TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Siim Romanov 132552IAPM

KUUPSATELIIDI MISSIOONIJUHTIMISTARKVARA ARHITEKTUUR

Magistritöö

Juhendaja: Evelin Halling, MSc

Tallinn 2017

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: Siim Romanov

15.05.2017

Annotatsioon

Töö eesmärgiks on välja töötada TTÜ tudengisatelliidi programmi raames loodava 1U kuupsatelliidi missioonijuhtimistarkvara tehniline lahendus.

Töö käigus analüüsitakse juba olemasolevaid tarkvaralahendusi ja nende sobilikkust programmi.

Käesoleva töö tulemusena valmib missioonijuhtimistarkvara tehniline lahendus ning selgitav analüüs tehtud valikutest lahenduse loomisel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 4 peatükki, 22 joonist, 1 tabelit.

Abstract

Architecture of CubeSat mission control system

The main purpose of this thesis is to develop mission control software architectural solution for TUT student satellite program, which would fulfil the requirements specified in system specification.

Thesis will give general overview of CubeSat missions and will consider existing solutions, which are used in other projects and their pros and cons. As result of this thesis, there will be architectural solution for mission control software with development and integration process.

Developing mission control software is not an easy task. Most of universities are interested to test some new scientific solutions in space environment. Also, often one purpose of CubeSat projects is to get full development process experience. As every university, CubeSat has its own main purpose and is assembled of different special hardware components, leads to often decision to develop special software also. Although there are some core standards which can be used in certain levels.

The thesis is in Estonian and contains 28 pages of text, 4 chapters, 22 figures, 1 table.

Lühendite ja mõistete sõnastik

1U	Kuupsatelliitide standardsuurus
ADCS	*Attitude Determenination and Control System*, asendi tuvastuse ja kontrollisüsteem
API	*Application Programming Interface*, programmiliides mille abil saab liidestada tarkvaraga uusi komponente
bps	*Bits per Second,* bitti sekundis, andmeedastus kiiruse mõõtühik
DSL	*Domain-specific language*, domeenispetsfiifiline keel.
EPS	*Electrical Power Supply*, toitesüsteem
Git	Hajutatud versioonihaldustarkvara
Github	Veebikeskkond, mis pakub majutust Git repositooriumidele ning koostöö tööriistasid
JMS	*Java Message Service*, Java sõnumvahetus standard
MCS	*Mission control system*, missioonijuhtimistarkvara
OBC	*Onboard Computer*, satelliidil paiknev arvuti
SGP4	*Simplified perturbations models*, lihtsustatud häirituse mudelid
SPA	*Single Page Application*, rakendus, kus kogu kasutajapoolne rakenduse osa on laetud brauserisse ühe lehe laadimisega.
TCP/IP	*Transmission Control Protocol/Internet Protocol*, arvutivõrkudes enim kasutusel olev internetiprotokollistik
TLE	*Two-line element set*, kaherealine andmeformaat objekti asukoha kirjeldamiseks kosmoses
UHF	*Ultra high frequencies*, ultrakõrgsagedused

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Järjest enam eraettevõtteid ning ülikoole avastavad kosmose, kui potentsiaalse äri-, arendus- ja teadusvaldkonna ning saadavad maa orbiidile enda poolt arendatud tehiskaaslasi. Kosmosetehnoloogia odavnemise ja regulatsioonide vabamaks muutumise tõttu ei ole maa tehiskaaslased enam ammu ainult väheste suurriikide ja riiklike organisatsioonide privileeg. Selliste tehiskaaslaste *defacto* standardiks on saanud kuupsatelliidid, mis võimaldavad väikeste kuludega saata kosmossesse pisikesi satelliite. Kuigi kuupsatelliit on oma mõõtmetelt väike, võimaldab see teadusasutustel testida uute tehnoloogiate toimimist ja vastupidavust kosmose erilistes tingimustes. Samas on siiski tegemist täiesti iseseisva tehiskaaslasega, mille tööd on tarvis maapinnalt jälgida ja juhtida.

Töö peamiseks eesmärgiks on uurida ja luua selliseks otstarbeks vajalikku tarkvara, mis täidaks TTÜ satelliidiprogrammis missioonijuhtimistarkvarale esitatud nõuded [1]. Käesoleva töö raames ei looda kogu süsteemi kõiki mooduleid, vaid töötatakse välja üldine arhitektuuriline lahendus ning luuakse olulisemad komponendid.

Töö esimene peatükk annab ülevaate kuupsatelliitidest ja nende ajaloost. Peatükis vaadatakse lähemalt, mis kuupsatelliit on ning milliseid erinevaid kuupsatelliite eksisteerib. Lisaks antakse statistiline ülevaade viimaste kümnendite jooksul läbi viidud projektidest, nende edukusest ja eesmärkidest satelliite arendatavate organisatsioonide valdkondade kaupa.

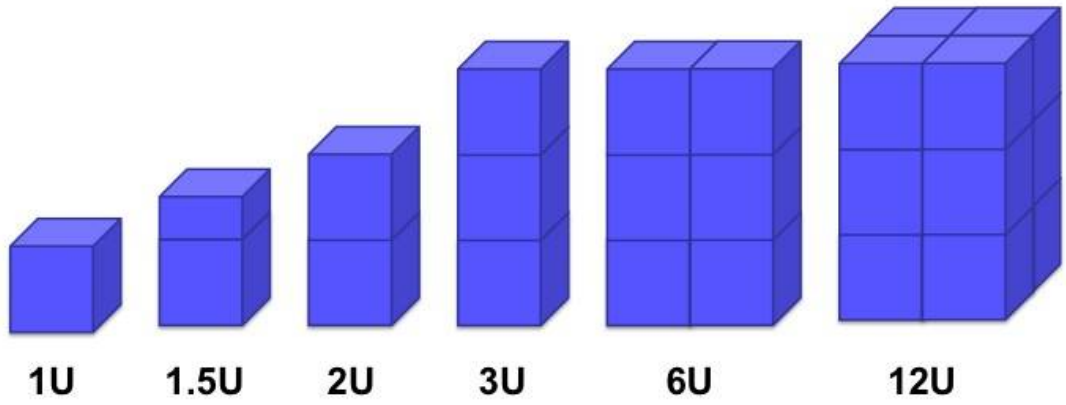
Teises peatükis kirjeldatakse lühidalt TTÜ tudengisatelliiti ja selle projekti eesmärki.

Töö kolmas peatükk vaatab olemasolevaid missioonijuhtimistarkvarasid. Tutvutakse nende eesmärkide, plusside ja miinustega.

Neljas peatükk keskendub TTÜ tudengisatelliidi missioonijuhtimistarkvara arhitektuurile. Peatükis on moodulite kaupa kirjeldatud ära erinevad tarkvara moodulid, nende toimise põhimõtted ja tehnilised detailid.

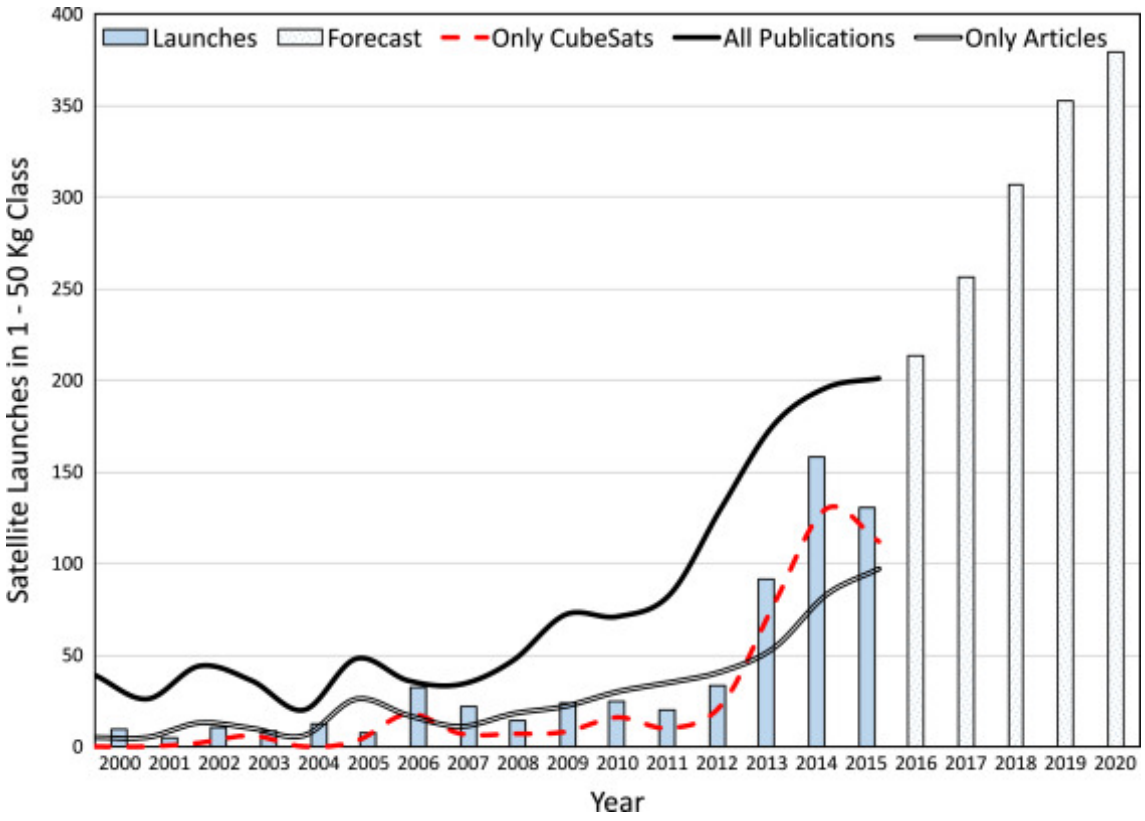
1 Kuupsatelliidid

Kuupsatelliitideks nimetatakse standardiseeritud suuruse ja kujuga nanosatelliite. Kuupsatelliitide suuruse referentsiks kasutatakse kõige väiksema kuupsatelliidi 1U suurust, mis on 10x10x10 cm [2].



Joonis 1. Kuupsatelliitide suurus [2]

Kuigi kuupsatelliidid on viimastel aastatel suurt populaarsust kogunud, siis tegemist ei ole uue asjaga, vaid standard töötati välja juba 1999. aastal Stanfordi ülikooli teadlaste poolt [3]. Esimesed kuupsatelliidid saadeti teele 2003. aastal Eurokot nimelise kanderaketiga [4]. Eesti esimene satelliit lendas kosmosesse 2013. aastal [5].



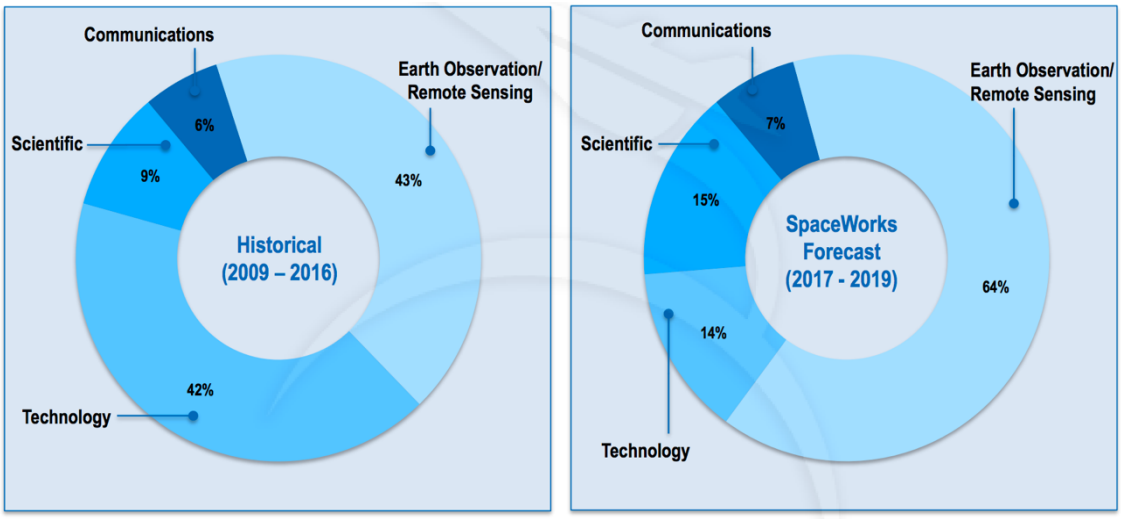
Joonis 2. 1 - 50 kg satelliitide startide statistika ja prognoos [6]

2016. aastal saadeti üles 101 nano- ja mikrosatelliiti ning statistika näitab kasvutrendi, hoolimata vahepealsetest väikestest langustest. SpaceWorks[1] ennustab, et perioodil 2017. a kuni 2023. a saadetakse üles ligi 2400 nano- ja mikrosatelliiti [7].

Kasvutrendi näitab ka kommertssektori poolt ülessaadevate satelliite osakaal. Kui perioodil 2009. a kuni 2016. a oli kommertssektori osakaaluks 40%, siis järgnevateks aastataks ennustaks ligi 30% kasvu akadeemilise ja avaliku sektori arvelt. [7]

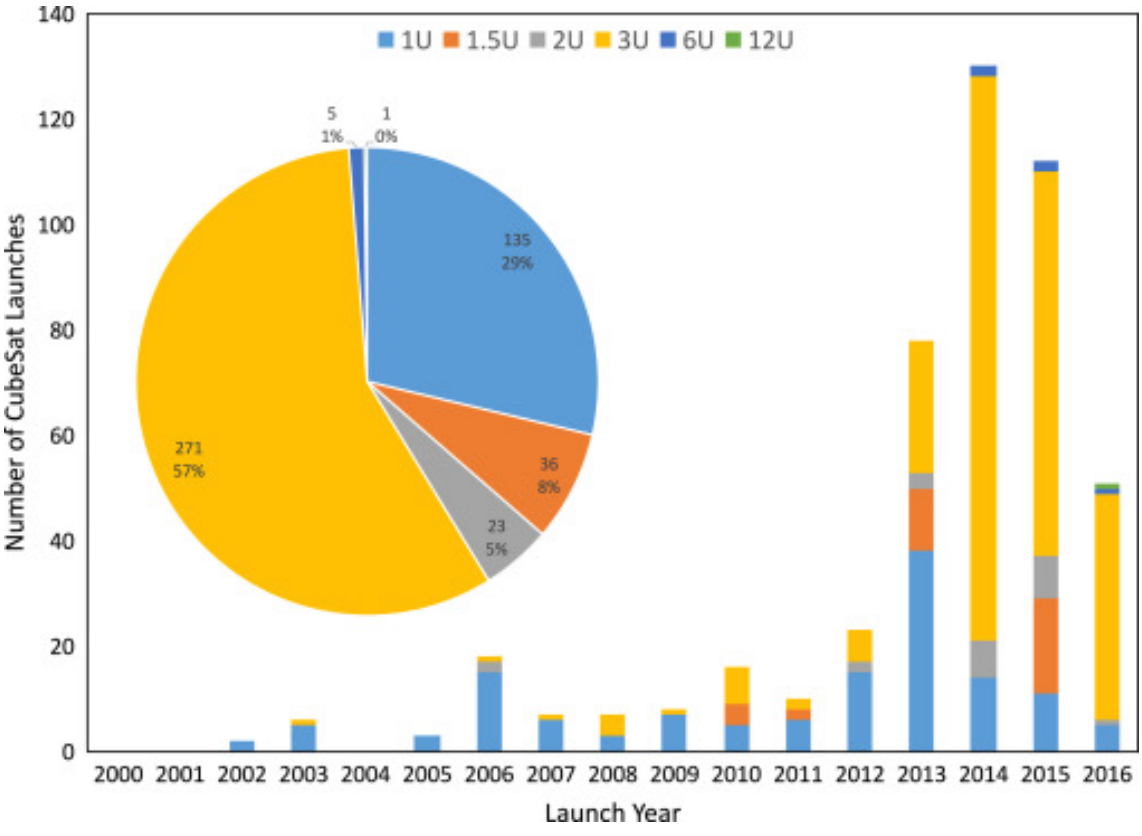
Sarnaselt turuosale on muutumas missioonide eesmärk. Populaarust on kogumas teadusmissioonid ning maa kaugseireks mõeldud missioonid. Satelliitide missioonid varieeruvad küllaltki suurel määral. Lisaks tavapärasetele pildistamise- ja kommunikatsioonimissioonidele võib leida ka põnevaid projekte nagu maavärinate ennustamine [8] või lennuliikluse jälgimine [9].

[1] http://spaceworkseng.com/



Joonis 3. Missioonide eesmärgid [7]

Kuigi võiks arvata (mida väiksem, seda odavam arendada), et kõige populaarsem kuupsatelliidi suurus on 1U, siis tegelikkuses see nii ei ole. Ülessaadetud kuupsatelliitidest 99% on suurusega 1U kuni 3U ning kõigist ülessaadetutest 57% on olnud hoopis 3U suurusega satelliidid [6].



Joonis 4. Kosmosse saadetud kuupsatelliidid suuruse järgi [6]

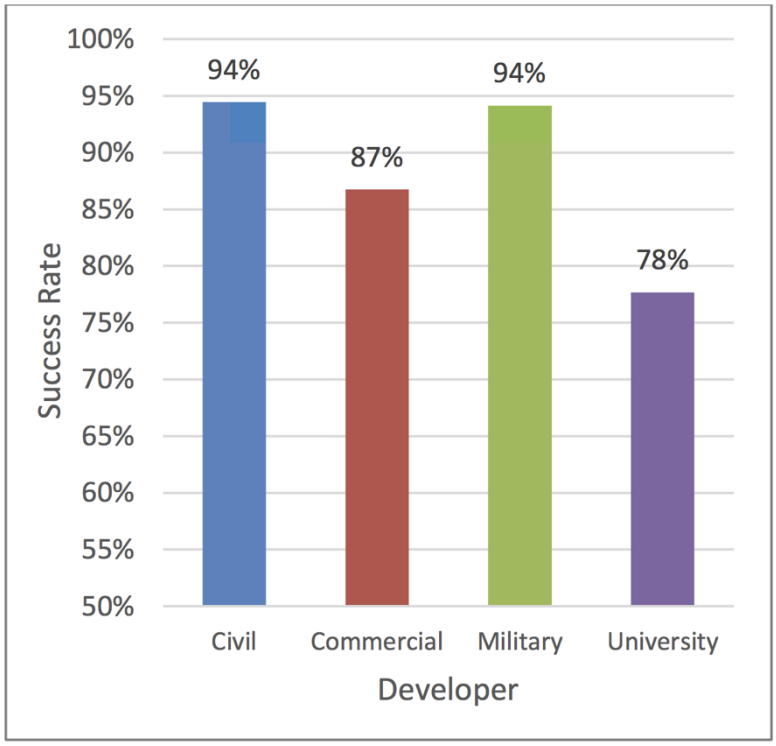
Vaadetes kuupsatelliidi projektide arendusaegasid võib kergesti märgata, et kommerts- ja militaarsektori keskmine arendusaeg on ligi poole väiksem, kui ülikoolide puhul. Kui üldjuhul lõpetatakse arendused 2 aasta jooksul, siis ülikoolide puhul on nii ainult 21% juhtudest. Ülikoolide pikema arendusaja võimalike põhjustena tuuakse artiklis [10] välja suur tudengite voolavus ning erinevatel aegadel loengutes/tundides osalemine, samas kui militaar- ja kommertsvaldkonnale on tegemist põhitööga.

Sarnast tendentsi märkas käesoleva töö autor ka TTÜ tudengisatelliidi projektis. Tudengite voolavus on küllaltki suur ning paljud juhendajad on hädas oma töögruppide komplekteerimisega. Mitmetel koosolekutel on probleemidena välja toodud, et üks hetk küll leitakse osa vajalikest tudengitest, kuid järgmine hetk huvi kaob ning tudeng loobub. Uute tudengite leidmine ei ole aga lihtne ning võtab aega. Isegi kui leitakse uued huvilised, siis neil jällegi omakorda kulub aega, et elada projekti sisse ning olla valmis reaalselt projektis osalema.

Arendaja	Keskmine arendusaeg (aastates)	% satelliitidest ehitatud 2a või vähema aastaga
Kommerts	1.7	100%
Militaar	1.6	92%
Ülikoolid	3.8	21%

Tabel 1. Kuupsatelliitide arendusajad sektori kaupa [10]

Vaadates missioonide õnnestumisi sektorite kaupa, võib jällegi märgata ülikoolide puhul tunduvalt väiksemat õnnestumise protsenti. Suure tõenäosusega on siingi põhjuseks suur tudengite vahetumine projektis, mille tõttu kvaliteet kannatab. Lisaks on kuupsatelliidi projektid paljudel ülikoolidel esimesteks projektideks, mis suurendab kindlasti ebaõnnestumise riski.



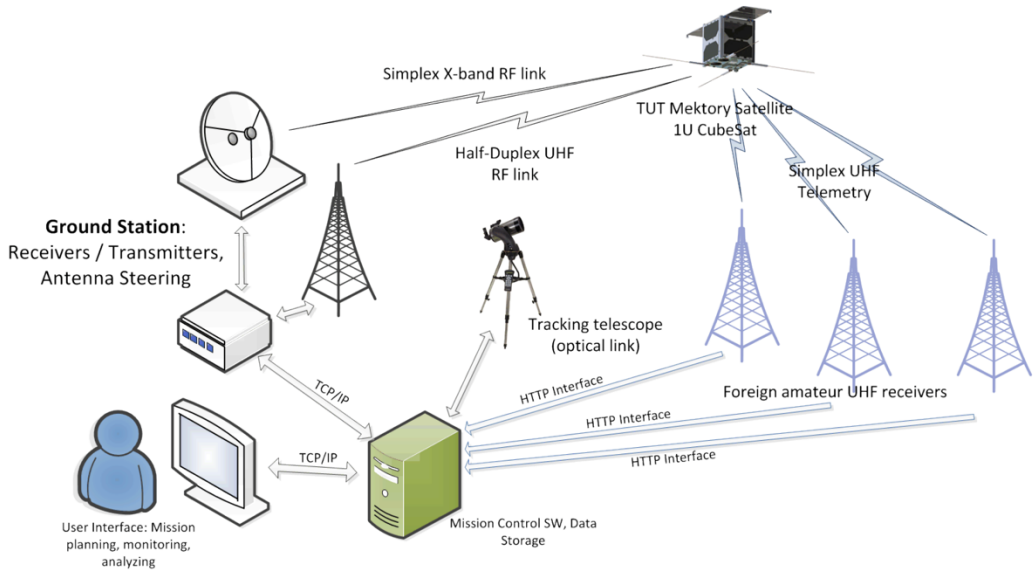
Joonis 5. Satelliitide missioonide õnnestumine sektorite kaupa [10]

2 TTÜ tudengisatelliit

TTÜ tudengisatelliit on 1U standardile vastav satelliit, mis on plaanis saata maa-lähedasele orbiidile (*low earth orbit*).

Satelliidiprogrammi eesmärkideks on [1]:

- Teostada maa kaugseiret maa-lähedaselt orbiidilt nähtavas ja/või infrapunases elektromagnet spektris;
- Testida uut kosmos-maa kõrge andmekiirusega side platvormi;
- Tehnoloogia demonstratsioon ja satelliidi positsiooni tuvastussüsteemi, satelliidi arvuti ja toitesüsteemi arendus.



Joonis 6. TTÜ satelliidiprojekti kontseptsioon [1]

3 Olemasolevad missioonijuhtimistarkvarad

Kuigi satelliite on kosmosse saadetud omajagu, siis nende missioonide juhtimiseks on erinevad teadusasutused arendanud tihti ise tarkvara ja laialt levinud ning avatud lähtekoodiga tarkvara, mida oleks võimalik erinevates projektides kasutada, ei leidu. Üheks peamiseks põhjuseks, miks sellist üldist tarkvara ei eksisteeri, on see, et kuupsatelliidid on väga erinevad oma riistvara poolest (alates maapealsetest antennidest kuni mikrokontrolleriteni satelliidi pardal) ning erinevad asutused on huvitatud oma uudsete lahenduste testimisest kosmose tingimustes. Riisvaraliste komponentide erinevus tingib vajaduse arendada selleks otstarbeks spetsiaalne tarkvara. Kogu kommunikatsiooniprotokoll erinevate satelliitide alamsüsteemide ja maapealsete komponentide vahel tuleb suures osas ise luua ja juurutada. Seega on paljudel juhtudel tegemist suures osas rätseplahenduse loomisega. Samas eksisteerivad siiski teatud standardid, näiteks TLE (*two-line element set*) või AX.25 amatöör raadioside protokoll[1], mis on loonud eeldused teatud komponentide puhul ühtse standardi ja tarkvara kasutamise.

3.1 Hummingbird

Hummingbirdi puhul on tegemist tarkvaraettevõte CGI[2] kosmosevaldkonna poolt välja arendatud monitooringu ja kontrolli baastarkvara kaugjuhitavate objektide jaoks. Hummingbird on kasutust leidnud nii ESTCube-1 kui ESTCube-2 projektis. Kui TTÜ oma satelliidi programmi alustas, siis esialgsetes kavatsustes oli samuti plaanis kasutada baaskomponendina Hummingbird tarkvara [11]

[1] https://www.tapr.org/pub_ax25.html

[2] https://www.cgi.ee/

3.1.1 Probleemid tarkvara käivitamisega

Tarkvara lähemalt uurides selgus, et selle käivitamine ei ole niivõrd lihtne. Kuigi tarkvara koduleheküljel oli erinevaid juhiseid, kuidas tarkvara kasutada ja laiendada, puudus seal korralik juhend tarkvara käivitamiseks ja seadistamiseks. Lugedes tarkvara Facebooki[1] leheküljel teiste kasutajate postitusi, võib märgata, et sarnased probleemid olid ka teistel. Uurides tarkvara lähtekoodi Github[2] keskkonnas, oli märgata, et projekt ei ole enam mitu aastat aktiivses seisus. 2016. a kevadel kadus internetist viimanegi kasutusjuhend hbird.de veebilehekülje näol. Tegemist on keeruka tarkvaraga, mille integreerimine ilma korraliku kasutusjuhendita ja inimeste poolt, kes pole selle tarkvaraga varem kokku puutunud, on väga vaevarikkas. ESTCube projektide puhul on tarkvaraarendaja olnud projektidega algusest peale tihedamalt seotud ning seetõttu on saadud tööle toimiv lahendus ja kohandatud seda vastavalt vajadustele.

3.2 NASA OpenMCT

Ühe võimaliku variandina leiti Open MCT[3] tarkvara. Tegemist on NASA poolt koordineeritud ning avatud lähtekoodiga tarkvara. Open MCT on arendatud siiski toetamaks kosmosemissioone, mitte nende täielikuks juhtimiseks [14]. Open MCT eesmärgiks on visualiseerida erinevaid ajaloolisi, kui ka reaalaja telemeetria andmeid. Selleks on olemas tarkvaral vastav API, mida kasutades saab selle liidestada enda missioonijuhtimistarkvarasse. Samas on tegemist ainult visualiseerimisevahendiga. Kogu juhtimistarkvara, läbi mille satelliidiga kommunikatsiooni pidada, tuleb siiski arendada eraldi.

[1] https://www.facebook.com/Hbird-The-Open-Satellite-Ground-Segment-156087881111212/

[2] https://github.com/Villemos/hbird-business

[3] https://nasa.github.io/openmct/

4 TTÜ satelliidi missioonijuhtimistarkvara

4.1 Nõuded missioonijuhtimistarkvarale

Tarkvarale on vastavalt spetsifikatsioonile esitatud järgmised nõuded [1]:

- N1. Kõik kaadrid, mille maale saatmiseks kasutatakse K_u-sagedusalas asuvat kiiret allalaadimislinki, tuleb säilitada nii töötlemata signaali kujul, kui ka lahti kodeeritult 1 kuu ulatuses.

- N2. Kõik andmed, mis on maajaamale saadetud UHF (*ultra high frequencies*) ja K_u sagedusala kaudu, tuleb säilitada eraldi serveril kuni 1 aasta peale missiooni lõppu.

- N3. Missioonijuhtimistarkvara peab rakendama kommunikatsiooniseanssides prioriteete. Kõrgema prioriteediga kommunikatsioonil on eesõigus madalama prioriteediga kommunikatsiooni ees.

- N4. Missioonijuhtimistarkvara peab võimaldama kasutada kommunikatsiooni sessioone, mis võivad kesta mitmeid ülelende.

- N5. Missioonijuhtimistarkvara peab kommunikatsiooni kaadrite koostamisel võtma arvesse võimalikku viivitust satelliidipoolsel sõnumi töötlemisel.

- N6. Missioonijuhtimistarkvaral peab olema kasutajaliides, mille kaudu läbi viia eksperimente kosmoses.

- N7. Missioonijuhtimistarkvara peab visualiseerima satelliidi asukoha maailmakaardil.

- N8. Missioonijuhtimistarkvara kaudu peab saama saata ja vastu võtta andmefaile satelliidi pardaarvutilt (*onboard computer*).

- N9. Missioonijuhtimistarkvaral peab olema liides, mille kaudu on võimalik kõrgkeeles kirjutatud käsujadasid teisendada saadetavate ja vastuvõetavate kaadrite jadadeks, mille kaudu toimub kommunikatsioon satelliidiga.

- N10. Missioonijuhtimistarkvaral peab olema liides kõrgetasemeliste missiooni kirjelduste koostamiseks satelliidile.

- N11. Missioonijuhtimistarkvara peab võimaldama kirjeldada punkti maapinnal, mida satelliit peab jälgima kaameratega ning tegema kindla arvu pilte ettemääratud ajahetkedel.

- N12. Missioonijuhtimistarkvara peab võimaldama kirjeldada punkte või mitmeid järjestikulisi punkte ja satelliidi orbitaalpositsioone, kust teha pilte satelliidi ühe või mõlema kaameraga.

- N13. Missioonijuhtimistarkvara avalik veebileht peab kuvama kasutajatele järgmist informatsiooni:

    - Satelliidi hetke asukoht maailma kaardil;

    - Viimase ülelennu ja kontakti aeg;

    - Viimaseid telemeetria andmeid;

    - Orbiitide arv alates kasutuselevõtust;

    - Aeg orbiidil alates kasutuselevõtust;

- N14. Missioonijuhtimistarkvara peab võimaldama vastu võtta UHF telemeetria andmeid raadioamatööridelt.

- N15. Missioonijuhtimistarkvaral peab olema välistele klientidele liides, mille kaudu ennustada satelliidi lennutrajektoori ja järgmisi kontakte.

- N16. Missioonijuhtimistarkvara peab saatma satelliidi orbiidiennustuse andmed maajaama süsteemile üle TCP/IP protokolli.

- N17. Missioonijuhtimistarkvara peab jagama satelliidi orbiidiennustuse andmeid väliste kasutajatega, võimaldamaks neil satelliiti jälgida optiliste teleskoopidega.

4.2 Arendusmetoodika valik

Agiilsed ja iteratiivsed arendusmetoodikad on kujunenud tarkvaraarenduses igapäevaseks ning koskmudeli (*waterfall*) (või sarnaste alternatiivide) kasutajaid võib kohata järjest vähem. Agiilse suuna kasvu viimase 10 aasta jooksul kinnitab ka Google Trends [14]. Samas on olemas valdkonnad, kus tänaseni kasutatakse edukalt koskmudelit. Üheks selliseks on missioonikriitilised infosüsteemid, mille hulka kuulub kindlasti ka satelliidi missioonijuhtimistarkvara. Mitte agiilsed arendusmetoodikad on üldiselt väga jäiga arendusprotsessiga, kus kõik nõuded on väga detailselt kirjeldatud projekti algus etappides ja projekti käigus suuri muudatusi lisada ei saa. See võimaldab tagada, et tarkvara vastab esitatud. Missioonikriitilistes infosüsteemides tähendab iga viga suurt rahalist kahju või halvimal juhul isegi ohtu inimeludele. Kuid see on tasapisi muutumas. Koskmudelit traditsiooniliselt järgiv NASA on otsustanud proovida agiilseid meetodikaid, mis annavad suurema paindlikkuse [15]. Paindlikkus on oluline ka ülikoolide poolt arendatavate kuupsatelliitide projektide juures. Tihti on tegemist ülikooli esimese projektiga, mistõttu võivad projekti käigus muutuda nii missiooni eesmärk kui ka kuupsatelliidil paiknev riistvara. Sellest tulenevalt ei ole võimalik detailselt kirjeldada kõiki võimalikke kasutusjuhte ja nõudeid. Iteratiivne arendusmudel on levinud ka kuupsatelliiti missioonijuhtimistarkvarade juures [16].

Iteratiivset arendusmetoodikat otsustati kasutada ka käesolevas projektis ning see on ennast igati õigustanud. Eelpool mainitud probleemid ülikooli esimeste projektidega, kus projekti käigus mitmed olulised aspektid muutuvad, on aset leidnud ka käesolevas projektis. Agiilne lahendus siinjuures on aga võimaldanud kiiresti muutustega kaasa minna.

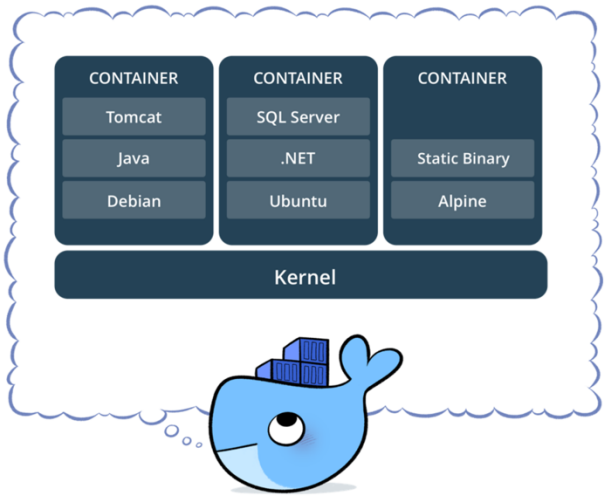
4.3 Juurutamine

Projekti käigus arendatava tarkvara juurutamiseks otsustati kasutada Dockeri[1] konteinerite raamistikku. Konteinerid võimaldavad pakkida kogu rakenduse ühte konteinerisse nii, et rakenduse installeerimisel piisab ainult konteineri käivitamisest. See võimaldab kerge vaevaga kogu rakenduse paigaldada teise serverisse, ilma et oleks tarvis

[1] https://www.docker.com/

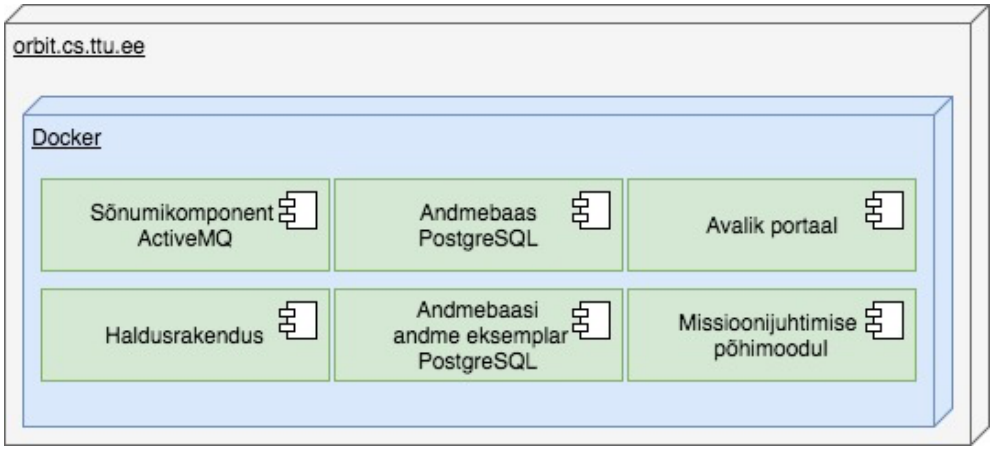
teha rakenduse spetsiifilist serveri konfiguratsiooni. Lisaks toetavad paljud suured pilvelahenduste pakkujad konteineritepõhist lahendust [17], [19], [20]. Konteineritepõhiseid lahendusi saab ka kergemini skaleerida, käivitades rakendusest rohkem versioone. Tihtipeale võimaldavad pilveteenuste pakkujad automaatset skaleeritavust, kui peaks tekkima järsk koormuse kasv, milletõttu olemasolevast ressursist ei piisa.

Sarnaselt virtuaalmasinatele on Dockeri konteinerid üksteisest täielikult isoleeritud. Kui virtuaalmasinad töötavad igaüks eraldi operatsioonisüsteemil, siis Dockeri konteineri puhul jagatakse *host* (võõrustaja) masina kernelit. See muudab konteinerid väikeseks ning ka ressursi kasutamise poolest säästlikumaks.



Joonis 7. Dockeri konteinerid [14]

Dockeri konteinerite põhise juurutusmudeli puhul on oluline konfigureerida konteinerite andmete *volume*'id (virtuaalsed kettad). Kuna konteinerite puhul on tegemist virtuaalsete süsteemiressurssidega, siis sama kehtib neis olevate andmete kohta. Kui rakendusest luuakse uus versioon ehk ehitakse uus *image* (süsteemikujutis) ning see paigaldatakse serverisse, siis eelmise rakenduse (sama rakenduse vanem versioon) konteiner eemaldatakse ning uuest *image*'st luuakse uus rakenduse konteiner. See tähendab, et vana konteiner ning selles olevad andmed eemaldatakse. Kui on aga andmeid, mis vajavad säilitamist erinevate rakenduste versioonide vahel, näiteks andmebaasi andmefailid või süsteemi logid, siis tuleb nende failide asukohad viia konteinerist välja *host* masinasse. Käesolevas projektis on viidud konteineritest välja andmebaasifailid, logifailid, sõnumivahetuskomponendi arhiivifailid ja failisüsteemi salvestatud meediafailid.



Joonis 8: Tarkvara komponentide paiknemine Dockeris

4.3.1 Pidev integratsioon ja versioonihaldus

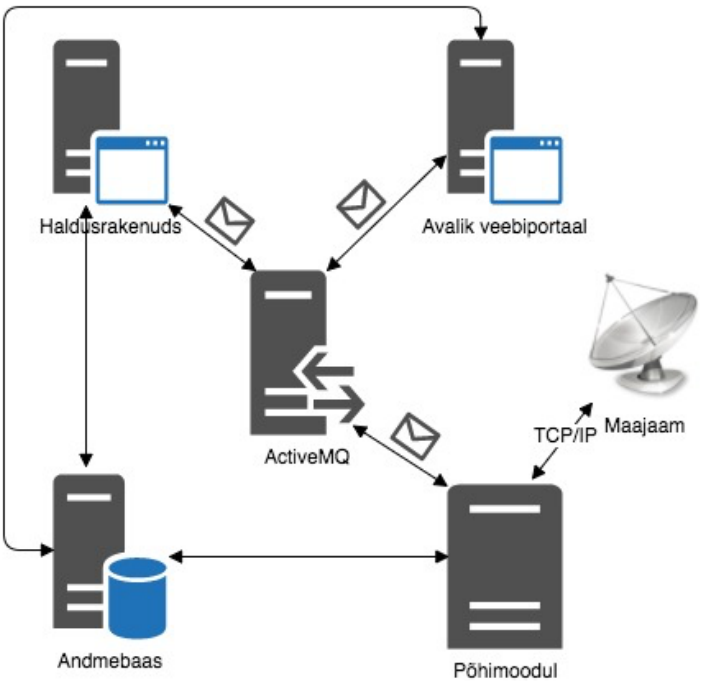
Tarkvara versioonihalduseks on kasutusel ülikooli Giti repositooriumide keskkond Gitlab[1]. Versioonihalduses järgitakse praktikat, kus *master*- (peamine haru) harus arendust ei toimu. Kogu arendus toimub arendajate poolt loodud harudes, mis on liigitatud funktsionaalsuste ja paranduste järgi. Arenduste valmimisel arendaja poolt liidetakse need harud arendusharru *develop*. Tarkvara automaattestide käivitamiseks ning tarkvara kompileerimiseks ning tarkvarapakettide ehitamiseks kasutatakse pideva integratsiooni keskkonda Jenkins[2]. Selleks on antud keskkonnas loodud vajalike seadistustega tööd, mida on võimalik käivitada. Peale töö õnnestumist paigaldatakse uued tarkvarapaketid (rakendused on pakitud Dockeri süsteemikujutiste sisse) arendusserverisse. Vajadusel on võimalik integratsioonitööde käivitamine muuta automaatseks (*git hook*), kui toimub uue koodi salvestamine versioonihaldusse (*git push*).

4.4 Arhitektuur

Missioonijuhtimistarkvara jaguneb peamiselt 5 suureks komponendiks: haldusrakendus, avalik veebiportaal, põhiarvutus- ja planeerimismoodul, sõnumivahetus komponent ja andmebaas (vt Joonis 9 ja LISA 2).

[1] https://gitlab.cs.ttu.ee/

[2] https://jenkins.io/



Joonis 9. Tarkvara üldine arhitektuur

Komponentide valikul üheks oluliseks lähtepunktiks on tarkvaralitsents. Eelistatud on avatud lähtekoodiga vabavaralist tarkvara.

4.4.1 Andmebaas

Relatsiooniliste ja vabavaraliste andmebaaside hulgas eksisteerib 2 suurt tegijat: MySQL ja PostgreSQL [15]. Uurides andmebaaside funktsionaalsusi, positiivseid ja negatiivseid argumente, siis leiab neid mõlema poole kasuks ja kahjuks. Näitena vahetas hiljuti Uber PostgreSQLi MySQLi vastu [16] ning TransferWise valis vastupidise tee [17]. See näitab, et alati ei ole võimalik ette näha kõiki andmebaasile esitavaid nõudmisi, mida võib tulevikus vaja minna. Samamoodi andmebaasi valikul ning hilisemal migratsioonil uuele, mängib suurt rolli arendajate varasem kogemus ja pädevus konkreetse andmebaasiga.

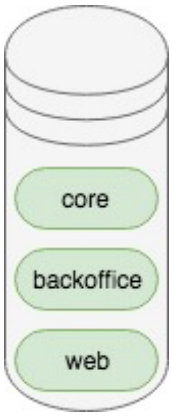
Ruumiandmete (*geodata*) hoidmise poolelt on eelis PostgreSQLil. Eelise annab PostGIS[1] laiendus, mis on endale leidnud suure kasutajaskonna. PostGIS võimaldab lisaks ruumiandmete hoidmisele kasutada päringutes erinevaid geograafilisi objekte. Ruumiandmete ja geograafiliste objektide tugi on kindlasti üheks argumendiks satelliidilt

[1] http://postgis.net/

kogutavate maaseireandmete hoidmisel. Samas ei saa märkimata jätta, et viimaste versioonidega on paranenud MySQL tugi erinevatele ruumiandmetele.

Andmebaasi valikul ei leitud ühegi konkreetset argumenti, mis oleks välistanud kummagi andmebaasi. Samas töö autoril oli olemas varasem mitmeaastane kogemus PostgreSQLi kasutamisel, mistõttu osutus valituks PostgreSQL andmebaas.

Paremaks andmebaasimuudatuste haldamiseks sai valitud Liquibase [1] tarkvara. Liquibase koos Git versioonihaldustarkvaraga võimaldab jälgida andmebaasi muudatuse ajalugu ning rakendada neid muudatusi erinevates keskkondades ühtemoodi. Käesoleval hetkel turul sarnase funktsionaalsusega alternatiivset toodet ei leidu.



Joonis 10. Andmebaasi skeemid

Loodud andmebaasis on erinevate moodulite andmed eraldatud eraldiseisvatesse andmebaasi skeemidesse. Vajaduse korral on võimalik moodulite andmed tõsta eraldi andmebaasidesse, kuid käesoleval hetkel selleks vajadust ei nähtud. Oluline on, et erinevatele moodulitele, mis andmebaasi kasutavad, oleks tekitatud eraldi kasutajad ning õigused antud vastavalt vajadusele.

4.4.2 Sõnumivahetuse komponent

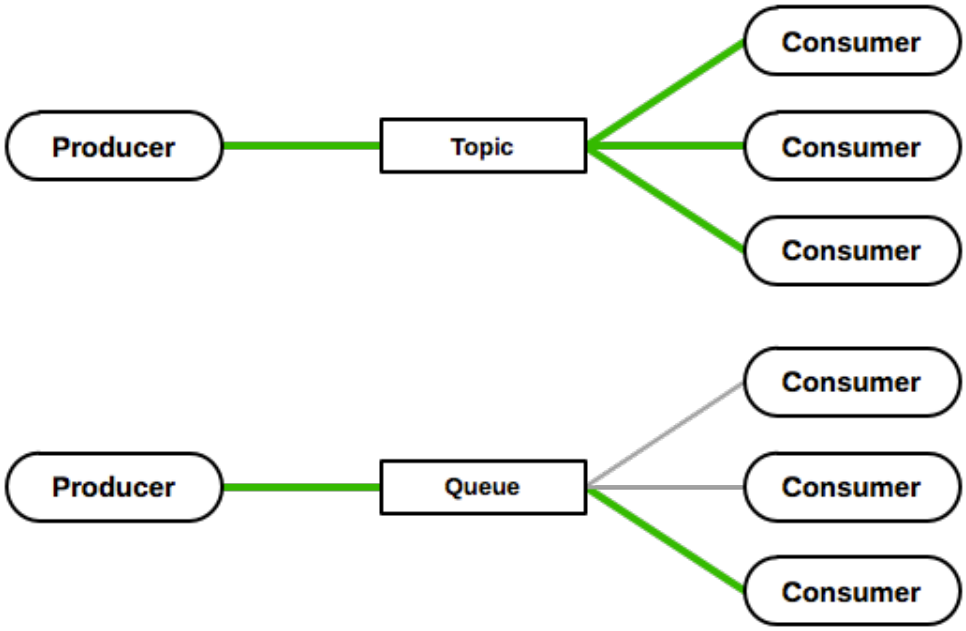
Tarkvara erinevate moodulite liidestamiseks on olemas mitmeid võimalusi. Ühe variandina saab kasutada päring-vastus sünkroonset lahendust. Samas toob see endaga kaasa moodulite jäiga sõltuvuse teineteisest. Alternatiivina saab süsteemides kasutada sündmustel põhinevat arhitektuuri (*event-driven arhtitecture*[2]). Sündmusetel põhineva

[1] http://www.liquibase.org/

[2] https://en.wikipedia.org/wiki/Event-driven_architecture

arhitektuuriga on lähedalt seotud sõnumipõhine arhitektuur, kui vaadelda igat sündmust sõnumina või sõnumit sündmusena. Sõnumipõhise arhitektuuri korral on süsteemis keskne sõnumivahetuskomponent, mis vahendab ja koordineerib sõnumite liiklust saatvate ja tarbivate osapoolte vahel.

Sõnumivahetusel põhineval arhitektuuril on mitmeid eelised, näiteks: andmeside asünkroonsus, sõnumite suunamine erinevate tarbijate vahel ja sõnumite puhverdamine. Sõltuvalt kasutatavast sõnumivahetuse komponendist erinevad konkreetsed funktsionaalsused.



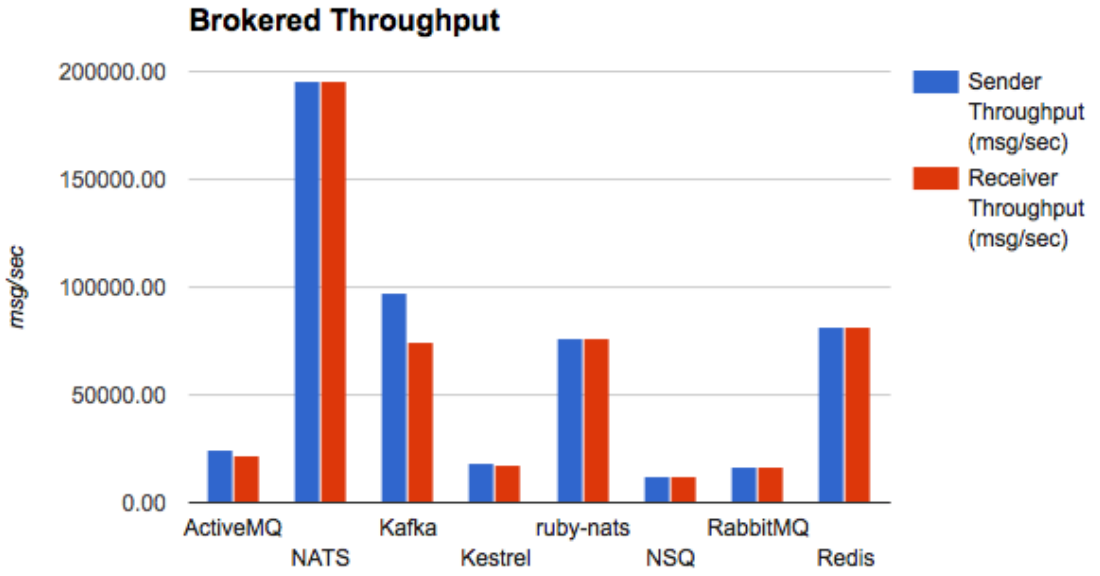
Joonis 11. Sõnumivahetuse mudelid [15]

Üldjuhul saab sõnumite saatmise mudelid jagada lõppsihtkoha järgi kaheks: teema (*topic*) ja järjekord (*queue*). Esimesel juhul saadetakse sõnum kõigile registreeritud tarbijatele (*consumer*) ning teisel juhul üks sõnum ainult ühele tarbijale. Samas erinevate tootjate tarkvaradedel realisatsioonid erinevad teineteisest ning pakkuvad täiendavalt spetsiifilisemaid funktsionaalsusi. Mõningate funktsionaalsuste näitena võib tuua klasterdavuse, replikeeritavuse, kõrgkäidelduvuse, sõnumite filtreerimise ja püsivuse. Lisaks on varieeruv toetatud sõnumiprotokollide standardite nimekiri. Java tehnoloogiate valdkonnas on paljude tarkvarade poolt toetatud JMS[1] (*Java Message Standard*) standard.

[1] https://en.wikipedia.org/wiki/Java_Message_Service

Sarnaselt funktsionaalsusete erinevusele ja probleemidele, mille jaoks on tarkvara toodetud, erineb nende jõudlus ning programmeerimiskeel, milles tarkvara on kirjutatud.

Sõnumivahetuskomponentide turg on lai ning enamik tooteid või nende baaskomponendid on avatud lähtekoodiga. Mõningate toodete puhul pakutakse täiendevalt tasulist tuge äriklientidele. Populaarsemad tarkvarad on tihti arenduse poolest toetatud või täielikult arendatud mõne globaalse tarkvaraettevõtte poolt kõrvaltootena. Näitena võib tuua RabbitMQ[1], mille arendaja on Pivotal Software (panustab palju ka Spring raamistiku arendusse) ja Kafka[2], mille esialgselt arendas välja LinkedIn ning mis hilisemalt on võetud Apache Software Foundation'i egiidi alla [16].



Joonis 12. Sõnumivahetus tarkvarade jõudlus [17]

Valides käesolevasse projekti sõnumikomponendi tarkvara, olid peamised esitatud nõuded:

- Sõnumite püsivus (*persistence*);
- Jõudlus;
- Dokumentatsiooni ning laia kasutajaskonna olemasolu;
- Integreeritavus (Java teekide olemasolu).

[1] ,https://www.rabbitmq.com/

[2] https://kafka.apache.org/

4.4.2.1 Jõudlus

Satelliidi ja maajaama vaheliseks suhtluseks on ettenähtud 2 raadioriba: aeglane *UHF* ja K_u. Aeglase riba puhul on kiiruseks ettenähtud 9600 bps (*bits per second*) ning kiirema riba puhul minimaalne kiirus 500 kbit/s. Kasutatava AX.25 protokolli ühe kaadri suuruseks on 2208 bitti [18]. Hea andmeside puhul on satelliidi poolt saadetavate AX.25 kaadrite arv minimaalselt ~ 225 kaadrit sekundis. Arvestades K_u riba keskmiseks kiiruseks 1-2 Mbps [19], suureneb kaadrite arv ligi 4 korda 1000 kaadrini sekundis. Võttes arvesse kaadrite väiksuse, siis enamik sõnumivahetuse komponente suudab sekundis töödelda kümneid kordi suuremaid koguseid (vt Joonis 12. Sõnumivahetus tarkvarade jõudlus Joonis 12).

Püstitatud nõutele vastasid mitmed uuritud tarkvarad nagu Kafka, RabbitMQ ja ActiveMQ. Kuna missioonijuhtimistarkvara mitmed teised komponendid on peamiselt realiseeritud Java platvormil, siis valituks osutus tarkvara ActiveMQ. See võimaldab hoida erinevate tehnoloogiate hulka väiksemana ning ühtsemana.

4.4.2.2 Sõnumite arhiveerimine

ActiveMQ'l on sissehitatud sõnumite arhiveerimise funktsionaalsus. Selleks tuleb konfigureerida *Message Store* ning seadistada *archiveDataLogs* väärtus tõeseks. See tagab, et peale sõnumite saatmist ja tarbimist ei kustutata sõnumeid ära, vaid jäetakse arhiivi kataloogi alles, kust on võimalik neid hiljem täiendavalt failisüsteemi tasemel varundada. See tagab, et missioonijuhtimistarkvarale esitatud nõuded N1 ja N2 on täidetud.

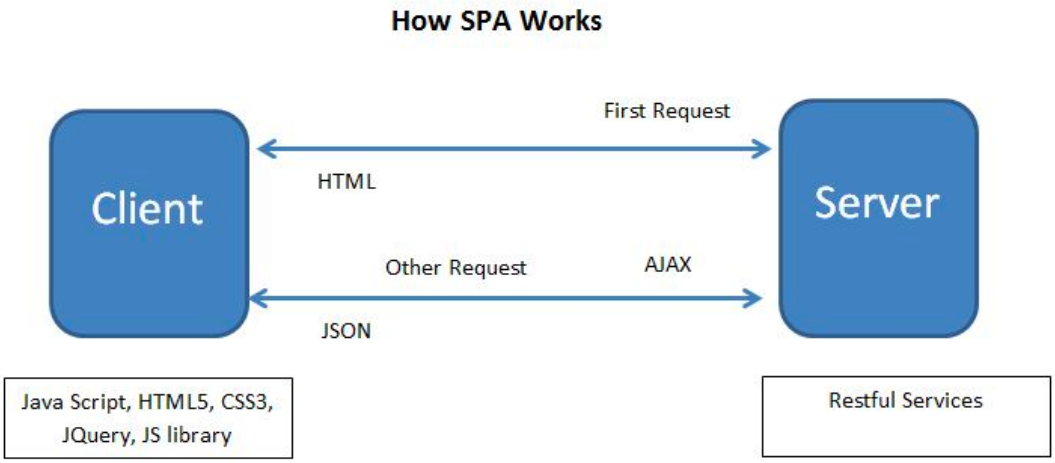
4.5 Haldusrakendus

Haldusrakendus on missioonijuhtimist läbiviivale kasutajale mõeldud veebipõhine rakendus. Rakenduse kasutajaliidese kaudu toimub vaid missioonijuhtimistarkvara töö juhtimine, põhitöö tehakse missioonijuhtimistarkvara põhikomponendis. Loodud kasutajaliidese kaudu on võimalik:

- Vaadata viimaseid ja järgmiseid ülelende;
- Vaadata ajas satelliidi asukohta kaardil;

- Vaadata satelliidi telemeetria andmeid;
- Planeerida järgimisi missioone ja saata käsklusi satelliidile;
- Vaadata alla laetud andmefaile;
- Hallata rakenduse kasutajaid.

Rakendus täidab missioonijuhtimistarkvarale esitatud nõuded N6 ja N7.



Joonis 13 SPA rakenduse arhitektuur [24]

Tegemist on *SPA* (s*ingle page application*) tüüpi veebilehega, mille kasutajaliidese peamiseks tarkvara raamistikuks on AngularJS ning tagarakenduse (*backend*) puhul Spring Boot raamistik. Täiendavalt kasutatakse reaalajas informatsiooni kuvamiseks WebSocketite põhist sõnumite vahetust. WebSocketite tehnoloogia võimaldab saata serveri poolt kasutajale sõnumeid, ilma et kasutaja brauseril oleks tarvis uuendusi intervalliga pärida. See on ressurssi säästlikum ning võimaldab luua samuti paremat kasutajakogemust. WebSocketi tehnoloogia on toetatud kõigi uuemate brauserite poolt [1].

Satelliidi asukoha ja lennutrajektoori joonistamiseks kaardile TLE põhjal on võimalik kasutada juba olemasolevaid tarkvara komponente nagu Isana[2]. Teise variandina saab sarnase lihtsa funktsionaalsuse ise arendada. Selleks on tarvis kasutada mõnda SGP4

[1] http://caniuse.com/#feat=websockets

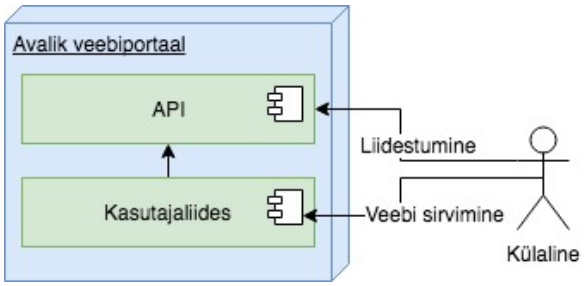
[2] http://www.lizard-tail.com/isana/tracking/

(*simplified perturbations models*) teisendust pakkuvat tarkvara teeki, mis TLE andmed konverteeriks geograafilisteks koordinaatideks. Üheks selliseks JavaScriptis kirjutatud teegiks on Satellite-JS[1] ning Javas Orekit[2]. Seejärel on võimalik saadud koordinaadid joonistada vektorina mõnele maakaardile, näiteks Google.

Rakenduse baasosa on loodud kasutades JHipster[3] generaatorit.

4.6 Avalik veebiportaal

Avalik veebiportaal on missioonijuhtimistarkvara osa, mis on mõeldud kõigile projektihuvilistele jälgimaks satelliidi missiooni kulgu, visualiseerides selleks erinevaid andmestikke ning kuvada satelliidi hetkeolekut. Täiendavalt on avaliku veebiportaali eesmärgiks pakkuda liidestust kolmandatele osapooltele. Näiteks raadioamatööridele, kes soovivad edastada satelliidilt saadud telemeetria andmeid. Avalik veebiportaal täidab missioonijuhtimistarkvarale esitatud nõuded N13, N14, N15 ja N17. Avaliku veebiportaali loomine ei ole käesoleva töö skoobis.



Joonis 14. Avaliku veebiportaali komponendid

Avalikul veebiportaalil on 2 komponenti: API (*application programming interface*) ja kasutajaliides. API eesmärk on luua võimalus kolmandatele osapooltele liidestumisvõimalus missioonijuhtimistarkvaraga. Avaliku veebiportaali kasutajaliides on ettenähtud kasutama sama API liidest andmete kuvamiseks. API kaudu on võimalik pärida erinevaid avalikke missiooniga seotud andmeid, kuid ka raadioamatööridel saata enda poolt kinni püütud raadiosignaalis sisalduvaid satelliidi telemeetria andmeid.

[1] https://github.com/shashwatak/satellite-js

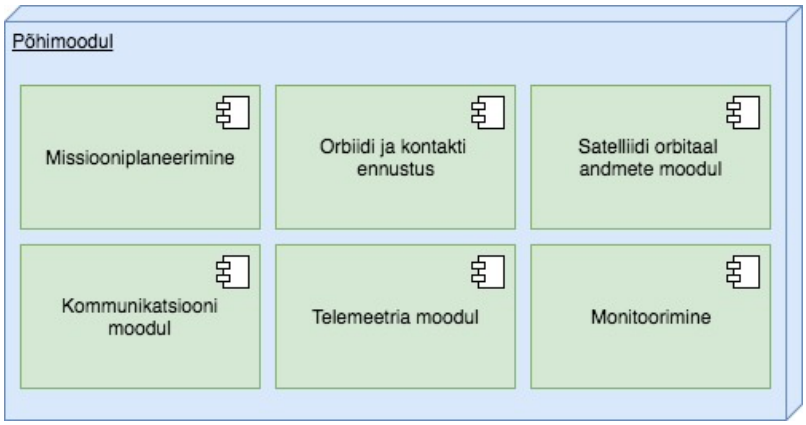
[2] https://www.orekit.org/

[3] https://jhipster.github.io/

Avaliku API kaudu missioonijuhtimisse saadetavad telemeetria andmed märgitakse missioonijuhtimise sisemises API-s ära kui andmed, mis on tulnud mitte usaldusväärsest allikast. See tagab, et võimaliku pahatahtliku kasutaja poolt edastatud väärate andmete põhjal ei käivitataks süsteemis ühtegi tegevust, mis võib nii missioonijuhtimistarkvara kui ka satelliidi tööd häirida.

4.7 Missioonijuhtimistarkvara põhimoodul

Missioonijuhtimistarkvara põhimoodul täidab süsteemis kõige suuremat ja olulisemat rolli. Põhimoodul jaguneb omakorda mitmeks erinevaks alamooduliks, mis kõik täidavad kindalt rolli. Peamiselt toimub moodulite vaheline kommunikatsioon sõnumikomponendi kaudu.



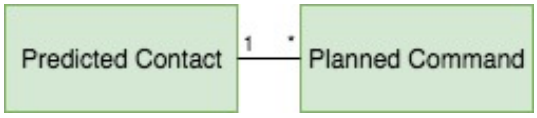
Joonis 15. Põhimooduli alammoodulid

4.7.1 Missiooniplaneerimine

Missioonijuhtimistarkvara üheks keskseks komponendiks on missiooniplaneerimise moodul. Teisi missioonijuhtimistarkvara komponente ja mooduleid võib vaadata, kui tugiteenuseid, mille abil missioone planeerida. Näiteks, et planeerida järgmiseks ülelennuks mõni käsklus, on meil vaja kõigepealt teada, millal järgmine ülelend ehk kontakt toimub. Kontakti arvutuseks vajalikud TLE andmed saadakse orbiidi ja kontakti ennustuse moodulist, sõnumivahetuse komponendi kaudu.

Kui järgmise ülelennu andmed (algus, kestvus ja orbiit) on olemas, siis järgmise asjana on tarvis teada satelliidi telemeetria andmeid, kontrollimaks, kas kõik planeeritud tegevused on üldse võimalikud selle ülelennu jooksul. Telemeetria andmeid on võimalik saada aga ainult satelliidilt endalt. Selleks algab üldjuhul iga ülelend käsuga küsida

telemeetria andmeid ning alles siis, kui kõik vajalikud telemeetria väärtused on nõutud vahemikus, saadetakse järgmine käsk.



Joonis 16. Ülelennuks planeeritud tegevused

Iga kontakti jaoks on haldusrakenduse kaudu planeeritud hulk käsklusi, mis selle ülelennu jooksul on tarvis satelliidile saata. Käskudel on olemas erinevad prioriteetsuse tasemed. Erandkorras võimaldab see muuta käskude täitmise järjekorda, kus kõrgema prioriteetsusega käsk saadetakse täitmisele enne madalama tasemega käske ning vajadusel katkestatakse teised tegevused [30].

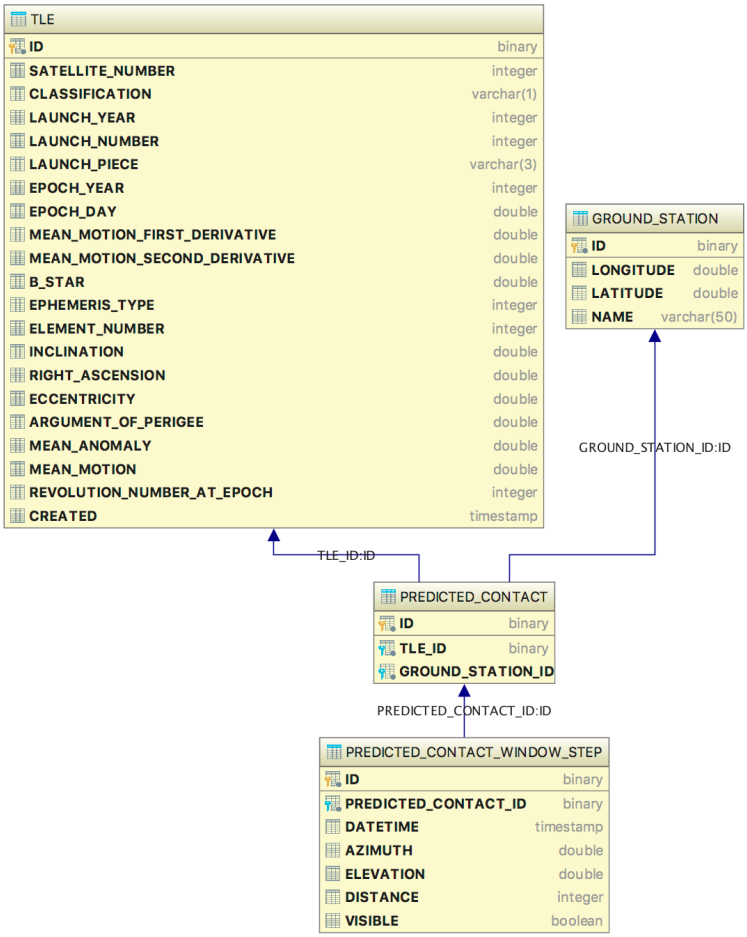
Sarnaselt üldistele telemeetria väärtustele, mis peavad nõutud vahemikus olema, võib kehtida samasugune kitsendus mõnele käsule. Näiteks planeerides käsku, mis nõuab suurt voolutarvet (pildistamine teatud asukohas ning satelliidi pööramine kaameraga vastavaks hetkes õigesse asendisse), tuleb arvesse võtta hetke satelliidi akude laetust ning voolu jätkumist kogu missiooniks. Samas akudes võib-olla piisavalt voolu teiste tegevuste jaoks. Seega ei ole mõtet saata satelliidile sellist käsku, mida ei ole võimalik täita ning planeerida selle asemel mõni teine käsk.

Kõikide käskude staatuse kohta peatakse logi, et hiljem oleks võimalik tuvastada, kas kõik planeeritud tegevused said ülelennu jooksul tehtud võib on tarvis teatud tegevusi korrata järgmisel ülelennul.

Ülelennu lihtsustatud tegevuste jadadiagramm on toodud LISA 1.

4.7.2 Satelliidi orbiidi- ja kontaktiennustuse alammoodul

Satelliidi orbiidiennustus- (*satellite orbit propagation*) mooduli peamiseks ülesandeks on arvutada välja järgmise kontakti täpne aeg ja orbiit. Saadud tulemus salvestatakse andmebaasi ning edastatakse sõnumivahetuse komponenti. Saadud arvutuse peamiseks tarbijateks on maajaam, et muuta enda asend vastavaks järgmise kontakti alguseks ning missiooniplaneerimise moodul, teadmaks planeerida järgmist sideseanssi (N16). Kolmandate osapooltega jagatakse arvutusi avaliku veebiportaali kasutajaliidese ja API kaudu (N15).



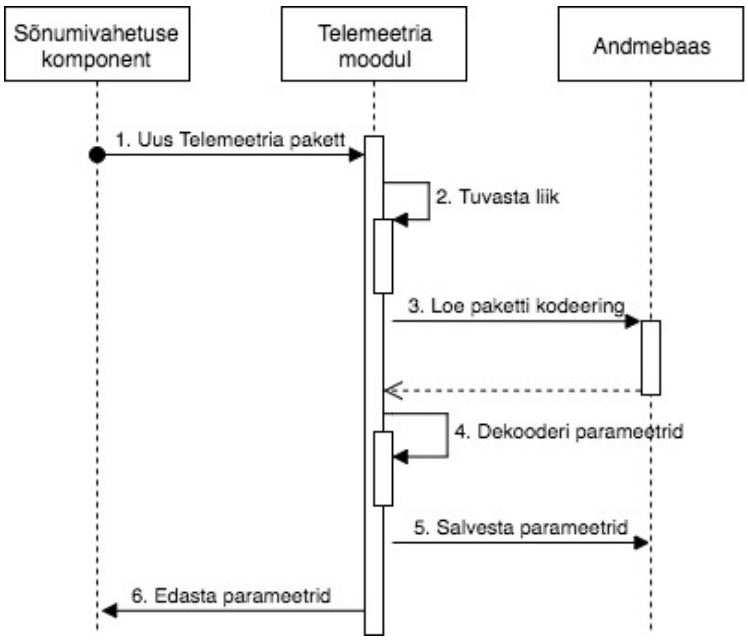
Joonis 17. Satelliidi järgmise kontakti ennustamisel saadav andmemudel

Orbiidiennustus on realiseeritud vastavalt kirjeldatud algoritmile [32]. Kontakti ennustus on tehtud kindlate koordinaatidega maajaama suhtes ja TLE andmete põhjal. Arvutuse tulemusena saadakse kontakti ajaaken, mille jooksul on satelliit nähtav maajaamale. Üks kontakti ajaaken koosneb mitmest sammust, mis kirjeldavad ära satelliidi asukohad ajalises järjestuses kontaktiaknas.

4.7.3 Telemeetria alammoodul

Telemeetria mooduli eesmärgiks on kirjeldada telemeetria sõnumite sisu ning dekodeerida see vastavalt iga satelliidi alamsüsteemi spetsifikatsioonile. Igal satelliidi alamsüsteemil on kindlaks määratud registrid, mis hoiavad infot konkreetse süsteemi toimimise kohta. Saades satelliidilt telemeetria paketi, liigub see kõigepealt sõnumivahetuse komponenti ning sealt saadetakse see edasi telemeetria moodulisse, kus tuvastatakse paketi päise järgi, mis liiki telemeetriga on tegemist. Pärast paketi

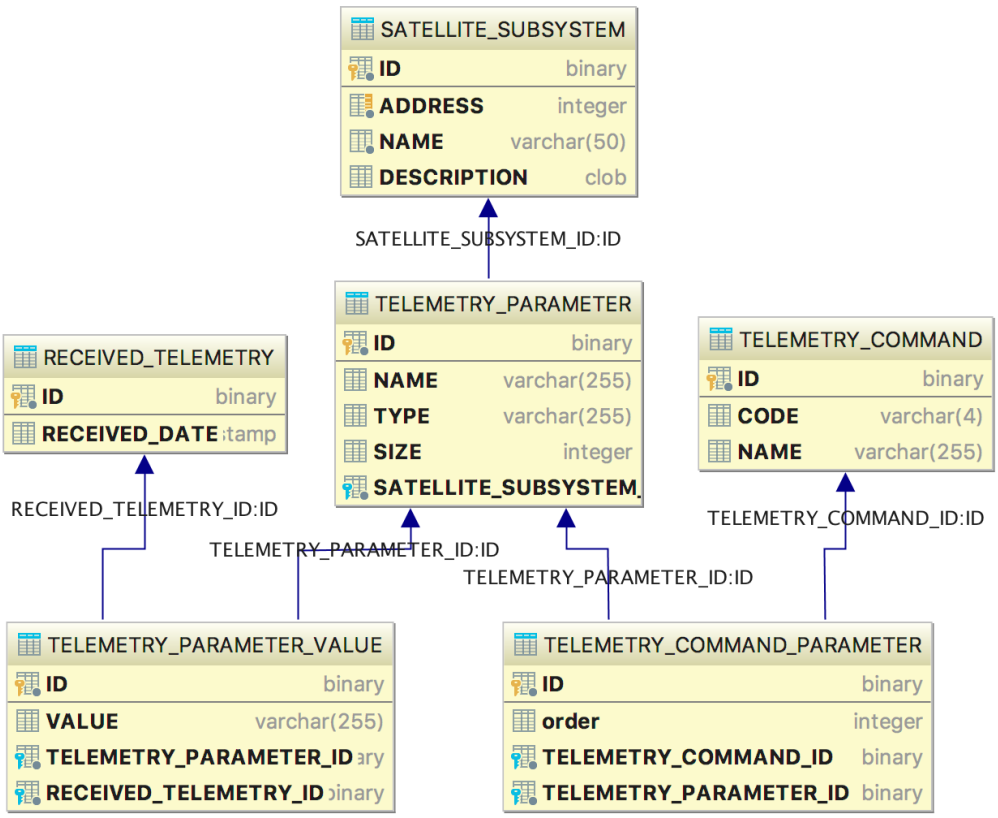
tuvastamist loetakse andmebaasist paketi kirjeldus, mille abil pakett dekodeeritakse ning saadud tulemused salvestatakse andmebaasi. Täiendavalt edastatakse saadud tulemused sõnumivahetusse, mis võimaldab reaalajas kõigil huvitatud komponentidel jälgida sissetulevaid telemeetria andmeid ning rakendada nende põhjal ettenähtud tegevusi.



Joonis 18. Telemeetria paketti dekodeerimine

Erinevate telemeetria parameetrite arv ulatub käesolevas projektis kasutusoleval satelliidil sadadesse. Ainuüksi EPS (*electrical power system)* alamsüsteemil on üle 100 erineva parameetri [25]. Rahvusvahelisel kosmosejaamal (ISS) ulatub telemeetria parameetrite arv ligi 150 000 [26].

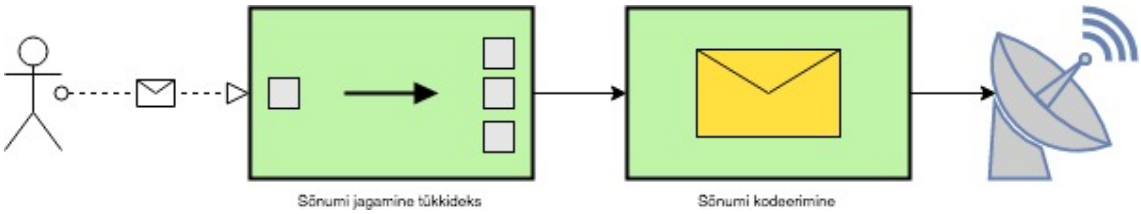
Telemeetria pakettide sisu kirjeldamiseks on loodud andmemudel (vt Joonis 19). Antud andmemudel võimaldab kirjeldada kõiki erinevaid kasutusel olevaid telemeetria pakette, nendes sisalduvate parameetrite kaupa. Igal parameetril on alamsüsteem, millesse ta kuulub, mis andmetüüpi ta on ning suurus bittides. Lisaks on iga paketi kohta kirjeldatud parameetri alguskoht paketis biti järgi. See on vajalik seetõttu, et parameetrite väärtused pakettides ei ole võti-väärtus paaridena, vaid ruumi kokkuhoidu mõttes asuvad kõik väärtused bittide kaupa üksteise järel. Kirjeldatud andmete struktuur võimaldab telemeetria paketi dekodeerides tuvastada, kust milline parameeter algab, mitmest bitist see parameeter koosneb ning millisesse andmetüüpi tuleks see parameeter konverteerida.



Joonis 19. Telemeetria pakettide kirjeldamise andmemudel

4.7.4 Kommunikatsiooni alammoodul

Kommunikatsiooni alammoodul omab ühte kõige suuremat tähtsust missioonijuhtimistarkvara põhimoodulis, kuna see realiseerib süsteemi kommunikatsiooniprotokolli dokumendis [30] kirjeldatud L3/L4 taseme protokolli. Kui haldusrakenduses on võimalik kõrgetasemeliselt ära kirjeldada satelliidile saadetavad käsud, siis kommunikatsioonimoodul kodeerib käsud vastavalt protokollile, edastab vastavalt kodeeritud paketid maajaamale ning juhib andmeside seanssi satelliidi ja maajaama vahel.



Joonis 20 Sõnumi tükeldamine ja kodeerimine

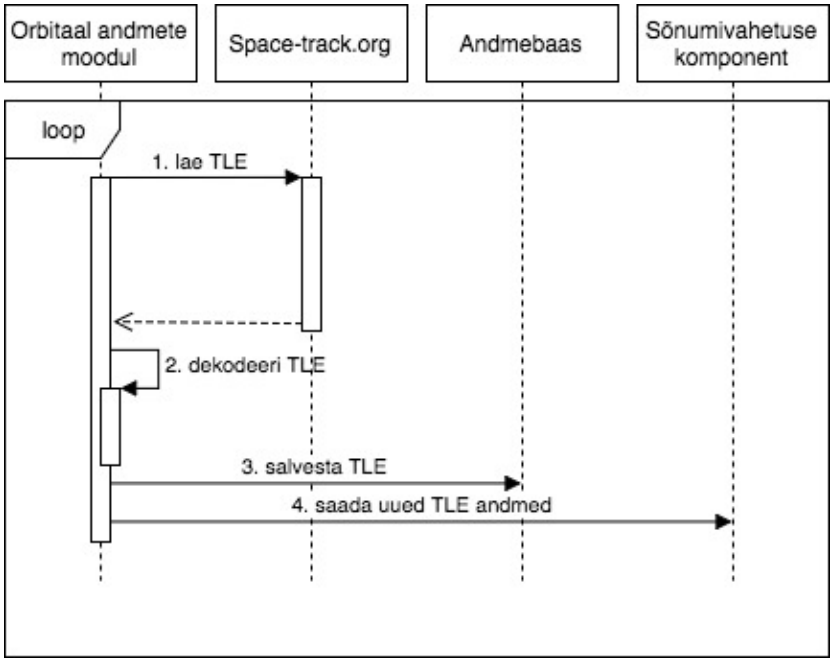
Näiteks ühte AX.25 kaadrisse mahub 2048 bitti informatsiooni [22], mistõttu on vajalik satelliidilt maapinnale või vastupidises suunas saadetav suurem sõnum ära tükeldada väiksemateks sõnumiteks ning kodeerida iga kaader vastavalt protokollile. Lisaks nõuab maajaama ja satelliidi vaheline kommunikatsioon juhtimist (saadetud pakettide üle arve pidamine, pakettide saatmis- ja kinnitussõnumite saatmine ning valideerimine, korduspakettide saatmine). Kommunikatsiooniprotokoll on täpsemalt ära kirjeldatud protokolli spetsifikatsioonis [32].

Kommunikatsioonimooduli loomine ei kuulu käesoleva töö skoopi. Moodul arendatakse välja teise tudengi poolt ning hiljem integreeritakse see missioonijuhtimistarkvara põhimoodulisse.

4.7.5 Satelliidi orbitaalandmete alammoodul

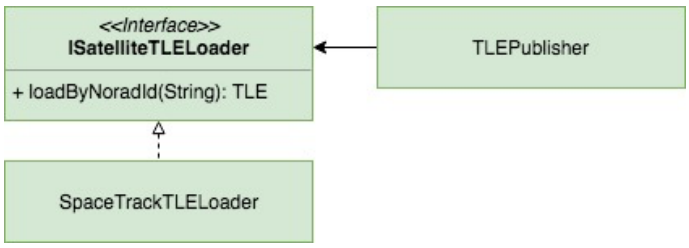
Mitmed missioonijuhtimistarkvara osad ja lõppkasutajad vajavad oma tööks informatsiooni satelliidi praeguse asukoha kohta. Teades satelliidi hetke asukohta, saab visualiseerida seda nii kaardil, kui ka kasutada seda järgmise kontakti arvutamiseks. Üle maailma on mitmeid veebilehti, kes jälgivad maa orbiidil olevaid satelliite ning jagavad seda veebis. Peamiseks andmeformaadiks mida kasutatakse on TLE. Üheks veebileheks, kes pakub TLE allalaadimise teenust, on Space-track.org[1]

[1] https://www.space-track.org



Joonis 21. TLE andmete laadimise protsess

Moodul on ehitatud fikseeritud intervalli tagant käivituma ning laadima alla uusi andmeid satelliidi asukoha ja asendi kohta. Pärast andmete alla laadimist toimub TLE andmeformaadi dekodeerimine, andmete salvestamine andmebaasi ning seejärel nende jagamine sõnumivahetuse komponenti.



Joonis 22. TLE alla laadimise liides

TLE sõnumivahetusse edastaja on SpaceTrackist alla laadijaga ühendatud liidese (*interface*) kaudu. See võimaldab vajaduse tekkimisel lihtsalt asendada üks teenuse realisatsioon mõne teisega, ilma teisi komponente muutmata.

Kokkuvõte

Käesoleva magistritöö eesmärgiks oli töötada välja tehniline lahendus, mis täidaks projektis missioonijuhtimistarkvarale esitatud nõuded.

Kuigi kuupsatelliite on saadetud kosmosesse juba sadades, siis ühtset missioonijuhtimistarkvara neile hetkel ei ole ning selle loomine ei ole lihtne. Kuigi satelliitidele on esitatud standardsed nõuded, millele need vastama peavad, siis sellega asi piirdubki. Iga ülikool soovib testida oma arendatud ja komplekteeritud riistvara. Iga selline unikaalne satelliit vajab aga paratamatult unikaalset juhtimist ja spetsiaalset tarkvara, mistõttu luuakse tihtipeale enda kommunikatsiooniprotokoll. On olemas küll standardseid andmeformaate (näiteks TLE), mida kasutatakse erinevates satelliitide süsteemides sarnaselt ja sama eesmärgiga, kuid suurem osa vajab siiski kohandamist kindla satelliidi ja missiooni jaoks.

Töö käigus uuriti olemasolevaid tarkvaralahendusi, nende tehniliste lahenduse puudusi ja eeliseid. Töö tulemusena loodi tehniline lahendus koos juurutamisega, võimaldamaks juhtida TTÜ satelliidi missioone. Loodud lahendus koosneb neljast suuremast moodulist: andmebaas, sõnumivahetus, haldusrakendus ja põhimoodul. Sõnumivahetus komponendi eesmärk süsteemis on käituda kui keskne integratsioonikiht, mis annab erinevatele komponentidele nõrgalt seotuse (*loosely coupled*). Põhimoodulis toimub erinevate satelliidi andmete töötlemine ning planeeritud missioonide täide viimine. Haldusrakendus võimaldab tarkvara kasutajal läbi veebipõhise keskkonna jälgida ja juhtida satelliidi missioone.

Loodud tehniline lahendus on paigaldatud TTÜ Orbit nimelisse serverisse.

Kasutatud kirjandus

[1] R. Adelbert - TUT-MEKTORY NANOSATELLITE System Requirements Specification - Tallinn, 2016.

[2] NASA - What are SmallSats and CubeSats? [WWW] https://www.nasa.gov/content/what-are-smallsats-and-cubesats (07.05.2017).

[3] NASA - What are cubesats? [WWW] https://www.nasa.gov/sites/default/files/thumbnails/image/what_are_cubesats.png (07.05.2017)

[4] About [WWW] http://www.cubesat.org/about/ (07.05.2017)

[5] Eurokot Launch 2003 [WWW] http://www.cubesat.org/past-launches/2015/6/22/eurorockot-2003 (07.05.2017)

[6] ESTCube-1 [WWW] https://www.estcube.eu/estcube-1-0 (07.05.2017)

[7] A. Poghosyan ja A. Golkar - CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions - *Progress in Aerospace Sciences,* 2017, 1. [E-ajakiri] (https://www.journals.elsevier.com/progress-in-aerospace-sciences) (07.05.2017)

[8] SpaceWorks - Nano/Microsatellite Market Forecast 2017 [WWW] http://spaceworksforecast.com/docs/SpaceWorks_Nano_Microsatellite_Market_Forecast_2017.pdf (07.05.2017)

[9] V. Korepanov - Possibility to detect earthquake precursors using cubesats - *Acta Astronautica,* 2016, 128, 203 - 209 [*Online*] ScienceDirect (07.05.2017)

[10] S. Nag, J. Rios, D. Gerhardt ja C. Pham - CubeSat constellation design for air traffic monitoring, *Acta Astronautica,* 2016, 128, 203 - 209 [*Online*] ScienceDirect (07.05.2017)

[11] G. Richardson, D. K. Schmitt, M. Covert3 ja C. Rogers, „Small Satellite Trends 2009-2013,“ The Aerospace Corporation, 2015.

[12] Hummingbird homepage [WWW] http://hbird.de (04.05.2016)

[13] OPEN MCT POSSIBILITIES [WWW] https://nasa.github.io/openmct/about-open-mct/#possibilities (07.05.2017)

[14] Y. Arslan - Is Agile and Scrum really better than Waterfall? [WWW] http://yusufarslan.net/agile-and-scrum-really-better-waterfall (07.05.2017)

[15] C. Webster ja I. S. S. Nija Shi - Delivering Software into NASA’s Mission Control Center Using Agile Development Techniques - *IEEE Aerospace Conference,* 2012 [*Online*] IEEE Xplore (07.05.2017)

[16] S. S. Yuhaniz ja N. Hamzah - Development of Mission Control Station Software for a CubeSat Mission - *Space Science and Communication*, 2015 [*Online*] IEEE Xplore (07.05.2017)

[17] Docker Basics [WWW] http://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html (07.05.2017)

[18] Azure Container Service [WWW] https://azure.microsoft.com/en-us/services/container-service/ (07.05.2017)

[19] Deploying Docker Containers on Compute Engine [WWW] https://cloud.google.com/compute/docs/instance-groups/deploying-docker-containers (07.05.2017)

[20] Docker [WWW] https://www.docker.com/what-container (07.05.2017)

[21] Popularity of open source DBMS versus commercial DBMS [WWW] https://db-engines.com/en/ranking_osvsc (07.05.2017)

[22] E. Klitzke - Why Uber engineering switched from Postgres To MySQL [WWW] https://eng.uber.com/mysql-migration/ (07.05.2017)

[23] F. Campoli - Scaling our analytics database [WWW] http://tech.transferwise.com/scaling-our-analytics-database/ (07.05.2017)

[24] JBoss Documentation [WWW] http://torquebox.org/documentation/1.0.0/messaging.html (07.05.2017)

[25] J. Koshy - Kafka Ecosystem at LinkedIn [WWW] https://engineering.linkedin.com/blog/2016/04/kafka-ecosystem-at-linkedin (07.05.2017)

[26] G. Senthilvel - NATS - Cloud Native Messaging System [WWW] https://www.linkedin.com/pulse/nats-cloud-native-messaging-system-ganesan-senthilvel (07.05.2017)

[27] W. A. Beech, D. E. Nielsen ja J. Taylor - AX.25 Link Access Protocol for Amateur Packet Radio [WWW] https://www.tapr.org/pdf/AX25.2.2.pdf (07.05.2017)

[28] L/Ku/Ka-band satellites – what does it all mean? [WWW] http://www.getconnected.aero/2014/11/lkuka-band-satellites-mean/ (07.05.2017)

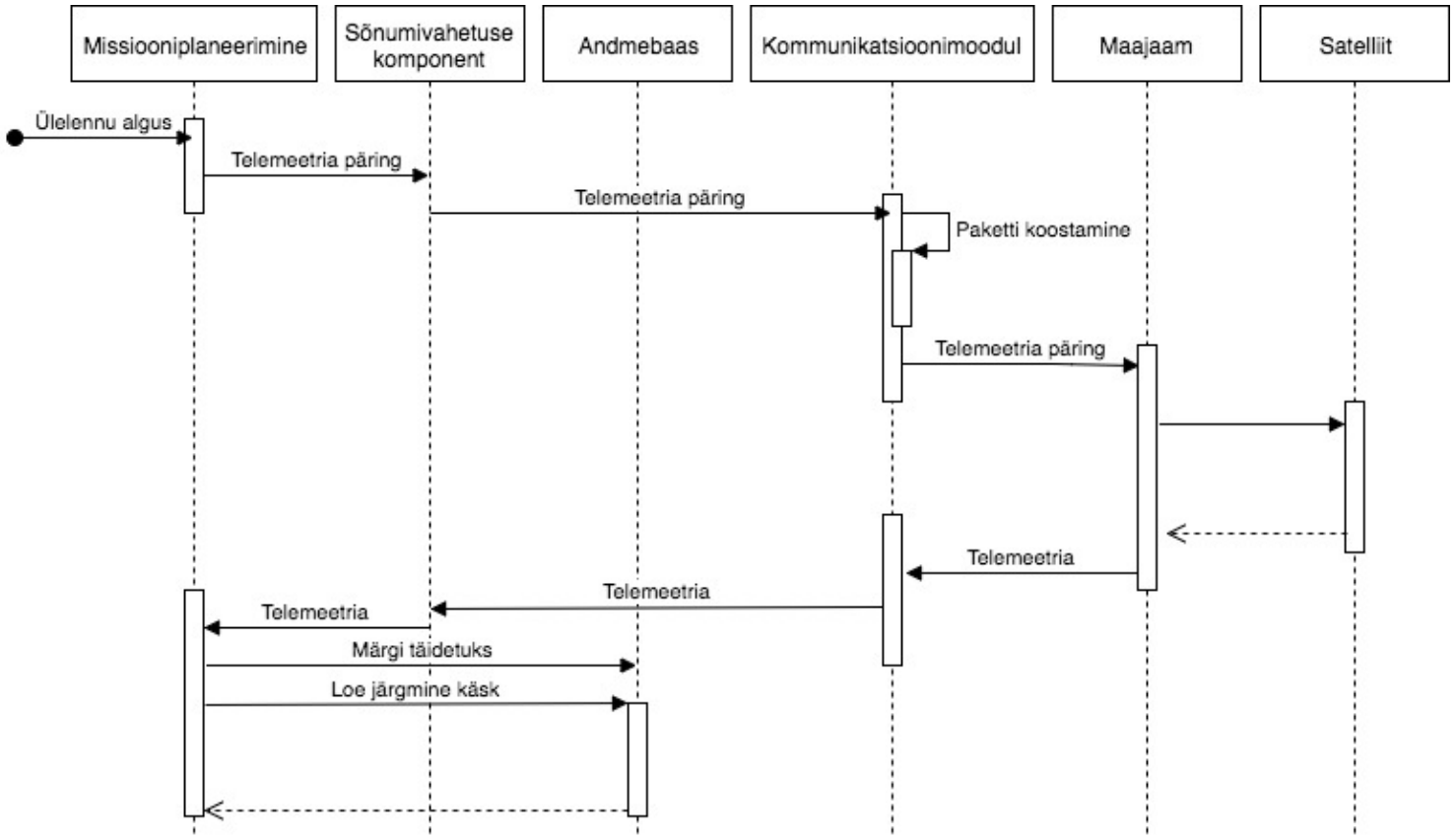
[29] P. Bajaj - Overview of Single Page Application [WWW] http://www.c-sharpcorner.com/blogs/overview-of-single-page-application-spa1 (07.05.2017)

[30] V. Pustynski - Communication window prediction - Tallinn, 2017

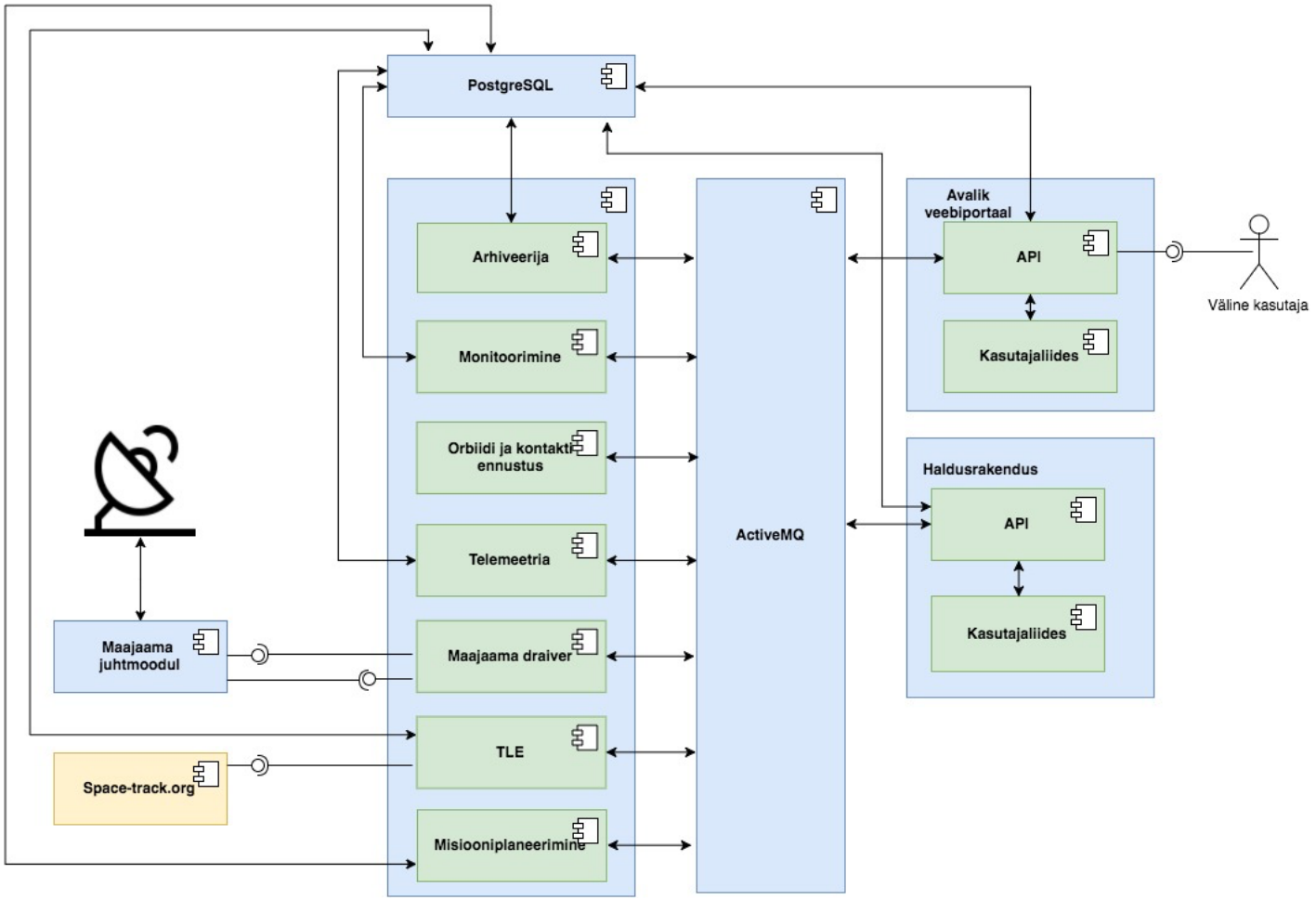
[31] V. Sinivee - EPS communication protocol and register map - Tallinn, 2017

[32] R. Adelbert -TUT-MEKTORY NANOSATELLITE Satellite TCTM Protocol Description - Tallinn, 2016.

LISA 1


Missiooniplaneerimine
Sõnumivahetuse komponent
Andmebaas
Kommunikatsioonimoodul
Maajaam
Satelliit
Ülelennu algus
Telemeetria päring
Telemeetria päring
Paketti koostamine
Telemeetria päring
Telemeetria
Telemeetria
Telemeetria
Märgi täidetuks
Loe järgmine käsk

LISA 2


PostgreSQL
Arhiveerija
Monitoorimine
Orbiidi ja kontakti ennustus
Telemeetria
Maajaama draiver
TLE
Misiooniplaneerimine
ActiveMQ
Avalik veebiportaal
API
Kasutajaliides
Haldusrakendus
API
Kasutajaliides
Väline kasutaja
Maajaama juhtmoodul
Space-track.org