel. *HTTP* lubab organisatsioonidel andmeid vahetada läbi tulemüüri. Secure Socket on turvalise *HTTP* standard *(S-HTTP)*, et pakkuda tugevat krüptot, mis takistab autoriseerimata ligipääsu andmetele, mida liigutatakse üle interneti *(Thoughtworks, 2002)*.

Sõltuvalt arendaja kogemusest, võib saada ka objekt-orienteeritud lähenemisega häid tulemusi intergatsiooni teostusel. Objekt-orienteeritud programmeerimine kujutab asju nagu objekte, milledest on lihtsam aru saada ja manipuleerida kui eraldiseisvate andmeelementidega ja funktsioonidega. Objekt-orienteeritud tehnoloogia kapseldab programmi moodulid, mis lihtsustavad arendust ja ühtlasi süsteemi ülalpidamist ja muudatusi, sinnamaani kuni üks rakenduse osa ei hakka häirima teisi osasid. See tehnoloogia lubab korduvkasutada analüüsi, disaini ja teostuse produkte, ilma vajaduseta uuesti programmeerida sama funktsiooni uuesti ja uuesti. Objekt-orienteeritud arendus lubab programmeerijatel ka läheneda kodeerimisele loomulikumal moel *(Thoughtworks, 2002)*.

Paljud organisatsioonid peavad integreerima mitmetele heterogeensetele süsteemidele. Sõltuvalt muidugi integratsiooni eesmärgist ja ulatusest tuleb vaadata sedagi, kas partneril on kogemust eri platvormidega nagu Windows NT ja UNIX süsteemidega.

Arendajate poolt tarnitav peaks sisaldama standardiseeritud UML diagramme, klasside diagramme, järgnevus diagramme, oleku mudelit, tegevusdiagrammi *(activity diagram)* rakenduste analüüsiks ja disainiks. Need diagrammid aitavad kindlustada, et rakendus on täielikult dokumenteeritud ja arusaadaval kujul, mis lihtsustab süsteemi ülalpidamist.

Lõppude lõpuks loeb muidugi ka eelnev kogemus ja inimesed, kes firmas töötavad. Firma, kus on paremad inimesed, lahendab alati probleemi kiiremini. Et kindlaks teha inimeste kvalifikatsioon, tuleks kindlasti küsida referentse ja seejärel uurida neid.

Sobiv arendusmetoodika. Enamik süsteemi integraatoreid kasutab mingit tüüpi metoodikat, mille abil lahendusi teostatakse. Metoodika tähendab süsteemset lähenemist rakenduste arendusprotsessis. Süsteemiintegratsiooniga tegelev firma *ThoughtWorks* leiab, et tänapäeva rakendustes, kus kiire turuletulek on hädavajalik on optimaalseks metoodikaks iteratiivne lähenemine. Traditsioonilisel koskmudelil põhinev arendusprotsess on tunduvalt ajamahukam. Selline lähenemine on riskantne, kuna ärivajadused muutuvad kiiresti ja neile on vaja koheselt reageerida. Iteratiivne lähenemine seevastu annab väiksemad tarned lühikese perioodi jooksul. See lähenemine võimaldab kohandada süsteemiarendust vastavalt firma ärivajadustele *(Thoughtworks, 2002)*.

Samal arvamusel on ka Walker Royce kirjutades oma raamatus: "seniste praktikate põhjal sobib integratsiooniprojektide arenduseks kaasaegne, iteratiivne lähenemine. Pikalevenivat integratsiooni ja hilisemaid disaini vigu aitab vältida

integratsiooni varane käsitlemine projekteerimisfaasis. Iteratiivne arendus produtseerib kõigepealt arhitektuuri, kus disaini faasis verifitseeritakse ka integratsioon. See võimaldab disaini vigu avastada ja lahendada need tarkvara elutsükli varasemas faasis. Selline lähenemine väldib suure pauguga integratsiooni projekti lõpufaasis, mis koormab jätkuvat integratsiooni läbi projekti" *(Royce, 1998)*.

Tavapärase arendusprotsessiga teostatud projektid kulutavad ümaralt ligikaudu 40% või rohkem kogu ressursist integratsiooni ja testimise peale. Ja seda ebatõhusa intergratsiooni ja disaini vigade ilmnemise pärast hilises faasis. Kaasaegsed projektid suudavad küpse iteratiivse arendusprotsessi puhul tarnida toote kasutades ca. 25% kogu eelarvest *(Royce, 1998)*.

Ka meie keskkonnas integratsiooni teostanud projektijuhid on kinnitanud iteratiivse arenduse sobivust integratsiooni läbiviimisel. Seega peaks antud meetod oma sobivuses olema küllalt tõestust leidnud.

Integratsiooni kompetents Teostades organisatsioonis integratsiooni arenduspartnerite abil tuleb kindlasti jälgida, et integratsiooni alane kompetents jääks organisatsiooni. Vastasel juhul tekib olukord, kus ollakse arenduspartneritest liiga sõltuvad, samuti on keeruline teostada järgnevad integreeritud arendusi kui puudub vastav kompetents. Selleks soovitab uuringufirma *Gartner* kasutada tsentraliseeritud integratsiooni kompetentsi keskust. Tsentraliseeritud integratsiooni kompetentsi keskus võib anda tulemusi väiksema praktika hinnaga, sest vähem inimesi peavad tundma integratsiooni detaile. See lühendab arenduse tsükleid, sest inimestel on kogemus integratsiooniprobleemide lahendamiseks ja vahevara tehnoloogiate kasutamises *(Schulte 2002)*.

2.7.1 Soovitused integratsioonitööde juhtimisel

Selles jaotises pakuksin välja kirjanduses käsitletu põhjal soovitused, millele tuleks tähelepanu pöörata integratsiooni juhtimisel.

- Arenduspartneri valik. Kui integratsiooni läbiviimiseks kaasatakse väline partner tuleb valikuid kaaluda põhjalikult. Õige arenduspartner omab kõrgkvalifikatsiooniga tööjõudu, laiahaardlist kogemust ja tõestatud laiaulatuslikke kogemusi integratsiooniprojektidega, kasutades viimast tehnoloogiat ja paindlikku metoodikat.
- Analüüsi kvaliteet. Tasub panustada projekti alguses kvaliteetsesse analüüsi ja kindlustada, et see vastaks nõudmistele, siis võib liikuda järgmise taseme detailidesse. Sel moel ei pea ootama disaini faasi saamaks aru, et osa analüüsist on vale, või ei lähe te liiga kaugele valet teed mööda ja olla sunnitud otsuseid tühistama, mis mõlemad on väga kallid.
- Allikandmete analüüs. Tuleb tagada allikandmete analüüs sobivas mahus ja sobiva aja jooksul. Tüüpiline andmete analüüsi projekt võib sõltuvalt alliksüsteemide suurusest kesta neljast nädalast kuue kuuni. Projektgrupid

analüüsivad tihti allikandmeid väljaspool konteksti: näiteks analüüsitakse tihtipeale kõiki andmeid kui vaja oleks ainult poolt sellest (*British Computer Society 2002).*

- Õiged prioriteedid. Oluline on paika panna õiged prioriteedid ja alustada keerulisemate töödega, et maanda riske. Keerulisteks ja suurema riskiosalusega töödeks võivad olla liidestamised teiste süsteemidega. Tihti juhtub, et projektgrupid alustavad kergemate töödega, kulutavad nendepeale liiga palju aega ja olulised riskantsed liidestamised jäetakse hiljemaks.
- Konsultantide kaasamine. Mõistlik on kaasata kogenud konsultante kohe algusest, ideaalis projekti hindamise faasis. See aitab õigesti ennustada tõenäolist ajagraafikut ja maksumust ja lubab projektil hästi alata. Sõltumatud eksperdid võivad aidata kindlustada analüüsi effektiivsust.
- Iteratiivne arendus. Mitmete projektide puhul on tõestust leidnud fakt, et iteratiivne arendusmetoodika on integratsiooni läbiviimisel tõhusam.
- Integratsiooni kompetentsi säilitamine. Teostades organisatsioonis integratsiooni arenduspartnerite abil tuleb kindlasti jälgida, et integratsiooni-alane kompetents jääks organisatsiooni. Vastasel juhul tekib olukord, kus ollakse arenduspartneritest liiga sõltuvad, samuti on keeruline teostada järgnevaid integreeritud arendusi kui puudub vastav kompetents.

3 INTEGRATSIOONI PRAKTIKAD

3.1 Empiirilise andmestiku kogumise meetodid

Integratsioonipraktikate kohta info saamiseks tundus kõige õigem olevat teha intervjuud projektijuhtide ja analüütikutega, kellel on rakenduste liidestamisalaseid kogemusi. Intervjuude valik küsitlusmeetodina annab suurema võimaluse saada läbikaalutud ja põhjalikumaid vastuseid. Tarkvaraprojektijuhte, kes on läbiviinud ulatuslikke rakenduste integreerimistöid pole eestis väga palju ja reeglina on nad ka üsna hõivatud. E-maili aadressile saadetud küsitlus ei anna tihti oodatud tulemusi ja on oht, et vastuseid ei saada üldse või saadakse osaliselt (siinkohal toetun teiste lõputöö tegijate kogemustele). Seega tundus mõistlik olevat intervjuude vorm, mida läbiviia nn. *face-to-face* meetodil, mis lubab intervjuu kulgu ka dünamiilisemalt juhtida, arvestades intervjueeritava kogemusi ja võimalikku kasutegurit tööle. Olles ise tegev tarkvara projektijuhina on mul piisavalt kokkupuuteid tarkvaraarenduse alal koostööpartneritega ja oma asutuse projektijuhtidega. Sellest tulenevalt on olemas ka head võimalused saada kokkuleppele sobivate inimestega, kellel on vastavad kogemused ja kes nõustuksid ka oma kogemustest lähemalt rääkima.

Intervjuudest tõin eraldi välja ka mõnede praktiliste juhtumite kirjeldused, et selgitada välja juhtumites kasutatud tehnoloogiad ning probleemid.

Intervjuude küsimustik. Küsimustiku koostamisel lähtusin kirjanduses käsitletud integratsiooni lahendustest, enim käsitlust leidnud probleemidest ja enda kogemustest. Kuna intervjueeritavateks on peamiselt projektijuhid ja äriprotsesside analüütikud, ei ole eesmärgiks süüvida tehnoloogilistesse üksikasjadesse, käsitleda ühe või teise lahenduse kasutamise spetsiifilisi aspekte. See on rohkem süsteemianalüütikute ja süsteemiarhitektide pärusmaa ja jääb antud töö skoobist välja. Küsimused on pigem üldised, püüdes intervjueeritavalt välja selgitada integratsiooni temaatikas ettetulevaid probleeme ja lahendusi seniste kogemuste põhjal.

Sõltuvalt intervjueeritava kogemustest, teadmistest ja intervjuu käigust, on küsimused varieeruvad ja mõni teemade valdkond leiab sealjuures laiemat käsitlust. Arvestades inimeste hõivatust ja mõistlikku ajakulu, püüdsin iga intervjuu mahutada maksimaalselt 40 - 60 minuti sisse, sellest tulenevalt on ka küsimusi ca 12 – 15. Järgnevalt on väljatoodud aluseks olnud küsimustik, mille alusel intervjuud on läbiviidud:

1. Kuidas Te defineeriite tarkvaraarenduse seisukohast integratsiooni ja integratsiooiprojekti mõistet?
2. Mis on liidestamine? Kas see termin on teistmoodi mõistetav kui integratsioon?
3. Mis on vahevara integratsioonis Teie arusaama järgi, kas on ka sellega kogemusi?

4. Millised on Teie suuremad integrasiooni/liidestamis projektid?
5. Kirjeldage palun mõne integratsiooni osalusega projekti käiku: projekti tellija, teostaja, lähteülesnne, teostatud lahendus, ajakava, probleemid.
6. Kas tuli ette ka probleeme andmeanalüüsiga, andmeallikatega? Kas tekkis olukordi, kus andmenalüüsi vigade tõttu ei hakanud liidesed korralikult tööle?
7. Kas arendamisel kasutasite iteratiivset lähenemist?
8. Kui jagada integratsioon ühenduse järgi, siis saab seda teha nii andmebaasi kihti, loogika kihti kui presentatsiooni kihti. Kas Teiepoolt kirjeldatud projekti saab ka *Microsofti* poolt pakutud skeemi järgi *(Trowbridge, 2004)* vaadelda?
9. Millised võiks olla integratsiooni või liidestamise juures lahendused, tehnoloogiad, mida eelistada?
10. Kas kliendid arvestavad partnerite poolt pakutud lahendustega ja kas neil on lahenduste suhtes ka mingid omad ettekirjutused?
11. Kas olete kohanud etevõtetes ka üldist integratsiooni strateegiat, kus rakendusi teostatakse viisil, mis tagab suhtlemise edaspidi ka teiste rakendustega? Või lahendatakse pigem antud projektiraames olev probleem?
12. Kas liidestamisprojekti peab ka kuidagi teistmoodi juhtima kui ilma liidesteta tavatarkvaraprojekti? On seal mingisugust spetsiifikat või mitte?
13. Kas liidesed suurendavad projekti riske?
14. Kas võib öelda, et olulise liidese puhul tuleb see projekti varajases faasis viivitamatult ette võtta?
15. Kas Te oma arenduses mingeid valmis adaptereid, müügilolevaid pakitooteid pole kasutanud?

Järgnevalt toon välja mõned konkreetsed integratsiooni juhtumid, mis selgusid läbiviidud intervjuudest ja kirjeldan ka endapoolt läbiviidud integratsiooni juhtumit (Projekt 4).

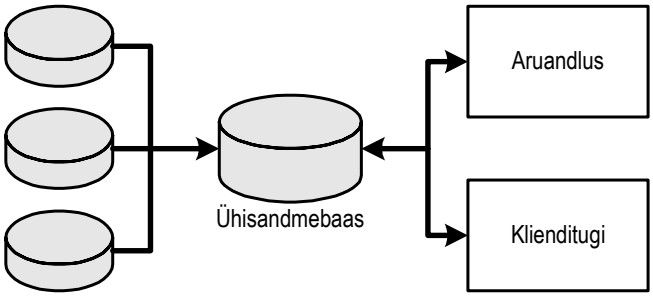
3.2 Integratsiooni juhtumid

Kirjeldatud juhtumid on Eestis läbiviidud projektid, kus liidestada on tulnud kaks või enam rakendust omavahel. Konfidentsiaalsus nõuete tõttu pole kõigis juhtumites mainitud projektides osalenute ettevõtete nimesid. Juhtumite kirjapanekul olen püüdnud väljatuua projektide lähteülesande, realiseeeritud integratsioonilahenduse, teostamise ajakava ja põhilised probleemid.

Projekt 1: Riigiettevõtte ühe valdkonna protsessi jälgimise süsteem . Tegemist on suuremat sorti riigiettevõttega ja eesmärgiks oli algselt integreerida omavahel mitmed rakendused, toetamaks äriprotsessi. Visoonis nähti ühtset infosüsteemi, mis pakuks infot nii klienditeenindaja kui juhtimistasandil. Kogu seda

informatsiooni pidi võimalik olema ka käepäraste vahenditega analüüsida ja töödelda.

Tellija: Riigiettevõte
Teostaja: Abobase Systems
Lähteülesanne: Esialgne ettevõtte poolt püstitatud ülesanne osutus siiski võimatuks ja analüüsi käigus otsustati pilootprojektina realiseerida funktsionaalsus ühe valdkonna protsessi piires. Projekti tulemiteks nähti ette kahe rakenduse teostust: aruandluse rakendus, mis toetab juhtimisotsuseid ja klienditoe rakendus, mis on suunatud teenindajatele.
Teostatud lahendus: Et saada vajalik info ühtekokku tuli see korjata välja mitmetest baasidest ja integreerida ühte ühisandmebaasi, vt. Joonis 22.



Joonis 22 Projekt 1 teostatud lahendus

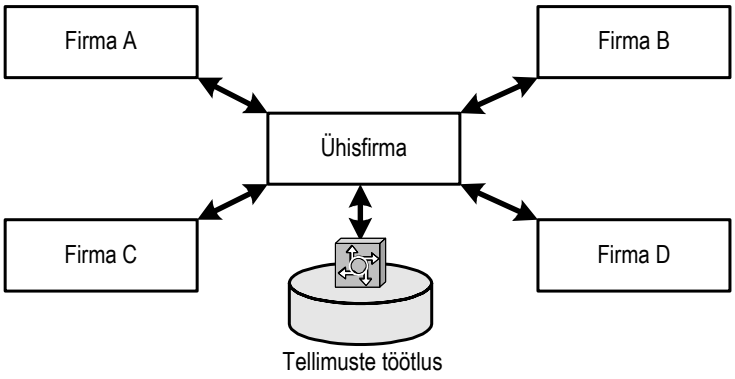
Andmete ülekandmine lahendati igaöise andmete ülekandmisega ja olemasolevad andmeallikad jäid kõik tööse. Uued klienditoe- ja juhtimistoe rakendused tarbisid juba ühisandmebaasi, sealjuures ka ühisandmebaasi agregeeritud andmete osa, et mitte ajada info mahtusid väga suureks. Antud juhul kasutati lahendusena andmete replikatsiooni, ühisandmebaasi ja punktist punkti topoloogiat. Kuna integratsioon toimus ettevõtte sees, siis tarbivad rakendused said baasi poole pöörduda otsepäringutega. See tagas piisava kiiruse ja ka turvalisuse jaoks ei olnud tarvis erimeetmeid rakendada.
Ajakava: Projekti täpsustamise ja analüüsi peale kulus 6 kuud ja realisatsiooniks teine 6 kuud. Realisatsiooni etapp kestis 1 kuu kauem esialgsest plaanitust, mis oli kooskõlastatud ka tellijaga.
Probleemid:

- Ülesande püstituse ebaselgus. Teliijal ei olnud konkreetset vaadet, mida tahetakse saavutada. Ülesande püstituse ebaselgusest kujunes pikem vajaduste täpsustamise etapp, mis siiski lahendati erinevate analüüside tulemina, kuid see kõik võttis aega esialgselt plaanitust rohkem.
- Kooskõlastuste pikk aeg. Suures organisatsioonis tuleb tihti ette, et erinevad kooskõlastused eri osakondade vahel võtavad aega ja see venitab protsessi.

- Muudatused äriloogikas. Realisatsiooni käigus tuli tellija poolt muudatusi ja ilmnes ka tähelepanuta jäänud üksikasju, mis mõjutasid arendust ja see omakorda projekti ajakava.

Projekt 2: Ühisettevõtte tellimuste infosüsteem. Nelja ettevõtte konsortsiumi eestvõttel loodi ühisettevõte, millele tuli ehitada ka infosüsteem, mis pidi liidestuma ka oma emaettevõtetega. Eesmärgiks oli teha ühine tellimuste infosüsteem, kuna leiti, et ühe protsessi lahendamine ühiselt tuleb märkimisväärselt odavam.

Tellija: Nelja ettevõtte konsortsium
Teostaja: Tarkvarafirma
Lähteülesanne: Luua ühtne tellimuste infosüsteem koos sellega kaasnevate liidestega emaettevõtete ja ühisettevõtte vahel.
Teostatud lahendus: Lahendusena teostati süsteem, kus tellimused liikusid emaettevõtte infosüsteemidest ühisettevõtte kesksesse infosüsteemi, kus toimus tellimuste töötlemine ja tagasiside Joonis 23. Liidestamise realisatsioon baseerus sõnumivahetusel ja kasutati punktist-punkti topoloogiat.



Joonis 23 Projekt 2 teostatud lahendus

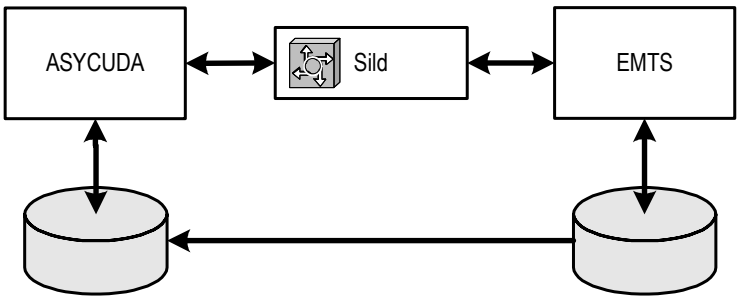
Kuna andmevahetus toimus erinevate organisatsioonide vahel, kasutati sõnumivahetuseks laivõrku ja turvakanalit, sest tegemist oli sensitiivsete andmetega.
Ajakava: Kogu projekt kestis neli aastat. Ühisfirma infosüsteem rakendati tööle, kuid tellimuste info liides sai valmis plaanitust üks aasta hiljem.
Probleemid:

- Muudatused äriloogikas.Tellimuste liides realiseeriti plaanitust hiljem ja esialgsest plaanist teistsugusel kujul. Projekti arenedes muutusid paljud nõuded, mis tõi esile vajaduse arhitektuuri muudatusteks, mis omakorda nõudis liideste disaini ümbertegemist.
- Turvalisuse nõuded. Kõrgendatud turvanõuded lisasid keerukust lahendusele, tuli otsida uusi võimalusi andmevahetuseks.

- Uus tehnoloogia. Arendajatel kulus täiendavat aega turvalisusega seotud uuema tehnoloogia juurdeõppimiseks ja sobiliku lahenduse leidmiseks ja teostamiseks.
- Tellija projektijuhtimine. Puudus konkreetne otsustaja, kes oleks kinnitanud pakutud lahendusi või strateegilisi otsuseid. Puudulik oli ka tellija projektimeeskonna tehniline kompetents.
- Erinevatest osapooltest tulenevad probleemid. Tulenevalt erinevatest osapooltest ja hilisematest firmade liitumistest tekkisid ka erinevad huvid süsteemi teostusel ja see häiris projekti arengut. Häiritud oli ka kommunikatsioon, keeruline oli leida õiget inimest ja saada ka pädevat informatsiooni.

Projekt 3: Tollideklaratsioonide süsteemi liidestamine Tollitariifide süsteemiga.
Seoses Eesti liitumisega Euroopa Liiduga, tekkis vajadus andmevahetuseks Euroopas kasutatavate tollisüsteemidega. Vaja oli kanda üle suuremas mahus andmeid ja teostada ka lühikesi päringuid reaalajas.

Tellija: Maksu- ja Tolliamet
Teostaja: UNCTAD *(United Nations Conference on Trade and Development)*
Lähteülesanne: Tollideklaratsioonide süsteem ASYCUDA tuli liidestada Tollitariifide süsteemiga EMTS. ASYCUDA vajas deklaratsioonide kinnitamiseks andmeid kaubakoodidele rakendatud meetmete kohta. Sealjuures tuli arvestada allikandmete muutumise võimalusega.
Teostatud lahendus: Integratsioon teostati kahel tasemel, andmete ülekandega baaside tasemel ja kaugpäringuga deklaratsiooni töötlemisel, vt. Joonis 24. EMTS-ist replikeeriti ASYCUDA-sse öösel teatmebaasid ja töö ajal oli võimalik teha otsepäring EMTS-I deklaratsiooni kinnituseks. Rakenduste vahele istutati kofigureeritav vahevara nimega "sild", mille funktsionaalsus on teisendada andmeid mõlema süsteemi jaoks vastavalt vajadusele vt joonis.



Joonis 24 Projekt 3 teostatud lahendus

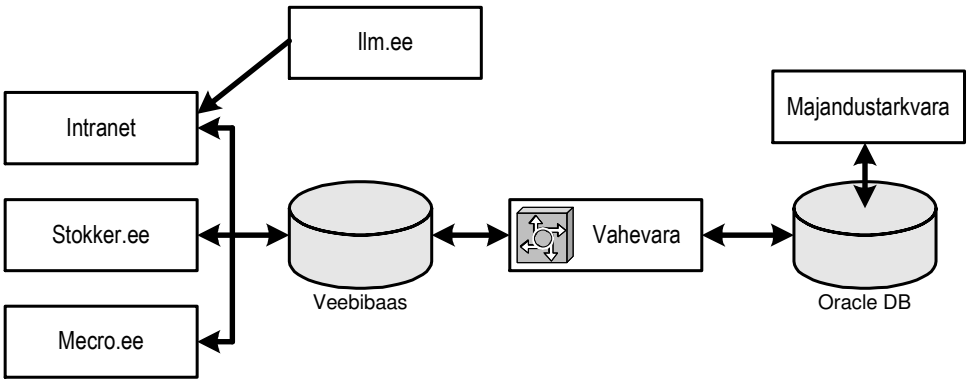
Otsepäringud liikusid EMTS-i XML sõnumitena internetis läbi turvakanali. Kui ASYCUDA-s oli esitatud uus deklaratsioon, siis esitati päring sillale, mis saadab selle edasi EMTS-le. EMTS otsib välja vastuse, saadab selle "sillale", mis teisendab selle ja saadab ASYCUDA-le tagasi. Veakoodi esinemise korral

korrigeeritakse algset päringut ASYCUDA-s päringu esitaja poolt. Kasutati andmete replikeerimist ja teenusele orienteeritud integratsiooni ning punktist-punkti topoloogiat.
Ajakava: Projekti kestvuseks oli algselt määratud ca 8 kuud, aga kokku läks 12 kuud.
Probleemid:

- Erinevatest osapooltest tulenevad probleemid. Teine osapool ei suutnud oma ülesandeid tähtaegselt lahendada. EMTS süsteemi ärioogika osa teostus venis plaanitust pikemaks. Seega ei suudetud ka vajalikku liidest pakkuda ja testimine ja sellega koos kogu projekti ajakava pikenes.
- Äriloogikast tulenev andmevahetuse keerukus. Eraldi oli vaja teostada vahevara andmete teisendamise jaoks, tõrgete töötlemine reeglid ja realisatsioon, tegevused eri tõrkekoodide korral. See kõik moodustas täiendas töömahtu, mida alguses ei olnud ette näha.

Projekt 4: Integreeritud ettevõtte infosüsteem. Selle juhtumi juures osalesin ise projektijuhina. Tööriistade müügiga tegelev ettevõte otsis võimalusi tõhustada oma müüki, otsides uusi võimalusi ja samas korraldada paremini ettevõttesisest infovahetust ja müügituge. Eesmärgi täitmiseks valiti veebipõhise ühtse keskkonna teostus, mis oleks omavahel integreeritud.

Tellija: AS Mecro
Teostaja: Mindworks Industries
Lähteülesanne: Projekti eesmärgiks oli luua integreeritud korporatiivne keskkond, mis oleks tsentraalselt hallatav. Eraldi rakendustena tuli luua jaemüügikeskkond, hulgimüügikeskkond ja intranet. Kõik need rakendused pidid liidestuma olemasoleva majandustarkvaraga.
Teostatud lahendus: Kogu firma korporatiivinfo - tootekataloog, allahindlused, kliendiinfo, tarnijad, transporditingimused ja kõik muu vajalik korporatiivinfo paiknes majandustarkvaras. Kogu selle info haldus toimus ka läbi vastava majandustarkvara liidese ja andmed salvestati ühtsesse andmebaasi. Kõik kolm rakendust teostati veebipõhistena. Liides teostati lokaalse firma andmebaasi ja veebibaasi vahel, kasutades vahevara, vt. Joonis 25.



Joonis 25 Projekt 4 teostatud lahendus

Majandustarkvara andmebaasist liikusid andmed läbi vahevara ja edasi läbi tulemüüri internetti ja mööda turvakanalit veebibaasi, mida kasutasid siis kõik kolm rakendust.

Andmeid replitseeriti öösel majandustarkvarast veebibaasi pakettidena punktist-punkti meetodil. Veebibaasist tagasi liikusid tellimused ca 15 minutilise intervalliga. Vahevara osa oli teisendada andmeid ja vahendada neid poolte vahel. Firma intraneti rakenduses kasutati ka presentatsiooni integratsiooni, presentatsioonikihti kuvati veebiteenusena saadaolev ilma info.

Ajakava: Projekt realiseeriti eraldi etappidena: intranet, hulgimüügikeskkond ja jaemüügikeskkond. Igale etapile oli planeeritud keskmiselt 4 kuud, kokku 12 kuud. Reaalselt kulus aga 16 kuud.

Probleemid:

- Puudulik andmeanalüüs. Liidese testimisel selgus hulgaliselt erinevusi analüüsis olevate tabelite kirjelduse ja andmebaasi reaalse oleku vahel. See põhjustas hulgaliselt koodiparandusi, mis teadagi pikendasid arendusprotsessi. Nõrk andmeanalüüs tulenes osaliselt tellijapoolsest valeandmete esitamistest ja arendaja kogenematusest.
- Andmete parandamine. Lisaks analüüsi vigadele avastati ka arenduse käigus andmete vigu, mis tuli parandada. See ei kujunenud siiski väga mahukaks tööks.
- Turvalisuse nõuded. Andmete liikumine toimus turvakanalis, kuid valitud turvakanal ei osutunud kuigi töökindlaks ja see tuli välja vahetada ja konfigureerida. Andmete ülekandmist tuli ka korduvalt optimeerida. Lisaks teostati monitooringusüsteem, mis jälgis andmete kohale jõudmist, vaheserveri ja veebiserveri valmisolekut.
- Muudatused äriloogikas. Arenduse käigus tuli ette ka muudatusi äriloogikas, mis samuti pikendas arendusprotsessi.

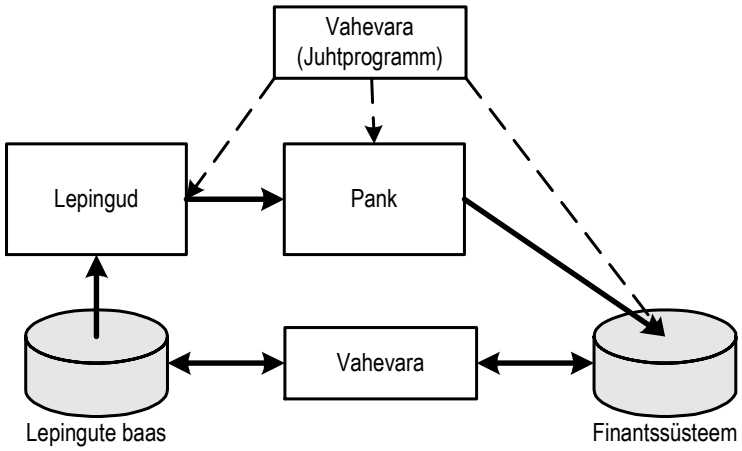
Projekt 5: Liisinglepingute haldamise infosüsteem. Panka osteti sisse uus rohkete võimalustega majandustarkvara ja äriprotsessi terviklikkuse huvides oli vaja see liidestada olemasoleva liisingu lepingute haldussüstemiga ja pangasüsteemiga.

Tellija: Hansapank

Teostaja: Hansapank

Lähteülesanne: Integreerida liisingu lepingute haldussüsteem finantssüsteemiga ja panga süsteemiga.

Teostatud lahendus: Süsteemid paikenesid kõik ühes organisatsioons, mingeid andmevahetusi väljaspool tulemüüri ei toimunud. Lepingute haldussüsteem oli panga enda arendatud ja finantssüsteemiks osteti *Oracle Financials*, mis on väga mastaapne ja paljude moodulitega terviklahendus. Finatssüsteemi ja lepingute halduse vahel toimus andmete vahetus andmebaasidee tasemel läbi vahevavara, mis tegeles andmete sünkroniseerimisega, vt. Joonis 26.



Joonis 26 Projekt 5 teostatud lahendus

Lepingute süsteemist realiseeriti ühendus pangasüsteemi ja sealt edasi jõudsid andmed taas läbi finantssüsteemi lepingutesüsteemi tagasi. Lepingutehalduse, panga- ja finantssüsteemide vahelist andmevahetust juhtis juhtprogramm e. vahevara. Kasutati andmete sünkroniseerimist süsteemide vahel ja funktsionaalset integratsiooni juhtprogrammi näol. Ühendused realiseeriti punktis-punkti meetodil.

Ajakava: Plaanitud 3 kuud, kestis ca. 6 kuud.

Probleemid:

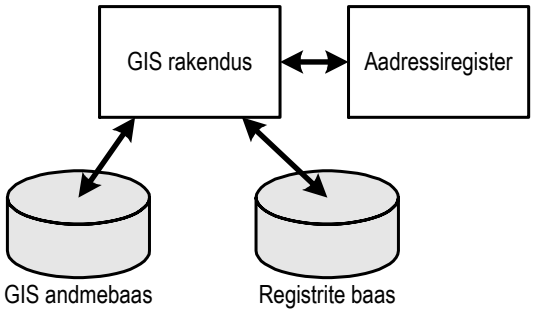
- Äriloogikast tulenev andmevahetuse keerukus. Nõudejäägi hoidmise realiserimine võttis tuduvalt kauem aega plaanitust.
- Uus tehnoloogia. Oracle teadmust ja kogemust arendajal praktiliselt ei olnud. Selleks palgati konsultant, kellest hakkas sõltuma ka kogu arenduse käik. Kohalikud arendajad omandasid uue tehnoloogia aja jooksul ja tegid hiljem ka teostatud lahendusi ümber.
- Uue töökorralduse juurutamine. Uue finantstarkvara kasutuselevõtuga kaasnes ka uus töökord ja selle juurutamine võttis aega. Integreeritud süsteemide tõttu tuli näiteks ka raamatupidajatel oma süsteemis tegeleda mingis osas lepingutega.
- Arendus- ja tootekeskkonna erinevused. Kui tarkvara paigaldati tootekeskkonda, tuli veel mitmeid vigu parandada ja siluda. Osaliselt tuleneb see andmete ja andekombinatsioonide erinevusest arenduse- ja tootekeskkonna vahel. Teine erinevus oli integreeritud süsteemi funktsionaalsuse muutumisest tootekeskkonnas.

Projekt 6: Integreeritud GIS rakendus. Ettevõte vajas erinevate objektide ja kaabelduste sõlmpunktide haldamiseks tarkvara, mis oleks seotud geograafilise infosüsteemiga (GIS) ja oskaks vastavate parameetrite järgi ka asukohti kaardil näidata.

Tellija: Eesti Telefon

Teostajad: Abobase Systems, Cell Network, BPW Consulting
Lähteülesanne: Teostada registritesüsteem ja integreerida see GIS rakendusega. Registritesüsteemi ülesanne oli kuvada geograafilist infot objektide asukoha näitamiseks kaardil ning lisada ja vaadata andmeid objektide kohta. Liidestamine pidi toimuma ka aadressiregistriga, kust saadi info objektide aadresside kohta.

Teostatud lahendus: Registrite rakendus ja GIS olid ühes organisatsioonis, mingeid väliseid päringuid ei tehtud. Registrite rakendus realiseeriti nn "paksu kliendina", kus osa funktsionaalsusest paiknes klienditöökohal. Selle vajaduse tingis graafiliste andmete kujutamine ja andmemaht. Eraldi paiknesid GIS andmebaas, registrite baas ja aadresside info. Rakendus tegi otsepäringuid erinevatesse andmebaasidesse, vt. Joonis 27. Kuna rakendusd paiknesid kõik ühes organisatsioonis, siis lahendati info kättesaamine otsepäringutena, punktist-punkti meetodil.



Joonis 27 Projekt 6 teostatud lahendus

Ajakava: Konkreetne lõpptähtaeg puudus, arendati etapi viisiliselt.
Probleemid:

- Erinevatest osapooltest tulenevad probleemid. Probleemid ei olnud arendajate vahel, vaid tellija organisatsiooni sees olevate osapoolte vahel. Raskusi oli oluliste otsuste vastuvõtmisel ja pakutud lahenduste valikul kokkuleppetele jõudmisel.
- Rakenduste ümbertegemine. Arenduse käigus selgus, et olemasolev rakendus "aadressiregister" pole olemasoleval kujul kasutatav ja tuleb otsast peale arendada.
- Andmete parandamine. Aadressiregistri andmed ei olnud ka olemasoleval kujul kasutatavad ja neid tuli täiendava tööna parandada. See ei kujunenud siiski väga pikaaegseks ja mahukaks tööks.

Tabel 1 näitab integratsiooni juhtumites esinenud probleeme projektide lõikes.

Probleem	Projekt1	Projekt2	Projekt3	Projekt4	Projekt5	Projekt6
Ülesandepüstituse ebaselgus	x					
Kooskõlastuste pikk aeg	x					
Muudatused äriloogikas	x	x		x		
Turvalisuse nõuded		x		x		
Uus tehnoloogia		x			x	
Tellija projektijuhtimine		x				
Erinevatest osapooltest tulenevad probleemid		x	x			x
Äriloogikast tulenev andmevahetuse keerukus			x		x	
Puudulik andmeanalüüs				x		
Uue töökorralduse juurutamine					x	
Arenduse ja tootekeskkonna erinevused					x	
Rakenduste ümbertegemine						x
Andmete parandamine				x		x

Tabel 1 Integratsiooni juhtumite probleemide koond

Nagu juuresolevalt tabelilt näha, siis mõned probleemid on kattuvad. Kolme projekti juures on tulnud ette keskmisest rohkem muudatusi äriloogikas ja erinevatest osapooltest põhjustatud probleeme. Korduvad probleemsed teemad on veel turvalisuse nõuded, uus tehnoloogia ja äriloogikast tulenev andmevahetuse keerukus.

Tabel 2 näitab kasutatud tehnoloogilisi lahendusi projektide lõikes.

Inetgratsiooni lahendus	Projekt1	Projekt2	Projekt3	Projekt4	Projekt5	Projekt6
Presentatsiooni integratsioon				x		
Funktsionaalne integratsioon					x	
Sõnumile orienteeritud		x				
Teenusele orienteeritud			x	x		
Hajusobjektid						
Andmete integratsioon						x
Faili ülekanne						
Ühisandmebaas	x					
Andmekoopiate ülalhoid	x		x	x	x	
Topoloogia						
Punktist-punkti	x	x	x	x	x	x
Jaotur						
Siini topoloogia						
Vahevara			x	x	x	
Turvakanal		x	x	x		

Tabel 2 Integratsiooni juhtumites kasutatud lahendused

Nagu tabelist näha, siis põhiliselt on projektides kasutatud andmete integratsiooni alla kuuluvat andmekoopiate ülalhoiu mustrit, kas replikeerimise või mõne muu

sinna alla liigituva tuletise näol. Topoloogiate kohapealt on kõik projektid teostatud punktist-punkti meetodil, kuna see on kõige lihtsam viis ja ilmselt ei olnud ka vastuolus nõudmistega. Pooltes projektides on kasutatud ka vahevara ja turvakanalit. Hallil alal äramärgitud funktsionaalse- ja andmete integratsiooni lahendused ei kvalifitseerunud päris täpselt allolevate mustrite alla ja nende kohta ei olnud võimalik hetkel täpsemat teavet saada.

3.3 Intervjuude kokkuvõte ja soovitused

Käesolevas jaotises käsitlen empiirilist materjali, tuues välja kokkuvõtte intervjuudes käsitletud teemadest. Selleks selekteerisin olulisemad osad intervjuudest, analüüsisin neid, koondasin ühte tüüpi vastused ja grupeerisin sarnased probleemid.

Oma intervjuusid alustasin reeglina küsimustega mõistete kohta: mis on integratsioon, -liidestamine, -vahevara? See tundus olevat oluline ära täpsustada, kuna kõigi jaoks ei pruugi need mõisted olla üheselt arusaadavad. Integratsiooni ja liidestamise mõistete tõlgendamisel võis täheldada kahe leeri kujunemist: üks osa mõistab integreerimise all seda, kui rakendused vahetavad üksteisega kahepoolselt infot ja funktsioone, on võimelised töötama ühtse süsteemina. Liidestamise puhul on aga tegu nõrgema seosega, näiteks üks rakendus saadab teisele andmeid ja tagasisidet ei saa. Teine osa küsitletavatest peab integreerimist ja liidestamist sünonüümideks või siis vastastikuse kokkuleppe küsimuseks ja ei tee sisulist vahet nendel terminitel.

Vahevara tähendust integratsioonis mõistetakse enamasti ühtemoodi: programm või moodul, mis aitab kommunikeeruda kahel rakendusel. Kui kirjanduses oli vahevarana kirjeldatud ka suuremaid valmislahendusi (transaktsioonitöötlus monitorid, rakendusserverid, integratsiooniserverid jne.), siis meie kontekstis käsitletakse vahevarana pigem spetsiaalselt andmevahetuse hõlbustamiseks arendatud moodulit, mis võib tegeleda andmete teisendamisega, summeerimisega, ja paljude muude tegevustega, mis võivad nõuda ka äriloogikat. Kasutatud vahevarad on üldjuhul andmebaasile orienteeritud ja tegelevad andmete sünkroniseerimise ja teisendamise korraldamisega.

Eelpool kirjeldatud juhtumite põhjal võib väita, et meie kontekstis on kõige enam kasutust leidnud andmebaaside tasemel integratsioon ja seda punktist-punkti meetodil. Seda osaliselt vajadustest liigutada suuri andmehulkasid kiiresti ja osaliselt teostuse lihtsuse- ja juba olemasoleva kogemuse pärast.

Tehnoloogiliste eelistuste küsimuses peeti oluliseks antud integratsiooni eesmärki ja konteksti ja siis sellest tulenevat tehnoloogilist valikut. Mõned küsitletavad küll eelistasid kasutada liidestamistel teenusele orienteeritud integratsiooni (näiteks veebiteenuseid), kuid enamikul juhtudel ei manitud mingeid konkreetseid tehnoloogilisi eelistusi.

Turul olevate valmislahendustega, integratsiooni toodetega, ei olnud küsitletavatel kokkupuuteid. Teati küll nende olemasolust, kuid suhtumine oli üsna skeptiline sedasorti toode kasutegurisse. Seda põhjusel, et iga integratsioon kipub olema erinev oma nõudmiste ja teostatud lahenduse poolest, ning universaalsed asjad ei pruugi neid nõudmisi lahendada ja on üleliia kallid.

Üldjuhul domineeris arvamus, et meie ettevõtetes ei mõelda suuremalt jaolt integratsiooni strateegia peale. Rakenduste arendamine on suunatud enamikjuhtudel konkreetse probleemi lahendamiseks ja seejuures ei pöörata suurt tähelepanu tehnilistele lahendustele integratsiooni seisukohast. Seetõttu oleme täna olukorras, kus mõnes suuremas ettevõttes on dubleeritud mitmete rakenduste funktsioonid ja suured andmehulgad.

Siiski mõnes suuremas organisatsioonis, on hakatud uute rakenduste arendusel arvestama funktsioonide ja andmete korduvkasutuse võimalustega ja kehtestama teatud reegleid, mis lihtsustavad tulevasi võimalikke liidestamisi. Aga eesmärki eraldi, liita kõik olemasolevad rakendused ühtseks süsteemiks ei ole esialgu veel küsitletavatele teadaolevalt organisatsioonidel plaanis lähiajal täita. Seda peetakse ebaotstarbekaks, liiga kalliks või teatud juhtudel isegi võimatuks. Küll on aga paljudel organisatsioonidel kindlad nõuded uutele rakendustele, mis tagavad nende sobivuse olemasolevasse IT keskkonda.

Arendusmetoodikana peeti ühiselt kõige sobivamaks iteratiivset lähenemist. Tarkvara funktsionaalsuse tükkideks jaotamine ja teostamine aitab ka tellijal paremini mõista oma vajadusi ja rakenduste arendust kujundada sobivas suunas. Arendajale on see lähenemine efektiivsem rakenduste arendamisel tervikuna ja integratsiooni seiukohast võimaldab varakult testida liidestamisi.

Varajast testimist peeti integratsiooni puhul vajalikuks. Oluline on testida liideseid võimalikult vara, kasvõi baastatsemel kontrollida, et ühendus üldse toimib ja andmete vahetus on võimalik. Liideste toimimisel tuleb planeerida ka käitumised veasituatsioonides, et liidese rikke tõttu ei lakkakas mõni rakendus töötamast.

Järgnevalt toon välja koondloetelu probleemidest ja eripäradest, mida intervjueeritavad mainisid eelpool käsitletud konkreetsetes juhtumites ja pidasid oluliseks rakenduste integratsioonis tervikuna. Loetelu sisaldab ka soovitusi, kuidas on võimalik probleemidele lahendusi leida.

- Mitmetest osapooltest tulenevad probleemid. Kui tegu on integratsiooniprojektiga, siis on suur võimalus, et ka projekti osapooli on mitu. See teeb keerulisemaks suhtlemise ja mõjutab projekti arengut. Tihti ei saa loota, et teine osapool on asjadest õieti arusaanud ja lahendab ka oma ülesanded tähtajaliselt. Seetõttu võib olla ohus lahenduste sobivus ja projekti ajakava. Lahendusena tuleb tagada tihe ja vigadeta kommunikatsioon osapoolte vahel, et kõigil oleks selge arusaam oma ülesannetest ja kaasnevatest riskidest. Operatiivse kommunikatsiooni korral saame vältida viivitusi ja möödarääkimisi ja sellega riske maandada.

Lisaks võivad tekkida mitmete osapoolte korral ka projekti sees erinevad osapoolte huvid. Näiteks ei suudeta olulisi otsuseid vastu võtta või töötatakse teineteisele mingil põhjusel vastu. Siinkohal on integraatorite poolsel projektijuhil väljakutse proovida oma võimeid suhte korraldajana. Vähemalt ühes eelpool käsitletud juhtumis andis see häid tulemusi.

- Ülesande püstituse ebaselgus. Võib juhtuda, et integratsioonilahenduse tellijal ei ole konkreetset vaadet, mida tahetakse saavutada. Tellijale ei saa ette heita ebakompetentsust ja tuleb arvestada sellisel juhul pikema analüüsietapiga, kus näiteks tuleb teha strateegiline analüüs vms. ja selle pinnalt kujundada konkreetne ülesandepüstitus.
- Tellija projektijuhtimine. See probleem pole kindlasti mitte ainult integratsioonile omane ja mahub üldiste probleemide alla tarkvaraarenduses. Võib tekkida olukord, kus tellija poolt puudub konkreetne otsustaja, kes kinnitab pakutud lahendused ja strateegilised otsused. Soovitav on varakult fikseerida projektis tekkivate küsimuste menetlemise kord. Vastasel juhul pole võimalik arendajal plaanitud ajakavas töötada.
- Kooskõlastuste pikk aeg. Kui tellijaks on suurem organisatsioon, siis tuleb planeerida kindlasti sisse piisav aeg kooskõlastusteks. Integratsiooni läbiviies on meil tegemist erinevate rakenduste ja andmetega, millel võivad olla erinevad omanikud. Isegi kui integratsioon toimub organisatsioonisiseselt, tuleb tihtipele arvestada erinevate osakondade ettekirjutuste ja arvamustega.
- Puudulik andmeanalüüs. Andmeanalüüs on integratsiooni läbiviimisel oluline osa. Puudulik andmeanalüüs võib saada arendust takistavaks asjaoluks ja projekti hilisemas faasis tuleb seetõttu programmi jätkuvalt ümber kirjutada ning sellega seoses ka uusi sõltuvusi ja vigu likvideerida. Need tegevused mõjutavad aga oluliselt projekti eelarvet ja ajakava. Lahendusena tuleb analüüsi etapile täiendavat tähelepanu pöörata, et tagada selle kõrge kvaliteet.
- Andmete kvaliteet. Kui andmete struktuur ja paiknemine baasides on ka heal tasemel analüüsitud, võib analüüsi käigus tekkida vajadus käivitada eraldi projekt andmekvaliteedi tagamiseks. Andmed on tihti kõikuva kvaliteediga, ei ole ühtsed või puudub piisav detailsuse aste. Pole ka harvad juhtumid, kus võidakse käivitada andmete kvaliteedi parandamise projektid, mis võivad kesta ka aastaid. Andmekvaliteedi tagamiseks saab kasutada ka automatseeritud vahendeid, mis küll ei pruugi kogu probleemi alati tervikuna lahendada. Igaljuhul andmete kvaliteedi vigade varajane avastamine jätab vajaliku aja selle probleemi lahendamiseks.
- Muudatused äriloogikas. Väiksemaid muudatusi äripoolt tuleb suure tõenäosusega ette enamikes tarkvaraprojektides. Integratsiooni teostusel võivad muudatused liidestamist oluliselt mõjutada. Kolmel käsitletud integratsiooni juhtumil oli muudatuste hulk tavapärasest suurem ja see

mõjutas ka lõpptähtaegu. Siinkohal võib soovitada tihedama äriloogika muudatuste ettenägemise korral kasutada tehnoloogiaid, mis võimaldavad muudatusi teha vähema vaevaga.

- Rakenduste ümbertegemine. Arenduse käigus võib selguda, et mõnda olemasolevat rakendust ei saa integreerida, või pole ta muul põhjusel terviksüsteemis kasutatav. Rakendus tuleb kas osaliselt muuta või tervenisti ümber kirjutada, mis võtab teadagi täiendavat lisaaega. Selliste ootamatuste vastu pole tavaliselt kaitstud ükski tarkvaraarendaja ja siinkohal saab kasutada varem kogu projekti jaoks planeeritud ootamatuste puhvrit.
- Uus tehnoloogia. Rakenduste integreerimisel võib juhtuda, et kasutada tuleb uusi tehnoloogiaid, millega arendajal puuduvad kogemused. See võtab täiendavat õppimisaega ja on riskantne projektile. Lahendada saab konsultandi palkamisega, kuid seejuures peab olema ettevaatlik, et konsultandist ei kujuneks võtmeisik, kelle lahendustest sõltub kogu projekt.
- Turvalisuse nõuded. Kui andmevahetus toimub üle interneti või mõne teise laivõrgu, tuleb täiendavat tähelepanu pöörata ka andmetekaitsele. See omakorda seab piirangud integratsiooni tehnilisele teostusele ja esitab omad nõudmised.
- Monitooring. Andmete ülekandmise õnnestumisest või mitteõnnestumisest võib sõltuda rakenduste töövõime. Seega oleks mõistlik monitoorida neid protsesse, et alati oleks selge ülevaade ja ülekandmise logist võimalik toimunud protsesse jälgida. See võimaldab vahest ka ennetada võimalikke ülekandmise probleeme.
- Äriloogikast tulenev andmevahetuse keerukus. Andmetevahetus pole tihtipeale korraldatav lihtsate vahenditega. Tulenevalt keerulisest äriloogikast ja süsteemide erinevusest tuleb näiteks kuskil hoida mingis üleminekustaatuses olevaid andmed, andmeid teisendada, töödelda tõrkeid ja käivitada uusi funktsioone teatud tõrgete puhul. Tihti juhtub nii, et sellised varjatud keerukused ilmnevad ikka arenduse käigus ja ei olnud varemalt ette planeeritud. Ei saa alati loota, et analüüsi etapis võetakse teostatav rakendus sellisel tasemel pulkadeks, kus kõik riskid tulevad välja ja on ette planeeritavad. Sedasorti probleemide lahenduse planeeriksin ma projekti "ootamatuste" puhvrisse.
- Uue töökorralduse juurutamine. Süsteemide integreerimisega võib muutuda ka töökorraldus. Näiteks tuli ühes juhtumis raamatupidajatel hakata oma süsteemis tegelema ka lepingutega. Juurutamisega tuleb arvestada ja planeerida varakult vajalikud koolitused.
- Arendus- ja tootekeskkonna erinevused. Tarkvara arendamisel ja liidestamisel võib juhtuda, et paigutades uue tarkvara tootekeskkonda on liidestatavad osad või andmed seal juba muutunud, mistõttu tarkvara ei tööta. Kuna arendatav rakendus võib olla liidestatud mitme teise

tootekeskkonnas toimiva rakendusega, on risk selle reha otsa astuda integratsiooni korral suurem. Siin aitab konfiguratsioonihaldus protsesside jälgimine – tuleb tagada, et muudatused tootekeskkonnas saaks sisse viidud ka arenduskeskkonda. Kõiki andmekombinatsioone ei pruugi arenduskeskkonnas olla võimalik kasutada andmete konfidentsiaalsuse nõuete tõttu. Mingi risk jääb alati, et tootekeskkond on erinev, kuid seda tuleb maandada niipalju kui võimalik.

- Kõrgendatud riskid. Kokkuvõtvalt juba eelmiste punktide all väljatoodule võib lisada, et integratsiooni teostuse puhul on riskid kõrgendatud. Kõik intervjueeritavad nõustusid asjaoluga, et liidestamise puhul on tegu täiendava riskiga, mis sõltuvalt liidese keerukusest ja vajalikkusest rakendusele võib olla ka prioriteediks number üks. Siin on lahenduseks vajalike liidestamiste teostuse ettevõtmine juba projekti varajases faasis, niipea kui see on võimalik. See jätab võimaluse liidestamise riske mingil määral maandada.
- Ajakavas püsimine. Tulenevalt paljudest eelmistest punktidest, kannatab probleemide lahendamisel tihti kogu projekti ajakava. Kõik eelpool käsitletud juhtumid ületasid plaanitud tähtaegu ühel või teisel põhjusel, va üks juhtum, kus lihtsalt puudus lõpptähtaeg. Seega kui plaanitakse rakenduste integratsiooni, siis tulenevalt kõrgendatud riskidest ja määramatusest tuleb arvestada projektiplaani piisav puhver. Arusaadavalt pole võimalik kõiki probleeme ette näha ja integratsiooniprojektis suure tõenäosusega neist ei pääse.

Eelnevale punktiloetelule tuleb autori hinnangul kindlasti integratsiooni teostusel tähelepanu pöörata. Tervikuna kumab intervjuudest läbi vajadus operatiivse ja üheseltmõistetava kommunikatsooni järgi kõikide osapoolte vahel ning valmisolek muudatusteks.

Intervjueeritavate vähese hulga tõttu ei saa neid tulemusi üldistada kogu Eesti konteksti. Isiklikult kaldun siiski arvama, et sarnased jooned ja lähenemised on valdaval osal Eestis läbiviidud rakenduste integratsiooni töödel.

4 KOKKUVÕTE

Käesoleva töö eesmärkideks oli koostada rakenduste integratsiooni käsitlev analüütiline ülevaade kirjandusest ning analüüsida integratsiooni kogemust Eestis, tuues välja probleemvaldkonnad ja pakkuda soovitused nende lahendamiseks.

Töö eesmärgid loen omaltpoolt täidetuks. Teostatud on asjakohase kirjanduse ja kogutud empiirilise andmestiku analüüs, välja on toodud probleemvaldkonnad ja vajalikud soovitused nende lahendamiseks.

Kirjanduse analüüsi põhjal võib öelda, et ei ole ühte kindlat integratsiooni alast käsitlust ja kindlat juhtivat autoriteeti selles vallas. Käsitlusi on mitmeid ja samuti erinevaid viise integratsiooni tüüpide ja vormide liigitamiseks - see sõltub eelkõige uurija vaatepunktist. Valdavalt käsitlevad viimase aja uuringud intergatsioonilahendusi mustritena, nagu ka paljusid teisi lahendusi tarkvaraarenduses.

Kirjanduses domineerinud üldise arvamuse kohaselt soovitatakse integratsioonilahendustes vältida punktist-punkti liidestamisi ja rakenduste jäigalt sidumist üksteisega, mida aga meie praktikas ikka kiputakse tegema. Nimelt viitab empiiriliste andmete analüüs, et meie kontekstis on kõige enam kasutust leidnud andmebaaside tasemel integratsioon ja seda punktist-punkti meetodil. Osaliselt on see tingitud vajadusest liigutada suuri andmehulkasid kiiresti ja osaliselt sobib antud lahendus paremini teostuse lihtsuse ja juba olemasoleva kogemuse pärast. Arendajate hinnangul saab tihti määravaks ka arendustööde hind ja seal on punktist-punkti meetodi kasutamisel tuntavad eelised.

Võrreldes muu maailma kogemusega tehakse meil rohkem *custom-made* ehk omatehtud ja kohandatud lahendusi ja ei osteta sisse niivõrd terviksüsteeme ja standardlahendusi. See on ka mõistetav, kuna terviksüsteemide ja integratsiooni toodete maksumus pole alati meie kaliibriga organisatsioonidele taskukohane. Seetõttu on odavam ja kättesaadavam kasutada kohalike programmeerijate oskusi ja teadmisi, seda enam kui siiamaani on seda tehtud ja paljud rakendused niikuinii nende poolt valmistatud. Seda rada valides on aga eriti oluline süsteemi tellija poolt jälgida, et rakenduste arendamisel oleks need tulevikus ka võimalikult integreeritavad.

Kohalikelt arendajatelt julgeks rohkem loota uute tehnoogiate kasutamist, kui juba läbiproovitud traditsioonilised andmebaasidest andmete sünkroniseerimised punktist-punkti meetodil. Selliseid lahendusi on kahtlemata riskivabam teha, aga tulevikus ei pruugi need organisatsioonide vajadusi enam rahuldada ja süsteem tervikuna muutub järjest keerukamaks. Mõningast tendentsi on siiski näha teenusele orienteeritud integratsiooni poole liikumisel. Vähemalt osad arendajad püüavad kasutada veebiteenuseid, kui see parasjagu võimalik on.

Loomulikult ei sõltu aga integreeritud ja hästiläbimõeldud IT arhitektuuri teostus ainult arendajast. Sageli polegi võimalik juurutada integreeritud ühtset

infosüsteemi, kuna erinevad töötavad rakendused elavad oma elu ja arenevad edasi. Oluliseks takistuseks võib saada ka tervikliku integratsiooni läbiviimise suur maksumus, mis kogu ettevõtmisele kriipsu peale tõmbab.

Kaldun arvama, et ettevõtted valivad tänasel päeval veel lahenduse, kus saab hädavajaliku rakenduse teostuse odavamalt ja kiiremini ning vaatavad mööda tulevastest võimalikest integreerimisvajadustest. Organisatsioonides peaks olema inimene, kes suudaks igapäevaste IT probleemide lahendamise kõrvalt vaadelda ka IT keskkonda, kui tulevast integreeritud süsteemi ja planeerida vastavalt sellele ka uusi arendusi.

Antud magistritöös kasutatud empiirilist materjali ei saa siiski kogu Eesti peale üldistada, kuna küsitletavate arv oli piiratud seitsme inimesega. Sellegipoolest on käsitletud intergratsiooni juhtumid ja kogemused märk meil toimuvast.

Edasist uurimistööd võib teha mitmes suunas, keskendudes integratsiooni klassifitseerimisele, tehnoloogiatele või juhtimisele. Näiteks võib uurida erinevaid projekti juhtimise metoodikaid ja kasutades parimaid praktikaid iteratiivsetest lähenemistest panna kokku sobiv metoodika integratsiooni projektide läbiviimiseks. Võimalik on täiendavalt suurendada empiiriliste andmete hulka ja analüüsida süvitsi projektide õnnestumiste ja ebaõnnestumiste tegureid. Vaadelda tuleks seejuures eraldi konkreetseid liideseid, analüüsida kasutatud tehnoloogiaid ja lahendusi. Selline materjal annaks arendajatele ja tellijatele hea ülevaate integratsiooni praktikatest ja võimalikest valikutest.

5 KASUTATUD KIRJANDUSE LOETELU

(WinterGreen Research, 2003)
Application Integration (EAI) Market: Opportunities, Strategies, and Forecasts, 2003 to 2008. (2003). WinterGreen Research, Inc. Jan 2003.
URL http://www.researchandmarkets.com/reportinfo.asp?report_id=32333

(British Computer Society, 2002)
British Computer Society (2002). Data Under Scrutiny. The Computer Bulletin, Nov, 2002. URL http://www.bcs.org.uk/publicat/ebull/nov02/integration.htm

(ComputerWire, 2002)
Integration Projects Fail Because of Lack of Strategy (2002). ComputerWire.

(Gormly,2001)
Gormly, L. (2001). EAI Overview. IT Toolbox.
URL http://eai.ittoolbox.com/documents/document.asp?i=1246

(Hall, 2002)
Hall, M. (2002) Integration: IT's Albatross. Computerworld, Jan., Vol 36, 22-24.

(Hohpe, 2003)
Hohpe, Gregor. Woolf, Bobby (2003). Enterprise integration patterns: designing, building, and deploying messaging solutions. Addison-Wesley. ISBN 0321200683.

(Thoughtworks, 2002)
How to Choose the Right Technology and the Right Partner to Develop your Integrated e-business solution. (2002). Thoughtworks.
URL http://www.thoughtworks.com/library/integrationwhitepaper.pdf

(Informatica, 2005)
Informatica.com: Technical Glossary. (2005).
URL http://www.informatica.com/solutions/resource_center/glossary/default.htm

(Inmon, 1999)
Inmon, W.H. (1999). A Brief History of Integration. Business Integration Journal.
URL http://www.bijonline.com/Article.asp?ArticleID=119&DepartmentId=5

(InformationWeek, 2002)
Integration Projects Take Aim At Partner Collaboration. (2002). InformationWeek.com, Jan., 7.
URL http://www.informationweek.com/story/IWK20020104S0006

(BCS, 1999)
IT juhtimise käsiraamat. (1999). Äripäev: käsiraamat. Tallinn: Baltic Computer Systems.

(Kaye, 2003)
Kaye, Doug. (2003). Loosely Coupled: The Missing Pieces of Web Services. RDS Press. ISBN 1881378241.

(Võõrsõnade leksikon, 1983)
Kleis, R., Silvet, J., Vääri, E. Võõrsõnade leksikon. (1983). Tallinn: Valgus.

(Linthicum, 2003)
Linthicum, D. (2003). Next generation application integration : from simple information to Web services. Addison-Wesley. ISBN 0201844567.

(Modern Practice, 2005)
Modern Practice: FindLaw's Law Practice & Technology Magazine. (2005). URL http://practice.findlaw.com/glossary.html

(Royce, 1998)
Royce, W. (1998). Software Project Management: a unified framework. Addison-Wesley. ISBN 0201309580.

(Schmelser, 2003)
Schmelser, R. (2003). The Value Proposition for Service-Oriented Integration. ZapThink research report. URL http://www.zapthink.com/report.html?id=WP-0111

(Schulte 2002)
Schulte R. (2002) Real-World Lessons for Systematic Application Integration. Gartner Group Research Article AV-15-5519.
URL http://www.gartner.com/resources/104800/104820/104820.pdf

(Trowbridge, 2004)
Trowbridge, D., Roxburgh, U., Hohpe, G., Manolescu, D. (2004). Microsoft Integration Patterns: patterns & practices. Microsoft Corporation. ISBN 073561850X.

6 SUMMARY

In today's software development we encounter the integration of different systems more and more often. Organizations are trying to combine separate applications into integral, interoperable systems that would be effective as well as easily maintained. At that, making applications implemented on different platforms and at different times to intercommunicate and exchange data is rather a complicated task How to do it wisely and advisedly, so that the resulting interfaces would be reliable and able to meet future needs.

In the author's view, projects involving the integration of various systems are more complicated both technology- as well as management-wise. As a rule, construction of a number of interfaces increases risks and indeterminacy, causing possible unexpected problems to emerge more often during the project. This paper gives an overview of integration solutions and problems, making also implementation suggestions for the integration of applications.

The first section of the paper studies the notion of integration, its reasons and the development of the market of integration products.

The second part of the paper contains the analysis of the literature, considering the available types of integration, different integration patterns and modern approaches to integration.

Third portion delves on the Estonian experience in application integration. For that purpose, seven project leaders with corresponding experience have been interviewed and the results analysed. As a result of the analysis, main problems in application integration have been outlined and recommendations for their solutions presented.

Compared to the rest of the world, instead of purchasing integration products and standard solutions, more custom solutions are used in Estonia. In the majority of enterprises, application integration is not much of an issue yet, but the need for it seems to increase continuously. Some organizations have in application design started to follow the principles according to which applications should be integrable and the necessary components and data reuseable.