Tallinna Pedagoogikaülikool

Matemaatika – Loodusteaduskond

Informaatika osakond

ERKI SAVISAAR

Infosüsteemide loomine .NET tehnoloogia abil

DIPLOMITÖÖ

Juhendaja: prof Peeter Normak

Autor: ..... „.." 2003

Juhendaja: ..... „.." 2003

Osakonna juhataja: ..... „.." 2003

Tallinn 2003

Sisukord

Sissejuhatus

Selle töö eesmärgiks on .NET tehnoloogia võimaluste analüüs ning koostada rakendus .NET tehnoloogia võimaluste proovimiseks ja tootja poolt antud lubaduste kontrollimiseks.

Tänapäeva infosüsteemid muutuvad järjest suuremaks ja keerulisemaks. Keerukama süsteemi ehitamine nõuab ka rohkem ressurssi ja aega. Infosüsteemi loomine vajab konkreetset ja süstemaatilist lähenemist organisatsiooni vajadustele ning olemasolevatele rakendustele ning koostööd erinevate tootjate vahel. Kõige olulisemateks nõueteks infosüsteemi osas on turvalisus, kiirus, integreeruvus ning sõltumatus platvormist ja seadmest.

Infosüsteemide ülesandeks on lisaks organisatsiooni sisese tööjõudluse ja -kvaliteedi parandamisele saanud ka andmevahetus ja suhtlemine partneritega.

Probleeme tänapäeva rakenduste loomisel on üsna mitmeid. Lisaks sellele, et rakendused peavad olema platvormist sõltumatud ja lihtsalt laiendatavad, peavad nad ka kiiresti valmima, töötama oodatud funktsionaalsusega ning olema kasutajasõbralikud. Erinevatest platvormidest tulenevad probleemid on aina süvenenud, kuna on lisandunud lugematul hulgal mobiilseid seadmeid, millega samuti soovitakse infosüsteemis toimetada. Vajadus luua platvormist sõltumatuid ja universaalseid rakendusi on jõudnud faasi, kus kõik rakendused püütakse teha veebi põhised. Enamkasutatavate veebi tehnoloogiate puuduseks on aga piiratud funktsionaalsus, pikad arendusajad, kiirus ja turvalisus. Need puudused ei ole tingitud mitte programmeerijast või arendajast vaid tehnoloogiast enesest, kuna need tehnoloogiad on loodud ajal, mil ei olnud oluline suur turvalisus, keeruline funktsionaalsus ja kiire arendus.

Kuna töömahud ja keerukus on kasvanud, on muutunud hädavajalikuks korduvkasutatavate komponentide ja moodulite loomine ning meeskonnatöö. Korduvkasutusega komponendid võimaldavad vähendada rakenduse kirjutamisele kuluvat aega ning vigade arvu. Meeskonnatöö on vajalik üsna mitmel põhjusel:

- Süsteemid on liiga mahukad, et üks inimene sellega arvestatava ajaga valmis saaks.
- Tellijad vajavad kindlust, et ka aasta pärast oleks kellegi poole abi saamiseks pöörduda. Oluline ka arendamise järjepidevus.

- Moodsad infosüsteemid ei piirdu ainult ühe organisatsiooniga vaid laienevad mitmetele organisatsioonidele.

Kirjanduse andmetel ei lõpe enamus tarkvara projekte nii nagu tellijad või tegijad seda sooviksid. Täielikult õnnestub umbes 10% kõigist loodavatest hajussüsteemidest ja täielikult ebaõnnestub umbes sama palju.

Lahenduseks nendele probleemidele peetakse .NET platvormi ja tehnoloogiat. Käesolev töö annab ülevaate .NET olemusest ning erinevustest olemasolevate tehnoloogiatega, püüab ühtlasi leida vastust küsimusele, kas .NET on tõesti midagi sellist, mis aitab teha veebipõhiseid infosüsteeme kiiremini ja paremini kui varem.

See töö on mõeldud kõigile, keda huvitab: mis asi on .NET ja miks sellest nii palju räägitakse. Töö koosneb järgnevatest osadest:

- Esimeses osas antakse ülevaade rakenduse loomise protsessist ning võrreldakse erinevaid veebirakenduste loomise tehnoloogiaid, püüdes välja selgitada, mis on need kitsaskohad, miks ei saa nende abil luua moodsaid veebi lahendusi ja miks peetakse olemasolevate tehnikate arendamise asemel lihtsamaks uue tehnoloogia loomist.
- Teine osa tutvustab .NET tehnoloogiat, selgitades miks kaasneb lühendiga .NET väide „.NET on tehnoloogia uue põlvkonna veebi rakenduste loomiseks”. Ühtlasi tuuakse välja ka .NET tehnoloogia eelised võrreldes teiste tehnoloogiatega.
- Kolmandas osas tehakse läbi ühe reaalse rakenduse loomise protsess, kasutades .NET tehnoloogiat, selleks et kinnitada teises osas toodud väiteid

1 Rakenduse loomise protsess

Käesoleval ajal tomub kiire üleminek personaalarvutikesksetelt rakendustel ja süsteemidelt võrgukesksetele, vallutades kohtvõrke (LAN), asutuse sisevõrke (intranet), asutustevahelisi võrke (extranet) ja üldkasutatavaid laivõrke (internet). Mitmed tarkvara tootvad firmad loovad tooteid, mis arvestavad võrgu, suhtlemise ja aredustöö iseärasusi.

Seega on tänapäeva rakendused väljunud organisatsiooni kaitsvate seinte vahelt ning liikunud suurde ja ebaturvalisse Internetti. Liikumise põhjus seisneb üleüldises globaliseerumises ja vajaduses suhelda oma partneritega ning olla platvormist sõltumatu. Seega rääkides rakendustest, ei mõelda tavaliselt enam rakendusi mitte Windows, Linux jne. keskkonnale, vaid rakendusi mida saab kasutada üle Interneti või Internetis. Suurimad probleemid Internetis on ebaturvalisus ja ebastabiilsus. Olles Internetis pole kunagi teada, kes ja millisel viisil püüab infosüsteemi tööd halvata. Interneti üheks tugevuseks on hajusus, see tagab süsteemi töö ka olukordades, kus mõni osa süsteemist ei tööta. Hajusus võib aga tekitada suuri probleeme rakenduste ehitamisel, kuna pole kunagi teada, millist teed mööda ja kuidas informatsioon liigub. See tähendab, et saates näiteks kirja üle Interneti ühest arvutist teise jagatakse see kiri pisikesteks osadeks, ning iga osa võib liikuda erinevat teed pidi, pole kindlustatud, et kõik teele saadetud pakid ka samas järjekorras kohale jõuavad. Kõik need Interneti ehtitusest tulenevad iseärasused ja ohud muudavad Internetist sõltumise veelgi raskemaks.

Jättes kõrvale Internetti moodustava riisvara võib Internetti vaadelda kui teenuste kogumikku, kus andmete transportimiseks ühest punktist teise kasutatakse TCP/IP protokolli ja levinumateks teenusteks e. teenusteks mis jõuavad kõikjale on veebi leheküljed (HTTP) ja E-kirjad (SMTP). Luues rakendusi Interneti jaoks on kõige mõistlikum kasutada just neid levinud ja standardseid teenuseid ja protokolle kogu vajaliku funktsionaalsuse loomiseks. Tulenevalt Interneti füüsilisest lahendusest ei tohi unustada, et kunagi pole teada kuidas üks või teine saadetis ühest punktist teise jõab ja Interneti ühendus võib igal hetkel katkeda. Katkestus võib olla väga väike kuid on siiski tuleb sellega arvestada.

Kuna inimesed on juba harjunud vaatama ja kasutama veebi lehti taandub kogu veebirakenduse loomise keerukus sellele, kui lihtsalt ja kiiresti suudetakse

tekitada rakendusele vajalikud HTML lehküljed. Probleemiks HTML lehekülgede loomise juures on HTML lehekülgede loomise ideoloogia. HTML on standard staatiliste lehekülgede kirjeldamiseks aga sisu, mida me soovime neil lehtedel kajastada on sageli dünaamiline!

Kirjaduse andmetel [20] on hajussüsteemide projektide täieluku läbikukkumise protsent maailmas ligikaudu 10 ja täieliku õnnestumise protsent on sama. Hajussüsteemide arendused on kallid ja aeganõudvad. Neid süsteeme arendatakse sageli aastaid, kulutades miljoneid kroone. Samas on kirjanduses väiteid, et taolisi, võib-olla tuhandeid inimesi hõlmava arenduse saab teha nt kuue kuuga. [8]

Tekkib küsimus – milles on probleem? Kas programmeerijad ei oska kirjutada või tellijad tellida? Miks ei õnnestu kirjutada rakendus, mis vastaksid ootustele?

Laias laastus on probleeme kolm:

- Tegijad ei selgita välja mida tellija soovib e. esmalt tehakse rakendus ja alles seejärel vaadatakse millise probleemi see lahendab
- HTML on standard staatiliste lehtede koostamiseks ja ei võimalda dünaamilise sisu kasutamist ning objekt orienteeritud lähenemist.
- Internet on alati olemas, kuid mis seal toimub pole kunagi täpselt teada.

Järgnevalt neist probleemidest pikemalt.

1.1 Süsteemi analüüs

Rakenduse loomise protsessist vaatleme protsessi üsna ülevaatlikutl, et oleks aru saadav, kus tulevad mängu erinevad veebi loomise tehnoloogijad ja kui suur on nende mõju kogu protsessile.

Rakenduse loomine koosneb kuuest etapist [8]. Selleks, et klient saaks sellise rakenduse nagu ta soovib, tuleb kõik need etapid läbi käia. Infosüsteemide arendamise kuus etappi on:

- Strateegiline analüüs – olemasolevate süsteemide analüüs ja kaardistamine
- Analüüs – vajaduste kaardistamine
- Disain – uue süsteemi disainimene
- Ehitus – rakenduse loomine
- Rakendus – rakenduse juurutamine

- Hooldus – järelhooldus

Infosüsteemide mahukuse ja nõudmiste kiire muutumise tõttu kasutatakse kaasaegsetes arendusmetoodikates enamasti nende etappide tsüklilist kordust.

Infosüsteemi arendamisel on üks raskemaid ülesaineid süsteemi määratlemine. Kerkib probleem – mida süsteemi määratlemisel üldse teha tuleb? Mida ja kuidas määratleda? Kuidas sellest saab tarkvara ja töötav arvutisüsteem?

Infosüsteemi määratlemise ülesannet lahendatakse süsteemianalüüsi abil. Määratlemine on seotud reaalsete süsteemide modelleerimisega ja tomitakse kontkestis „mida selles ettevõttes või organisatsioonis arvutiga teha". Hajussüsteemi puhul on oluline ka süsteemi erinevate osade paiknemine ehk terviksüsteemi loomine. [8]

Süsteemi tegelikul määratlemisel kaasneb sellega hulk teisi tegevusi. Olulisemad on intervіueerimine, tellija dokumentide uurimine ja tulemuste dokumenteerimine.

Enamus probleeme rakenduste mitte ootuspärases tegevuses peitub just poolikus disainis mitte aga programmerija oskamatuses või suutmatuses. Väiksemate ja keskmiste infosüsteemide disain käib tihtipeale jooksvalt töö käigus. See tähendab aga seda, et tihtipeale ei mõisteta probleemi olemust – minnakse parandama „aia auku" teadmata millisele aiale see kuulub ja kui suur see „auk" tegelikult on. Tulemuseks ongi rakendus, mis ei vasta päris sellele mida loodeti ja millele on tagant järele vaja kasutusala välja mõelda.

Tootjad, kes oma klientidest hoolivad ja soovivad neile pakkuda rakendusi, mis oleksid sellised nagu kliendid soovivad, pööravad väga palju tähelepanu ka süsteemi analüüsi ja rakenduse disaini faasidele.

1.1.1 Strateegiline analüüs

Rakenduse loomise esimese faasi ülesandeks on välja selgitada, millised protsessid juba tomivad ning mis on nende ülesanne. Seda sammu lihtsustaks oluliselt dokumentatsiooni olemasolu. Kui eelenvad infosüsteemid on korralikult loodud ja dokumenteeritud piirdub see faas vaid dokumentatsiooni läbitöötamisega. Kui dokumentatsioon olemasolevate süsteemi osade kohta puudub on selle sammu ülesandeks selle dokumentatsioon loomine.

Selle faasi lõpuks peavad selguma kogu infosüsteemi strateegilised ja poliitilised suunad, millest lähtutakse edasises analüüsis [8].

1.1.2 Analüüs

Analüüsi faasis tegeldakse kliendi vajaduste kaardistamisega. Analüüsi faasi lõpuks tuleb leida vastused küsimustele [8]:

- Mida klient soovib?
- Mis olemasolevatest funktsioonidest kliendile meeldib? Mida soovib jätta?
- Mis olemasolevatest funktsioonidest kliendile ei meeldi? Mida kõrvaldada/muuta?
- Kuidas uus süsteem peaks sulanduma olemasolevatesse süsteemidesse?
    - Millised tegevused/protsessid asendab?
    - Millised teveused/protsessid lisab?
- Kui palju uus süsteem võib ja hakkab olemasolevaid süsteeme mõjutada.

Analüüsi faasi on väga oluline tervikpildi loomiseks. Tervikpilt on vajalik selleks, et mõista kogu süsteemi keerukust ja toimimist. Analüüsi faasis tehtud eksimuste hilisem parandamine on väga kulukas kuna need vead sealguvad tihtipeale siis kui ka süsteemi disain ja ehitus on tehtud.

1.1.3 Disain

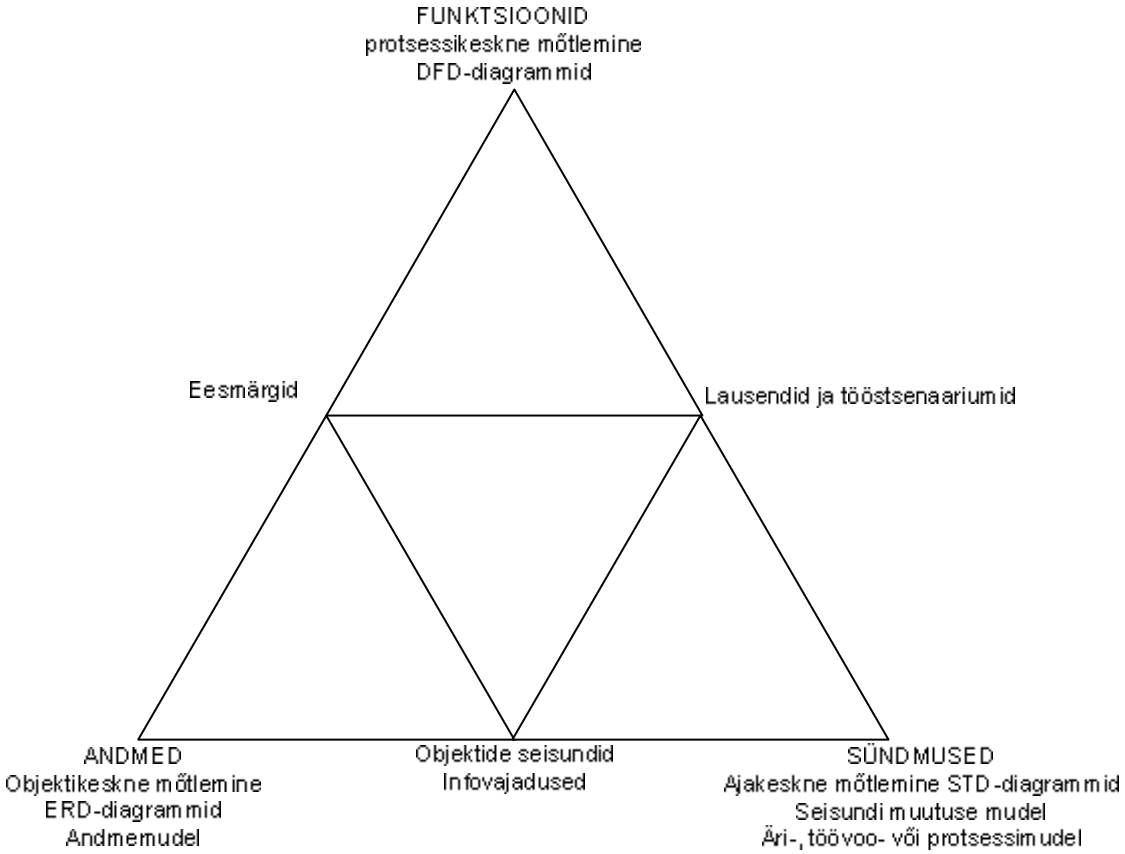
Disaini faas on rakenduse eduka loomise seisukohalt üks olulisemaid. Disaini põhjal hakatakse rakendust looma ja kui siin on tehtud viga siis on seda hiljem väga raske parandada.

Disaini faasis:

- Määratakse ära konkreetsed tehnilised lahendused süsteemi loomiseks.
- Koostatakse süüsteemile vajaminevad objektid
- Määratakse objektide vajelised vahelised seosed.
- Valmivad andmebaaside ning protsesside skeemid.

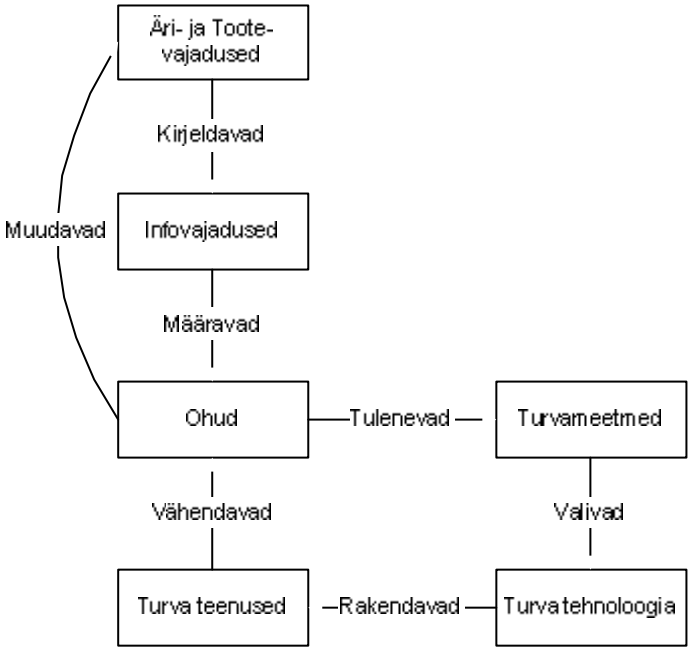
Kuna enamasti sisaldavad infosüsteemid privaatseid või salajasi andmeid tuleb mõelda ka nende andmete kaitsmisele ehk süsteemi disain peab olema turvaline.

Süsteemi modelleerimise on mitmeid võtteid, kuid üks lihtsamaid neist on „kolme mütlemisviisi keskne modelleerimine” [8]



- Tihti lisatakse turvalisus veebrakendustele alles peale rakenduse valmimist. Selline lähenemine teeb turvalisuse rakendamise veel keerulisemaks kui see tegelikult on. Juba töötavale süsteemile uue komponendi lisamine on oluliselt keerulisem kui algusest peale selle komponendi kaasamine.

Selleks, et rakendust turvaliselt disainida tuleks järgida järgnevat skeemi



1.1.5 Rakendus

Rakendamise faasis võetakse loodud rakendus reaalses elus kasutusele. Vaadeldakse, mis juhtub siis kui rakendus väljub test keskkonnast reaalsesse ellu ning kontrollitakse, kas kõik funktsioonid tomivad nii nagu sai disainitud [8].

See on faas kus süsteemi igapäevased kasutajad saavad seda süsteemi proovida.

1.1.6 Hooldus

Hooldus faasi põhiülesanneteks on [8]:

- Kasutajate aitamine süsteemi kasutamisel
- Kasutajate koolitamine
- Pisivigade kõrvaldamine
- Turvaaukude parandamine

Süsteemi hooldust käsitletakse ja rahastatakse arendusest eraldi. Sõltuvalt tootjatest võib olla järelhooldus nii tasuline kui ka tasuta ja vajada eraldi teenuslepinguid.

1.2 Veebipõhise rakenduse loomise vahendid

Vaadeldes süsteemi loomise erinevaid etappe võib väita, et eduka rakenduse loomise võti peitub oskuslikus disainis, mis baseerub oskuslikul nõute analüüsil. Programmerijate ülesandeks on vaid disainitud lahendus koodiks valada. Disaini faasis otsustatakse ära ka milliseid tehnoloogiaid edaspidi kasutatakse. Igal tehnoloogial on omad head ja halvad omadused.

Järgnevalt ülevaade veebi rakenduse loomise erinevatest vahenditest ja tehnoloogiatest.

1.2.1 Probleemid Internetis töötava programmi kirjutamisel

Internetis töötava rakenduse kirjutamisel on kolm suurt probleemi:

- Turvalisus

  Alati peab arvestama võimalusega, et ledub mõni „pahalane” kes soovib teie rakendust panna tööle nii nagu ta ei peaks töötama või teha sellega operatsioone, mida ei peaks tegema.

- Internet

  Internet on oma olemuselt kontrollimatu ja juhuslik. Võib esineda katkestusi või ka häireid, mida pole võimalik ise kontrollida
  Näiteks: kirjutades kaks kirja ja saates nad korraga ära (mõlemad kirjad lähevad ühele adressaadile), pole võimalik öelda kumb neist kahest kirjast jõuab kohale esimesena.

- HTML

  HTML on standard/keel staatiliste lehtede kirjeldamiseks. Tänapäeva elu aga nõuab, et lehtedel olev info muutub pidevalt.

Töötades Internetis tuleb arvestada turvariskidega ja ennast nende vastu kaitsta kasutades viirusetõrje tarkvara, tulemüüre, turvalisi protokolle jne ning disainides rakendused turvaliseks kohe alguses. Lisaks sellele käib turvalise süsteemi juurde ka plaab B ehk mida teha siis kui ikkagi midagi läheb halvasti nt kuidas, kui kiiresti ja mis seisuga saab taastada admed varukoopiast.

Interneti juhuslikkus on paratamattus ja sellega tuleb lihtsalt arestada. Kuna tegemist on meist sõltumatute asjaoludega siis ei saa nende vähendamiseks midagi ette võtta. Võib mõelda välja vaid meetodeid, mis juhuslikkusest tulevaid hädasid vähendavad.

Probleem, mida tuleks osavalt lahendada on HTMLi staatilisus. Kuidas saada staatiline HTML dünaamiliselt muutuma või mõelda välja mõni uus standard, mis on oma olemuselt on dünaamiline nagu nt. Windowsi vormid.

HTMLi asendamine millegi intelligentsemaga on agakeeruline, kuna kõik on selle keelega juba harjunud ja oskavad seda lugeda.´

HTML (lühend sõnades HyperText Markup Language). [1]

- Avaldada dokumente, mis sisaldavad pealkirju, teksti, tabeleid, fotosid jne.
- Hüperlinkide abil tuua infot ühe nupuvajutusega
- Koostada vorme suhlemiseks serveritega, mille abil on võimalik reserveerida ruume, osta tooteid ja teenused, otsida informatsiooni jne.
- Panna ühele dokumendile nii tabeleid, videosid, muusika klippe ja teisi rakendusi ühes dokumendis

HTML keele dünaamiliseks tegemiseks on mõeldud välja hulk programmeerimis ja skriptimis keeli. Kõik need keeled saab jagada kaheks: keeled, mis loovad

dünaamika serveris ja keeled, mis teevad seda kliendi juures veebi sirvijas (Browser).

Serveris kasutatavad vahendid on kliendi suhtes mugavamad, kuna ei vaja kliendi pool mingeid eriseadistusi. Seadistada tuleb vaid veebi serveeriv server. Serveri poolsete keelte tööpõhimõte seisneb selles, et kliendi päringu peale käivitatakse programm, mis koostab kliendile täesti standardse HTML dokumendi. Puudused on aga suur koormus serverile ja aeglasem reageerimine kliendi nupuvajutustele kuna iga liigutuse jaoks tuleb tulla veebi serverisse uurima, mida edasi teha.

Kliendi juures kasutatavad skritpimis keeled täidavad oma ülesanded veebi sirvijas. Seega peab igal külastajal olema sirvija, mis mõistab seda skriptimis keel. Kliendi poolsete skriptimis keelte eeliseks on töö kiirus ja väiksem koordmus veebi serverile. Miinusteks on aga ebaturvalisus (rakenduse kood on avalik), ebakindusl (mis saab klientidest, kelle veebi sirvija ei mõista kasutatud skriptimise keel).

1.2.2 Tehnoloogiad

Järgnevalt vaatleme pisut lähemalt levinumaid tehnoloogiaid, mida tänapäeval veebi loomise juures kasutatakse.

DHTML

Dynamic HTML annab kontrolli standardsete HTMLi elementide üle. See annab võimaluse muuta lehti ka peale sirvijasse laadimist. Lehe iga osa on võimalik muuta dünaamiliselt ükskõik millisel ajal.

DHTML funktsionaalsust saab anda kasutadaes JScripti, JavaScripti ja teisi kliendi pool töötavaid keeli. [5]

Kuna tegemist on kliendi poolel töötavate skriptidega siis on nad (kliendi poolt vaadates) väga kiired kuna ei vaja serveri poolseid instruktsioone kuid sõltuvad väga palju veebi sirvijast ja adminsitraatorist. Kõik veebi sirvijad ei toeta DHTMLi ning turvakaalutlustel on paljud kasutajad DHTML toe oma veebi sirvijatest välja lülitanud. Seega ei saa olla kindel ei kõik nõnda loodud dünaamilised tegevused kõigi klientide juures toimiksid. DHTML funktsionaalsus on ette määratud veebi sirvija looja poolt.

CSS ja XSL

Cascading Style Sheets (CSS) on keel omadustest, mis kirjeldavad HTML tagide väljanägemise. Tegemist on lihtsalt keelega, mis võimaldab lihtate vahenditega kontrollida ühe lehe aga ka terve veebi stiili. [1]

CSS võimaldab lehe välimuse kirjeldada eraldi ja sõltumatult tegelikust lehest.

CSS on integreeritud pea kõikidesse veebi sirvijatesse ja W3C (Wold Wide Web consortium) on stiilifailide kasutamist soovitanud alates oma loomisest 1994 aastal.

Lisaks CSS standardile on stiilide kirjeldamiseks ka XSL standard. Põhjustele, mis neid keeli on kaks heidab valgust järgnev tabel [1]:

	CSS	**XSL**
Saab kasutada HTML dokumendis?	Jah	Ei
Saab kasutada XML dokumendis?	Jah	Jah
Transformeerimis keel?	Ei	Jah
Süntaks	CSS	XML

Tegemist pole mitte programmeerimise keeltega vaid vahendiga lihtsalt ja kiirest ning keskselt kujundada veebi lehekülgi.

VBScript

Microsofti Visual Basic Skript võimaldab kirjutada skripte mitmele poole seal hulgas ka Microsoft Internet Explorer (IE) brauseritele ja Microsoft Internet Information Serverile (IIS). [9]

VBScript on lihtsalt omandatav. Kui kasutaja on varem puutunud kokku programmide kirjutamisega Visual Basic for Applications keskkonnas (VBA – nt MS Excel'i jaMS Word'i makrod) programmerimisega siis pole keeruline ka VBScript'i kirjutamine ning vastupidi. Tuleb endale vaid selgeks teha veebis kasutatavad objaktid.

VBScripti suurimad puudused on:

- Saab kasutada vaid eelnevalt koostatud objekte
- Ei võimalda objektorienteeritud lähenemist rakenduse koostamisele

- Tegemist on interpeteeritava keelega
- Kleindi poolse interpreteeringu peab tegema veebi sirvija, kuid kõik veebi sirvijad ei saa VBScriptist aru
- VBScripti interpreteerimine on võimalik intelligentsemates veebi sirvijates välja lülitada

ASP

Microsoft Active Server Pages (ASP) on server poolne skriptimise keel, mida saab kasutada dünaamilise ja interaktiivse veebirakenduse loomiseks. ASP'iga saab kombineerida HTML lehekülgi, skripimis käske ja COM komponente. ASP on interpreteeritav keel. Serveritest toetab ASP lehekülgi vaid MS IIS server. ASP lehti töötleb server ja tagastab kliendile standardse HTML dokumendi [10]

ASP lehed toetavad ODBC andmebaase ja võimaldavad dünaamiliselt koostada ka Exceli tabeleid, PDF dokumente jne.

ASPi suurimad puudused on:

- Saab kasutada vaid eelnevalt koostatud objekte
- Ei võimalda objektorienteeritud lähenemist rakenduse koostamisele. (väike kergendus sellele probleemile on COM komponentide kasutamine)
- Tegemist on interpeteeritava keelega
- Ei võimalda hoida koodi kujundusest eraldada
- Ei suuda hoida sessiooni veebi farmides

PHP

PHP on serveri poolne skriptimis keel. Väga sarnane ASP'ile. Nagu kas ASP lehti töötleb ka PHP lehti veebi server. Peale töötlust tagastab veebi server standardse HTML dokumendi. PHP toetab mitmeid andmebaase seal hulgas Informix, Oracle Sybase ja ka ODBC. PHP's on võimalik kasutada ka standardseid interneti tehnoloogiaid nagu autentimine, XML, dünaamiline pildi loomine, jne. [5]

PHP toetus on olemas mitmetel platvormidel seal hulgas Windows, Linux, Unix.

PHP suurimad puudused on:

- Saab kasutada vaid eelnevalt koostatud objekte
- Ei võimalda objektorienteeritud lähenemist rakenduse koostamisele

- Tegemist on interpeteeritava keelega
- Ei võimalda hoida koodi kujundusest eraldada
- Ei suuda hoida sessiooni veebi farmides

.NET

Microsoft .NET on platvorm järgmise põlvkonna interneti tehnoloogiatele.

Microsoft .NET võimaldab igal arendajal ja organisatsioonil tulu lõigata tervelt hulgalt uutelt interneti seadmetelt ja programmeeritavatelt veebi teenustelt, mis iseloomustavad ühtlasi ka Interneti järgmist põlvkonda. [5]

Võttes kokku parima tööst arvutiga, kommunikatsioonist ning baseerudes Interneti standarditel nagu XML ja SOAP aitab .NET Internetil muutuda HTML'i põhistest lehtedest XML'il baseeruvaks informatsiooniks.

.Net annab organisatsiioonidele väga võimsa vahendi keerulisete lahenduste loomiseks ning kasutajatele lihtsa ja turvalise võimaluse Internetis „surfamiseks".

.NET suurimad puudused on:

- Puudub tomiv .NET platvorm Linuxile
- Ei saa kasutada misioonikriitilistes rakendustes

J2EE

Java on Sun Microsystems, Inc poolt loodud objekt-orienteeritud programmeerimis platvorm. Java programmid – kutsutakse ka „applet" – annavad veebile funtksionaalsuse, mis puhta HTMLiga pole võimalik. Näiteks saab muuta hiire kursori kuju, kui see liigub üle nupu või klikitakse kusagil lehel.

Java applet'e saab trantsportida üle võrgu ning ta töötab igas süsteemis, kuhu on installeeritud Java Virtual Machine. Sellele vaatamata võib appletite jooksutamine osutuda mitmetele veebi sirvijatele üsna keeruliseks. [2]

IE ja Netscape kasutajatel on võimalik appletite käivitamine ka keelata.

Hetkel toetatakse järgnevaid platvorme Windows 2000/XP, Solaris, Linux

Oma ehituselt sarnaneb J2EE üsna palju .NET platvormile. Olulisim erinevus seisneb selles, et .NET ei nõua kliendilt midagi peale arusaamise HTML'ist.

J2EE suurimad puudused on:

- Vajab klient arvutite eraldi konfigureerimist (Java Virutal Machine instaleeerimist)
- Ei suuda hoida sessiooni veebi farmides

JavaScript

JavaScript on objektorienteeritud skriptimis keel. Sisaldab skriptimise võimalusi nii kliendi kui ka serveri poolel. JavaSkript võimaldab lihtsalt reageerida sündmustele nagu hiire liigutamine üle objekti, klik kusagil lehel jne. JavaScript on mõldud HTML lehtedele dünaamika lisamiseks. [11]

JavaScript ja Java on kaks täiesti erinevat asja ja peale nime ning sarnase süntaksi pole neil midagi ühist.

IE ja Netscape kasutajad saavad JavaScripti käivitamise ka keelata

VBScripti suurimad puudused on:

- Saab kasutada vaid eelnevalt koostatud objekte
- Ei võimalda objektorienteeritud lähenemist rakenduse koostamisele
- Tegemist on interpeteeritava keelega
- Kleindi poolse interpreteeringu peab tegema veebi sirvija, kuid kõik veebi sirvijad ei saa JavaScript'ist aru
- VBScripti interpreteerimine on võimalik intelligentsemates veebi sirvijates välja lülitada
- Ei suuda hoida sessiooni veebi farmides

JScript

JScript on Microsofti interpretatsioon ECMA 262 standardist. Tegemist on täiesti standardse lahendusega, kuhu on lisatud lisafunktsionaalsus Microsfti objektide kasutamiseks.

JScript on täiesti omaette keel ja pole ühegi teise keele mugandus. Javaga seob JScripti vaid sarnane süntaks.

JScript on limiteeritud keel ja saab eksiteerida vaid interpreteerivas keskkonnas nagu ASP, IE või Windows Scriptig Host. JScriptis ei saa kirjutada iseseisvaid rakendusi. Samuti puudub igasugune võimalus failide avamiseks või lugemiseks. [12]

JScripti suurimad puudused on:

- Saab kasutada vaid eelnevalt koostatud objekte
- Ei võimalda objektorienteeritud lähenemist rakenduse koostamisele
- Tegemist on interpeteeritava keelega
- Kleindi poolse interpreteeringu peab tegema veebi sirvija, kuid kõik veebi sirvijad ei saa JScriptist aru
- VBScripti interpreteerimine on võimalik intelligentsemates veebi sirvijates välja lülitada
- Ei suuda hoida sessiooni veebi farmides

Perl

Perl on interpreteeritav kõrgtaseme keel. Enamus CGI programme on kirjutatud Perlis. Perli kasutatakse palju ka prototüüpide koostamiseks ja „liimina" süsteemide ühendamisel. [5]

Kuigi perli juured on pärit Unix'ist leidub perli tugi paljudel platvormidel. Kuna Perl on interpreteeritav keel on Perl'is kirjutatud rakenduste tratsport platvormide vahel väga lihtne.

Perl suurimad puudused on:

- Saab kasutada vaid eelnevalt koostatud objekte
- Ei võimalda objektorienteeritud lähenemist rakenduse koostamisele
- Tegemist on interpeteeritava keelega
- Ei suuda hoida sessiooni veebi farmides

CGI

Common Gateway Interface (CGI) on standard näo ehitamiseks välistele süsteemidele nagu HTTP või veebi server.

CGI käivitiatakse reaalajas, seega võimaldab ta kuvada dünaamilist informatsiooni

Kuna CGI on käivituv programm on see sisuliselt sama kui lasta maailmal oma veebi sereris programme jooksutada – mis pole aga sugugikõige turvalisem tegevus. Seega kaasnevad CGI kasutamisega mõningad turvaprobleemid. Selleks, et veebi server saaks aru, et CGI tuleb käivitada mitte kuvada, tuleb CGI

programm panna eraldi kausta. On ka teisi võimalusi CGI lubamiseks ja sõltub põhiliselt veebi administraatorist. [5]

CGI programme saab kirjutada mitmetes keeltes, nagu näiteks C/C++, PERL, Visual Basic, AppleScript. Sõltub lihtsalt sellest, mis on sinu süsteemis saadaval.

Kui on tegemist nt C's kirjutatud programmiga siis tuleb see eelnevalt ka kompileerida, kui aga kasutad interpreteeritavaid keeli nagu nt PERL peab programmikastas olema kättesaadav vaid skript ise.

CGI suurimad puudused on:

- Turvaprobleemid käivituvate komponentidega
- Ei suuda hoida sessiooni veebi farmides
- Muud omadused sõltuvad kasutatavast keelest

XML

Extensible Markup Language (XML) on väga lihtne ja paindlik tekstifaili formaat, mis on tuletatud SGML standardist. XML on mõeldud rakenduste vaheliseks andmevahetuseks ning pole mõeldud inimesele lugemiseks [1].

2 .NET tehnoloogia

Mis asi on .NET? Lihtne vastus oleks .NET on kõik st. .NET on kaubamärtk. Tooted, mida me siinkohal .NET nime all vaatleme on .NET raamistik (framework), XML veebi teenused (XML web services) ja ASP.NET.

Vaadeldes esimeses osas käsitletud infosüsteemi arendamise võtteid ja veebirakenduste loomise tehnoloogiaid võib väita, et rakenduse loomise protsessis pole probleemi (tuleb vaid kliendile seletada, mis mille jaoks ja miks), probleem on hoopis selles, et pole tehnoloogiat, mis võimaldaks selle protsessi abil lihtsalt ja kiiresti rakendust ehitada. Microsoft väidab, et nüüd on selline tehnoloogia olemas ja selle nimeks on .NET.

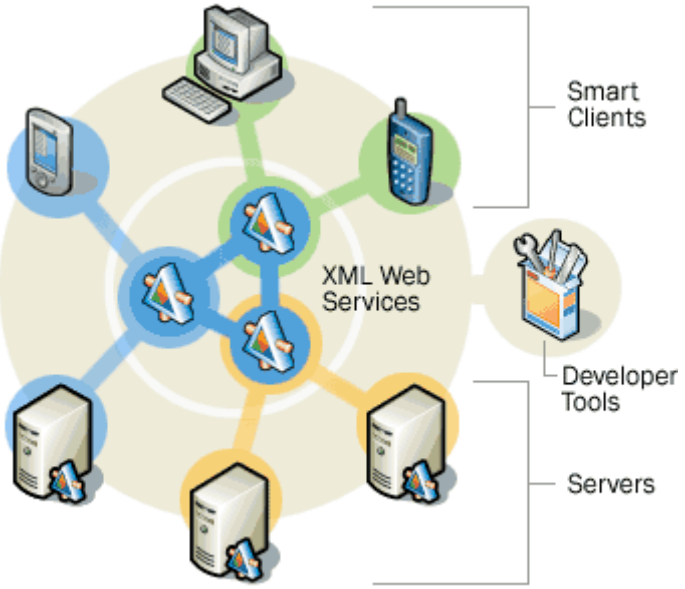
.NET on tehnoloogia, mis mõeldud infosüsteemide loomiseks ning disainitud arvestades tänapäeva maailma infotehnoloogilist olukorda ja nõuded infosüsteemidele. .NET aluseks on raamistik „.NET Framework". Sellele raamistikule baseeruvad kõik muud tehnikad: veebi lehed, Windows'i vormid, veebi teenused jne.

2.1 .NET Framework

Microsoft .NET raamistik on tarkvara, mis ühendab inimesi, süsteeme ja seadmeid. See hõlmab kliente, servereid ja arendusvahendeid ning sisaldab [16]:

- .NET framework 1.1 kasutatakse igasusuguse tarkvara ehitamiseks ja käivitamiseks, seal hulgas veebi rakendused, intelligentsed klientrakendused, XML veebi teenused – komponendid mis lihtsustavad integreerimist jagades andmeid ja funktsionaalsust üle võrgu kasutades standardseid platvormist sõltumatuid protokolle nagu XML, SOAP ja HTTP
- Vahedid arendajatele nagu Microsoft Visual Studio .NET 2003, mis sisaldab integreeritud arenduskeskkonda, et arendaja saaks maksimaalselt kasutada kogu .NET raamistiku pakutavaid võimalusi
- Komplekt teenuseid, seal hulgas Microsoft Windows Server 2003, Microsoft SQL Server ja Microsoft BizTalk Server mis integreerivad, jooksutavad ja haldavad veebi teenuseid ja veebi põhiseid rakendusi

- Klienditarkvara nagu Windows XP, Windows CE ja Microsoft Office XP, mis aitab aitavad arendajatel luua kasutajatele tuttava väljanägemisega kasutajaliideseid.

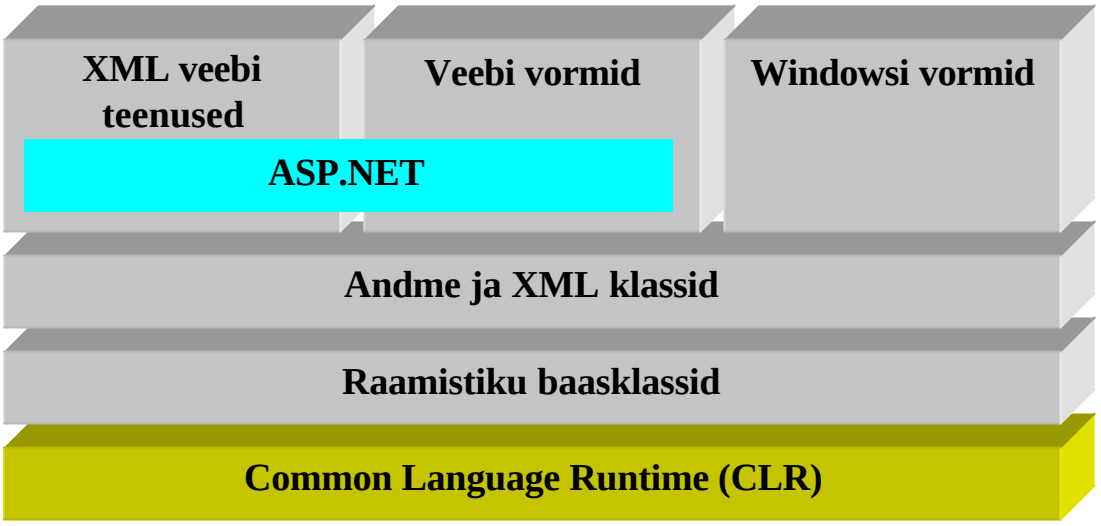


Joonis 3. .NET põhikomponendid

.NET raamistik on Windowsi komponent, mis on mõeldud järgmise põlvkonna rakenduste ja veebi teenuste loomiseks ja jooksutamiseks. .NET raamistik[16]:

- Toetab üle 20 programmeerimis keele
- Teeb programmeerimis musta töö programmeerijate eest ära. Võimaldades programmeerijatel tegelda rohkem äriloogika kirjutamisega
- Muudab lihtsamaks kui kui kunagi varem viimistletud, turvaliste ja töökindlate rakendsute loomise, rakendamise ja haldamise

.NET raamistik koosneb CLR'ist (Common Language Runtime) ja ühendatud klass moodulitest.



Joonis 4. .NET raamistiku ehitus

CLR vatutab teenundavate teenuste eest nagu keelte ingratsioon, turvalisus, mälukasutus, protsessor, vigade püüdmine. Lisaks sellele tegeb arenduse ajal CLR tüübi kontrolliga, keelte vaheliste vigade haldamisega, lihtsamate tegevuste interpreteerimine.

Raamistiku baasklassi moodulid pakuvad standardfunktsioone nagu sisend väljund, stringi töötlus, turvalisuse haldus, suhtlemine võrgus, vigadele reageerimine ja kasutajaliidese disain

ADO.NET klassid võimaldavad arendajatel pöörduda XML, OLEDB, Oracle ja SQL andmete poole XML klassid võimaldavad XML andmete töötlust. ASP.NET klassid võimaldavad koostada kasutajatele veebi liidesid, Windows forms annab võimaluse windowsi kasutajaliideste koostamiseks.

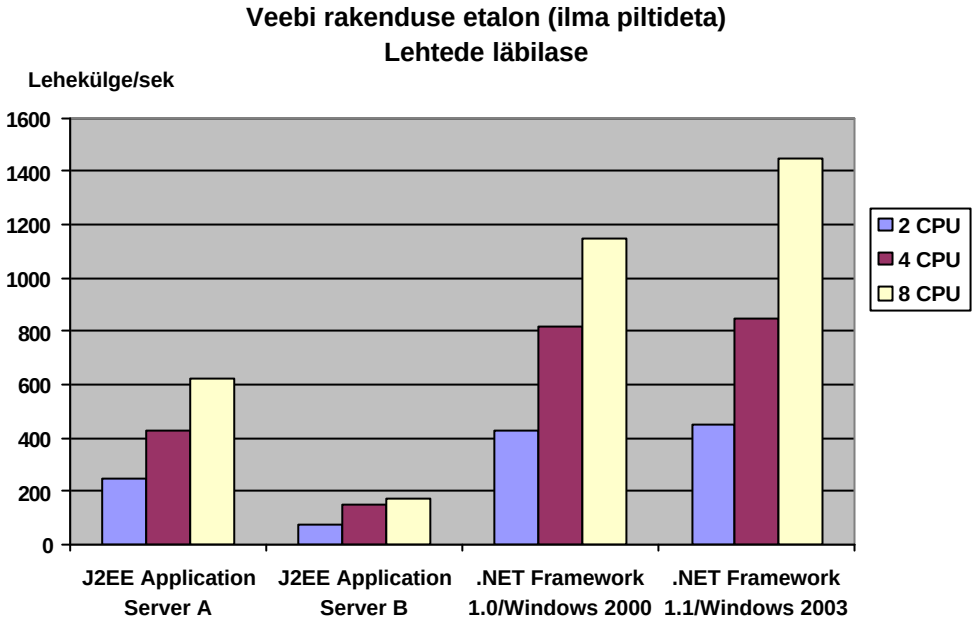
Koos pakuvad klasside kogud ühtse arenduskeskkonna kogu .NET raamistikus sõltumata keelest, mida arendaja kasutab.

Keeled, mida hetkel toetatakse on järgnevad:

APL	Fortran	Pascal
C++	Haskell	Perl
C#	Java Language	Python
COBOL	Microsoft JScript®	RPG
Component Pascal	Mercury	Scheme
Curriculum	Mondrian	SmallTalk
Eiffel	Oberon	Standard ML

Forth	Oz	Microsoft Visual Basic®

.NET raamistikuga konkureerivaks tehnoloogiaks on Sun Microsystems'i poolt pakutav J2EE. Kui võrrelda nende kahe tehnoloogia tööjõudlust samal riistvaral, kuid tootjate poolt soovitatud platvormil ei jää kahtlustki kumb tehnoloogia suudab rohkem kliente ära teenindada [16].



Joonis 5. J2EE ja .NET võrdlus

Hetkel on .NET raamistik olemas vaid Windows platvormile, kuid käivad arendustööd analoogse raamistiku ehitamiseks ka Linux platvormile. Kas ja millal see raamistik Linux'i peal tööle hakkab on veel vara öelda, kuid lootus on olemas.

Mõned põhjused, mis valida veebi rakenduste tegemiseks .NET platvorm [16]:

- Saadavus

  .NET tehnoloogia võttis windows 2000 paremad uuendused ja viis need järgmisele tasemele. Väga head rakenduse jälgimise vahendid ning rakenduste üksteisest isoleerimine võimaldavad .NET platvormile ehitatud rakendustel töötada järjest kauem kui varem

- Tööjõudlus

  Tänu kompileerimisele ja parandatud puhverdamis tehnoloogiatele on ASP.NET rakendused .NET platvormil kolm kuni viis korda kiiremad kui analoogsed ASP rakendused

- Programmeerijate produktiivsus

  Kõik tehnoloogiat katsetanud programmeerijad leiavad, et suudvad kiiremini programmeerida. Parem produktiivsus on saavutatud loogilise programmeerimis skeemi, ette kirjutatud koodiga klass moodulites ja tervele hulgale töödele, mida tehakse eesriide taga nagu näiteks mälu haldus

- Turvalisus

  Koodi kasutamise turvalisus on disainitud vastavalt oludele tänapäeva Internetis. .NET raamistik suudab koguda andmeid rakenduse autori ja päritolu kohta. Saadud andmeid saab võrrelda seatud turva reeglitega ning vastavalt sellele öelda kas kood käivitub või mitte. .NET raamistik suudab selles osas suhelda ka rakendusega andes vajadusel loa töötada edasi kuigi näiteks kirjutamine mõnesse kausta oli keelatud.

- Lihtne paigaldus

  .NET raamistikule kirjutatud rakendusi on lihtne paigaldada ja hallata. Programmide isoleerimine ja automaatne komponentide versiooni kontroll aitab vältida versioonikonflikte. .NET raamistikule kirjutatud rakenduste paigaldamiseks tuleb nad lihtsalt kopeerida soovitud masinasse. Mingeid komponentide registreerimisi pole vaja teostada.

- Mobiilsus

  .NET raamistik pakub arenduse tegemiseks ühtset programmeerimis mudelit nii veebi rakendustele kui ka platvorm rakendustele ja seda nii PC dele kui ka mobiilsetele seadmetele nagu mobiil telefonid ja pihuarvutid.

- Sisseehitatud tugi veebi teenustele

  .NETi veebi teenuste tugi on disainitu täiesti uuesti. Tegemist on platvormide vahel laiali jagatud rakendustega, mis kasutavad suhtlemiseks standardseid protokolle nagu XML, SOAP ja HTTP. Veebi teenused võmaldavad siduda erinevatel platvormide töötavadi rakendusi või jagada rakendust nagu teenust. Rakendus muutmine veebi teenuseks käib vaid ühe reaga koodis.

- Tugi enam kui kahekümnele programmeerimiskeelele

  .NET raamistik võimaldab integreerida elemente kahekümnest programmeerimis keelest, kujal millest varem võis vaid unistada. Kõik programmeerimis keeled kasutavad ühte ulatuslikku ja laiendatavat klasside

kogu. Erinevates keeltes tehtud komponendid saavad suhelda sujuvalt, ilma COM komponentideta.

- Paindlik andmete haldus

  ADO.NET on loodud andmete kasutamiseks nii nagu tänapäevased rakendused seda vajavad. Kasutades ADO.NET'i saavad programmeerijad töötada platvormist sõltumatult kasutades andmete XML puhvreid otseste andmete asemel. Selline lähenemine andmetele vabastab andmebaasi ühendusi ja võimaldab suuremat laiendatavust ja tööjõdlust.

2.2 XML veebi teenused

Komponentidel baseeruv ja objekt orienteeritud programmeerimine on muutunud järjest populaarsemaks. Tänapäeval näeb harva rakendust, mis ei koosne erinevatest tasemetest ühel või teisel moel. Pole haruldased ka olukorrad, kus erinevate komponentide tootjad on täiesti erinevad.

Kõige paremaks näiteks mitmetasemelisest rakendusest on e-poe lahendus. E-poe lahendus, mis paikneb nt veebifarmis peab võtma vastu tellimusi ja edastama need taustal töötavale tellimusi töötlevale ja täitvale rakendusele (back-end Enterprice Recourse Planning e. ERP). Tihtipeale paikneb ERP teises arvutis ja töötab tihtipeale sootuks teisel operatsioonisüsteemil.

Microsofti jagatud komponentide objekti mudel (Distributed Component Object Model e. DCOM) on süsteem, mis võimaldab kutsuda välja COM (Component Object Model) komponende, mis on installeeritud mõnda teise serverisse. Kuigi COM tugi on olemas mitmetel mitte Windowsil baseeruvatel operatsioonisüsteemidel, pole see muutunud neil platvormidel kunagi eriti populaarseks.

Probleemide lahendamiseks on tarkvaratootjad hakanud looma oma komponente. Loodud komponendid tuleb kliendi poole installeerida ja nende ülesandeks on andmevahetus serveriga. Siin on aga probleemiks mitmed tulemüürid, mis paiknevad kliendi ja serveri vahel ning asjaolu, et servereid, millega suhelda võib olla mitmeid.

Tulemüüride administraatorid on muutunud ünsa ettevaatlikeks ja mitte hädavajalike portide avamine peaaegu võimatu. Lisaks sellele ei tule sellised lahendused tihtipeale toime ühenduse katkemisega.

Seega kuna olemasolev tehnoloogia ei võimalda kõike on hakanud tarkvaratootjad looma oma infrastruktuuri eesmärgi saavutamiseks.[13]

Microsoft vastas sellele väljakutsele XML veebi teenustega. Põhimõtted, mida selle tehnoloogia loomisel kasutati olid järgnevad [13]:

- Platvormist sõltumatu – lahenduse kliendi poolne osa ei tohi sõltuda platvormist
- Interneti sõbralikkus – lahendus peab töötama hästi Interneti segastes oludes
- Konkreetsed tüübid – andmevahetusel ei tohi tekkida mingit kahtlust vahetatavate andmete tüübi osas. Lisaks sellele andmetüübid peavad kattuma hästi ka levinumate programmeerimiskeelte andmetüüpidega
- Kasutama maksimaalselt olemasolevaid standardeid – lahendus peab kasutama võimalikult palju hetkel olemasolevaid interneti standardeid.
- Iga keele toetus – lahendus ei tohi olla seotud ühegi konkreetse programmeerimis keelega. Kliendil peab olema võimalik veebi teenuse kasutamine sõltumata keelest, milles see kirjutati.
- Toetus igale jagatud komponentide infrastruktuurile – Lahendus ei tohi olla seotud ühegi konkreetse komponentidel baseeruva infrastruktuuriga. Teenuste paigaldamiseks ei pea midagi installeerima.

Lahenduseks neile nõuetele olid XML veebi teenused.

XML veebi teenuse ehitamiseks on kasutada järgnevad ehituskivid.

- Leidmine, UDDI, DISCO – Klientrakendus, mis vajab veebis avaldatud teenuse funktsionaalsust vajab mehhanismi selle teenuse leidmiseks. Otsingu sooritamiseks kasutatakse vastavaid katalooge, mis võivad paikneda nii kohapeal kui ka kusagil mujal
- Kirjeldus, WSDL, XML Schema, Docs – Kui teenus on leitud tuleb kliendile rääkida, millist infot antud teenus tööks vajab.
- Kirja formaat, SOAP – andmevahetuseks peavad klient ja server leidma ühise viisi andmete esitamieks. Ilma standardse andmevahetuseta on andmevahetuse abstraktsuse tõstmine võimatu. DCOM suurimaks miinuseks oligi see, et programmeerija pidi teadma andmeid liigutava protokolli ehtust. Muutes andmevahetuse abstraktsemaks saab

programmeerija tegelda rohkem äriloogika kirjutamisega ja ei pea ehitama infrastruktuuri lahenduse ellu viimiseks.

- Kodeerimine, XML – kuidas andmevahetusel andmeid edastatakse
- Tratsport, HTTP, SMTP jne – Kui kiri on kokku pandud ja vastavalt reeglitele kujundatud siis tuleb see viia kliendi juurest serverisse.

Veebi teenuse loomisel tuleb neid ehituskive oskuslikult laduda. Selleks vaateleme neid ja nende valiku põhjuseid natukene lähemalt.

2.2.1 Tratspordi protokolli valimine

Esimene samm veebi teenuse loomisel on välja selgitada, kuidas klient ja server omavahel suhtlevad. Klient ja server võivad paineda nii kohalikus võrgus (LAN) kui ka suhelda üle Interneti.

Tehnoloogiad nagu DCOM ja Java ei ole mõeldud suhtlemiseks üle Interneti. Samas sellised protokollid nagu HTTP ja SMTP on ennest tõestanud kui interneti protokollid.

HTTP põhised veebi rakendused on olekuvabad. Nad ei vaja püsivat ühendust serveriga. Kui server kuhu nad esialgu pöördusid enam ei vasta saab nad suunata sujuvalt teise serverisse. Pea igal asutusel on olemas infrastruktuur SMTP pakettide vahetamiseks.

Lisaks sellele levivad need protokollid kõikjale. Kõigil kasutajatel on olemas veebi sirvijad veebi lehtede vaatamiseks. HTTP läbib edukalt kõik NAT ja proxy serverid. Kohalikul SMTP serveril on aga öeldud kuhu ta peab neid kirju edasi saatma seega levitakse ka Internetti. [13]

SMTP pakettidel on omadus kujuda, kui töötlev server ei suuda neid piisavalt vastu võtta. Sellisel juhul saab server neid töödelda sellises järjekorras nagu nad saabusid, ilma et midagi vahepealt kaduma läheks

2.2.2 Kodeerimine

HTTP ja SMTP kirjeldavad kuidas andmed liigvad. Samas ei määra nad seda kuidas andmed on esitatud. Kuna on vaja platvormist sõltumatut kodeerimist siis on XML loomulik valik. XML'il on mitmeid eeliseid seal hulgas platvormist sõltumatus, toetus klassikalisteke tüüpidele, standardsele tähestikule.

Binaarsed kodeeringud nagu DCOM ja Java võivad kogeda siin ühilduvuse probleeme. Näiteks on erinevatel platvormidel on mitmebaidiste numbrite hoidmiseks siseselt erinevad meetodid. XML välistab sellised probleemid kasutades tekstipõhist andmete esitust standardse kooditabeliga. Lisaks sellele mõned protokollid toetavadi vaid tekstipõhist sisu nagu SMTP. [13]

Lisaks sellele on binaarsete kirjade koostamiseks vaja eraldi rakendusi, mis aitaksid arendajaid arendusprotesessis. XMLi kirjutamiseks saab aga kasutada iga tekstiredaktorit, millele lisanduvad aega mööda ka spetsiaalsed XMLi koostamise programmid

2.2.3 Formaadi valimine

Tihtipeale on vajalik kaasta saadetisse lisainformatsiooni kirja sisu kohta. Näiteks võib tekkida vajadus lisada kirjale teenuse tüüp või ruutimis info. XML ei paku mingeid vahendeid teate sisu ja andmete vahel. Trantspordi protokollid nagu HTTP pakuvad võimalust päise koostamiseks, kuid päisesse kirjutatav info ei pruugi olla standardne näiteks soovitakse saadetist saata mitmesse kohta kasutades erinevaid protokolle. Seega tuleks enne saatmist päises olev info ümber teisendada. Teisendamise käigus võib aga midagi kaduma minna. Kuna saadetav päise info käib kirja mitte tratspori kohta siis peaks see info olema koos kirjaga. [13]

Lahenduseks on SOAP. Kõik SOAP pakid peavad olema koostatud kui kirjad kus on päis ja kirja sisu. SOAP ei sea mingeid piiranguid sellele kuidas kirja sisu on esitatud.

Kuna SOAP sisaldab standardset lähenemist XML kirja koostamisele saab kogu selle loomise spetsiifika peita nt ASP.NET platvormi.

2.2.4 Kirjelduse valimine

SOAP pakub standardset meetodit kirjavahetuseks kliendi ja veebi teenuse vahel. Kuid lisaks sellele vajab klient ka lisainfot selle kohta, millises järjekorras infot lugeda ja kuidas loetut interpreteerida. XML Schema on raamistik, mille abil saab kirjeldada kirja sisu.

XML Schema pakub tervet hulka sisse ehitatud andmetüüpe kirja sisu kirjeldamiseks. Lisaks sellele on võimalik kirjldada ka oma andmetüüpe.

Raamistik sisaldab hulka andmetüüpide ja elemetide kirjeldusi. Veebi teenus ei kasuta seda raamistiku mitte ainult andmevahetuseks vaid ka saadud kirja õigsuse kontrollimiseks. [13]

Ainuüksi raamistikust ei piisa veebi teenuse kirjeldamiseks. Raamistik ei kirjelda kirjade vahetamise rütmi kliendi ja serveri vahel. Näiteks peab klient teadma kas saadetud pakile tuleb ka vastus, millist tratrspordi protokolli teenus kasutab ja kuhu pakk tuleb saata.

Seda informatsiooni pakub Web Services Description Language (WSDL) dekoment. WSDL dokument on XML dokument, mis kirjeldab täielikult veebi teenuse. WSDL dokumente kasutavad ka arendus vahendid, tehes programmeerijale teenuse kasutamise lihtsamaks. [13]

Hea dokumentatsioon peaks lisaks hädavajalikule informatsioonile sisaldama ka veebi teenuse dokumentatsiooni. Dokumentatsioonis peaks olema kirjas mida veebi teenus teeb, milliste liidete jaoks ta on mõeldud ning kuidas seda teenust kasutada. Selline info on eriti oluline kasutatele üle interneti

2.2.5 Leidmis mehhanismi valimine

Kui oled loonud veebi teenuse, kuidas kliendid peaksid need ülesse leidma? Kui kasutajad on sisevõrgus saab neile saata lihtsalt teekonna WSDL dokumendini. Kui tegemist on aga klientidega internetist on teenuse leidmine aga hoopis teistsugune.

Teenus, mida veebi teenuse leidmiseks vaja on Universal Description, Discovery and Integration (UDDI). UDDI on standartne, keskne kataloogiteenus, mida saab kasutada veebi teenuste leidmiseks. UDDI võimaldab teenust leida mitmete tunnuste abil. Sealhulgas tootja nimi, kataloog ja veebi teenuse tüüp. [13]

Veebi teenuseid saab reklaamida ka kasutades DISCO’t so Microsofti poolt kirjeldatud XML dukoment, mis võimaldab veebi teenuseid leida. Peamine DISCO kasutaja on Visual Studio.´

2.3 ASP.NET

Üks asi, mida XML veebi teenused ei kirjelda on kasutajaliides. See on koht, kus tulevad ASP.NET leheküljed appi.

ASP.NET puhul tekib küsimus: miks oli vaja uut ASP versiooni? Vastus on lihtne ASP ei täida enam neid funktsioone, mida temalt oodatakse. Põhjusi, miks traditsiooniline ASP enam ei kõlba on mitmeid [14]:

- ASP on skripti keel, tuginedes VBScriptile ja Jscriptile. Lisa interpretaatorite olemasolul saab kasutada ka teisi keeli. Interpreteeritava keele miinusteks on tugevate tüüpide ja kompileeritud koodi puudumine. ASP ei puhverda koodi see aga vähendab jõudlust
- ASP ei võimalda kasutada objekt orienteeritud programmeerimist. Ajal, kui ASP loodi olid lehed väiksed. ASP tegi aga võimalikuks väga suurte ja keeruliste lahenduste loomise. Kuid koodi ja kujunduse sedamine ühele lehele viis probleemideni kui meeskonnas oli nii programmeerija kui ka disainer
- ASP lehtedel tuleb kirjutada koodi iga asja kohta. Näiteks ka nii lihtne asi nagu kontroll kas vormil on kõik väljad täidetud. Samuti vajas koodi sisu puhverdamine sessiooni hoidmine jne.
- Veebi sirvijate ühilduvuse probleem on muutunud veebi seadmete ühilduvuse probleemiks. Hetkel käiakse veebis veel põhiliselt PC ja veebi sirvijaga kuid kui kauaks see nii jääb? Mobiilsed seadmed muutuvad üha populaarsemaks muutes arendstööd järjest keerukamaks. Kui soovid oma veebile maksimaalset saadavust pead kirjutama koodi, mis avastab neid seadmeid ja serveerib sobiva vastuse.
- Ühilduvus standarditega on äga oluline veebis. XHTML muutub üha populaarsemaks. XML ja XSL on laialt kasutusel. Mobiilsete seadmete rünnak võib tähendada ka vajadust WML toe järele

Need on vaid mõned probleemid, mis tekkisid ASP rakenduse loomisel kuid need pole ainsad. Suured muutused Internetis vajavad suuri muutuseid ka arendus tehnoloogiates.

ASP.NET ei ole mitte ASP järgmine versioon vaid on täiesti uus keel ja keskkond, pakkudes arendamiseks kõike seda, mida tänapäevane Internet vajab.

ASP.NET peamised eesmärgid olid [14]

- Puhtam kood
- Parandada arendust, laiendatavust, turvalisust ja töökindlust

- Parem tugi erinevatele veebi sirvijatele ja seadmetele
- Anda platvorm uue põlvkonna veebi rakenduste loomiseks

ASP toetas vaid limiteeritud skriptimis keeli VBScript ja JScript. :NET raamistik (framework) toetab mitmeid keeli. See võimaldab kirjutada keeles, milles antud lahendus kõige paremini välja tuleb. Vaikimisi toetab CLR (Common Language Runtime) Visual Basic .NET, C# ja JScript.NET'i kõik kompileeritavad keeled. Kompilaatorite olemasolul saab kasutada ka Perli, COBOL'it ja paljusid teisi. Lisaks sellele toetatab Visual Studio .NET C++ ja Java edasiarendust J#. Kuna tegemist on osaga raamistikust pole vahet millises keeles kirjutad.

Mitme keele tugi pole piiratud vaid sellega, mis on saadaval vaid võimaldab määrata ka seda kuidas kasutatakse. On võimalik kirjutada komponent ühes keeles ja kutsuda ta välja komponendist, mis on kirjutatud hoopis teises keeles. [14]

ASP lehtede üheks suureks probleemiks oli asjaolu, et lehekülg moodustas ühe funktsiooni. Lehekülje interpreteerimist alustati lehekülje algusest ja interpreteerimise järjekorras moodustus ka HTML. Puudus võimalus kusagil lehekülje alaosas muuta objekti lehe ülaosas. Iga muudatus HTMLis vajas eraldi koodi õigel kohal. Lahenduse sellele probleemile tõid ASP.NET server poolsed kontrollid. Tegemist on täiesti tavaliste kontrollidega (nupud, teksti kastid jne), mida töödelakse serveris, kuid mida saab kasutada kliendi juures. Eeliseks on see, et nende objektide poole saab pöörduda kõikjal kasutades objekti atribuute. See muudab aga koodi oluliselt paremini loetavaks.

ASP.NET pakub ka võimalust eraldada kood ja kujundus. See võimaldab teha koodi puhtamaks ja paremini letavaks ja kasutada ühe lehe arendamisel kahte inimest (disainerit ja programmeerijat)

Lihtsamaks on muudetud ASP.NET veebi konfigureerimine. Kogu konfiguratsiooni saab ära teha kasutades kahte XML faili. Muudatusi arvestatakse kohe ilma serverile või teenusele restarti tegemata!

ASP.NET'il on puhas objektorienteeritud ja sündmustel baseeruv programmeerimise mudel, mis muudab programmeerimise sarnasemaks tradistsioonilisele programmeerimisele Visual Basic'us või C's. ASP.NET lehed vajavad oluliselt vähem koodi kui sama funktsionaalsed ASP lehed. Erinevalt ASP lehtedest on ASP.NET lehed kompileeritud, mis muudab nende kasutamise tunduvalt kiiremaks. [14]

ASP.NET on osa .NET raamistikust – uus platvorm mis mornoseerib ja lihtsustab nii Windowsi rakenduste tegemist kui ka kasutamist.

.NET raamistik koosneb mitmetest osadest, kuid olulisemad neist on:

- Tegemist on täiesti uue platvormiga, mis on loodud Interneti rakenduste loomiseks võimaldades kasutada selliseid avatud stanardeid nagu XML, HTTP, SOAP
- Platvorm, mis pakub palju võimsaid rakenuse loomise tehnoloogiaid nagu Windows Forms klassikaliste graagiliste kasutajaliideste loomiseks ja ASP.NET veebipõhiste rakenduste loomiseks
- Platvorm, kus on kaasas ulatuslik klasside kogumik, mis annab lihtsa ligipääsu andmetele nii relatsioonilistele kui ka XML ja palju teisi
- Platvorm, kus on kaasas sadu baasklasse tüüpiliste programmeerimis ülesannete lahendamiseks
- Keelest sõltumatu paltvorm, mis võimaladab kasutada igat keelt nii nagu parajasti parem tundub
- Platvorm, mis ei unusta oma eelkäiaid võimaldades lihtsalt kasutada kõiki varem kirjutatud komponente, seal hulgas ka COM ja DLL
- Platvorm sõltumatu koodi käivitamise ja haldamise keskkonnaga, mida kutsustakse CLR (Common Language Runtime). CLR kindlustab et koodi käivitamine on turvaline, tekitades operatsiooni süsteemi peale ühe täiendava kaitseliini. See tähendab aga seda, et .NET raamistik võib töötada mitmetel platvormidel ja seadmetel

Turvalisus on väga oluline komponent tänapäeva rakenduste juures. Selleks, et kasutada maksimaalselt .NET võimalusi turvalise tagamiseks peab vaatlema tervikuna nii IIS (Internet Information Server) kui ka oma rakendust. .NET pakub näiteks kasutajate tuvastamiseks kolme võimalust: windows, passport ja oma oloogika. Ning IIS 6.0 lisab sinna basic, diggest, widows integrated ja passport [18]. Millal millist autentimis meetodit kasutada, sõltub juba rakenduse disainist.

3 Testimiskeskkonna Infosüsteem

Selles peatükis vaatleme rakenduse loomise protsessi kasutades TPÜ testimiskeskuse näidet. Selle protsessi tulemust .NET platvormil saab ka reaalselt vaadata Internetist aadressil http://koolitus.bcs.ee/~erkis/peda.

3.1 Ülevaade

Järgnevalt esitatakse üldised nõuded infosüsteemile, eesmärgid, põhiprotsesside, põhiobjektide, sündmuste ja tegutsejate kirjeldused ning infovajaduste loetelu.

3.1.1 Definitsioonid ja üldised nõuded ning eesmärgid

Järgnevalt on toodud definitsioonid ja nõuded, mis olid aluseks infosüsteemi loomisel.

Definitsioonid ja nõuded

- Testimiskeskkond (edaspidi keskkond) on abivahend.
- Keskkonnal on administraator.
- Administraator annab teistele kasutajatele ligipääsu keskkonda.
- Teisteks kasutajateks on testi koostajad ja registraatorid.
- Testi koostajad loovad ja parandavad teste.
- Registraatorid registreerivad üliõpilasi testima ning teevad kokkuvõtteid tulemustest.
- Testi küsimused on grupeeritud teemadesse.
- Üks küsimus saab olla vaid ühes teemas.
- Teemad moodustavad aine või testi.
- Iga teema võib osaleda mitmes ainekirjelduses.
- Üliõpilased täidavad teste.
- Üliõpilased võivad teha mitmeid erinevaid teste.
- Lubatud on ka sama testi uuesti sooritamine. Selleks tuleb üliõpilasele registreerida sama test uuesti
- Muudatused testi küsimustikes arhiveeritakse. Tehtud testid peavad olema taastatavad testi sooritamise seisuga.

Eesmärgid

- Teadmiste hindamise adekvaatsuse suurendamine.
- Õppejõudude töö efektiivsuse tõstmine.
- Testirühmade teadmiste statistilise analüüsi võimalus.

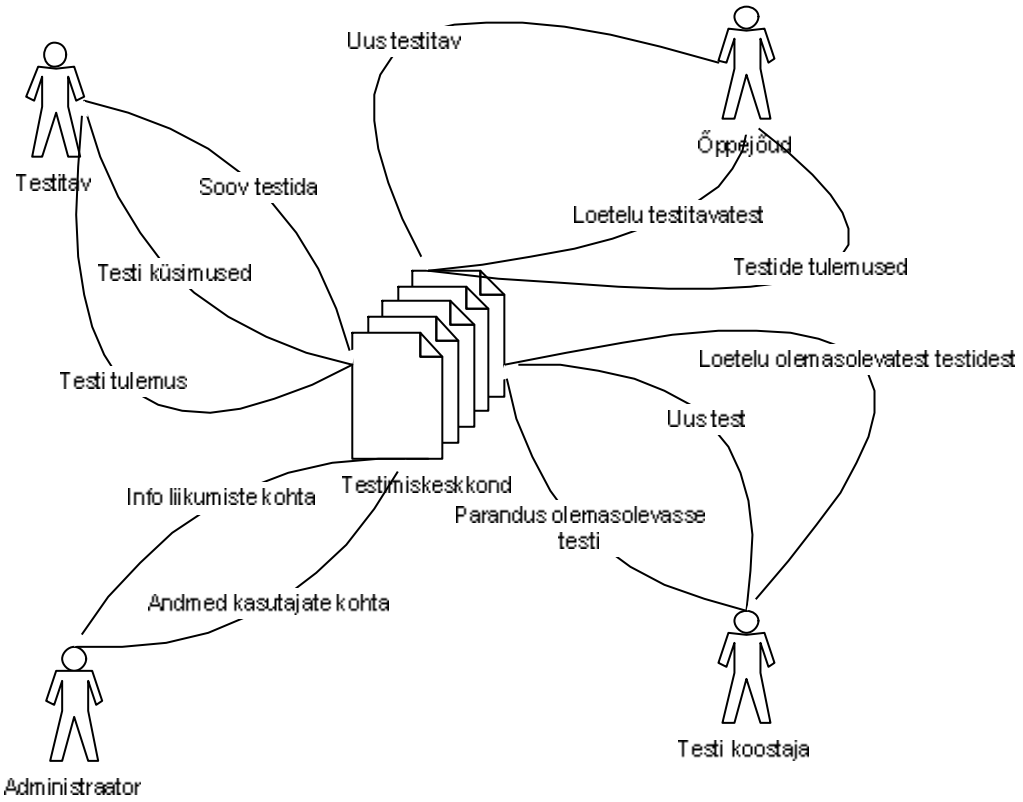
3.1.2 Põhiprotsesside loetelu

Testimiskeskkonna põhiprotsessid on:

- Testimise läbiviimine
- Testi tulemuste leidmine
- Testi tulemustest kokkuvõtete koostamine

Põhiprotsessi toetavad protsessid

- Testide koostamine
- Arhiveerimine



3.1.3 Põhiobjektide loetelu

Testimiskeskkonna põhiobjektid on:

- üliõpilane;
- registraator;
- administraator;
- testi koostaja;
- test;
- testimine;
- testide analüüsimine;
- küsimustikes tominud muudatuste arhiveerimine.

3.1.4 Sündmuste loetelu

Testimiskeskkonnaga seotud sündmused on:

- Üliõpilane registreeritakse testima
- Üliõpilane sooritab testib
- Testitav soovib juhendit keskkonna kasutamiseks
- Õppejõud soovib registreerida üliõpilasele testi
- Testi koostaja soovib koostada uut testi
- Testi koostaja soovib muuta olemasolevat testi
- Administraatori soovib jagada õiguseid
- Õppejõud soovib kokkuvõtet testi tulemustest
- Testi sooritamiseks ettenähtud aeg on läbi

3.1.5 Tegutsejate loetelu

Testimiskeskkonnas tegutsejad on:

- üliõpilane;
- registraator;

- administraator.
- testi koostaja;

3.1.6 Infovajaduste loetelu

Testimiskeskkonnas tegutsejate olulisemad infovajadused on:

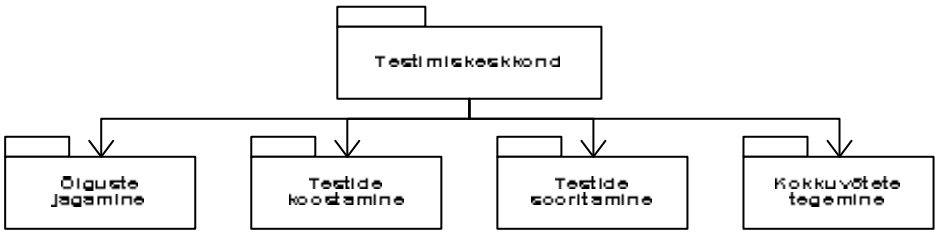
- Keskkonnas olevate testide loetelu
- Testi tulemused
- Registreeritud testid
- Kokkuvõte üliõpilaste rühma testimis tulemustest
- Üliõpilasele registreeritud test

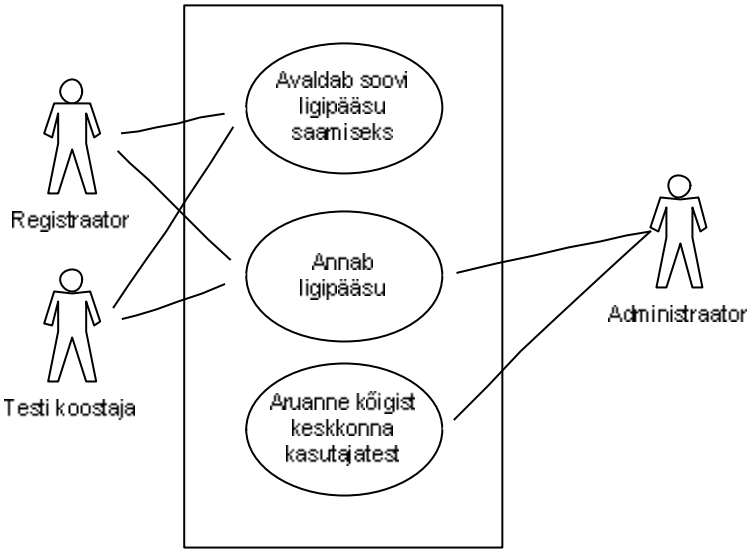
3.2 Infosüsteemi funktsionaalne vaade

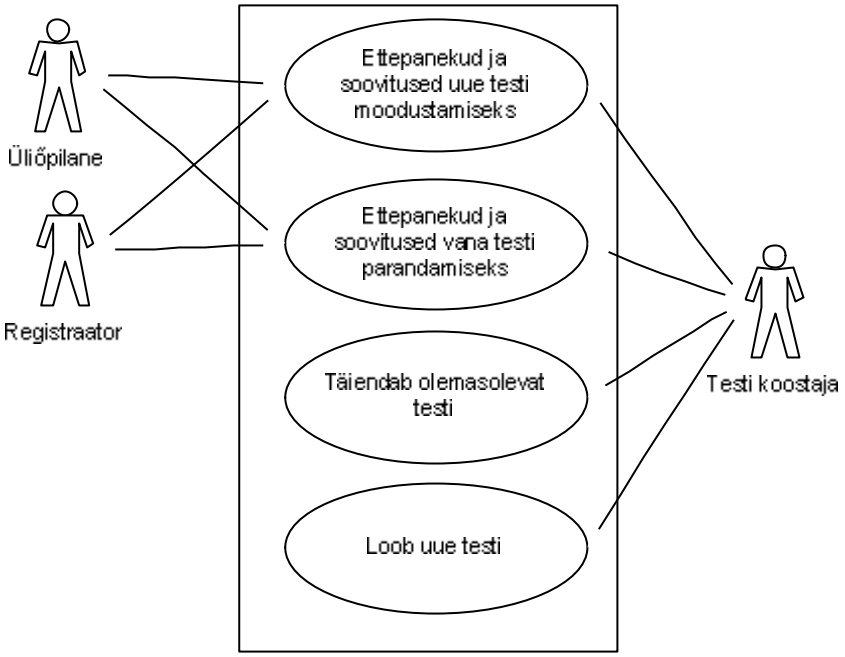
Järgnevalt esitatakse infosüsteemi põhifunktsioonid ja kasutusjuhud.

3.2.1 Infosüsteemi põhifunktsioonid

Infosüsteemi põhifunktsioonid on esitatud järgneva paketid iagrammina:



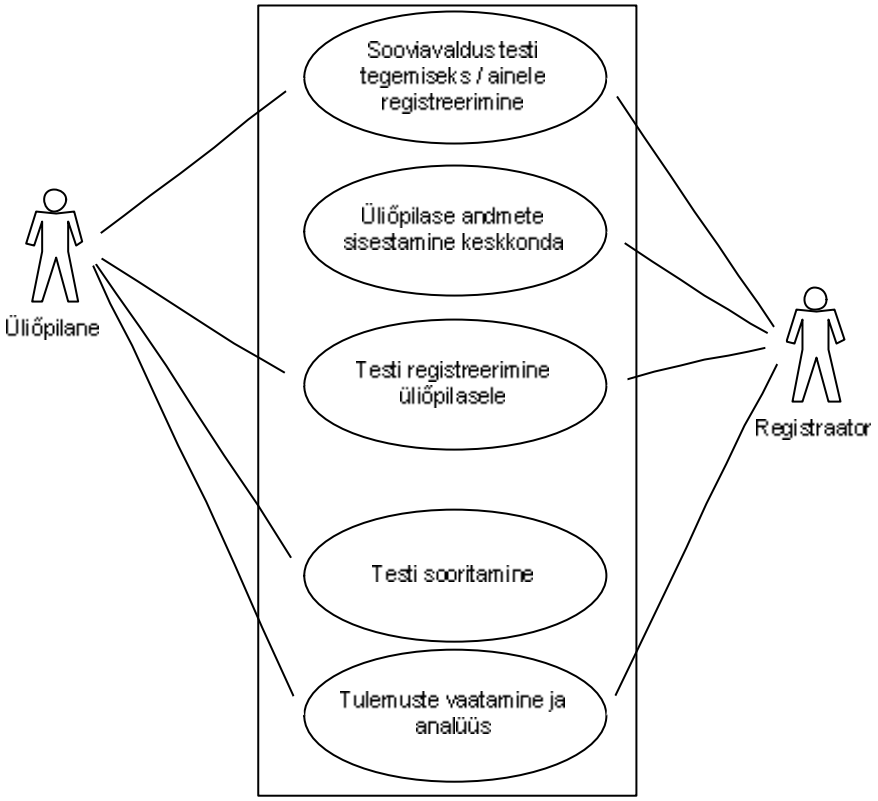

Avaldab soovi ligipääsu saamiseks
Annab ligipääsu
Aruanne kõigist keskkonna kasutajatest
Registraator
Testi koostaja
Administraator


Üliõpilane
Registraator
Ettepanekud ja soovitused uue testi moodustamiseks
Ettepanekud ja soovitused vana testi parandamiseks
Täiendab olemasolevat testi
Loob uue testi
Testi koostaja

Kirjeldus: Vatavalt laekunud ettepanekutele soovib testi koostaja moodustada uut testi. Käivitatakse õiguste kontroll. Testi koostaja moodustab uue testi.

Testi sooritamine

Testi sooritamise kasutusjuhtude diagramm on järgmine:



Kirjeldus: Õppejõud soovib üliõpilase taset määrata. Käivitatakse õiguste kontroll. Õppejõud määrab sobiva testi. Õppejõud tevitab üliõpilast ning edastab ka testi sooritamiseks vajaliku kasutajanime ja parooli.

Nimi: Testi sooritamine

Tegutsejad: üliõpilane

Kirjeldus: Üliõpilane soovib ainele pääsemiseks sooritada testi. Käivitatakse õiguste kontoll. Üliõpilane täidab registreeritud testi(d).

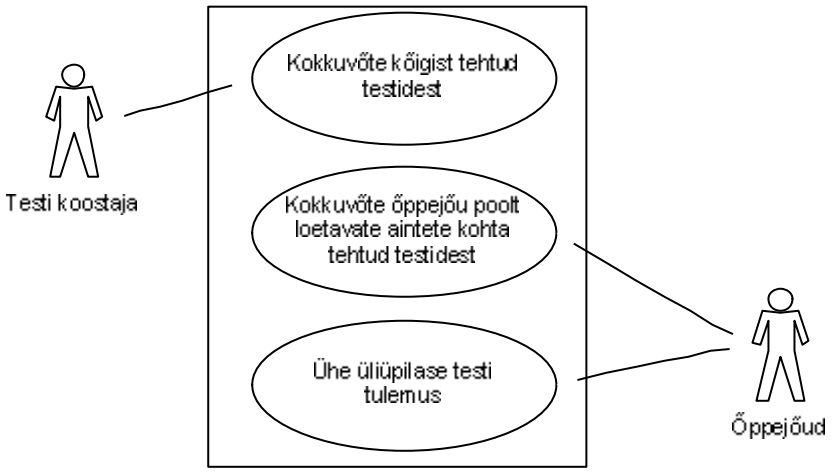
Nimi: Tulemuste vaatamine ja analüüs

Tegutsejad: õppejõud ja üliõpilane

Kirjeldus: Üliõpilane soovib teada testi tulemust. Käivitatakse õiguste kontroll. Väljastatakse testi üld skoor. Õppejõd soovib teada testi tulemust. Käivitatakse õiguste kontroll. Näidatakse õppejõule testi tulemusi teemade lõikes

Kokkuvõtete tegemine

Kokkuvõtete tegemise kasutusjuhtude diagramm on järgmine:



Nimi: Ühe üliõpilase testi tulemus

Tegutsejad: õppejõud

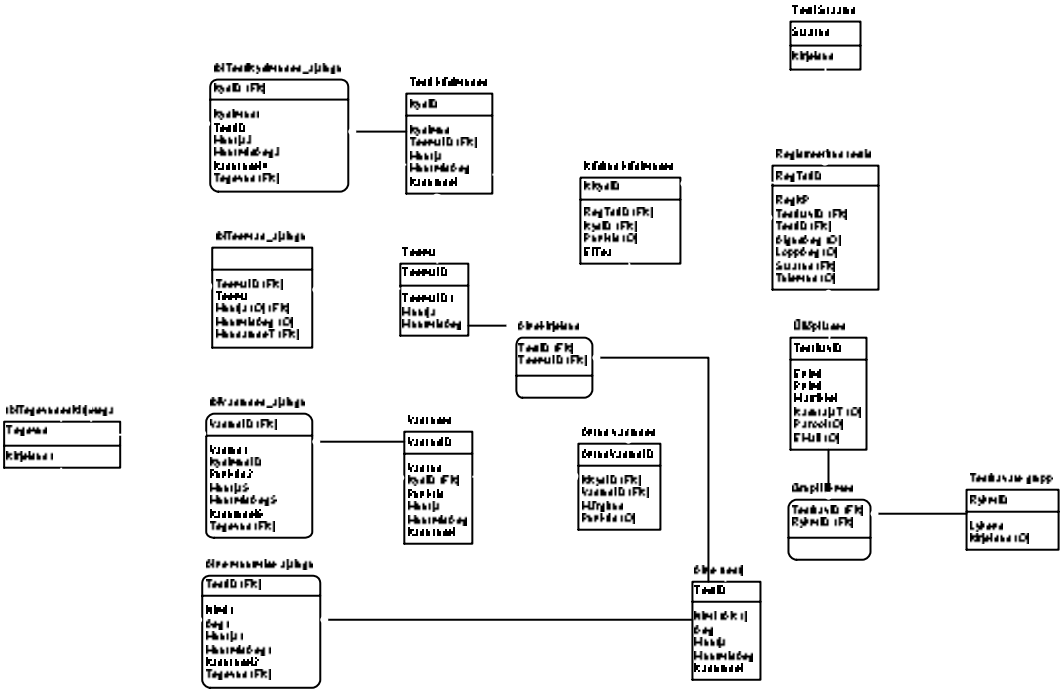
Kirjeldus: Õppejud soovib vaadelda ühe üliõpilase poolt sooritatud teste. Käivitatakse õiguste kontroll. Väljastatakse info soovitud testi kohta

3.3 Infosüsteemi andmevaade

Järgnevalt esitatakse testimiskeskkonna infosüsteemi loogiline andmemudel ja objektide atribuutide semantika.

3.3.1 Loogiline andmemudel

Testimiskeskkonna infosüsteemi loogiline andmemudel on järgmine:



Nimetus	Semantika
TESTITAVATE GRUPP	Testitavate gruppide nimetused. Kasutatakse aruandluses.
RyhmID (PK)	Rühma unikaalne kood
Lyhend	Rühma lühend
Kirjeldus	Rühma pikem kirjeldus
GRUPI LIIKMED	Testitavate kuuluvus gruppi antud ajahetkel
TestitavID (FK)	Testitava kood
RyhmID (FK)	Rühma kood
Üliõpilased	Testitavad üliõpilased
TestitavID (PK)	Testitava unikaalne kood
Enimi	Testitava eesnimi
Pnimi	Testitava perekonnanimi
Matrikkel	Testitava matrikli number
KasutajaT	Kasutajanimi testimiskeskkonda sisenemiseks
Parool	Parool testimiskeskkonda sisenemiseks
Email	E-Maili aadress
REGISTREERITUD TESTID	Üliõpilasele registreeritud testid. Sisaldab kõiki läbi aegade registreeritud teste, ka neid, mis on juba täidetud.
RegTstID (PK)	Registreeritud testi unikaalne kood
RegKP	Testi registreerimise aeg
TestitavID (FK)	Üliõpilase kood, kellele test on registreeritud
TestID (FK)	Registreeritud testi kood
AlgusAeg	Testi alustamise aeg
LoppAeg	Testi lõpetamise aeg
Staatus (FK)	Testi staatus
Tulemus	Saadud punktisumma. Arvutatakse testi kontrollimisel.

TESTI STAATUS	Testi staatuste kirjeldused
Staatus (PK)	Unikaalne identifikaator. Staatuse lühend
Kirjeldus	Staatuse kirjeldus
AINE (TEST)	Testid
TestID (PK)	Unikaalne identifikaator
Nimi	Testi nimetus
Aeg	Maksimaalne aeg testi sooritamiseks (minutites)
Muutja	Testi kirjelduse viimane muutja
MuutmisAeg	Viimase muudatuse tegemise aeg
Kasutusel	Kas test on veel kasutusel
AINEKIRJELDUS	Millistest teemadest test koosneb
TestID (FK)	Testi kood
TeemaID	Teema kood
TEEMA	Küsimuste grupid
TeemaID (PK)	Unikaalne kood
Kirjeldus	Teema kirjeldus
Muutja	Teema viimane muutja
MuutmisAeg	Viimase muudatuse tegemise aeg
TESTI KÜSIMUSED	Testis küsitavad küsimused
KusID (PK)	Unikaalne identifikaator.
Kysimus	Kysimuse tekst
TeemaID (FK)	Millise teema kohta küsimus käib
Muutja	Viimane küsimuse muutja
MuutmisAeg	Viimase muudatuse tegemise aeg
Kasutusel	Kas küsimus on veel kasutusel

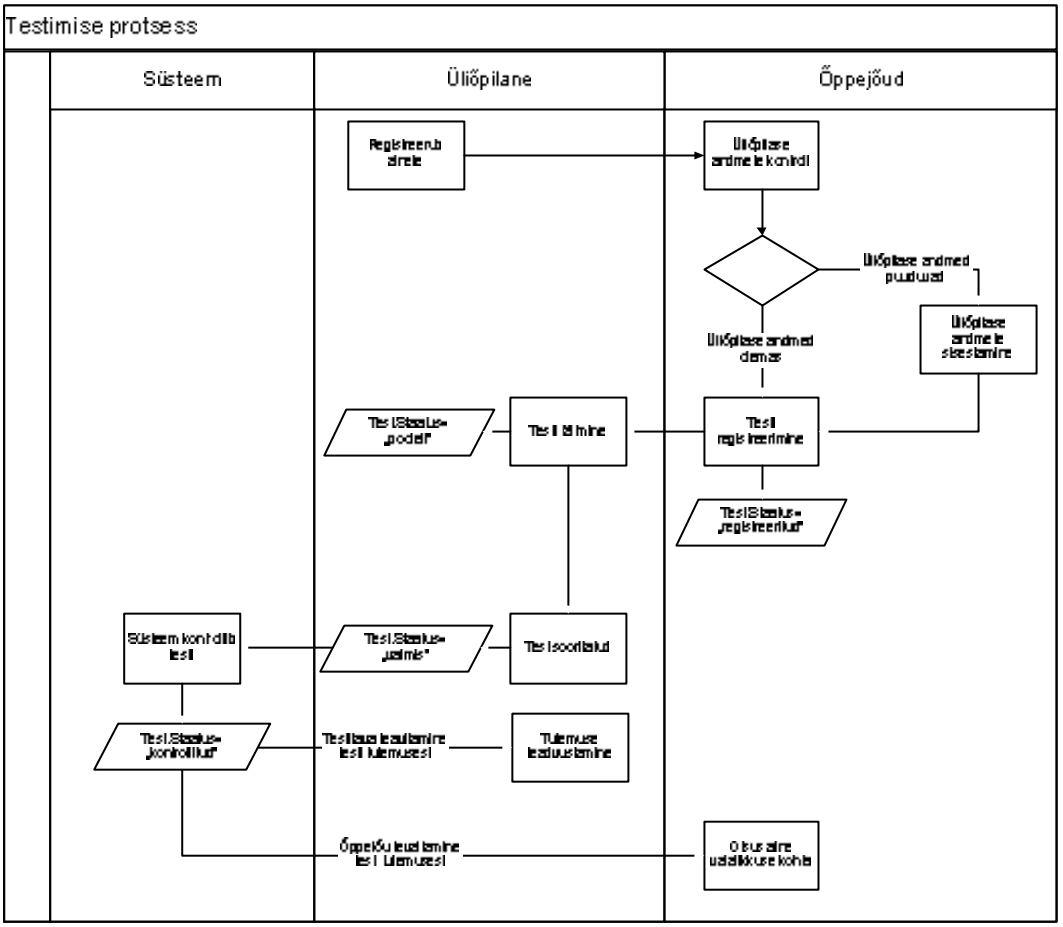
VASTUSED	Vastusevariandid küsimustele
VastusID	Unikaalne identifikaator.
Vastus	Vastusevariandi tekst
KysID (FK)	Millise küsimuse vastusevariandiga on tegemist
Muutja	Viimane vastusevariandi muutja
MuutmisAeg	Viimase muudatuse tegemise aeg
Kasutusel	Kas vastusevariant on veel kasutusel
KÜSITUD KÜSIMUSED	Küsimused, mis üliõpilastelt on küsitud
KKysID	Unikaalne identifikaator.
RegTstID (FK)	Registreeritud testi kood, mille raames küsimus esitati
KysID (FK)	Esitatud küsumuse kood
Punktid	Küsimuse vastuse eest saadud punktid
ANTUD VASTUSED	Üliõpilasele esitatud vastusevariandid ning antud vastused
AntudVastusID	Unikaalne identifikaator
KKysID (FK)	Küsitud küsimuse kood
VastusID (FK)	Esitatud vastusevariandi kood
Märgitud	Kas testitav märgistas vastusevariandi
Punktid	Vastuse väärtus

3.4 Infosüsteemi ajaline vaade

Järgnevalt esitatakse testimiskeskkonna testimisprotsessi kajastav tegevusdiagramm, sündmuste-kasutusjuhtude vastavustabel ja põhiobjektide seisundidiagrammid.

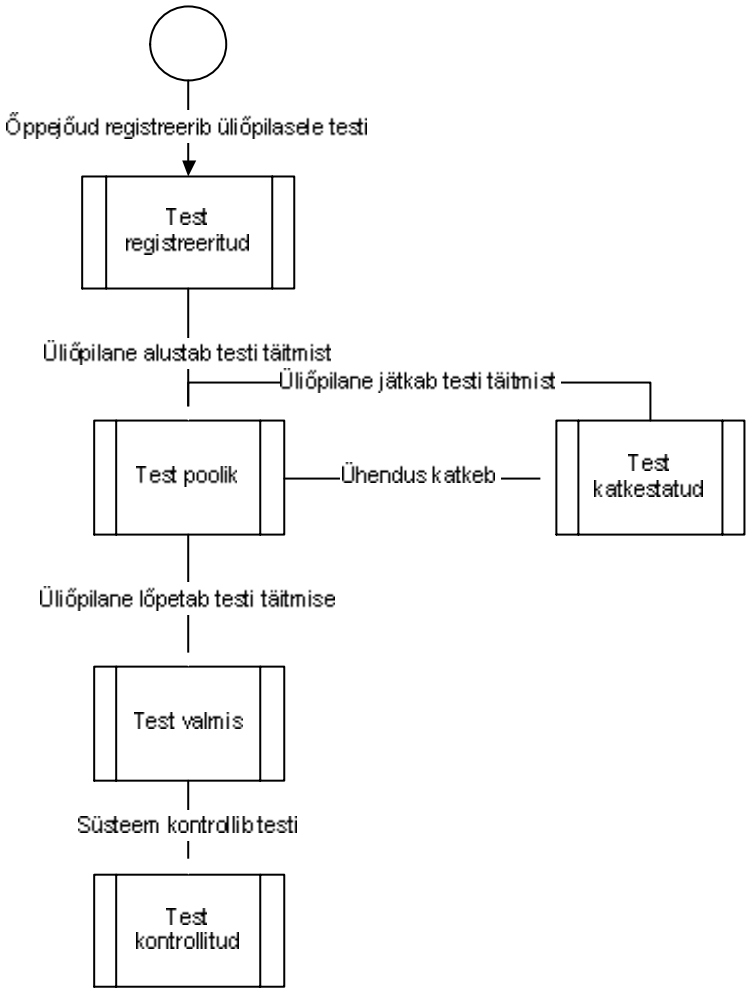
3.4.1 Testi sooritamise protsess

Järgnevalt esitatakse tegevusdiagramm, kus kirjeldatakse täpsemalt testi sooritamise protsessi. Tegemist on ühe testimiskeskkonna põhiprotsessiga.



3.4.2 Testi seisundidiagramm

Järgnevalt esitatakse testimiskeskkonna põhiobjekti “test” seisundidiagrammid.



Kokkuvõte

Käesoleva diplomitöö teemaks oli „Infosüsteemide loomine .NET tehnoloogia abil”. Valisin sellise teema kuna .NET on kuulutatud kõigi murede lahendajaks tänapäeva infosüsteemide loomisel. Inglise keeles on räägitud .NET’ist palju, kuid Eestis ja eesti keeles on sellest väga vähe juttu olnud.

Kuna nõudmised infosüsteemide funktsionaalsusele on viimaste aastatega kasvanud, on muutunud töötavate ja ootusele vastavate infosüsteemide loomine järjest keerulisemaks. Tänapäeval taotletakse süsteemi ehitamisel lisaks platvormist sõltumatusele ka sõltumatust seadmest.

Üks võimalus saavutada sõltumatust platvormist ja seadmetest on kasutada veebi lahendusi. Internet on kättesaadav peaaegu kõikjalt ning tekib järjest rohkem seadmeid, mis oskavad Internetis suhelda.

Siiani on olnud veebi tehnoloogiad lihtsate lahenduste loomiseks. Lisaks staatilistele HTML lehekülgedele on võimalik kasutada tervet hulka skriptimiskeeli, kuid vähesed nendest võimaldavad objekt-orienteeritud lähenemist programmeerimisele.

Platvorm ja keskkond, mis pakub turvalist ja efektiivset arendusplatvormi veebi rakenduste ehitamiseks sõltumata platvormist ja seadmest, kus neid rakendusi kasutatakse, on .NET raamistik. Kõik programmeerijad, kes on selle platvormiga kokku puutunud kinnitavad, et nad suudavad kirjutada rohkem ja paremini kui varem. Objekt-orienteeritud tehnoloogia muudab aga koodi korduvkasutuse palju lihtsamaks ning vähendab vigu rakenduste koodis tõstes sellega infosüsteemide töökindlust ja turvalisust.

Töö esimene osa andis ülevaate, kuidas tänapäeval infosüsteeme luuakse ja mida nende loomise juures arvestatakse, nõutakse ja eeldatakse. Selle analüüsi tulemusena selgus, et enamlevinud veebi lahenduste loomise tehnoloogiad ei ole sobivad moodsate veebipõhiste infosüsteemide loomiseks. Kuna infosüsteemid on muutunud keeruliseks ja arendusajad veninud väga pikaks, on hädavajalik programmi koodi korduvkasutus. Lahendus koodi korduvaks kasutamiseks on olemas olnud juba ammu (objekt-orienteeritud programmeerimine), kuid siiani pole olnud võimalik seda tehnikat kasutada veebi rakenduste loomisel.

Teises osas vaadeldi .NET tehnoloogiat, kuna see tehnoloogia ja platvorm on kuulutatud lahenduseks arendajaid vaevavatele probleemidele. Selgus, et kirjutades rakendusi .NET platvormile on võimalik kasutada objekt-orienteeritud programmeerimise võtteid, platvorm on oma ehituselt turvaline ning arendusprotsess on oluliselt lihtsam kui varem.

Kolmandas osas koostati rakendus kasutades .NET tehnoloogiat. Rakendus sai loodud selleks, et reaalselt neid väiteid .NET'i kohta kontrollida. Rakenduseks sai loodud testimiskeskus Pedagoogikaülikoolile, mille abil on võimalik testida üliõpilasi. Olles loonud analoogseid rakendusi ka varem võin kindlalt väita, et kasutades .NET platvormi oli selle rakenduse loomine oluliselt lihtsam kui kasutades ASP või PHP lehekülgi.

Summary

Modern applications are getting bigger and more complex. To build these applications need developers more time and recourses to design and write the structure of solution. The most important demands for modern applications are security, speed, integration with old systems, user friendly interface and platform and device independences.

The solutions for these problems are standards and easy to user tools for developing.

To get rid of depending from platform the developers are building web-based solutions. The problem with web is that you never know who is in there and how the information is moving. To write a web application you need to write HTML pages, which represent user interface. But HTML is standard for static pages, it doesn't support dynamic content. You can make these pages dynamic with several scripting languages as ASP, PHP, VBScript and JavaScript and many more, but none of them gives you the opportunity to use object oriented programming model and they aren't easy to use.

To make developing easier and more productive Microsoft has developed the .NET framework. This framework contains libraries to do everyday tasks. .NET framework is the platform for next generation application development.

This document gives a short overview of web developing technologies. Introduces .NET technology and gives a overview of developing application to .NET platform.

In first part it gives overview of application development process and web technologies.

The second part introduces .NET technology.

The third part gives overview of application that was developed using .NET technology. To see the application in action go to http://koolitus.bcs.ee/~erkis/peda

Kasutatud kirjandus

1. World Wide Web Consortsiumi ametlik veebileht, sisaldab veebi standardite kirjeldusi
   http://www.w3.org
2. Sun Microsystems Inc., lehekülg java spetsifikatsioonidega
   http://java.sun.com
3. Sun Microsystems Inc., lehekülg arendajatele
   http://www.sun.com/developers/
4. Torino ülikooli lühijuhendid HTML dokumentide ja nii serveri kui ka kliendipoolse JavaScripti koostamiseks
   http://www.polito.it/www/html/netscape/
5. Thomas A. Rowe'i ülevaade veebitehnoloogiatest
   http://www.ycoln-resources.com/kb/technotes/
6. Microsofti nägemus turvalise veebi rakenduse loomisest
   https://mocl.one.microsoft.com/cwdl/CWDL_ContentList.asp?FolderID=3638
7. .NET Framework Frequently Asked Questions, Andy McMullan
   http://www.andymcm.com/dotnetfaq.htm
8. Sissejuhatus Infosüsteemidess, Toomas Mikli TTÜ 1998
9. Microsoft'i VBScript keele tutvustus
   http://msdn.microsoft.com/library/en-us/script56/html/vbswhat.asp?frame=true
10. Microsofti ASP lehekülgede tutvustus
    http://msdn.microsoft.com/nhp/default.asp?contentid=28000522&?frame=true
11. Netscape JavaScripti õpik
    http://developer.netscape.com/docs/manuals/js/client/jsguide/index.html
12. Microsofti JScript tutvustus
    http://msdn.microsoft.com/library/en-us/script56/html/js56jsconabout.asp?frame=true
13. Building XML web servicer for the Microsoft .NET platform, Scott Short, Microsoft Press 2002

14. Professional ASP.Net 1.0, Richard Anderson, Brian Francis, Alex Homer, Rob Howard, Dave Sussman, Karli Watson, Wrox Press Ltd 2002
15. MOC #2300A Developing Secure Web Applications
16. Microsoft .NET Framework toote tutvustus http://msdn.microsoft.com/netframework/productinfo/overview/default.asp
17. Programming Microsoft SQL Server 2000 with XML second edition, Graeme Malcolm, Microsoft Press 2002
18. ASP.NET turvalisuse käsiraamat http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/authaspdotnet.asp
19. .NET turvalisuse tutvustus http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28001369&frame=true
20. Gibbs. Software's Chronic Crisis. Scientific American, September, 1994