Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Ohjelmistotuotantoprojekti

Esimerkkituoteperhe

Vaatimusdokumentti

12.03.2004

Ryhmä 6
Juha Andersson
Jarmo Kielosto
Leo Linnamaa
Jan Tilles
Joose Vettenranta

Versiohistoria

Versio	*Päivämäärä*	*Muutokset*
1.6	12.03.2004	Jäädytetty versio
1.5	11.03.2004	FTR:n jälkeiset päivitykset
1.4	08.03.2004	Asiakastapaamisen jälkeen tarkennettu vaatimukset
1.3	04.03.2004	Asiakastapaamisten jälkeen päivitetty vaatimukset ja arkkitehtuuri
1.2	26.02.2004	Arkkitehtuuriluvun päivitys
1.1	22.02.2004	Asiakkaan kommenttien mukaan muutettu
1.0	16.02.2004	Ensimmäinen versio ryhmän ja asiakkaan arvioitavaksi

SISÄLTÖ

1. Johdanto

Tässä luvussa esitellään projektin aihepiiri yleisesti.

1.1 Projekti

Projektin aikana toteutetaan yksinkertainen tuoteperhe Java-ohjelmointikielellä. Tuoteperheellä tarkoitetaan tässä joukkoa samankaltaisia sovelluksia, joilla on yhteinen kehysrakenne. Tämän projektin tuoteperhe sisältää kolme sovellusta, jotka toimivat esimerkkiaineistona RITA-projektissa. Sovellusten aihepiiriksi asiakas ehdotti harrastuspelien kirjanpito- ja pistelaskuohjelmia. Projektiryhmä toteuttaa seuraavat pelit: **1) mökkitikka, 2) snooker ja 3) keilailu**.

Kehysrakenteeseen liittyy erityisiä arkkitehtuurisia vaatimuksia, jotka asiakas määrittelee yhdessä projektiryhmän kanssa. Arkkitehtuuriset vaatimukset liittyvät asiakkaan kehittämään RITA-projektiin. Tuoteperheeseen kuuluvien sovellusten osittaista yhtenäisyyttä hyödynnetään perheen arkkitehtuuria suunniteltaessa.

Ohjelmistolla on oliopohjaisen ohjelmistokehysmallin mukainen ydin, josta erikoistetaan tämän tuoteperheen sovellukset ja joka on myös laajennettavissa uusilla sovelluksilla. Jokainen perheen sovellus noudattaa samaa, tuoteperheen yhteiset osat sisältävää, arkkitehtuuria.

1.2 Termit ja lyhenteet

Core

Ohjelmiston pääkomponentti, jossa on sovellukset käynnistävä luokka

Erikoistaminen

Komponentteja laajennetaan (extends) sovelluskohtaisesti. Erikoistaminen voidaan toteuttaa abstraktilla luokalla, konkreettisen luokan perinnällä tai rajapintaluokalla.

GUI

Graafinen käyttöliittymä (Graphical User Interface)

Hook-metodi

Metodi, joka on luokassa erikoistamiskohta, jonka erikoistaja syrjäyttää (kts. myös template-metodi)

Hotspot

Sovelluskohtaisesti erikoistettava kohta

Java

Ohjelmistossa käytetty Sun-yhtiön kehittämä ohjelmointikieli. URL: http://java.sun.com/

JUnit

Java-ohjelmien testaukseen kehitetty testikehys. URL: http://www.junit.org/

Ohjelmistokehys

Ohjelmistokehys (framework) tarkoittaa toisiinsa liittyvien luokkien ja rajapintojen kokoelmaa, joka toteuttaa tietyn ohjelmistoperheen perusarkkitehtuurin.

RITA

Ohjelmistotuoteperheiden testaamiseen keskittynyt projekti (Environment for Testing Framework-based Software Product Families). URL: http://www.cs.helsinki.fi/group/rita/

Template-metodi

Metodi, joka kutsuu hook-metodia

Tuoteperhe

Joukko samankaltaisia sovelluksia, joilla on yhteinen kehysrakenne

Ydin

Ohjelmiston kehys kokonaisuudessaan. Ytimestä erikoistetaan sovellukset.

2. Asiakkaan vaatimukset

Tässä luvussa selvitetään asiakkaan antamat vaatimukset tuotettavalle ohjelmistolle.

2.1 Asiakkaan toimiala

Laitoksella tammikuussa 2002 aloitetussa RITA-projektissa (http://www.cs.helsinki.fi/group/rita/) tutkitaan kehyspohjaisten ohjelmistotuoteperheiden testausteoriaa ja -menetelmiä sekä kehitetään näitä tukevaa työkalua. RITA-projekti on osa ROOSAa (http://www.cs.helsinki.fi/research/roosa/), joka on Helsingin yliopiston tietojenkäsittelytieteen laitoksen ohjelmistoarkkitehtuureihin keskittyvä tutkimusryhmä.

Ohjelmiston kehittämisen tärkein syy on se, että RITA-työkalulle saadaan testi- ja esittelyaineistoa. RITA-työkalu on kehitysvaiheessa oleva kehyspohjaisten tuoteperheiden testausohjelma. Ohjelmalla voidaan esittää yksittäisestä tuotteesta tai koko tuoteperheestä erilaisia näkymiä, kuten komponenttirakenteita, luokkakaavioita tai yksittäisen metodin vuokaavioita. Tuotteiden eri osiin voidaan liittää JUnit-testejä, ja ohjelma laskee ja esittää visuaalisesti erilaisia yleisiä ja kehysarkkitehtuuriin liittyviä testikattavuuksia.

2.2 Toiminnalliset vaatimukset

Tuoteperheen sovellukset ovat toiminnallisesti yksinkertaisia. Kaikkien sovellusten tulee kyetä tarjoamaan pelikirjanpidon perustoiminnot (vaatimus 2.2.1), jotka ovat:

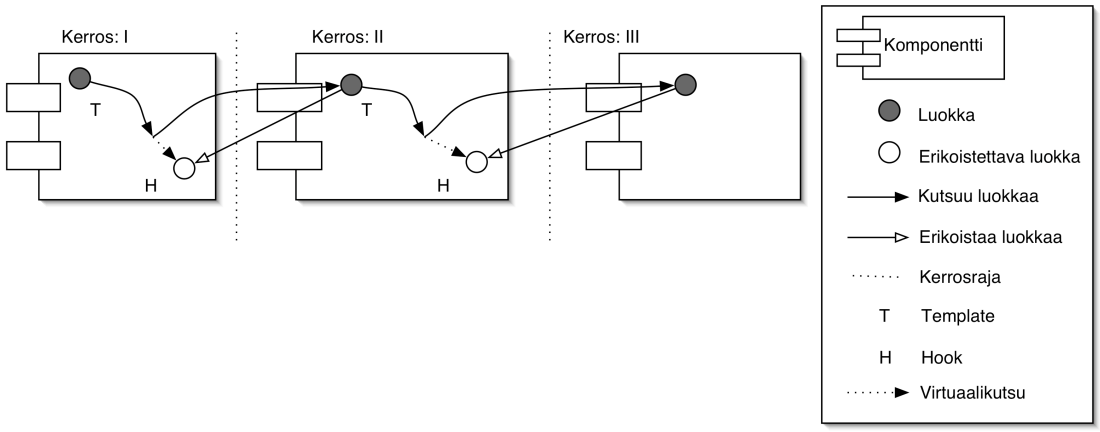
- pelaajien lisääminen,
- pistekirjanpito ja
- tulosten ilmoittaminen.

Lisäksi ohjelmiston ytimen tulee olla laajennettavissa siten (vaatimus 2.2.2), että siihen voidaan lisätä muita, vaihtoehtoisia palveluita. Tällaisia toimintoja voivat olla esimerkiksi pelin tallentaminen ja avaaminen, pelaajahistorian, parhaiden tulosten, päiväyksen ja pelipaikan tallentaminen.

2.3 Laadulliset vaatimukset

Asiakkaan ohjelmistolle asettamat laadulliset vaatimukset ovat seuraavat:

- Vaatimus 2.3.1: Projektin kaikissa tuotoksissa käytetään eksplisiittistä ja yhtenäistä esitystapaa. Projektin eri vaiheissa esitystavan tulee olla johdonmukainen, jotta ohjelmistoperhettä voidaan demonstroida.
- Vaatimus 2.3.2: Sovellukset ovat eritasoisia; yksi sovellus toteutetaan mahdollisimman yksinkertaisesti ja yksi selkeästi muita monimutkaisempana.
- Vaatimus 2.3.3: Sovelluskohtaisten eli erikoistettujen komponenttien tulee olla vaihdettavissa toisiin komponentteihin siten, että lopputuloksena saadaan kääntäjästä läpi menevä tuote. Tällaisen tuotteen ei kuitenkaan tarvitse olla toiminnaltaan järkevä. Lisäksi komponenttien vaihtamisen tulee olla mahdollisimman yksinkertaista.
- Vaatimus 2.3.4: Kaikki luokat ovat osa jotakin komponenttia, ja komponentit eivät saa olla sisäkkäisiä. Yksiluokkaisia komponentteja voidaan toteuttaa.
- Vaatimus 2.3.5: Ydin on mahdollisimman sama kaikilla sovelluksilla, ja kaikki sovelluskohtaiset lisäykset tehdään määriteltyjen hotspottien kautta.
- Vaatimus 2.3.6: Kerrosarkkitehtuurissa noudatetaan ns. Hollywood-periaatetta (”Don´t call us, we’ll call you”).
- Vaatimus 2.3.7: Ohjelmistoarkkitehtuuri toteutetaan puhtaalla (ei kutsuja kerrosten yli) kolmikerrosarkkitehtuurilla, jolla asiakas tarkoittaa kuvassa 2.1 esitettyä rakennetta:



Kuva 2.1: Kolmikerrosarkkitehtuuri

- Vaatimus 2.3.8: Ytimeen toteutetaan mahdollisimman paljon ohjelmistosta. Ytimen toteutuksen tulee painottua siten, että ensimmäiseen kerrokseen tulee mahdollisimman paljon ja kolmanteen kerrokseen mahdollisimman vähän toteutusta.
- Vaatimus 2.3.9: Kolmikerroksinen rakenne täytyy toteuttaa joidenkin, muttei välttämättä kaikkien komponenttien osalta.
- Vaatimus 2.3.10: Arkkitehtuuri toteutetaan siten, että jotkut komponentit ovat kaikkien kolmen sovellusten käytössä, jotkut kahden sovelluksen käytössä ja jotkut ovat sovelluskohtaisia.
- Vaatimus 2.3.11: Arkkitehtuurissa erikseen nimetyt komponentit toteutetaan vain tyhjänä runkona (skeleton), jotka ovat myöhemmin toteutettavissa (implementoitavissa). Tyhjä runko toteutetaan luokkana, joka ei sisällä kuin tyhjät metodit.
- Vaatimus 2.3.12: Erikoistaminen voidaan toteuttaa kolmella tavalla: 1) abstrakteilla luokilla, 2) rajapintaluokilla (interface) ja 3) konkreettisen luokan perinnällä. Asiakkaan vaatimus on, että jokaista tapaa käytetään ainakin kerran.
- Vaatimus 2.3.13: Koodi (luokkien, metodien ja muuttujien nimet sekä kommentit) ja käyttöliittymät toteutetaan englanniksi.
- Vaatimus 2.3.14: Sovellusten on erotuttava toisistaan noin 5-10 kehyksestä erikoistettavan tai käytettävän luokan osalta.
- Vaatimus 2.3.15: Sovellusten tulee olla yksinkertaisia, mutta silti niiden pitää sisältää riittävästi variaatiota analysointia varten.

- Vaatimus 2.3.16: Ohjelmisto ei saa käyttää **java.lang.reflect**-paketin luokkia.

2.4 Testivaatimukset

Testauksessa on noudatettava seuraavia vaatimuksia:

- Vaatimus 2.4.1: Ohjelmistoa tulee voida demonstroida RITA-työkalulla.
- Vaatimus 2.4.2: Ohjelmisto testaus tulee toteuttaa JUnit-työkalulla.
- Vaatimus 2.4.3: Arkkitehtuuri toteutetaan niin, että samaa testikoodia on mahdollista käyttää eri sovelluksille.
- Vaatimus 2.4.4: Yksikkö- ja integraatiotestit erotetaan toisistaan sekä nimetään selkeästi.
- Vaatimus 2.4.5: Kaikki konfiguraatiot ovat testattavissa; myös eri sovelluksien komponentteja yhdistämällä tehdyt sovellukset tulee voida testata.
- Vaatimus 2.4.6: Ohjelmistolle toteutetaan testitapauksia, joissa kerros-arkkitehtuuria käydään läpi siten, että kaikki kerrokset tulevat testattua.
- Vaatimus 2.4.7: Testit ryhmitellään testijoukoiksi (test suite) eri ominaisuuksien mukaan.

3. Ohjelmiston toiminnot

Tässä luvussa esitellään kuvaus ohjelmistoperheen sovelluksista sekä niiden yksittäisistä toiminnoista. Ensin käsitellään kunkin sovelluksen toiminnot, minkä jälkeen kaikki toiminnot eritellään.

Tämän projektin aikana jätetään kaikissa sovelluksissa toteuttamatta tietokanta- ja tietoliikenneyhteyksien luominen ja sulkeminen. Ytimen kehysrakenteen osana saatetaan toteuttaa luokkia tai sellaisia toimintoja, joiden osat mahdollistavat tulevaisuudessa edellä mainittujen toimintojen toteutukset, mutta itse sovellukset eivät tässä projektissa ota mitään kantaa niihin.

3.1 Mökkitikka

Mökkitikassa pelaajien määrä voi olla periaatteessa mikä hyvänsä, mutta sovelluksessa pelaajien määrä rajataan maksimissaan kymmeneen. Jokainen pelaaja heittää vuorollaan viisi tikkaa, joista jokainen antaa 0-10 pistettä. Tavoitteena on saada mahdollisimman paljon pisteitä. Sovellus tarjoaa mahdollisuuden lisätä pistemäärän joko syöttämällä pistemäärä syötekenttään, jolloin sovellus tarkistaa, että syöte on oikea (kokonaisluku väliltä 0-10), tai klikkaamalla käyttöliittymässä olevaa tikkataulua. Kun kaikki pelaajat ovat heittäneet viisi tikkaa, sovellus tulostaa pelin tulokset (pelaajatiedot sekä pisteet) pisteiden mukaisessa järjestyksessä.

Koska asiakkaan vaatimus on, että yksi sovellus tulee pitää erittäin yksinkertaisena, mökkitikka toteutetaan sellaisena. Mökkitikan toiminnot ovat kaikki osana asiakkaan toiminnallisia perusvaatimuksia. Toiminnot ovat seuraavat:

- **Pelaajien lukumäärän valitseminen**. Käyttäjä valitsee mökkitikan pelaajien lukumäärän. Syötteen arvon tulee olla väliltä 1-10. Sovellus tarkastaa, että syöte on oikealta arvoalueelta.

- **Pelaajatietojen syöttäminen**. Käyttäjä syöttää jokaisen pelaajan nimen. Nimen pituus rajataan 1-50 merkkiin. Sovellus tarkastaa, että nimet ovat oikean pituisia.
- **Pelin aloittaminen**. Käyttäjä aloittaa pelin. Aloitukseen toteutetaan joko pudotusvalikko- tai painonappivaihtoehto. Pelaaja voi aloittaa uuden kierroksen myös kesken pelin, jolloin pistekentät nollataan.
- **Pisteiden lisääminen**. Käyttäjä voi lisätä pisteitä kahdella tavalla: käyttöliittymään toteutetaan tikkataulua symboloiva kuva, jota klikkaamalla voi valita oikean pistemäärän (jolloin pistemäärä siirtyy myös tekstikenttään), tai kirjoittamalla pistemäärän tekstikenttään. Tällöin sovellus tarkastaa, että syöte on oikealta arvoalueelta 0-10.
- **Loppuraportin katsominen**. Kun kaikki pelaajat ovat heittäneet viisi tikkaa, sovellus tulostaa pelaajat pisteiden mukaisessa järjestyksessä siten, että eniten pisteitä saanut heittäjä on ensimmäisenä.
- **Pelin lopettaminen**. Käyttäjä sulkee ikkunan tai valitsee pudotusvalikosta sulje-toiminnon.

3.2 Snooker

Snookerissa pelaajia on aina kaksi, ja kierroksia pelissä on niin kauan, kunnes kaikki pallot on pussitettu. Pelin ideana on saada mahdollisimman paljon pisteitä lyömällä palloja pussiin ja saada oman vuoron jälkeen valkoinen pallo sellaiseen paikkaan, että vastustaja tekee virhelyönnin. Vastustajan virhelyönnistä pelaaja saa neljästä seitsemään pistettä.

Pelin alussa yritetään pussittaa punainen pallo, joita on 15 kappaletta. Pussitetusta punaisesta pallosta saa yhden pisteen. Kun pelaaja on saanut punaisen pallon pussitettua, hänen pitää lyödä muun värinen pallo pussiin, mistä hän saa 2-7 lisäpistettä pallon värin mukaan. Kun kaikki 15 punaista palloa on saatu pois pöydältä, muita värejä ei nosteta enää sen jälkeen takaisin pöydälle. Värilliset pallot pussitetaan pienimmästä suurimpaan, eli kahden pisteen pallo ensin ja viimeiseksi seitsemän pisteen pallo. Vuoroa pitää pelissä voida vaihtaa myös muuten kuin virheestä. Peli loppuu, kun kaikki pallot on lyöty pusseihin.

Snooker-sovelluksen toiminnot ovat seuraavat:

- **Pelaajien lukumäärän valitseminen**. Snookerissa pelaajien lukumäärä on rajoitettu kahteen. Lukumäärän valitseminen ei ole käyttäjälle näkyvä toiminto, mutta kuitenkin sovelluksen toiminto.
- **Pelaajatietojen syöttäminen**. Käyttäjä syöttää kahden pelaajan nimet. Nimen pituus rajataan 1-50 merkkiin. Sovellus tarkastaa, että nimet ovat oikean pituisia.
- **Pelin aloittaminen**. Käyttäjä aloittaa pelin. Aloitukseen toteutetaan joko pudotusvalikko- tai painonappivaihtoehto. Uusi Snooker-peli voidaan aloittaa myös kesken käynnissä olevan pelin, jolloin käynnistetty peli alkaa alusta ja vanha peli häviää.
- **Pisteiden lisääminen**. Käyttäjä voi lisätä pisteen tai virhepisteen käyttöliittymässä. Sovellus pitää kirjaa pussitetuista palloista ja määrittelee sen mukaan, mitä pisteitä voidaan syöttää.
- **Tallentaminen**. Sovellus laskee yhdessä lyöntivuorossa tehdyt pisteet ja pitää kirjaa parhaimmista. Snooker-sovellus pitää yllä parhaan lyöntivuoron Top 10 –listaa.
- **Top 10 -listan selaaminen**. Snooker-sovelluksessa käyttäjä voi hakea halutessaan parhaiden lyöntivuorojen tilastoja nähtäväkseen. Lista pitää hakea erikseen nähtäväksi valitsemalla kyseinen toiminto alasvetovalikosta.
- **Loppuraportin katsominen**. Sovellus kertoo pelin päätyttyä voittajan, pisteet, parhaimman breikin (lyöntivuoron) sekä mahdollisen sijoittumisen Top 10 -listalle.
- **Pelin lopettaminen**. Käyttäjä sulkee ikkunan tai valitsee pudotusvalikosta sulje-toiminnon.

3.3 Keilailu

Keilailusovelluksen tarkoituksena on pitää kirjaa 1-4 pelaajan keilailupelin tuloksista. Ohjelma kirjaa pelaajista nimen ja sukupuolen. Sukupuolen kirjaus on tarpeellista mahdollisen tasoituksen huomioimiseksi pistelaskussa.

Pelitilanteessa ohjelma ottaa vastaan numerosyötteenä näppäimistöltä pelaajien tuloksia. Tulokset lasketaan reaaliaikaisesti yhteen kaikkien pelaajien osalta. Reaaliaikaisuudella tarkoitetaan pisteiden laskua sitä mukaa, kun syötteitä annetaan ohjelmalle. Ohjelmassa on mahdollisuus muokata jo annettuja syötteitä virheellisen syötteen korjaamiseksi.

Pelin loputtua ohjelma tuottaa pelaajista tuloslistan, jossa on esitetty pelaajien suoritukset ja yhteistulos paremmuusjärjestyksessä. Tuloslistan luonnin jälkeen syötteitä ei voi enää muuttaa. Ohjelman avulla käyttäjä voi halutessaan tallentaa saadut tulokset muistiin, jolloin ohjelma voi myös tuottaa tuloslistan, jossa näkyvät parhaat tulokset pidemmältä aikaväliltä. Aikaväliä ei ole määritelty, vaan se lasketaan aina ensimmäisestä tallennetusta tuloksesta asti. Tuloslistan esityksen jälkeen on mahdollista aloittaa seuraavan pelin tulosten kirjaaminen ilman sovelluksen uudelleenkäynnistystä.

Optiona sovelluksessa on pelaajamäärän laajentaminen siten, että ohjelma voisi hallinnoida myös useita 2-4 pelaajan joukkueita samanaikaisesti. Joukkueiden yhteistulokset olisivat myös näkyvissä sitä mukaa, kun pelaajien tuloksia syötetään.

Koska asiakkaan vaatimus on, että yksi sovellus olisi muita sovelluksia monimutkaisempi, keilailu toteutetaan sellaisena. Keilailusovelluksen toiminnot ovat seuraavat:

- **Pelaajien lukumäärän valitseminen**. Käyttäjä syöttää pelaajien lukumäärän sovellukselle; pelaajia voi keilailusovelluksessa olla 1-4 samanaikaisesti.
- **Pelaajatietojen syöttäminen**. Nimen pituus rajataan 1-50 merkkiin. Sovellus tarkastaa, että nimet ovat oikean pituisia.

- **Pelin aloittaminen**. Käyttäjä aloittaa pelin. Aloitukseen toteutetaan joko pudotusvalikko- tai painonappivaihtoehto. Pelaaja voi aloittaa uuden kierroksen myös kesken pelin, jolloin pistekentät nollataan.
- **Pisteiden lisääminen**. Käyttäjä lisää pisteet sovelluksen tekstikenttiin. Sovellus pitää huolen siitä, että syötetty pistemäärä on keilailun sääntöjen mukainen. Pisteitä voi lisätä lukuvälillä 0 – 9, lisäksi erilliset keilailun ”/” (paikko)- ja ”X” (kaato)-merkit ovat käytössä. Pisteiden korjaaminen voidaan tehdä suoraan tekstikenttiin.
- **Loppuraportin katsominen**. Keilailusovellus kertoo pelin päätyttyä voittajan ja pisteet. Sovelluksen käyttäjä voi halutessaan tallentaa loppuraportin myöhempää selailua varten.
- **Loppuraporttien selaaminen**. Keilailusovelluksen käyttöliittymä tarjoaa automaattisesti mahdollisuuden selailla loppuraportteja ilman, että niitä tarvitsee hakea erikseen katseltavaksi. Viimeisen kolmen pelin loppuraportit ovat nähtävissä automaattisesti.
- **Tallentaminen**. Sovellus tallentaa parhaat tulokset.
- **Pelin lopettaminen**. Käyttäjä sulkee ikkunan tai valitsee pudotusvalikosta sulje-toiminnon.

3.4 Toimintojen koonti

Kaikkiin sovelluksiin toteutetaan graafinen käyttöliittymä. Koska asiakkaalla ei ole erityisiä vaatimuksia käyttöliittymästä, käyttöliittymän tarkempi kuvaus esitetään vasta suunnitteludokumentissa.

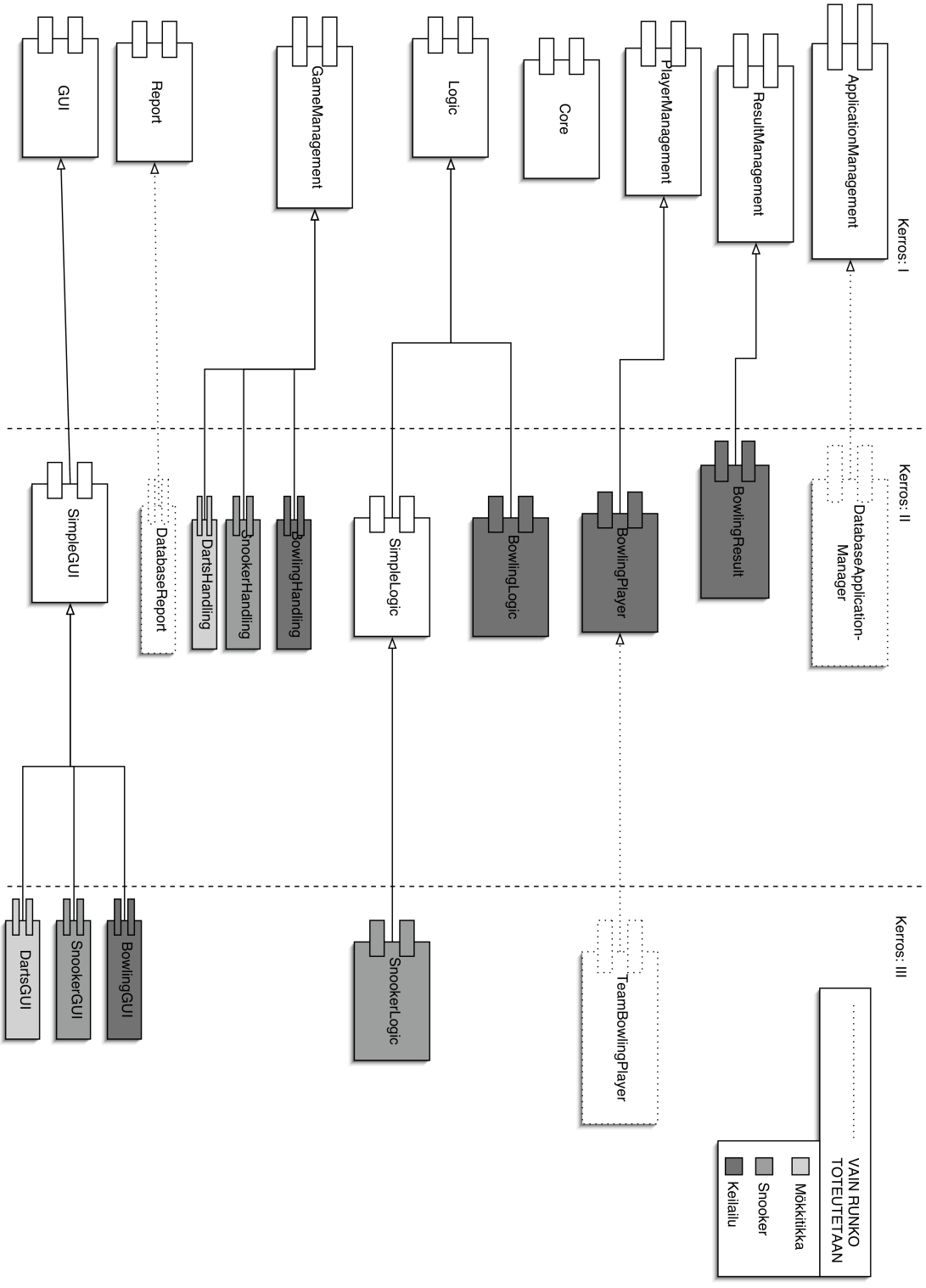
Kaikki sovellusten (käyttäjälle näkyvät) toiminnot on koottu taulukkoon 1:

Toiminto	**Mökkitikka**	**Snooker**	**Keilailu**
Pelaajatietojen syöttäminen	X	X	X
Pelin aloittaminen	X	X	X
Pisteiden lisääminen	X	X	X
Loppuraportin katsominen	X	X	X
Pelin lopettaminen	X	X	X
Pelaajien lukumäärän valitseminen	X		X
Tallentaminen		X	X
TOP 10 -listan selaaminen		X	
Loppuraporttien selaaminen			X

Taulukko 1: Sovellusten toiminnot

4. Arkkitehtuurisuunnitelma

Ytimen sisältämä toteutus pyrkii vastaamaan mahdollisimman paljon mökkitikkasovelluksen toteutusta, sillä mökkitikkasovellus sopii perustoteutukseltaan useisiin peleihin (yksinkertainen pistelaskulogiikka). Tuoteperheen ydin tarjoaa sovelluksille yhteisen rakenteen ja toimintamallin. Sovellukset voivat erikoistaa osia ytimen komponenttien tarjoamista palveluista tai käyttää suoraan ytimen tarjoamia perusratkaisuja. Tässä dokumentissa ei oteta kantaa siihen (vaan vasta suunnitteludokumentissa), miten erikoistaminen toteutetaan, vain erikoistettavat asiat on mainittu. Tuoteperheen kaikki komponentit on esitetty kuvassa 4.1.



Kuva 4.1 Komponenttirakenne

Ytimen komponenttikuvassa nuolimerkintä tarkoittaa, että komponenttien sisältämissä luokissa on erikoistamiskohtia (hotspotteja), jotka seuraavan kerroksen komponentti toteuttaa.

Seuraavaksi esitellään jokainen komponentti tarkemmin. Erikoistettavista komponenteista on lisäksi omat kuvansa.

4.1 Core

Core-komponentti sisältää ainoastaan luokan Core, jossa on main()-metodi, joka käynnistää sovellusten alustuksen. Varsinainen alustus tapahtuu kuitenkin ApplicationManager-komponentissa. Alustuksessa käytetään apuna sovelluskohtaisia alustustiedostoja.

4.2 ApplicationManagement

ApplicationManagement-komponentti määrittelee perustoiminnot sovelluksen käynnistymiseen ja lopetuksen hallinnointiin. Komponentti huolehtii myös alustuksessa käytettävän sovelluskohtaisen alustustiedoston läpikäymisestä. Alustustiedosto pitää sisällään tiedot sovelluksen erikoistamista komponenteista. Erikoistettavia toimintoja ovat:

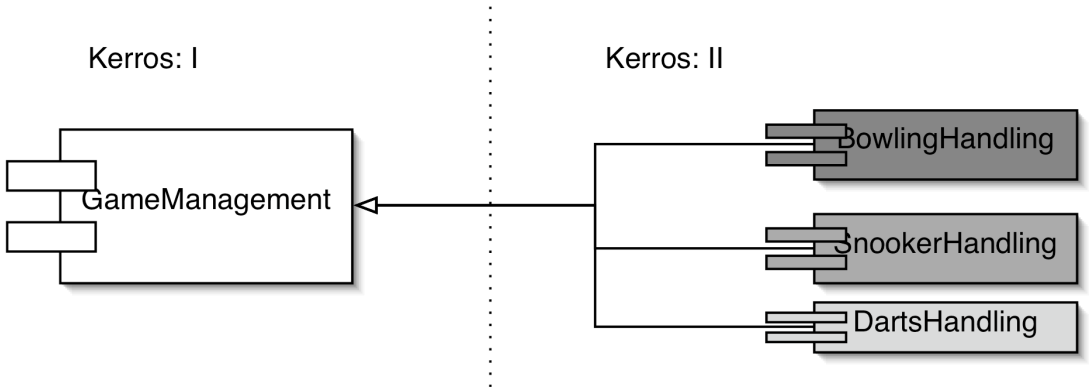
- sovelluksen alustaminen ja
- sovelluksen lopettaminen.

Komponentin perustoteutusta käyttävät kaikki tuoteperheen sovellukset. Lisäksi komponentista erikoistetaan DatabaseApplicationManagement, joka kuitenkin toteutetaan vain tyhjänä runkona (esitetty kuvassa 4.1).

4.3 GameManagement

GameManagement ja siihen liittyvät komponentit tarjoavat sovelluksille pelin aloittamiseen, lopettamiseen ja pelinaikaisten tapahtumien reagointiin tarvittavat toiminnot. Erikoistettavia toimintoja ovat:

- pelin aloittaminen,
- pelin lopettaminen,
- tapahtumiin reagoiminen (syötteiden oikeellisuuden kontrolli) ja
- pelitilannekontrolli (kierrosten ja vuorojen hallinta).

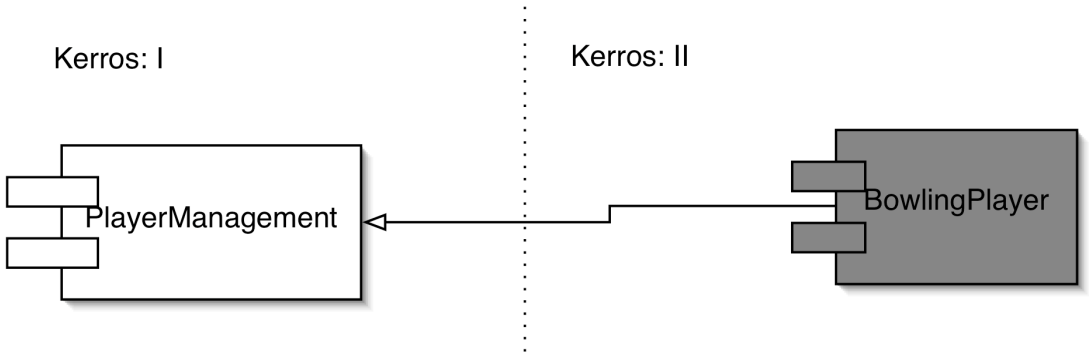


Kuva 4.2 GameManagement ja siihen liittyvät komponentit

4.4 PlayerManagement

Komponentti huolehtii pelaajahallinnasta sekä pelaajakohtaisista tiedoista, kuten pelaajien nimistä. PlayerManagement-komponenttia erikoistamalla voidaan lisätä joillekin peleille ominaisia tietoja pelaajista. PlayerManagement-komponentin erikoistettava ominaisuus on:

- pelaajatiedot.



Kuva 4.3 PlayerManagement ja siihen liittyvä komponentti BowlingPlayer

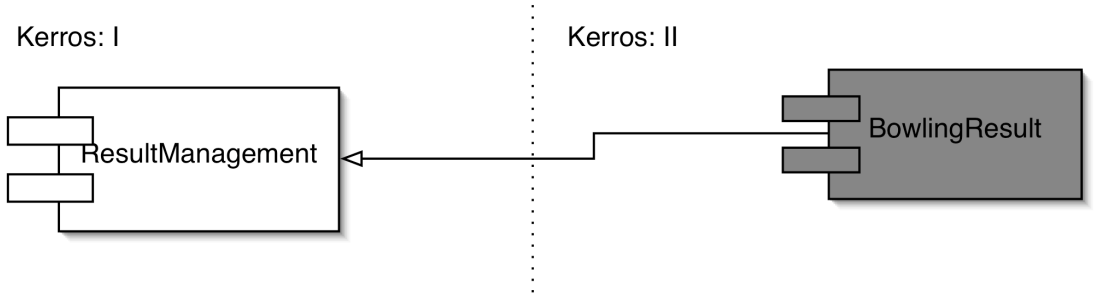
Snooker- ja mökkitikkasovellukset käyttävät PlayerManagement-komponenttia suoraan, ja keilailu erikoistaa oman BowlingPlayer-komponentin PlayerManagement komponentista. Lisäksi komponentista BowlingPlayer erikoistetaan TeamBowlingPlayer, joka kuitenkin toteutetaan vain tyhjänä runkona (esitetty kuvassa 4.1).

4.5 ResultManagement

ResultManagement-komponentti huolehtii pelinaikaisesta tietojen, esimerkiksi pisteiden, tallentamisesta tietorakenteeseen. Komponentista erikoistettavia ominaisuus on:

- tietorakenteen toteutus.

Komponentti tarjoaa perustietorakenteen sovellusten käyttöön. Tietorakenne voi olla esimerkiksi linkitetty lista.



Kuva 4.4 ResultManagement ja siihen liittyvä komponentti BowlingResult

Snooker- ja mökkitikkasovellukset käyttävät ResultManager-komponenttia suoraan, ja keilailu erikoistaa oman BowlingResult-komponentin ResultManager komponentista.

4.6 Report

Report-komponentti määrittelee työkalut pelien tuloslistojen luomiselle ja niiden vertailulle. Komponentin erikoistettavia toimintoja ovat:

- tiedostonhallinta,
- tulosluettelon luominen ja
- pelin tulosten vertaileminen.

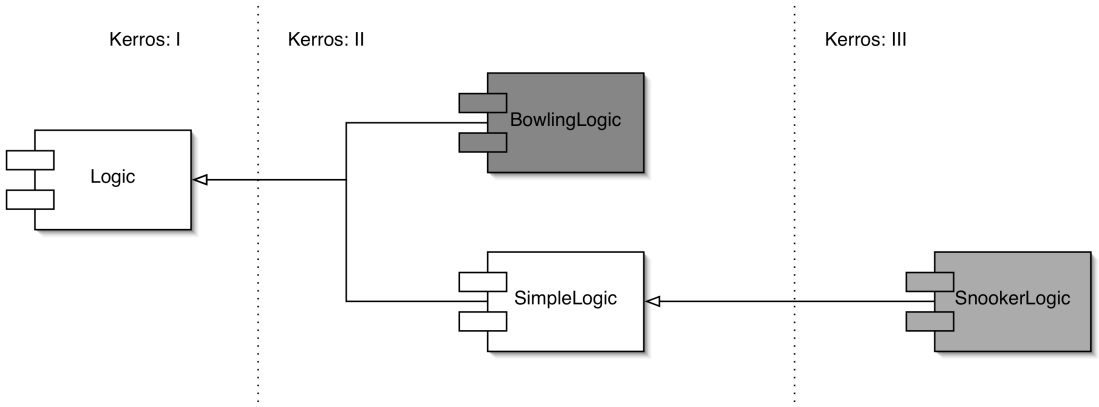
Komponentti tarjoaa toiminnot sovellusten käytettäväksi.

Lisäksi komponentista erikoistetaan DatabaseReport-komponentti, joka kuitenkin toteutetaan vain tyhjänä runkona (esitetty kuvassa 4.1). Tähän liittyy siis mahdollisuus kytkeä komponentti tietokantaa hyödyntäväksi.

4.7 Logic

Logic-komponentti määrittelee pelin logiikkaa koskevat palvelut. Erikoistettavia toimintoja ovat:

- pisteiden oikeellisuuden tarkistaminen,
- pisteiden raja-arvojen määrittäminen ja
- pelaajamäärän raja-arvojen määrittäminen.



Kuva 4.5 Logic ja siihen liittyvät komponentit

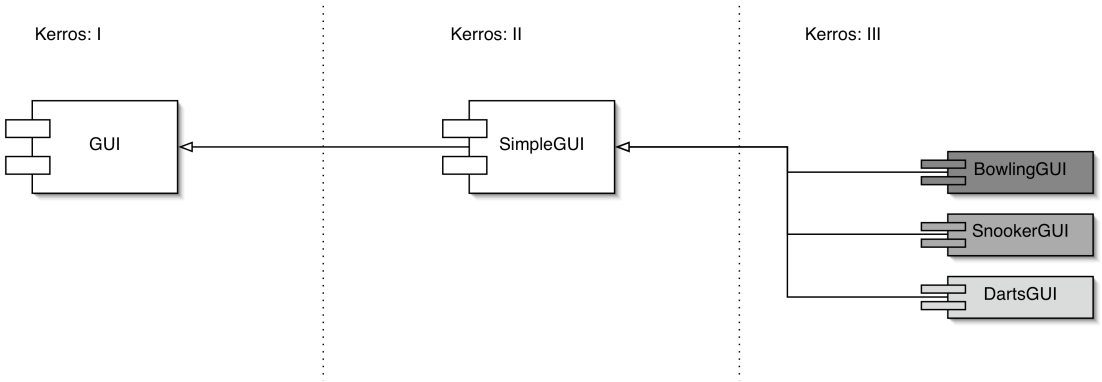
Mökkitikkasovellus käyttää SimpleLogic-komponenttia suoraan. Snooker-sovellus erikoistaa SimpleLogic-komponentista oman SnookerLogic-komponentin. Keilailusovellus erikoistaa Logic-komponentista oman BowlingLogic–komponentin.

4.8 GUI

GUI-komponentti on sovelluksen käyttöliittymän sisältävä komponentti, jonka kautta käyttäjä tuottaa tapahtumat.

Erikoistettava toiminta on:

- ikkunointi.



Kuva 4.8 GUI ja siihen liittyvät komponentit

SimpleGui-komponentti erikoistetaan Gui-komponentista. SimpleGui-komponentin idea on toteuttaa graafinen ikkunoitu käyttöliittymä. Tässä projektissa jokainen sovellus erikoistaa oman komponenttinsa SimpleGui-komponentista.

5. Validointi

Validoinnin tarkoitus on osoittaa, että ohjelmisto toteuttaa asiakkaan vaatimukset. Projektin erikoisuus on, että vaatimukset eivät liity kovinkaan paljon itse tuotteeseen ja sen ulkoiseen toiminnallisuuteen vaan arkkitehtuuriin. Taulukossa 2 käydään läpi asiakkaan vaatimukset sekä projektiryhmän tapa toteuttaa ne.

Vaatimus	Validointi
Luvun 2.2 vaatimukset:	
Perustoimintojen toteutus: pelaajien lisääminen, pistekirjanpito sekä tulosten ilmoittaminen.	Toteutetaan ytimessä ja erikoistetaan jokaiselle pelille.
Luvun 2.3 vaatimukset:	
Eksplisiittinen ja yhtenäinen esitystapa	Ryhmä on sopinut koodauksen konventioista projektisuunnitelmassa. Komponenttien nimeämistapaa käytetään myös suunnitteluvaiheessa.
Eritasoiset sovellukset	Mökkitikka toteutetaan yksinkertaisena, keilaus monimutkaisempana sovelluksena.
Komponenttien vaihdettavuus	Komponenttien välille toteutetaan yhteinen rajapinta, joka määritellään suunnitteluvaiheessa.
Komponentit eivät ole sisäkkäisiä	Otetaan suunnittelussa huomioon. Kts. kuva 4.1.
Mahdollisimman yhtenäinen ydin sovelluksilla	Toteutetaan lihava (fat) kehysrakenne, jossa koodia on eniten ensimmäisellä kerroksella.
Puhdas kerrosarkkitehtuuri	Kerrosmalli on otettu huomioon arkkitehtuuria suunnitellessa. Kts. kuva 4.1.
Hollywood-periaate	Toteutusta kuvattu luvussa 4 sekä esitetty kuvassa 4.1
Ytimessä mahdollisimman paljon ohjelmistosta, toisessa kerroksessa osa ja sovelluskohtaiset kolmannessa kerroksessa	Komponentit on suunniteltu siten, että erikoistamisessa tarvitaan mahdollisimman vähän koodia.
Jotkut komponentit kaikkien sovellusten käytössä, jotkut osan sovelluksista	Otetaan suunnittelussa huomioon. Kts. kuva 4.1.
Osa komponenteista tyhjiä runkoja	Toteutetaan komponentit tyhjinä runkoina.
Erikoistaminen kolmella tavalla	Otetaan suunnittelussa huomioon.

Koodi ja kommentit englanniksi	Ryhmä on sopinut tästä jo projektisuunnitelmassa.
Sovellukset yksinkertaisia mutta varioituja	Pelit on valittu siten, että niissä on erilaisia pelaajamääriä ja pistelaskutapoja.
Sovellusten erotuttava toisistaan 5-10 kehyksestä erikoistettavan tai käytettävän luokan osalta	Erot on kuvattu eri väreillä kuvassa 4.1.
Ohjelmisto ei saa käyttää **java.lang.reflect**-paketin luokkia	Niitä ei käytetä.
Luvun 2.4 vaatimukset:	
Testivaatimukset	Kaikki testivaatimukset otetaan huomioon testaussuunnitelmassa, joka liitetään osaksi suunnitteludokumenttia.

Taulukko 2: Validointi