Auri Laitinen – Niko Mehiläinen - Samuel Sarimo

Jätehuoltosimulaattori

Metropolia Ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotuotanto
Ohjelmistoprojekti
16.03.2025

Sisällys

Liite 1. Javadoc-dokumentaatio

1 Johdanto

Jätehuollon tehokkuus on keskeinen osa kaupunkien kestävää kehitystä ja resurssien hallintaa. Tässä dokumentissa käsitellään jätehuoltosimulaattorin kehitystä, jonka tavoitteena on mallintaa ja optimoida roska-autojen tyhjennysaikatauluja sekä roskakatosten kapasiteetin mitoittamista asukasmäärän nähden.

Simulaattorilla voidaan helposti kokeilla eri skenaarioita ja saada arvokasta tietoa oikean päätöksen tueksi. Oli kyseessä sitten neljän huoneiston rivitalo tai isompi asukaskompleksi.

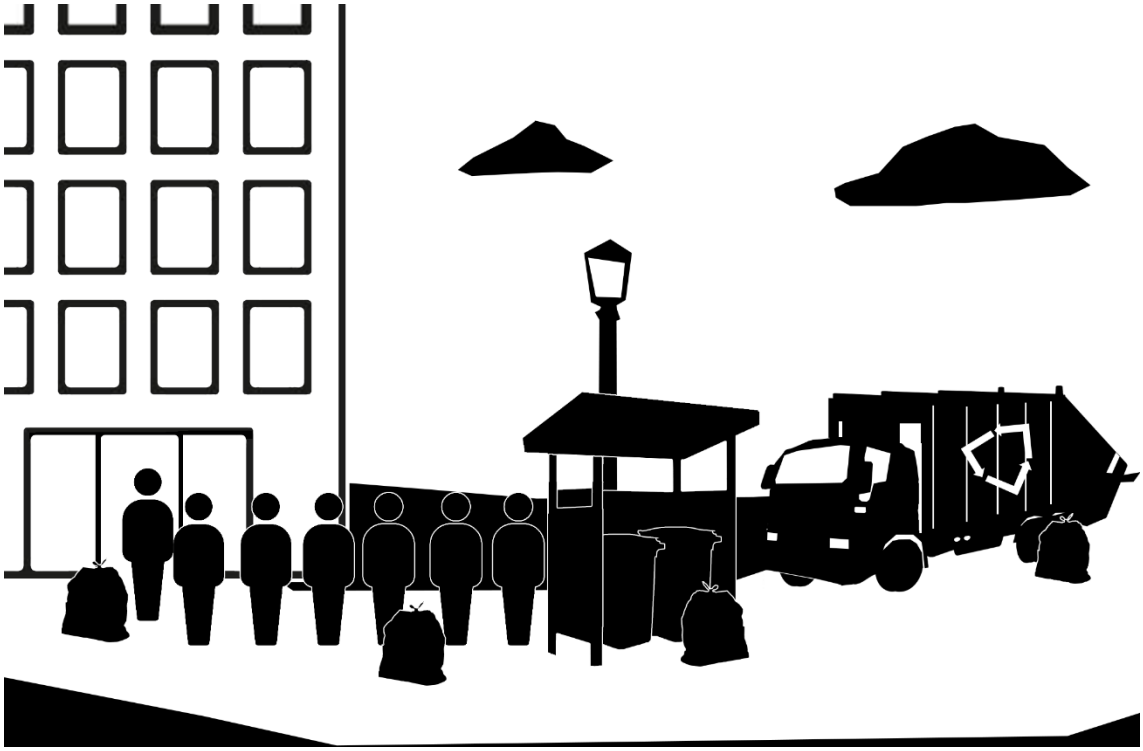
Dokumentti auttaa ymmärtämään simulaattorin toiminnan syvällisemmin esittelemällä tarkasti sen toimintaperiaatteen, ohjelmointiteknisen toteutuksen sekä käsittelemällä simulointituloksia.

2 Visio

Projektin tavoite on kehittää jätehuoltosimulaattori, joka optimoi roska-autojen käyntivälit, sekä taloyhtiön roskakatoksen jäteastioiden kapasiteetin asukasmäärään nähden. Simulaattorin avulla voidaan määrittää eri kokoisille taloyhtiöille optimaaliset roskakatoksen tyhjennysaikavälit ja näin edistää kestävää kehitystä.

Lopputuotteena syntyy tehokas työkalu, joka auttaa jätehuoltoyhtiöitä ja kaupunkisuunnittelijoita tekemään optimaalisia dataperusteisia päätöksiä.

Kuvassa 1. on yksinkertaistettu havainnointikuva dokumentissa esiintyvästä simulaattorista. Siinä huoneiston asukkaat tulevat korkeasta kerrostalosta tyhjentämään kertynyttä jätettä roskakatokseen.



Kuva 1. Roskakatos simulaatio

Kuten kuvasta 1. huomaamme, kyseiselle kerrostalolle on mitoitettu liian pieni roskakatos, näin ollen kaikki roskat eivät mahdu roskakatokseen. Simulaation avulla voidaan optimoida roskakatoksen kapasiteetti ja ennaltaehkäistä vastaavanlaisia tapauksia.

3 Käsitteet, määritelmät

Seuraavaksi käydään läpi dokumentissa ilmeneviä käsitteitä, jotka ovat sen lukemisen ja ymmärtämisen kannalta tärkeitä.

Taloyhtiö on useasta *huoneistosta* ja yhdestä *roskakatoksesta* koostuva kokonaisuus.

Huoneistolla tarkoitetaan *taloyhtiössä* sijaitsevaa asuntoa, joka käyttää roskakatosta täyttäen siinä sijaitsevia jäteastioita. Huoneiston koko määrittää heitettävän roskan määrän. Yksiö tuottaa huomattavasti vähemmän roskaa neliöön nähden.

Roskakatoksella tarkoitetaan tilaa, johon *asukas* jonottaa vuoroaan täyttäen sitä systemaattisesti. Roskakatos sisältää useamman jäteastian, jonka avulla voidaan säätää tilan roskakapasiteettia.

Roska-auto on järjestelmällisesti määritetyn ajan välein *taloyhtiön roskakatosta* tyhjentävä taho.

Roska-astian käyttöasteella tarkoitetaan sen tyhjennysvaiheessa olevaa käytön hyödyllisyyttä. Tällä tiedolla voidaan selvittää, onko taloyhtiössä mahdollisesti liian suuri määrä käyttämättömiä jäteasioita vai onko niiden kapasiteetti aina täynnä viitaten niiden mahdolliseen lisätarpeeseen.

Roska-astian saatavuus kertoo roska-astian saatavuudesta päivinä. Jos roska-astia on saatavilla jokaisena päivinä tarkoittaa se sitä, että astia on tarpeen tullen tarpeeksi tyhjä eli roskakatoksessa on, joko tarpeeksi roska-astioita tai se tyhjennetään tarpeeksi säännöllisesti.

Roskakatoksen käyttöasteella on *roskakatoksen* tyhjennys vaiheessa määräytyvä arvo, joka kertoo, onko *roskakatoksen* kapasiteetti mahdollisimman optimaalinen eli tarpeeksi täynnä, että se olisi fiksua tyhjentää. Tämän ollessa alhainen tarkoittaa, se sitä, että yksittäisen *roska-astian käyttöaste* on matala sen tyhjennyksen aikana.

Tapahtuma on yksittäinen tietyllä ajanhetkellä tapahtuva ilmiö. Tapahtumia ovat esimerkiksi: *roskakatoksen* tyhjennys, *jäteastian* täyttyminen tai *asukkaan* ilmaantuminen roskakatokseen.

Tapahtumalista on yksittäisiä tapahtumia orkestroiva elementti, johon kaikki simulaattorin tapahtumat on syötetty. Tapahtumalista pitää huolen siitä, että kaikki siihen lisätyt tapahtumat toteutetaan oikealla hetkellä.

Data Access Object (DAO) on suunnittelumalli, joka vastaa tietokannan käsittelystä ja eristää tietojen hakemisen, tallentamisen ja poistamisen sovelluksen muusta logiikasta.

Entiteetti on olio, joka vastaa tietokannan taulua ja mallintaa sovelluslogiikan kannalta merkityksellistä tietoa. Jokainen entiteetti sisältää attribuutteja, jotka vastaavat taulun sarakkeita, ja ne määritellään usein luokkina, joita hallitaan esimerkiksi ORM-työkaluilla (Object-Relational Mapping), kuten JPA:lla, mitä hyödynnetään tässä projektissa.

CRUD (Create, Read, Update, Delete) on lyhenne neljästä keskeisestä tietokantaoperaatiosta: Luominen, Lukeminen, Päivittäminen, Poistaminen.

4 Käsitteellinen malli

4.1 Tavoite

Jätehuoltosimulaattorin tavoite on pyrkiä selvittämään millä tavoin roskan keruuta voidaan optimoida jätehuoltoyritysten, roskakatoksen käytettävyyden ja ekologisuuden näkökulmasta. Tuloksia voidaan käyttää löytämään optimaalinen roska-astioiden määrä taloyhtiön koon ja väestörakenteen suhteen, sekä löytämään paras hyötysuhde tyhjennysaikataululle.

4.2 Syötteet

Simulaattorin käyttäjä voi halutessaan syöttää eri lähtötietoja simulaattorille kokeillakseen erilaisia jätehuoltokonfiguraatioita.

Lähtötietoina simulointiajolle voi syöttää:

- Simulaatioaika
- Keskimääräinen roskan määrä per heitto
- Roskakatoksen tyhjennysväli
- Taloyhtiön eri kokoisten huoneistojen määrä
    - Yksiö
    - Kaksio
    - Kolmio
    - Neliö
- Taloyhtiön roska-astioiden määrät
    - Sekajäte
    - Muovijäte

- Biojäte
- Lasijäte
- Paperijäte
- Metallijäte

4.3 Tulosteet

Simulointiajoista saadaan dataa määrittämään sopiva roskakatosten koko (jäteastioiden määrä) minkä tahansa kokoiseen taloyhtiöön sopivaksi. Tulosteet myös auttavat löytämään sopivan roskien tyhjennystiheyden taloyhtiöiden roskantuotannon suhteen. Tulosteilla voi tarkastella tarvittavaa roska-auto kuljettajan työtehokkuutta annettuun työkierrokseen. Simuloimalla saadaan myös laskelmia jätehuollon kustannustehokkuudesta.

Tärkeimmät simulaation aikana ja jälkeen saatavat suureet ovat: jätehuoneen käyttöaste, tyhjennystehokkuus, sekä roska-auton kustannustehokkuus. Näiden tietojen perusteella käyttäjän on helppo havainnoida asettamien syötteiden optimaalisuus.

4.4 Sisältö

Malli rajoitetaan sisältämään pääpiirteittäin jätehuollon tärkeimmät osa-alueet asiakkaiden ja jätehuoltoyritysten näkökulmasta. Sisällytettynä ovat jätteenkeruuta tekevä roska-auto, huoneistot ketkä tuottavat jätettä ja jätteen väliaikaissäilöminen roskakatoksissa.

Simulaatiomalli pyrkii ottamaan huomioon reaalimaailman jätetutkimuksista saatua dataa. Suomalaisten vuodessa tuottamasta jätteestä löytyy erinäisiä tutkimuksia esim. HSY:n jätetutkimus 2024. Näistä hyödynnetään tietoa ihmisten keskimääräisestä jätteentuotosta ja miten jäte on jakautunut painollisesti biojätteen ja sekajätteen välillä. Jätehuollon aikataulut ovat mallinnettu käyttäen laatimisen apuna Helsingin kaupungin jätehuoltomääräyksiä.

4.5 Oletukset ja yksinkertaistukset

Malli tekee olettamuksia jäteastioiden ja tuotujen roskien tilavuuksista. Jäteastioiden kapasiteetti on siis roskien painoon suhteutettu arvio. Oletamme myös, että asukkaat eivät halua pakkautua roskakatokseen samanaikaisesti, vaan odottavat että aikaisempi asukas

on saanut lajiteltua roskansa. Roskia kerääntyy reaalimaailmassa ihmisten vuorokausirytmin mukaan epätasaisesti, mutta tämä jätetään alustavasti huomiotta.

4.6 Mallin kuvaus

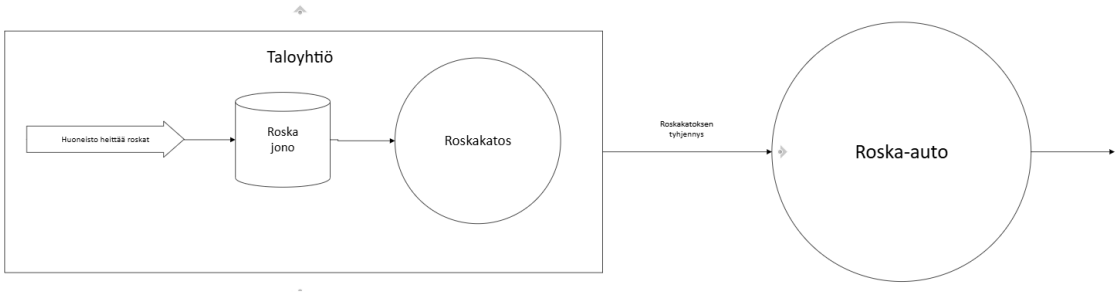
4.6.1 Komponenttilista

Komponentti	**Ominaisuuksia**
Roskakatosjono	Roskakatoksen jono. Ihmiset haluavat viedä roskansa jäteastiaan rauhassa. Jonoa alkaa muodostumaan eniten silloin kuin roskakatos on täynnä, eikä asukkaat saa heitettyä heidän roskiaan.
Huoneisto	Roskan saapumisväliaikojen jakauma.
Roska-auto	Tyhjentää roskakatoksen käyttäjän haluamalla aikavälillä.
Roskakatos	Rajoitettu kapasiteetti.
Roska-astia	Isot ja pienet roska-astiat, yhteensä 6 eri tyyppiä.

4.6.2 Prosessikaavio

Kappaleessa esiintyvät kaaviot havainnollistavat simulaattorin toimintaa ja auttavat ymmärtämään komponenttien ja tapahtumien välisiä suhteita.

Kuvassa 2 on esimerkki jätekatoksen toiminnasta, sekä sen tyhjennykseen liittyvästä prosessista. Huoneisto täyttää roskakatosta, ja niin kauan, kun siellä on tilaa niin roskat menevät onnistuneesti roska-astiaan.



Kuva 2. Taloyhtiön kuvaus prosessikaaviossa

Kuvassa 3 on hahmotelmaa simulaatiomallin tapahtumarakenteesta.

B EVENTS (Bound/Booked)			
Event	Type	Change in state	Future events to schedule
B1	Arrival (single,double,triple,quad)	Resident comes to roskakatos and wants to throw away trash.	B1
B2	Exit	Thrash has been thrown away.	
B3	Empty trash	Garbage shelter cans are emptied.	B3

C EVENTS (Conditional)			
Event	Type	Change in state	Future events to schedule
C1	Throw trash	Apartment succesfully thorws it's thrash into the garbage cans.	B2

Kuva 3. Tapahtumalista

5 Mallin ohjelmointitekninen toteutus

5.1 Käytetyt ohjelmointikielet ja kirjastot (ulkoiset API:t).

Simulaatioprojekti toteutettiin Java-ohjelmointikielellä hyödyntäen JavaFX-kirjastoa käyttöliittymän rakentamiseen ja visualisointiin. Käyttöliittymän tyylittely toteutettiin CSS kielellä.

Tietokantaratkaisuna simulaatio käyttää sql relaatiotietokantaa, projektin relaatiokantajärjestelmänä on käytössä *MariaDB*. Se on integroituna ohjelmistoon käyttäen Hibernate kehysrakennetta.

5.2 Arkkitehtuuri

Simulaatioprojektin arkkitehtuuri noudattaa MVC-mallia, jossa pääkomponentteja ovat Model, View ja Controller. MVC-mallin ideana on erottaa ohjelman logiikka käyttöliittymästä, mikä parantaa koodin ylläpidettävyyttä, laajennettavuutta sekä mahdollistaa komponenttien uudelleenkäytön eri projekteissa. Kuva 4. havainnollistaa MVC-mallia.

Kuva 4. MVC-malli

Kuvasta 4. näemme, että Controller toimii välikätenä Modelin ja Viewin välillä. Se vastaanottaa Viewiltä käyttäjän syötteet, kuten painikkeiden klikkaukset tai tekstikenttien syötteet, ja välittää ne Modelille. Tämän jälkeen Controller vastaanottaa käsitellyt tiedot Modelilta ja välittää ne Viewille, joka esittää ne visuaalisesti käyttäjälle.

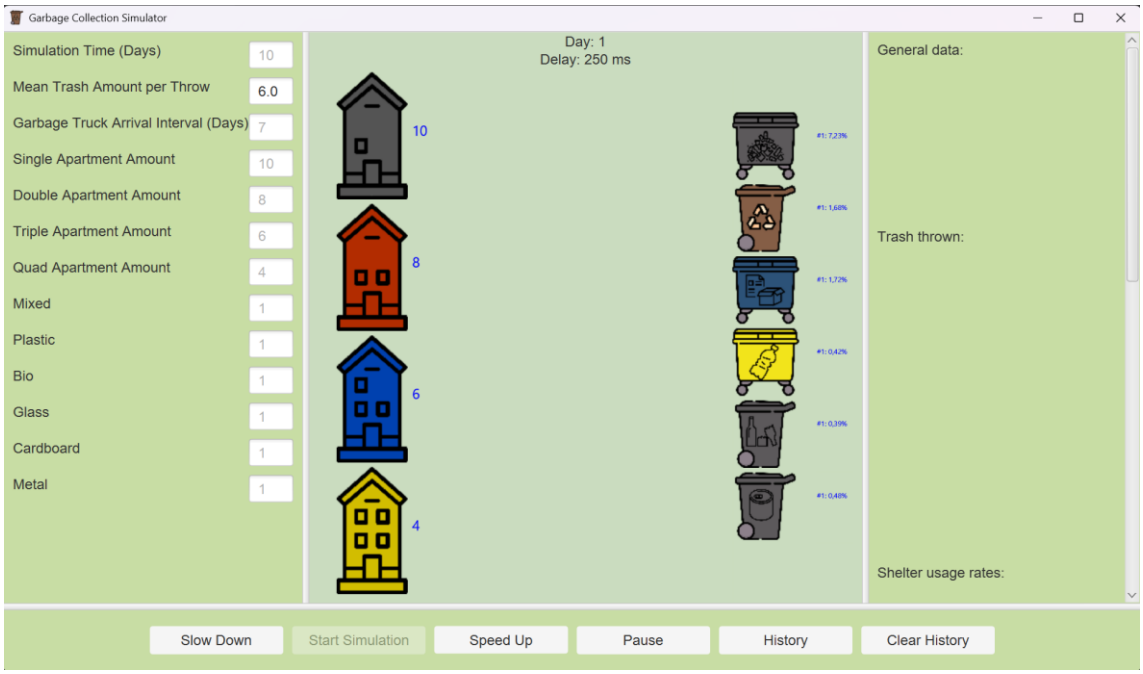
(25)

Tietokannan käsittely on jaettu kahteen osaan, jotta kerrosarkkitehtuuri säilyy mahdollisimman selkeänä ja eriytettynä. Controller käyttää DAO –rajapintaa, millä on mahdollista vain lukea tietokantaa. Model käyttää omaa DAO –rajapintaa, millä on täydet CRUD-oikeudet (Create, Read, Update, Delete). Tietokannan päivittäminen on näin eristetty täysin Modelille.

5.3 Käyttöliittymän rakennekuvaus.

Simulaattorin käyttöliittymä on pyritty toteuttamaan mahdollisimman pelkistetysti ja käytettäväksi kaikille. Se koostuu yhdestä näkymästä, johon käyttäjä voi syöttää haluamansa lähtöarvot ja aloittaa simuloinnin.

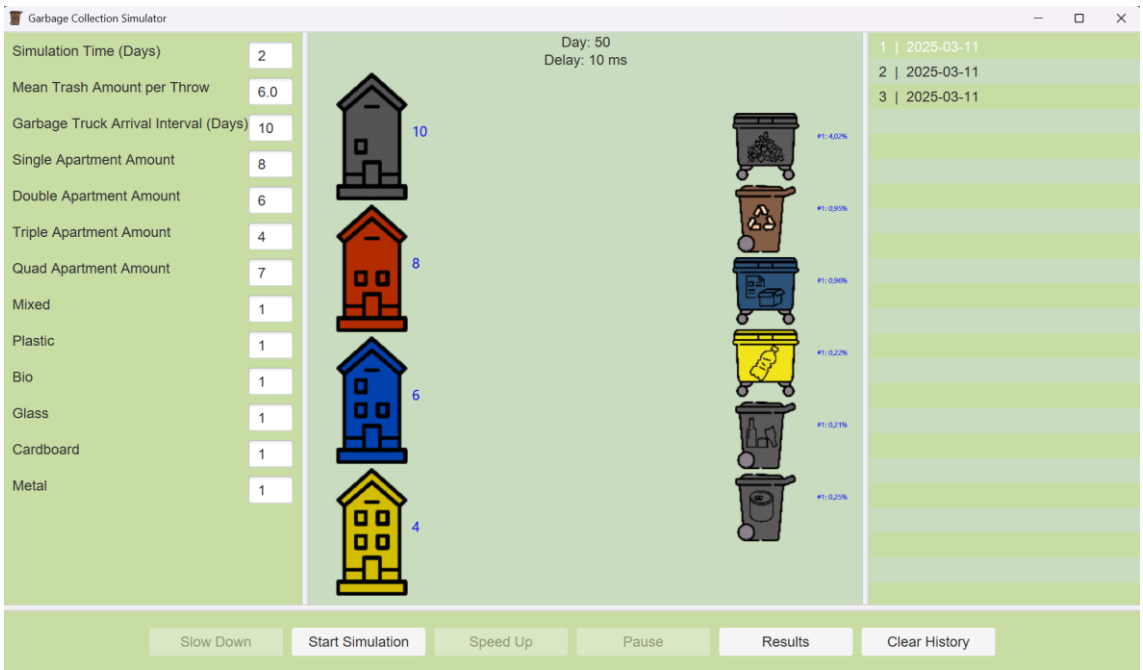
Käyttöliittymän voi jakaa karkeasti neljään osaan. Keskikenttä on simulaattorin visualisointi, vasemmalla on käyttäjän syöttämät alkuarvot, oikealle ilmestyy simuloinnin loputtua tulokset ja alakentässä käyttäjä voi ohjata simulaattoria painamalla haluamaansa näppäintä. Kts. Alempana kuvat 5, 6 ja 7.



Kuva 5. Kuvassa on simulaation ajon aikana oleva näkymä, joka avautuu, kun simulaattorin laittaa käyntiin. Vasemmassa kentässä on asetettu valmiiksi oletusarvot, joita käyttäjä voi halutessaan muuttaa.

(25)

Kuva 6. Kuvassa simulaattori simulaation päätyttyä. Kun simulaattorin aika on saavuttanut käyttäjän määritellyn ajan se tulostaa oikeanpuoleiseen kenttään simulaatioista saadut tiedot.



Kuva 7. Kuvassa käyttäjä selaa tietokantaan tallennettuja simulaatioajoja. Jokaisen simulaation alettua ohjelma tallentaa ajon lähtöarvot paikallisesti relaatiotietokantaan ja niitä voi tarvittaessa kokeilla uudelleen.

5.4 Sisäisen logiikan kuvaus

Ohjelmiston toiminta pohjautuu kolmivaiheiseen simulaatiomalliin (Three-Phase-Simulation). Simulaatio pyörii järjestelmän sisäisen kellon ympärillä, joka määrää sen, mikä ohjelmiston tietty tapahtuma suoritetaan ja milloin simulointi päättyy.

Jokainen yksittäinen roskan vienti on tapahtuma, joka suoritetaan tiettynä ajan hetkenä. Riippuen huoneiston koosta, roskien vientiväli on erilainen. Oletamme, että mitä suurempi on huoneiston koko, sitä suuremmalla todennäköisyydellä roskat kertyvät nopeammin, eli roskia on vietävä useammin. Jokaisella huoneistokoolla on uniikki tapahtuma moottorin tapahtumalistassa. Jokainen roskien vientitapahtuma generoi uuden saman tyyppisen tapahtuman. Nämä tapahtumat ovat määritelty ohjelmistossamme B-tyypin tapahtumiksi, eli ne toteutuvat aina kun niiden aika koittaa.

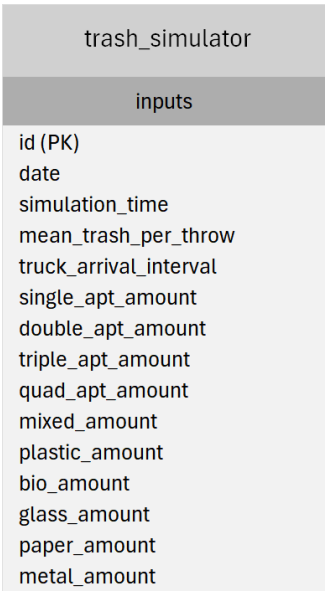
Roskien vientien lisäksi meillä on yksi uniikki tapahtuma, jota kutsumme roska-auto tapahtumaksi. Se kutsutaan ohjelmistoon asetetuin aikavälein esim. 7 päivän välein. Se on myös B-tyypin tapahtuma ja sen tehtävä on tyhjentää roska-astiat.

Simulaatiomallissa on myös yksi C-tyypin tapahtuma, joka määrittelee, onko tietyn tyyppinen roska-astia tarpeeksi tyhjä heitettävälle roskalle. Järjestelmässämme ei ole kolmivaihesimulaatiolle tyypillistä jonotusjärjestelmää missä roskia heittävät huoneistot odottaisivat milloin roska-auto tyhjentää astiat, vaan ylikertyneet roskat kirjataan ylös ja huoneiston poistumistapahtuma kutsutaan. Tällä tavoin saamme tärkeää dataa siitä, kuinka paljon huoneistoista kertynyttä roskaa menee yli roskakatoksen kapasiteetin.

Simulaatio etenee siten, että A-tyypin tapahtuman aikana kello katsoo tapahtumalistan läpi ja tarkistaa niiden aikaleimat. Simulaattori hyppää seuraavaan tapahtumaan, joka on ajallisesti lähimpänä järjestelmän kelloa. Hypättyään suoritettavaan tapahtumaan järjestelmä siirtää simulaattorin kellon vastaamaan tapahtuman aikaa. Se suorittaa kaikki sille ajalle määritellyt B-tyypin tapahtumat ehdoitta ja pyrkii suorittamaan kaikki ehdolliset C-tyypin tapahtumat. Näin ohjelman silmukka pyörii, kunnes järjestelmän kello vastaa asetettua simulaation pituutta.

5.5 Ulkoisten tietovarastojen (tiedostot, tietokannat) kuvaukset.

Simulaation alkaessa käyttäjän syöttämät tiedot tallennetaan paikalliseen trash_simulator SQL-relaatiotietokantaan, mikä mahdollistaa haluttujen simulaatioiden uudelleen ajon. Rakenteeltaan tietokanta on hyvin yksinkertainen, se sisältää yhden taulun, joka sisältää id:n, päivämäärän sekä simulaatioon syötetyt alkuarvot. Kuvassa 8. näemme tietokannan visuaalisesti.



Kuva 8. inputs-taulu trash-simulator tietokannassa.

Kuvasta näemme, että id toimii taulun pääavaimena (PK). Pääavainta käytetään halutun simulaatioajon hakuun tietokannasta.

5.5.1 Julkinen data

Tilastokeskus: Yhdyskuntajätteet lajittelutavoittain.

https://pxdata.stat.fi/PxWeb/pxweb/fi/StatFin__jate/statfin_jate_pxt_12cv.px/

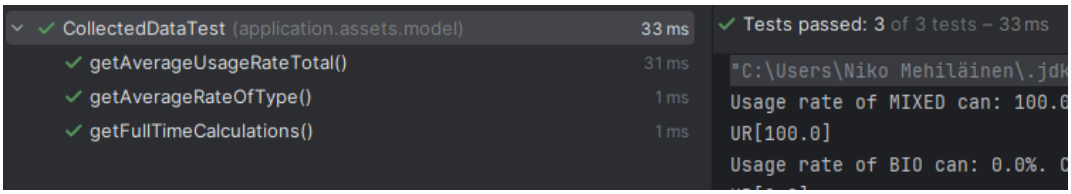
Yle: Kansalaisten kierrätysaste.

https://yle.fi/a/74-20035569

(25)

5.6 Testaus

Ohjelmiston kehityksen yhteydessä suoritettiin useampi yksikkötesti. Projektiryhmä keskittyi eniten testaamaan datan oikeellisuuden testaamista, jotta kaikki yksikkömuunnokset, sekä todennäköisyyslaskelmat olisivat suunnitellun mukaisia.



Kuva 9. Esitys suoritetuista yksikkötesteistä, jotka ovat menneet läpi. Jokainen testattava metodi pyrittiin testaamaan useammassa olosuhteessa, sekä monipuolisilla lähtöarvoilla.

6 Simulaattorin käyttöohje

6.1 Syötteet

Simulaattorin syötteet (kts. Kuva 10.) löytyvät ikkunan vasemmasta laidasta.

(25)

Kuva 10. Simulaattorin syötteet

Kuten kuvasta x näemme, syötteillä on valmiiksi asetettu alkuarvo, joka näkyy haalean harmaana. Jos käyttäjä ei muuta syötteitä, simulaatio toteutetaan alkuperäisarvoilla. Syötteet:

- Simulation Time
  Simulaatioaika päivinä.

- Mean Trash Amount per Throw
  Keskimääräinen roskan määrä heittoa kohden kilogrammoina.

- Garbage Truck Arrival Interval
  Roska-auton saapumisväli päivinä.

- Single Apartment Amount

(25)

Yksiöiden määrä.

- Double Apartment Amount
  Kaksioiden määrä.

- Triple Apartment Amount
  Kolmioiden määrä.
- Quad Apartment Amount
  Neliöiden määrä.

- Mixed
  Sekajäteastioiden määrä (660 L)

- Plastic
  Muovijäteastioiden määrä (660 L)

- Bio
  Biojäteastioiden määrä (240 L)

- Glass
  Lasijäteastioiden määrä (240 L)

- Cardboard
  Pahvijäteastioiden määrä (660 L)

- Metal
  Metallijäteastioiden määrä (240 L)

6.2 Käyttöliittymän Painikkeet

Ikkunan alalaidasta löytyvät käyttöliittymän painikkeet (kts. Kuva 11.).

(25)

Kuva 11. Käyttöliittymän painikkeet

Kuvassa 11. osa painikkeista on harmaana, mikä tarkoittaa, että simulaatiota ei ole vielä käynnistetty, eikä niitä voi vielä käyttää.

- Start Simulation
  Käynnistää simulaation syötetyillä alkuarvoilla.

- Slow Down
  Hidastaa simulaation kulkua 10 %.

- Speed Up
  Nopeuttaa simulaation kulkua 10 %.

- Pause
  Pysäyttää simulaation kulun.

- History / Results
  Vaihtaa tulosnäkymän historianäkymään. Painike vaihtuu "Results" painikkeeksi, jolla voi vaihtaa näkymän takaisin.

- Clear History
  Tyhjentää historian tietokannasta.

6.3 Historia

Simulaation alkaessa tietokantaan tallennetaan käyttäjän syöttämät alkuarvot. Tietokannasta tuodaan historianäkymään alkuarvo -entiteetin id sekä päivämäärä, kuten kuvassa 12. on esitetty.

Kuva 12. Historia näkymä

Kuvassa 12. viides tallennettu alkuarvo -entiteetti on valkoisella väritettynä ja tarkoittaa, että se on valittuna. Valitun entiteetin alkuarvot syötetään automaattisesti simulaattorin syötteisiin.

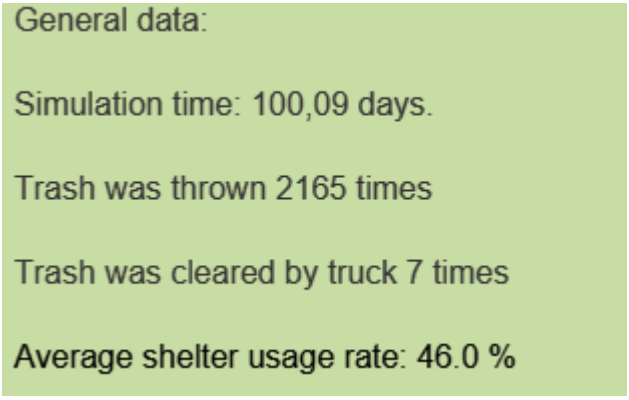
6.4 Tulokset

Simulaation annettujen tulosten perusteella on mahdollista optimoida kiinteistössä sijaitsevan jätekatoksen kapasiteettia, sekä tilattavan roska-auton tyhjennys väliä. Malli ei ota huomioon huoneistojen poikkeuskäytöksiä, esimerkiksi vääränlaisen jätteen tuomista katokseen esim. huonekalut tai elektroniikka. Simulaatiossa on oletettavasti poikkeamaa todellisen roskakatokseen.

Simulaatioajon jälkeen oikeaan ikkunan osaan listautuvat tulokset ovat jaoteltu seuraaviin osioihin:

- **Yleisessä dataosiossa** kuvataan, mitä simulaatioajossa tapahtui korkealla tasolla. Ilmoitetaan simulaation suorittamat päivät, kuinka monta kertaa roskia on simulaatioajan aikana viety ja monta kertaa ne on roska-auton toimesta tyhjennetty. Ilmoitetaan myös yleinen roskakatokset käyttöaste. Sillä indikoidaan, kuinka täynnä keskimäärin roska-astiat olivat tyhjennyksen yhteydessä.

(25)



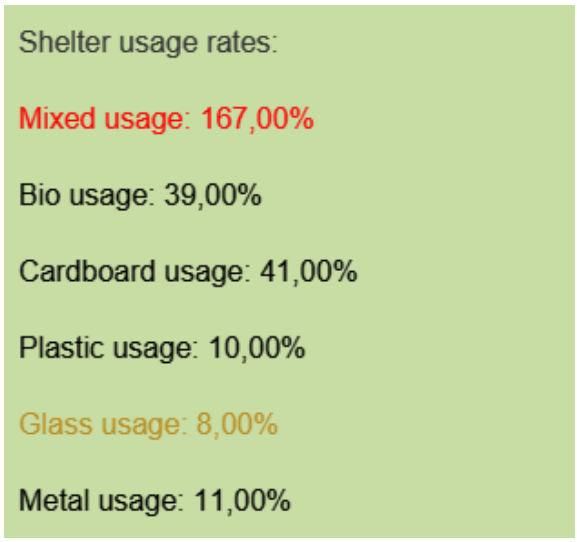
Kuva 13. Yleinen data

- **Heitetty roskaosio** listaa roskat tyypeittäin, sekä yleisesti kuinka paljon niitä kertyi simulaation aikana. Se kertoo, kuinka paljon roskia on heitetty kiloissa, sekä litroissa. Osio erittelee vielä jokaisen roska-astia tyypin erikseen ja ilmoittaa niiden kertyneen roskamäärän kilogrammoissa.

Kuva 14. Heitetty roska

(25)

- **Roskakatoksen käyttöasteosiossa** kuvataan roska-astia tyyppien käyttöasteita, toisin sanoen kuinka täynnä astiat ovat keskimääräisesti roska-auton tyhjennyksen yhteydessä.



Kuva 15. Roskakatoksen käyttöasteet. Huomaamme, että sekajätteen käyttöaste on liian suuri, mutta muiden astioiden käyttöaste on matalampi. Tätä voitaisiin mahdollisesti optimoida kasvattamalla sekajäte astioiden määriä ja harventamalla roska-auton käyntiväliä.

- **Ylivuotavan roskan osiossa** on listattuna roskamäärät kiloina, jotka eivät mahtuneet roska-astioihin, koska ne olivat liian täynnä.

(25)

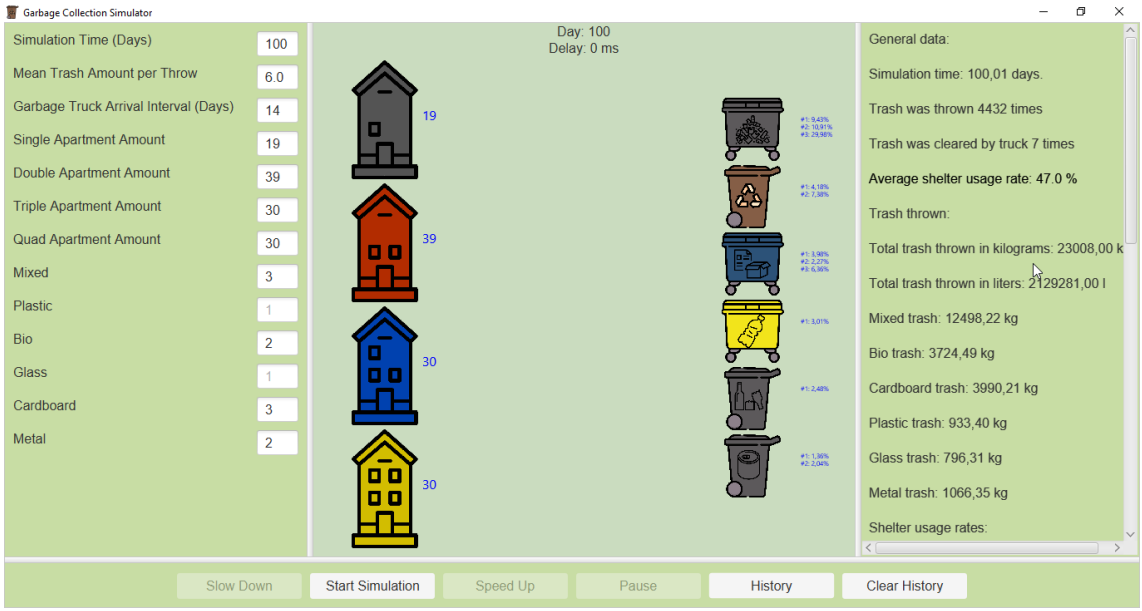
Kuva 16. Ylivuotaneiden roskien data. Tässä esimerkissä sekajätettä on kertynyt enemmän, kun kapasiteetti on sallinut. Voimme kuvitella tämän roskan olevan esimerkiksi kertynyt roskakatoksen lattialle.

- **Saavutettavuus osiossa** kuvataan sitä, kuinka pitkän aikaa roska-astiat ovat saavuttamattomissa, kun ne ovat täynnä.

Kuva 17. Saavutettavuus ajat. Sekajäte on ollut koko simulaation ajan noin 4.7 päivää täynnä.

7 Tehdyt simulointikokeet

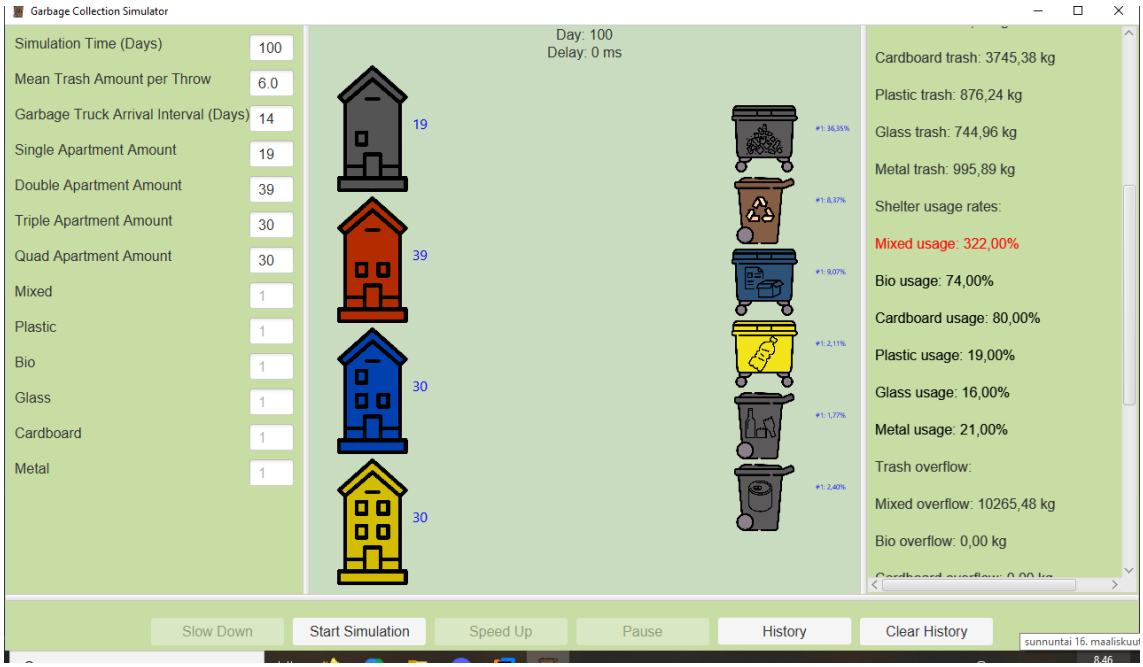
Projektiryhmän jäsen mallinsi simulaattorilla entistä vuokra-asuntoaan käyttäen tutkimustietoa asuintalojen asuntojakaumista. Kyseisessä talossa oli 117 asuntoa:



Kuva 18. reaalimaailman simulointia

Edellä kuvatussa simulaatioajossa roska-astioita on tarpeeksi, eikä roskien ylivuotoa tapahdu, vaikka roska-auto käy vain kerran kahdessa viikossa. Seuraavassa simulaatioajossa on pienennetty roska-astioiden määrää ja siinä huomataankin, että nyt astiat täyttyvät harmillisen nopeasti:

(25)



Kuva 19. vähempi määrä roska-astioita täyttyy nopeammin

Mitä enemmän ihmiset vievät roskia kerralla (mean trash throw), sen isommalla todennäköisyydellä tulee tilanteita, joissa roska-astioista loppuu tila:

Kuva 20. roskakatoksen käyttöasteet nousevat, kun heitetyn roskan määrä kasvaa.

Roska-auton käyntitiheys voisi olla muovi-, lasi- ja metalliroskalle harvempi kuin muille roskatyypeille, sillä näitä jätetyyppejä kertyy kotitalouksissa huomattavasti vähemmän. Tämä voi tietysti johtua myös kierrätyksen puutteesta.

Simulaatio vastaa jotakuinkin reaalimaailmassa observoitua roskakatoksen toimintaa, kun sen tuloksia vertaa omaan kokemukseen asumisesta.

8 Yhteenveto

Kaiken kaikkiaan ohjelmistoprojekti on onnistunut. Projektiryhmä sai toteutettua suunnittelemansa idean käytäntöön. Simulaattori on mielenkiintoinen ja sen tuottama tieto pohjautuu reaalimaailman tilastoihin, joten sen käyttöä voitaisiin hyödyntää kiinteistöjen roskakatos suunnittelussa.

Projektiryhmän ainoa huolenaihe projektia työstäessä oli se, että se käyttää kolmivaihe simulaatiomallia hyvin yksinkertaisesti hyödykseen, monimuotoisten tapahtumatyyppien puutteellisuuden vuoksi. Vaikka projektiryhmä tiedosti mahdollisen heikkouden projektin tavoitteiden kannalta, päätti se silti toteuttaa visionsa. Projektin reaalimaailman käytännöllisyys, sekä innovatiivisuus tuntui projektiryhmästä tärkeämmältä.

Vaikka projekti saatiin onnistuneesti toteutettua, siinä on edelleen parannettavaa, kuten koodin refaktorointi ja uusien ominaisuuksien lisääminen. Esimerkiksi ominaisuus, joka ottaisi huomioon myös huoneistojen poikkeuskäytökset edes jollain tasolla parantaisi varmasti simulaattorin käytettävyyttä reaalimaailmassa.

Liitteet

1 JavaDoc: https://users.metropolia.fi/~aurila/trash_simulator_javadoc/