TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutisüsteemide instituut

Raul Leemet 142791

ARVUTUSTEHNILINE KURSUSEPROJEKT (ÕPPEVAHEND) TAHVLIKELLA NÄITEL

Bakalaureusetöö

Juhendaja: Vladimir Viies
Dotsent

Tallinn 2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: [Ees- ja perenimi]

[pp.kk.aaaa]

Annotatsioon

Käesolev töö analüüsib arvutustehniliste kursuseprojektide ülesehitust, liigitust ning nendes seatud õpieesmärkide täitmist. Vaatluse alla on võetud projektide loomisel kasutatavad arendusmeetodid ning viisid, kuidas kuidas eri meetmeid projektides rakendada. Grupitöö raames loodud tahvlikella projekt iseloomustab töö käigus kasutatud meetodeid erinevates projekti etappides.

Töö esimene osa toob välja projekti definitsiooni ning IT projektide liigituse. Lisaks analüüsib see kursuseprojektide tähtsust ja kasulikkust õppekavades. Teises osas on täpsemalt kirjeldatud erinevaid arendusmeetodeid, tööjaotust ning eduka projekti loomiseks läbitavaid etappe. Kolmas osa hõlmab tahvlikella kirjeldust ning projektis kasutatud lahendusmeetodeid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38 leheküljel, 3 peatükki, 19 joonist, 3 tabelit.

Abstract

Integrated course project (study guide)

Since TuT study programme IASB02/14, different IT themed projects are compulsory subjects. Students wish to create grandiose projects, while not thinking how to complete it within the time period. The problem is, these kind of project will seldom be finished. Unfinished projects give some experience and knowledge to students, but learning outcomes usually will not be covered, if all project phases are not completed.

The goal is to provide students with a study guide, which gives an idea how to form different course projects and their phases. The thesis analyzes the structure of different course projects and review learning outcomes. Furthermore, it examines different development styles and methods and ways to divide tasks in a team. The plotclock, integrated course project, gives an example how to use different methods in various project phases.

The study guide helps students to plan ahead progress of a project. By that, the team efficiency rises and unexpected expenses decrease.

The first chapter gives an idea of an IT project and describes, by which bases projects can be divided into different categories. Furthermore, it analyses efficiency factor of course projects. The second chapter specifies different development ideologies, project phases and collaboration. The third part focuses on the plotclock project and brings out its proof of concept, how the project was solved and which methods was used completing the project.

The thesis is in Estonian and contains 38 pages of text, 3 chapters, 19 figures, 3 tables.

Lühendite ja mõistete sõnastik

3D	***Kolme dimensiooniline***
DSDM	***Dynamic systems development method***
	Süsteemide dünaamiline arendusmeetod- üks paljudest agiilsetest arendusmeetoditest
FPGA	***field-programmable gate array***
	väliprogrammeeritav väravamassiiv ehk programmeeritav ventiilmaatriks
IT	***Infotehnoloogia***
LED	***Light-emitting diode***
	Valgusdiood
PFM	***Pulse-frequency modulation***
	Pulsisagedusmodulatsioon- püsisageduslike impulsside sagedusega mootorite kiiruse reguleerimise meetod
PO	***Product owner***
	Toote omanik, tellija, esimese taseme klient
PWM	***Pulse-width modulation***
	Pulsipikkusmodulatsioon- püsisageduslike impulsside pikkusega mootorite kiiruse reguleerimise meetod
RX	***Serial bus receiver pin***
	Jadaedastussiini vastuvõtja viik
TX	***Serial bus transmitter pin***
	Jadaedastussiini edastaja viik
WiFi	***Wireless Fidelity***
	"Traadita kvaliteetne"- IEEE 802.11 sarja standarditel põhinev traadita kohtvõrgu tehnoloogia
XP	***Extreme programming***
	Ekstreemprogrammeerimine, üks paljudest agiilsetest arendusmeetoditest

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Alates õppekavast IASB02/14 on muutunud kohustuslikuks aineks erinevad IT temaatilised projektid. Probleemiks on tudengite visiooni erinevus projekides seatud eesmärkidest. Alatihti on sooviks luua suurejooneline toode või teenus, keskendumata lahendusmeetodile. Selline lähenemine annab küll kogemust prototüüpide loomisel, kuid reaalsete projektide elluviimisel kasutatavad põhimõtted jäävad tagaplaanile.

Töö on vajalik, kuna tudengite projektide ambitsioonikad plaanid jäävad tihtipeale lõpetamata prototüüpide tasemele liialt mahukate, ette planeerimata lahenduste tõttu. Planeerimatus väljendub töö käigus lisanduvate tingimustena- kvaliteetset lahendust saab luua vaid juhul, kui projekti algfaasis on teada kõik nõuded. Põhjalik analüüs enne projekti realiseerimist tõstab töötajate efektiivsust ning vähendab ootamatuid ressursikulusid. Kui hilisemas arendusjärgus lisandub nõudeid, tähendab see muutustest mõjutatavate moodulite osalist või täielikku ümberdisainimist. Optimaalse töö tegemiseks on tarvis esmalt planeerida ning seejärel liikuda teostuse juurde.

Töö sisaldab vajalikke pidepunkte sobiva lahendusmeetodi leidmiseks kolmandas peatükis. Tahvlikella projekti kirjeldus annab ülevaate nii agiilsest kui ka kose mudeli alusel sooritatud arendusprotsessist. Lisaks kirjeldab see arvutustehniline projekt ära, milliste projekti osade puhul on võimalik agiilseid meetodeid kasutada ning milliste puhul tuleb piirduda kose mudeli alusel põhineva arendusega.

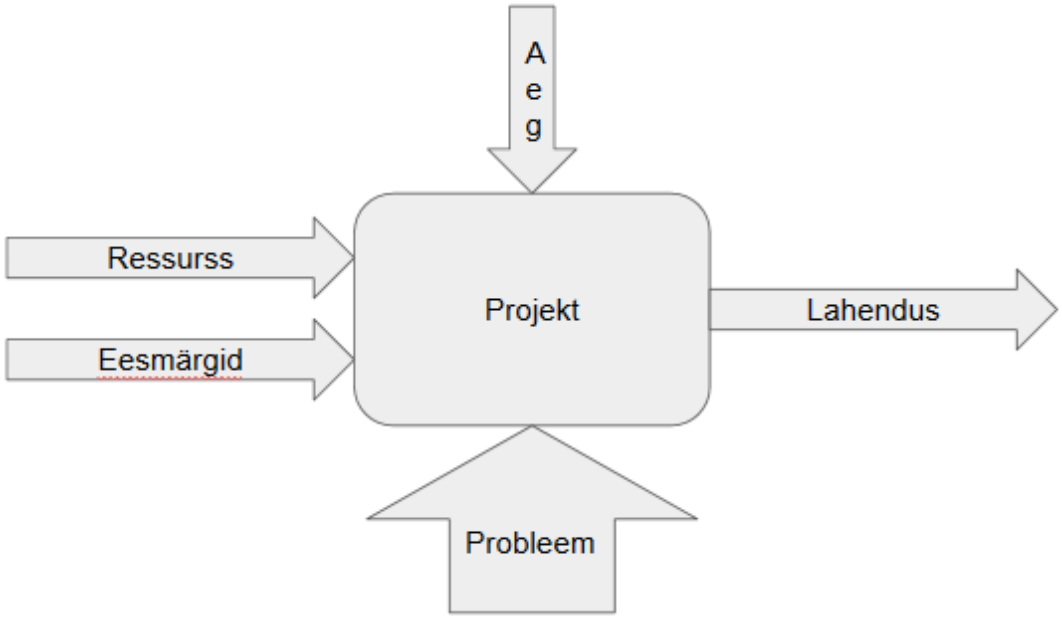
Lõputöö eesmärgiks on pakkuda juhend- ja näidismaterjali erinevate grupitööde lahendamiseks ning anda ülevaade projektide loomisel läbitavatest etappidest. Näidismaterjaliks on projekti aruanne, mida on kasutatud tarkvaraprojekti aruande näidisena. Juhendmaterjalina saab rakendada antud lõputööd, mis hõlmab endas teoreetilisi kirjeldusi ning rakendab neid projekti loomise käigus ära.

1 Projekt kui õppeaine

1.1 Projekti mõiste

Projekt on planeeritud toimingute kogum ühe või mitme püstitatud eesmärgi saavutamiseks kindla ajavahemiku jooksul [1]. Projektil on järgmised tunnused:

Eesmärgid: saavutatavad, protsessi kirjeldavad, üheselt mõistetavad [2].
Teostamise periood: sel on kindel algus- ning lõppaeg.
Ressurss: vajalike oskustega inimesed ja vahendid ideede teostamiseks [1].



Joonis 1. Projekti üldskeem

1.2 Kursuseprojektide erinevus teistest ainetest

Arvutisüsteemide instituudis kuulub IACB17/17 õppekavaversiooni kohustuslike projektide hulka tarkvara projekt. Valikainete hulka kuuluvad nii skeemitehnika kui ka arvutisüsteemide projekt. Need õppeained erinevad teistest, kuna õppekorraldus on orienteeritud väljaspoole klassiruumi. Tudengid on harjunud, et lahendatavad probleemid on püstitatud ning lahenduskäikude leidmiseks antakse suund õppejõudude

poolt kätte. Kursuseprojektid motiveerivad tudengeid valima neid huvitavaid ning neile sobivaid ülesandeid. Taaskäivitatud kohustuslike projektide kord õppekava IASB02/14 jooksul näitas, et tudengid seovad võimalusel enda ideed kursuseprojektidega. Õppeaine raames juhendamine ei sarnane teiste ainete praktikumide juhendamisega, kuna õppejõud ise pole projektis seisnevat probleemi ega selle lahendusmeetodit püstitanud. Täpsete juhiste asemel on võimalik tehtud töö alusel pidevalt tagasisidet anda ning tudengeid kogemuste baasil informeerida, kas valitud lahendused on sobivad või leidub paremaid alternatiive.

IT teaduskonna bakalaureuseõpe on suunatud iseseisvale tööle. Neli väljatoodud alusainet kirjeldavad seal kasutatavaid praktiliste ülesannete korraldusi:

Tabel 1. Õppeainete praktilise osa korraldus

Õppeaine	Praktiliste ülesannete tüüp	Praktiliste ülesannete korraldus
Automaatjuhtimissüsteemid	Laboriülesanded	Klassis juhendi alusel töötamine
Algoritmid ja andmestruktuurid	Kodune ülesanne	Iseseisev ülesanne
Arvutid	Praktikumid	Individuaalsed ülesanded
Arvutivõrgud	Laboriülesanded	Meeskonnatöö (3 liiget)

Tabelist selgub, et valdavalt on õppeainete raames tarvis induviduaalselt ülesandeid lahendada. Kursuseprojektid on oma korralduselt paindlikumad ning annavad tudengitele võimaluse vastavalt oma soovidele ja vajadustele koormust meeskonnaliikmete vahel jaotada. Suurimaks erinevuseks on projekti kui õppeaine puhul on mõttelaad, kus ühine eesmärk kaalub individuaalse panuse üle. Enim väärtustatakse dünaamilisust, kus meeskonnaliikmed teineteise puudusi ja eeliseid tasakaalustavad.

1.3 IT projektide liigitus õppeaine raames

Projektidel on tunnused, mille järgi neid liigitada. Jagan projektid järgmiste tunnuste järgi 3 kategooriasse: tarkvaraprojektid, riistvaraprojektid ning arvutitehnilised projektid.

Tabel 2. Projektide liigitus õppeaine raames

Projekti liik	Üldine tunnus	Näited valdkondadest	Näited projektidest
Tarkvara projekt	Rakenduste loomine platvormidele	Andmebaase kasutavad rakendused, automaattestid, veebilehe analüsaatorid	Diskreetse matemaatika kodutöö lahendaja
Riistvara projekt	Riistvara masinalähedane juhtimine	Mikrokontrolleritel põhinevad ülesanded, loogikalülitustel töötavad lahendused	„Snake“ mäng FPGA’l
Arvutustehniline projekt	Mehhanismide projekteerimine, riistvara juhtimine	Tarkvara, mehaanikat ning riistvara integreerivad tooted	Kerise kütmise juhtimine läbi veebiliidese Sumo robot

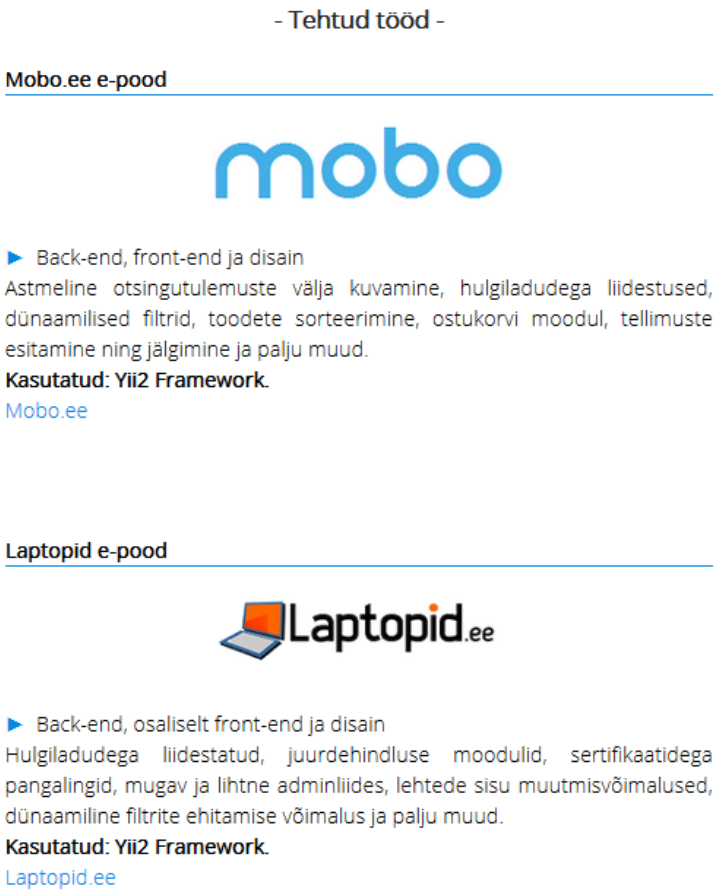
Tarkvaraprojektid hõlmavad endas algoritmide koostamist, andmestruktuuride valikut ning nende realiseerimist programmeerimiskeeltes. TTÜ ainekavas olev tarkvaraprojekt kujutab endast infotehnoloogilise ülesande grupiviisilist lahendamist. [3] Lahendatavate probleemide hulka kuuluvad näiteks infosüsteemide rakendamine, andmete töötlusülesanded või modelleerimised.

Riistvaraprojektide hulka kuuluvad mikrokontrollerite baasil lahendatavad ülesanded. Lisaks lähevad ka arvesse täielikult riistvara loogikalülitustel töötavad lahendused. Kuna integreeritud lülituste tootmine on kulukas, on mõistlik kasutada riistvara simuleerimiseks vastavaid tarkvaralisi lahendusi ning arendusplaate, näiteks Nexys.

Arvutustehnilised projektid kuuluvad hübriidprojektide hulka. Nendes on ühendatud riistvaralised lahendused tarkvaralisega oma funktsionaalsuse poolest. Näiteks kui valmistada mikrokontrolleri baasil miniatuurne lift, mis paikneb laboratooriumis ning luua kasutajale veebiliides selle jälgimiseks ja modifitseerimiseks, kvalifitseerub see tark- ja riistvara hübriidprojektiks, ehk arvutustehniliseks projektiks.

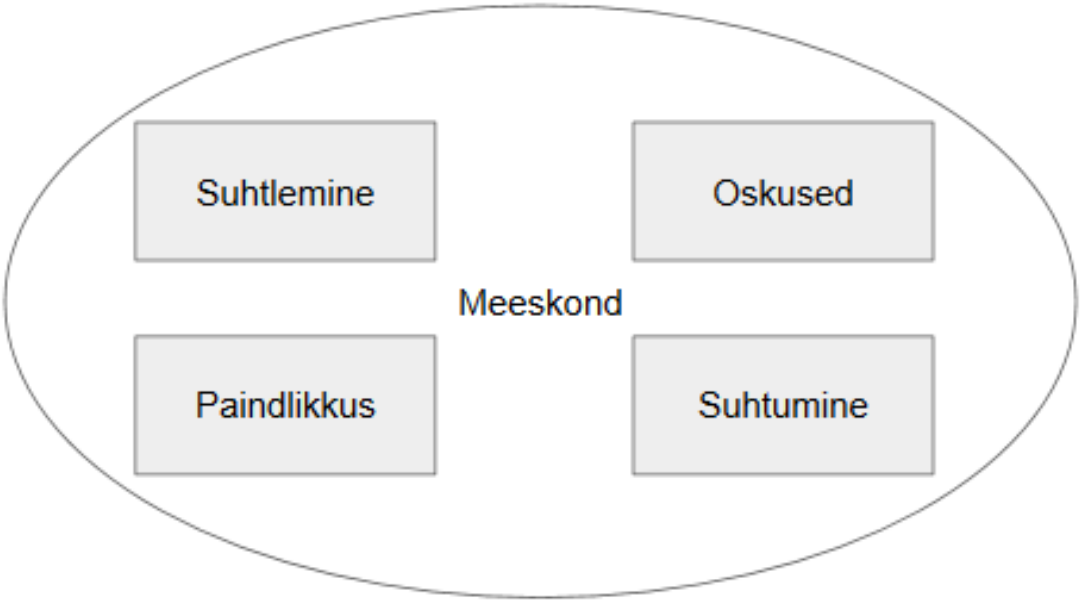
1.4 Kursuseprojektide kasutegur

Erinevad kursuseprojektid on kasulikud, kuna need annavad reaalset kogemust IT projektide läbiviimisel. Tööandjad soosivad sellist õppimismeetodit, kuna see annab koolilõpetajatele portfoolio, millega oma oskusi tõestada. Portfoolio on nimekiri projektidest, kus isik osalenud on (Joonis 2). See iseloomustab projektidesse antud panust ning kasutatud vahendeid, mis annab ülevaate inimese oskustest. Ilma seda tüüpi aineteta oleks vaid lõputöödel selline funktsioon. Portfoolio alusel on tööandjatel lihtsam tööotsijaid eri kategooriatesse jaotada, arvestades nende huvide ja oskustega. Ülikoolist saadud praktiline kogemus muudab vilistlased ettevõtetele kiiremini tulutoovaks, kuna eelnev projektidega kokkupuude kiirendab ametikoolituste läbimist.



Joonis 2. Portfoolio näide

Kursuseprojektid annavad ettekujutuse projektides olevast töökorraldusest. Kursuseprojektide üheks eesmärgiks on soodustada meeskonnatööd. Tarkvaraarenduse ettevõtete töökorraldus on projektipõhine. Töötajad osalevad eri projektides ning täidavad neis olevaid ülesandeid. Selline töökorraldus eeldab oskust töötada meeskonnas. Õppeprojektides osalemine kinnistab tulevastesse töötajatesse meeskonnatöö põhiväärtused (Joonis 3). Suhtlemine omab sama suurt rolli kui tehniline taip. Meeskonna töö juures on veel tähtis olla paindlik ning suhtuda väärikalt kaaslastesse.

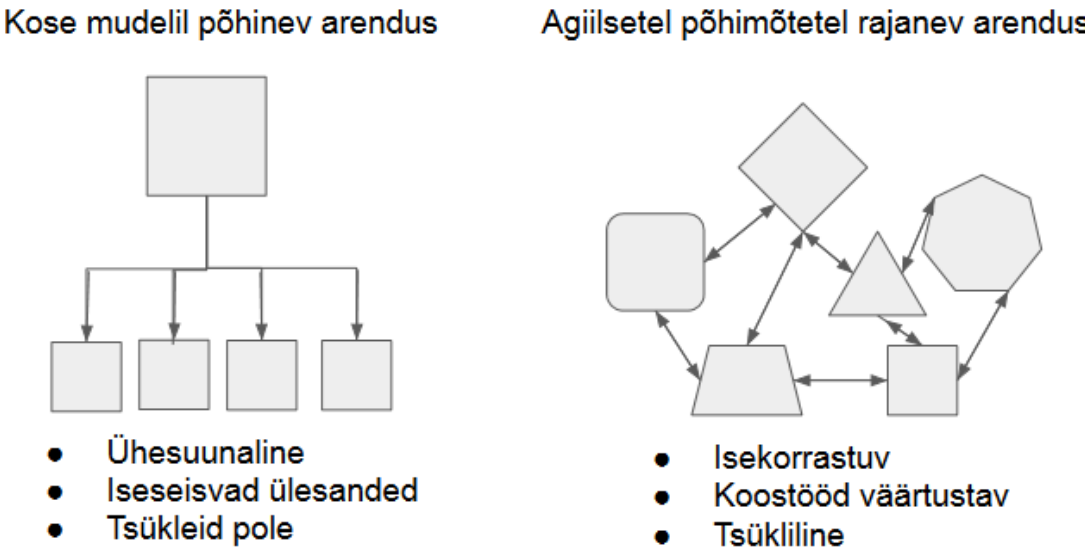


Joonis 3. Meeskonnatöö põhiväärtused

2 Projektides kasutatavad arendusmeetodid

IT projektides kasutatakse kaht tüüpi arendusmeetodeid:

- Dünaamiline ning voolujooneline arendus, ehk agiilsetel printsiipidel rajanev arendus
- Kavandi alusel loodavad projektid, ehk kose mudelil põhinev arendus



Joonis 4. Arendusmeetodite võrdlus

2.1 Agiilsus

Agiilsus ei ole arendusmeetod, raamistik või protsess- see on väärtuste ja printsiipide kogum. Agiilse tarkvaraarenduse manifest paneb paika selles kasutatavad põhimõtted. Manifest sündis tarkvara loomise käigus ning tekkinud pidepunktid, mis suurendasid efektiivsust, võeti uute arendusstiilide aluseks. Mainitud on nelja väärtust [4]:

- suhtlemist ning inimesi enam kui arendusvahendeid ja protsesse.
- funktsioneerivat tarkvara enam kui põhjalikku dokumenteerimist.

- kliendi soove ja kaasamist arenduses rohkem kui kirjalikke lepinguid.
- võimet kohaneda muutustega, kui algsest plaanist kinnipidamist.

Agiilse tarkvaraarenduse manifestis sisaldub lisaks neljale väärtusele 12 põhimõtet [5]:

1. Tarkvara toimetatakse kliendile võimalikult kiiresti ja tihti, et rahulolu tagada.
2. Kohanetakse muutustega plaanis ka arenduse lõpus, tagades sellega kliendi konkurentsivõime.
3. Tarkvaraarendust plaanitakse aja alusel, ehk tarnitakse kliendile tihti tulemusi iga paari nädala kuni paari kuu tagant.
4. Soositakse igapäevast koostööd eri valdkonna spetsialistide ja inseneride vahel kogu projekti vältel.
5. Kui tagada töötajatele sõbralik ja pingevaba töökeskkond, tõstab see motivatsiooni ning töö kiirust ja kvaliteeti.
6. Parim viis info vahetamiseks on silmast-silma vestlus.
7. Projekti progressi hinnatakse töötava tarkvara alusel.
8. Tagatakse jätkusuutlik areng samas tempos
9. Arenduse käigus pööratakse rõhku optimeerimisele ja korrektsele disainile, et tagada tarkvaraarenduse kiirus ja paindlikkus
10. Soositakse lihtsaid lahendusi ning välditakse ebaefektiivse töö tegemist
11. Meeskondadele tegevusvabadus annab eeldusi parimate lahenduste tekkimiseks.
12. Tiimil on võimalus end vastavalt oludele kohandada, et võimalikult efektiivselt probleeme lahendada.

2.2 Agiilsed arendusmeetodid

Agiilsete arendusmeetodite alla kuuluvad Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming ning paljud teised. Agiilseid arendusmeetodeid on kümneid ning neil on ühised väärtused ja põhimõtted. Antud töös võtame vaatluse alla SCRUM ning XP meetodi. Erinevus seisneb neis kasutatavates tavades. [6]

Sarnasuseks on arendustsüklite ehk sprintidele rakendamine. Pärast igat sprinti esitletakse toote funktsionaalsust kliendile ning järgnevas arendustsüklis kohandatakse toodet vastavalt kliendi tagasisidele. Toode on valmis, kui kõik kliendi nõudmised on täidetud.

Joonis 5. Agiilse arendusmeetodi printsiip [17]

2.2.1 Scrum

SCRUM ideoloogia seisneb iseorganiseeruvates meeskondades. Tarkvaraarenduse meeskonnale ei edastata täpseid juhiseid, kuidas projekti lahendada. Tiimile edastatakse probleemid ja kriteeriumid ning selle põhjal on meeskonnal vabadus valida, kuidas nende alusel lahendus leida. SCRUM tiimides pole kindlaid rolle. Kõik liikmed peavad mingi rakenduse osa valmis tegema, kuid tööjaotus teineteise vahel otsustatakse ühiselt ära. [7]

Arenduse käigus on meeskonnal kaks välistuge: scrum master ning toote omanik (PO). Toote omanik suunab arendusmeeskonda, et valmiks kliendile sobilik toode. Scrum masteri ülesanneteks on aidata: [8]

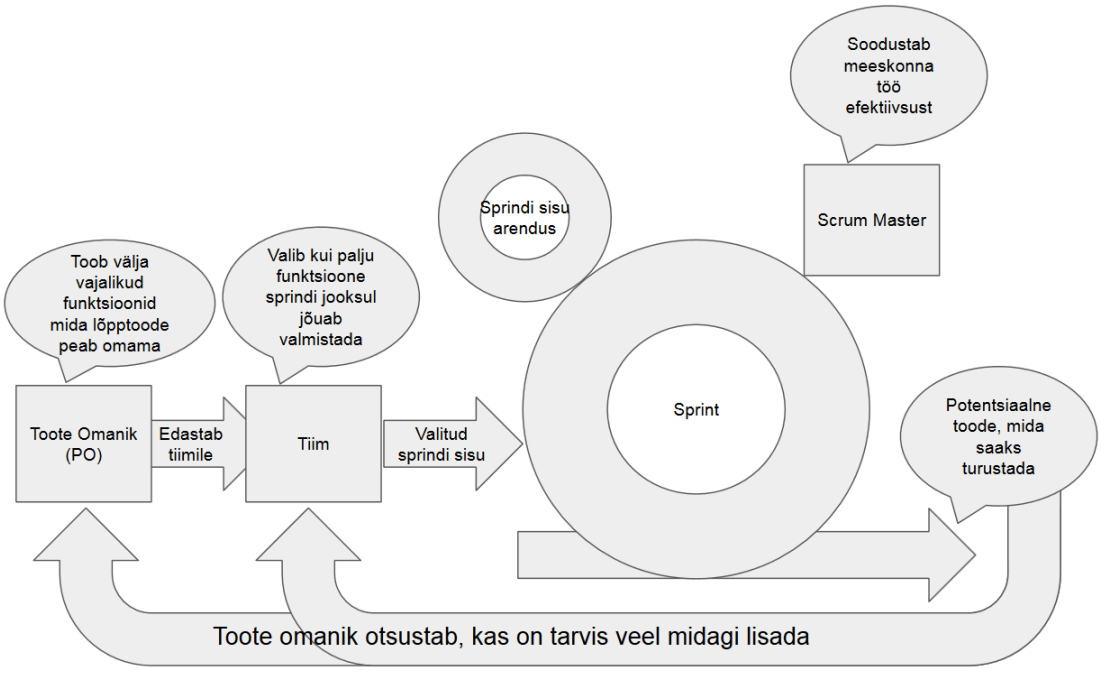
1. meeskonnal jõuda otsustes ühisele meelele.

2. kinni pidada tiimisiseselt kehtestatud reeglitest.

3. lahendada probleeme, mis pidurdavad meeskonna edusamme.

4. kõrvaldada väliseid segavaid faktoreid.

SCRUM arendusmeetodi käigus jaotatakse projekti teostamine aja alusel sprintideks. Näiteks võetakse ühe sprindi pikkuseks 2 nädalat ning pannakse paika eesmärgid, mida selle ajavahemiku sees tuleb täita. Eesmärgiks võib olla toote mingi osa valmistamine koos testimise ning arendatavasse süsteemi integreerimisega, mida kliendile demostreerida. Kui produkti osa valmis oodatust kiiremini, saab ülejäänud aega järgmise detaili loomiseks kasutada. Sprindijärgselt tutvustatakse valmistatut kliendile, mille põhjal toodangule tagasisidet kogutakse. Kui klient soovib oma tootele veel funktsionaalsust, liigub arendusmeeskond järgmise sprindi juurde. Protsessi korratakse, kuniks klient potentsiaalse väljastatava produktiga rahul on. [7]



Joonis 6. SCRUM arendusmeetod

2.2.2 Ekstreemprogrammeerimine (XP)

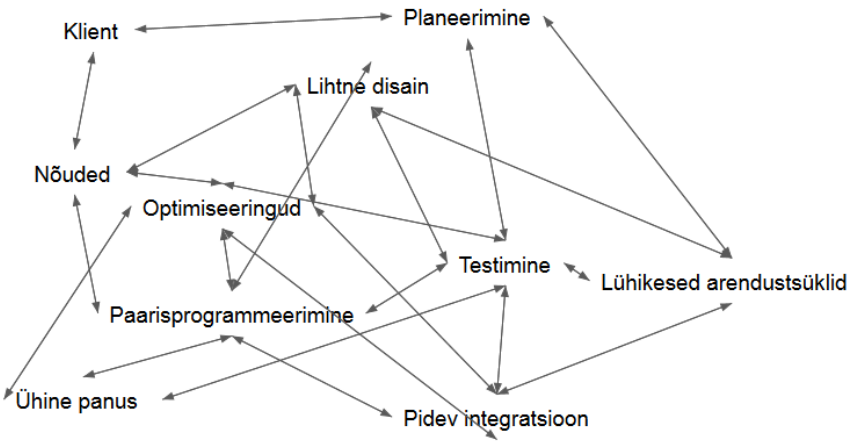
Ekstreemprogrammeerimist iseloomustavad järgmised põhimõtted:

- Programmkoodi analüüsitakse kogu projekti jooksul ning kõigi meeskonnaliikmete poolt (paarisprogrammeerimine).

- Esmalt kirjutatakse testid ning seejärel asutakse alles toote funktsionaalsust välja töötama.
- Kõik tegelevad programmkoodi optimiseeringutega (refactoring).
- Soodustatakse kõige lihtsaima disaini loomist, mis täidab etteantud nõudeid.
- Arenduses kasutatakse väga lühikesi tsükleid- tavaliselt kestavad alla nädala.

XP meetod loodi, et toodet vastavalt ärimudeli muutustele kiiremini kohendada, suurendada tiimi produktiivsust ning muuta meeskonnatöö käigus tehtav tarkvaraarendus meeldivamaks. Tänu automaattestidele ning pidevale kontrollile leitakse XP meetodit rakendades tekkinud vead varajases staadiumis. [9]

Ekstreemprogrammeerimine soodustab kõigi osapoolte vahelist suhtlust. Enamik projektides tekkivatest probleemidest saavad alguse möödarääkivustest. Et selliseid olukordi ära hoida, rakendab XP võtteid, kus suhtlusest ei saa hoiduda. Testimine, paarisprogrammeerimine ning analüüs kohustab programmeerijad, kliendid ja juhatajad omavahel suhtlema (Joonis 7).



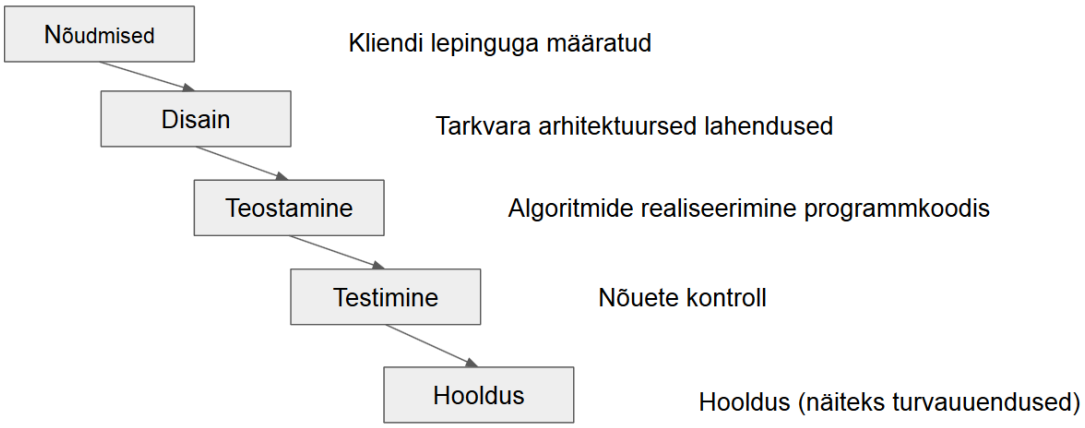
Joonis 7. XP valdkondadevaheline suhtlus

Lisaks suhtlusele, pannakse rõhku lihtsatele lahendustele. Nii suhtlus kui ka lihtsus on omavahel tihedalt seotud. Mida rohkem suheldakse, seda paremini saadakse aru nõuetest, mis annab programmeerijatele enesekindlust lihtsaima lahenduse rakendamiseks. Ekstreemprogrammeerimises välditakse liiasust- pigem eelistatakse lihtsat lahendust aja jooksul täiendada kui luua keerukas funktsioon, mida ei pruugita lõpptootes üldse kasutada.

2.3 Kose mudeli põhine meetod

Waterfall mudel võeti tootva tööstuse käest üle ning seda hakati tarkvaraarenduse algusaastatel rakendama. Kose mudel rajaneb kolmel alustalal:

1. Kliendi osalus projekti arendusel on minimaalne
2. Projekti dokumenteeritakse täiel määral
3. Projektide etappe läbitakse kindlas järjekorras



Joonis 8. Kosemudeli printsiip

Kose mudeli meetodit kasutades luuakse projektist täpne kavand. Esmalt suheldakse kliendiga ning kogutakse valmistatava toote või teenuse kohta kõikvõimalikud nõudmised. Nende põhjal koostatakse dokumendid, et klient saaks pärast arendusprotsessi algfaasis kokkulepitud funktsionaalsusega toote. Arendusfaasis kliendile prototüüpi ei esitleta ning seega puutub klient oma tootega kokku vaid projekti alg- ja lõppfaasis. [10]

Kose mudeli puhul valmistatakse põhjalik kavand sellest, kuidas kliendi nõudeid täita. Selles püstitatakse kasutatavad arhitektuursed lahendused. Arendusprotsessis luuakse nõuete alusel algoritmid ning need realiseeritakse programmkoodis. Realiseeritud toote kõrvale valmib kõrvalproduktina põhjalik kirjeldus sellest, kuidas valminud projekt töötab.

Kose mudeli rakendamisel on vajalik, et kõik nõuded oleks fikseeritud, kuna hiljem ei ole soodne süsteemi lisafunktsionaalsust juurde lisada. Hilisem nõuete lisamine tekitab

probleeme, kuna kose malli puhul iga tarkvara loomise protsessi etapp lõpetatakse enne kui järgmisega alustatakse. Seega, eelnevate etappide muudatused võivad järgnevate sisu tühiseks muuta, mistõttu on tarvis juba projekti algul kogu töö käik kindlaks määrata.

2.4 Projekti etapid

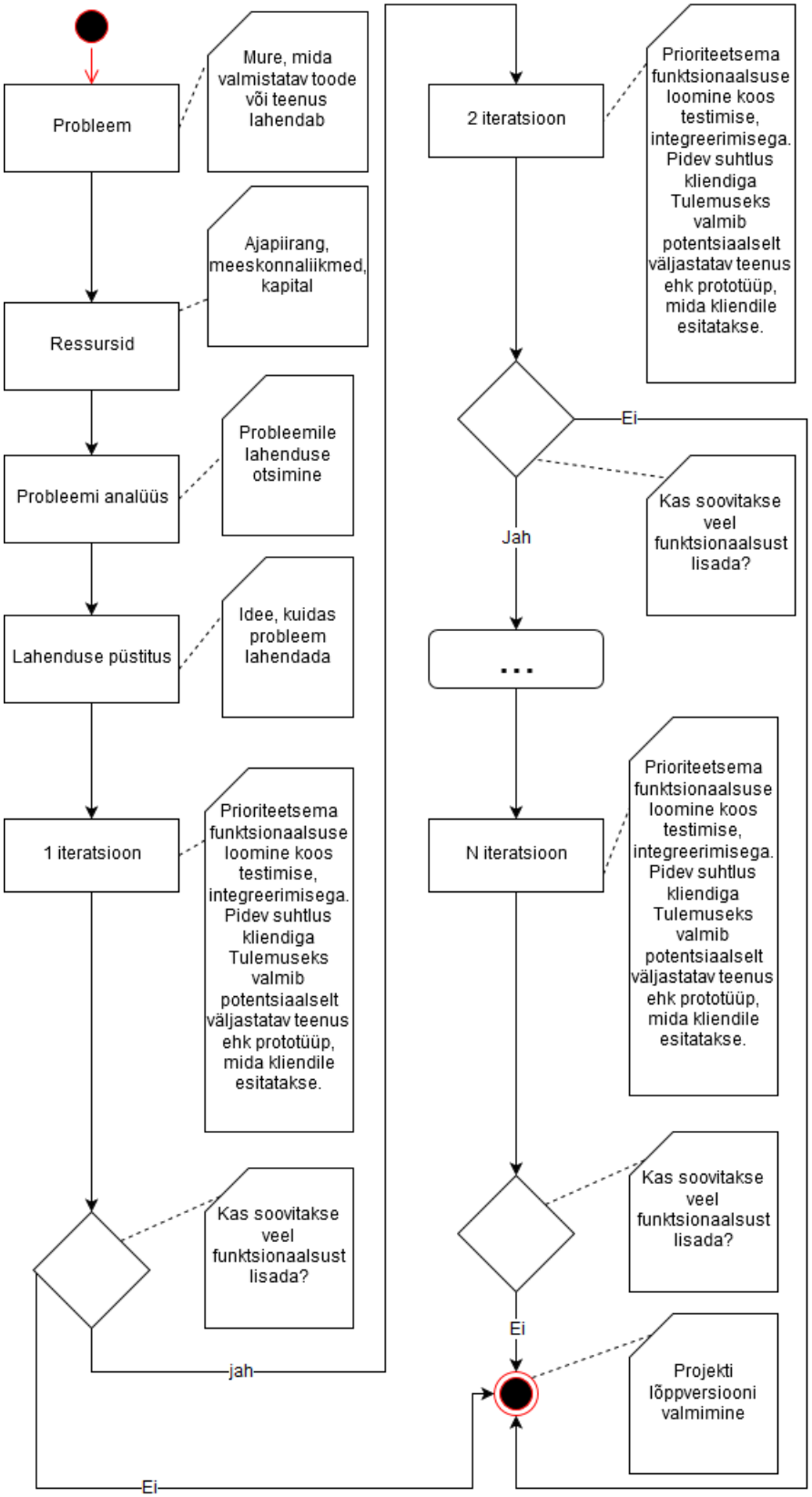
Selleks, et etappide kirjeldusi mõista, defineerin antud töö raames ära mõisted:

- Prototüüp- pooleliolev toode, millel on põhifunktsionaalsus olemas. Funktsionaalsuse lisamisel saab sellest toode.

- Iteratsioon- agiilse arenduse tsükkel, mis koosneb skriptimisest, testimisest, integreerimisest kliendiga pidevas kontaktis olles.

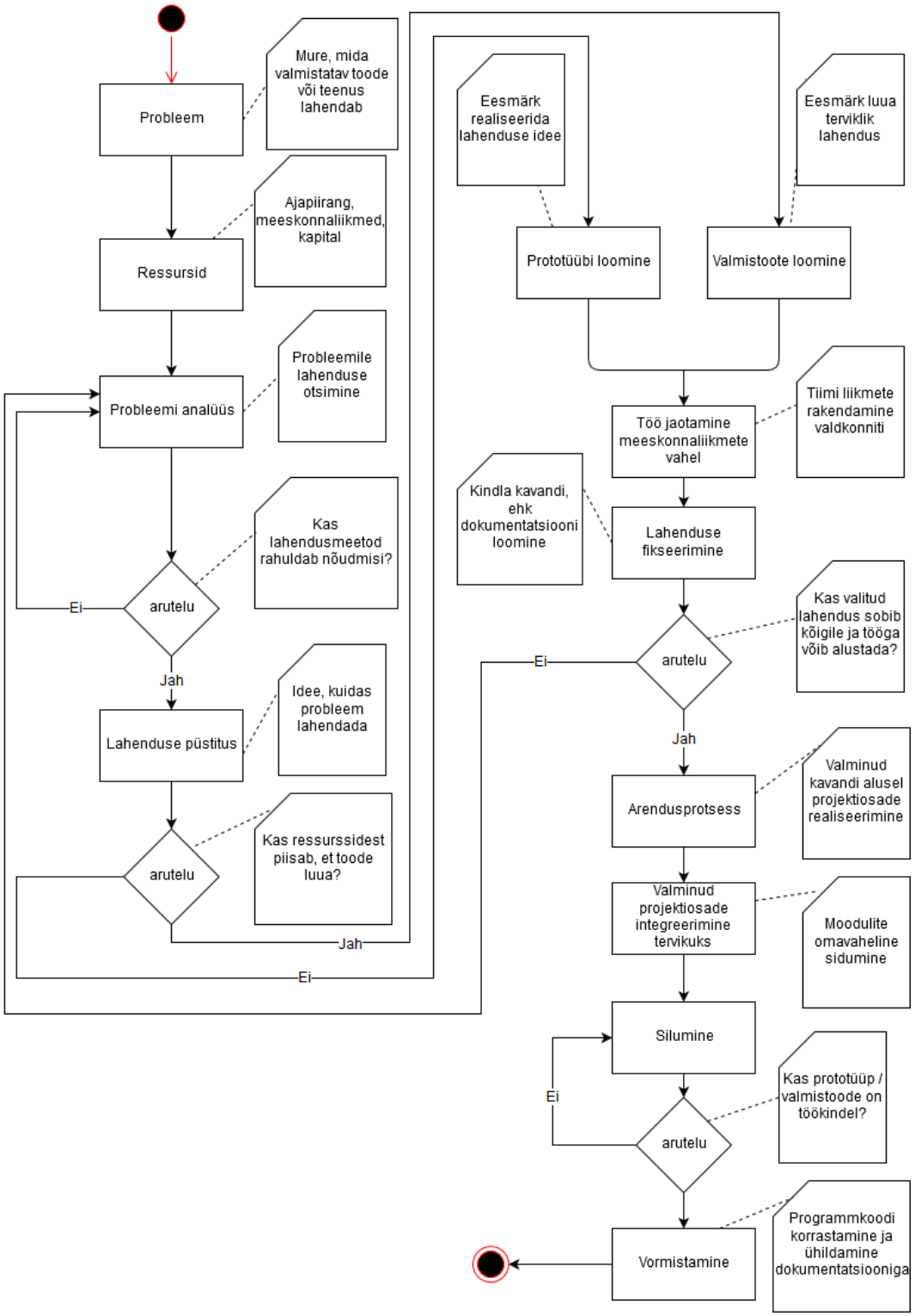
2.4.1 Projekti etapid kasutades agiilseid meetodeid

Agiilsete meetodite etapid on tihedalt kliendi soovidega seotud. Tellija määratleb, millist tulemust ta soovib ning kui palju ressursse projekti alla kulutada. Probleemi analüüs ning lahenduse püstitus valitakse koos kliendiga välja. Iteratsioonide käigus luuakse projekti osad, alustades põhifunktsionaalsusest. Põhifunktsionaalsus annab projektile vormi ja osa lahendusest, kuid tervikuna püstitatud probleemi see ei lahenda. Iteratsiooni sees testitakse ja integreeritakse loodud tarkvara ning ollakse pidevalt kontaktis kliendiga, kelle soovidega arvestatakse. Iga itearatsiooni lõpuks valmib prototüüp, mida saab tellija soovil turule paisata (Joonis 9). Pärast iga iteratsiooni esitletakse valmistatut kliendile ning arutatakse, kas loodud prototüüp kvalifitseerub lõpp-produktina või mitte. Kui soovitakse funktsionaalsust lisada, liigutakse uue iteratsiooni peale, kui klient jääb tulemusega rahule, loetakse projekt sooritatuks.



Joonis 9. Projekti etapid agiilse meetodi korral

2.4.2 Projekteerimise etapid kasutades kose meetodit



Joonis 10. Projekti etapid agiilse meetodi korral

Võrreldes agiilsete meetoditega, on kose meetodi puhul paikneb klient aredusprotsessist kaugemal. Tellija käest on tarvis teada valmistatava projekti nõudeid, mille alusel lahendusmeetod valida. Kui esitatud nõudeid täitev lahendus on püstitatud, võrreldakse seda meeskonna võimekusega: kui vahenditest jääb puudu, otsustatakse piirduda prototüübi loomisega. Seejärel jaotatakse projekt osadeks, mille alusel iga meeskonnaliiget rakendatakse. Enne arendusprotsessiga alguse tegemist fikseeritakse lahenduskäik- valitud meetodeid esitletakse koos eeldatava tulemusega kliendile. Fikseeritud lahenduse põhjal luuakse täpne dokumentatsioon. Kui lahenduskäik ning projekti eesmärk tellijale sobib, asutakse projekti arendama. Juhul kui klient valitud meetodit või tulemust ei aksepteeri, liigutakse tagasi probleemi analüüsima. Arendusprotsessis eri meeskonnaliikmete poolt loodu integreeritakse tervikuks, mille järel asutakse projekti tulemust siluma. Tulemusega rahule jäädes kohendatakse eelnevalt loodud dokumentatsioon loetavamaks, mile järel loetakse projekt sooritatuks (Joonis 10).

2.5 Tööjaotus

Tööjaotust saab meeskonnatöö puhul jaotada kahte suurde rühma: kus neile jagatud ülesandeid vaatlevad liikmed iseseisvalt või kui vaadeldakse terve projekti probleeme koos. Neid meetodeid iseloomustavad järgmised tunnused:

Tabel 3. Tööjaotuse tabel

Tööjaotus	Idee	Põhimõte	Jaotuse keerukohad	Tugevus
Eri valdkondade järgi	Projekti distributsioon segmentideks	Liikmetele sobiva projekti osa määramine	Meeskonna-liikmete oskuste analüüs	Liikmete efektiivne rakendamine
Ühistööna vaatlemine	Paralleerne projekti lahendamine	Kompromisside leidmine projekti igas osas	Kõigile sobiva aja ning suhtluskanali leidmine	Tõhusa lahendi saavutamine vähemate katsetega

2.5.1 Ülesannete jagamine eri valdkondade alusel

Üks võimalus meeskonnatööd jaotada, on anda igale liikmele talle sobiv ülesanne. See nõuab teadmisi iga meeskonnaliikme tugevuste ja nõrkuste kohta, et projekti osi efektiivselt osavõtjate vahel ära jaotada. Lisaks tuleb arvestada tööde osakaalu, et iga liige oleks piisavalt koormatud nii ülesannete keerukuselt kui ka ajaliselt. Enne tööle asumist on tarvis meeskonnaliikmeid analüüsida, et nende tugevusi ning nõrkusi tundma õppida. Eelkõige soodustab töö käiku see, kui isik on eelnevalt sarnaseid ülesandeid juba täitnud. Peale selle on tarvis uurida projekti ülesehitust, et seda eri mooduliteks jaotada. Näiteks kui projektis on tihedalt põimunud 2 funktsionaalsust, ei ole kasulik neid eri inimeste vahel jagada. Kui üks meeskonnaliige need kaks osa edukalt lahendab, on see põimik üks suur moodul, mida on hiljem kergem teiste osadega siduda. Kui hiljem peaks neid 2 funktsionaalsust omavahel siduma, ilmneks seal probleeme kuna eri isikute lahendused ei pruugi koheselt omavahel kokku klappida. Antud tööjaotuse eeliseks on induviduaalne töö, kus koosviibimine pole töö läbiviimiseks vajalik.

2.5.2 Ülesannete ühistööna vaatlemine

Teiseks variandiks on kogu projekti paralleerselt vaadelda. Selleks on kasutuses paarisprogrammeerimise võte, kus vaheldumisi 2 osapoolt programmkoodi kirjutavad. Pealtnäha tundub selline lähenemine asjatu tööjõu raiskamisena, kuid see viis aitab paremini puuduseid läbi näha, mis hoiab lõppkokkuvõttes aega kokku. Kõrvalvaatleja annab juurde ideid ning näeb võimalikke tekkivaid probleeme paremini läbi. Tänu sellele avastatakse vead juba algfaasis ning nende likvideerimine on kergem, kui puudused alles arenduse lõppfaasis väljenduksid. Panuse poolelt on sellisel viisil meeskonnaliikmed võrdsed. Projekti jagamisel osadeks ning meeskonnaliikmete rakendamisel ei ole tarvis suurt planeerimist ja analüüsi algfaasis läbi viia, kuna meeskonnatöö on kogu süsteemi loomise juures tihe. See annab võimaluse jooksvalt planeerimiseks ning pidev suhtlus tagab töö dünaamilise kulgemise. Sellise lähenemise puuduseks on ühise aja leidmine, kuna eduka suhtluse jaoks on parim viis olla füüsiliselt ühes ruumis, mitte sidevahendeid kasutada.

3 Projekt „tahvlikell“ teostus

Kuna tahvlikella puhul on tegu arvutustehnilise projektiga, kitsendas see lahendusmeetodite valikut. Agiilsed printsiibid on eelkõige tarkvaraarenduses rakendatavad, kuid peale tarkvaralise osa sisaldas tahvlikella projekt veel mehaanikat ning elektroonikat. Variatsiooni tekitamiseks osutus tarkvaraarenduses valituks agiilne printsiip, kuna soovisime projekti kui terviku käigus erinevaid lähenemesi katsetada. Tarkvaraarenduse käigus jaotasime tööülesandeid dünaamiliselt, ehk tegelesime kõik ühiselt iteratsioonide kallal.

Tahvlikella tarkvara loomiseks kasutasime agiilseid printsiipe, kuna soovisime varajases arendamise staadiumis ettenäidatavat tulemust saada. See oli vajalik, kuna projekti lahendamiseks oli aeg 1 semestriga piiratud. Otsustasime, et kose mudeli rakendamine 3-pealise meeskonna ning vahetus läheduses oleva tellijaga ei ole tarkvara osas mõistlik. Projekt sisaldas endas veel nii mehaanika kui ka elektroonika osi. Antud valdkondades ei saa agiilseid meetodeid rakendada, kuna need sisaldavad füüsilisi komponente, mille käigu pealt muutmine ei ole võimalik.

3.1 Projekti agiilne osa

Lahendatava probleemi tahvlikella projekti puhul võtab kokkku lause: kuidas kuvada kellaaega efektselt tahvli pinnale?

Kasutatavateks ressurssideks olid kolmeliikmeline meeskond, 100 euro piiresse jääv toote omahind ning ühe semestriga piiratud aeg.

Probleemi analüüsi käigus kaalusime erinevaid variante, kuidas projekti teostada. Esimeseks ideeks oli päikesekella ideel põhineva süsteemi loomine. Tööpõhimõte seisneb ümber posti paiknevate valgust eristavate dioodide rakendamisel, kus posti abil tahvlipinnale vari tekib. Idee ei läinud teostamisele, kuna minutite ja tundide kuvamiseks tuleb kasutada mitmekihilist disaini. Selle tulemusena muutuks toode suureks ning kogukaks, mis ei paistaks tahvlipinnal efektne välja. Tavaliselt paikneb tahvli kohal valgusti, mis põledes sellise kella töökõlbmatuks muudaks.

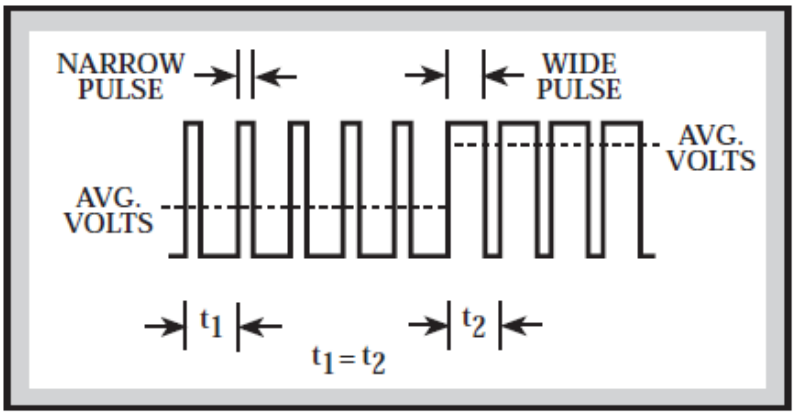
Järgmiseks lahendusvariandiks kaaluti loendamise põhist kellaaja esitamist. Kellaaja kuvamiseks kasutataks mooduleid, kus iga neist kuvab vastavalt eraldi minuteid, tunde või sekundeid. Disaini poolest sarnaneks moodulid doomino klotsidega, millel asetsevad LED lambid. Põlevate lampide arv kuvab vastavat kellaaja järgu väärtust. Kirjeldatud moodulid sobiks tahvli pinnale, kuna need on kerged ning magnetitega kinnitamine ja väike voolutarve muudaks need kergesti paigutatavaks. Antud kuvamismeetodi nõrkuseks on selle loetavus- süsteemi tundmatu vaatleja ei pruugi sellest esmapilgul aru saada.

Kandvaks ideeks jäi tahvli kui objekti täielik rakendamine. Kui saab tahvlipinnale vastava markeriga kirjutada ning lapiga seda maha kustutada, oleks efektseim lahendus selle protsessi automatiseerimine.

Lahenduse püstituseks osutuks valituks analüüsi käigus viimane kaalutud variant. Kirjutamise automatiseerimiseks mõeldi välja tahvlimarkeri tüürimiseks hoovastik, mille teekonda saab servomootoritega juhtida. Selleks, et püstitatud ideele tarkvaralisi lahendusi luua, oli tarvis platvorm välja valida. Kõigi pakutud ideede puhul kattus plaan kellaaja hankimisel. Taimereid kasutades tekiks pikas perspektiivis ebatäpsused. Selleks, et kõige väiksema nihkega kellaaega saaks kuvada, on tarvis seda läbi veebi hankida. Arvestades püstitatud toote omahinda ning aja- ja tööjõupiiranguid, valisime kliendiga koostöös platvormiks Iteaduino mikrokontrolleri. Kaalusime ka teisi platvorme, nagu Raspberry Pi miniarvutit või Arduino mikrokontrollereid. Miniarvuti tellimine oleks liialt palju aega enda alla võtnud. Valituks sai Iteaduino kontroller, kuna see oli Arduino seeria omadest kordades odavam. Laiendasime platvormi (Iteaduino v.2.2) WiFi mooduliga (ESP8266) ning servomootoritega (HX5010), et saaks tarkvaraarendusega algust teha. Laiendused olid tarviklikud süsteemile vastavalt sisendi ning väljundi teostamiseks.

Esimese iteratsioonil keskendusime numbrite kirjutamise funktsionaalsuse loomisele. Numbrite defineerimise jaoks lõime käepärastest vahenditest hoovastiku prototüübi, et servomootorite juhtimise põhjal markeri liikuvust ennustada. Iteratsioonide käigus suhtlesime pidevalt kliendiga. Tellija oli hoovastikuga rahul, sest see täitis liikuvuse nõudeid- selle abil sai markerit piisavalt liigutada, et sobiva suurusega numbreid tahvlile kirjutada. Esmatähtis oli platvormiga piisavalt tutvuda, et servomootorid vastavalt soovile liikuma panna. Selleks oli tarvis mikrokontrolleri dokumentatsiooniga tutvuda,

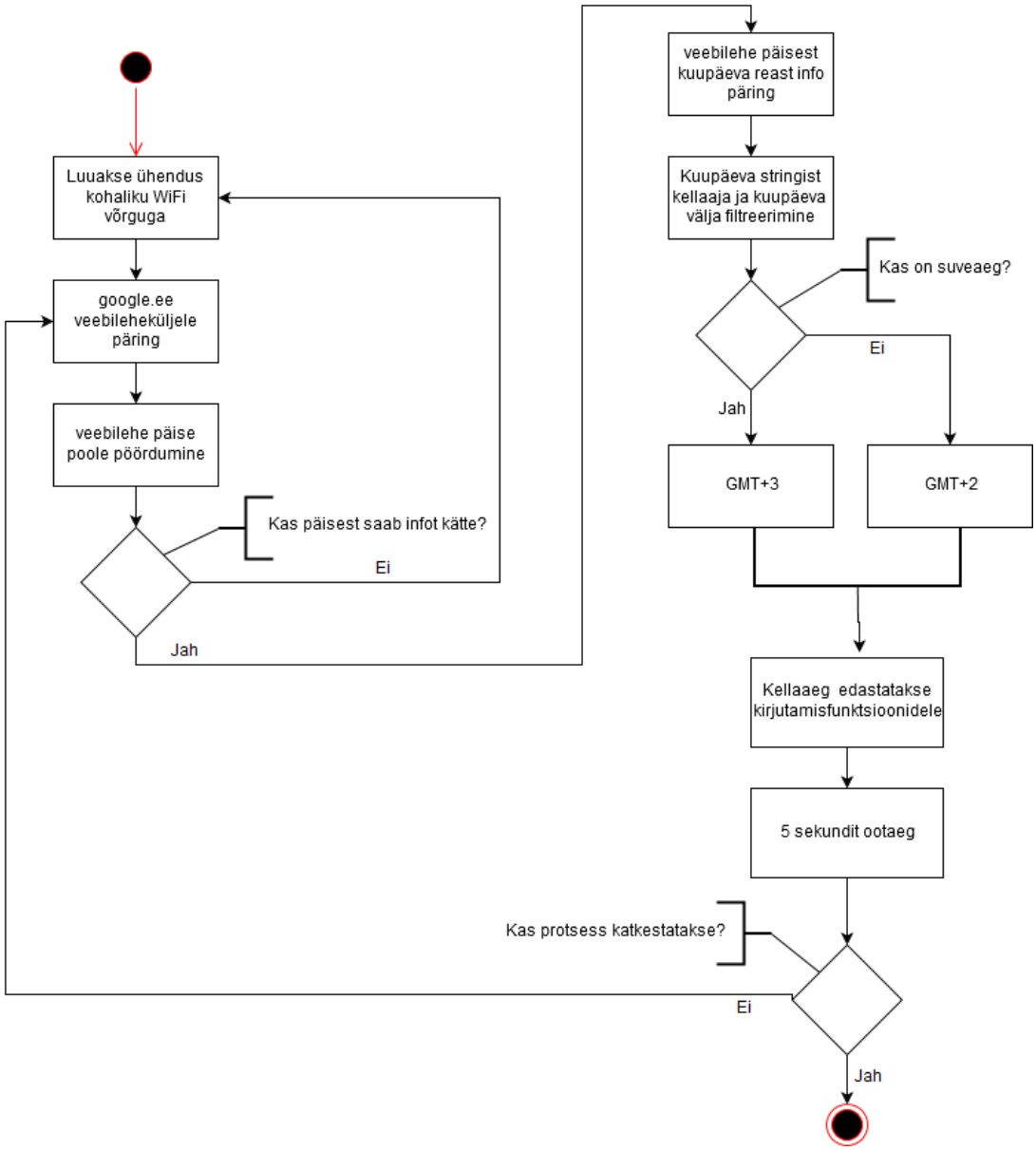
et sealseid võimalusi enda kasuks rakendada. Tutvusime erinevate servomootorite juhtimisviisidega, testides nii lineaarset, PFM ning PWM juhtimise meetodit (Joonis 11).



Joonis 11. PWM [14]

Kliendile sobis neist viimane, kuna see pakkus kõige energiasäästlikumat lahendust. Valitud juhtimismeetodit kasutades määratlesime ära markeri liikuvuse, nii et oleks võimalik numbreid kujutada. Sellega lugesime esimese iteratsiooni lõppenuks ning valminud oli tahvlikella esmane prototüüp, mis oli võimeline vertikaalsele tahvlipinnale numbreid kirjutama. Esitlesime tehtud tööd kliendile ning numbrite kuju, kirjutamise kiiruse ja suurusega jäi klient rahule.

Järgmise iteratsiooni käigus asusime funktsionaalsust lisama. Järgmine prioriteetne ülesanne oli kellaaega veebist pärida. Kellaaja pärimiseks oli mitmeid viise, mida kliendiga läbi arutasime. Üheks võimaluseks oleks olnud luua oma veebileht, kus kehtivat aega kuvada, mis mõnest usaldusväärsest portaalist pärineb. Selline lahendus lihtsustaks kellaaja edastamist kirjutajale, kuna üleliigne info puuduks. Kliendile antud lahendus ei sobinud, arvates et oma veebilehe haldamine vaid kellaaja pärimiseks on kulukas ning pole usaldusväärne. Selleks, et info pärimise võimalusi rohkem mõista, lõime veebilehele ühendumise funktsionaalsuse, teades et see on igal juhul vajalik kellaaja hankimiseks. Uurides lähemalt veebilehtede ülesehitust, selgus et iga portaali päises on globaalne aeg kuvatud. Pakkusime kliendile välja, et kellaaja saamiseks ühendume google.com veebisaidile, mille puhul on kättesaadavus pea 100 protsendiliselt tagatud. See idee sobis kliendile, kuna see rahuldas püstitatud nõudmist kellaaja allika kättesaadavusele. Idee teostamiseks korjasime veebilehe päisest infot ning stringitöötluse käigus eraldasime kellaaja muust infost (Joonis 12).



Joonis 12. WiFi mooduli päringu vooskeem

Pärast kellaaja pärimise funktsiooni testimist integreerisime selle kirjutusfunktsiooniga. Teise iteratsiooni lõpuks oli valminud tahvlikella prototüüp, mis oli võimeline automaatselt kirjutatavat väärtust veebist pärida ning see tahvlipinnale väljastada. Esitluse käigus ei esinenud kellaaja pärimises tõrkeid ning klient veendus, et selline lahendus on aktsepteeritav.

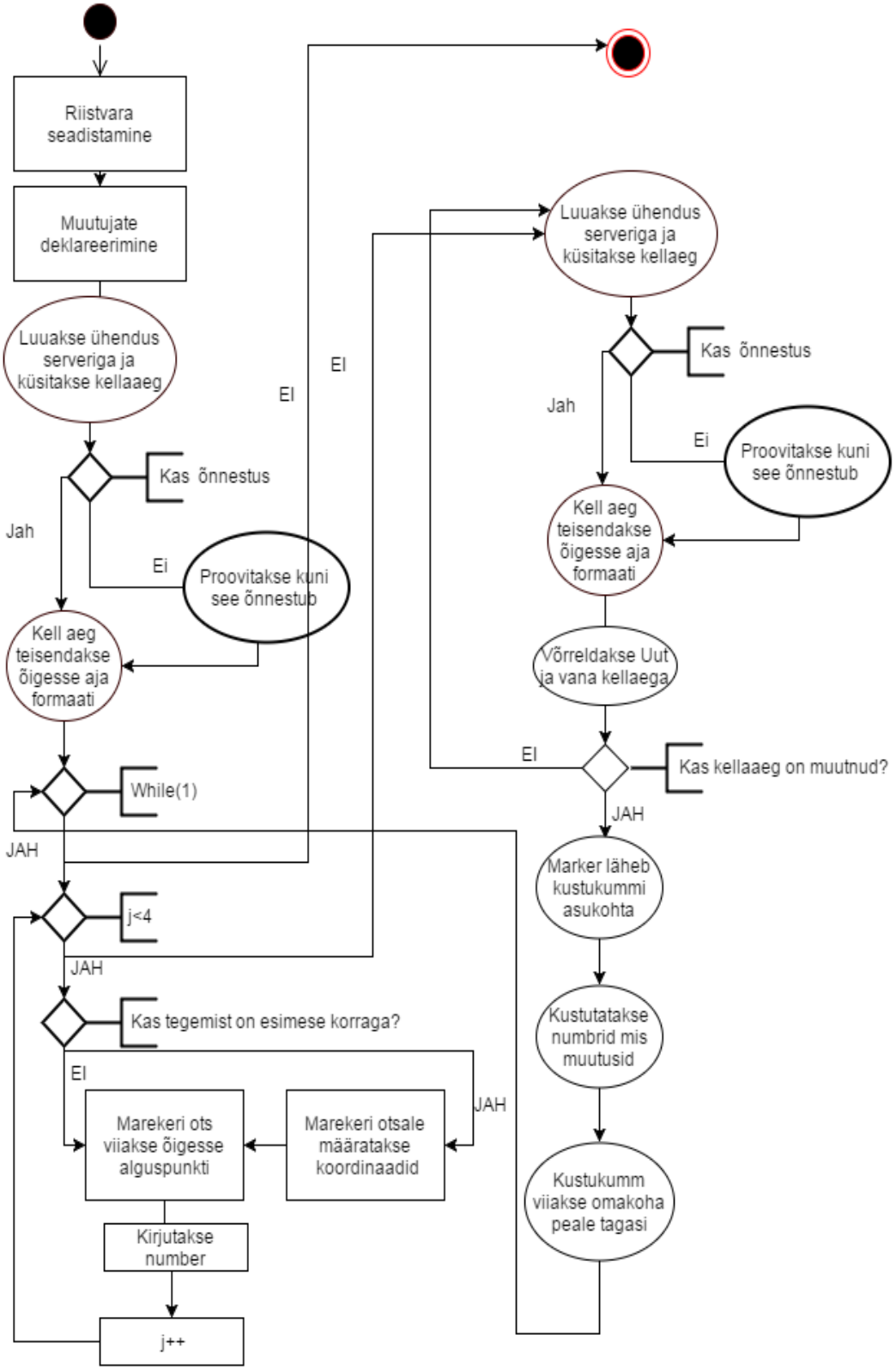
Kolmanda iteratsiooni eesmärgiks oli luua tõstmisfunktsioon, mis tekitaks numbritesse vahed ning mis oleks baasiks hiljem loodava kustutamisfunktsiooni arendamiseks. Kliendi üheks nõudeks oli kooloni paiknemine tunni ja minuti väärtuste vahel. Antud iteratsioon oli olemuselt otsene ning selle käigus pikemat arutelu kliendi ja meeskonna vahel ei toimnud. Testimise käigus ilmnes nähtus, kus servomootorid üritasid oma

positsiooni säilitada kui markeri ots oli tahvlist eemal ning tegid selle käigus programmis määratlemata liigutusi. Nüüd, kui kasutuses olid korraga 3 servomootorit, saime pinge mõõtmise tulemusena teada, et mikrokontrolleris tekib sel hetkel pingelang. Arvates, et see põhjustab määratlemata liigutusi, otsustasime probleemile elektroonika faasis keskenduda. Presentatsiooni käigus ilmnes sama nähtus ning selgitasime kliendile eeldatavat põhjust, miks see toimub. Tulemus oli sellest olenemata kättesaadav ning tulemus oli aksepteeritav. Kolmanda iteratsiooni lõpuks oli prototüüp võimeline

Joonis 13. Kirjutamisel tekkinud määratlemata liigutus

vahedega eraldatud kellaaega tahvlile autonoomselt joonistama, kuid mõnel kirjutamistsüklil võisid numbrid määratlemata liigutuste tõttu loetamatud olla (Joonis 13).

Neljas ehk viimane iteratsioon hõlmas endas kustutamisfunktsiooni loomist. Hoobade liikuvus ning kustutamislapi suurus näitas, et parim asukoht kustutamislapile oleks tundide ja minutite väärtuste vahel. Klient oli nõus kooloni nõudest loobuma, nähes et kustutuslappi ei ole mõistlik muule positsioonile määrata. Kustutamine oli esialgselt plaanis lahendada kogu numbrijada uuendamisega, kuid tellija arvates oleks selline viis liialt markeri tinti kulutanud ja sellega projekti toimimisaega oluliselt vähendanud. Seega rakendasime vaid muutuvate numbrite uuendamist. Testimisel selgus, et hoovastiku võnkumiste tõttu on keeruline igal ringil kustutuslappi haarata. Tarkvaraliselt lahendus toimis ning puudusi saaks mehaanikat ning elektroonikat täiendades leevendada. Esitlus kinnitas kliendile, et tahvlikella tööks vajalikud tarkvaralised lahendused olid loodud ning vajalikud nõuded sellega täidetud. Tellija luges tarkvara prototüüpimise lõppenuks ning kinnitas sellega programmi täisversiooni (Joonis 14).



Joonis 14. Tahvlikella tööpõhimõtte vooskeem

3.2 Projekti kose mudeli osa

Probleemiks mehaanika puhul, mida lahendada, oli tahvlimarkeri liikuvuse tagamine tahvlipinnal. Lisaks liikuvusele laienes probleem süsteemi tahvli külge kinnitamisele.

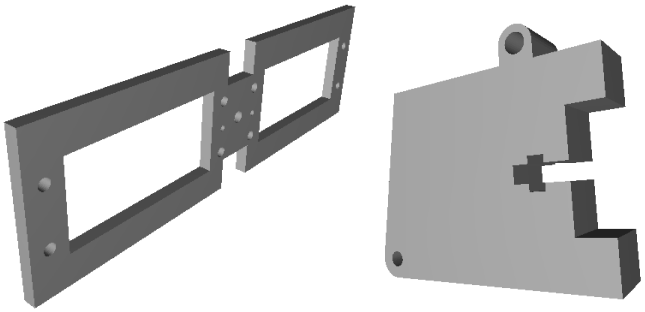
Ressurss mehaanika loomiseks oli 50 euroga piiratud kapital, 1 meeskonnaliige, kes disainimisega tegeles ning projekti lõpptähtajaga piiratud aeg.

Probleemi analüüsi käigus kogusime kliendilt nõudmisi, mida mehaanika pidi täitma. Nendeks nõudmisteks olid kompaktne disain ning käepärane tootmismeetod. Kaalusime erinevaid meetodeid, kuidas tahvlikella mehaanikat valmistada. Algne plaan oli disainitud korpust ja hoovastikku vineerist toota. Selline lähenemine oleks hulgaliselt käsitööd, materjali ning tööriistu nõudnud.

Lahenduse püstitus kujutas endas modernse tehnoloogia kasutamist. Otsustasime mehaanika tootmiseks 3D printimise tehnoloogiat rakendada. Materjalide hindade järgi seadsime ressurssidest tuleneva piiri, et disaini valmistamiseks saame kasutada maksimaalselt 100 grammi materjali. Sellise lähenemisega seadsime **eesmärgiks** luua mehaanika lõppversioon, mitte piirduda vaid prototüübiga.

Töö jaotasime meeskonnaliikmete vahel järgmiselt: liige 1 tegeles modelleerimisega ning konseptsiooni välja mõtlemisega. Liige 2 kaardistas kliendilt nõudmisi ning fikseeris nende alusel lahenduse käigu. Liige 3 pakkus tuge loodud lahenduste arvustamisega, et hilisemas staadiumis vigu ära hoida.

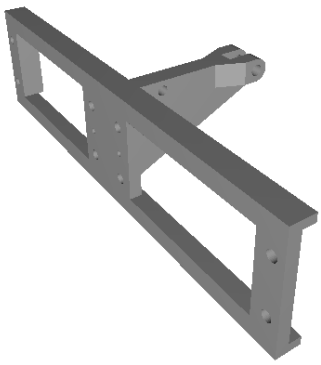
Enne **lahenduse fikseerimist** arutasime disaini üle: esimesed visandid detailidest kujutasid endast küll 3D prinditavaid osi, kuid selline lahendus sisaldas endas palju detaile, mida oleks pidanud omavahel kruvidega kinnitama. Selline lahendus oleks kogu mehaanika lõtkude tõttu ebakindlaks muutnud (Joonis 15).



Joonis 15. Tõstuki detailide disain lõigatavast materjalist valmistamiseks

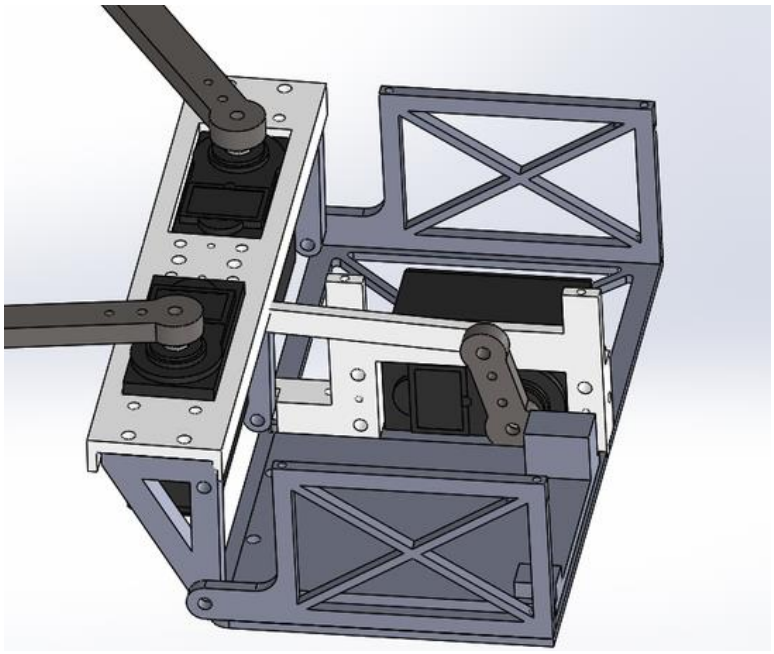
Uues eskiisis planeerisime modelleerida terviklikke detaile, et saavutada vajalikku jäikust ja vähendada materjalikulu. Selline lahendus oli kõigile osapooltele sobiv ning alustasime selle alusel modelleerimist.

Arendusprotsessi käigus lõime vajalikud detailid ning printisime need välja. Komplekteerisime mehaanika ning testisime selle töökindlust. Selgus, et tõstukiosa on ebakindel, kuna see on korpusega ühest punktist seotud (Joonis 16).



Joonis 16. Ebastabiilseks osutunud tõstuk

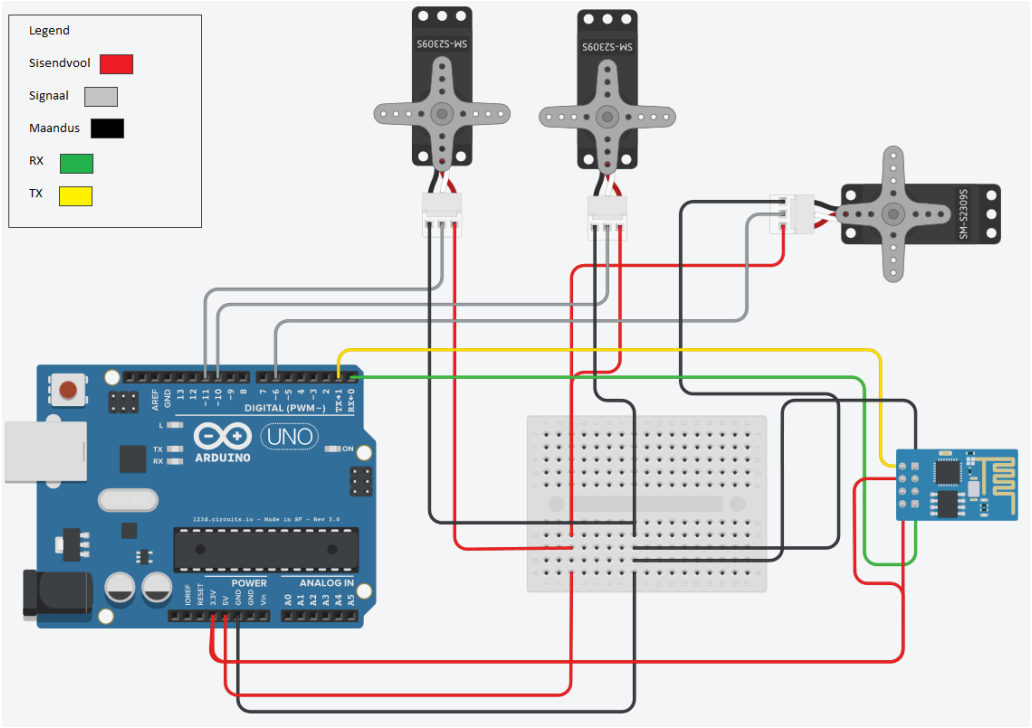
Silumise käigus muutsime tõstukiosa disaini, et see kinnituks korpusega kolmest punktist (Joonis 17). Täiustatud mehaanika oli stabiilsem ning see muutis loodu töökindlaks.



Joonis 17. Kolmest punktist korpusele kinnituv tõstukiosa

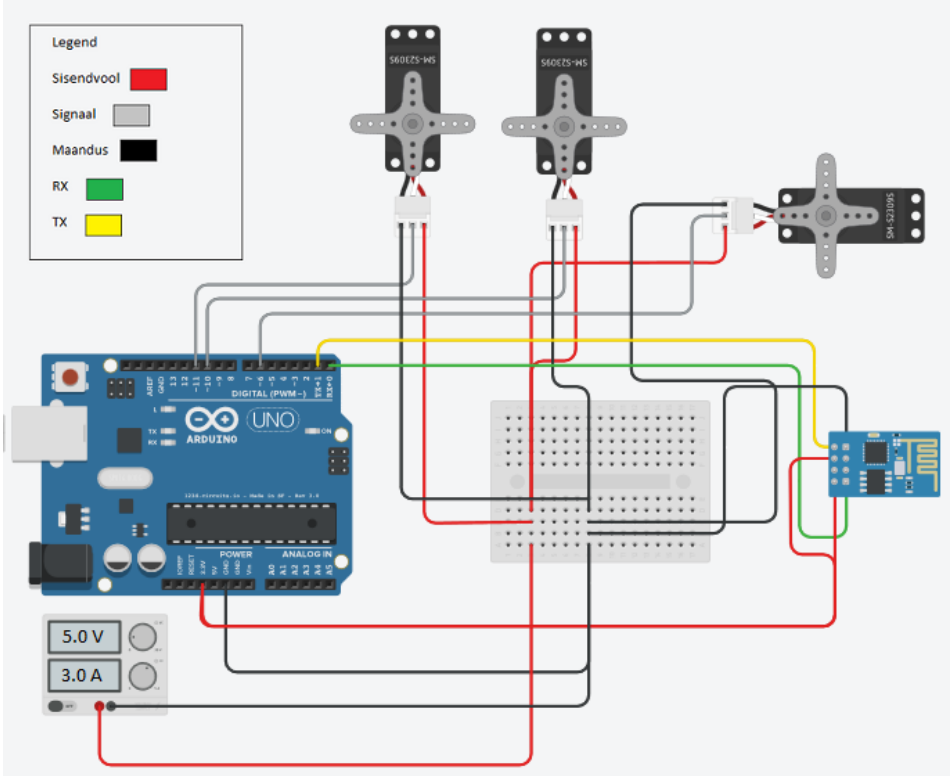
Vormistamise käigus lisasime igale detailile nimetuse, korrastasime mudelites defineeritud mõõtmeid. Sellega kinnitasime mehaanika valminud lõppversiooni.

Asendasime ajutise mehaanika prototüübi, mida tarkvara loomiseks rakendasime, mehaanika lõppversiooniga. Peale mehaanika komplekteerimise oli tarvis ka elektroonikamoodulid omavahel siduda. Algne ühendusskeem kasutas servomootorite toiteks mikrokontrolleri viike (Joonis 18). Tarkvara testides selgus, et sellisel viisil tekib mikrokontrolleris pingelang, mis projekti tööd häirib.



Joonis 18. Tahvlikella moodulite ühendusskeem

Kolme servomootori korraga tööle rakendamine nõuab suuremat voolutugevust kui mikrokontroller viikudest väljastab. Lõime toitemooduli, kus mikrokontroller ning servomootorid saavad oma energiat eri kanaleid pidi, et pingelangu vähendada (Joonis 19). WiFi mooduli puhul ei teki käivitamise hetkel nii suurt voolutarbe kasvu kui mootorite puhul. Seega hangib WiFi moodul oma toite endiselt mikrokontrollerilt, ilma et see viimase tööd häiriks. Rakendades mehaanika lõppversiooni ning uut ühendusskeemi selgus, et liigutuste ebatäpsused muutusid väiksemaks, kuid ei kadunud täielikult. Uute mõõtmiste tulemusena veendusime, et pingelangu enam ei teki.



Joonis 19. Uus ühendusskeem

3.3 Tahvlikella projekti analüüs

Tahvlikella kui õppeprojektiga jäime rahule, kuna see õpetas meeskonnas töötamist, erinevate arendusmeetodite rakendamist ning andis kogemust tarkvaraarenduses, modelleerimises.

Analüüsides tahvlikella projekti lõpptulemust, selgus, et servomootorite kasutamine sellisel kujul ei ole parim lahendus. Vertikaalasendis paiknevad pikad hoovad omavad inertsi, kui need ei puutu vastu tahvlipinda. Servomootorid üritavad oma asukohta säilitada, kuid kiire suuna- ja kiirusemuutuse tõttu „võngub“ mehaanika ning servomootorid üritavad seda tasakaalustada. Sellest tulenesid ka liigutuste ebatäpsused tahvlikella lõppversioonis. Kasutades servomootoreid, esineb tahvlikella töös aeg-ajalt tõrkeid ning ebatäpsusi kustutamislapi haaramisel. Sobilikum on sellise projekti puhul steppermootoreid kasutada.

Kokkuvõte

Töö eesmärgiks oli anda ülevaade projektide loomisel läbitavatest etappidest ning läbi selle pakkuda juhend- ja näidismaterjali kursuseprojektide lahendamiseks. Teise eesmärgina sai vaadeldud tahvlikella projekti, et selle alusel erinevaid arendusmeetodeid rakendada.

Dokumendi esimesed peatükid annavad ülevaate kursuseprojektide olemusest, kasulikkusest ning tutvustavad levinud arendusmeetodeid. Viimases osas rakendati teooriat projekti käigus, kus on kasutatud nii agiilset kui ka kose mudeli meetodit.

Käesoleva töö lõpptulemusena loodud tahvlikella projekt töötab ning seega on valitud lahendusmeetodid püstitatud nõuetele vastavad. Analüüsides saavutatud lõpptulemust, selgub et edaspidi on võimalik tahvlikella täiendada, et selle töökindlust parendada.

Kasutatud kirjandus

[1] EMSL, “Koosta oma projektitaotlus,” in *Väike käsiraamat alustajatele*, S. Kivimäe, Ed., Tallinn, Katrin Kala, 2003, p. 3.

[2] D. Carr, “Make Sure Your Project Goals are SMART,” The Project Management Hut, 2007.

[3] A. instituut, *IAS1410 Tarkvaraprojekt õppeaine sisu lühikirjeldus*, TTÜ õppeinfosüsteem, 2017.

[4] K. Beck, J. Grenning, R. Martin, M. Beedle, J. Highsmith, S. Mellor, A. van Bennekum, A. Hunt, K. Schwaber, A. Cockburn, R. Jeffries, J. Sutherland, W. Cunningham, J. Kern, D. Thomas, M. Fowler and B. Marick, “Manifesto for Agile Software Development,” 11 February 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed 10 April 2017].

[5] K. Beck, J. Grenning, R. Martin, M. Beedle, J. Highsmith, S. Mellor, A. van Bennekum, A. Hunt, K. Schwaber, A. Cockburn, R. Jeffries, J. Sutherland, W. Cunningham, J. Kern, D. Thomas, M. Fowler and B. Marick, “Principles behind the Agile Manifesto,” 11 February 2001. [Online]. Available: http://agilemanifesto.org/principles.html. [Accessed 10 April 2017].

[6] L. Milovanov, “Agile Software Development in an Academic Environment,” Painosalama Oy, Abo, 2006.

[7] M. Cohn, “Scrum Methodology and Project Management,” Mountain Goat Software, 12 November 2009. [Online]. Available: https://www.mountaingoatsoftware.com/agile/scrum. [Accessed 11 April 2017].

[8] M. Cohn, “What is a Scrum Master?,” Mountain Goat Software, 11 November 2009. [Online]. Available: https://www.mountaingoatsoftware.com/agile/scrum/roles/scrummaster. [Accessed 11 April 2017].

[9] K. Beck, Extreme Programming Explained, Indianapolis, Indiana: Addison-Wesley, 2000, pp. 29-33.

[10] W. Pierce, “Atlaz,” Atlaz Inc., 14 05 2016. [Online]. Available: https://atlaz.io/blog/waterfall-principles/. [Accessed 2017 05 03].

[11] E. Lynch, “Laughing Squid,” Laughing Squid LLC, 19 February 2014. [Online]. Available: https://laughingsquid.com/plotclock-a-robotic-clock-that-writes-the-time-with-a-marker/. [Accessed 17 March 2017].

[12] B. Webb, “Space Archive,” Brian Webb, 6 March 2016. [Online]. Available: http://www.spacearchive.info/military.htm. [Accessed 17 March 2017].

[13] W. W, “Banana Robotics,” Williams and Wang LLC, 11 11 2013. [Online]. Available: https://www.bananarobotics.com/shop/Iteaduino-v2.2. [Accessed 18 03 2017].

[14] I. Studio, “ITead Studio,” ITEAD Intelligent Systems Co.Ltd., 8 August 2011. [Online]. Available: http://store.iteadstudio.com/images/produce/Platform/ArduinoCom/Iteaduino2/iteadsuinov2.0_DS.pdf. [Accessed 18 03 2017].

[15] I. Studio, “ITead Studio,” ITEAD Intelligent Systems Co.Ltd., 11 11 2012. [Online]. Available: ftp://imall.iteadstudio.com/Mainboard/IM120905006_Iteaduino_UNO/DS_IM120905006_IteaduinoUNO.pdf. [Accessed 18 03 2017].

[16] B. E. Company, “Baldor- A member of the ABB group,” Baldor Electric Company, 12 August 1998. [Online]. Available: http://www.baldor.com/Shared/manuals/1205-394.pdf. [Accessed 21 March 2017].

[17] K. Taveter, *Tarkvara arendusprotsess ja tarkvaratehnika eetika,* Tallinn: Tallinna Tehnikaülikool, 2015.

[18] D. McCandless, “Liquidhub,” THINK Interactive Inc., 06 07 2013. [Online]. Available: http://www.thinkinc.com/wp-content/uploads/2013/07/Agile-Development-diagram_03.png. [Accessed 03 05 2017].

[19] I. B. Setiawan, “IT Project Management,” itpmpro inc., 03 07 2007. [Online]. Available: http://itpmpro.blogspot.com.ee/2007/07/key-activities-of-project-analysis.html. [Accessed 03 05 2017].

[20] Digital Humanities, “Digital Humanities,” Digital Humanities at Berkeley , 04 12 2013. [Online]. Available: http://digitalhumanities.berkeley.edu/blog/13/12/04/choosing-platform-your-project-website. [Accessed 03 05 2017].

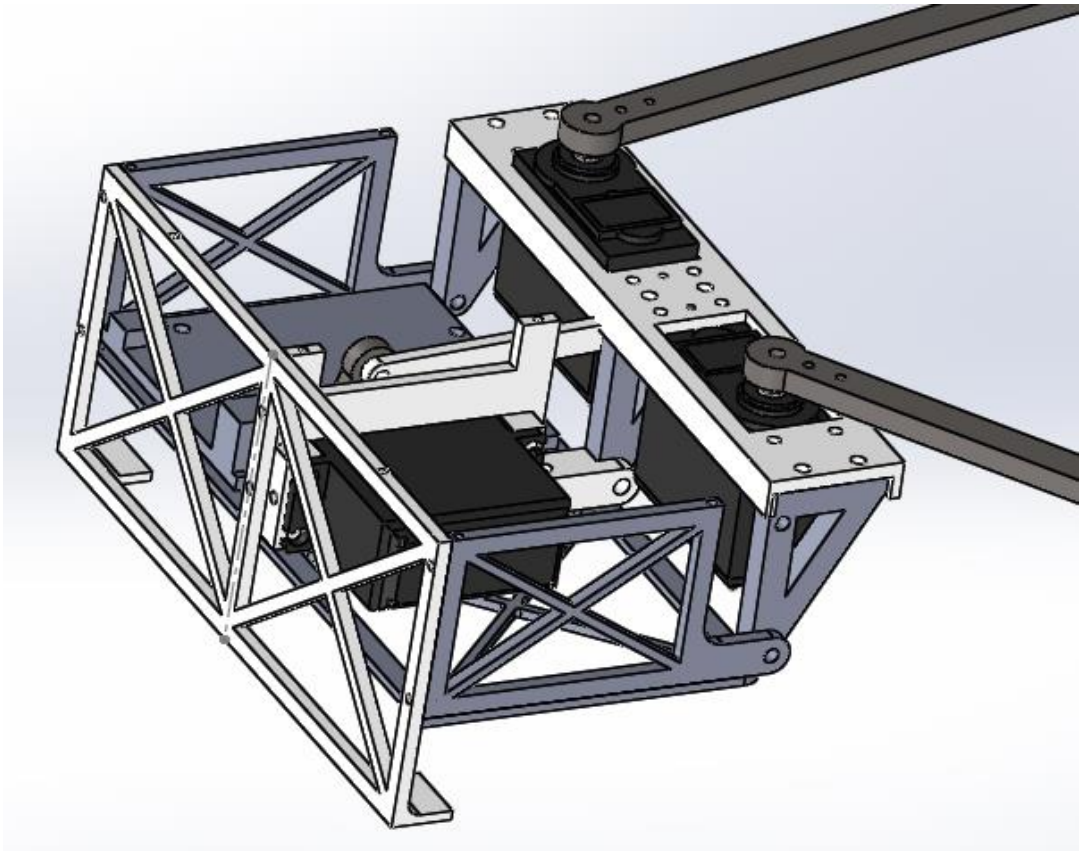
[21] T. Mochal, “How to design a prototype to fit the project approach,” CBS Interactive, 11 11 2002. [Online]. Available: http://www.techrepublic.com/article/how-to-design-a-prototype-to-fit-the-project-approach/. [Accessed 03 05 2017].

[22] Software Testing Fundamentals, “Unit Testing,” Software Testing Fundamentals (STF), 08 12 2010. [Online]. Available: http://softwaretestingfundamentals.com/unit-testing/. [Accessed 03 05 2017].

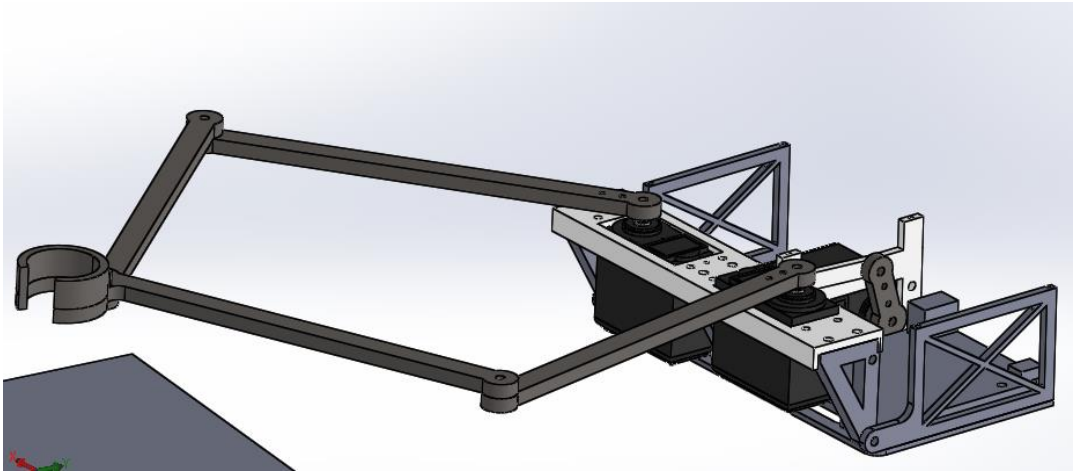
[23] Software Testing Help, “How to Plan and Manage Testing Projects Effectively (Tips),” Software Testing Help, 17 04 2017. [Online]. Available: http://www.softwaretestinghelp.com/test-project-planning/. [Accessed 03 05 2017].

[24] N. C. Zakas, “Why Coding Style Matters,” Smashing Magazine, 25 10 2012. [Online]. Available: https://www.smashingmagazine.com/2012/10/why-coding-style-matters/. [Accessed 03 05 2017].

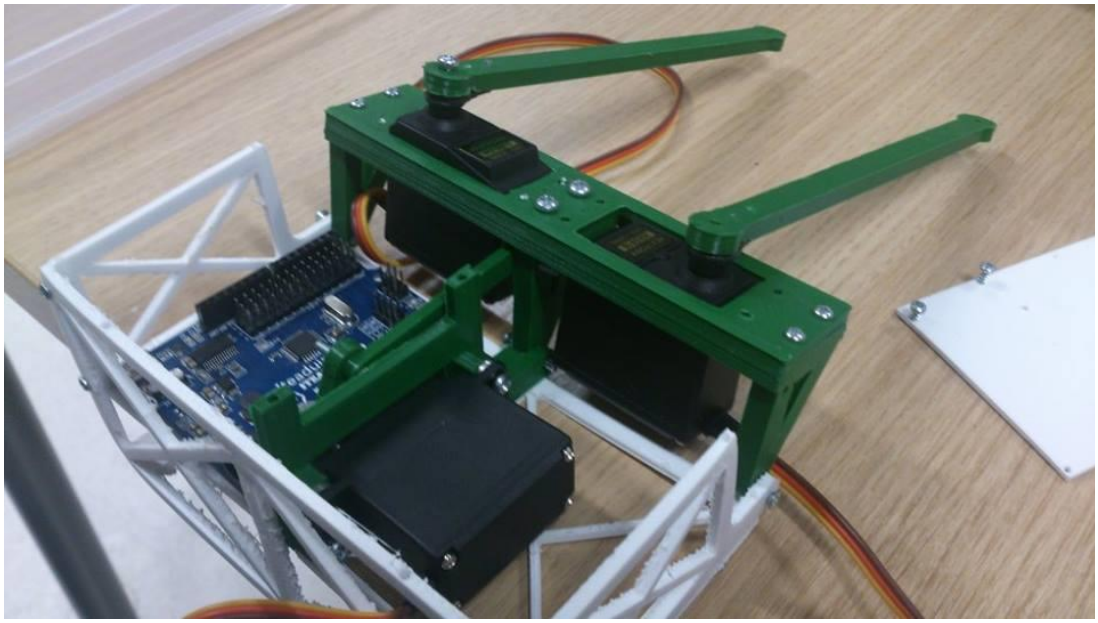
Lisa 1 – Tahvlikella pildid



Joonis 20. Modelleeritud tahvlikella korpus koos tõstukiga

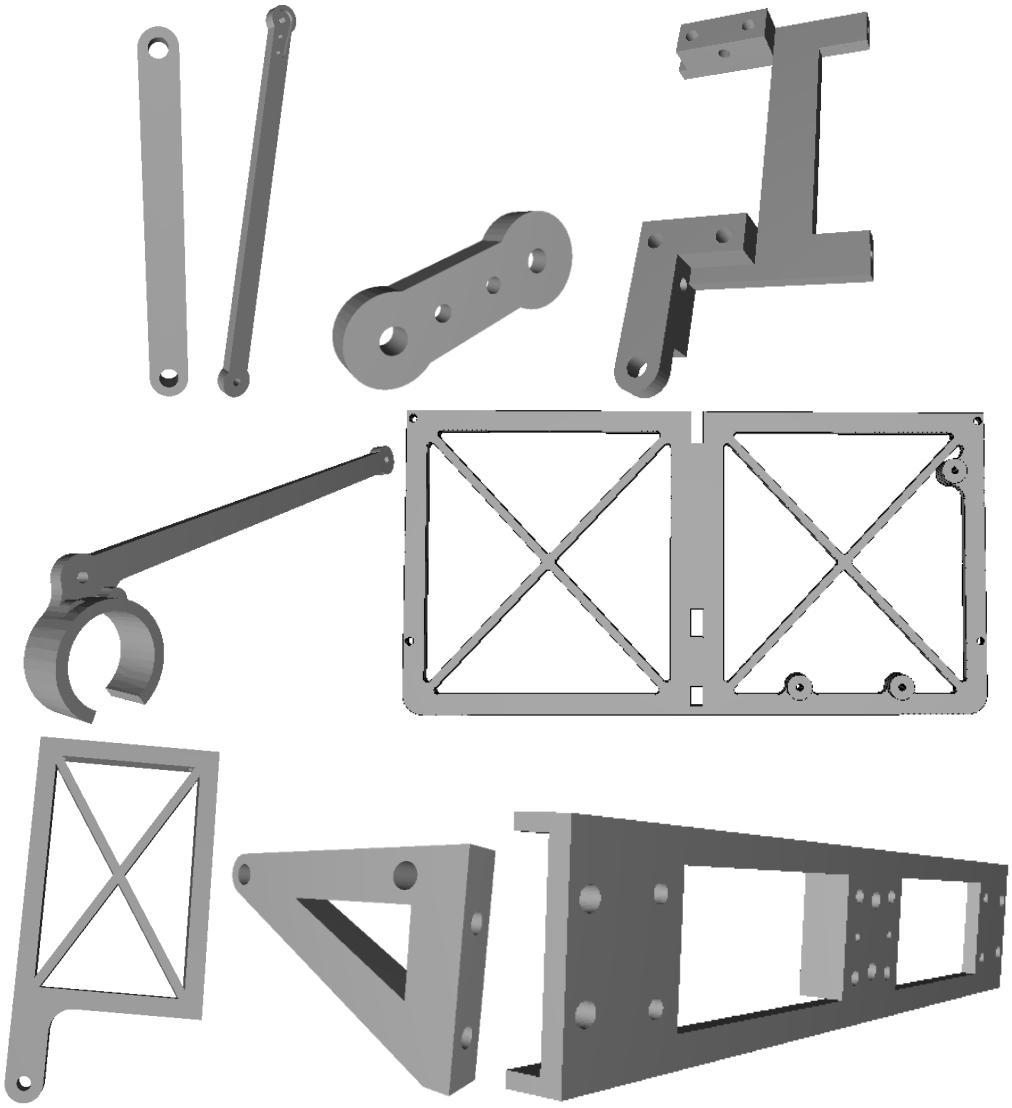


Joonis 21. Modelleeritud tahvlikella hoovastik



Joonis 22. Reaalne tahvlikell

Joonis 23. Kella kinnitumine tahvlile



Joonis 24. Tahvlikella mehaanikas kasutatavad erinevad detailid