Metropolia

TEKNIIKKA JA LIIKENNE

Tietotekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

PELIAUTOMAATIN PYSYVÄISMUISTIMODUULIN LUOTETTAVUUDEN KEHITTÄMINEN

Työn tekijä: Niko Hämäläinen

Työn ohjaaja: Jussi Lampiselkä

Työn ohjaaja: Auvo Häkkinen

Työ hyväksytty: __.__.2010

_____

Auvo Häkkinen
Yliopettaja

ALKULAUSE

Tämä insinöörityö tehtiin Raha-automaattiyhdistykselle. Työn ohjaajana toimi Jussi Lampiselkä. Kiitoksia kovasti erityisesti siitä, että kerkesit muiden kiireiden keskellä ohjata ja seurata työni kulkua. Kiitokset myös NVRAM-projektissa mukana olleille Markku Linnoskivelle sekä Jussi Seppälälle.

Lisäksi kiitokset Kimmo Koskiselle mahdollisuudesta työskennellä Raha-automaattiyhdistyksessä sekä Erno Langinkoskelle NVRAM-projektin ideoinnista. Lopuksi vielä kiitos kaikille työtä auttamassa olleille, niin RAY:ssä kuin sen ulkopuolella.

Tästä on hyvä jatkaa!

Helsingissä 19. marraskuuta 2010.

Niko Hämäläinen

Metropolia

TIIVISTELMÄ

Tekijän nimi: Niko Hämäläinen

Työn nimi: Peliautomaatin pysyväismuistimoduulin luotettavuuden kehittäminen

Päivämäärä: 19. marraskuuta 2010 **Sivumäärä:** 44 s.

Koulutusohjelma:	**Suuntautumisvaihtoehto**
Tietotekniikka	Ohjelmistotekniikka

Työn ohjaaja: Yliopettaja Auvo Häkkinen

Työn ohjaaja: Pelialusta-arkkitehti Jussi Lampiselkä

Tässä työssä tutustuttiin Raha-automaattiyhdistyksen nykyisen pysyväismuistimoduulin toteutukseen ja toteutettiin uusi ja luotettavampi pysyväismuistimoduuli nykyisen tilalle.

Pysyväismuistimoduuli huolehtii, että peliautomaatin käyttämien laskimien tietoja ei menetetä peliautomaatin sulkeutuessa. Fyysisenä tallennusmediana käytetään kahta erillistä toimilaitekorttia, joilla sijaitsee oma pysyväismuistipiiri. Molemmille toimilaitekorteille kirjoitetaan sama identtinen sisältö, mikä mahdollistaa toimilaitekorttien vaihtamisen ilman, että pysyväismuistin sisältöä menetetään.

Raha-automaattiyhdistyksen peliautomaattien käyttämä laitteisto asettaa omat haasteensa ja rajoitteensa pysyväismuistimoduulin toiminnallisuudelle. Jo olemassa oleva nykyinen pysyväismuistimoduuli asettaa uudelle pysyväismuistimoduulille tarkat vaatimukset. Uuden pysyväismuistimoduulin tulisi korvata nykyinen ilman, että pysyväismuistimoduulia käyttäviin moduuleihin tarvitsee tehdä muutoksia.

Uuden pysyväismuistimoduulin toteutuksessa otettiin huomioon nykyisen pysyväismuistimoduulin puutteet ja heikkoudet. Näitä silmällä pitäen jo kehityksen alkuvaiheessa otettiin käyttöön luotettavuutta parantavat menetelmät kuten yksikkötestit sekä staattinen analysointi. Uusi pysyväismuistimoduuli suunniteltiin ja toteutettiin yhteensopivaksi nykyisen pysyväismuistimoduulin kanssa.

Työn tavoitteena oli toteuttaa uusi, mutta luotettavampi pysyväismuistimoduuli nykyisen pysyväismuistimoduulin tilalle. Kattavien testausmenetelmien ansioista voidaan odottaa, että uusi pysyväismuistimoduuli on merkittävästi luotettavampi kuin nykyinen pysyväismuistimoduuli.

Avainsanat: Luotettavuus, Pysyväismuisti, NVRAM, Tiedostojärjestelmät

Metropolia

ABSTRACT

Name: Niko Hämäläinen

Title: Improving the reliability of the slot machine's NVRAM-module

Date: Friday 19th November, 2010 **Number of pages:** 44 p.

Department: Information Technology

Study Programme Software Engineering

Instructor: Auvo Häkkinen, Principal Lecturer

Supervisor: Jussi Lampiselkä, Game Platform Architect

The goal of this Bachelor's thesis was to study the current implementation of the NVRAM-module (Non-Volatile Random Access Memory) of RAY's (Raha-automaattiyhdistys, Finland's Slot Machine Association) slot machine and to implement new, more reliable one as a replacement.

Currently the NVRAM module stores counter sets to physical NVRAMs and ensures that no data is lost when the slot machine is shutdown. All data is written to two different NVRAM-chips which are located on two different peripheral cards. Writing identical data to two separate physical NVRAM-chips allows peripheral cards to be replaced without losing any data stored in NVRAM.

The hardware used in RAY's slot machines sets some challenges and limitations to the functionality of the module. The existing module also sets strict requirements as the new module should be able to replace the current module without making any changes to the client modules.

The weaknesses of the current NVRAM-module were taken into account when designing and implementing the new module. Unit testing and static analysis was brought to the development process from the very beginning of the project in order to improve the reliability of the new module. The new module was also designed and implemented to be backwards compatible with the current module.

The objective of this thesis was to implement new, but a more reliable replacement for the current NVRAM-module. Due to comprehensive testing, it is reasonable to expect that the reliability of the new implementation has improved compared to the current one.

Keywords: NVRAM, Reliability, File Systems

SANASTO

Sana	Merkitys
Bash	Bourne Again Shell.
Build Server	Käännöspalvelin.
CRC	Cyclic Redundancy Check, tarkisteavaimen luontiin tarkoitettu tiivistealgoritmi.
EEPROM	Electronically Erasable Programmable Read-Only Memory, ohjelmallisesti uudelleen kirjoituksia salliva haihtumaton puolijohdemuisti.
Ehtokattavuus	Kuinka suuri osa kaikista ohjelmiston ehtolauseiden eri arvoista suoritetaan läpi yksikkötesteissä.
GCC	GNU Compiler Collection.
Lausekattavuus	Kuinka suuri osa ohjelmiston lähdekoodin lauseista suoritetaan yksikkötesteissä.
MD4	Message Digest 4 -algoritmi, 128-bittinen tiivistealgoritmi.
Mock Object	Jäljitelmäolio.
NVRAM	Non-Volatile Random Access Memory, pysyväismuisti.
NVRAM-piiri	Pysyväismuistipiiri.
PSI	Nykyisen peliautomaattisukupolven nimi. PSI ei ole lyhenne mistään.
PSI-sovellus	Peliautomaatissa pyörivä itsenäinen moduuli.
RAM	Random Access Memory, satunnaishakumuisti.
RAY	Raha-automaattiyhdistys.
STL	Standard Template Library.
Testauskehys	Test Framework.
USB	Universal Serial Bus.
Verifiointi	Varmennetaan järjestelmän toimivuus yksityiskohtaisesti.
XML	Extensible Markup Language, standardoitu merkintäkieli.

SISÄLLYS

1 JOHDANTO

Raha-automaattiyhdistys on rahapeleihin erikoistunut organisaatio. RAY:n liikevoitto tulee lähes kokonaisuudessaan peliautomaateista. RAY:llä on noin 20 000 peliautomaattia ympäri Suomea. Peliautomaatteja on sijoitettu useisiin erilaisiin toimipisteisiin, joissa myös käyttäjäkunta vaihtelee paljon. Erilaiset ympäristöt ja käyttäjät tuovat mukanaan suuren osan satunnaisilmiöitä, joihin peliautomaatin tulee reagoida ilman, että mitään kriittistä tietoa menetetään ilmiöiden jälkiseurauksina. Kriittiset tiedot tallennetaan pysyväismuistiin, jonka kirjoitus- ja lukupalveluita NVRAM-moduuli hallinnoi.

Tämän insinöörityön tavoitteena on suunnitella ja toteuttaa uusi luotettavampi NVRAM-moduuli. Luotettavuutta lähdetään pääosin tavoittelemaan kattavien testausmenetelmien avulla.

Työ koostuu kahdesta osasta. Ensimmäisessä osassa tarkastellaan nykytilannetta. Luvussa 2 tutustutaan Raha-automaattiyhdistyksen nykyisen sukupolven peliautomaatteihin. Peliautomaatti asettaa ohjelmistokehitykselle omat vaatimuksensa ja rajoitteensa, mitkä on syytä tuntea ennen työn muihin osiin siirtymistä. Luvussa 3 käydään läpi nykyisen NVRAM-moduulin toteutus. Samalla tutustutaan nykyisen toteutuksen heikkouksiin. Näiltä heikkouksilta halutaan välttyä uuden NVRAM-moduulin toteutuksessa.

Toisessa osassa tarkastellaan uuteen toteutukseen liittyviä ratkaisuja ja luotettavuuden kehittämismenetelmiä. Luvussa 4 käsitellään uudelle NVRAM-moduulille asetetut vaatimukset. Lisäksi tutustutaan uuden moduulin toteutussuunnitelmiin sekä esitellään uuden moduulin arkkitehtuuri. Luvussa 5 keskitytään vikasietoisuuden kehittämiseen. Vikasietoisuutta kehitetään korjaamalla nykyisestä NVRAM-moduulista tunnistetut kriittisimmät heikkoudet. Lisäksi käsitellään testausmenetelmät, joilla uutta NVRAM-moduulia testattiin.

Lopuksi tarkastellaan tuloksia luvussa 6. Tuloksissa arvioidaan NVRAM-moduulin luotettavuutta kolmella eri menetelmällä, joita ovat: vaatimusten verifiointi, testausraportti sekä käytännön kokemukset.

2 PELIAUTOMAATTI

Raha-automaattiyhdistyksellä on vajaa 20 000 peliautomaattia ympäri Suomea. Peliautomaattien sijoittaminen on jaettu kolmeen eri jakelualueryhmään. Suurin osa peliautomaateista on niin kutsutussa hajasijoituksessa, eli ne sijaitsevat kioskeissa

ja elintarvikekaupoissa. Toisena ryhmänä ovat ravintolat ja muut anniskelupaikat. Kolmantena on Raha-automaattiyhdistyksen omat pelisalit, joissa työskentelee RAY:n omaa henkilökuntaa. Jakelualueiden välillä panokset ja päävoitot vaihtelevat. Hajasijoituksessa olevilla peliautomaateilla on pienimmät panokset ja pienimmät päävoitot, kun taas pelisaleissa olevilla peliautomaateilla voi suuremmilla panoksilla saada myös suurempia päävoittoja.

Tässä luvussa käydään läpi, minkälainen laite Raha-automaattiyhdistyksen peliautomaatti oikeastaan on. Tarkoituksena on selventää pysyväismuistimoduulin taustaan liittyviä ohjelmallisia ja laitteistollisia rakenteita sekä niihin liittyviä rajoitteita.

2.1 Ohjelmistoarkkitehtuuri

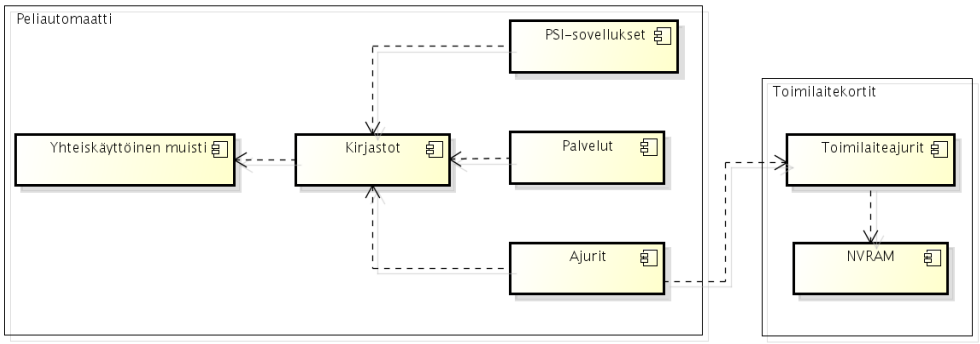
Peliautomaatin kuorien sisällä on periaatteessa x86-suoritinarkkitehtuurin mukainen tietokone, johon on kytketty lisälaitteita USB:n ja sarjaportin välityksellä. Peliautomaattien käyttöjärjestelmien ytimenä käytetään Linuxia. Yleisten jakeluversioiden sijaan käytetään automaatissa sulautettuihin järjestelmiin paremmin soveltuvaa BusyBoxia [1]. BusyBox tuo mukanaan yleisimmät UNIX-ympäristöistä tutut työkalut kätevässä pienessä paketissa.

Nykyisten PSI-sukupolven automaattien ohjelmiston tuotekehitys aloitettiin vuosituhannen vaihteessa, ja ensimmäiset julkaisut lähtivät kentälle vuoden 2003 aikana. Ohjelmisto koostuu useista moduuleista, joita ajetaan peliautomaatissa omina prosesseinaan. Jokaisella prosessilla on oma vastuualueensa peliautomaatin toiminnallisuudessa. Peliautomaatin moduuleja kutsutaan PSI-sovelluksiksi.

Kaikkien PSI-sovellusten kantaluokka on PSIClient, joka toteuttaa PSI-sovellusten yhteisen perustoiminnallisuuden. Perustoiminnallisuuksiin kuuluu XML-parametrien (Extensible Markup Language) luku, prosessien välisten viestien sekä virhe- ja vikailmoitusten lähettäminen. Viestien lähettämiseen palataan tarkemmin luvussa 2.2. PSIClientin lisäksi peliautomaatista löytyy *Startup*-niminen moduuli, joka on käytännössä kaikkien PSI-sovellusten isäntäprosessi. Isäntäprosessilla tarkoitetaan sitä, että Startup käynnistää kaikki muut PSI-sovellukset ja valvoo niiden elämää. Toisin sanoen kaikki muut PSI-sovellukset ovat Startup-moduulin lapsiprosesseja.

Monien PSI-sovellusten on mukauduttava useisiin eri jakelualueisiin ja toimittava eri alueilla eri tavalla. Mukautuvuus toteutetaan PSI-sovellusten lukemien XML-parametrien avulla. Jokaisella PSI-sovelluksella on oma moduulikohtainen XML-parametritiedostonsa, johon kehittäjät voivat määrittää toimintaa muuttavia parametreja. Parametrit luetaan PSI-sovelluksen käynnistyksen yhteydessä. XML-parametrit mahdollistavat sen, että PSI-sovelluksia ei tarvitse haarauttaa jakelualueiden mukaan, jolloin ylläpidettävyys on helpompaa. XML-parametrien haarauttaminen jakeluversioihin on huomattavasti ylläpidettävämpi ratkaisu.

PSI-sukupolven automaattien ohjelmistohierarkia voidaan jakaa komponenteiksi kuvan 1 mukaisella tavalla.



Kuva 1: PSI-sukupolven automaatin komponenttikaavio.

PSI-sovellus on yleinen kategoria peliautomaatin moduuleille, mutta osa PSI-sovelluksista luokitellaan tarkempiin kategorioihin, eli palveluihin ja ajureihin. Kyseisiin kategorioihin kuulumattomat PSI-sovellukset tarvitsevat usein palvelut-kategoriaan kuuluvien PSI-sovellusten apua toiminnallisuutensa toteuttamisessa.

Palveluiksi luokitellaan ne PSI-sovellukset, joiden toiminnallisuus perustuu muiden moduulien lähettämien pyyntöjen palvelemiseen. PSI-sovellukset pääsevät yleisimpiin palveluihin käsiksi PSIClient-kantaluokan mukana tulleiden funktioiden avulla. Osa palvelumoduuleista tarvitsee palveluidensa hoitamiseen kommunikointia peliautomaatin ajurisovellusten kanssa. Näissä tilanteissa palvelut toimivat välikätenä PSI-sovellusten ja ajureiden välillä.

Ajurisovellukset hoitavat peliautomaatin puolella toimilaitteiden ohjaamisen. Ajurisovellukset voivat keskustella palveluiden sekä toimilaiteajureiden kanssa. Mikäli toimilaiteajurilta saapuu tapahtuma ajurisovelluksille, ajurisovellus tulkkaa sen PSI-viestiksi ja ohjaa sen palveluiden hoidettavaksi.

Toimilaitekorteilla on oma suoritin, jonka muistiin ladataan peliautomaatin käynnistyessä toimilaiteajurit. Toimilaiteajurit ohjaavat kyseistä toimilaitetta sekä mahdollistavat peliautomaatin käyttäjien aiheuttamat ulkoiset tapahtumat, kuten rahansyötön, pelikierroksen aloittamisen ja rahan ulos maksamisen. Toimilaitteet on selitetty tarkemmin luvussa 2.3.

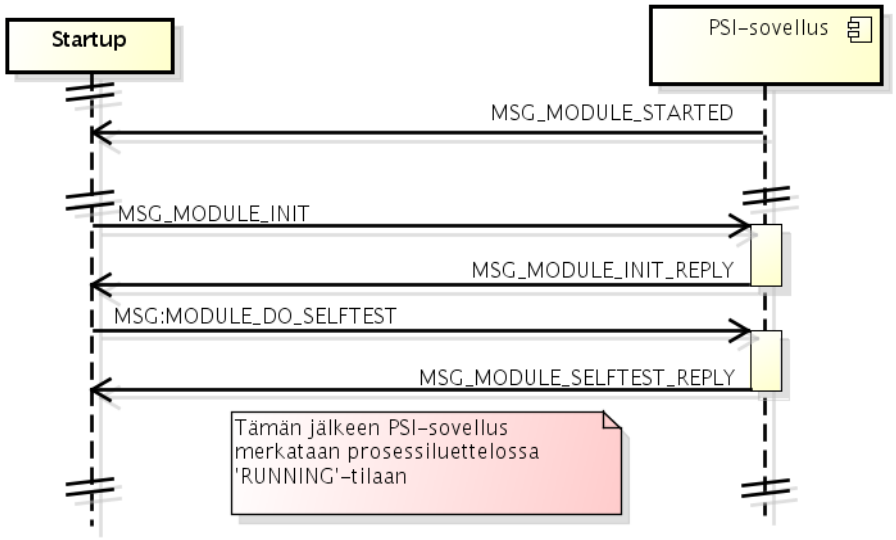
Peliautomaatissa pyörivät PSI-sovellukset, palvelut ja ajurit tarvitsevat usein toimintojensa toteuttamiseen ulkopuolisia kirjastoja. Osa kirjastoista on RAY:n oman tuotekehityksen tekemiä kirjastoja ja loput avoimen lähdekoodin koodikirjastoja, kuten C++-standardikirjasto STL (Standard Template Library).

2.2 Viestintä

Modulaarisessa järjestelmässä ei olisi mitään oikeaa toiminnallisuutta, mikäli moduulit eivät pysty kommunikoimaan toistensa kanssa. PSI-sukupolven peliautomaateissa PSI-sovellusten väliseen viestintään on kehitetty PSIMessenger-kirjasto, jonka tarkoituksena on mahdollistaa PSI-sovellusten välillä liikkuvien PSI-viestien vastaanotto ja välitys. PSIMessenger tunnistaa lähettäjät ja vastaanottajat yksilöllisten moduulitunnisteiden perusteella. Viestien avulla välitetään signaaleita, virheilmoituksia, käskyjä sekä dataa PSI-sovellusten välillä. PSI-sovellusten on määrä toteuttaa kuvassa 2 esitelty prosessien välinen viestirajapinta.

PSI-viestejä on kahdenlaisia: kohdistettuja ja kuulutettuja. Kohdistetut viestit lähetetään suoraan jollekin toiselle PSI-sovellukselle. Vastaanottajan tulee toteuttaa kyseiselle viestityypille viestinkäsittelijä. Vastaanottajan on vastattava kohdistettuun viestiin, sillä lähettäjän suoritus on odotustilassa vastaukseen asti. Tästä syystä kohdistetuilla viesteillä on aikakatkaisu, joka aiheuttaa kohdistetun viestin epäonnistumisen, mikäli viestiin ei lähetetä vastausta tietyn ajan kuluessa. Tällöin lähettäjän on reagoitava virheelliseen lähetykseen.

Kuulutetut PSI-viestit lähetetään globaalisti kaikille peliautomaatissa pyöriville PSI-sovelluksille. PSI-sovellusten on itse toteutettava kuulutettavien viestien tilaaminen, mikäli viestin sisältö kiinnostaa. Tilaaminen aiheuttaa sen, että tilatun viestin saapuessa PSI-sovelluksen sisältämä viestittäjä aiheuttaa tapahtuman PSI-sovelluksen viestinkäsittelijään, josta viesti käsitellään samalla tavalla kuin kohdistetut viestit. Kuulutetun viestin lähettäminen ei jätä lähettävän PSI-sovelluksen viestittäjää odottamaan vastausta, vaan ohjelman suoritus jatkuu välittömästi. Tästä syystä kuulutettuja viestejä voidaan myös ajastaa lähtemään tietyn ajan kuluttua, mitä kohdistetuilla viesteillä ei voi tehdä.



Kuva 2: PSI-sovellusten käynnistyksen kättelysekvenssi.

Kuvassa 2 vasemmalla puolella esitetty Startup on myös PSI-sovellus, mutta kyseinen moduuli poikkeaa kriittisesti muista moduuleista. Startupin tehtävä on käynnistää muut moduulit sekä valvoa, että jokainen moduuli käynnistyy oikein. Oikealla puolella esitelty PSI-sovellus tarkoittaa yleisesti kaikkia muita PSI-sovelluksia paitsi Startupia.

Jokaisen PSI-sovelluksen on omien tietorakenteiden alustamisen ja valmistautumisen jälkeen lähetettävä MSG_MODULE_STARTED-viesti Startup-moduulille. Viestin tarkoitus on kertoa Startupille, että uusi moduuli on käynnistetty ja olisi valmis suorittamaan oman alustusvaiheensa. Tässä vaiheessa moduuli sijoitetaan prosessilistalle tilassa FORKED, joka kertoo sen, että moduuli on olemassa, muttei valmis.

Luettuaan MSG_MODULE_STARTED-viestin, Startup lähettää kyseiselle moduulille MSG_MODULE_INIT:n, jonka seurauksena moduulin tulisi tehdä kaikki alustustoimet, jotka vaativat PSI-viestien lähettämistä. Valmistuessaan moduuli lähettää paluuviestinä MSG_MODULE_INIT_REPLY:n, joka kertoo Startupille, että moduulin alustustoimet on tehty.

Alustustoimien jälkeen Startup pyytää moduulia suorittamaan käyttökokeen MSG_MODULE_DO_SELFTEST-viestillä, joka moduulista riippuen voi olla lähes mitä vain. Moduuli palauttaa MSG_MODULE_SELFTEST_REPLY-viestillä käyttökokeen tulokset. Käyttökokeen jälkeen moduuli siirretään prosessilistassa RUNNING-tilaan, mikä tarkoittaa sitä, että moduuli on valmis palvelemaan sille lähetettyjä pyyntöjä.

MSG_MODULE_INIT ja MSG_MODULE_DO_SELFTEST -viesteihin vastataan periaatteessa totuusarvoilla OK/ERROR. Peliautomaatin vikajärjestelmän takia käytännössä kaikissa virhetilanteissa lähetetään kuitenkin vastauksena OK. Erikseen lähetetään heti perään vikailmoitus, joka aiheuttaa automaatin uudelleen käynnistymisen tai sulkeutumisen. Jos jompaan kumpaan viesteistä vastataan ERROR, aiheuttaa se peliautomaatin uudelleen käynnistämisen, mistä tuskin on siinä tilanteessa hyötyä. Sulkevan vian lähettäminen sen sijaan kutsuu huoltomiehen paikalle selvittämään ja korjaamaan vian. Vikailmoitukset tehdään PSI-viesteillä, jotka *ErrorClient*-niminen moduuli vastaanottaa ja käsittelee.

2.3 Toimilaitteet

Peliautomaatin toiminnallisuuden kannalta oleellisia toiminnallisuuksia ei kuitenkaan voida toteuttaa suoraan pelkällä tietokoneella ja siihen kirjoitetulla ohjelmistolla. Lisäksi siihen tarvitaan ulkoisia toimilaitteita, joiden avulla peliautomaatin toiminnallisuutta täydennetään. Toimilaitteet on kytketty peliautomaatin keskusyksikköön perinteisen sarjaportin sekä USB-portin avulla. Taulukossa 1 on nimetty yleisimmät toimilaitteet lyhyen kuvauksen kanssa.

Taulukko 1: Peliautomaattien yleisimmät toimilaitteet.

Toimilaite	**Kuvaus**
Kolikkolukko	Mahdollistaa kolikoiden käytön pelirahana.
Maksupääte	Mahdollistaa pelirahan ostamisen pankkikortilla.
Setelivalitsin	Mahdollistaa setelien käytön pelirahana.
Hopperit	Säilyttävät automaattiin syötettyjä kolikoita ja mahdollistavat kolikoiden ulosmaksamisen.
Painikkeet	Mahdollistavat pelien ja valikkojen kontrolloinnin fyysisin näppäimin.
Kosketusnäyttö	Mahdollistaa pelaajan syötteet suoraan ruudulta.
Lamppuohjain	Mahdollistaa automaatin valojen ohjauksen.
Kolikkovaihde	Huolehtii, että hoppereista tiputettavat kolikot tippuvat oikeisiin kassasäiliöihin tai pelaajalle.

Jotta peliautomaateilla pystytään pelaamaan, on pelaajan syötettävä peliautomaattiin rahaa. Tällä hetkellä RAY:n peliautomaatit tukevat kolmea erilaista sisään tulevan rahan hyväksymismenetelmää ja näistä jokaista varten on kehitetty oma toimilaite. Yleisin näistä kolmesta on *kolikkolukko*-niminen toimilaite, joka huolehtii sisään tulevien kolikoiden laskemisesta ja validoinnista. Lisäksi useisiin automaatteihin on myös kytketty *maksupääte*, jonka avulla pelaajat voivat ostaa pelejä pankkikorttia käyttäen. Samaan kategoriaan kuuluu myös vastikään laajempaan levitykseen lähtenyt *setelivalitsin*, jonka vastuualueeseen kuuluu sisään tulevien setelien laskeminen ja validoiminen.

Sisään tulevan rahan lisäksi peliautomaatin on pystyttävä selviytymään ulos syötettävästä rahasta eli voitonmaksuista. Kolikoilla ja seteleillä pelattaessa voitonmaksut palautetaan kolikkoina, jolloin peliautomaatissa olevat *hopperit* tiputtavat voittoa vastaavan rahasumman *kolikkovaihteen* kautta pelaajalle. Toistaiseksi ei ole vielä mahdollista saada voittoja seteleinä takaisin. Mikäli pelaaja on ostanut pelinsä käyttäen maksupäätettä, saa hän voittonsa takaisin suoraan tilillensä. Kolikkovaihde huolehtii, että hoppereista tiputettava raha tippuu oikeaan osoitteeseen. Hoppereiden täyttyessä

kolikoita aletaan tiputtaa kassasäiliöön, jolloin kolikkovaihteen tehtävänä on jakaa hoppereilta tulleet kolikot oikeisiin kassapusseihin.

Peliautomaatti ottaa vastaan pelaajan syötteitä joko fyysisiltä *painikkeilta* tai nykyisin uudemmissa malleissa myös *kosketusnäytöltä*. Syötteet liittyvät pelien valitsemiseen, pelin sisäisiin kontrolleihin sekä ohjeistuksiin.

Toimilaitteita varten peliautomaatissa on toimilaitekortteja, joiden avulla toimilaitteita voidaan ohjelmallisesti ohjata. Joillain toimilaitekorteilla voidaan ohjata useampia toimilaitteita. Kahdella näistä toimilaitekorteista sijaitsee Non-Volatile Random Access Memory, eli NVRAM (suomennettuna pysyväismuistipiiri).

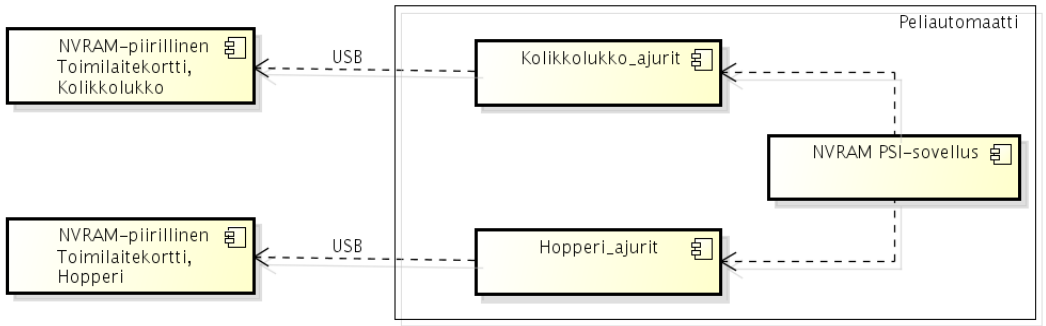
2.4 NVRAM

NVRAM eli pysyväismuisti on elektroninen muistipiiri. NVRAM-piiri mahdollistaa muistista lukemisen sekä muistiin kirjoittamisen niin kutsutulla satunnaishaulla. Satunnaishaulla tarkoitetaan sitä, että muistin fyysisiin osoitteisiin pystytään suoraan osoittamaan yksiselitteisesti jollakin muistiosoitteella [14]. Pysyväismuistia ei pidä sekoittaa tavalliseen RAM-muistiin, joka myös tukee satunnaishakua muistiosoituksissaan. RAM-muisti kadottaa tiedot sillä hetkellä, kun piiristä katkeaa virta, toisin kuin NVRAM-piirissä. Pysyväismuistin yleisin toteutus on, että NVRAM-piiriin on koko ajan kytketty erillinen paristo, jonka avulla piirissä kulkee koko ajan virta. Tämä ei ole kuitenkaan ainoa tapa tuottaa pysyväismuistipiirejä. Esimerkiksi *ferrosähköisyyttä* [18] hyödyntämällä ollaan pystytty tuottamaan NVRAM-piirejä, joissa tieto säilyy pitkiä aikoja jopa ilman virtaa dielektrisen aineen avulla [4; 2]. Pysyväismuistipiirien muistikapasiteetti on yleensä paljon pienempi kuin tavallisissa RAM-piireissä.

Satunnaishaku on myös se suurin ero verrattuna kiintolevyihin, joissa luvut ja kirjoitukset tehdään fyysistä lukupäätä siirtämällä levyn pinnan lähellä. Satunnaishakua tukevat muistit ovat yleensä myös kertaluokkaa nopeampia kuin tavanomaiset kiintolevyt johtuen mekaanisten osien puuttumisesta [7].

NVRAM-muistia ei pidä myöskään sekoittaa flash-muistiin, joka on myös yhdenlaista pysyväismuistia. Flash-muistien valmistamiseen käytetään elektronisesti tyhjennettäviä ja ohjelmoitavia muistipiirejä. Pääasiallisesti niitä ei ole tehty kestämään toistuvaa kirjoittamista. Valmistajasta riippuen flash-muisti kestää 10 000 - 100 000 uudelleenkirjoitusta [3].

RAY:n peliautomaateissa pysyväismuistiin tallennetaan paljon kriittistä tietoa, joita ei saa menettää peliautomaatin sammuessa odotetusti, saati odottamattomasti. Kuva 3 esittää pysyväismuistin hallinnointiin liittyvän komponenttikaavion.



Kuva 3: Komponenttikaavio NVRAM PSI -sovelluksesta toimilaitekortteihin.

Nykyisellään konkreettisia pysyväismuistipiirejä on kaksi, ja ne sijaitsevat kolikkolukon ja hopperien toimilaitekorteilla. Molemmilla toimilaitteilla on oma ajurimoduuli, joka toteuttaa konkreettiset kirjoitukset USB:n yli. PSI-sovellusten ei kuitenkaan tarvitse itse kutsua kyseisiä ajureita, vaan kirjoituksia hallinnoidaan siihen räätälöidyllä moduulilla, jonka luotettavuutta tässä tutkielmassa halutaan parantaa. Kyseessä on siis NVRAM-moduuli. NVRAM-moduulin tarkoitukseen ja heikkouksiin palataan tarkemmin luvuissa 3.1 ja 3.3.

3 NYKYINEN TOTEUTUS

Nykyinen NVRAM-moduuli on ollut Raha-automaattiyhdistyksen PSI-sukupolven peliautomaattien yksi tärkeimmistä moduuleista jo reilun seitsemän vuoden ajan. Värikkääseen historiaan kuuluu paljon kaikenlaista. Tämän luvun tarkoituksena on selventää, mihin moduulia tarvitaan, kuinka se on toteutettu ja minkä takia moduuli halutaan korvata uudella.

3.1 Pysyväismuistimoduulin tarkoitus

Pysyväismuistimoduulin tehtävänä on tarjota peliautomaatissa pyöriville prosesseille pienehkö säilytyslokero. Pysyväismuistia käyttäviä moduuleita kutsutaan asiakasmoduuleiksi. Asiakasmoduulit tallentavat pysyväismuistiin usein muuttuvia arvoja, joita ei saisi kadottaa odottamattomien virran menetysten seurauksena. Myös peliautomaattikohtaisia arvoja voidaan tallentaa muistilohkoihin. Näiden avulla pystytään tunnistamaan ja ylläpitämään peliautomaatin identiteettiä. Peliautomaatista löytyy kaksi pysyväismuistipiiriä, ja ne sijaitsevat kahdella eri toimilaitekortilla.

Peliautomaatin pääasiallisena tallennusmediana käytettävä CompactFlash-muistikortti kestää rajallisen määrän kirjoituksia, minkä takia tallennusmedialle tehtävien kirjoitusten määrä halutaan minimoida. Samasta rajoituksesta kärsii myös toimilaitekorteilla sijaitsevat NVRAM-piirit. Poikkeavan valmistustekniikan ansiosta NVRAM-piirit kestävät reilusti suuremman määrän kirjoituksia kuin CompactFlash-muistikortit. Kahden erillisen

pysyväismuistipiirin avulla muistin sisältöä ei myöskään menetetä, vaikka toinen pysyväismuistipiiri hajoaisikin. Rikkoutunut piiri pystytään vaivattomasti vaihtamaan uuteen, jolloin ehjältä piiriltä pystytään palauttamaan uudelle piirille kaikki vanha tieto.

Pysyväismuistimoduulin toiminnalle asetettiin myös muita toiminnallisia vaatimuksia. Taulukossa 2 on lueteltu nykyisen eli NVRAM 1.0 -moduulin toiminnalliset vaatimukset.

Taulukko 2: Nykyisen moduulin toiminnalliset vaatimukset.

Vaatimus	**Selitys**
LF01	Jokaiselle asiakasmoduulille on pystyttävä tarjoamaan luku- ja kirjoitusrajapinta haihtumattomaan pysyväismuistiin.
LF02	Moduulin tulee estää virheellisten lukujen ja kirjoitusten suorittaminen.
LF03	Moduulin tulee suorittaa muistikopioiden hallintatoimet, kuten formatointi ja uusien muistilohkojen luonti.
LF04	Moduulin on pystyttävä säilyttämään peliautomaatin identiteettiä.
LF05	Moduulin on pystyttävä havaitsemaan vikatilanteet.
LF06	Moduulin on pystyttävä vikatilanteissa palautumaan hukkaamatta tietoja.

Asiakasmoduulien on pystyttävä yksinkertaisella tavalla päivittämään lohkonsa datasisältöä ilman, että niiden tarvitsee tuntea ajureita ja tiedon tallennusmuotoa. Vastaavasti asiakasmoduulien on pystyttävä pyytämään lohkonsa datasisältö tuntematta ajureita ja tiedon tallennusmuotoa. NVRAM-moduulin tehtävä on palvella asiakasmoduulien kirjoitus- ja lukupyyntöjä yksinkertaisen viestirajapinnan avulla (LF01).

Virheellisillä luvuilla ja kirjoituksilla tarkoitetaan operaatioita, jotka kohdistuvat lohkoihin, joita ei ole olemassa tai joissa yritetään suorittaa operaatiota olemassa olevan lohkon yli. Esimerkiksi jos asiakasmoduulilla A on käytössä 16 tavua ja sen lohkosta yritetään lukea 24 tavua, on kyseessä virheellinen lukuoperaatio (LF02).

Peliautomaatissa pyörivää ohjelmistoa kehitetään jatkuvasti ja välillä luodaan uusia PSI-sovelluksia, joiden olisi tarvetta saada oma datalohko pysyväismuistista. NVRAM-moduulin on pystyttävä tarjoamaan kenelle tahansa tarvitsevalle oma lohkonsa, mikäli muistia on jäljellä (LF03).

Jokaisella peliautomaatilla on oma yksilökohtainen tunnistenumeronsa, jonka perusteella voidaan paikantaa, missä mikäkin peliautomaatti sijaitsee. Tieto on varsin tärkeä tilanteissa, joissa peliautomaatin toiminnassa ilmenee vikoja, joiden korjaamiseen tarvitaan huoltomiestä. Identiteetistä on pidettävä kiinni myös tilanteissa, joissa peliautomaatin osia joudutaan vaihtamaan (LF04).

NVRAM-moduulin perimmäinen tarkoitus on varmentaa asioita, joita ei olisi hyvä menettää odottamattomien tai odotettujen uudelleenkäynnistymisten sattuessa. Odottamattoman vikatilanteen sattuessa on kuitenkin pystyttävä huomaamaan tapahtuma ja pystyttävä toimimaan tilanteen mukaisesti (LF05, LF06). NVRAM-moduulin on pystyttävä tunnistamaan, että jokin kirjoitus ei ole onnistunut tai on jäänyt kesken. Tämä ongelma on ratkaistu käyttämällä kahta fyysistä toimilaitekorttia, joilla molemmilla sijaitsee oma NVRAM-piiri. Kahden kopion keskinäisten vertailujen avulla NVRAM-moduuli pystyy tunnistamaan poikkeavuuksia muistikopioissa. Peliautomaatin identiteettiin liittyvän ongelman vuoksi pysyväismuistin sisältö on nykyisellään tallennettu kolmeen kopioon, joista ensimmäinen ja kolmas kopio on sijoitettu kolikkolukon toimilaitekortille. Toinen kopio sijaitsee hopperin toimilaitekortilla.

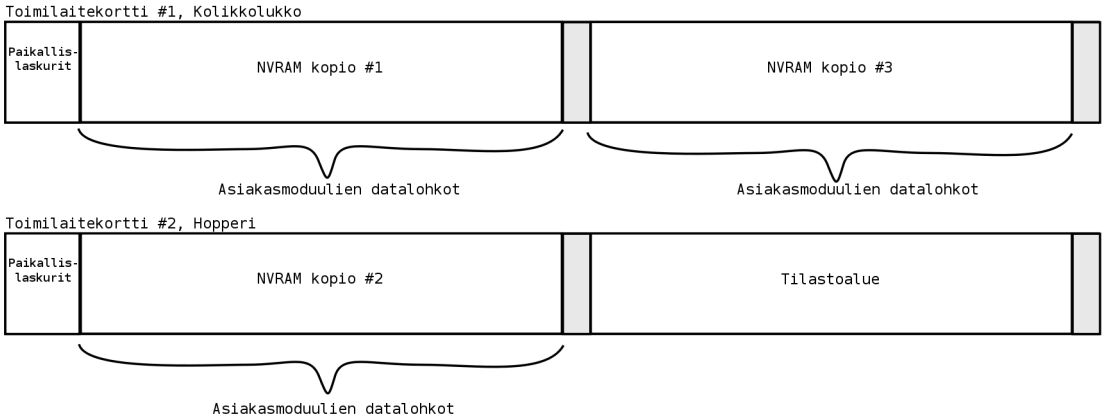
3.2 Rakenne

Nykyinen NVRAM-moduuli on noin 5 000 koodirivin mittainen sovellus. Kyseessä on siis vielä kohtuullisen pieni sovellus, jonka ylläpidettävyyden ei kokonsa takia vielä pitäisi kärsiä. Vaikka moduuli on kirjoitettu C++-kielellä, on se toteutettu pääosin C-mäisesti. Moduuli koostuu yhdestä luokasta, joka sisältää kolme tietuetta, toista sataa jäsenmuuttujaa ja melkein sata funktiota.

Moduulia suoritetaan kahdessa säikeessä: viestisäikeessä ja pääsäikeessä. Viestisäikeessä suoritetaan PSI-viestejä vastaanottavaa silmukkaa, jonka ainoana tehtävänä on vastaanottaa viestejä ja sijoittaa niitä moduulin sisäiseen viestijonoon. Viestisäie estää sen, ettei PSIMessenger-kirjaston käyttämä muisti pääsisi täyttymään puskuroitujen viestien takia. Varsinaisen viestipuskurin sisältö sijaitsee Linuxin kernelin muistiavaruudessa. Kun viestisäie vastaanottaa viestin, kopioidaan sen sisältö moduulin omaan muistiavaruuteen, jolloin kyseinen viesti voidaan poistaa viestipuskurista. Pääsäie sen sijaan suorittaa moduulin toiminnallisuuden, joka pyörii viestijonon ympärillä. Viestijonoa käsitellään kahdessa säikeessä, joten siihen kohdistuvat operaatiot synkronoidaan lukoin. Viestijono on yksisuuntainen, eli viestit sijoitetaan jonon perälle ja viestejä voi saada ulos ainoastaan jonon kärjestä. Pääsäie poistaa viestijonon kärjestä yhden viestin kerrallaan ja prosessoi viestiä vastaavan toiminnallisuuden.

Pääsäikeelle on tunnistettavissa seitsemän erillistä tilaa. Tilojen avulla kontrolloidaan viestisäikeen hyväksymiä viestejä sekä pääsäikeen suoritusta. Suurin osa tiloista liittyy luvussa 2.2 esiteltyyn käynnistymisessä tehtävään kättelysekvenssiin. NVRAM-moduulin käynnistyksen yhteydessä tehtävä alustus on monimutkainen ja monivaiheinen prosessi, jonka aikana suoritetaan useita kyselyitä muille PSI-sovelluksille. Nämä vaiheet on haluttu määritellä omiksi tiloikseen, mutta oikeaa tilakonetta ei kuitenkaan niiden käsittelyä varten ole käytetty. Tilakoneen sijaan toiminnallisuuksia on ehdollistettu useissa funktioissa jäsenmuuttujana olevan tilalipun perusteella.

NVRAM-moduulin vastuualueeseen kuuluu fyysisten NVRAM-muistipiirien hallinnointi. Jotta muistia voidaan hallinnoida, on ensin määriteltävä sille hallinnoitavalle muistille muistimalli. Kuvassa 4 on esitetty nykyisen NVRAM-moduulin hallinnoima muistimalli.



Kuva 4: Nykyinen NVRAM-muistimalli.

Kuvan 4 mukaisen muistimallin perusteena on, että mikäli yksi toimilaite tai toimilaitekortti hajoaa, on toisella kortilla kopio, jolla pystytään palauttamaan pysyväismuistiin tallennettu peliautomaatin identiteetti. Kolmannella kopiolla on haettu lisävarmuutta kopioiden väliseen vertailuun. Mikäli kaksi kopiota täsmää keskenään eikä kolmas kopio täsmää kahden muun kanssa, niin voidaan varmemmin sanoa, että kolmannessa kopiossa on jotain korruptoitunut ja se täytyy korjata. Kaikissa kolmessa kopiossa on lisäksi kolmen toimilaitteen tunnistetiedot, jotta pystyttäisiin tunnistamaan, onko NVRAM-piiriä vaihdettu. Nämä toimilaitteet ovat kolikkolukko, sähköovi ja hopperi. Näiden kolmen toimilaitteen tunnistetietojen perusteella voidaan luotettavasti päätellä, onko yksi toimilaitekortti vaihtunut ja jos on, niin mikä.

Hopperissa on toisen kopion lisäksi vielä tilastoalue, johon tallennetaan peliautomaattikohtaisia tilastoja eri tapahtumista. Tilastotietojen perusteella ei suoriteta mitään logiikkaan liittyviä ohjauksia, joten sitä ei varmenneta toisella kopiolla. Tilastoalueen sisältö lähetetään kuitenkin verkon yli tietyin väliajoin RAY:n palvelimille. Tästä huolimatta tilastoalueen tietoja ei haluta menettää, sillä niiden palauttaminen takaisin kyseiseen peliautomaattiin on mahdotonta.

NVRAM-moduulin sisällä muistimalli on pilkottu useisiin taulukoihin. NVRAM-moduuli säilöö alustuksessa kaikki kolme NVRAM-kopiota muistiinsa. Jokainen kopio koostuu kolmesta osasta, NVRAM-otsikkotiedoista, asiakasmoduulien data-alueista sekä yksittäisten data-alueitten yli lasketuista CRC-tarkisteavaimista (Cyclic Redundancy Check). Taulukoiden tietoja on fyysisten muistikirjoitusten takia myös ryhmitelty kolmen tietueen avulla, joita päivitetään samalla kuin taulukoitakin. Nämä

tietueet pitävät sisällään tietoja tehdyistä kirjoituksista, NVRAM-moduulin otsikkotietoja, asiakasmoduulien data-alueista sekä tarkisteavaimista.

Tilastoaluetta käsitellään samalla tavalla kuin muidenkin asiakasmoduulien data-alueita paitsi, että jäsenmuuttujien ja funktioiden nimet eroavat. Tilastoalueen datasisältö kuitenkin kopioidaan fyysisiä kirjoituksia varten samanlaiseen tietueeseen kuin asiakasmoduulien data-alueet.

3.3 Heikkoudet

Nykyistä NVRAM-moduulia ei ole suunniteltu kovin hyvin, ja se näkyy välittömästi usealla rintamalla. NVRAM-moduuliin kohdistuvia vikoja havaitaan lähes 7000 kuukaudessa, joista noin 5500 tapauksessa suoritetaan NVRAM-muistiin liittyviä korjaustoimenpiteitä. Ongelma kuitenkin on, että näiden korjausten jälkeen ei kuitenkaan olla varmoja NVRAM-muistin sisältöjen oikeellisuudesta. Nykyisen NVRAM-moduulin lähdekoodia täytyi analysoida mahdollisten toimintavirheiden löytämiseksi. Lähdekoodia analysoitiin reilun viikon verran ja kyseisen ajanjakson aikana nykyisen NVRAM-moduulin toiminnassa havaittiin lukuisia virheitä ja puutteita.

Nykyisen NVRAM-moduulin lähdekoodista näki jo kaukaa, että siinä on useita ongelmakohtia, joihin tulisi puuttua. Kuten luvussa 3.2 jo mainittiin, nykyinen NVRAM-moduuli on noin 5000 rivin pituinen komponentti, jossa kaikki koodi sijaitsee yhdessä tiedostossa. Vaikka 5000-rivinen komponentti ei itsessään ole suuri, niin 5000-rivinen luokka on jo massiivinen. Jäsenmuuttujiakin luokalla oli toista sataa, minkä perusteella pystyi jo arvelemaan, että lähdekoodin seassa on paljon kahdennettua koodia [12, s. 78].

Aavistus oli ikävä kyllä oikein, sillä esimerkiksi fyysiselle toimilaitekortille kirjoittava funktio oli kahdennettu. Näissä lähes kahdensadan rivin pituisissa funktioissa oli kolme eroavaisuutta, jotka eivät selitä, miksi funktio oli kahdennettu. Ensimmäinen ero oli funktion nimi. Ensimmäisen funktion nimi viittasi kirjoitukseen, kun toisen funktion nimi viittasi samaan kirjoitukseen mutta ilman CRC-tarkisteavainta. Näin asia olikin, sillä toisena erona oli ajurille lähetettävän viestin tyyppi. Viestin sisäisellä datarakenteella ei kuitenkaan ollut mitään eroa, joten toimilaitteiden ajureille olisi ollut aivan sama, kumpaa viestiä käytetään. Lisäksi kolmantena erona oli ajurille lähetettävässä viestissä CRC-kentän täyttäminen. Normaalissa kirjoitusfunktiossa CRC-kenttään laskettiin ja kopioitiin CRC-tarkisteavaimen arvo, mutta CRC:ttömässä kirjoitusfunktiossa kyseiseen kenttään sijoitettiin nollaa. Kahdentamiselta oltaisiin vältytty ehdollistamalla CRC-kentän täyttäminen. Lähdekoodissa tehtiin myös paljon tarkistuksia joidenkin muuttujien arvoille. Samat tarkistukset löytyivät useiden funktioiden seasta joissain jopa useita kertoja. Nämä tarkistukset olisi selkeämpi irrottaa omaksi kuvaavasti nimetyksi tarkistusfunktioksi, jota sitten kutsuttaisiin useita kertoja.

Monistettu koodi ja pitkät funktiot ovat suurin syy lähdekoodin hankalalukuisuuteen ja ylläpitokelvottomuuteen, mutta ei luettavuutta ainakaan yhtään paranna lähdekoodista löytyvät taikanumerot ja harhaanjohtavat kommentit. Nvram-luokan jäsenmuuttujina olevia taulukoita käsiteltiin vaihtelevasti taikanumeroin ja kuvaavin vakioin. Hyvänä ohjeena pidetään, että mikäli johonkin tarvitsee asettaa jokin numeerinen arvo tai taulukon jotain tiettyä indeksiä täytyy käsitellä, tulisi kyseinen numeerinen arvo määritellä vakioksi [15, s. 19]. Varsinkin tämmöinen vakioiden ja taikanumeroiden sekakäyttö aiheuttaa turhautuneisuutta ja ahdistuneisuutta lähdekoodin toimintaa lukiessa. Jatkuvasti joutuu miettimään, “miksi juuri tähän alkioon asetetaan juuri tuo arvo”. Joissakin näistä tilanteista taikanumeroita on yritetty selventää sopivilla kommenteilla, mutta valitettavasti kommenttien sisältö ei kuitenkaan aivan täsmää suoritettavan lähdekoodin kanssa. Kommenteissa kerrotaan, mitä tehdään, mutta oikeasti tehdäänkin juuri päinvastainen operaatio. Hyvän kommentin ei muutenkaan pitäisi vastata kysymykseen *mitä* vaan *miksi*. Koodin pitäisi itse pystyä omalla ilmaisuvoimallaan kertoa mitä ollaan tekemässä, mutta syy voi olla silti epäselvä [15; 12, s. 85].

NVRAM-moduulia haluttiin alunperin elvyttää refaktoroimalla, eli ohjelman rakenteen ehostamisella ilman, että ohjelman toiminnallisuus muutetaan [12; 11, s. 5]. Lähdekoodin analysointi kuitenkin osoitti, että refaktoroiminen olisi liian haastavaa ja aikaa vievää. Refaktoroinnin perustana ovat hyvät yksikkötestit, joita peliautomaattiympäristö ei nykyisellään tue laisinkaan. Tästä syystä NVRAM-moduulille ei ole yksikkötestejä kirjoitettu. Yksikkötestit pitäisi kuitenkin ensin toteuttaa, jotta refaktorointia voitaisiin aloittaa tekemään. Koska peliautomaatissa ei yksikkötestejä pystytä ajamaan, pitäisi NVRAM-moduulia suorittaa omalta työkoneelta, mikä taas vaatisi NVRAM-moduulin riippuvuuksien tuomista kehitysympäristöön. Ongelmaksi kuitenkin muodostuu riippuvuuksina olevat laiteajurit, joita ei aivan tuosta noin tuodakaan kehitysympäristöön. Riippuvuuksia voidaan kyllä poistaa kommentoimalla koodia ulos tai käyttämällä ei-toiminnallisia versioita tarvituista moduuleista. Riippuvuuksien poistaminen vaatii kuitenkin muutoksia jo alkuperäiseen lähdekoodiin, mutta lähdekoodia ei voida muuttaa ilman yksikkötestejä. Tämä johtaa sykliseen riippuvuuteen: jotta refaktorointia voitaisiin tehdä, tarvitsee kirjoittaa alkuperäistä koodia varten yksikkötestejä, mutta yksikkötestien tekeminen vaatii alkuperäisen koodin refaktorointia.

Edellä mainitut heikkoudet ovat olleet enemmänkin tyylillisiä heikkouksia, mutta myös muutamassa kriittisessä toiminnallisuudessa havaittiin heikkouksia. Virheiden tunnistamisen kanssa oli havaittavissa selviä puutteita. Funktioista palautetaan ahkerasti vikakoodeja virheitä havaittaessa, mutta paluuarvoja ei tarkisteta kutsujan puolella laisinkaan. Tämä johtaa siihen, että virhetilanteissa jotain jää aivan selvästi suorittamatta, mutta asiakasmoduulille tämä tieto ei koskaan välity. Tästä syystä suurin osa ajonaikaisista virheistä jää huomaamatta täysin, joten kuukaudessa havaittavien virheiden määrä voisi helposti olla jopa kaksinkertainen. Virheet huomataan aikaisintaan, kun NVRAM-moduulia käynnistetään seuraavan kerran ja tällöinkin ainoastaan NVRAM-muistissa oleviin datasisältöihin kohdistuvat virheet huomataan. Jokaisella

asiakaslohkolla on oma CRC-tarkisteavain, jonka perusteella NVRAM-moduuli pyrkii huomaamaan korruptoituneita kirjoituksia peliautomaatin käynnistyksen yhteydessä. Kolmesta kopiosta etsitään käynnistyksen yhteydessä se kopio, josta löytyy NVRAM-moduulin mielestä parhain eli ehein kopio. Nykyisen NVRAM-moduulin mukaan kopion ei tarvitse olla täysin ehjä, että sitä voidaan käyttää.

Toinen ja ehkä kriittisin toiminnallinen virhe löytyy nykyisen NVRAM-moduulin kirjoituslogiikasta. Kirjoitukset tehdään nykyisellään aluksi NVRAM-moduulin omiin muistikopioihin, jonka jälkeen kirjoitukset lähetetään toimilaitekorttien ajureille. Kirjoitukset lähtevät toimilaitekorteille lähes täysin samanaikaisesti. Ajureilta ei odoteta mitään paluuarvoa kirjoitusten onnistumisesta, joten NVRAM-moduuli palauttaa välittömästi ajureille lähteneiden kirjoitusviestien jälkeen kutsuvalle asiakasmoduulille, että kirjoitus on nyt tehty. Koska kirjoituspyynnöt saapuvat toimilaitekorttien ajureille samanaikaisesti, tarkoittaa se sitä, että kirjoituksia aletaan tehdä kahdella eri toimilaitekortilla täsmälleen samaan aikaan. Samanaikaisuus johtaa hyvin helposti siihen, että kaksi kopiota on rikki ja täysin eri tilassa, mikäli peliautomaatista katkeaa sähköt kesken kirjoituksen. Jotta kirjoitusten onnistumisesta voitaisiin olla varmoja, tulisi toimilaitteille tehtävät kirjoitukset suorittaa yksi kerrallaan.

Kolmantenta epäluotettavana toiminnallisuutena voidaan pitää pysyväismuistin formatointia. Pysyväismuistin formatointia tarvitaan tilanteissa, joissa jostain syystä NVRAM-muisti on mennyt niin sekaisin, ettei millään korjauksella pystytä palautumaan toimintakykyiseksi. Nykyisellään näistä syistä johtuneita formatointeja joudutaan suorittamaan noin sata kertaa kuussa. Pysyväismuisti formatoidaan myös ennen kuin peliautomaatti tuodaan tehtaalta levitykseen. Vaikka pysyväismuistin formatointi on kohtuullisen harvinainen tapahtuma, niin sen pitäisi silti toimia mahdollisimman varmasti. Nykyinen pysyväismuistin formatointilogiikka pohjautuu siihen, että tietty määrä kirjoituksia ehditään tehdä 20 sekunnin sisällä. Formatointipyynnöstä lähetetään Control-moduulille viesti. Control-moduuli lukee viestin ja kuuluttaa vastaavanlaisen viestin kaikille peliautomaatissa suoritettaville moduuleille. Mikäli moduulia kiinnostaa määritellä oman NVRAM-lohkonsa oletussisältö, on kyseisen asiakasmoduulin lähetettävä NVRAM-moduulille kirjoitusviesti, jossa oletusdata annetaan mukana. Control-moduuli aiheuttaa peliautomaatin uudelleen käynnistymisen 20 sekunnin jälkeen kuulutuksestaan. NVRAM-moduulin on siis ehdittävä kirjoittamaan tämän 20 sekunnin aikana tuntematon määrä moduulien oletustietoja. Kirjoitusten onnistumisesta ei saada mitään tietoa, koska asiakasmoduuleilta tulevien kirjoitusten määrä on täysin tuntematon. Lähes kaikki näistä asiakasmoduuleista, jotka formatointipyyntöön vastaavat, kirjoittavat useissa palasissa omat oletustietonsa.

Nykyinen NVRAM-moduuli on ylläpidettävyyden kannalta huonolla mallilla. Lähdekoodin luettavuus on haastavaa sekavien kommenttien, taikanumeroiden, pitkien funktioiden ja monistetun koodin takia. Lähdekoodin siistimistä haittaa yksikkötestien puute. Peliautomaattiympäristön tuomien riippuvuuksien takia niiden mukaan tuominenkin

on hyvin haastavaa. Osa nykyisen NVRAM-moduulin kriittisistä toiminnoista toimii jokseenkin hyvällä tuurilla, mikä ei missään nimessä olisi sallittavaa. Uudessa NVRAM-moduulissa on ehdottomasti panostettava kirjoituslogiikan, virheentunnistuksen sekä ylläpidettävyyden kehittämiseen. Luvussa 4 pohditaan, kuinka uudesta NVRAM-moduulista saataisiin parempi ja luotettavampi.

4 UUSI NVRAM-MODUULI

Nykyinen NVRAM-moduuli on tarkoitus korvata uudella moduulilla ilman, että muuhun järjestelmään tarvitsee tehdä muutoksia. Viestirajapinnan on pysyttävä samana, mutta samaan aikaan uusia ominaisuuksia halutaan lisätä, moduulin ylläpidon tulisi olla kevyempää sekä luotettavuuden tulisi parantua. Tässä luvussa käydään läpi uudelle moduulille asetetut vaatimukset, arkkitehtuurin pääpiirteet sekä suurimmat toiminnalliset muutokset, joiden olisi tarkoitus taata luotettavammin toimiva kokonaisuus.

4.1 Vaatimukset

Uuden NVRAM-moduulin on täytettävä kaikki nykyisen NVRAM-moduulin toiminnalliset vaatimukset. Nykyisen moduulin toiminnalliset vaatimukset on esitelty taulukossa 2. Näiden lisäksi uudelle moduulille määriteltiin lisävaatimuksia, joiden tarkoitus on laajentaa ja parantaa pysyväismuistin toiminnallisuutta. Uudet toiminnalliset vaatimukset on esitelty taulukossa 3.

Taulukko 3: Uuden moduulin toiminnalliset vaatimukset.

Vaatimus	**Selitys**
F01	Viestirajapinta yhteensopiva nykyisen moduulin kanssa.
F02	Nykyisen muistimallin (kuva 4) konvertointi uuteen muistimalliin (kuva 5).
F03	Kirjoitusten onnistuminen varmennettava.
F04	Moduulin tulee olla haihtumattoman muistin määrästä riippumaton.
F05	Yksittäisille asiakasmoduuleille on pystyttävä tarjoamaan yli neljän kilotavun kokoisia muistilohkoja (rajoitettu vanhassa neljään kilotavuun).

Yhteensopivuus nykyisen moduulin kanssa halutaan säilyttää, joten uuden pysyväismuistimoduulin on pystyttävä korvaamaan vanha pysyväismuistimoduuli ilman, että asiakasmoduuleihin tarvitsee tehdä muutoksia (F01).

Uuden moduulin tarkoitus on korvata nykyinen moduuli niissäkin peliautomaateissa, jossa nykyinen on jo käytössä. Näissä tilanteissa datamalli on pystyttävä konvertoimaan nykyisestä datamallista uuteen, sillä datamallit poikkeavat toisistaan merkittävällä tavalla. Konversion tehtävä on säilyttää aiemmin kirjoitetut tiedot, mutta muuttaa niiden säilytysmuoto siihen muotoon, mitä uusi moduuli tukee ja käyttää (F02).

Luvussa 3.3 todettiin, että kirjoitusten onnistumisesta ei tähän mennessä ole lähetetty minkäänlaista paluuviestiä, joten tähän halutaan muutos. Tästä lähtien pysyväismuistimoduulin on pystyttävä ilmoittamaan epäonnistuneista kirjoituksista (F03). Kirjoituslogiikan luotettavuuden parantamista käsitellään tarkemmin luvussa 5.1.

Nykyinen pysyväismuistimoduuli toteutettiin silloisia 32 kilotavun NVRAM-piirejä varten, mutta tulevaisuudessa halutaan käyttää mahdollisesti myös suurempia piirejä. Tästä syystä uuden moduulin on pystyttävä hyödyntämään NVRAM-piirin koosta riippumatta koko muistialuetta (F04).

Nykyinen moduuli rajoittaa asiakasmoduulien lohkojen kooksi neljä kilotavua, mutta tälle rajoitukselle ei ole mitään kunnollisia perusteluita. Uuden moduulin tulisi poistaa tämä rajoitus (F05).

Toiminnallisten vaatimusten lisäksi projektille asetettiin muutama ei-toiminnallinen vaatimus. Ei-toiminnalliset vaatimukset on esitelty taulukossa 4 ja selitetty tarkemmin taulukon jälkeen.

Taulukko 4: Uuden moduulin ei-toiminnalliset vaatimukset.

Vaatimus	**Selitys**
NF01	Luotettavuuden parantaminen
NF02	Virheiden raportointi
NF03	Dynaamisuus
NF04	Ylläpidettävyys
NF05	Suorituskyky

Nykyisen moduulin luotettavuus ei ole aivan parhaimmasta päästä. Monien ohjelmointivirheiden seurauksena pysyväismuistien korruptoituminen on aivan liian yleistä. Moduulin tehtävähän on estää korruptoitumisten tapahtuminen, mutta esimerkiksi päällekkäiset kirjoitukset suorastaan kerjäävät muistikorruptioiden tapahtumista. Varsinkin näihin ongelmiin on puututtava uuden moduulin toteutuksessa (NF01). Vikasietoisuuden kehittämistä käsitellään luvussa 5.

Tiedetään varmasti, että virheistä ei voida mitenkään päästä eroon sataprosenttisen varmasti. Virheisiin pitää kuitenkin varautua ja ne pitää osata tunnistaa ajonaikana. Jos

ja kun virheitä pääsee tapahtumaan, on niistä lähetettävä virheilmoitus peliautomaattien virhejärjestelmään mahdollisimman hyvin yksilöivillä tiedoilla (NF02). Nykyisestä NVRAM-moduulista ei saada kattavia virheilmoituksia virheiden sattuessa.

Nykyisen moduulin kanssa ongelmaksi on muodostunut sen dynaamisuuden puute (NF03). Vaikka moduuli määritteleekin monia muuttujia XML-parametreissaan, jotka luetaan moduulin käynnistyessä, niitä ei kuitenkaan käsitellä oikealla tavalla. Osa kriittisistä muuttujista, jotka on määritelty parametreihin, on sen lisäksi kovakoodattu esikääntäjämakroin Nvram-luokan otsaketiedostoon. Näitä kriittisiä tietoja ovat muun muassa pysyväismuistin määrä, asiakasmoduulien määrä, asiakasmoduulien lohkojen maksimikoko sekä tilastoalueen koko. Tämän puutteen korjaaminen nykyiseen järjestelmään ei olisi ongelmallista, mikäli jompaa kumpaa tapaa olisi käytetty yhtenäisesti koko moduulissa. Näin ei kuitenkaan ole, vaan lähdekoodista on luettavissa kolme toisistaan poikkeavaa tyyliä käsitellä yllä mainittuja tietoja. Ensimmäinen on esikääntäjämakrot, toinen on parametrit ja kolmantena numeerisessa muodossa täsmälleen samat numerot. On siis selvää, että puutteen korjaaminen voi aiheuttaa yllättäviä komplikaatioita. Uutta NVRAM-piiriä varten tarvittiin tuki 128 kilotavun kokoiselle pysyväismuistille, mutta moduulin tulee myös toimia nykyisen 32 kilotavun kokoisen NVRAM-piirin kanssa. Lisäksi kokorajoituksista halutaan päästä eroon.

Nykyisen moduulin ongelmia on hyvin haastavaa lähteä korjaamaan, eikä vikakoodienkaan avulla päästä käsiksi kaikkiin ongelmakohtiin. Uuden moduulin ei tulisi kompastua epäselvään lähdekoodiin tai sekaviin tietorakenteisiin. Dokumentaation tulisi myös olla ajan tasalla. Näiden avulla uusien ominaisuuksien lisäämisen pitäisi olla tulevaisuudessa helpompaa (NF04).

Vanhat peliautomaatit asettavat uudelle moduulille suorituskykyyn liittyviä vaatimuksia. Joissakin peliautomaattimalleissa käytetään vielä hyvinkin vanhoja prosessoreita, joten laskentatehon puute voi hyvin äkkiä muodostua pullonkaulaksi. NVRAM-moduulin tulee varmistaa pysyväismuistikopioiden välinen eheys jokaisen kirjoituksen yhteydessä, mutta tämä ei saa näkyä peliautomaatin suorituskyvyssä eikä varsinkaan käyttäjälle eli pelaajalle (NF05).

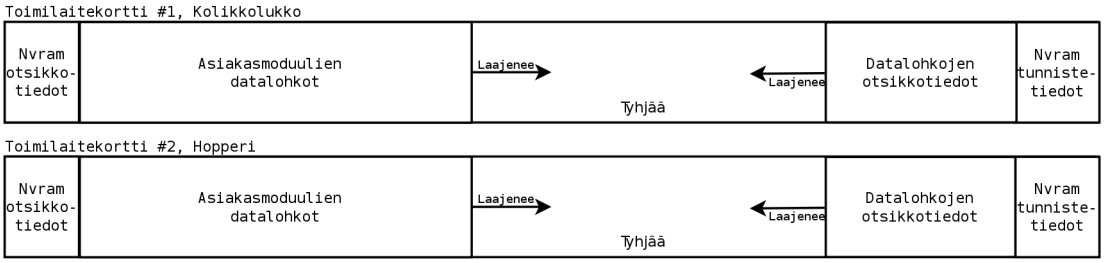
4.2 Datamalli

Uutta moduulia lähdettiin suunnittelemaan jatkokehitettävyys ja luotettavuus mielessä. Ensimmäisenä kehitystiimi kyseenalaisti vanhassa moduulissa käytetyn kolmen kopion mallin (katso kuva 4).

Kehitystiimin mukaan kolmas kopio varaa neljänneksen käytettävästä NVRAM-muistin määrästä turhaan. Myös kolmannen kopion tuoma luotettavuus kyseenalaistettiin. Haihtumaton muistihan sijaitsee kahdella toimilaitekortilla ja nykyisessä muistimallissa ensimmäisellä kortilla säilytettiin kahta kopiota ja toisella kolmatta. Syynä tähän

muistimalliin oli, että kahdella sisällöltään pätevällä kopiolla voidaan aina korvata kolmas kopio, eli yhden kopion korruptoituminen ei tällöin toisi ongelmia. Kolmas kopio ei kuitenkaan tuo mitään lisäarvoa. Mikäli toimilaitekortilla, jolla sijaitsee kaksi kopiota, havaitaan poikkeavuuksia, niin voi olla täysin mahdotonta sanoa, kumpi näistä kopioista on se oikea. Kopioiden korjaaminen pystytään tekemään yhtä hyvin kahdella kopiolla käyttäen hyväksi tarkistesummia ja aikaleimoja.

Uuden muistimallin tilalle pohdittiin korvaavia muistimalleja, jotka pääasiassa pohjautuivat tiedostojärjestelmätyyppisiin ratkaisuihin. Yksinkertaiseen tiedostojärjestelmään olisi suhteellisen yksinkertaista mahduttaa mukaan asiakasmoduulien muistilohkojen sijainnin siirtäminen ja koon muuttaminen dynaamisesti. Tiedostojärjestelmää ei kuitenkaan voitu lähteä toteuttamaan rajallisen muistimäärän takia. Tiedostojärjestelmän vaatimat varausyksiköt olisivat rajoittaneet hyödynnettävän muistin määrää liikaa. Lopulta päädyttiin kuvan 5 mukaiseen muistimalliin.



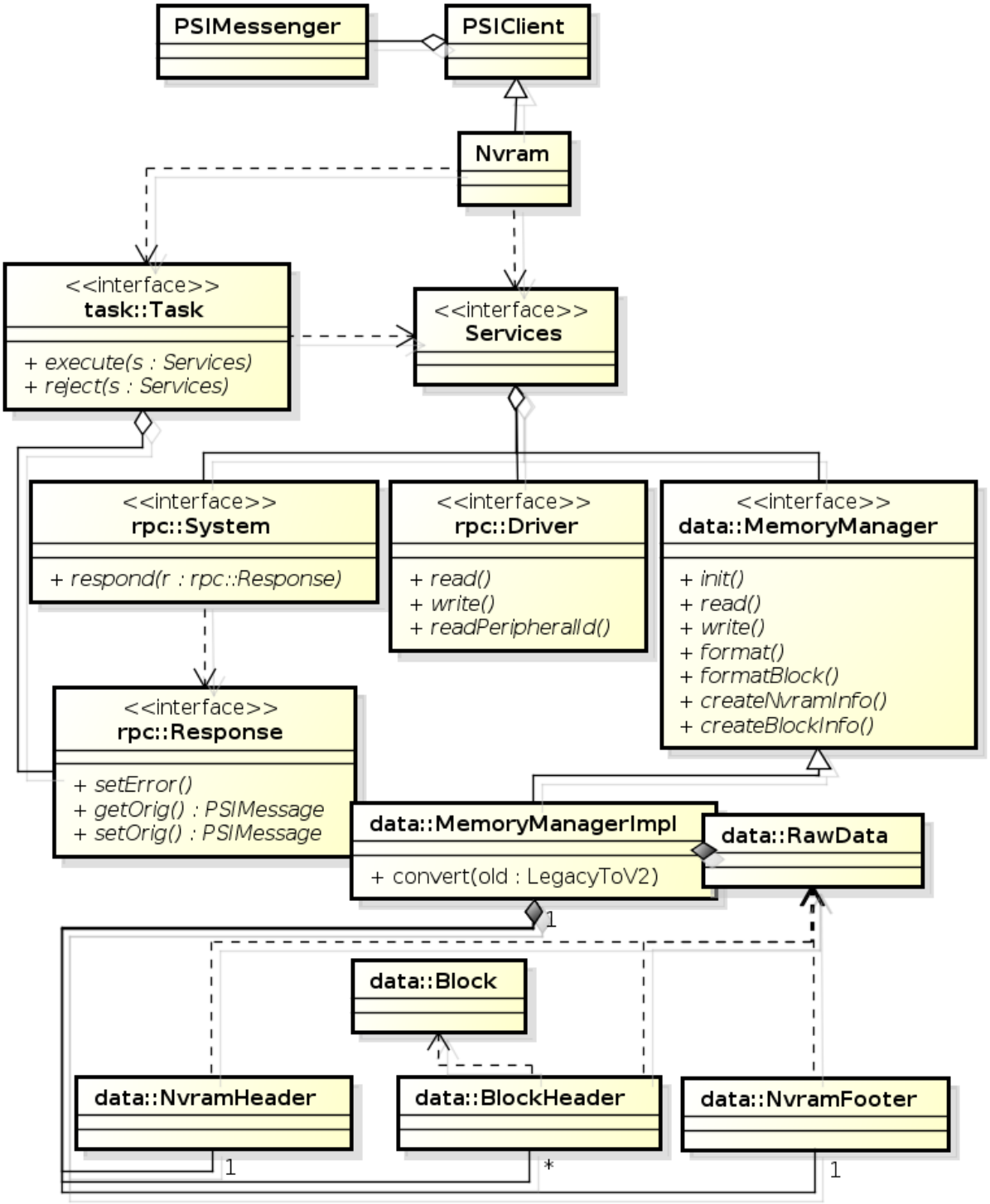
Kuva 5: Uuden arkkitehtuurin mukainen muistimalli.

Kuvassa 5 esitellään kaksi sisällöltään identtistä muistipiiriä. Muistin alkuun kirjoitetaan NVRAM-otsikkotiedot, jotka sisältävät peliautomaatin identiteetin tunnistamisessa käytettävää peliautomaatin numeroa sekä kolmen toimilaitekortin tunnistetiedot. Toimilaitetunnisteita käytetään tunnistamaan vaihtuneet toimilaitekortit. NVRAM-moduulin alustuksessa toimilaitekorttien tunnisteet luetaan pysyväismuistista sekä toimilaitekorteilta, jolloin keskinäisen vertailun avulla saadaan tietää, onko vaihdettuja toimilaitekortteja. Toimilaitekortteja saa vaihtaa vain yhden kerrallaan, mikä mahdollistaa vaihdon tunnistamisen silloin, kun pysyväismuistiin on tallennettu vähintään kolme toimilaitetunnistetta. Otsikkotiedoissa ilmoitetaan tunnisteiden lisäksi myös käytetyn muistin määrä ja mistä datalohkojen otsikkotietojen viimeinen alkio alkaa. Muistin lopusta löytyvät NVRAM-tunnistetiedot, joihin on varattu tilaa tarkistesummalle ja kirjoituslaskurille. Laskettaessa tarkistesumma koko käytettävän muistialueen yli voidaan tarkistesumman avulla yksinkertaisesti päätellä, onko edellinen kirjoitus onnistunut. Mikäli laskettu tarkistesumma täsmää luettuun, on kirjoitus onnistunut. Kirjoituslaskuria sen sijaan käytetään tunnistamaan, kumpi kopioista on uudempi tilanteita varten jossa molemmissa kopioissa on onnistuneet kirjoitukset, mutta kopioiden datasisältö poikkeaa toisistaan. Molempiin kopioihin kirjoitetaan eri arvo kirjoituslaskurille varattuun kenttään.

Lisäksi muistiin kirjoitetut datalohkojen otsikkotiedot muodostavat hakutaulun, jonka avulla etsitään asiakasmoduulien datalohkoja. Hakutaulun alkioihin kirjoitetaan datalohkon fyysinen sijainti, omistaja ja käytettävän tilan määrä. Hakutaulun koko määräytyy asiakasmoduulien määrän mukaisesti ja kasvaa lineaarisesti kohti alkua alkaen NVRAM-tunnistetietojen edeltä. Vastaavasti asiakasmoduulien datalohkot alkavat välittömästi NVRAM-otsikkotietojen jälkeen, ja niiden koot ovat asiakaskohtaisia.

4.3 Luokkahierarkia

Luokkakaaviota suunniteltaessa tuli äkkiä ilmi se, ettei uusi NVRAM-moduuli tule toistamaan vanhan moduulin yhden luokan mallia. Uuden NVRAM-moduulin toteutuksessa päätettiin hyödyntää Gamman oppeja suunnittelumalleista [13]. Yhteneviä toiminnallisuuksia ryhmiteltiin rajapintojen avulla. Rajapintojen avulla pystyttiin väistämään vanhan moduulin määrittelemät PSI-sovellusriippuvuudet. Kuvassa 6 on uuden NVRAM-moduulin luokkakaavio, jossa on esitelty moduulin tärkeimmät rajapinnat ja luokat.



Kuva 6: Uuden NVRAM-moduulin luokkakaavio.

Kuvan 6 perusteella voidaan jakaa NVRAM-moduulin luokkahierarkia neljään eri osa-alueeseen: Muistimanageriin, Järjestelmä-rajapintaan, Ajuri-rajapintaan sekä tehtäviin. Jokaisella näistä neljästä on oma vastuualueensa NVRAM-moduulin toiminnallisuuden toteuttamisessa. Muistimanageri, Järjestelmä-rajapinta sekä Ajuri-rajapinta ovat tehtävien vaatimia palveluita.

4.3.1 Muistimanageri (MemoryManager)

NVRAM-moduulin sisäistä datamallia hallinnoi Muistimanageri, joka koostuu neljästä hallinnointia helpottavasta apuluokasta. Apuluokkien lisäksi Muistimanageri koostuu *RawData*-luokasta, joka sisältää kopion fyysisen NVRAM-muistin sisällöstä. Apuluokat jakavat muistikopion kuvan 5 mukaiseen muistimalliin. Jokainen apuluokka tarjoaa omat luku- ja kirjoitusfunktiot data-alueittensa hallinnointiin.

NvramHeader-luokka määrittelee NVRAM-otsikkotietojen sisäisen datamallin ja tarjoaa Muistimanagerille funktiot, joiden avulla otsikkotietojen päivittäminen onnistuu. Otsikkotietojen päivitykset tehdään transaktiomallisesti. Tämä tarkoittaa sitä, että muutokset tehdään ensin NvramHeaderin sisäiseen välimuistiin. Lopuksi muutokset tallennetaan varsinaiseen *RawData*-muistikopioon.

BlockHeader-luokka määrittelee datalohkojen otsikkotietojen sisäisen datamallin. Lisäksi kyseisen apuluokan avulla Muistimanageri pääsee käsiksi otsikkotietoa vastaavaan datalohkoon. Otsikkotiedot sisältävät tiedon lohkon omistajasta, koosta ja sen sijainnista fyysisellä NVRAM-muistilla. Otsikkotietoja on yhtä monta kappaletta kuin datalohkoja. Otsikkotietojen perusteella muodostetaan hakutaulu, jonka avulla Muistimanageri pystyy etsimään tarvitsemansa asiakaslohkon. Hakutaulun avaimena käytetään asiakaslohkon moduulitunnistetta.

Block-luokka kuvastaa asiakasmoduulin datalohkoa. Tämän luokan ilmentymän avulla tehdään asiakasmoduulin datalohkoihin kohdistuvat kirjoitukset ja luvut. Block-olioita on olemassa yhtä monta kuin BlockHeader-olioita.

NvramFooter-luokka määrittelee NVRAM-muistin eheyteen liittyvät tunnistetiedot. Tunnistetietoja ovat kirjoituslaskuri sekä tarkistesumma koko fyysisen NVRAM-muistin yli (poislukien kirjoituslaskurin ja tarkistesumman). NvramFooter-luokan vastuulla on päivittää omassa hallinnointialueessaan olevaa tarkistesummaa. Tarkistesumman päivitykset tehdään aina kun jotain muutetaan muistikopiossa. Tarkistesumman laskenta-algoritmina käytetään MD4:ää. MD4 tarjoaa riittävän hyvän luotettavuuden NVRAM-muistin eheydelle.

4.3.2 Järjestelmä-rajapinta (System)

Järjestelmä-rajapinnan tarkoitus on abstrahoida kommunikointirajapinta muihin PSI-sovelluksiin päin. Järjestelmä-rajapinta tarjoaa funktiorajapinnan, jonka avulla NVRAM-moduuli pystyy lähettämään pyyntöjä muille moduuleille sekä vastaamaan sille lähetettyihin PSI-viesteihin. Abstrahoinnin avulla voidaan unohtaa peliautomaattiympäristön komponenttiriippuvuudet siihen asti, kunnes NVRAM-moduuli siirretään peliautomaattiympäristöön.

NVRAM-moduulille tulleisiin pyyntöihin vastataan antamalla Järjestelmä-rajapinnan *respond()*-funktiolle parametrinä *Response*-tyyppinen olio. Response eli vastausolio toimii lisäabstraktiona PSI-viestien välittämiselle. PSI-viestien vastausviestit vaativat alkuperäisen viestin parametrikseen, mutta kyseistä konkreettista tietotyyppiä ei haluttu tuoda riippuvuudeksi rajapintaan asti. Rajapinnassa riittää tieto siitä, että kyseinen oliotyyppi on olemassa.

Konkreettisia Response-toteutuksia on useita. Mikäli jokin *tehtävä* vaatii suoriutumisestaan jonkinlaisen vastauksen, tarvitsee tehtävälle toteuttaa oma vastausolio. Vastausolioiden sisäiset toteutukset vaihtelevat riippuen siitä, mitä tehtävän vastaukselta odotetaan. Lukupyynnöissä vastausviestissä lähetetään mukana luettu data, mutta kirjoittaessa vain tieto siitä, onnistuiko kirjoitus.

4.3.3 Ajuri-rajapinta (Driver)

Ajuri-rajapinta abstrahoi toimilaitekorttien ajureille lähtevät kirjoitus- ja lukupyynnöt. Rajapinta paljastaa kolme funktiota: *read()*, *write()* ja *readPeripheralId()*. Näiden kolmen funktion avulla NVRAM-moduuli pystyy ylläpitämään fyysisen NVRAM-muistin tilaa. Jokainen näistä kolmesta funktiosta ottaa sisäänsä parametrina tiedon, kummasta ajurista on kyse.

Lukufunktio read() lähettää toimilaitekortin ajurille pyynnön lukea kyseisen toimilaitekortin NVRAM-muistin sisältö. Sisältö palautetaan takaisin read()-funktion kautta kutsujalle. Näitä kutsuja tehdään käytännössä vain NVRAM-moduulin käynnistyessä.

Kirjoitusfunktio write() sen sijaan ottaa sisäänsä määrittelemättömän määrän dataa tavumuodossa ja lähettää sen toimilaitekortin ajurille. Toimilaitekortin ajuri vastaanottaa viestin ja kirjoittaa tiedon fyysiselle NVRAM-muistille. Funktio palauttaa kutsujalle tiedon kirjoituksen onnistumisesta.

Kolmantena funktiona mainittu readPeripheralId() lukee toimilaitekortin ajurilta toimilaitteen tunnistenumeron. Lukufunktio palauttaa kutsujalleen toimilaitteen tunnistenumeron. Tunnistenumeroiden avulla pystytään varmistamaan peliautomaatin identiteetti.

4.3.4 Tehtävä (Task)

NVRAM-moduulin palvelemat palvelut päätettiin toteuttaa tehtävämäisesti. Jokainen palvelu muodostaa oman tehtävänsä ja jokaisella tehtävällä on oma toiminnallinen logiikkansa. Tehtäville asetettiin Task-niminen rajapinta, jonka julkisina funktioina ovat *execute()* ja *reject()*. Funktio execute() suorittaa konkreettisen tehtäväluokan toteutuksen. Funktiolla reject() sen sijaan on oletustoteutuksena tehdä virheilmoitus odottamattomasta viestistä. Joihinkin viesteihin kuitenkin odotetaan vastausta, joten näitä viestejä varten tehdään myös lisätoteutus, joka ylikirjoittaa oletuskäyttäytymisen. Kuormittava funktio lähettää kutsujalle viestissä tiedon epäonnistumisesta.

Aikaisemmin mainitut Muistimanageri, Järjestelmä-rajapinta sekä Ajuri-rajapinta ovat tehtävien tarvitsemia palveluita. Näiden palveluiden avulla pystytään luomaan useita toiminnaltaan poikkeavia tehtäviä. Tehtävät pääsevät palveluihin käsiksi Service-rajapinnan kautta. Kaikissa tehtävissä tarvitaan Muistimanageria, jonka avulla tietoa joko luetaan tai kirjoitetaan NVRAM-moduulin omaan muistikopioon. Kirjoituksia tehdessä tarvitaan Ajuri-rajapintaa, koska kirjoitukset tulee tehdä muistikopion lisäksi myös fyysisille pysyväismuistipiireille. Asiakasmoduuleiden pyyntöihin tulee usein myös vastata, minkä takia tehtävät tarvitsevat pääsyn myös Järjestelmä-rajapinnan funktioihin.

Kaikenkaikkiaan erilaisia tehtäviä on suunniteltu yhdeksän kappaletta, joista suurin osa on nykyisessä NVRAM-moduulissa määriteltyjä toimintoja. Uuden moduulin olisi tarkoitus tukea kolmeatoista PSI-viestiä yhteensä, joista neljä ovat tämän projektin mukana tulleita ominaisuuksia. Viestirajapinta haluttiin pitää samana, jolloin uudella moduulilla voidaan korvata nykyinen tekemättä muutoksia asiakasmoduuleihin. Tämä näkyy siten, että tehtävien ja tuettujen PSI-viestien määrä ei ole täsmälleen sama ja joissakin tilanteissa useampi eri PSI-viesti tulkataan samaksi tehtäväksi. Ne tilanteet, joissa samalla PSI-viestillä saadaan useampi eri tehtävä, johtuu viestien sisällöstä. Joissain viestityypeissä on mukana kenttä, joka kertoo viestin funktion. Tällaisia viestejä ovat esimerkiksi nykyisen NVRAM-moduulin luku- ja kirjoitusviestit. Taulukossa 5 on kartoitettu, mitä mikäkin PSI-viesti tarkoittaa tehtävämallisessa toteutuksessa.

Taulukko 5: Uuden moduulin toiminnallisuudet.

PSI-viesti	**Tehtävä**	**Selitys**
MSG_MODULE_INIT	Init	Suorittaa NVRAM-moduulin alustuksen.
MSG_MODULE_ DO_SELFTEST	SelfTest	Suorittaa NVRAM-moduuliin liittyvät kuntotarkistukset.
MSG_NVRAM	Write	Suorittaa asiakasmoduuleiden kirjoituspyynnöt.
MSG_NVRAM_STAT	Write	Kirjoittaa tilastopäivityksen (yhteensopivuus taaksepäin).
MSG_NVRAM	Read	Suorittaa asiakasmoduuleiden lukupyynnöt.
MSG_NVRAM_STAT	Read	Suorittaa tilastosta lukemisen (yhteensopivuus taaksepäin).
MSG_NVRAM_FORMAT	Format	Suorittaa pysyväismuistin formatoinnin.
MSG_NVRAM_FORMAT	FormatBlock	Suorittaa tilastojen formatoinnin (yhteensopivuus taaksepäin).
MSG_NVRAM2_ FORMAT_BLOCK	FormatBlock	Formatoi asiakasmoduulin datalohkon (uusi toiminnallisuus).
MSG_GIVE_ AUTOMAT_NUMBER	ReadAutomatNumber	Palauttaa peliautomaatin numeron (yhteensopivuus taaksepäin).
MSG_NVRAM2_INFO	NvramInfo	Palauttaa pysyväismuistin otsikkotiedot (uusi toiminnallisuus).
MSG_NVRAM2_ SET_BLOCK_VERSION	ResetBlockVersion	Asettaa asiakasmoduulin lohkon versionumeron (uusi toiminnallisuus).
MSG_NVRAM2_ GET_BLOCK_INFO	BlockInfo	Palauttaa asiakasmoduulin lohkon otsikkotiedot (uusi toiminnallisuus).

Viestit MSG_MODULE_INIT ja MSG_MODULE_DO_SELFTEST kuuluvat PSI-sovellusten kättelyrutiiniin. Kättelyrutiini on käsitelty luvussa 2.2. Luku- sekä kirjoituspyynnöt tehdään molemmat samalla MSG_NVRAM-viestillä. Kyseinen viesti sisältää kentän, jonka avulla NVRAM-moduuli pystyy tunnistamaan kummasta pyynnöstä on kyse. Lukupyyntöjä tehtäessä viesti kertoo, kenen asiakasmoduulin lohkosta halutaan tietoa lukea ja kuinka paljon. Kirjoituspyynnöissä samaiset kentät kertovat, kenen asiakasmoduulin lohkoon kirjoitetaan ja kuinka paljon. Lisäksi viestin sisältämä datakenttä sisältää kirjoitettavan tiedon. Vastaavanlainen toteutus

on tehty myös tilastoalueeseen kohdistuville luvuille ja kirjoituksille, jotka tehdään siis MSG_NVRAM_STAT-nimisellä viestillä. Uudessa NVRAM-moduulissa tilastoja käsitellään täsmälleen samalla tavalla kuin muitakin asiakaslohkoja. Ainoana erona on, että NVRAM-moduulille on erikseen ilmoitettu parametreissa, mikä on tilastolohkon tunnistenumero, jolloin NVRAM-moduuli pystyy tulkkaamaan MSG_NVRAM_STAT-viestit suoraan MSG_NVRAM-viesteinä.

Viestillä MSG_NVRAM_FORMAT on myös kaksi eri merkitystä: koko pysyväismuistin formatointi sekä pelkän tilastoalueen formatointi. Uudessa moduulissa nämä kolme toimintoa on irrotettu kahdeksi eri tehtäväksi, sillä niiden sisäiset toiminnallisuudet poikkeavat toisistaan kriittisesti.
Pysyväismuistin formatoinnista huolehtii Format-tehtävä, joka huolehtii muistimallin oikeaoppisesta nollaamisesta sekä tarvittavien esitietojen täyttämisestä. Tilastoalueen formatointi tehdään FormatBlock-tehtävällä, jota voidaan kutsua myös erikseen MSG_NVRAM2_FORMAT_BLOCK-viestillä. Erona viesteissä on se, että MSG_NVRAM_FORMAT kohdistaa lohkon formatoinnin aina tilastolohkolle, kun taas MSG_NVRAM2_FORMAT_BLOCK-viestille annetaan parametrina tieto siitä, mikä lohko halutaan formatoida.

NVRAM-moduulilta voidaan kysyä sen tunnistetietoja MSG_NVRAM2_INFO-viestillä. Viesti palauttaa lähettäjälle NVRAM-moduulin versionumeron, peliautomaatin numeron, pysyväismuistin kokonaismäärän sekä jäljellä olevan muistin määrän. Käyttöä tehtävälle ei vielä ole, mutta tästä toiminnallisuudesta voi olla tulevaisuudessa ainakin kehittäjätiimeille jotain hyötyä.

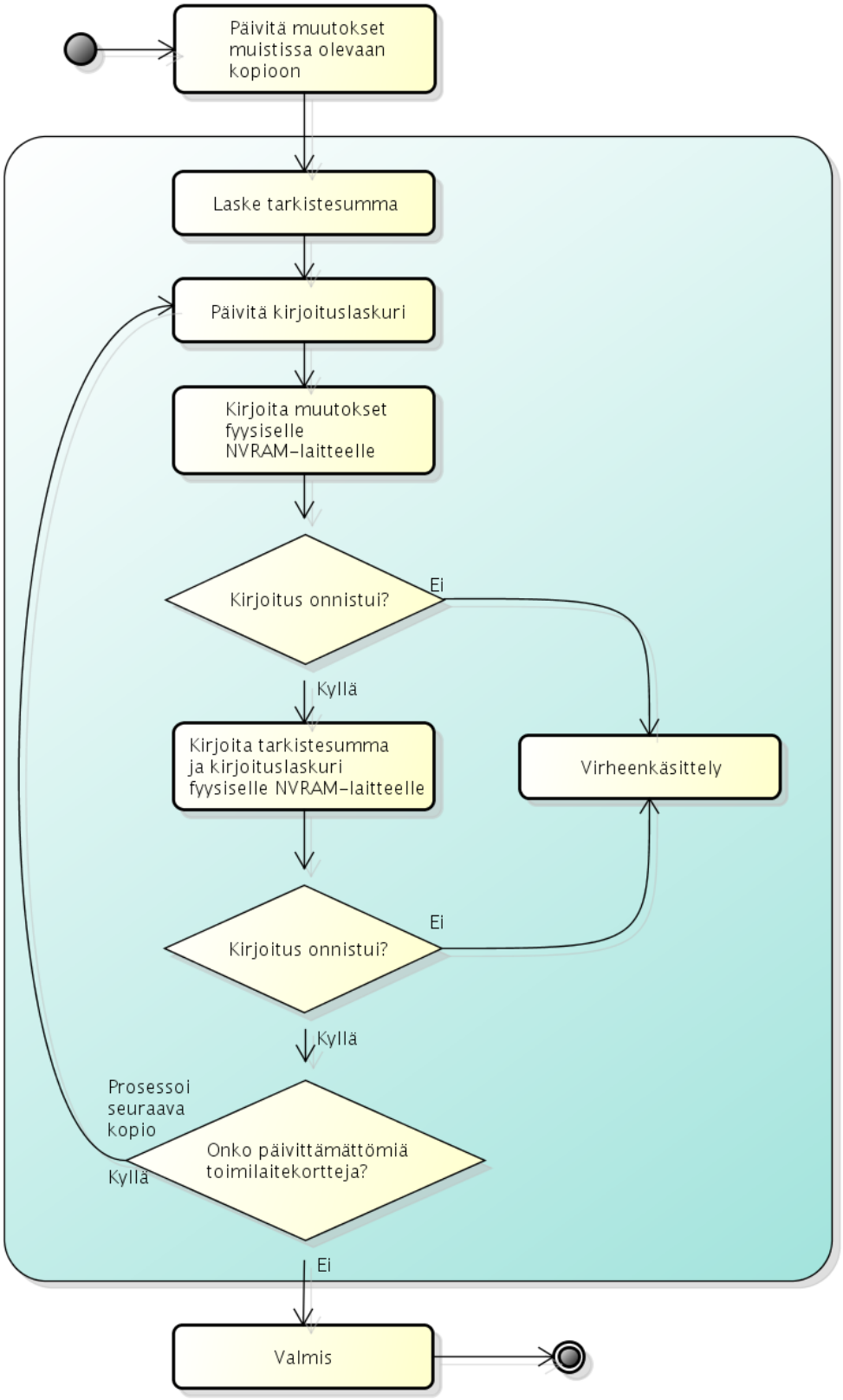
Asiakasmoduulien sisäisiä datamalleja ajatellen lisättiin NVRAM-moduulin palveluksi asiakasmoduulin lohkon version numerointipalvelu. Asiakasmoduuli voi kysyä NVRAM-moduulilta oman lohkonsa versiota ja näin ollen suorittaa luetulle datalle mahdolliset konvertoinnit, ennen kuin data sijoitetaan asiakasmoduulien omiin muistikopioihin. Asiakasmoduuli voi asettaa lohkonsa versionumeron MSG_NVRAM2_SET_BLOCK_VERSION-viestillä ja MSG_NVRAM2_GET_BLOCK_INFO-viestillä asiakasmoduuli voi kysyä NVRAM-moduulilta lohkonsa versionumeron sekä käytettävissä olevan tilan määrän.

5 VIKASIETOISUUDEN KEHITTÄMINEN

Tässä luvussa keskitytään NVRAM-moduulin kolmen ongelmakohdan kehittämiseen, joista kaksi on tunnistettu heikkoudeksi luvussa 3.3. Nämä kaksi heikkoutta olivat fyysiselle pysyväismuistille tehtävät kirjoitukset sekä pysyväismuistin formatointiin liittyvät riskitekijät. Moduulin alustuksen aikana tehtävät operaatiot aiheuttavat sen, että moduulin alustus ollaan tunnistettu NVRAM-moduulin monimutkaisimmaksi ominaisuudeksi. Tästä syystä se on haluttu nostaa kolmanneksi ongelmakohdaksi.

5.1 Kirjoittaminen

Nykyisessä moduulissa kirjoitukset ovat osoittautuneet jokseenkin epäluotettaviksi ja ne aiheuttavat silloin tällöin vakavia muistikorruptioita, joista ei pystytä palautumaan. Epäluotettavuus johtuu siitä, että kaikki toimilaitteelle lähtevät kirjoitukset tehdään rinnastetusti. Ajureilta ei odoteta paluuviestinä tietoa, ovatko kirjoitukset onnistuneet. Silti asiakasmoduulille kerrotaan, että kirjoitus on onnistunut. Jotta kirjoitusten luotettavuudesta voitaisiin olla varmoja, on kirjoitukset tehtävä atomisesti. Käytännössä tämä tarkoittaa sitä, että kirjoitukset tehdään sekventiaalisesti kullekin toimilaitekortille ja odotetaan aina, että toimilaitekortin ajuri lähettää vastauksen kirjoituksen onnistumisesta, mikä on myös tarkistettava. Asiakasmoduulille voidaan lähettää vastaus onnistuneesta kirjoituksesta vasta, kun molemmille toimilaitekorteille on kaikki kirjoitukset tehty onnistuneesti. Päällekkäisten kirjoitusten tilalle kehiteltiinkin kuvan 7 mukainen sekventiaalinen kirjoituslogiikka.



Kuva 7: Toimilaitekortille tehtävän kirjoituksen vuokaavio.

Kuvassa 7 esitelty kirjoituslogiikka on jaettu kolmeen osaan. Kirjoituspyynnön saapuessa NVRAM-moduuli päivittää oman sisäisen muistikopionsa kirjoituspyynnön mukana

tulleella datalla, jonka jälkeen koko muistikopion yli lasketaan uusi MD4-tarkistesumma. Muistikopion päivityksen jälkeen voidaan aloittaa ensimmäiselle toimilaitekortille kirjoittaminen. Jotta virheisiin voitaisiin reagoida välittömästi, tehdään toimilaitekortille kirjoittaminen kahdessa osassa. Ensimmäisessä osassa kirjoitetaan fyysiselle NVRAM-muistille muistikopioon tehdyt muutokset, jonka jälkeen varmistetaan, onko kirjoitus onnistunut. Mikäli kirjoitus onnistuu, jatketaan kirjoittamalla tarkistesumma ja kirjoituslaskuri niille varatulle alueelle. Kun molemmat vaiheet on suoritettu ensimmäiselle toimilaitekortille onnistuneesti, siirrytään käsittelemään toista toimilaitekorttia. Jos molemmille toimilaitekorteille onnistutaan kirjoittamaan onnistuneesti, ilmoitetaan kirjoituspyynnön tehneelle asiakasmoduulille kirjoituksen onnistumisesta.

Kirjoitus voi epäonnistua, mikäli peliautomaatti menettää yhteyden kirjoitettavaan toimilaitekorttiin. Peliautomaatti yrittää muodostaa uutta yhteyttä kyseiselle toimilaitekortille. Mikäli yhteyden muodostaminen epäonnistuu, ilmoitetaan asiakasmoduulille, että kirjoitus on epäonnistunut ja estetään peliautomaatilla pelaaminen. Kirjoitus voi myös epäonnistua, mikäli asiakasmoduulin tekemä kirjoituspyyntö kohdistuu datalohkon rajojen ulkopuolelle tai olemattomaan datalohkoon.

Uusi kirjoituslogiikka on suorituskyvyltään hitaampi kuin nykyisen NVRAM-moduulin kirjoituslogiikka, mutta vaihdon mukana tullut luotettavuus on sen arvoista. Sekventiaalisella kirjoituslogiikalla voidaan taata, että vaikka peliautomaatti sulkeutuisikin kesken kirjoituksen, yhdellä toimilaitekortilla on aina ehjä ja korruptoimaton kopio. Toimilaitekortin hajoaminen voi edelleen kuitenkin johtaa tilanteeseen, josta on mahdotonta palautua. Tavallisessa käytössä NVRAM-moduuli voi kuitenkin käynnistyä vain taulukon 6 mukaisissa NVRAM-tiloissa, jolloin pystytään aina valitsemaan ehjä ja korruptoimaton muistikopio. Tässä vaiheessa erikoistapausten käsittely ei ole oleellista. Erikoistapaukset esitellään taulukon 7 yhteydessä.

Taulukko 6: NVRAM-kopioiden mahdolliset eheystilat normaaleissa käynnistyksissä.

Tila	**Kolikkolukkokortin tila**	**Hopperikortin tila**	**Valinta ja syy**
NS1	Tarkistesumma OK, kirjoituslaskuri 1	Tarkistesumma OK, kirjoituslaskuri 2	Hopperi, molemmat kopiot ehjiä, hopperikortissa on uudempi data.
NS2	Tarkistesumma OK, kirjoituslaskuri 3	Tarkistesumma OK, kirjoituslaskuri 2	Kolikkolukko, molemmat kopiot ehjiä, Kolikkolukon kortissa on uudempi data.
NS3	Tarkistesumma OK	Tarkistesumma EI OK	Kolikkolukko, peliautomaatti sulkeutunut kesken kirjoituksen hopperikortille.
NS4	Tarkistesumma EI OK	Tarkistesumma OK	Hopperi, peliautomaatti sulkeutunut kesken kirjoituksen kolikkolukon kortille.

Käytännössä peliautomaatin odotetaan aina käynnistyvän taulukon 6 tilassa NS1. Molempien tarkistesummien ollessa OK valitaan käytettävä kopio kirjoituslaskurin perusteella. Kirjoituslaskurien arvot poikkeavat kopioiden välillä aina, sillä kummallekin toimilaitekortille kirjoitetaan aina uusi numero. Toisella toimilaitekortilla on aina parilliset kirjoituslaskurin arvot, kun taas toisella ovat parittomat. Kirjoitukset tehdään aina samassa järjestyksessä, joten täysin onnistuneen kirjoituspyynnön jälkeen hopperikortissa on aina suurempi kirjoituslaskurin arvo.

On mahdollista, vaikkakin hyvin epätodennäköistä, että peliautomaatti sulkeutuu juuri sillä hetkellä, kun kirjoitus kolikkolukon korttiin on päättynyt ja juuri ennen kuin hopperikortille aloitetaan kirjoittamaan. Tällöin peliautomaatti käynnistyy tilaan NS2, eli molemmilla toimilaitekorteilla on täysin ehjä kopio, mutta kirjoituslaskurin arvo on kolikkolukon kortilla suurempi kuin hopperikortilla, eli kolikkolukkokortti sisältää uudemman datan.

Tiloihin NS3 ja NS4 joudutaan, kun virran menetys tapahtuu kirjoituksen ollessa kesken jommalle kummalle toimilaitekorteista. NS3-tila on siinä mielessä edullisempi, että kolikkolukon kortilla sijaitsevaan kopioon ollaan jo onnistuneesti kirjoitettu, jolloin kolikkolukon kortin avulla voidaan hopperikortin kopio korjata ilman, että yhtään kirjoitusta menetetään. NS4-tilaan joutuessa viimeisin kirjoitus menetetään väistämättä, sillä kirjoitus on keskeytynyt kolikkolukkokorttia päivitettäessä.

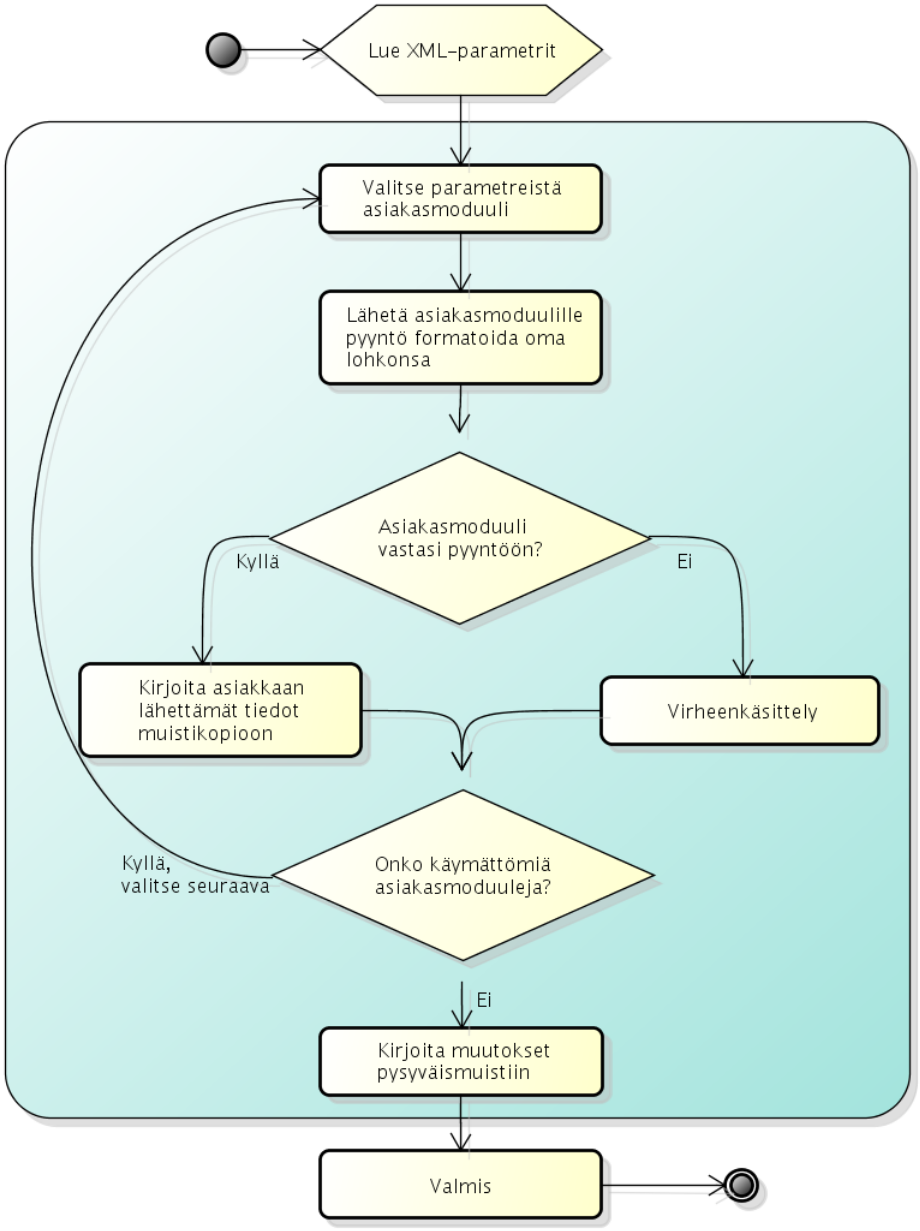
5.2 Pysyväismuistin formatointi

Pysyväismuistin formatointi on harvinainen toiminto, mutta nykyisellään sen toteutus toimii tuurilla. Pysyväismuisti joudutaan formatoimaan silloin, kun uusi peliautomaatti on valmis siirtymään tehtaalta kentälle. Pysyväismuisti voidaan joutua formatoimaan myös peliautomaatin normaalin elinkaaren aikana, mikäli pysyväismuisti korruptoituu syystä tai toisesta korjaamattomaan tilaan. Nykyisen NVRAM-moduulin kanssa totaalisia muistikorruptoitumisia havaitaan noin sata kappaletta kuukaudessa. Korruptiot menevät pääosin kirjoituslogiikan virheellisen toiminnan piikkiin, mutta formatoinnin toimivuus pitäisi silti varmistaa.

Nykyisellään peliautomaatin pysyväismuistin formatointioperaatio nollaa aluksi koko pysyväismuistin, jonka jälkeen NVRAM-moduuli kirjoittaa pysyväismuistiin omat tunnistetietonsa sekä luo kuvan 5 mukaisen datamallin. Tämän jälkeen *Control*-niminen PSI-sovellus lähettää kuulutetun PSI-viestin, millä pyydetään kaikkia PSI-sovelluksia lähettämään omat oletusarvonsa NVRAM-moduulille. Tämän pyynnön seurauksena osa asiakasmoduuleista lähettää NVRAM-moduulille tavallisen kirjoituspyynnön, jonka mukana oletusarvot annetaan NVRAM-moduulille. Oletusarvot ovat moduulikohtaisia. Mikäli asiakasmoduuli ei pidä oletusarvoja tärkeänä, jättää NVRAM-moduuli kyseisen asiakasmoduulin lohkon täyteen nollaa. Nykyisen formatointiprosessin ongelma on se, ettei sitä pystytä valvomaan eikä kontrolloimaan ollenkaan. Kirjoitukset tehdään, kunhan ehditään ja jos ehditään. Kirjoitukset joudutaan tekemään jokseenkin kiireellä,

sillä Control-moduuli käskee peliautomaattia käynnistämään itsensä uudelleen lyhyen odottamisen jälkeen. Näin ollen ei voida koskaan tietää, ehdittiinkö kaikkia kirjoituksia tehdä ajallaan.

Heikkouksien korjaamista varten kehitettiin kuvan 8 kaltainen toimintasekvenssi. Toiminnan toteuttaminen vaatii muutamia lisäyksiä niiden asiakasmoduulien lähdekoodeihin, jotka nykyisellään haluavat itse asettaa omat oletusarvonsa. Näiden asiakasmoduulien tunnistenumerot täytyi myös lisätä XML-parametreihin, jotta NVRAM-moduuli tietää, keneltä oletustietoja täytyy kysyä. Tästä syystä pysyväismuistin formatoinnille on määritelty esiehdoksi, että XML-parametrit ovat luettu. Käytännössä tämä esiehto on aina täytetty, sillä XML-parametrit luetaan ja tulkataan heti ohjelman käynnistyessä.



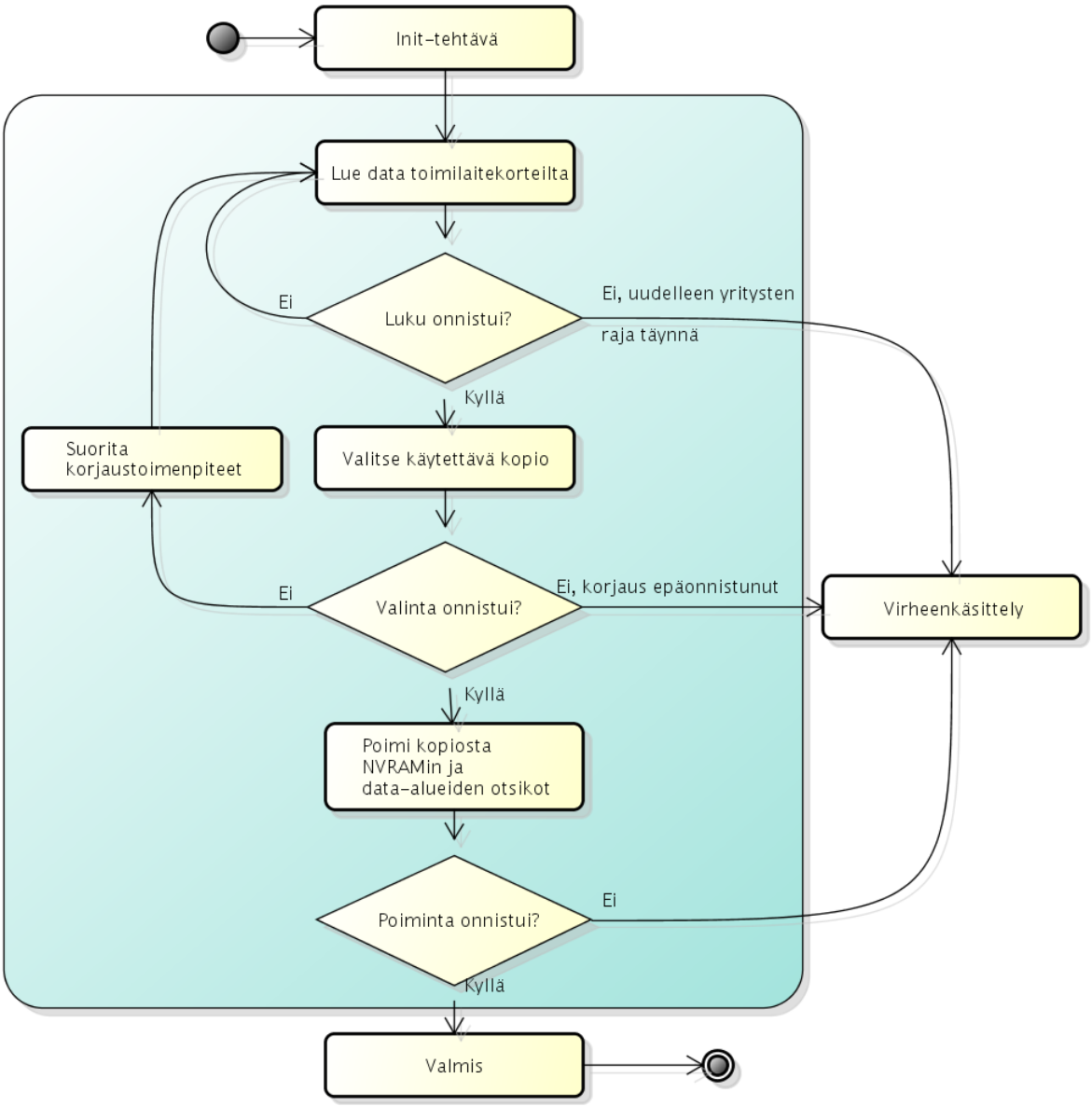
Kuva 8: Pysyväismuistin formatoinnin vuokaavio.

NVRAM-moduuli valitsee parametrien perusteella ensimmäisen asiakasmoduulin, jonka oletustiedot täytyy tiedustella. NVRAM-moduulille lähettää kohdistetun PSI-viestin kyseiselle asiakasmoduulille. Mikäli asiakasmoduuli vastaa viestiin tietyn ajan kuluessa, kirjoitetaan asiakasmoduulin lähettämät muutokset NVRAM-moduulin omaan muistikopioon. Muistilohko jätetään tyhjäksi, jos asiakasmoduuli ei vastaa viestiin määritellyn aikarajan sisällä. Tämän lisäksi asiasta lähetetään vikailmoitus. Muistipäivityksen tai virheenkäsittelyn jälkeen tarkistetaan, onko jäljellä muita asiakasmoduuleja. Tätä silmukkaa suoritetaan kunnes kaikki parametreissa listatut asiakasmoduulit ovat käsitelty. Lopuksi koko NVRAM-muistin sisältö kirjoitetaan vielä molemmalle fyysisille pysyväismuistipiirille.

Formatoinnin jälkeen peliautomaatti käynnistetään uudestaan, jolloin kaikki asiakasmoduulit saavat puhtaan kopion omista lohkoistaan. NVRAM-moduuli pystyisi käytännössä suorittamaan formatoinnin ilman peliautomaatin uudelleen käynnistämistä, mutta asiakasmoduulit eivät nykyisellään osaa käsitellä itsensä uudelleenalustamista ilman uudelleen käynnistämistä.

5.3 Moduulin alustus

NVRAM-moduulin alustusvaihe on sen monimutkaisin toiminnallisuus. Alustusvaiheen aikana suoritetaan pysyväismuistin eheyteen liittyvät toimenpiteet, suoritetaan mahdolliset korjaukset sekä datamallin päivitykset. Alustuksen jälkeen NVRAM-moduulin on oltava käyttökunnossa, mikäli mitään kriittisiä virheitä ei löydetä. Uuden NVRAM-moduulin alustus tehdään kuvan 9 mukaisesti kolmessa vaiheessa.



Kuva 9: NVRAM-moduulin alustuksen vuokaavio.

Ensimmäisenä NVRAM-moduuli lukee toimilaitekorteilta kokonaisuudessaan molempien NVRAM-piirien sisällöt omiin muistikopioihinsa. Tämän lisäksi toimilaitekorttien ajureilta pyydetään kyseisten toimilaitekorttien tunnistetiedot toisessa vaiheessa suoritettavaa identiteettivarmistusta varten. Sisältöjen lukeminen voi epäonnistua, mikäli toisessa toimilaitekortissa havaitaan vika. Vikatilanteessa NVRAM-moduulin alustus epäonnistuu eikä seuraavia vaiheita voida suorittaa.

Toisessa vaiheessa suoritetaan kaikki mahdollinen luettuun dataan liittyvät tarkastelut, jotka voivat johtaa lisätoimenpiteisiin. Mahdolliset lisätoimenpiteet ovat kopion korjaus ja datamallin päivittäminen. Luetut kopiot analysoidaan ja niille määritellään taulukon 7 mukaisesti eheystila.

Taulukko 7: Kopion mahdolliset eheystilat peliautomaatin käynnistyessä.

Tila	**Kuvaus**
S1	Tarkistesumma ok + 3 toimilaitetunnistetta ok (ehjä kopio)
S2	Tarkistesumma ok + 2 toimilaitetunnistetta ok (yksi toimilaitekortti vaihdettu)
S3	Vanha datamalli + 3 toimilaitetunnistetta ok (NVRAM-moduuli päivitetty)
S4	Vanha datamalli + 2 toimilaitetunnistetta ok (yksi toimilaitekortti vaihdettu)
S5	Tarkistesumma ok + 1 toimilaitetunnistetta ok (tämä toimilaitekortti vaihdettu)
S6	Kopio rikki

Kopiot laitetaan paremmuusjärjestykseen eheystilojen perusteella; S1 on parempi kuin S2, S2 on parempi kuin S3 ja niin edelleen. Mikäli molemmille tiloille saadaan eheystilaksi S1 ja kopiot ovat keskenään identtisiä, niin siirrytään suoraan kolmanteen vaiheeseen. Kopiota yritetään korjata, mikäli paremmaksi tunnistettu kopio on tilassa S1 tai S2 ja heikompi kopio tilassa S2-S6. Kopion korjausoperaatiossa heikompi kopio korvataan paremmalla kopiolla. Mikäli parempi kopio ei ole tilassa S1, päivitetään myös toimilaitetunnisteet molempiin kopioihin. Korjausoperaation jälkeen siirrytään takaisin ensimmäiseen vaiheeseen, jolloin vaihe kaksi joudutaan käymään vielä uudelleen. Tarkoituksena on sulkea pois korjauksen epäonnistumisen mahdollisuus. Huomioitavaa on, että kopion ei tarvitse välttämättä olla edes rikki, vaikka korjausta tarvitaan. Korjausta tarvitaan myös silloin, kun kopiot ovat ehjiä, mutteivat keskenään identtisiä. Parametritiedostoon lisätyt uudet asiakaslohkot huomataan myös tässä vaiheessa, ja niiden lisääminen aiheuttaa korjausoperaation molempiin kopioihin.

Jos parempi kopio on joko tilassa S3 tai S4, niin suoritettava operaatio on nimeltään konversio, eli vanhan datamallin päivittäminen uuteen muotoon. Vanhassa datamallissa ei ole tarvittavia tarkistetietoja, joiden perusteella pystyttäisiin luotettavasti sanomaan mitään kopion eheydestä. Tästä syystä uudenmallinen datamalli menee aina vanhan edelle jopa silloin, kun vanhan mallisesta kopiosta löydetään kolme oikeaa toimilaitetunnistetta ja uuden mallisesta vain kaksi. Käytännössä kyseiseen tilanteeseen ei voida mitenkään joutua. Kun molemmista kopioista löydetään vanhan datamallin mukainen kopio, niin oletuksena suositaan aina hopperikortilla sijaitsevaa kopiota. Kyseisellä kopiolla sijaitsee vanhan datamallin mukaan tilastoalue eikä sen tietoja haluta datamallin päivityksen yhteydessä menettää. Paremman kopion sisältämä muistimalli päivitetään uudemmaksi ja kirjoitetaan ensin heikomman kopion päälle. Mikäli peliautomaatti sammuu kesken ensimmäiselle heikommalle kortille kirjoittamista, huomataan seuraavassa käynnistyksessä, että toinen pysyväismuisti on rikki ja toinen on vanhan datamallin mukainen, joten konvertointia yritetään uudelleen. Vastaavasti kun ensimmäinen kirjoitus onnistuu, mutta toisen kirjoituksen aikana sattuu virran menetys,

niin toisessa pysyväismuistissa on jo täydellinen kopio, joten rikkinäinen korvataan sillä. Konvertoinnin jälkeen palataan takaisin ensimmäiseen vaiheeseen korjausoperaation tavoin.

Eheystila S5 voitaisiin hyvin myös sisällyttää tilaan S6, sillä paremman kopion ollessa tilassa S5, tehdään täsmälleen sama toiminnallisuus kuin tilassa S6. Molemmissa tapauksissa peliautomaatti ohjataan sulkutilaan, josta poistuminen vaatii pysyväismuistin formatoinnin. Erona kuitenkin on se, että periaatteessa S5-tilassa olevassa kopiossa on täysin ehjä sisältö, mutta kyseinen data on selkeästi tullut jostain toisesta peliautomaatista. Tämän erotuksen avulla saadaan tarkempi virheilmoitus kyseisessä vikatilanteessa. Sulkutilaan siirtymisen jälkeen ohitetaan kolmas vaihe ja alustuksen ilmoitetaan olevan valmis.

Kolmannessa vaiheessa poimitaan käytettävästä kopiosta NVRAM:n otsikko- ja tunnistetiedot sekä asiakasmoduulien datalohkot NVRAM-moduulin omiin sisäisiin tietorakenteisiin. Sisäiset tietorakenteet on esitelty luvussa 4.3.1. Poiminnan jälkeen moduulin alustus on vihdoin valmis ja NVRAM-moduuli on valmis palvelemaan asiakasmoduulien pyyntöjä.

5.4 Testaus

Testaus on tärkeä osa nykyaikaisissa ohjelmistokehitysprosesseissa. Kohtuullisen vähäisellä vaivalla saadaan kehitettävästä ohjelmistosta luotettavampi ja helpommin ylläpidettävä kokonaisuus [16, s. 6]. Huolellista testausta on kuitenkin tehtävä alusta asti, jotta testeistä saadaan täysi hyöty irti [10, s. 150-160]. Tässä luvussa käydään läpi, miten testaus otettiin mukaan uuden NVRAM-moduulin kehitysprosessiin ja mitä menetelmiä testauksessa käytettiin.

Uuden moduulin suunnitelmassa panostettiin testattavuuteen. NVRAM-moduulin komponenttien välillä haluttiin pitää mahdollisimman löyhä riippuvuussuhde, jolloin komponentteja pystyttäisiin testaamaan yksittäin. Riippuvuuksilta ei voida täysin välttyä, mutta rajapintojen avulla riippuvuudet kohdistuvat pelkästään funktioiden olemassa oloon, muttei toteuttajiin. Testeissä voidaan käyttää jäljitelmäolioita (engl. Mock Object), joiden avulla riippuvuudet selvitetään hallitusti ilman ei-toivottuja sivuvaikutuksia [11, s. 47]. Testaustavoitteeksi asetettiin, että kaikki julkiset funktiot tulee testata yksikkötestein.

Testaustavoitteen vuoksi olikin syytä ottaa heti alussa käyttöön jokin yksikkötestejä varten suunniteltu testauskehys. Alkuun ei kuitenkaan osattu päättää mikä testauskehys soveltuisi uuden NVRAM-moduulin kehitysprosessiin parhaiten. Kehitystiimin kanssa asetettiin tavoitteeksi löytää testauskehys, jolla yksikkötestien tekemisestä ei tulisi liian suurta taakkaa, mikä hidastaisi kehitystyötä merkittävästi. C++-kielelle tarkoitetuissa testauskehyksissä joutuu usein kirjoittamaan pitkiäkin alustuksia yksikkötesteille ennen kuin oikeata testauskoodia pääsee edes kirjoittamaan eikä näitä testialustuksia pysty

kovin kätevästi uusiokäyttämään seuraavissa testeissä. Tähän kynnykseen UnitTest++ [8] ei kuitenkaan kompastunut, sillä UnitTest++ osottautui nopeasti hyvin yksinkertaiseksi ja tehokkaaksi testauskehykseksi. Automaattisen testausajurin kirjoittaminen sujuu hetkessä ja kirjaston määrittelemien esikääntäjämakrojen avulla yksikkötestien alustukseen ei tarvitse nähdä ylimääräistä vaivaa. C++:n perintäominaisuuksia hyödyntämällä pääsee UnitTest++:n avulla testaamaan yksittäisiä funktioita hyvin vaivattomasti. Yksikkötestifunktiolle tarvitsee ilmoittaa vain testattavan luokan nimi, jolloin testauskehys mahdollistaa kyseisen luokan jäsenmuuttujien ja funktioiden kutsumisen, vaikka normaalisti kyseiset jäsenmuuttujat ja funktiot olisivatkin näkyvyysalueen ulkopuolella.

Hyvien yksikkötestien tulisi kattaa hyvä osa sallituista syötteistä, mutta niiden tulisi huomioida myös virheelliset syötteet. Testien oikeellisuuden varmistamiseksi otettiin käyttöön myös testikattavuutta mittaava gcov-ohjelmisto [6]. Gcov on GCC-kääntäjän [5] (GNU Compiler Collection) lisäosa, joka lisää käännöksen aikana ohjelman sekaan laskureita. Ohjelman suorittamisen jälkeen laskureiden arvoista muodostetaan suoritusraportti, josta nähdään, mitkä ohjelman lähdekoodin rivit suoritettiin ajon aikana. Suoritusraporttia analysoimalla saadaan tulokseksi tiedostokohtainen testikattavuus. Lisäksi ohjelman muistinkäyttöä valvottiin Valgrindin [9] avulla. Valgrindillä tarkistettiin, ettei uusi NVRAM-moduuli vuoda muistia, eikä käsittele alustamattomia muuttujia.

Kehitystyön aikana yksikkötestejä suoritettiin aina uusien muutosten jälkeen käännöspalvelimella. Sen tehtävänä oli kääntää projektin lähdekoodeja sekä suorittaa yksikkötestit. Käännöspalvelimen avulla saatiin aina välitön palaute testien toiminnasta sekä testikattavuuksista. Testejä suoritettiin myös omilla kehityskoneilla, mutta käännöspalvelin varmistaa, että testit menevät läpi myös puhtaassa ympäristössä. Kehityksen tehostamiseksi testikattavuusraportti oli koko ajan näkyvillä omalla ruudullaan. Menetelmä oli tehokas, sillä se motivoi kehitystiimiä testaamaan kirjoitetun lähdekoodin ennen sen julkaisemista. Menetelmä oli itseasiassa niin tehokas, että uuden NVRAM-moduulin testikattavuus oli lähes 95 % siihen asti, kunnes peliautomaatin tuomat riippuvuudet lisättiin projektiin mukaan. Riippuvuudet tuotiin vasta siinä vaiheessa, kun muu toiminnallisuus oli jo valmiina. Peliautomaattiympäristöön siirryttäessä testikattavuus tippui lopulta noin 400 testillä 85 prosenttiin.

Uutta NVRAM-moduulia testattiin komentorivityökalun avulla ennen peliautomaattiympäristöön siirtymistä. Komentorivityökalulla lähetettiin PSI-viestejä NVRAM-moduulille, jolloin moduulin toiminnallisuutta pystyttiin testamaan yksikkötestejä ylemmältä tasolta. Yksikkötestien avulla pystyttiin olemaan varmoja moduulin funktioiden toimivuudesta, mutta vasta moduulitestauksella saatiin varmuus, että moduuli toimii kokonaisuutena oikein. Moduulitesteillä käytiin läpi kaikki moduulin suorittamat tehtävät (esitelty luvussa 4.3.4). Tehtäviä luotiin useilla eri syötteillä, joista osa myös sisälsi virheitä. Näissä testeissä käytettiin fyysisten toimilaitekorttien sijasta tiedostoja, mikä mahdollisti tulosten automaattisen analysoinnin yksinkertaisella Bash-skriptillä

(Bourne again shell). Tiedostojen avulla pystyttiin hallinnoimaan täysin, minkälaisessa tilassa uuden NVRAM-moduulin tulisi käynnistyä, jolloin varsinkin alustuslogiikan virhetilanteiden testaaminen helpottui huomattavasti.

Jäljitelmäoliot mahdollistivat kattavien moduulitestien tekemisen ilman, että peliautomaattin tarvitsi siirtyä ennen kuin moduuli oli lähes valmis. Tehokkuus näkyi erityisesti siinä, että kun vihdoin peliautomaattiympäristöön siirryttiin, uusi NVRAM-moduuli toimi lähes suoraan. Joitakin pieniä ohjelmointivirheitä havaittiin liittyen PSI-viestien käsittelyyn, mutta virheiden määrä oli vähäistä. Tämän jälkeen uutta NVRAM-moduulia alettiin testata peliautomaatilla. Muun muassa alustuslogiikkaa rasitustestattiin tekemällä testisovellus, joka lähetti NVRAM-moduulille satoja kirjoituspyyntöjä, joiden aikana kyseinen testisovellus aiheutti satunnaisen uudelleen käynnistyksen. Rasitustestiä ajettiin yhden viikonlopun ajan. Yhtään muistikorruptiota ei rasitustestien aikana havaittu. Tuloksia analysoimalla havaittiin, että yksittäinen kopio jäi peliautomaatin sulkeutumisen takia virheelliseen tilaan, mutta NVRAM-moduuli pystyi aina toisen toimilaitekortin kopion avulla korjaamaan rikkinäisen kopion.

Raha-automaattiyhdistyksellä on useita laitteistoltaan erilaisia peliautomaatteja. Osa näistä laitteistoista on suorituskyvyltään heikkoja, joten uutta moduulia testattiin myös heikoimmilla laitteistoilla. Testaus osoitti, että uusi NVRAM-moduuli ei ole suorituskyvyltään yhtään hitaampi kuin nykyinen.

Raha-automaattiyhdistyksen testausasiantuntijoiden on kuitenkin vielä testattava uusi moduuli ennen kuin moduulia voidaan viedä kentällä oleville toimipisteille testattavaksi. Uusi NVRAM on vielä tällä hetkellä testausasiantuntijoiden testattavana.

6 TULOKSET

Ohjelmiston absoluuttista luotettavuutta on mahdotonta määrittää matemaattisin menetelmin. Luotettavuuden määrittelemiseen on kuitenkin kehitetty useita menetelmiä, joiden avulla pystytään antamaan jonkinlainen käsitys ohjelmiston luotettavuudesta. Luotettava ohjelmisto täyttää vaatimukset, suoriutuu erilaisista syötteistä ja ympäristöistä, pystytään huoltamaan vioista, pystytään testaamaan ja verifioimaan sekä pystyy palautumaan virhetilanteista [17]. Tässä luvussa tarkastellaan uuden NVRAM-moduulin luotettavuutta vaatimusten perusteella, testien avulla sekä kenttäkokemuksien avulla.

6.1 Vaatimusten verifiointi

Luotettava ohjelmisto täyttää sille asetetut vaatimukset, joten on luonnollista lähteä tarkastelemaan uuden NVRAM-moduulin luotettavuutta sille asetettujen vaatimusten perusteella. Vaatimusten verifioinnin avulla saadaan selville, onko uusi NVRAM-moduuli toteutettu oikein. Tässä luvussa määritellään vaatimusten verifiointimenetelmä. Sitä hyödyntäen verifioidaan uuden NVRAM-moduulin vaatimukset.

Vaatimusten verifiointikokoukseen osallistui kehittäjien lisäksi Raha-automaattiyhdistyksen alusta-arkkitehdit. Aluksi kokouksessa käytiin läpi uuden NVRAM-moduulin toiminnallisuus pääpiirteittäin sekvenssi- ja vuokaavioiden avulla. Tämän jälkeen sekvenssi- ja vuokaavioiden täsmällisyyttä varmistettiin ohjelmakoodia katselmoimalla. Katselmoinnin yhteydessä myös moduulin testausmenetelmät tarkistettiin ja hyväksyttiin. Testausmenetelmät on käsitelty luvussa 5.4 ja testausraporttia käsitellään luvussa 6.2. Lopuksi vaatimuksista tehtiin yhteenveto ja kokouksen perusteella verifioitiin uuden NVRAM-moduulin vaatimukset. Uuden NVRAM-moduulin täydellinen vaatimuslistaus on esitelty taulukossa 8.

Taulukko 8: Uudelle moduulille asetetut vaatimukset ja niiden verifioinnin tulos.

Vaatimus	**Selitys**	**Tulos**
LF01	Jokaiselle asiakasmoduulille on pystyttävä tarjoamaan luku- ja kirjoitusrajapinta haihtumattomaan pysyväismuistiin.	OK
LF02	Moduulin tulee estää virheellisten lukujen ja kirjoitusten suorittaminen.	OK
LF03	Moduulin tulee suorittaa muistikopioiden hallintatoimet, kuten alustus ja uusien muistilohkojen luonti.	OK
LF04	Moduulin on pystyttävä säilyttämään peliautomaatin identiteettiä.	OK
LF05	Moduulin on pystyttävä havaitsemaan vikatilanteet.	OK
LF06	Moduulin on pystyttävä vikatilanteissa palautumaan hukkaamatta tietoja.	OK
F01	Viestirajapinta yhteensopiva nykyisen moduulin kanssa.	OK
F02	Nykyisen muistimallin (kuva 4) konvertointi uuteen muistimalliin (kuva 5).	OK
F03	Kirjoitusten onnistuminen varmennettava.	OK
F04	Moduulin tulee olla haihtumattoman muistin määrästä riippumaton.	OK
F05	Yksittäisille asiakasmoduuleille on pystyttävä tarjoamaan yli neljän kilotavun kokoisia muistilohkoja (rajoitettu vanhassa neljään kilotavuun).	OK
NF01	Luotettavuuden parantaminen	Huomioitavaa
NF02	Virheiden raportointi	OK
NF03	Dynaamisuus	OK, Huomioitavaa
NF04	Ylläpidettävyys	Huomioitavaa
NF05	Suorituskyky	OK

Taulukon 8 pohjalta voidaan todeta, että uusi NVRAM-moduuli täyttää kaikki sille asetetut vaatimukset. Taulukon perusteella ei kuitenkaan voida todeta, kuinka hyvin vaatimukset ovat täyttyneet, sillä myös nykyinen NVRAM-moduuli täyttää sille asetetut vaatimukset.

Ei-toiminnallisten vaatimusten verifioinnin kanssa havaittiin kuitenkin ongelmakohtia. Suurin osa ei-toiminnallisista vaatimuksista vaatii käytännön kokemuksia pidemmältä aikaa. Käytännön kokemuksien puutteita käsitellään luvussa 6.3.

Uuden NVRAM-moduulin luotettavuutta on yritetty mitata kattavin yksikkötestein, moduulitestein sekä rasitustestein, mutta luotettavuuden parantamisesta (NF01) ei kuitenkaan voida vielä olla täysin varmoja. Moduulin uskotaan olevan paljon luotettavampi kuin nykyisen moduulin, mutta testausolosuhteissa on hankala aiheuttaa oikeiden käyttäjien ja olosuhteiden aiheuttamia satunnaisilmiöitä. Uusi moduuli kuitenkin korjaa kaikki nykyisen moduulin heikkoudet, joten tässä vaiheessa voidaan sanoa, että luotettavuus on parantunut.

Uusi NVRAM-moduuli täyttää tämän projektin tavoitteiden osalta dynaamisuusvaatimuksen (NF03). Uusi moduuli ei aseta muita rajoitteita asiakaslohkojen määrälle kuin fyysisen pysyväismuistin määrän. Uusia asiakaslohkoja pystytään lisäämään XML-parametrimuutoksilla eikä parametrien järjestyksellä ole väliä. Parametrien perusteella pystytään myös määrittelemään käytettävän pysyväismuistin kokonaismäärä. Tulevaisuudessa asiakaslohkojen kokoa halutaan pystyä muuttamaan tarpeen mukaan. Tätä ei kuitenkaan vaadittu tämän projektin tavoitteissa.

Ylläpidettävyyttä (NF04) on hyvin haastavaa arvioida ennen varsinaisia ylläpitotarpeita. Tästä syystä kyseistä vaatimusta ei tässä vaiheessa vielä pystytä verifioimaan.

6.2 Testausraportti

Luotettavuutta on kuitenkin luontevampi mitata testausmenetelmien avulla, sillä hyvät ja kattavat testit kertovat luotettavuudesta paljon enemmän kuin lista vaatimuksista. Testien avulla vaatimukset on sitä paitsi helpompi verifioida. Tässä luvussa tarkastellaan uuden NVRAM-moduulin testaustuloksia. Tulosten analysointi rajataan NVRAM-moduulin kriittisimpään toimintoon, eli NVRAM-moduulin alustukseen. Alustuksen yhteydessä havaitaan mahdolliset muistikorruptiot, joiden seurauksena suoritetaan pysyväismuistiin kohdistettuja korjausoperaatioita. Taulukossa 9 on esitelty uuden NVRAM-moduulin alustuslogiikan testausraportti.

Taulukko 9: Alustuslogiikan testausraportti pohjautuen taulukossa 7 määriteltyihin eheystiloihin.

Testi	Kolikkolukkokortin tila(t)	Hopperikortin tila(t)	Odotettu tulos	Tulos
T01	S1, kirjoituslaskuri = 1	S1, kirjoituslaskuri = 2	Valitse 1	OK
T02	S1, kirjoituslaskuri = 3	S1, kirjoituslaskuri = 2	Valitse 2	OK
T03	S1	S2, S3, S4, S5, S6	Valitse 1	OK
T04	S2, S3, S4, S5, S6	S1	Valitse 2	OK
T05	S2, kirjoituslaskuri = 1	S2, kirjoituslaskuri = 2	Valitse 1 + korjaa toimilaitetunnisteet	OK
T06	S2, kirjoituslaskuri = 3	S2, kirjoituslaskuri = 2	Valitse 2 + korjaa toimilaitetunnisteet	OK
T07	S2	S3, S4, S5, S6	Valitse 1 + korjaa toimilaitetunnisteet	OK
T08	S3, S4, S5, S6	S2	Valitse 2 + korjaa toimilaitetunnisteet	OK
T09	S3, S4, S5, S6	S3	Valitse 2 + päivitä datamalli	OK
T10	S3	S5, S6	Valitse 1 + päivitä datamalli	OK
T11	S3*, S4, S5, S6	S4	Valitse 2 + päivitä datamalli	OK
T12	S4	S5, S6	Valitse 1 + päivitä datamalli	OK
T13	S5, S6	S5, S6	Valinta ei mahdollinen, NVRAM on alustettava	OK

Alustuslogiikan testeissä on otettu huomioon kaikki mahdolliset kombinaatiot, joihin kahdella erillisellä toimilaitekortilla voidaan joutua. Taulukon 9 perusteella voidaan todeta, että uusi NVRAM-moduuli suoriutuu kaikista tilanteista odotusten mukaisesti. Testissä T11 on kolikkolukkokortin mahdollisiin tiloihin määritelty S3, mutta käytännössä tämä ei kuitenkaan ole mahdollista silloin, kun hopperikortin tila on S4. Vaikka tilannetta ei pystykään syntymään luonnollisissa olosuhteissa, niin uuden NVRAM-moduulin painotus kohdistuu vanhojen datamallien kohdalla aina hopperikorttiin. Hopperikortilla on tilastoalue, jota ei haluta menettää. Nykyisen NVRAM-moduulin datamallissa tilastoalue tallennettiin pelkästään hopperikortille, eikä tilastoaluetta haluta menettää datamallia päivitettäessä.

Kattavat testit helpottavat myös moduulin ylläpitoa. Testien avulla uusien ominaisuuksien lisääminen on helpompaa, sillä uutta ominaisuutta lisättäessä voidaan testien avulla todeta nopeasti, onko uusi toiminnallisuus rikkonut vanhoja toimintoja.

6.3 Kenttäkokemukset

Uusi pysyväismuistimoduuli on testattu useilla eri menetelmillä, mutta tärkeintä menetelmää ei ole vielä pystytty hyödyntämään. Kyseessä on tietenkin käytännön kokemuksella saadut tulokset. Kehitysympäristö ja testauslaboratorio ovat liian kontrolloituja ympäristöjä, että pystyttäisiin 100-prosenttisella varmuudella toteamaan, että uusi NVRAM-moduuli on luotettavampi kuin nykyinen. Peliautomaatteja sijaitsee monenlaisissa ympäristöissä, mikä tuo mukanaan paljon satunnaisuutta, jota ei voida ennustaa.

Uutta NVRAM-moduulia on tarkoitus asentaa joulukuussa muutamiin pelisaleissa sijaitseviin peliautomaatteihin. Mikäli mitään hälyyttävää ei moduulin toiminnassa ilmaannu, niin moduulia lähdetään levittämään muihinkin peliautomaatteihin muiden päivitysten mukana. Levitys on kuitenkin sen verran hidasta, että kattavaa raporttia moduulin toiminnasta ei kuitenkaan voida odottaa kuin aikaisintaan tammikuussa 2012. Tästä huolimatta kehitystiimillä on kova luotto uuden NVRAM-moduulin toimintaan. Tiimi uskoo vikojen vähentyvän verrattuna nykyiseen NVRAM-moduuliin.

7 YHTEENVETO

Tässä insinöörityössä oli päätavoitteena toteuttaa uusi luotettavampi pysyväismuistimoduuli Raha-automaattiyhdistyksen peliautomaatteihin. Nykyisestä moduulista löydettiin useita ongelmakohtia, joita analysoimalla uuden moduulin luotettavuutta pystyttiin parantamaan verrattuna nykyiseen. Lisäksi luotettavuutta lähdettiin tavoittelemaan testauslähtöisillä kehitysmenetelmillä.

Kehitystyön yhteydessä uuden NVRAM-moduulin toiminnallisuutta testattiin yksikkötestien ja moduulitestien avulla. Yksikkötesteillä varmistettiin, että yksittäiset funktiot toimivat oikein sallituilla sekä virheellisillä syötteillä. Projektin lopussa yksikkötestit kattoivat 85 % uuden NVRAM-moduulin lähdekoodista. Uutta moduulia testattiin yksikkötestejä korkeammalta tasolta käyttäen moduulitestejä. Moduulitestien avulla pystyttiin varmistumaan, että moduulin suorittamat toimintoketjut suoritetaan oikein ja virheelliset syötteet havaitaan.

Työn aikana suurimmaksi ongelmaksi muodostui nykyisen moduulin vanhentunut dokumentaatio. Suurimman osan nykyisestä toiminnallisuudesta joutui päättelemään lähdekoodin avulla, mikä oli lähdekoodin laadun takia työlästä. Tulkkausvirheiden takia esimerkiksi konversiologiikassa tuli tehtyä väärä olettamus liittyen nykyiseen datamalliin.

Muutamia tulkkausvirheitä ja kehitystyönaikaisia lisävaatimuksia lukuunottamatta projekti sujui ongelmitta.

Tavoitteiden voidaan sanoa täyttyneen luvussa 6 esiteltyjen tulosten perusteella. Uusi moduuli selviytyy projektin aikana tehtyjen testien perusteella kaikista vaadituista tilanteista, mukaan lukien erikoistilanteet. Uusi toteutus on lähdekoodirivien perusteella selkeästi isompi kuin nykyisen moduulin toteutus, mutta hyvien ohjelmointitapojen ja selkeiden rajausten ansiosta jatkokehitettävyyden pitäisi olla aivan toista luokkaa.

Uuden NVRAM-moduulin todellinen luotettavuus jää kuitenkin tässä vaiheessa vielä avoimeksi, sillä uutta NVRAM-moduulia ei olla vielä levitetty kentälle toimipisteisiin. Uutta moduulia on määrä asentaa muutamaan peliautomaattiin tämän vuoden lopulla. Jos lyhyen testijakson aikana moduulin toiminnallisuudesta ei löydetä hälyttäviä ongelmia, niin uutta moduulia voidaan lähteä asentamaan päivitysten yhteydessä muihinkin peliautomaatteihin. Uuden moduulin toimivuudelta ja luotettavuudelta odotetaan kuitenkin paljon.

7 VIITELUETTELO

[1] About busybox. Internet-julkaisu: http://www.busybox.net/about.html. Viitattu 7.10.2010.

[2] Dielectric. Internet-julkaisu: http://en.wikipedia.org/wiki/Dielectric. Viitattu 23.9.2010.

[3] Eeprom. Internet-julkaisu: http://www.tech-faq.com/eeprom.html. Viitattu 7.10.2010.

[4] Ferroelectric ram. Internet-julkaisu: http://en.wikipedia.org/wiki/Ferroelectric_RAM. Viitattu 23.9.2010.

[5] Gcc, the gnu compiler collection. Internet-julkaisu: http://gcc.gnu.org/. Viitattu 6.11.2010.

[6] gcov(1) - linux man page. Internet-julkaisu: http://linux.die.net/man/1/gcov. Viitattu 6.11.2010.

[7] Ram disk. Internet-julkaisu: http://en.wikipedia.org/wiki/RAM_disk. Viitattu 7.10.2010.

[8] Unittest++. Internet-julkaisu: http://unittest-cpp.sourceforge.net/. Viitattu 8.10.2010.

[9] Valgrind. Internet-julkaisu: http://valgrind.org/info/about.html. Viitattu 6.11.2010.

[10] DUSTIN, E. *Effective Software Testing - 50 Specific Ways to Improve Your Testing*. Addison Wesley Longman Inc, 2002.

[11] FEATHERS, M. C. *Working Effectively with Legacy Code*. Prentice Hall PTR, 2004.

[12] FOWLER, M. *Refactoring - Improving the Design of Existing Code*. Addison Wesley Longman Inc, 1999.

[13] GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. *Design Patterns*. Addison Wesley Longman Inc, 1994.

[14] HALTSONEN, S., LEVOMÄKI, J., AND RAUTANEN, E. T. *Digitaalitekniikka*. Edita Publishing Oy, 2006.

[15] KERNIGHAN, B. W., AND PIKE, R. *The Practice of Programming*. Addison Wesley Longman Inc, 1999.

[16] MYERS, G. J., BADGETT, T., SANDLER, C., AND THOMAS, T. M. *The Art of Software Testing*. John Wiley and Sons Inc, 2004.

[17] NEUFELDER, A. M. *Ensuring Software Reliability*. Marcel Dekker, Inc, 1993.

[18] SEITZ, F. *Solid State Physics: Advances in Research and Applications*. Academic Press Inc, 1957.