Modeliranje, dizajn i implementacija objektno orijentiranog simulatora perilice rublja

Živko, Dominik

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

MODELIRANJE, DIZAJN I IMPLEMENTACIJA OBJEKTNO ORIJENTIRANOG SIMULATORA PERILICE RUBLJA

Diplomski rad

Dominik Živko

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 13.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Dominik Živko
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1260R, 06.10.2021.
OIB studenta:	12350081631
Mentor:	izv. prof. dr. sc. Zdravko Krpić
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Mirko Köhler
Član Povjerenstva 1:	izv. prof. dr. sc. Zdravko Krpić
Član Povjerenstva 2:	doc. dr. sc. Bruno Zorić
Naslov diplomskog rada:	Modeliranje, dizajn i implementacija objektno orijentiranog simulatora perilice rublja
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Pristupnik ovim diplomskim radom treba istražiti sustavan objektno orijentirani pristup izradi simulacija mehaničkih uređaja. U tematici pokriti izazove u navedenom području, poput odabira fizikalnih veličina, vjernost modela za određene pojave, usporediti postojeće primjere rješenja navedenog problema (barem 8 postojećih), predstaviti modele međuovisnosti fizikalnih veličina i dobre prakse u području.Istraženo demonstrirati na postupku izrade simulatora perilice rublja. Započeti definiranjem zahtjeva na programsko rješenje neki od standardiziranih metoda zapisa, ovisno o načinu izrade (slučajevi korištenja, dijagrami aktivnosti, korisničke priče, scenariji i sl.). Nakon toga predstaviti najvažnije tehnologije pogodne za implementaciju programskog
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	13.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTIRADA

Osijek, 07.10.2023.

Ime i prezime studenta:	Dominik Živko
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1260R, 06.10.2021.
Turnitin podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Modeliranje, dizajn i implementacija objektno orijentiranog simulatora perilice rublja**

izrađen pod vodstvom mentora izv. prof. dr. sc. Zdravko Krpić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

1. UVOD

Objektno orijentirana (OO) paradigma je programska paradigma koja se zasniva na konceptu objekta, pojma u obliku samostojeće cjeline s poznatim stanjem i ponašanjem. Složena ponašanja u nekom OO sustavu proizlaze iz komunikacije između objekata, dok je svaki pojedinačni objekt jednostavan i lako razumljiv u izolaciji. Iz tog razloga OO paradigma je izrazito pogodna u području programiranja i tada se govori o objektno orijentiranom programiranju (OOP). Najpoznatiji moderni OO programski jezici su jezici opće namjene poput C# i Jave koji pružaju veliku fleksibilnost u izradi bilo kakvih programskih paketa. Jedna od primjena za koju su OO programski jezici pogodni je simulacija, naročito jer omogućuju modeliranje složenih pojava tako što omogućuju rukovanje informacijama na intuitivan način.

U ovom se diplomskom radu na primjeru perilice rublja pokazuje primjena objektno orijentiranog pristupa izradi simulacije. Aplikacija koja je rezultat ovog rada putem grafičkog sučelja omogućuje korisniku upravljanje simulacijom moderne kućne perilice rublja te praćenje njezinog unutarnjeg stanja, kao i stanja rublja tijekom pranja kroz podatke o vlažnosti, količini i vrsti nečistoća, i slično. Navedeni model perilice ima sve ključne funkcionalnosti koje su očekivane od suvremene električne perilice s prednjim punjenjem, kao što je skup programa pranja s mogućnošću postavljanja željene temperature i brzine vrtnje u fazi centrifuge.

Osim ranije navedenog, u ovom će radu biti pokazano kako koncepti OO paradigme, poput klasa i sučelja, mogu predstavljati sve važnije gradivne sklopovske i upravljačke programske elemente simuliranog uređaja. Prema tome, perilica rublja simulirana u okviru ovog rada se sastoji od zasebno simuliranih funkcionalnih komponenti.

U narednom poglavlju opisuje se OO pristup simulaciji uz različite razine detalja, nakon čega se fokus usmjerava na simulaciju uređaja gdje se provodi kratak pregled područja kroz nekoliko primjera. U trećem se poglavlju definiraju zahtjevi na programsko rješenje - simulator, dok se u četvrtom poglavlju obrazlažu tehnologije odabrane za izradu programskog rješenja te se opisuje arhitektura sustava. Upotreba simulatora demonstrira se u posljednjem poglavlju te se analiziraju rezultati simulacije pranja.

2. OBJEKTNO ORIJENTIRANA SIMULACIJA

U području modeliranja i simulacije postoji više pristupa prevođenju koncepata iz stvarnog svijeta u digitalni, tj. računalni oblik pogodan za simulaciju. Prema [1], najistaknutiji pristupi modeliranju su oni temeljeni na konceptima događaja, aktivnosti, procesa, objekata i drugih. Fokus u ovom radu stavljen je na pristup temeljen na objektima - objektno orijentiranu simulaciju.

2.1. Objektno orijentirani pristup simulaciji

Ovakva vrsta simulacije je vrlo intuitivna pri razradi širokog spektra mogućih simulacija. Intuitivnost objektno orijentiranog dizajna proizlazi iz sličnosti s prirodom stvarnosti i načinom na koji ljudi koncipiraju svijet. Pojam objekta je primjenjiv ne samo na fizičke predmete, nego i na apstraktne koncepte i pojave kao što su događaji, virtualni zapisi ili matematički simboli, kako je spomenuto u [2].

Za razliku od proceduralnih programskih jezika koji se baziraju na funkcijama i algoritmima, objektno orijentirani jezici uglavnom podupiru komunikaciju između objekata koji odgovaraju konceptima iz stvarnosti, više u [2,3]. Objekti posjeduju unutarnje stanje koje se može mijenjati pozivima izloženih funkcija, tzv. metoda. Objekti su proširivi, ponovno iskoristivi, te mogu postojati neovisno ili u ovisnosti jedni o drugima, često u hijerarhijskim strukturama. Sposobnost objekata da međusobno komuniciraju, i tako stvaraju složena ponašanja, otvara brojne mogućnosti u bilo kojem području primjene, a pogotovo u području simulacije. Ovaj pristup dizajnu simulacije daje veliku fleksibilnost i mogućnost izrade naprednih modela širokog opsega.

Nedostatak objektno orijentiranog pristupa simulaciji je u tome što nije uvijek očito kakav dizajn objekata i njihovih sučelja je najprikladniji za danu svrhu. Ista pojava se može opisati na više načina gdje neki od njih mogu imati skrivene nedostatke koje je teško unaprijed otkriti. Problem skalabilnosti je također istaknut u ovom pristupu jer svaki objekt sa sobom nosi određeni memorijski i procesorski trošak koji nije vezan uz samu svrhu onoga što on predstavlja. Zbog toga, iznimno velike količine objekata koje istovremeno sudjeluju u simulaciji mogu uzrokovati lošije performanse od modela koji koristi neki drugi pristup. Kako predlaže [4], potencijalno rješenje ovog problema može biti podjela objektnog prostora u dijelove koji se mogu obrađivati paralelno, za što je opet objektno orijentirani pristup pogodan.

Programski jezici kojima se mogu izrađivati simulacije mogu biti specijalizirani jezici za izradu simulacija, bilo u određenom znanstvenom području, ili simulacija općenito. Takvi jezici

često imaju formu programskih paketa s alatima za jednostavnije provođenje čestih operacija i postupaka vezanih za simulaciju. Obično se baziraju na, ili su inspirirani, programskim jezicima opće namjene te pružaju skup često potrebnih i detaljno razrađenih specijaliziranih funkcija koje olakšavaju izradu određenih tipova simulacija, vidljivo u [5].

Simulacije se mogu izrađivati i korištenjem jezika opće namjene, što daje veću fleksibilnost. Ovisno o zahtjevima na efikasnost izvođenja, mogu se koristiti razni jezici, od onih niske razine poput C i C++, do onih s automatski upravljanom memorijom kao što su Java i Python. Prva spomenuta skupina jezika programeru dozvoljava preciznu alokaciju i dealokaciju memorije što otvara mogućnosti optimizacije u pogledu zauzeća memorijskog prostora i efikasnosti izvođenja simulacije. Nedostatak ovdje predstavlja činjenica da je složenost pisanja koda uvećana jer programeri moraju brinuti o pitanjima koja nisu izravno vezana za zadatak koji se rješava, poput pravilnog zauzimanja i oslobađanja memorije. Jezici koji samostalno brinu o održavanju memorije uklanjaju ovaj nedostatak što olakšava koncentraciju na problem simulacije, ali ujedno povećava rizik smanjenja efikasnosti njenog izvršavanja. Ako zahtjevi na učinkovitost i memorijski otisak simulacije nisu strogi, poželjno je koristiti jezik koji uklanja spomenute sporedne mogućnosti kontrole i tako programerima olakšava fokus na rješavanje središnjeg zadatka.

2.2. Razine detalja simulacija

Pri izradi simulacije u bilo kojem području potrebno je definirati odgovarajuću razinu detalja. Raspon mogućih razina detalja (apstrakcije) modela i procesa je širok i može imati velik značaj za složenost modela i jednostavnost izrade simulacije. Ovisno o potrebama, modeli mogu biti jednostavni i opisivati samo osnovne koncepte simuliranih pojava. To su modeli visoke razine i relativno su jednostavni za razradu i implementaciju, a fokus im je na opisivanju apstraktnih pojava vezanih za srž onoga što se simulira, više nego na opisivanju pojava koje omogućuju to specifično ponašanje.

Kako se tema ovog rada odnosi na simulaciju uređaja, govori se o rasprostranjenim kućanskim aparatima poput perilice rublja ili uređaja koji imaju sličan princip rada i uporabe. Na primjeru perilice rublja, visoka razina apstrakcije odnosi se na funkcije koje su specifične za samu perilicu, ne uzimajući pritom u obzir fizikalne pojave koje bi se morale odvijati kako bi opisivano ponašanje bilo moguće. Fizikalne pojave i njihovi rezultati se u takvim modelima pojednostavljuju, pretpostavljaju, ili zanemaruju kako se ne bi trošili resursi na ono što nije središnji predmet razmatranja. Kako se ide prema nižim razinama apstrakcije, opisivanje sve

generalnijih pojava poprima sve veći značaj. Fizikalne pojave koje omogućuju naprednija ponašanja počinju se precizirati, a model postaje složeniji. U konačnici je važno odabrati razinu koja najbolje odgovara onome što se želi naučiti iz simulacije i ne trošiti resurse (i uvoditi potencijalne izvore grešaka) na izradu elemenata sustava koji se bez posljedica mogu zamijeniti pojednostavljenim reprezentacijama istih pojava.

2.3. Simulacija uređaja

Postoje razne svrhe simuliranja uređaja. One mogu biti analiza rada, utvrđivanje potrošnje energije, pronalaženje kvarova i slabih točaka itd. Simulacije također mogu biti rezultat istraživanja u području izrade istih, kao i demonstracija mogućnosti objektno orijentiranih jezika u određenom području interesa. Velik broj javno dostupnih simulacija kućanskih uređaja i sličnih tehnologija kao glavnu svrhu imaju demonstraciju primjene OOP koncepata na predmete iz stvarnog svijeta. Kućanski uređaji su pogodan primjer za izradu takvih simulacija jer se često sastoje od više gradivnih komponenti koje mogu samostalno funkcionirati i obavljati svoju svrhu, ali i povezivati se u naprednije cjeline koje zajedno postižu viši cilj. OO paradigma iznimno dobro odgovara ovom opisu po tome što spomenute gradivne komponente mogu biti predstavljene sučeljima koja definiraju tražena ponašanja, i objektima koji definirana sučelja implementiraju na jedan ili više načina. Ovisno o odabranoj razini apstrakcije, ti objekti se mogu i sami sastojati od drugih objekata niže razine (npr. tranzistori i druge komponente) i tako opisivati složenija ponašanja.

U nastavku slijedi kratak opis nekoliko primjera postojećih projekata koji nastoje simulirati tematiku srodnu ovom radu.

Perilica rublja

Projekt opisan u [6,7] je osmišljen kao zadatak za vježbu objektno orijentiranog pristupa programiranju. Sadrži početnu postavu simulacije sa zahtjevima na dovršenje, kao i jedno moguće rješenje zadatka. Projekt je popraćen objavom na blogu koja objašnjava propuste koji se pojavljuju kod lošeg objektno orijentiranog dizajna i daje pravilan oblik rješenja koji je razumljiv, proširiv, lako izmjenjiv i testabilan.

Rješenje zadatka je u programskom jeziku Java i koristi apstraktne klase, nasljeđivanje i polimorfizam kako bi se na generaliziran način izradila simulacija pranja više vrsta odjeće uz različite opcije programa pranja. Razina apstrakcije je visoka jer se ne opisuju nikakva fizička obilježja same perilice nego je fokus na opcijama programa pranja i vrstama odjeće. Projekt koji

se izrađuje u ovom radu je tematski sličan ovom projektu, a nastoji ga unaprijediti proširenjem opsega i nižom razinom detalja.

Aparat za kavu

U [8] autor opisuje Java model aparata za kavu koji se sastoji od komponenti za pohranu i komponenti za obradu podataka. Komponente su skrivene iza objekata fasade (engl. *facade*) prema istoimenom oblikovnom obrascu, a obrada namirnica postiže se obrascem posjetitelja (engl. *Visitor*). Postoji nekoliko recepata koji predstavljaju vrste napitaka koje aparat može proizvesti. Kroz metode posjetitelja i fasade, elementi recepta se pretvaraju u obrađene namirnice koje zajedno čine traženi napitak. Ovaj sustav simulira uređaje čija je struktura rastavljena na komponente što je u skladu sa simulatorom koji se izrađuje u ovom radu. Koncepti iz navedenog primjera se u ovom radu nastoje dublje istražiti i predstaviti na nižoj razini detalja.

Pametni dom

U [9] je definiran model pametnog doma u programskom jeziku C#. Model omogućuje daljinsko upravljanje raznim električnim uređajima poput žarulje, automatiziranog prozora i klima uređaja. Sustav podržava kuće koje mogu imati više katova od kojih se svaki sastoji od soba unutar kojih su postavljeni uređaji. Uređaji mogu imati više mogućnosti i mogu se međusobno povezivati. Modelirana je komunikacija između uređaja, udaljena i ručna kontrola, sigurnosna obilježja i slično. Koristi se klijent-server arhitektura koja omogućava zamjenu simuliranog modela doma s onim stvarnim. Ovaj primjer se fokusira na mogućnosti upravljanja uređajima, dok je ovaj rad usmjeren na samu simulaciju konkretnog uređaja što je također prisutno u navedenom primjeru, ali na pojednostavljen način.

Energetska potrošnja pametnog doma

U [10] je dostupan rezultat projektnog zadatka čija je svrha istaknuti načine korištenja OOP principa izradom programa koji izvršava složen zadatak korištenjem polimorfizma u programskom jeziku Java na jasan i razumljiv način. Projekt simulira proizvodnju i potrošnju električne energije, te potrošnju vode u kućanstvu koje koristi pametne tehnologije. Modelirani su uređaji koji troše, generiraju ili pohranjuju energiju, kao i uređaji koji troše vodu. Uz njih su modelirana i mjerila potrošnje koja iz sata u sat prate promjene i daju izvještaje o troškovima. Način predstavljanja koncepata iz stvarnosti u ovom primjeru je pogodan za simuliranje uređaja te je sličan onome koji se pojavljuje u ovom radu.

Navedeni primjeri imaju različite razine detalja, razloge postojanja i kontekste nastajanja, ali se svi obaziru na principe iza kojih stoji objektno orijentirana simulacija. Projekt kojim se bavi ovaj rad ima slična obilježja koja nastoji unaprijediti i detaljnije obraditi.

Ostatak rada se bavi opisom i izradom konkretnog simulatora perilice rublja. Sljedeće poglavlje opisuje specifičnosti rađene simulacije i definira zahtjeve na željeni rezultat.

3. RAZRADA SIMULACIJE

Za dosljednu provedbu simulacije potrebno je napisati specifikaciju zahtjeva na programsko rješenje - simulator. Zahtjevi su rastavljeni na one koji opisuju simulaciju i na one koji opisuju korisničko sučelje koje omogućuje interakciju sa sustavom i pregled stanja simulatora.

3.1. Zahtjevi na simulaciju

Zahtjevi su podijeljeni na glavne koncepte poredane prema razini ovisnosti, tj. od onih s najmanje ovisnosti do onih s najviše. Glavni zahtjevi se dijele na podzahtjeve koji ih pobliže opisuju. Složeniji podzahtjevi se dodatno dijele na još detaljnije uvjete koji su potrebni za potpuno ispunjavanje zahtjeva. Ovi zahtjevi će biti pobliže objašnjeni konceptualnim dijagramima nakon izlistanja.

U nastavku slijedi izlistanje zahtjeva na simulaciju:

1. U sustavu moraju postojati pojmovi fizikalnih veličina: volumen, energija, temperatura, vrijeme, snaga, protok, frekvencija, brzina vrtnje (broj okretaja u jedinici vremena).
- **1.1.** Fizikalne veličine moraju biti predstavljene odgovarajućim mjernim jedinicama.
- **1.2.** Volumen se predstavlja samo numerički - simulacija ne mora imati koncept vidljivog prostora.

2. U sustavu mora postojati koncept tvari (engl. *substance*).
- **2.1.** Elementarne tvari moraju imati određeni tip, uključujući vodu, kavu, blato i, po potrebi, druge česte oblike nečistoća koje se pojavljuju na odjeći.
  - **2.1.1.** Tipovi tvari moraju imati svojstvo tvrdokornosti, tj. težine uklanjanja iz tkanine prilikom pranja.
  - **2.1.2.** Tipovi tvari moraju imati svojstvo snage čišćenja, tj. utjecaja na mogućnost uklanjanja drugih tvari iz tkanine koja se pere.
  - **2.1.3.** Tipovi tvari moraju imati svojstvo svježine, tj. razinu dojma svježine kakvog tkanini daje omekšivač nakon pranja (otvoreno za slobodnu interpretaciju).
  - **2.1.4.** Voda mora imati sposobnost isparavanja u dopuštajućim uvjetima (izvan perilice).
- **2.2.** Nakupina tvari mora imati količinu mjerenu u jedinici volumena.
- **2.3.** Nakupine tvari se moraju moći sastojati od više različitih elementarnih tvari u različitim omjerima (npr. mješavina od 10 mL kave i 100 mL vode).

2.4. Tvari se moraju moći miješati s drugim tvarima i razdvajati na više dijelova.

2.5. Tvari moraju imati svojstvo topline koje utječe na učinkovitost pranja.

3. U sustavu mora postojati koncept elektriciteta, odnosno električne energije.

3.1. Nakupina elektriciteta mora imati količinu mjerenu u jedinici energije (džul).

3.2. Nakupina elektriciteta se mora moći dodavati drugim dijelovima i rastavljati na više dijelova.

4. U sustavu mora postojati koncept toka tvari i elektriciteta (odnosno toka količine općenito).

4.1. Dijelovi protočnih tvari se moraju moći spajati i rastavljati prema količini koju predstavljaju.

4.2. Tok mora biti predstavljen količinom u jedinici vremena.

4.3. Tok se mora moći odvijati barem u jednom smjeru.

4.4. Protočne tvari se moraju moći vući iz izvora toka i gurati u odljev toka.

4.5. Izvori i odljevi toka se moraju moći međusobno spajati u spojevima ‘više na više’ i omogućavati lančani prijenos protočnih tvari.

5. U sustavu mora postojati koncept tijela (fizičkog objekta).

5.1. Tijelo mora imati definiran volumen.

5.2. Tijelo mora dopustiti dodavanje stranih tvari (prljanje), kao i uklanjanje tih tvari (čišćenje).

5.3. Tijelo mora omogućiti izjednačavanje temperature tvari koje sadrži s temperaturom trenutne okoline.

6. U sustavu mora postojati koncept upijajućeg tijela.

6.1. Upijajuća tijela su tijela - moraju ispunjavati zahtjeve pod točkom 5.

6.2. Upijajuća tijela moraju omogućiti upijanje stranih tvari, kao i uklanjanje tih tvari (sušenje).

6.3. Upijajuća tijela moraju omogućiti izjednačavanje temperature upijene tvari s temperaturom trenutne okoline.

7. U sustavu mora postojati koncept tkanine.

7.1. Tkanina je upijajuće tijelo - mora ispunjavati zahtjeve pod točkom 6.

8. U sustavu mora postojati koncept odjeće.

8.1. Odjeća je oblik tkanine - mora ispunjavati zahtjeve pod točkom 7.

8.2. Moraju postojati barem dva različita tipa odjeće (npr. košulja, hlače).

9. U sustavu mora postojati deterdžent za pranje (tip tvari - zahtjev 2.1.), i to u nekoliko vrsta gdje svaka ima različito svojstvo snage čišćenja.

10. U sustavu mora postojati omekšivač rublja (tip tvari - zahtjev 2.1.), i to u nekoliko vrsta gdje svaka ima različito svojstvo svježine.

11. U sustavu mora postojati koncept vrtnje oko osi.

11.1. Vrtnja mora biti moguća određenom brzinom (broj okretaja u jedinici vremena) uz određeno trajanje.

11.2. Vrtnja mora biti moguća u oba smjera.

12. U sustavu mora postojati koncept električnog uređaja.

12.1. Električni uređaj mora imati nazivnu snagu.

12.2. Električni uređaj mora imati način primanja električne energije koju koristi za rad.

13. U sustavu mora postojati električni grijač.

13.1. Električni grijač, kao električni uređaj (točka 12.), mora moći korištenjem električne energije povećavati temperaturu zadane tvari. Zbog jednostavnosti, zagrijavanje svih tvari se može odvijati na isti način: korištenjem 1 cal za zagrijavanje 1 mL tvari za 1 °C.

13.2. Električni grijač mora imati podesivu snagu, od nule do nazivne snage. Odabrana snaga određuje brzinu zagrijavanja tvari i potrošnju energije.

13.3. Električni grijač mora imati uključeno i isključeno stanje.

14. U sustavu mora postojati električni motor (elektromotor).

14.1. Električni motor, kao električni uređaj (točka 12.), mora moći korištenjem električne energije uzrokovati vrtnju zadanog tereta koji to dozvoljava, tj. koji ispunjava zahtjeve pod točkom 11.

14.2. Električni motor mora imati podesivu brzinu vrtnje, od nule do maksimalne brzine dodijeljene tom motoru. Potrošnja energije je proporcionalna postavci brzine vrtnje.

14.3. Električni motor mora omogućiti odabir smjera vrtnje.

14.4. Električni motor mora imati uključeno i isključeno stanje.

15. U sustavu mora postojati ventil kao element sustava za kontrolu prijenosa tvari.

15.1. Ventil mora imati sposobnost kontrole brzine toka, od potpunog zaustavljanja do određene maksimalne vrijednosti toka.

16. U sustavu mora postojati rezervoar kao element sustava za kontrolu prijenosa tvari.

16.1. Rezervoar mora imati određen kapacitet pohrane volumena tvari.

16.2. Rezervoar mora imati postavljenu razinu unutar granica kapaciteta na kojoj je moguć odljev tvari. Ako je u rezervoaru pohranjena razina tvari ispod te razine, odljev može i ne mora biti moguć.

16.3. Rezervoar mora podržavati spontani odljev pohranjene tvari koji može biti omogućen po potrebi (istjecanje zbog gravitacije).

17. U sustavu mora postojati električna pumpa kao element sustava za kontrolu prijenosa tvari.

17.1. Električna pumpa, kao električni uređaj (točka 12.), mora moći korištenjem električne energije prenositi protočnu tvar iz spojenog izvora u spojeni odljev i tako inducirati tok.

17.2. Električna pumpa mora imati podesivu snagu, od nule do nazivne snage. Odabrana snaga mora određivati brzinu induciranog toka i potrošnju energije.

17.3. Električna pumpa mora imati uključeno i isključeno stanje.

17.3.1. U isključenom stanju može djelovati ili kao zatvoreni ventil (ne propuštati tok) ili kao obični cjevovod (bez utjecaja na tok).

17.3.2. U uključenom stanju, ako ima dovoljnu količinu energije za rad, mora inducirati tok u linearnoj ovisnosti o odabranoj snazi rada, od nule do maksimalnog protoka određenog kao svojstvo protočnog elementa.

17.3.3. U uključenom stanju, ako nema dovoljnu količinu energije za rad, može ili raditi smanjenim performansama, ili ne raditi uopće i ponašati se u skladu sa zahtjevom 17.3.1.

18. U sustavu mora postojati koncept perilice, tj. uređaja namijenjenog pranju.

18.1. Perilica mora moći sadržavati bilo kakva tijela.

18.2. Perilica mora imati stanje uključenosti i isključenosti (engl. *running/not running*).

19. U sustavu mora postojati barem jedna instanca perilice rublja.

19.1. Perilica rublja je vrsta perilice - mora ispunjavati zahtjeve pod točkom 18.

19.2. Perilica rublja je električni uređaj - mora ispunjavati zahtjeve pod točkom 12.

19.3. Perilica rublja mora imati vanjski izvor i odljev za prijenos tvari.

19.4. Perilica rublja mora imati dispenzer sredstva za pranje.

19.4.1. Dispenzer mora imati vanjski izvor i odljev za prijenos tvari.

19.4.2. Dispenzer mora imati jedan ili više kanala koji omogućuju zaseban protok tvari odnosno punjenje perilice sredstvom za pranje.

19.4.3. Dispenzer mora imati otvorivu ladicu s jednim ili više odjeljaka za doziranje tvari (sredstva za pranje) određenih kapaciteta.

19.4.4. Kada je ladica dispenzera zatvorena, protok tvari pri punjenju perilice treba ispirati sadržaj odjeljaka ladice.

19.4.5. Kada je ladica dispenzera otvorena, protok tvari pri punjenju perilice ne smije ispirati sadržaj odjeljaka ladice - ladica ne sudjeluje u protoku.

19.5. Perilica rublja mora imati bubanj za pranje.

19.5.1. Bubanj mora imati sposobnost vrtnje - mora ispunjavati zahtjeve pod točkom 11.

19.5.2. Bubanj mora moći sadržavati tijela i tvari do određenog kapaciteta.

19.5.3. Bubanj mora omogućiti protok tvari.

19.5.4. Bubanj mora omogućiti instalaciju grijača koji će zagrijavati sadržane tvari.

19.5.5. Prilikom vrtnje, bubanj mora uzrokovati upijanje tvari u upijajuća tijela, tj. u objekte koji ispunjavaju zahtjeve pod točkom 6.

19.5.6. Prilikom vrtnje, bubanj mora uzrokovati uklanjanje ili dodavanje mrlja na sadržana tijela (čišćenje/prljanje) ovisno o brzini vrtnje, te količini, sastavu i svojstvima tvari koje se nalaze u bubnju.

19.5.7. Ako se vrtnja odvija brže od neke određene granične brzine centrifuge, proces opisan točkom 19.5.6. mora prestati, te se umjesto toga mora odvijati postepeno uklanjanje upijenih tvari iz upijajućih tijela (sušenje), u ovisnosti o brzini vrtnje.

19.6. Perilica rublja mora osigurati zaključavanje i otključavanje vrata. Zaključano stanje onemogućuje dodavanje i vađenje tijela iz bubnja.

19.7. Perilica rublja mora imati motor za vrtnju bubnja (točka 14.).

19.8. Perilica rublja mora imati pumpu za izbacivanje vode iz bubnja (točka 17.).

19.9. Perilica rublja mora imati kontroler (upravljačku jedinicu).

19.9.1. Kontroler je električni uređaj - mora ispunjavati zahtjeve pod točkom 12.

19.9.2. Kontroler mora omogućiti paljenje i gašenje perilice.

19.9.3. Kontroler mora definirati moguće programe pranja. Moraju postojati barem osnovni programi pranja: glavni program pranja s opcijom pretpranja, program ispiranja i osvježavanja, program centrifuge.

19.9.4. Kontroler mora omogućiti odabir željenog programa pranja, kao i pokretanje i zaustavljanje tog programa.

19.9.5. Kontroler mora pratiti stanje aktivnog programa pranja i omogućiti njegovo pauziranje.

19.9.6. Kontroler može za programe pranja omogućiti odabir između nekoliko postavki temperature i brzine centrifuge koje se mogu i ne moraju moći mijenjati tijekom aktivnosti programa.

19.10. Perilica rublja mora imati tipku za paljenje/gašenje.

19.10.1. Kada je perilica isključena, pritiskom tipke perilica se mora uključiti (ako ima dovoljno energije iz spojene električne mreže).

19.10.2. Kada je perilica uključena, bez aktivnog programa pranja, pritiskom tipke perilica se mora isključiti.

19.10.3. Kada je perilica uključena, i dok je program pranja aktivan, pritiskom tipke program se mora zaustaviti umjesto gašenja perilice.

19.11. Perilica rublja mora imati mogućnost odabira programa pranja.

19.11.1. U svakom trenutku mora biti odabran jedan i samo jedan program pranja.

19.12. Perilica rublja mora imati tipku za pokretanje/pauziranje programa pranja.

19.12.1. Kada nema aktivnog programa, pritiskom tipke se mora pokrenuti trenutno odabrani program.

19.12.2. Kada postoji aktivni program pranja, pritiskom tipke se on mora pauzirati.

19.12.3. Kada postoji aktivni, ali pauzirani program pranja, pritiskom tipke se mora nastaviti izvršavanje tog programa.

19.13. Perilica rublja mora imati tipku za aktivaciju faze pretpranja na programima koji ju podržavaju.

19.14. Perilica rublja mora imati skup tipki za povećanje i smanjenje temperature programa pranja i skup tipki za povećanje i smanjenje brzine centrifuge programa pranja. Tipke odgovaraju kontrolama opisanim u točki 19.9.6.

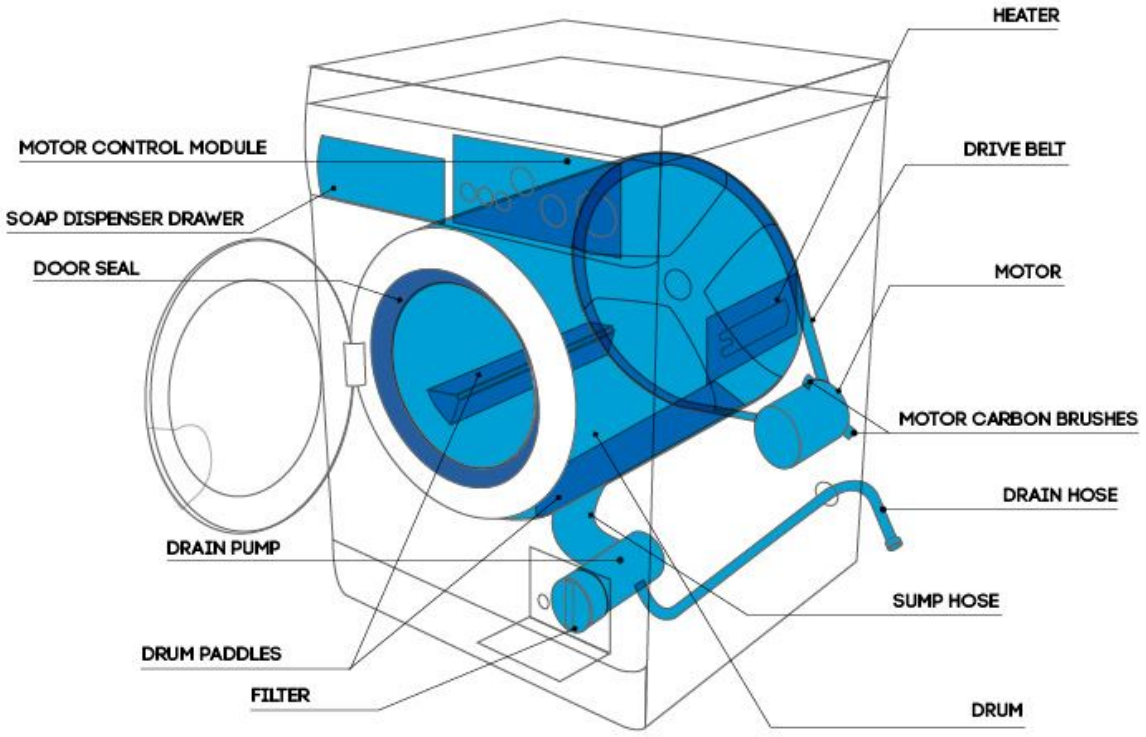
Ovim zahtjevima postavljena je osnova za ono što čini ovu simulaciju perilice rublja, ostavljajući pritom određenu količinu slobode što se tiče specifičnosti komponenti i interpretacije šireg značenja određenih pojmova.

U simulaciji nije predviđen koncept prostora prvenstveno zato što bi takva niža razina apstrakcije zahtijevala modeliranje pojava kao što je tlak. Modeliranje takvih pojava bi nepotrebno povećalo složenost simulacije, bez uvođenja dodatnih prednosti na izvođenje iste.

3.2. Konceptualni dizajn sustava

Iako zahtjevi na simulaciju ne definiraju striktno o kakvom tipu perilice se radi, fokus tijekom definicije zahtjeva i izrade simulacije je bio na perilici s prednjim punjenjem (za razliku

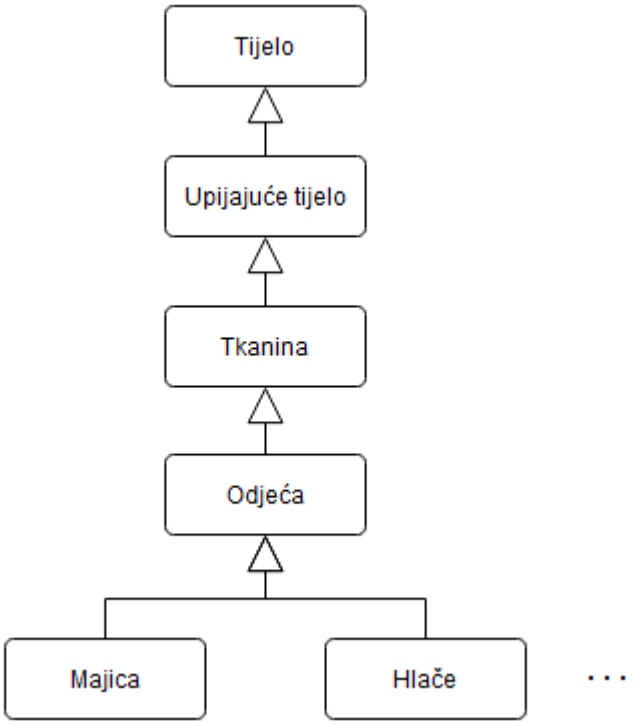
od perilice s gornjim punjenjem). Tip perilice je vidljiv na slici 3.1. gdje je prikazana konstrukcija perilice, odnosno raspored komponenti njenih komponenti. Pojmovi na slikama, dijagramima, kao i u programskom kodu implementacije, su svi na engleskom jeziku radi usklađenosti, lakšeg praćenja opisivanih koncepata i slijeđenja jezičnih standarda u području programiranja.



Slika 3.1. Konstrukcija perilice, raspored komponenti.

Izvor: ransomspares.co.uk

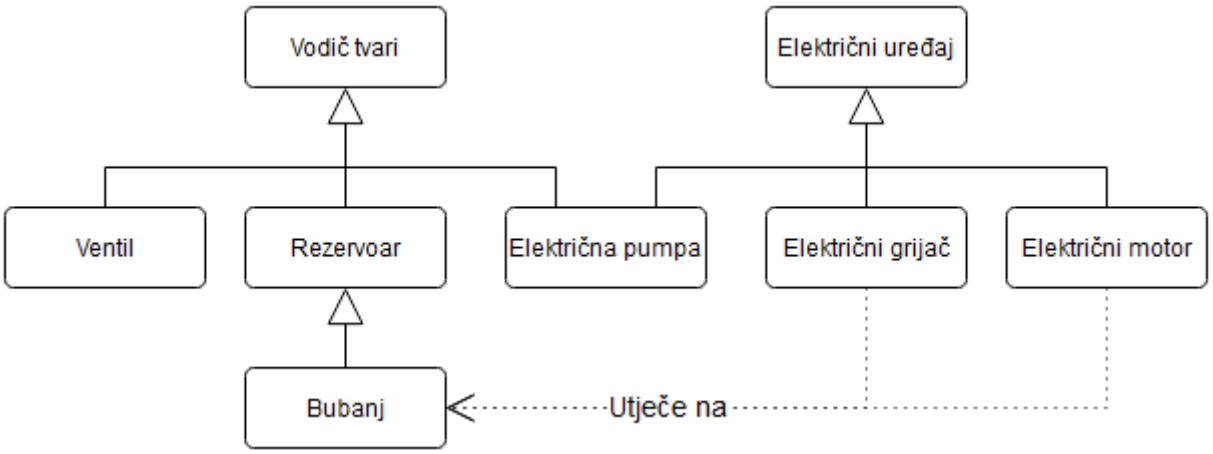
Radi lakšeg predočenja spomenutih koncepata, u nastavku su prikazani dijagrami odnosa između srodnih koncepata.



Slika 3.2. Hijerarhija pojma upijajućeg tijela i odjeće.

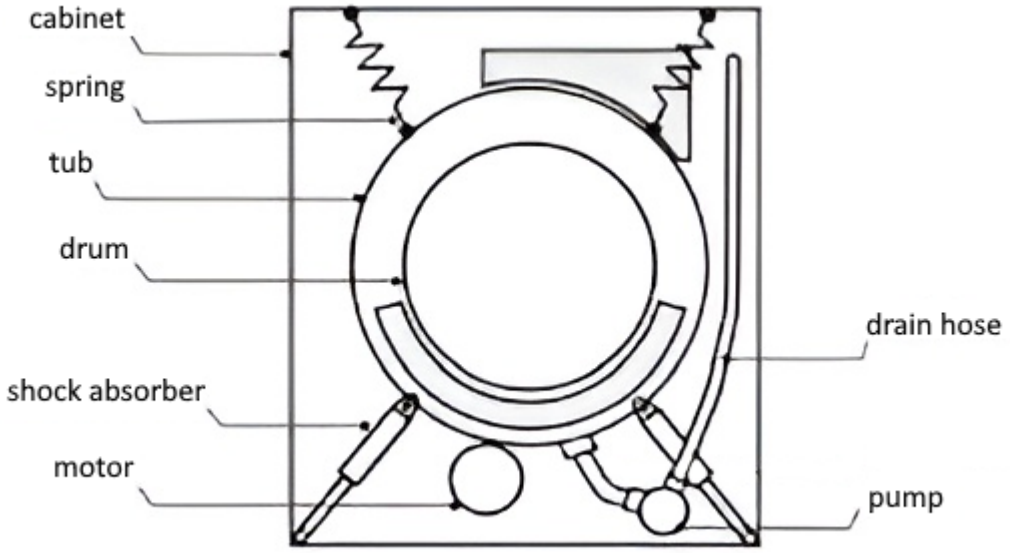
Kao što je spomenuto u zahtjevima, košulja i hlače su odjevni predmeti. Odjeća je ujedno i tkanina, a ona predstavlja oblik upijajućeg tijela. Ovaj odnos je prikazan dijagramom na slici 3.2. Razlika između pojmova odjeće, tkanine i upijajućeg tijela je napravljena kako bi se ostavila otvorenom mogućnost postojanja upijajućih tijela koja nisu tkanine, poput spužve, ili tkanina koje nisu odjeća, poput ručnika.

Na sličan način je dijagramom na slici 3.3. prikazan odnos između različitih vrsta protočnih elemenata za tvari i električnih uređaja. Ventil, rezervoar i električna pumpa su protočni elementi. Bubanj perilice je ujedno i rezervoar jer omogućava protok i pohranu tvari. Električni grijač, električni motor i ponovno električna pumpa su električni uređaji. Električna pumpa je posebna po tome što ima obilježja i protočnog elementa i električnog uređaja. Grijač i motor asocirani su s bubnjem jer mogu utjecati na njegovo stanje, tj. grijati tvari u njemu, odnosno okretati ga.



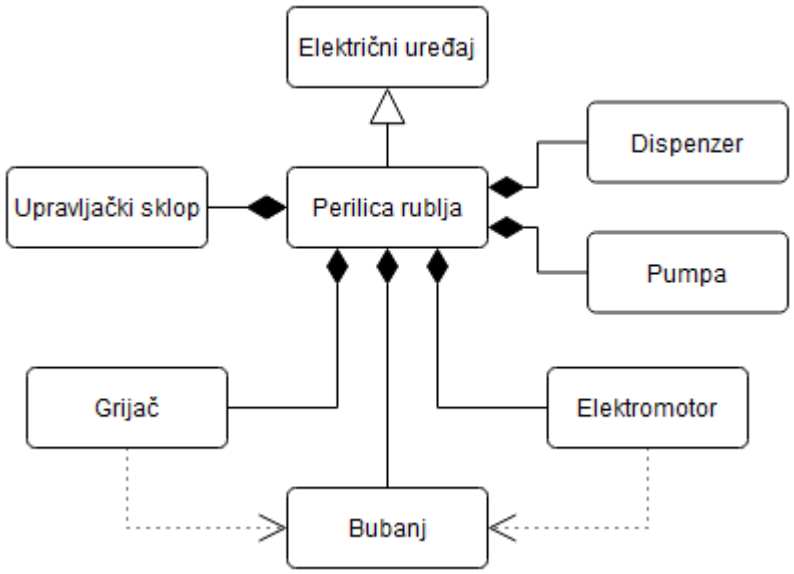
Slika 3.3. Hijerarhija i odnos između protočnih elemenata i električnih uređaja.

Ovdje je dobro napomenuti stvarnu razliku između kade perilice (engl. *tub*) i samog bubnja. Naime, perilica rublja ima kadu koja sadržava svu tekućinu tijekom pranja. Unutar kade je suspendiran bubanj. Samo bubanj se može okretati oko svoje osi, dok se kada ne okreće ali je amortizirana u odnosu na ostatak perilice. Građa bubnja je porozna i zbog toga propušta tekućinu za pranje iz kade u svoju unutrašnjost kako bi došla u kontakt s rubljem. Ovaj odnos kade i bubnja je prema zahtjevima ove simulacije konceptualno spojen u jedno pod nazivom ‘bubanj’. To znači da spomen pojma ‘bubanj’ ili ‘*drum*’ kroz opis simulacije, kao i u njezinoj tehničkoj izvedbi, zapravo opisuje jedinstvo bubnja i kade. Ovo funkcionira jer je simulacija izvedena na razini koja ne opisuje detaljna fizička odvijanja između kade i bubnja. Zbog ovoga se može reći kako grijač utječe na bubanj, iako bi se na nižoj razini detalja trebalo reći kako utječe na kadu koja zapravo sadrži tekućinu za pranje. Ova distinkcija (odnosno nedostatak iste) postaje jasnija kada se razmotre detalji implementacije bubnja kasnije u dokumentu. Na slici 3.4. je vidljiva razlika između kade i bubnja gledano sprijeda. Vidljivo je kako je kada amortizirana u tijelu perilice i kako su pozicionirane druge komponente perilice.



Slika 3.4. Raspored pokretljivih komponenti perilice.

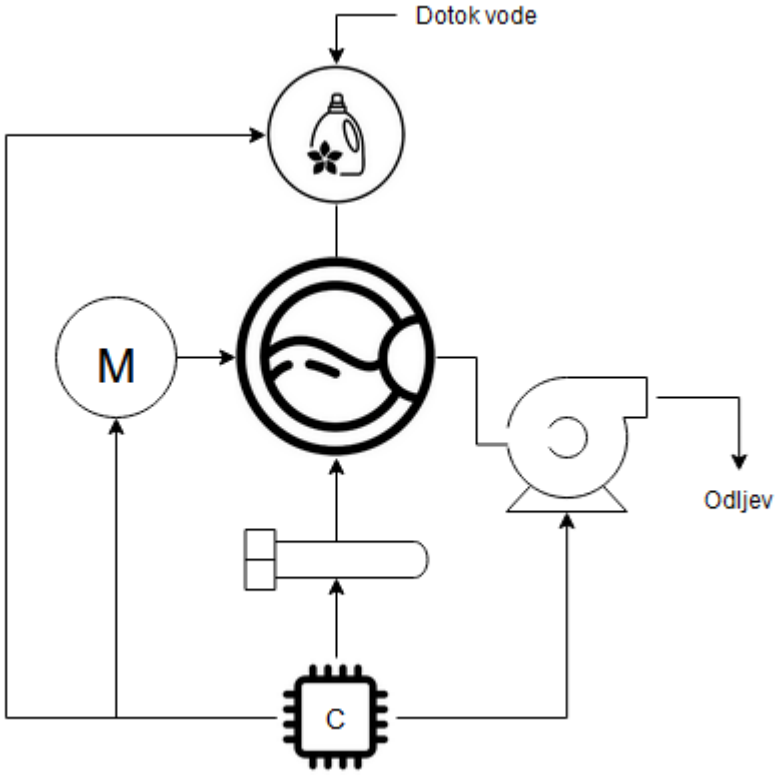
Izvor: J. Paris, B. S. Tabuenca, "Acoustic analysis of the drainage cycle in a washing machine" [Modificirano]



Slika 3.5. Konceptualni sastav perilice rublja.

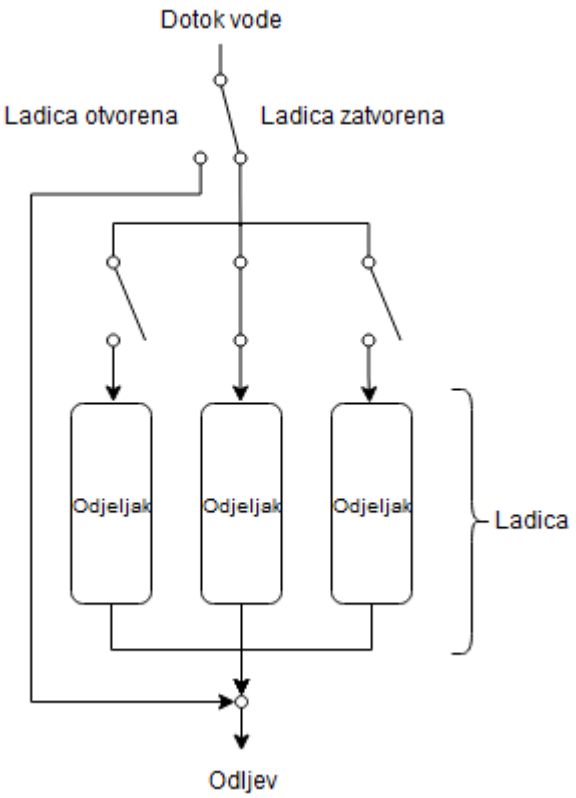
Slika 3.5. prikazuje konceptualni dijagram strukture perilice rublja u pogledu komponenti koje ju čine. Perilica se sastoji od slijedno povezanih komponenti toka - dispenzer, bubanj, električna pumpa. Kroz te komponente protječu tvari. Dispenzer je početna točka tog podsustava gdje se tvar koja dolazi izvana miješa sa sredstvima za pranje pohranjenim u ladici dispenzera. Izlaz dispenzera spojen je na bubanj koji u primljenoj tvari pere svoj sadržaj. Pumpa koja se nalazi na izlazu bubnja upravlja istjecanjem tvari iz perilice. Osim ovih komponenti

postoje elektromotor, grijač i kontroler. Elektromotor je zaslužan za vrtnju bubnja, grijač zagrijava tvari sadržane u bubnju, a kontroler upravlja radom perilice kroz programe pranja. Povezanost tih komponenti prikazana je dijagramom na slici 3.6. gdje je dispenzer označen simbolom sredstva za pranje, a bubanj simbolom vrata perilice. Element dijagrama označen slovom C predstavlja kontroler. Grijač, elektromotor i pumpa predstavljeni su standardnim simbolima pripadajućih komponenti.



Slika 3.6. Dijagram povezanosti komponenti perilice.

Dispenzer je komponenta koja igra važnu ulogu kada je riječ o protoku tvari u perilicu. Koncept protoka tvari kroz dispenzer ilustriran je dijagramom na slici 3.7. Dispenzer sadrži otvorivu ladicu koja u svakom trenutku može promijeniti protočni spoj perilice. Kada je ladica zatvorena, ona sudjeluje u protoku, što znači da se tvari koje se u njoj nalaze ispiru i postaju dio tvari koja ulazi u bubanj. U slučaju otvorene ladice, ulaz i izlaz dispenzera efektivno su kratko spojeni, te se sadržaj ladice ne ispire u bubanj. Dispenzer je ujedno podijeljen na kanale što omogućava postojanje više odjeljaka koji se mogu zasebno ispirati.



Slika 3.7. Protočni spoj dispenzera.

3.3. Zahtjevi na korisničko sučelje

Ovi zahtjevi dolaze kao nadopuna prethodnim zahtjevima, a određuju uvjete i način prikaza ranije opisanih koncepata. Ovi zahtjevi su podijeljeni u glavne skupine gdje se svaka odnosi na određeni dio korisničkog sučelja. Podzahtjevi pobliže opisuju glavni zahtjev pod kojim se nalaze. Neki od podzahtjeva se dodatno dijele na još preciznije uvjete koji su potrebni da bi se u potpunosti ispunili zahtjevi na višoj razini.

1. Korisničko sučelje mora prikazivati ilustraciju perilice rublja s prednjim punjenjem.

1.1. Perilica mora imati bubanj na kojemu je moguće vidjeti trenutnu brzinu i smjer vrtnje.

1.2. Perilica mora imati kontrolnu ploču s opcijama za upravljanje perilice.

1.2.1. Kontrolna ploča mora imati tipku za paljenje/gašenje perilice. Tipka odgovara funkcionalnosti opisanoj pod točkom 19.10. zahtjeva na simulaciju.

1.2.2. Kontrolna ploča mora imati gumb za odabir programa pranja. Gumb odgovara funkcionalnosti opisanoj pod točkom 19.11. zahtjeva na simulaciju.

1.2.3. Kontrolna ploča mora imati tipku za pokretanje/pauziranje programa. Tipka odgovara funkcionalnosti opisanoj pod točkom 19.12. zahtjeva na simulaciju.

1.2.4. Kontrolna ploča mora imati tipku za odabir opcije pretpranja na programima koji ga podržavaju. Tipka odgovara funkcionalnosti opisanoj pod točkom 19.13. zahtjeva na simulaciju.

1.2.5. Kontrolna ploča mora imati skup tipki za povećanje i smanjenje temperature programa pranja i skup tipki za povećanje i smanjenje brzine centrifuge programa pranja. Tipke odgovaraju funkcionalnostima opisanim pod točkom 19.14. zahtjeva na simulaciju.

1.3. Perilica mora imati zaslon s informacijama o programu pranja.

1.3.1. Zaslon mora prikazivati preostalo vrijeme tekućeg programa ako takav postoji, ili ukupno vrijeme trenutno odabranog programa ako nije aktivan niti jedan program.

1.3.2. Zaslon mora prikazivati odabrane postavke temperature i brzine vrtnje za trenutno aktivni program, ili odabrane postavke za trenutno odabrani program ako nije aktivan niti jedan program.

1.4. Perilica mora imati ladicu s dozatorom sredstva za pranje.

1.4.1. Ladica mora biti otvoriva na klik. Kada je otvorena, mora se prikazati prozor koji omogućava dodavanje i uklanjanje tvari iz odjeljaka ladice. Zatvaranjem prozora, ladica se zatvara.

2. Korisničko sučelje mora prikazivati informacije o odabranom programu pranja.

2.1. Informacije o programu pranja uključuju: naziv programa, ukupno trajanje programa, proteklo vrijeme od početka programa ako je on pokrenut, informacije o svim stadijima pranja (parametri vrtnje, trajanja, i sl.).

3. Korisničko sučelje mora prikazivati informacije o stanju bubnja i tvari unutar bubnja.

3.1. Informacije uključuju: brzina vrtnje bubnja, temperatura tvari unutar bubnja, količina i sastav tvari u bubnju.

4. Korisničko sučelje mora prikazivati rublje koje je izvan ili unutar perilice.

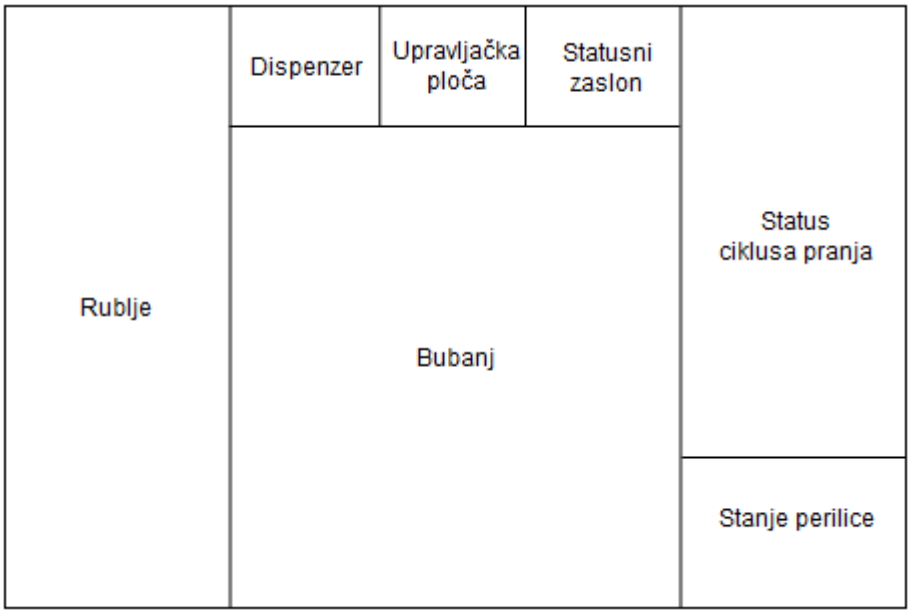
4.1. Rublje se mora moći prebacivati izvana u perilicu i obratno, ali samo ako su vrata perilice otključana.

4.2. Prezentacija rublja mora prikazivati količinu uprljanosti kao postotak volumena tog tijela.

4.3. Prezentacija rublja mora prikazivati količinu upijene tekućine kao postotak volumena tog tijela.

4.4. Prezentacija rublja mora prikazivati razinu svježine koja ovisi o količini i svježini umrljanih i upijenih tvari, i volumenu rublja.

Jedan način izvedbe korisničkog sučelja prikazan je nacrtom na slici 3.8. Središnji element je ilustracija perilice koja se sastoji od bubnja gdje je vidljiva vrtnja, dispenzera preko kojeg se dodaju sredstva za pranje, ploče za upravljanje perilicom, i statusnog zaslona. S obje strane perilice nalaze se informacijski paneli. Lijevi panel treba prikazivati stanje rublja i omogućavati punjenje perilice. Desni panel treba dati uvid u unutarnje stanje perilice kroz detalje o programu pranja i stanju komponenti.



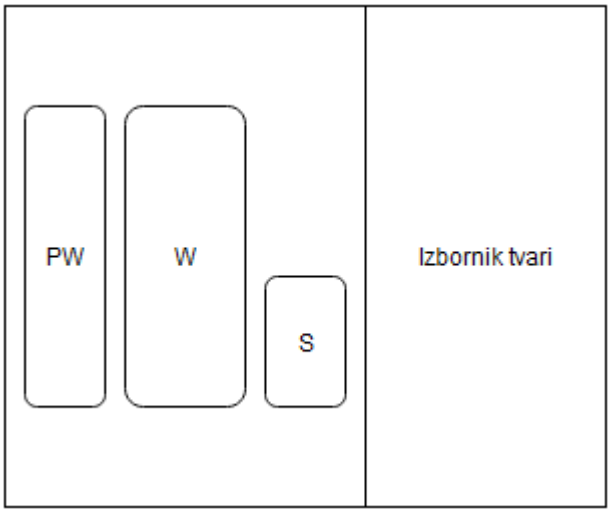
Slika 3.8. Nacrt glavnog prozora korisničkog sučelja.

Otvorena ladica dispenzera treba imati vlastiti prozor koji omogućava dodavanje i uklanjanje tvari, kako je prikazano nacrtom na slici 3.9.

Odjeljci ladice na nacrtu su označeni slovima na sljedeći način:

- PW (engl. *PreWash*) - odjeljak za deterdžent za pretpranje
- W (engl. *Wash*) - odjeljak za deterdžent za pranje
- S (engl. *Softener*) - odjeljak za omekšivač rublja

Panel s desne strane ladice treba omogućiti odabir tvari s kojima se odjeljci ladice mogu puniti.



Slika 3.9. Nacrt sučelja otvorene ladice dispenzera.

4. DIZAJN I IMPLEMENTACIJA SIMULATORA

U ovom poglavlju opisana je implementacija perilice, tj. prevođenje zahtjeva definiranih u prethodnom poglavlju u stvarni sustav. Najprije se obrazlažu korištene tehnologije, zatim se provodi arhitekturalni dizajn simulatora nakon čega slijedi detaljan prikaz načina rada simulacije uz konačnu implementaciju.

4.1. Korištene tehnologije

Za implementaciju simulatora korišten je objektno orijentiran programski jezik Kotlin[1]. Varijanta jezika korištena u ovom radu je bazirana na JVM tehnologiji, što znači da je interoperabilna s programskim jezikom Java. Prednost Kotlina je njegova fleksibilnost i sintaksa koja minimizira sporedne jezične konstrukte i olakšava fokus na domenu problema.

Kotlin također omogućava korištenje korutina[2] (engl. *coroutines*). Korutine dozvoljavaju pisanje funkcija čije se izvođenje može zaustaviti (suspendirati, engl. *suspend*) i nastaviti kasnije, čak i na drugoj niti izvođenja. Ovakve funkcije omogućavaju pisanje blokirajućeg odnosno asinkronog koda na neblokirajući način. Korutine uklanjaju potrebu za povratnim pozivima (engl. *callbacks*) kod asinkronih operacija jer se one promatraju kao da su sinkrone. Funkcija čije se izvođenje prekida ili traje dulje periode piše se jednako kao i klasična funkcija, što značajno olakšava pisanje složenih paralelnih operacija koje su česte u simulacijama.

Za izradu korisničkog sučelja korišten je Compose Multiplatform[3] okvir. Okvir je baziran na Jetpack Compose okviru za izradu nativnih sučelja za Android platformu, a proširuje ga tako što omogućava izradu sučelja za druge platforme poput iOS, desktop računala i web okruženja. U Compose-u se korisničko sučelje definira deklarativno, što znači da se prvenstveno opisuje struktura sučelja umjesto načina na koji se ono sastavlja. Ovakav pristup pojednostavljuje implementaciju sučelja i smanjuje greške pri razvoju.

Razvojno okruženje korišteno za implementaciju simulatora je IntelliJ IDEA[4]. To je napredno integrirano razvojno okruženje za rad s Java i Kotlin programskim jezicima koje pruža alate potrebne za iskorištavanje svih mogućnosti Kotlin platforme, poput *debuggera* i sintaktičkih prijedloga.

[1] Kotlin - https://kotlinlang.org/
[2] Kotlin Coroutines - https://kotlinlang.org/docs/coroutines-overview.html
[3] Compose Multiplatform - https://github.com/JetBrains/compose-multiplatform
[4] IntelliJ IDEA - https://www.jetbrains.com/idea/

Kao alat za upravljanje ovisnostima među klasama (engl. *dependency management*) korišten je Koin[5]. Koin pruža konstrukte za jednostavno ubrizgavanje ovisnosti u klase objekata. Koristan je kod izrade modularnih programskih rješenja čistom arhitekturom.

Osim navedenih alata korištena je biblioteka Measured[6] kao implementacijski detalj sustava mjernih jedinica o čemu će biti riječi dalje u radu. Za reprodukciju zvukova korištena je biblioteka AudioCue[7].

4.2. Arhitektura sustava

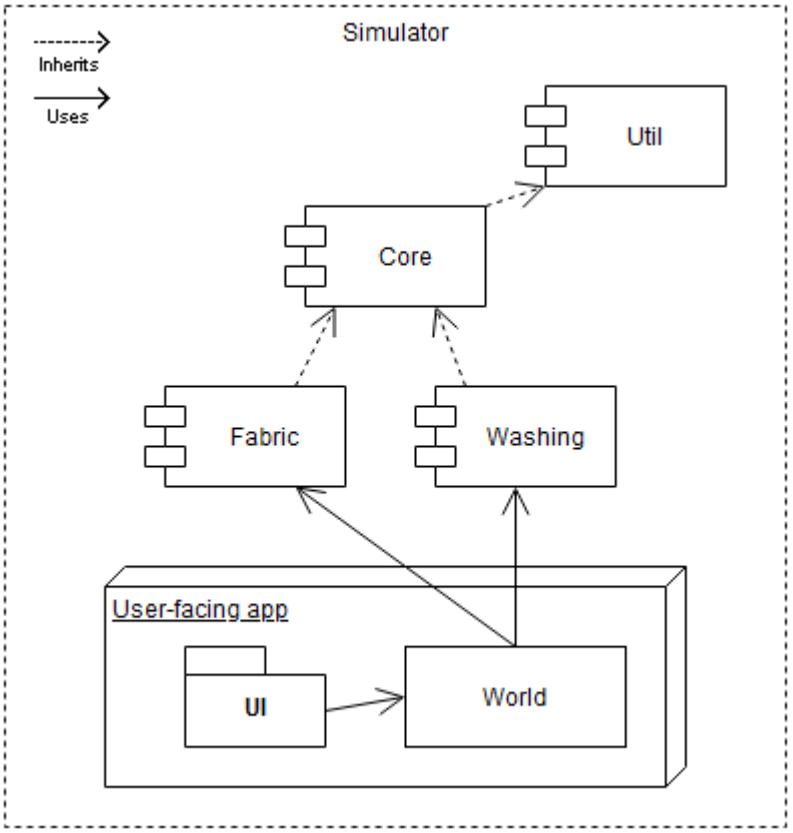
Sustav je podijeljen na module koji čine logički rastavljene samostojeće cjeline programskog koda. Moduli se sastoje od međusobno povezanih sučelja, klasa i funkcija koje zajedno tvore smislenu cjelinu. Različiti moduli međusobno komuniciraju preko izloženih sučelja. U sustavu postoje sljedeći moduli:

- *glavna aplikacija* - polazna točka sustava, povezuje druge module i definira korisničko sučelje
- *core* - modul koji definira osnovne zakone simulacije (mjerne jedinice, tvari, elektricitet, tijela, vrtnja, tok)
- *fabric* - modul koji definira tkaninu i odjeću
- *washing* - modul koji opisuje koncepte pranja i definira perilicu
- *util* - pomoćni modul s univerzalno korisnim pomagalima za pisanje koda

Međusobna ovisnost ovih modula dana je na slici 4.1. koja prikazuje gradivne komponente simulatora. Gradivne komponente su programski elementi koji čine aplikaciju na visokoj razini, poput logički rastavljenih modula i drugih samostojećih dijelova sustava. Komponenta aplikacije s kojom korisnik rukuje se sastoji od paketa korisničkog sučelja i „*svijeta*" simulacije. *Svijet* služi kao stanište koje obuhvaća perilicu i rublje koje postoji u simulaciji. Korisničko sučelje prikazuje stanje stvari sadržanih u *svijetu*.

Komponente na dijagramu na slici 4.1. povezane su strelicama koje dolaze u dvije vrste, prikazane legendom u gornjem lijevom kutu slike. *Inherits* predstavlja relaciju nasljeđivanja, a *Uses* predstavlja relaciju korištenja.

[5] Koin - https://insert-koin.io/
[6] nacular/Measured - https://github.com/nacular/measured
[7] philfrei/AudioCue - https://github.com/philfrei/AudioCue



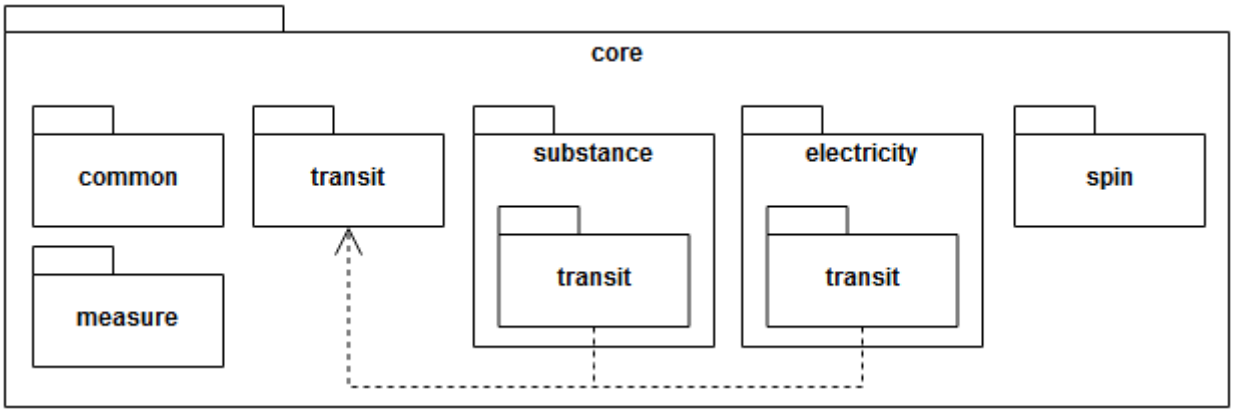
Slika 4.1. Ovisnost gradivnih komponenti simulatora.

Modul *core* je važan jer definira fizičke zakone simulacije. Osnovni koncepti definirani u zahtjevima na simulaciju implementirani su u ovom modulu.

Modul *fabric* ispunjava zahtjeve koji se odnose na tkaninu i odjeću. Ovdje su definirani svi odjevni predmeti koji se koriste kao primjeri perivih objekata.

Modul *washing* implementira perilicu i uvodi sve koncepte vezane uz pranje i osvježavanje.

Pojedini važni koncepti iz modula *core* su smješteni u odvojene pakete vidljive na slici 4.2. Komponenta mjernih jedinica *measure* prisutna je u svim dijelovima programa. Komponenta koja modelira tok naziva se *transit* i definira sučelja potrebna za opis toka. Tvari i elektricitet su pojave koje imaju sposobnost toka, stoga one imaju vlastite *transit* komponente. Te komponente preciziraju značenje i ponašanje generalnog toka za vlastite potrebe. Na primjeru toka tvari, ondje se definiraju elementi protočnog sustava tekućine poput ventila i rezervoara. Paket *common* definira općenite često korištene elemente poput sučelja tijela koje može biti implementirano u raznim kontekstima i dijelovima simulacije. Paket *spin* definira sve vezano uz koncept vrtnje.



Slika 4.2. Komponente središnjeg modula.

4.3. Detaljna implementacija sustava

Slijedi detaljan dizajn i implementacija simulacije prema definiranim zahtjevima. Ovaj dio rada je podijeljen prema tematici dijela simulacije koji se implementira, počevši od najnižih, osnovnih koncepata koji čine gradivne elemente za sljedeću razinu, sve do konstrukcije same perilice i programa pranja.

4.3.1. Fizikalne veličine

Osnova koja je potrebna u gotovo svim dijelovima simulacije je sustav mjernih jedinica. Zahtjev 1. definira potrebu za postojanjem fizikalnih veličina i pripadajućih mjernih jedinica koje će opisivati razna fizička svojstva objekata simulacije. Kako bi mjere bile smislene, na njima se moraju moći provoditi osnovne matematičke operacije, a određene mjerne veličine se moraju moći predstaviti različitim jedinicama, kao i međusobno kombinirati.

U sustavu je korištena javno dostupna biblioteka za Kotlin programski jezik koja ispunjava navedene potrebe. Biblioteka se zove Measured i kratko je spomenuta u poglavlju korištenih tehnologija. Korištenje ovakve biblioteke olakšava rukovanje mjernim jedinicama i čini kod razumljivijim. Uz pružanje nekolicine često korištenih mjera, biblioteka omogućava definiranje vlastitih mjera i odnosa među njima. Od postojećih mjera je iskorištena mjera vremena koja je predstavljena u jedinicama sekundi, milisekundi, minuta, i sati. Ostale veličine uvedene su prema potrebama simulacije, a one su:

- volumen mjeren u litrama i mililitrima,
- energija mjerena u džulima, kilodžulima i kalorijama,
- temperatura mjerena u stupnjevima Celzijusa,
- snaga mjerena u vatima (džul/sekunda) i kilovatima,
- protok mjeren kao volumen kroz vrijeme,

- frekvencija mjerena u Hertzima,
- brzina vrtnje, predstavljena kao broj okretaja u minuti.

Navedene veličine su dovoljne za opis svih koncepata ove simulacije.

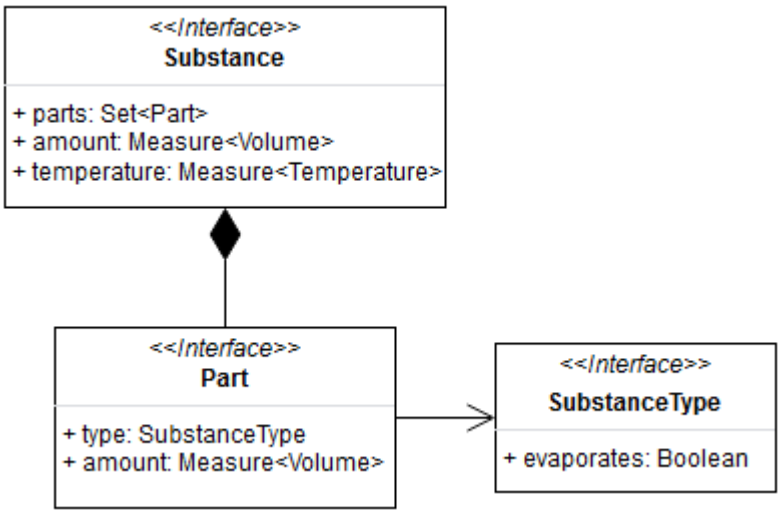
4.3.2. Tvari

Za ispunjavanje zahtjeva pod točkom 2. potreban je tip podatka koji predstavlja nakupinu određene količine tvari. Najprije se definiraju najniži konstrukti, odnosno oni koji nemaju ovisnosti o konceptima koji do sada nisu definirani. U ovom slučaju to je tip tvari. Tip tvari se može predstaviti sučeljem *SubstanceType*. Budući da voda, koja je tip tvari, mora imati sposobnost isparavanja, sučelju je dodano binarno svojstvo *evaporates* koje označava tu sposobnost. Sami tipovi mogu biti definirani kao članovi enumeracijskih klasa koje implementiraju spomenuto sučelje.

Kada postoji pojam tipa tvari, moguće ga je koristiti za izradu modela tvari, *Substance*. Prema zahtjevima poznato je da se nakupina tvari može sastojati od više dijelova različitih tipova. Stoga se uvodi podsučelje *Part* koje definira jedan dio tvari homogenog tipa.

Zahtjev 2.5., koji određuje postojanje vrijednosti temperature tvari ispunjen je svojstvom *temperature* koje je jednostavna mjera temperature.

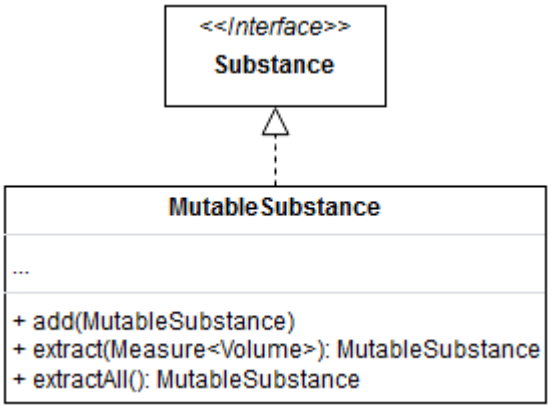
Opisan odnos i sastav ovih sučelja prikazan je klasnim dijagramom na slici 4.3.



Slika 4.3. Model tvari.

Kako će kasnije u procesu dizajna tvari morati imati sposobnost toka, nakupine tvari moraju moći biti djeljive u manje količine, kao i spojive u veće nakupine. Ova sposobnost omogućit će tvarima tok kroz cjevovode i punjenje rezervoara. Promjenjivost nakupina tvari postiže se konkretnom implementacijom sučelja tvari - *MutableSubstance*. Klasi je dodijeljena

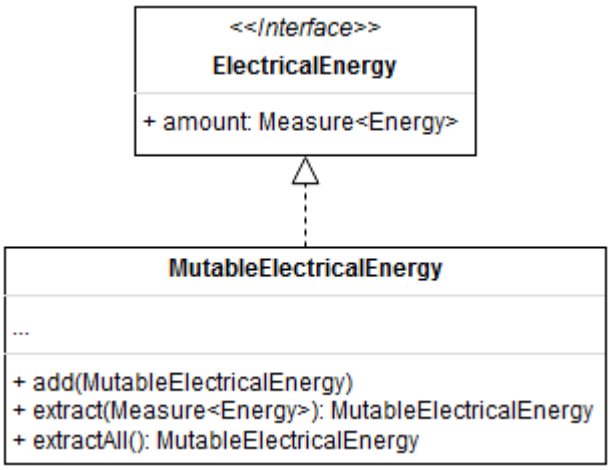
metoda za dodavanje drugih tvari u vlastiti sadržaj, te metode za vađenje određene količine tvari, stvarajući novu, odvojenu nakupinu (slika 4.4.).



Slika 4.4. Izmjenjiv oblik modela tvari.

4.3.3. Elektricitet

Elektricitet ima veliku sličnost s tvarima u pogledu implementacije njihovih programskih modela u ovoj simulaciji. Obje pojave predstavljaju neku količinu, u slučaju tvari je to volumen, a u slučaju elektriciteta je to energija. Na klasnom dijagramu na slici 4.5. električna energija je predstavljena sučeljem *ElectricalEnergy* koje posjeduje količinu mjerenu u jedinici energije. To je analogno s nakupinom tvari, ali jednostavnije jer elektricitet nema vidno različite tipove koji se mogu miješati. Elektricitet također ima sposobnost toka, pa tako implementira varijantu *MutableElectricalEnergy* s metodama za dodavanje i vađenje zasebne količine elektriciteta.

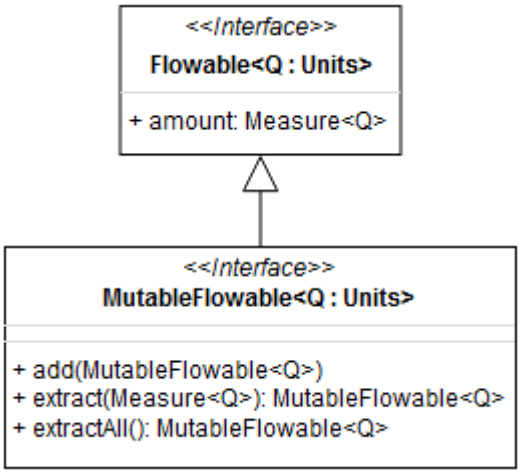


Slika 4.5. Model električne energije.

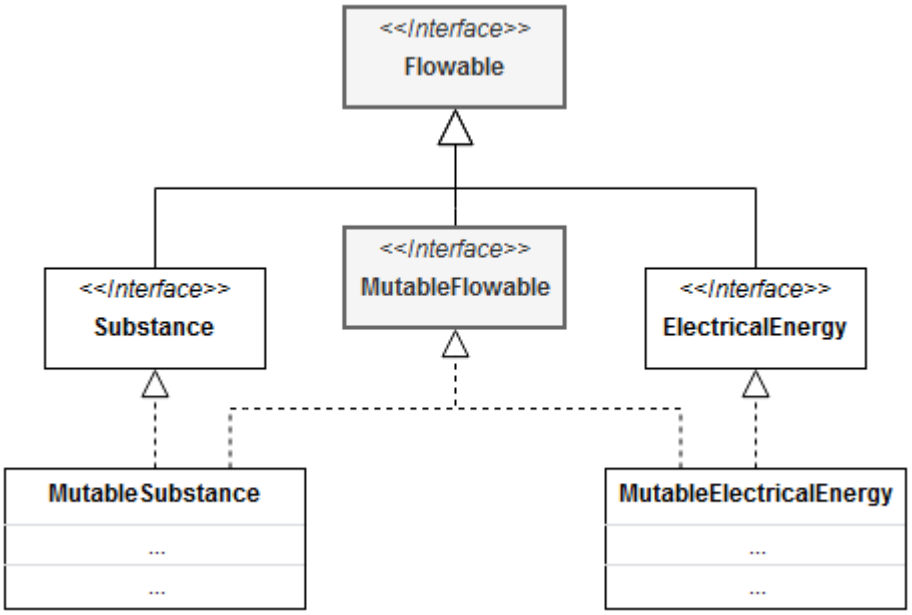
Više o toku i mogućnostima generalizacije između tvari i elektriciteta nalazi se u sljedećem potpoglavlju.

4.3.4. Tok

U do sada opisanom modelu tvari i elektriciteta napravljene su pripreme za implementaciju njihove sposobnosti toka u obliku omogućavanja djeljivosti njihovih nakupina. U ovom dijelu se ta sposobnost toka formalizira uvođenjem sučelja *Flowable* i *MutableFlowable* koja označavaju sve što može teći, odnosno dijeliti svoj sadržaj na više dijelova. Razlika ovdje je u tome što su ova sučelja generalizirana tako da se mogu odnositi ne samo na tok tvari, nego i na tok elektriciteta. Generalizacija je postignuta korištenjem klasnog parametra. Klasni parametri omogućavaju korištenje generičkih tipova podataka, tj. tipova koji mogu imati različite vrijednosti u različitim kontekstima. U ovom slučaju definiran je klasni parametar *Q* ograničen na tipove koji su nasljednici tipa *Units*. Tip *Units* uvodi prethodno opisana biblioteka mjernih jedinica, a označava bilo kakvu mjernu jedinicu. Vrijednost parametra *Q* određuje veličinu koja teče, u ovom slučaju se to odnosi na volumen i energiju. Sučelja i objekti tvari i elektriciteta nasljeđuju opisano svojstvo toka što je vidljivo uspoređivanjem sadržaja njihove strukture s opisom sučelja na klasnom dijagramu sa slike 4.6. Konačna hijerarhija koncepata sa svojstvom toka prikazana je na klasnom dijagramu na slici 4.7.



Slika 4.6. Model toka.



Slika 4.7. Generalizacija toka tvari i elektriciteta.

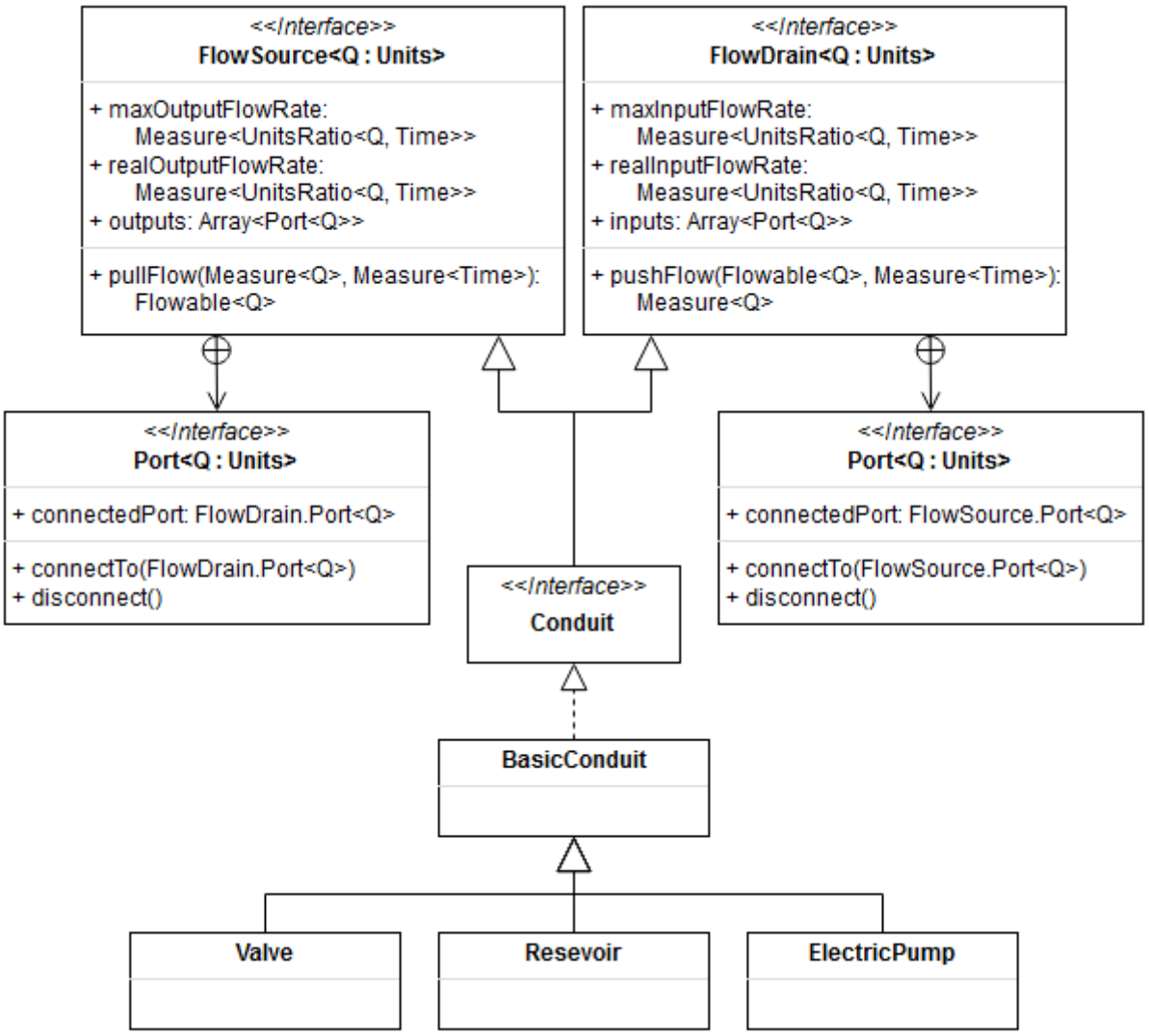
Definirana je sposobnost toka, ili preciznije, sposobnost dijeljenja količine što će omogućiti tok, ali ovaj sustav je nepotpun bez struktura koje će podržavati tok. Osnovni takav element je jednostavan vodič (engl. *conduit*). U slučaju toka tvari to može biti cijev, a u slučaju toka elektriciteta električni vodič poput žice. Tok je u osnovi omogućen putem dva sučelja: izvor toka *FlowSource*, i odljev toka *FlowDrain*. Izvor toka prema van omogućava vađenje protočne količine, dok ju odljev može primati. I izvor i odljev imaju svojstvo maksimalne brzine toka koju podržavaju. Radi povećanja fleksibilnosti, ovo svojstvo se može rastaviti na svojstvo maksimalnog protoka i svojstvo stvarnog (trenutnog) protoka. Maksimalni protok se može koristiti za označavanje nazivne protočnosti cijevi, dok stvarni protok može značiti vrijednost protočnosti koju cijev ima u danom trenutku. Ta vrijednost se može razlikovati od maksimalne iz raznih razloga, primjerice utjecajem otvorenosti ventila ili nakupine prljavštine u cijevima koja prigušuje protok. Potonji primjer nije simuliran ovom simulacijom, ali je otvoren kao mogućnost unaprjeđenja.

Izvor i odljev toka imaju svojevrsne priključke (engl. *port*) koji služe za njihovo međusobno povezivanje. Pri povezivanju, priključci se mogu spajati samo s priključcima suprotnog protočnog sučelja. Drugim riječima, priključak izvora može se spojiti samo s priključkom odljeva. Kako bi se ispunio zahtjev 4.5., izvori i odljevi se moraju moći spajati spojevima ‘više na više’, pa tako jedan izvor ili odljev može sadržavati niz priključaka koji se mogu neovisno spajati.

Vodiče karakterizira to što su ujedno i izvori i odljevi. To znači da se oni mogu ulančavati i tako stvarati složeniji cjevovod ili električnu mrežu.

Nedostatak ovakvog modela je činjenica da protočni elementi dozvoljavaju tok samo u jednom smjeru, no budući da dvosmjeran tok nije nužan za potrebe ove simulacije, ovo ne predstavlja problem.

Dijagram na slici 4.8. prikazuje strukturu i odnos spomenutih sučelja toka. Ovdje se također nalazi klasni parametar *Q* koji kao i kod opisa protočnih pojava određuje tip veličine koja teče. Brzina protoka (engl. *flow rate*) označena je tipom *Measure<UnitsRatio<Q, Time>>*, gdje *UnitsRatio* predstavlja omjer dviju mjera, u ovom slučaju bilo koje protočne veličine i vremena. Ovaj koncept također dolazi podržan korištenom bibliotekom mjernih jedinica. Klasa *BasicConduit* je osnovna implementacija jednostavnog univerzalnog vodiča. Ostali elementi protočnog sustava proširuju ovu klasu dodajući joj vlastite specifične sposobnosti. Više riječi o tim elementima može se pronaći u opisu upravljanja tokom tvari kasnije u dokumentu.



Slika 4.8. Klasna struktura protočnih elemenata.

Implementacija *BasicConduit* klase koja predstavlja podlogu za sav tok u simulaciji dolazi s brojnim izazovima. Najprije treba riješiti problem spajanja dvaju vodiča. Isprva se ovaj pothvat čini jednostavan jer je kao svojstvo sučelja već definiran spojeni priključak te metode za spajanje i odspajanje. Jednostavna implementacija metoda za upravljanje spojevima dodjeljuje ili uklanja dani priključak varijabli *connectedPort*. Ovim primjerom, pozivanje metode *connectTo* na priključku izvora spojit će ga na dani odljev. Gleda li se na ovaj problem samo iz perspektive jednog elementa, napravljeno je sve što je potrebno. Ipak, važno je obratiti pozornost na potrebu da oba elementa budu „svjesna" da su spojena s drugim elementom. Trenutno samo element na kojemu je pozvana metoda spoja će znati za taj spoj, dok će drugi element ostati netaknut i ponašati se kao da nije spojen ni na što. Rješenje koje uklanja ovaj problem bez promjene vanjskog načina korištenja sučelja za spajanje je implementirati metode tako da recipročno pozivaju iste metode na elementima na koje se spajaju odnosno odspajaju. Kako bi se izbjeglo beskonačno kružno pozivanje tih metoda, treba paziti da se recipročni poziv odvija nakon što je stanje spoja na trenutnom objektu izmijenjeno, tako da priključci mogu znati tko je pozvan prvi, a kome je upućen recipročan poziv obavijesti o spoju. Slično tome, potrebno je odspojiti bilo kakav drugi spoj koji možda postoji na priključcima prije spajanja novog. Za lakše razumijevanje, programski kod spajanja i odspajanja priključaka odljeva na priključku izvora toka dan je na slici 4.9. Ovaj dio ponašanja dovoljno je općenit da se implementacija može izvesti kao izvorna implementacija u sučeljima priključaka izvora i odljeva, umjesto u klasi *BasicConduit*.

```
infix fun connectTo(receiverPort: FlowDrain.Port<QuantityType, FlowableType>) {
  if (connectedPort == receiverPort) return
  if (connectedPort != null) disconnect()
  connectedPort = receiverPort
  receiverPort connectTo this
}

fun disconnect() {
  connectedPort?.let {
    connectedPort = null
    it.disconnect()
  }
}
```

Slika 4.9. Programski kod spajanja i odspajanja priključka odljeva na priključku izvora toka. [Kotlin]

Sama pojava toka odvija se kroz lančano pozivanje metoda *pullFlow* ili *pushFlow*, ovisno o tome povlači li se tok iz izvora ili se gura u odljev. Implementacija ovih metoda

zahtijeva rješavanje nekoliko potencijalnih problema. Spoje li se elementi tako da naprave zatvoreni krug toka, mora se spriječiti situacija u kojoj tok beskrajno kruži sustavom. Postojanje nekakvog oblika inercije toka riješilo bi ovaj problem, ali takvo svojstvo nije predviđeno za ovu simulaciju. Kako i dalje ne bi bilo moguće dobiti beskonačno kruženje toka, svakom valu toka dodjeljuje se jedinstveni identifikator te se tok zaustavlja čim jedna od komponenti protočnog sustava uoči isti identifikator po drugi put. Ova mjera je poduzeta kao dobra praksa, iako u konačnoj implementaciji perilice neće biti cikličkih tokova.

Drugi problem implementacije toka je samo prenošenje materije. Vodič mora podržati više paralelnih izvora i odljeva od kojih svaki može u bilo kojem trenutku postati ispražnjen odnosno zasićen. Primjerice, ako je jedan od dva spojena odljeva zasićen, tj. ne može progurati traženu količinu tvari, drugi odljev mora preuzeti njegov neprovedeni tok.

Kod opisa implementacije prvo dolazi guranje toka, tj. metoda *pushFlow*. Metoda prima parametar protočnog tipa kojeg se želi progurati i vremenski okvir unutar kojeg se ta akcija odvija. Osim ovoga, obje metode toka primaju ranije opisani jedinstveni identifikator vala toka za sprječavanje beskonačnih ciklusa u kružnim spojevima. Kako bi se na svakom koraku kroz lanac znalo prolazi li tok uspješno, kao povratni tip metode guranja toka je postavljena mjera količine veličine koja se gura. Ta mjera označava preostalu količinu materije koja nije uspjela biti progurana dalje kroz lanac. Najčešći razlog postojanja takve količine je zatvoreni ventil, napunjen rezervoar, ili manja protočnost neke karike u lancu. Informacija o neproguranoj količini se tako propagira uzvodno kroz lanac odakle je val došao kako bi svaka karika znala koliko je materije stvarno progurano.

Kako je već spomenuto, u svakom trenutku se neki od odljeva može zasititi i tako prestati propuštati tok. Ovo se ne može predvidjeti unutar komponente, stoga se za početak postavlja optimistična pretpostavka kako će svaki odljev primiti sve što mu se preda. Ostane li višak koji nije proguran kroz neke od odljeva, ta količina se ponovno pokušava progurati kroz odljeve koji se još nisu zasitili. Kad takvih odljeva ponestane, više nema drugih mogućnosti nego stati i uzvodno vratiti informaciju o preostaloj neproguranoj količini.

Način na koji se količina dijeli na paralelno spojene elemente je jednostavan. Ukupna količina se rastavlja na dijelove čije su veličine proporcionalne udjelima protočnosti pripadajućih odljeva u ukupnoj protočnosti svih spojenih odljeva. Drugim riječima, protočna količina se ravnomjerno rasporedi na sve odljeve uzimajući u obzir njihovu protočnost. Svaki dio se zatim prosljeđuje pripadajućem odljevu pozivom njegove *pushFlow* metode čime se lanac nastavlja. Kako bi se sve preostale neprogurane količine pouzdano obradile, opisani proces se odvija rekurzivno na način da je ulaz svakog koraka rekurzije količina koju treba progurati i niz

odljeva koji su u stanju primiti barem dio te količine. Sva preostala količina i odljevi koji se nisu zasitili nakon izvršenja koraka rekurzije se šalju u sljedeći korak sve dok se ne progura sva količina ili dok ne ponestane nezasićenih odljeva.

Povlačenje toka funkcionira na sličan način, metodom *pullFlow*. Glavni ulaz i izlaz metode su zamijenjeni u odnosu na *pushFlow*; uz vremenski okvir, funkcija prima mjeru količine koja se želi povući iz izvora, a kao izlaz daje stvarno povučenu materiju. Kao što se odljevi mogu zasititi, izvori se mogu isprazniti i tako dati manju količinu od tražene. Količina se na izvore dijeli na isti način, odnosno prema udjelu protočnosti pojedinih izvora u ukupnoj protočnosti svih spojenih izvora. Lančano pozivanje metode *pullFlow* na nizu spojenih izvora je ono što čini tok. Povlačenje se također vrši rekurzivno kako bi se povukla količina što bliža traženoj u slučaju da neki od izvora presuše. Tek nakon povlačenja materije iz svih izvora se doznaje koliko je stvarno dobiveno, a svi izvori koji nisu bili u stanju pružiti traženu količinu izbacuju se iz rekurzije. Ulaz koraka rekurzije je stoga preostala mjera količine koju je potrebno izvući i niz izvora koji su u stanju pružiti barem dio te količine. Koraci se ponavljaju sve dok se ne izvuče sva tražena količina ili dok ne ponestane neispražnjenih izvora.

Vremenski okvir koji se predaje kao parametar objema metodama toka prvenstveno služi za izračun količine o kojoj se radi. Poznato je da protočni elementi imaju vrijednost protočnosti, tj. brzine protoka. Vremenski okvir stoga određuje maksimalnu količinu koja može proteći u jednom pozivu metode. Ako se želi povući ili progurati više od toga, rezultat će biti isti kao da količina nije protekla zbog opstrukcije u cjevovodu.

Važno je napomenuti kako vremenski okvir ne prisiljava stvarno trajanje toka, već se on odvija trenutačno. Sustav toka podrazumijeva da pozivatelj metoda toka poštuje vremenski okvir koji predaje prilikom poziva te da neće pozive vršiti brže od trajanja perioda koji je naznačen tim vremenskim okvirom. Ako se to ipak dogodi, kroz cjevovod će efektivno proći količina veća od maksimalne količine dozvoljene svojstvom protočnosti.

Zbog jasnoće nazivlja, konkretne komponente vodiča u sustavima toka tvari i toka elektriciteta imaju posebne nazive u stvarnoj implementaciji. Umjesto referiranja na komponentu cijevi za protok tvari kao *Conduit<Volume>*, taj naziv se zamjenjuje oznakom *SubstanceConduit*. Isto tako, *Conduit<Energy>* se zamjenjuje s *ElectricalConduit*. Sve ostale klase koje sudjeluju u sustavu toka također dobivaju vlastite varijante naziva na isti način.

4.3.5. Upravljanje tokom tvari

Za upravljanje tokom tvari koje je potrebno za rad perilice potrebne su tri komponente: ventil, rezervoar, i pumpa.

Ventil

Ventil je najjednostavnija komponenta za implementaciju, a čini ga obična cijev s podesivom protočnosti. Lako je omogućiti podešavanje protočnosti s punom preciznosti željene brzine toka, iako će perilica pri radu koristiti samo krajnje vrijednosti otvorenosti - otvoren, zatvoren. Sve mogućnosti koje posjeduje obična cijev prisutne su i kod ventila jer njegova klasa nasljeđuje klasu osnovnog vodiča tvari.

Rezervoar

Sljedeća ključna komponenta je rezervoar kojeg karakterizira sposobnost pohrane određene količine tvari. Glavno obilježje rezervoara je njegov kapacitet koji određuje tu količinu. Rezervoar je također nasljednik jednostavne cijevi, ali modificira način povlačenja i guranja toka tako što uzima u obzir pohranjenu tvar.

Postoji više načina na koje se može implementirati rezervoar ovisno o tome kakav sustav se želi simulirati. Primjerice, rezervoar u spoju s drugim komponentama se može ponašati kao dio hermetički zatvorenog sustava. Ako se na takvom rezervoaru pozove metoda povlačenja toka, negativni pritisak tog povlačenja se može prenijeti uzvodno na druge spojene komponente i tako povlačiti tok i iz njih. Nasuprot tome, rezervoar može biti otvoren, gdje povlačenje toka iz praznog rezervoara neće uzrokovati povlačenje toka iz izvora koji je na taj rezervoar spojen. Bubanj u standardnom spoju perilice se ponaša kao otvoren rezervoar što znači da rad pumpe koja povlači vodu iz njega ne rezultira povlačenjem vode iz uzvodno spojenog dispenzera jer se negativni fluidni pritisak disipira kroz prazninu rezervoara. Rezervoar kojeg implementira ova simulacija se stoga mora ponašati u skladu s ovim opisom. Ako je rezervoar prazan, metoda povlačenja toka neće imati utjecaj, tj. neće vratiti nikakvu tvar, bez obzira je li na rezervoar spojen izvor koji je u stanju pružiti tok. Ako rezervoar nije prazan, iz njega se tada jednostavno odvaja tražena količina i predaje pozivatelju.

Metoda guranja toka u rezervoar funkcionira tako da se tok najprije provodi kao kroz normalnu cijev. Tek kada odljev počne primati manje količine nego što se gura u rezervoar, taj višak će ostati pohranjen u rezervoaru. Kada je rezervoar napunjen do vrha i njegov odljev zasićen, tada je ponašanje ponovno jednako običnoj cijevi.

Do sada opisan sustav ne definira što se događa sa sadržajem rezervoara kada ne postoji aktivan val toka kroz komponentu, a spojeni odljev je nezasićen. Zadano ponašanje je tada nepostojeće; ništa se neće dogoditi. U nekim situacijama to može biti u redu, ali za naš tip rezervoara očekivano ponašanje je da sadržaj spontano krene istjecati. Radi toga je potrebno napraviti pozadinski mehanizam koji će imati ulogu gravitacije i neprestano gurati sadržaj rezervoara prema odljevu. Ovaj mehanizam se može implementirati u korutini koja nekim vremenskim intervalom poziva metodu guranja toka, a u svakom pozivu joj predaje period trajanja tog intervala. U potpuno sastavljenoj perilici, ovo svojstvo neće doći do izražaja jer je istjecanje tekućine iz bubnja kontrolirano pumpom spojenom na njegov odljev.

Pumpa (u kontekstu sustava toka)

Pumpa je komponenta koja ima mogućnost induciranja toka od svog izvora prema odljevu. To se izvodi tako što pumpa samostalno poziva vlastitu metodu povlačenja toka iz izvora, te povučenu količinu gura komponenti spojenoj na odljev. Pumpa kakva je ovdje potrebna je električni uređaj te za rad troši električnu energiju. Više o pumpi iz perspektive električnog uređaja nalazi se u poglavlju električnih uređaja. Pumpa ima podesivu snagu koja određuje brzinu induciranog toka prilikom rada. Slično kao i kod ventila, u praksi će se koristiti samo krajnje radne vrijednosti, odnosno snaga će uvijek biti maksimalna, a pumpa će biti ili uključena ili isključena.

4.3.6. Električni uređaji

Električni uređaj je apstraktni pojam koji se odnosi na svaki uređaj koji iz spojenog izvora troši električnu energiju za obavljanje nekog posla. Svaki električni uređaj ima nazivnu snagu koja određuje normalnu potrošnju energije pri radu. U primjerima implementiranim u ovoj simulaciji, nazivna snaga predstavlja maksimalnu snagu, a stvarna snaga je ručno podesiva. Svi ti uređaji imaju i stanje uključenosti. Kada je uređaj uključen, iz izvora se vuče potrebna količina energije, i u ovisnosti o dostupnoj energiji uređaj obavlja svoju zadaću.

Nalik načinu na koji je implementirano spontano istjecanje tekućine iz rezervoara, tijekom rada uređaja aktivna je korutina koja periodički poziva apstraktnu metodu *operate*. Ovu metodu implementiraju svi konkretni uređaji te se u njoj obavlja stvarni rad uređaja, kao i povlačenje energije za taj rad.

Grijač

Električni grijač je jednostavan uređaj koji električnu energiju pretvara u toplinsku. Ovo se može postići tako da se povučena energija, koja ovisi o postavci snage i dostupnoj energiji,

preračuna u vrijednost temperature koja se stvara. Kako u ovoj simulaciji samo tvari imaju svojstvo topline, grijač se jednostavno može „povezati" s tvari koju grije i mijenjati joj temperaturu.

Slijedeći fizikalne zakone zagrijavanja vode, za zagrijavanje jednog grama, odnosno mililitra vode za jedan Celzijev stupanj potrebna je jedna kalorija energije, kako se može zaključiti iz [13]. Zbog jednostavnosti, u ovoj simulaciji se zagrijavanje svih tvari vrši jednako kao zagrijavanje vode, iako u stvarnosti različite tvari mogu zahtijevati različite količine energije za zagrijavanje na istu temperaturu.

Elektromotor

Elektromotor je električni uređaj koji uzrokuje vrtnju objekata koji implementiraju *Spinnable* sučelje. Uz nazivnu snagu motor je definiran maksimalnom brzinom vrtnje. Konačna brzina vrtnje motora je proporcionalna snazi pri kojoj on radi u danom trenutku. Snaga je podesiva i ograničena nazivnom snagom uređaja.

Detaljna fizika koja stoji iza elektromotora je pojednostavljena jer se u ovoj simulaciji ne opisuje koncept okretnog momenta niti električnih veličina koje se odvijaju u sklopu motora. Sve to je apstrahirano iza dijade snage i brzine vrtnje. Otpor vrtnji jednako tako nije definiran simulacijom što znači da snaga motora jedina određuje brzinu vrtnje, bez obzira na ono što se vrti. Ovaj nedostatak se može zanemariti budući da fokus nije na preciznim vrijednostima potrošnje energije. Kako vrtnja ovdje nema pojam inercije, promjene u vrtnji su skokovite. Razlika u ponašanju je zanemariva za potrebe ove simulacije, a animacija vrtnje bubnja na korisničkom sučelju se može zasebno zagladiti.

Ako motor povlači manje energije nego što mu je potrebno, simulirana konkretna implementacija motora uzrokuje smanjenu brzinu vrtnje ravnomjerno s razinom nedostatka energije, premda je implementacija mogla biti takva da se vrtnja uopće ne događa ako nedostaje energije. Bilo koja varijanta je zadovoljavajuća jer u praksi simulacije neće nedostajati energije da bi ovo ponašanje došlo do izražaja.

Pumpa (u kontekstu električnog uređaja)

Pumpa iz perspektive komponente protočnog sustava je već objašnjena u poglavlju upravljanja tokom tvari. Ovdje se pumpa razmatra kao električni uređaj. Električna pumpa ima podesivu snagu koja je, kao i na svim uređajima do sad, ograničena nazivnom snagom. Pri najvećoj postavci snage, pumpa inducira maksimalni protok ograničen svojstvom protočnosti pumpe. Smanjenjem postavke snage, brzina induciranog toka linearno opada. Isto vrijedi i pri nedostatku električne energije.

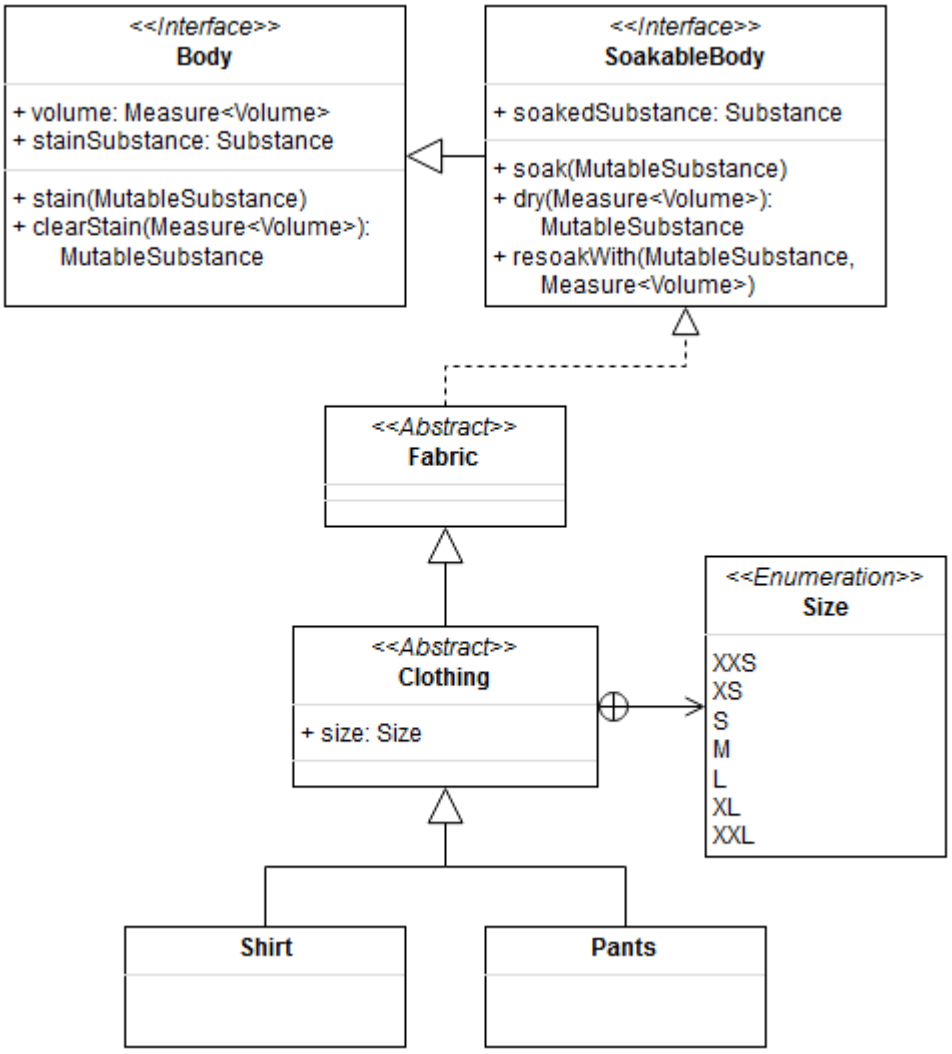
4.3.7. Tkanina i odjeća

Tkanina je pojam kojemu je dodijeljen poseban modul u sustavu. Svaki komad tkanine predstavljen apstraktnom klasom *Fabric* implementira sučelje upijajućeg tijela *SoakableBody*. Ovo sučelje je proširenje na sučelje običnog tijela *Body*. Sva tijela imaju određen volumen i mogućnost prljanja tvarima, a upijajuća tijela dodatno omogućuju upijanje tvari. Koncepti prljanja i upijanja su slični po implementaciji, ali se bitno razlikuju po onome što predstavljaju.

Uz očekivane metode natapanja i sušenja: *soak* i *dry*, upijajući objekti definiraju i metodu miješanja upijenog sadržaja s drugom tvari - *resoakWith*. Ovo je pomoćna metoda koja u jednom koraku odvaja traženu količinu tvari iz vlastitog upijenog sadržaja kao i iz dane tvari, te ih zamjenjuje. Rezultat ovoga je djelomično miješanje upijene tvari s onom danom u pozivu metode. Ovo je korisno u procesu pranja gdje se upijeni sadržaj mora miješati s tekućinom u kojoj se pere.

Odjeća predstavljena klasom *Clothing* je tip tkanine koji definira nekoliko oznaka veličine. Veličina odjeće predstavljena je enumeracijskom klasom *Size* koja izlistava potrebne veličine, a služi prvenstveno za kategoriziranje i lakše raspoznavanje različitih komada odjeće na grafičkom sučelju.

Pojedini odjevni predmeti predstavljeni su klasama koje nasljeđuju apstraktnu klasu odjeće i svakoj instanci omogućuju postavljanje željene veličine i volumena. U konačnici, za primjer se može stvarati neograničen broj košulja različitih veličina i volumena. Ovako definirana odjeća dovoljna je za uspješno demonstriranje procesa pranja. Klasni dijagram na slici 4.10. prikazuje opisanu klasnu strukturu odjeće.



Slika 4.10. Klasna struktura odjeće.

4.3.8. Sredstva za pranje

Kao pomoć kod pranja definiraju se dvije vrste sredstva: deterdžent za pranje rublja (engl. *laundry detergent*) i omekšivač tkanine (engl. *fabric softener*). Svrha deterdženta je da olakša, odnosno ubrza proces skidanja prljavštine iz tkanine, dok omekšivač služi kao završni tretman. Karakteristično svojstvo deterdženta prema zahtjevu 2.1.2. se opisuje svojstvom snage čišćenja (engl. *cleaning power*). Ovo svojstvo je decimalni broj s domenom od 0 do 1, a predstavlja utjecaj na veličinu dijela prljavštine koji će se ukloniti u svakom vremenskom koraku. Snaga čišćenja tvari u kojoj se pere s vrijednošću 0 znači da se prljavština neće uklanjati uopće, dok vrijednost od 1 u potpunosti potiče čišćenje, te je ono ograničeno samo drugim faktorima, poput temperature.

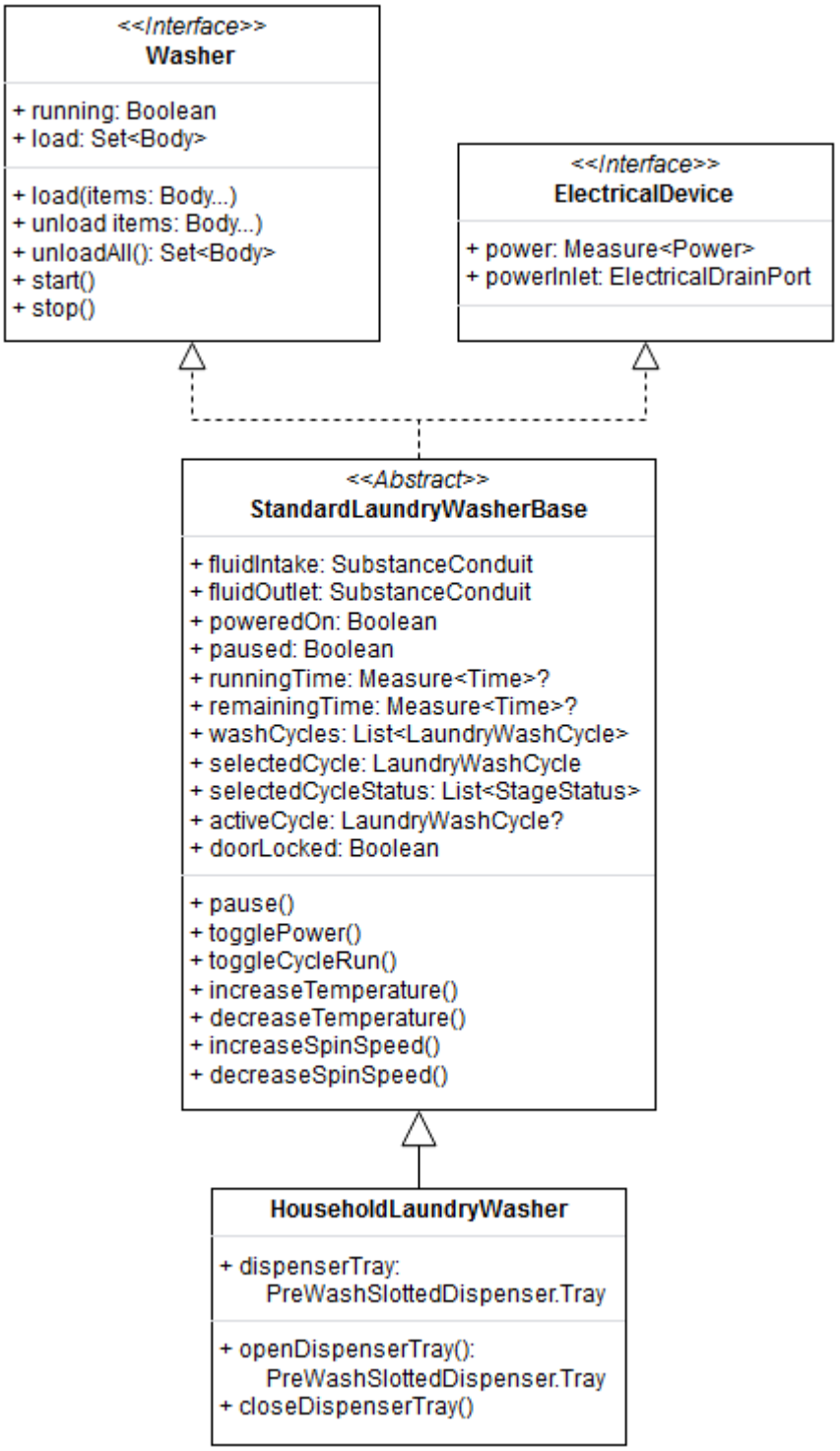
Kako bi postojala mjera prema kojoj se mogu uspoređivati rezultati uzrokovani pranjem uz omekšivač, definirano je svojstvo svježine (engl. *freshness*) prema zahtjevu 2.1.3. Ovim

svojstvom se nastoji kvantificirati promjena koju na odjeći uzrokuje pranje s omekšivačem. Može se reći da to svojstvo grubo predstavlja rezultirajući miris i mekoću tkanine. Domena vrijednosti je od 0 do 1, kao i kod snage čišćenja. Za razliku od snage čišćenja, ovo svojstvo je slobodnije za subjektivnu interpretaciju. Razina svježine nekog komada odjeće može se odrediti na više načina, ali u svakom slučaju se promatra svojstvo svježine tvari koje su upijene u objekt, ili prisutne na tijelu kao mrlja ako svježinu želimo interpretirati na taj način. Značenje vrijednosti svježine i poželjnog udjela omekšivača u odjeći se može interpretirati proizvoljno. Važno je samo da je vidljiva razlika u karakteristikama odjeće oprane s omekšivačem i one oprane bez njega.

4.3.9. Sklop perilice rublja

Kada su definirani svi preduvjeti za izgradnju perilice, može se prijeći na koncepte pranja i definiranje perilice. Kako bi postojala široka fleksibilnost u vezi toga što čini perilicu dobro je razraditi osnovno sučelje općenite perilice (ne samo perilice rublja) - *Washer*. Ono predstavlja bilo kakav objekt namijenjen pranju. Takav objekt može sadržavati punjenje za pranje u obliku skupa bilo kakvih tijela, te može biti uključen i isključen. To je sve što je propisano najosnovnijim sučeljem. Sve specifičnosti pojedinih tipova perilica mogu se definirati podsučeljima ili implementirajućim klasama.

Osnova perilice koja se izrađuje u ovom radu definirana je apstraktnom klasom *StandardLaundryWasherBase*. Ova klasa obuhvaća sve što je zajedničko široko prisutnim perilicama koje rade na principu predefiniranih programa pranja. Klasa implementira sučelje perilice i električnog uređaja te definira niz vlastitih funkcionalnosti koji pobliže opisuju standardnu perilicu rublja. Konkretna implementacija kućne perilice rublja s prednjim punjenjem je klasa *HouseholdLaundryWasher*. Ova klasa konceptualno predstavlja takvu perilicu, a u odnosu na standardnu perilicu koju nasljeđuje dodaje dispenzer s ladicom i odjeljkom za pretpranje. Opisana klasna struktura i unutarnja građa klasa prikazana je na slici 4.11.



Slika 4.11. Klasna struktura perilice rublja.

Određeni parametri koji utječu na ponašanje perilice i koji potencijalno mogu biti izmjenjivi ovisno o situaciji i željama za simulaciju su definirani kroz podatkovnu klasu *LaundryWasherConfig*. Ona predstavlja konfiguraciju fizikalnog ponašanja perilice preko parametara poput protočnosti ulaznog i izlaznog dijela cjevovoda, brzine vrtnje bubnja pri kojoj

započinje djelovanje centrifuge, razine vode pri kojoj vrata moraju biti zaključana, frekvencije provođenja mjerenja razine vode u bubnju, i slično.

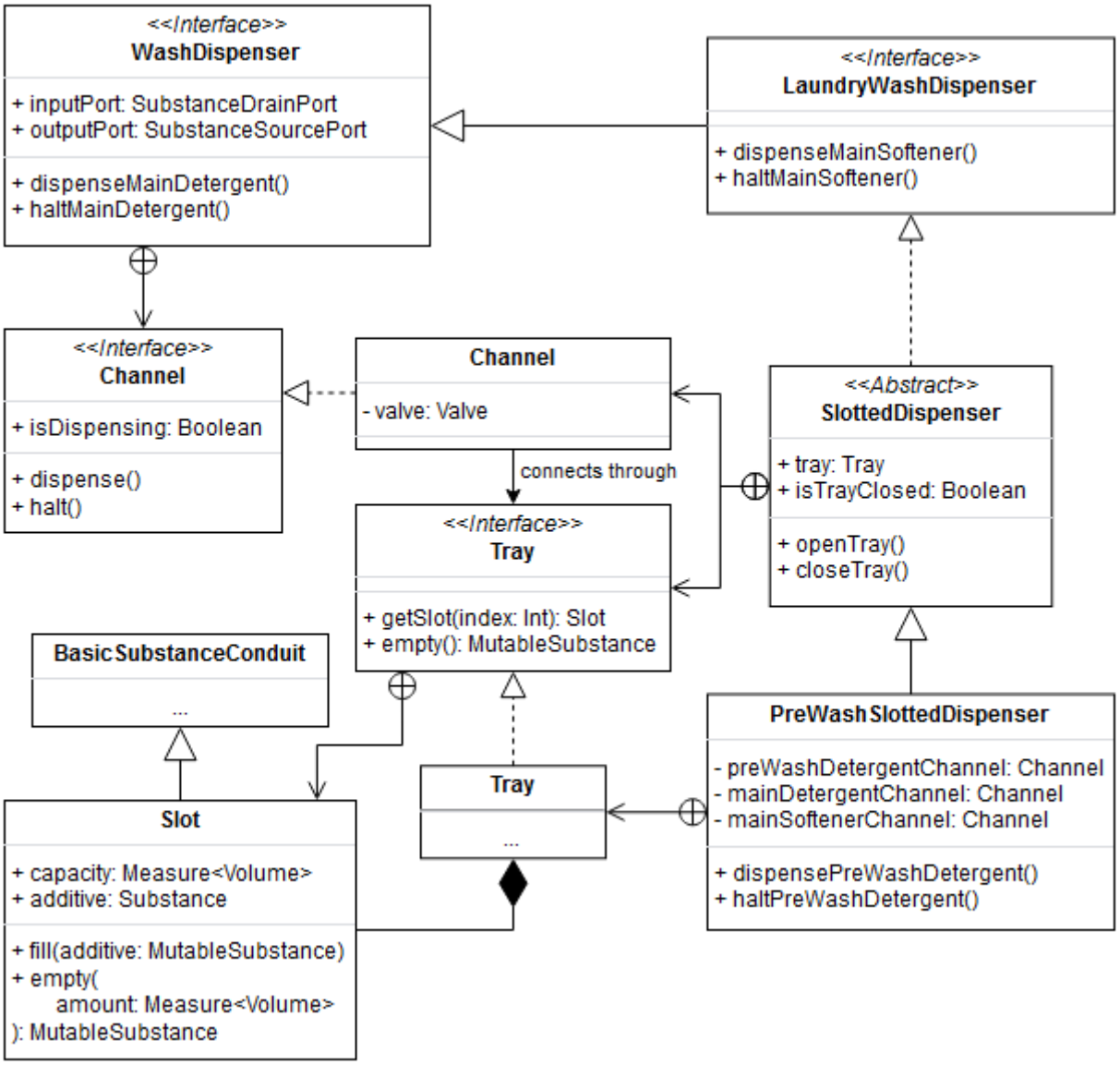
Komponentni sastav perilice dan je prethodno, dijagramom na slici 3.5., a u nastavku je opis pojedinih komponenti perilice koje do sada nisu detaljno analizirane.

Dispenzer

U teoriji, dispenzer deterdženta za pranje može postojati u raznim oblicima i načinima rada. Njegova osnovna svrha je omogućavanje doziranja sredstava za pranje radi kontroliranog ispuštanja tijekom pranja. Kao polaznu točku s najopćenitijim mogućnostima definirano je osnovno sučelje *WashDispenser*. Ono sadrži podsučelje *Channel* koje predstavlja kanal toka za punjenje perilice. Svaki dispenzer ima priključak za dotok i priključak za odljev tekućine, te podržava ispiranje kroz jedan kanal namijenjen deterdžentu. Dispenzer namijenjen perilici rublja može se definirati kao proširujuće sučelje *LaundryWashDispenser* koje dodaje kanal za omekšivač.

Opći oblik dispenzera namijenjen kućnim perilicama rublja definiran je apstraktnom klasom *SlottedDispenser* koja uvodi koncept ladice predstavljen sučeljem *Tray*. Ladica se sastoji od odjeljaka (*engl. slot*) zadanog kapaciteta u koje se mogu usipavati tvari namijenjene pranju (tzv. engl. *additive*). Odjeljci ladice se mogu modelirati kao elementi protočnog sustava, što znači da se mogu spajati s drugim protočnim elementima te uz korištenje ventila perilici omogućiti kontrolirano ispiranje sadržaja ladice. Više paralelnih kanala, koji su kod dispenzera s ladicom implementirani korištenjem ventila, mogu se povezati s izvorom dispenzera. Kada je ladica zatvorena, kanal uključuje pripadajući odjeljak ladice kao element svog toka, a ako je ladica otvorena tok ide izravno prema odljevu dispenzera, zaobilazeći ladicu. U ovakvom sustavu perilica može jednostavno upravljati ventilom kako bi se napunio bubanj do valjane razine, a kroz ladicu korisnik perilice može odrediti količinu i vrstu sredstva za pranje.

Konačni oblik dispenzera koji će se koristiti u perilici je *PreWashSlottedDispenser*. On uz spomenute kanale za deterdžent i omekšivač posjeduje i dodatni kanal za deterdžent pretpranja. Pojednostavljena klasna struktura opisanih vrsta i gradivnih elemenata dispenzera prikazana je klasnim dijagramom na slici 4.12. Potpun prikaz svojstava i međuodnosa prikazanih klasa ograničen je zbog preglednosti, a kompletna struktura može se pronaći u programskom kodu u prilogu rada.



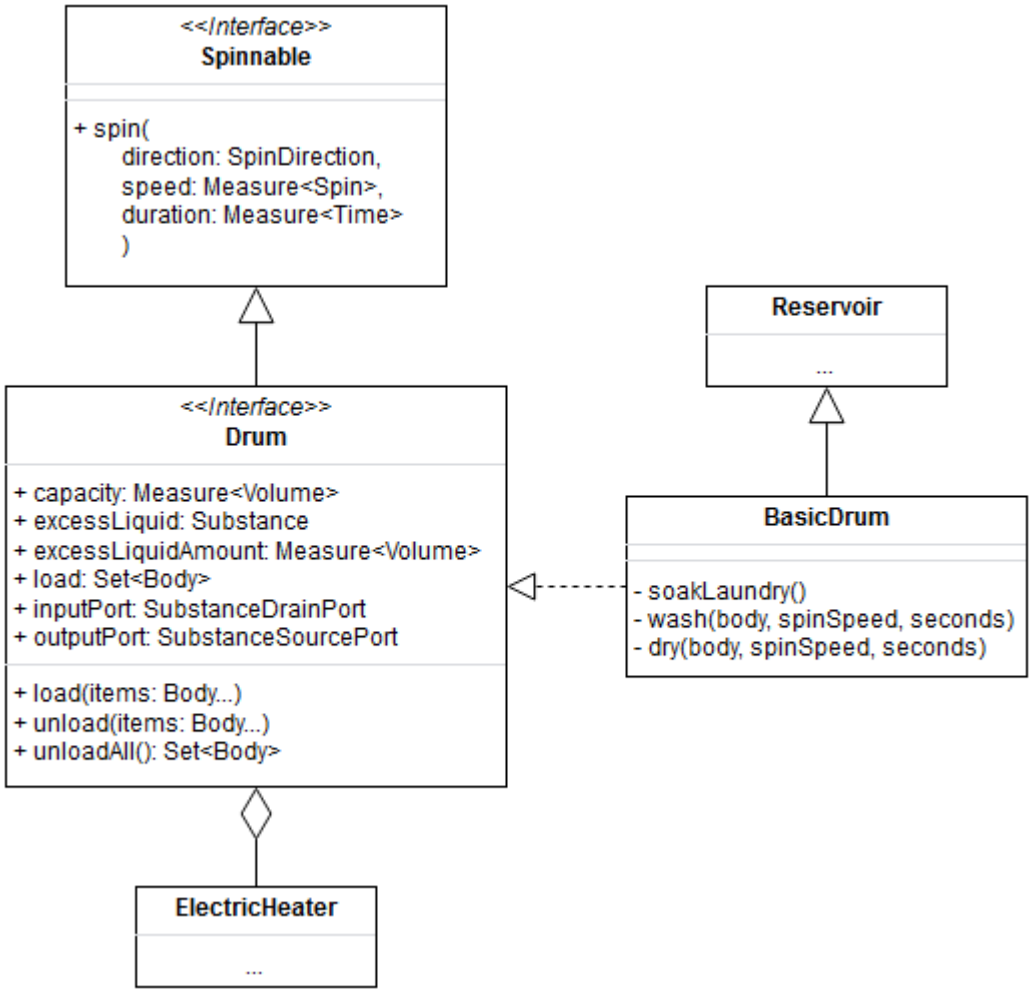
Slika 4.12. Pojednostavljena klasna struktura dispenzera.

Bubanj

Bubanj, odnosno kako je prethodno objašnjeno, jedinstvo kade i unutarnjeg okretljivog bubnja je središnja komponenta perilice koja provodi pranje. Bubanj je zapravo rezervoar čiji kapacitet uz količinu vode određuje i količinu volumena odjeće koja se u njega može pohraniti. Kao komponenta protočnog sustava, bubanj ima izvor koji se spaja na izlaz dispenzera, i odljev koji vodi prema pumpi za izbacivanje tekućine.

Na bubanj je moguće priključiti grijač koji će zagrijavati sadržanu tekućinu. Uz grijač je potrebno imati način mjerenja temperature. Ovu ulogu ispunjava termostat predstavljen klasom *Thermostat* (alias za klasu *MeasuringTrigger<Temperature>*) koji može okinuti zadanu akciju kada izmjerena temperatura dosegne zadane granice. Klasni dijagram na slici 4.13. prikazuje

strukturu sučelja bubnja *Drum* te kako je ono implementirano kroz jednostavan konkretni bubanj *BasicDrum*.



Slika 4.13. Klasna struktura bubnja.

Ključno svojstvo bubnja je sposobnost vrtnje dano sučeljem *Spinnable*. Vrtnja bubnja je ono što omogućuje pranje i, pri većim brzinama, sušenje odjeće. Granica brzine vrtnje pri kojoj pranje prelazi u sušenje definirana je u sklopu ranije spomenute konfiguracije perilice, a izvorno iznosi 80 okretaja u minuti. Odjeća koja se nalazi u bubnju koji se vrti brže od ove brzine neće se prati, nego će se sušiti, tj. izbacivati upijenu tekućinu. Sve do te granične brzine, odjeća će se prati ili prljati, ovisno o svojstvima i odnosu čistoće tekućine pranja i prane odjeće. Brzina ovog procesa je proporcionalna brzini vrtnje gdje brzina koja je neposredno manja od 80 okr/min ima najveći potencijal čišćenja.

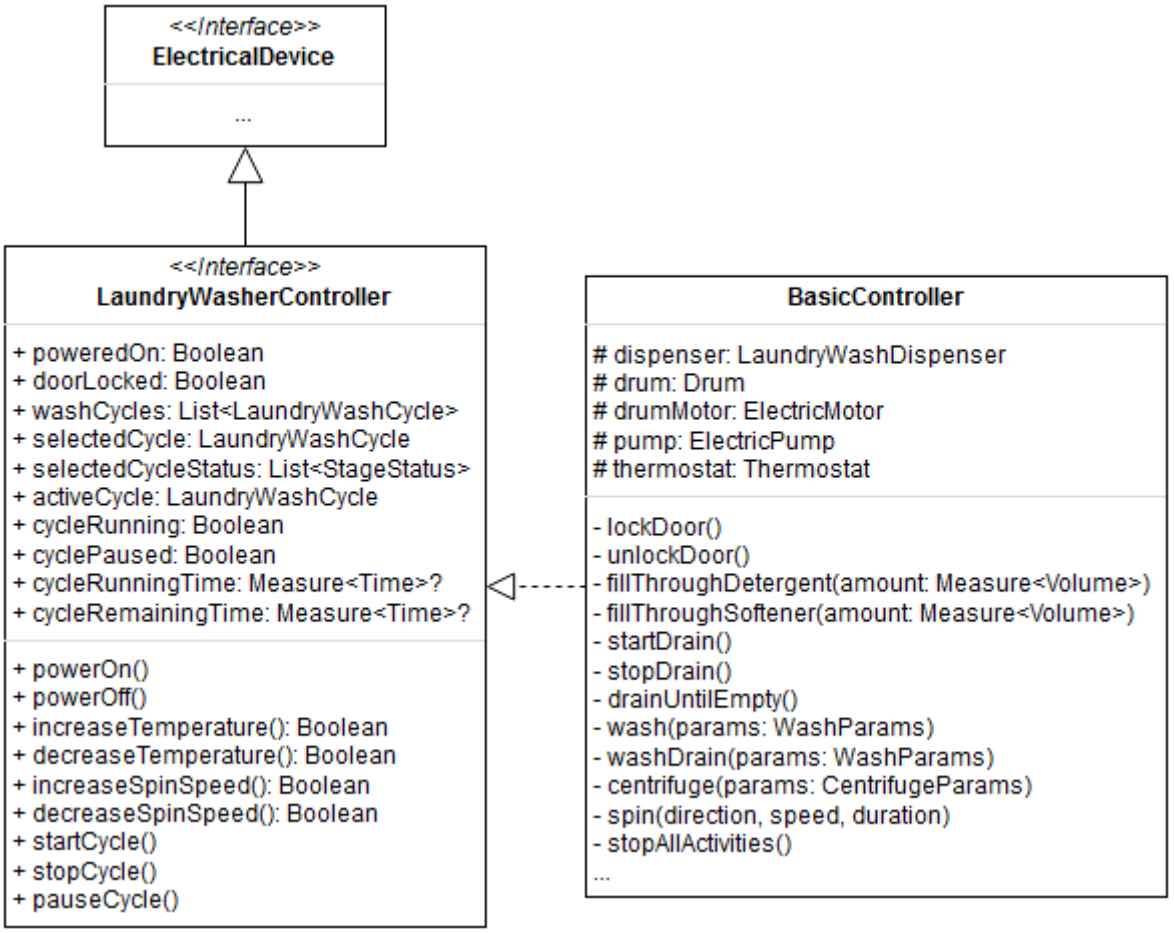
Prilikom vrtnje, sva upijajuća tijela nastoje upiti dostupnu tekućinu u bubnju do zasićenosti. Uz to, upijena tekućina i slobodna tekućina u bubnju se okretanjem kontinuirano

miješaju između sebe po stopi određenoj svojstvom u ranije spomenutoj konfiguraciji perilice i brzinom vrtnje bubnja.

Efikasnost čišćenja odjeće, osim o brzini vrtnje, ovisi i o količini upijene tekućine kao i njezinoj temperaturi i snazi čišćenja. Količina nečistoća koja se otklanja u određenom periodu dobiva se izračunom faktora koji predstavlja postotak od ukupne količine nečistoća. Precizan način dobivanja ovog faktora može se naći u programskom kodu u prilogu dokumenta. Izračuni efikasnosti pranja i sušenja se nalaze u sklopu klase jednostavnog bubnja *BasicDrum*, pod funkcijama *wash*, *dry*, i *calcCleaningPower*.

Kontroler

Kontroler je komponenta koja upravlja radom perilice, tj. koordinira rad svih njezinih komponenti. Glavna vodilja kontrolera su predefinirani programi pranja. Korisnik perilice izabire i pokreće programe pranja, a kontroler slijedi program i provodi potrebne operacije nad komponentama perilice kako bi se program ispoštovao. Za kontrolu perilice mogu se definirati procedure koje obavljaju konkretne zadaće potrebne za izvođenje segmenata programa pranja. Te procedure uključuju punjenje perilice tekućinom kroz pojedine kanale, pražnjenje bubnja, te pranje i centrifugiranje uz zadane parametre vrtnje. Uz zaključavanje vrata i druge sitne poslove kontroler tada ima sve potrebno za provedbu programa pranja. Specifičnosti sustava programa pranja objašnjene su pod idućim naslovom. Klasna struktura kontrolera perilice prikazana je na slici 4.14.



Slika 4.14. Klasna struktura kontrolera.

4.3.10. Programi pranja

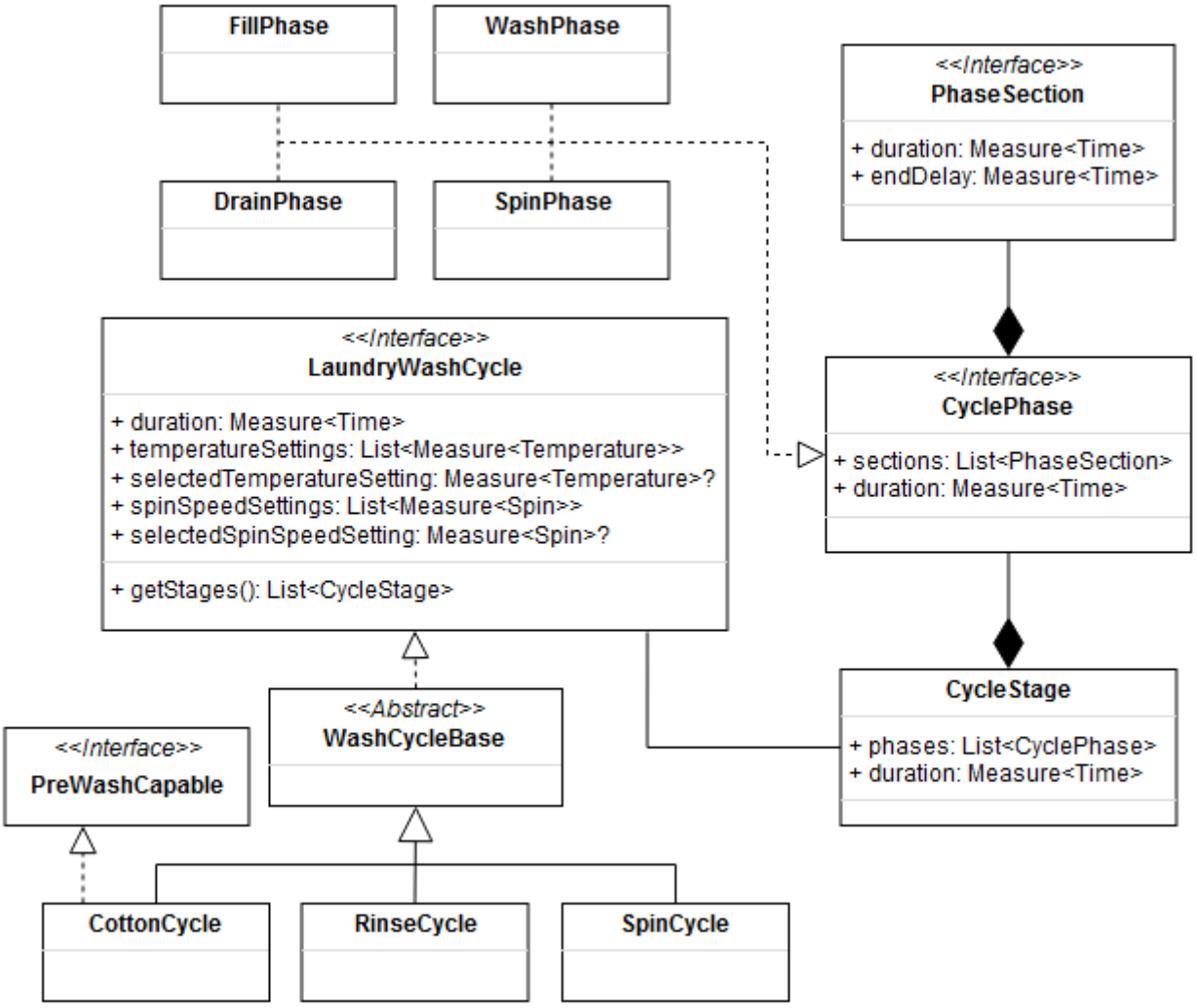
Program pranja je niz uputa koje kontroler slijedi kako bi upravljao radom perilice. Perilica može definirati više programa od kojih u svakom trenutku samo jedan može biti aktivan. Većina programa korisniku dozvoljava podešavanje parametara temperature u fazi pranja i brzine vrtnje u fazi centrifuge. Dostupne postavke ovise o samom programu.

Program se mora moći pauzirati u bilo kojem trenutku i mora moći nastaviti izvršavanje od dijela na kojem je zaustavljen. Za vrijeme trajanja programa, vrata su zaključana, a u slučaju pauziranja programa se mogu otključati ako razina vode u bubnju ne prelazi razinu definiranu u ranije spomenutoj konfiguraciji perilice. Potpuno zaustavljanje programa također mora biti moguće, pri čemu se perilica mora vratiti u početno stanje tako što će izbaciti svu vodu iz bubnja i na kraju otključati vrata.

Struktura programa pranja je podijeljena u stadije predstavljene klasom *CycleStage*. Program, tj. ciklus pranja se može sastojati od jednog ili više stadija. Stadij obično predstavlja jedan puni krug od punjenja bubnja vodom do pražnjenja. Ova podjela postoji primarno zbog

preglednosti ciklusa, dok je važnija podjela ona pri kojoj se stadiji dijele na faze predstavljene sučeljem *CyclePhase*. Faze predstavljaju moguće akcije koje perilica obavlja u danom trenutku, a to su: punjenje vodom kroz neki od kanala dispenzera (klasa *FillPhase*), pranje (klasa *WashPhase*), pražnjenje (klasa *DrainPhase*), i centrifuga (klasa *SpinPhase*). Faze su dodatno podijeljene u sekcije predstavljene sučeljem *PhaseSection* koje detaljnije definiraju parametre pojedinih faza u pojedinim trenucima.

Faza punjenja pri svakoj sekciji određuje kanal punjenja te razinu vode do koje se bubanj treba napuniti. Sekcije faze pranja definiraju periode vrtnje i mirovanja, te brzinu vrtnje bubnja. Faza pražnjenja ima dvije sekcije gdje prva nalaže fokusirano izbacivanje vode sve dok se bubanj ne isprazni, nakon čega slijedi sekcija koja uz rad pumpe propisuje i okretanje bubnja s istim mogućnostima kontrole kao i kod faze pranja. Faza centrifuge se sastoji od jedne ili više sekcija koje zasebno definiraju brzinu i trajanje vrtnje. Dodatno, programi koji imaju opciju pretpranja implementiraju sučelje *PreWashCapable* koje označava tu mogućnost. Ovakav dizajn omogućuje izradu programa koji pokrivaju velik dio očekivanog ponašanja perilice tijekom klasičnih ciklusa pranja. Klasni dijagram na slici 4.15. prikazuje opisanu strukturu. Pojedinačne implementacije sekcija koje čine faze pranja su izostavljene zbog preglednosti.



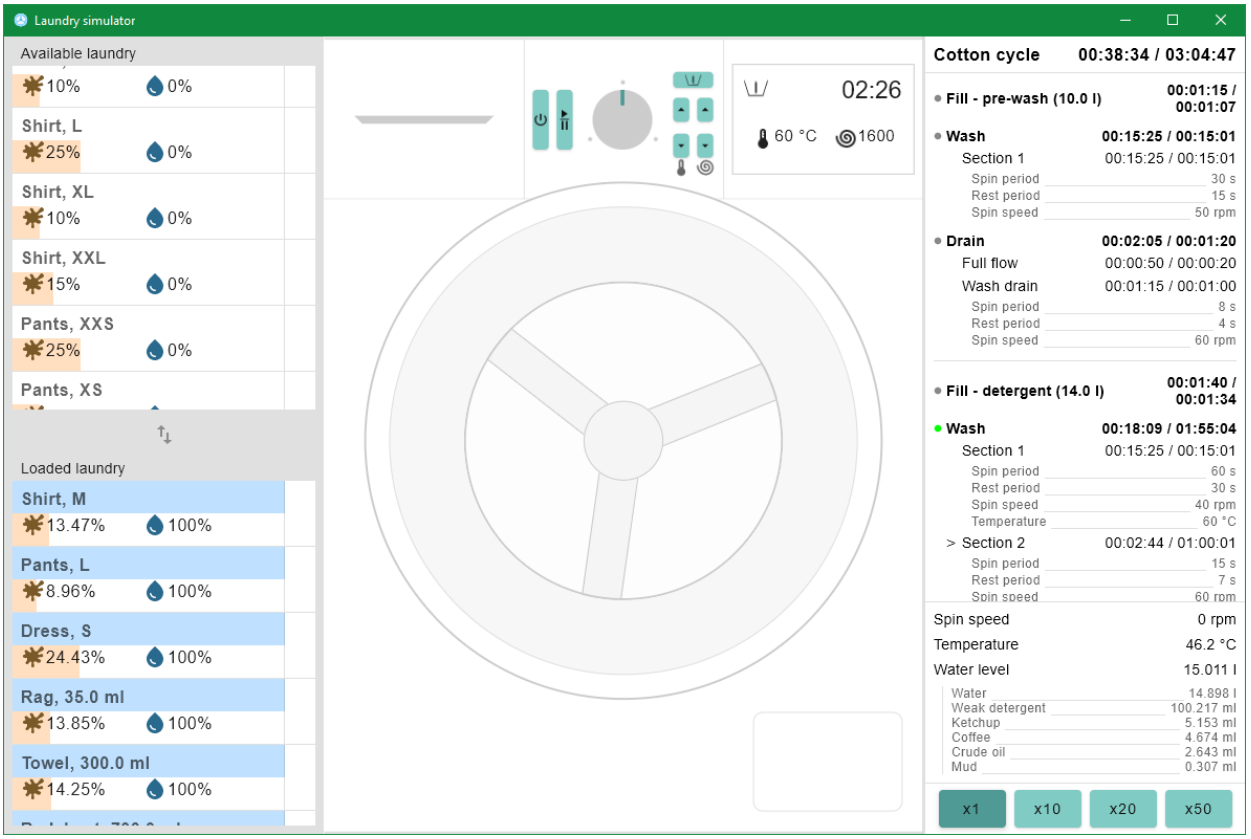
Slika 4.15. Klasna struktura programa pranja.

5. PROVOĐENJE SIMULACIJE

Radi utvrđivanja rezultata provođenja simulacije, u nastavku se provodi simulacija pranja različitih vrsta zaprljanog rublja. Prije toga je smješten opis najvažnijih dijelova grafičkog sučelja simulatora.

5.1. Grafičko sučelje simulatora

Simulator izlaže grafičko korisničko sučelje koje prikazuje trenutno stanje perilice i rublja, te omogućuje upravljanje perilicom. Slika 5.1. prikazuje izgled sučelja. Na lijevoj strani prozora moguće je vidjeti rublje s kojim se raspolaže. Ono može biti u jednoj od dvije liste. Gornja lista sadrži svo rublje koje je izvan perilice, dok lista ispod predstavlja rublje koje je u perilici. Dok god su vrata perilice otključana, rublje se može prebacivati iz jedne liste u drugu. Svaki komad rublja predstavljen je kratkim nazivom koji opisuje njegov tip i veličinu, bila ona u obliku konfekcijskog broja, kao u slučaju odjeće, ili u obliku mjere volumena izražene u mililitrima, kao što je slučaj kod ostalih tipova tkanine. Uz to, za svaki komad rublja se prikazuje mjera uprljanosti i mjera upijene tekućine, oboje kao postotak volumena tog komada rublja. Vertikalni indikator s desne strane prikaza pojedinih komada rublja predstavlja jednu interpretaciju razine svježine koju taj objekt posjeduje na osnovi upijenih i uprljanih tvari.



Slika 5.1. Grafičko sučelje simulatora.

Središnji element prozora simulatora čini ilustracija perilice na kojoj je moguće vidjeti njezino stanje u obliku u kakvom bi ono bilo izloženo prema stvarnim korisnicima. Odavde je moguće upravljati funkcijama perilice i dodavati sredstva za pranje preko dodatnog sučelja dozatora.

S desne strane perilice se nalazi panel s informacijama o odabranom programu pranja i unutarnjem stanju perilice. Na dnu tog panela se nalazi i kontrola brzine simulacije.

Napomena: zbog načina na koji je izvedena simulacija, povećanje brzine prolaska vremena rezultira smanjenjem preciznosti izvođenja izračuna. Stoga je moguće dobiti različite rezultate provodi li se simulacija normalnom brzinom ili ubrzano. Mogućnost ubrzanja vremena u simulaciji uvedena je prvenstveno zbog olakšanja demonstracije i provođenja testnih pranja.

5.2. Pranje rublja

Cilj provođenja simulacije je prikazati kako se prljavo rublje uz korištenje deterdženta za pranje postepeno čisti dok se tvari prljavštine miješaju s tekućinom u perilici. Po završetku pranja, promatranjem rezultirajućeg stanja rublja mogu se donositi zaključci o uspješnosti pranja.

Skup rublja se sastoji od raznih vrsta odjeće i druge tkanine raznih veličina i razina uprljanosti. U svrhu prikaza šireg raspona mogućih stanja, definirano je nekoliko tipova tvari s različitim mjerama tvrdokornosti. Tipovi su predstavljeni enumeracijskom klasom za koju su definirane numeričke vrijednosti svojstva tvrdokornosti. Kečap je odabran kao nečistoća koja je najlakša za pranje u odnosu na ostale. Nakon toga slijedi blato, kava, te sirova nafta kao najtvrdokornija nečistoća. Mjere tvrdokornosti ovih tipova nečistoća su dobivene isprobavanjem pranja s različitim vrijednostima dok nisu postignuti zadovoljavajući i dovoljno različiti rezultati pranja između različitih tipova nečistoće. Odabir samih tipova tvari je subjektivan i služi prvenstveno za olakšanje raspoznavanja različitih tvari u grafičkom sučelju. Svaki komad rublja je u različitoj mjeri uprljan nekom od tih tvari, kako je prikazano u tablici 5.1. Volumen pojedinačnih komada rublja izražen je u mililitrima, dok je mjera uprljanosti tih tijela izražena u postocima kao volumni udio strane tvari u volumenu tijela. Rezultati pranja, u pogledu preostale nečistoće na rublju, vidljivi su u desnoj polovici tablice.

Simulacija pranja je provedena u tri slučaja:

1. pranje bez ikakvih sredstava za pranje
2. pranje uz sredstva za pranje (deterdžent i omekšivač)
3. pranje uz sredstva za pranje i fazu pretpranja s dodatnim deterdžentom

Kako je prethodno napomenuto, ubrzanje vremena simulacije u ovom simulatoru rezultira malom promjenom u rezultatima pranja. Stoga je simulacija u svim navedenim slučajevima provedena u stvarnoj brzini, tj. bez ubrzanja vremena.

Sva pranja su provedena uz obični program pranja osnovnog trajanja od 2 sata i 47 minuta pri postavci temperature od 60 °C. Detalji samog programa pranja dostupni su u programskom kodu u prilogu dokumenta. Korišten je program pranja pamuka, definiran klasom *CottonCycle*.

Korištena sredstva za pranje u slučajevima 2. i 3. uključuju 150 ml deterdženta umjereno jake snage pranja te 70 ml omekšivača umjereno jake snage osvježivanja. Slučaj 3. u dodatnom dijelu pretpranja, koji traje 15 minuta, dodaje 75 ml već spomenutog deterdženta. Mjere količine sredstava za pranje u slučaju 3. također su vidljive u prikazu grafičkog sučelja dozatora na slici 5.2. Početno stanje rublja u svakom od slučajeva pranja te rezultati tih pranja prikazani su tablicom 5.1.

Tablica 5.1. Rezultati pranja rublja.

Vrsta rublja	Tip mrlje	Početni volumni udio mrlje	Vol. udio mrlje nakon pranja (bez sredstva za pranje)	Vol. udio mrlje nakon pranja (uz sredstva za pranje)	Vol. udio mrlje nakon pranja (uz sredstva i pretpranje)
Majica M, 100 ml	Kečap	15 %	10,66 %	5,19 %	4,62 %
Hlače L, 500 ml	Kečap	10 %	7,18 %	5,16 %	4,6 %
Haljina S, 570 ml	Nafta	25 %	23,38 %	7,42 %	6,28 %
Krpa, 35 ml	Blato	15 %	11,55 %	5,29 %	4,67 %
Ručnik, 300 ml	Kava	15 %	12,86 %	5,57 %	4,88 %
Posteljina, 700 ml	Kava	10 %	8,67 %	5,39 %	4,75 %

Slika 5.2. Grafičko sučelje dozatora sredstva za pranje, uz primjer punjenja.

Iz rezultata pranja je vidljivo kako korištenje sredstava za pranje znatno pospješuje čišćenje, a uključivanje faze pretpranja dodatno poboljšava ishod pranja.

Korištenje različitih vrsta i količina sredstava za pranje ima različite utjecaje na ishod pranja. Na učinkovitost pranja također utječe količina rublja koje se pere te mjera njegove uprljanosti. Tvrdokornije mrlje teže su za uklanjanje i lakše se reapsorbiraju u rublje tijekom pranja, rezultirajući u nepotpunom uklanjanju iz rublja. Dodatno, pranje na višim temperaturama olakšava uklanjanje nečistoća i tako poboljšava rezultate pranja. Navedene pojave, iako raspoznatljive promatranjem programskog koda simulacije, radi vremenskih zahtjeva i pojednostavljenja sadržaja dokumenta nisu demonstrirane provođenjem simulacije i prikazom rezultata u ovom dokumentu. Navedeni primjer simulacije u tri spomenuta slučaja dovoljan je za prikaz korištenja simulatora i demonstraciju osnovnog principa pranja koji iza njega stoji.

6. ZAKLJUČAK

Simulacija složenih uređaja uvijek je predstavljala problem u području programiranja. Izrada efikasnih modela na razumljiv i održiv način je bio izazovan pothvat. Danas, pojavom sofisticiranih alata poput objektno-orijentiranih programskih jezika koji olakšavaju modeliranje i simulaciju, izrada opširnih simulatora je postala znatno pristupačnija. Postojeće simulacije često ne ulaze duboko u detalje rada uređaja na razini gradivnih komponenti već se koncentriraju na ponašanje na visokoj razini. U ovom radu izrađen je simulator koji simulira perilicu rublja preko njenih najvažnijih gradivnih komponenti. Ovaj pristup omogućuje analiziranje pojava koje se odvijaju u sklopu svake pojedine komponente.

Korištenjem objektno orijentiranih načela dizajniran je i implementiran skup nezavisnih komponenti koje zajedno čine perilicu rublja. Jasno razgraničene zadaće svake pojedine komponente uređaja i njihova međusobna interakcija demonstriraju prednosti objektno orijentiranog pristupa rješavanju problema. Dijelovi sustava razrađeni su počevši od osnovnih gradivnih elemenata koji su kasnije korišteni za izradu složenijih komponenti.

Izrađen sustav demonstriran je simuliranjem pranja nekoliko komada nečistog rublja pod različitim uvjetima u svrhu prikaza razlike u ishodima pranja. Podešavanje brojnih parametara u programskom kodu simulatora omogućuje precizno kontroliranje načina uklanjanja nečistoća iz rublja. Dobiveni rezultati se stoga mogu analizirati, a simulacija prilagođavati kako bi preciznije predstavljala stvarni proces pranja.

Daljnja moguća unaprjeđenja sustava mogu se osvrnuti na probleme poput gubitka preciznosti ubrzanjem vremena, nedostatka dvosmjernog toka, inercije pri vrtnji i slično. Potencijalno funkcionalno proširenje sustava može biti uvođenje potrošnosti komponenti što bi otvorilo nove mogućnosti analize. Primjerice, neuravnoteženost rublja u bubnju prilikom vrtnje bi mogla uzrokovati trošenje mehaničkih komponenti perilice što bi omogućilo uspoređivanje različitih načina kontrole vrtnje. Također, elementi cjevovoda bi mogli uvesti mogućnost prigušenja protoka kao rezultat nakupljanja nečistoća, i slično. U konačnici, ovaj sustav daje proširiv niz funkcionalnosti koje se osim za simulaciju perilice rublja mogu iskoristiti i pri izradi drugih simuliranih uređaja.

LITERATURA

[1] S. D. Roberts, D. Pegden, “The history of simulation modeling”, 2017 Winter Simulation Conference, IEEE, (pp. 308-323), 2017.

[2] D. P. Bischak, S. D. Roberts, “Object-oriented simulation”, 1999 Winter Simulation Conference, IEEE, (pp. 194-203), 1991.

[3] J. A. Joines, S. D. Roberts. “Fundamentals of object-oriented simulation”, 1998 Winter Simulation Conference, Proceedings (Cat. No. 98CH36274), Vol. 1, IEEE, (pp. 141-149), 1998.

[4] J. Rothenberg, "Object-oriented simulation: where do we go from here?", 1986 Winter Simulation Conference, (pp. 464-469), 1986.

[5] R. G. Sargent, “Introduction to simulation languages”, Institute of Electrical and Electronics Engineers (IEEE), 1978.

[6] P. Pluta, “The OOP Has Been Explained Wrongly to Me” [online], pawelpluta.com, 2020., dostupno na: https://pawelpluta.com/the-oop-has-been-explained-wrongly-to-me/ [29.5.2023.]

[7] pawelpluta: oop-kata [online], GitHub, 2020, dostupno na: https://github.com/pawelpluta/oop-kata [29.5.2023.]

[8] mwalenia: coffee-machine [online], GitHub, 2018, dostupno na: https://github.com/mwalenia/coffee-machine [29.5.2023.]

[9] zeeshanejaz: SmartHomeSimulator [online], GitHub, 2016, dostupno na: https://github.com/zeeshanejaz/SmartHomeSimulator [29.5.2023.]

[10] EjupiAlked: COMP1202 [online], Github, 2018, dostupno na: https://github.com/EjupiAlked/COMP1202 [29.5.2023.]

[11] Gopalan College of Engineering and Management: “Object oriented modeling and design” [online], dostupno na: https://www.gopalancolleges.com/gcem/course-material/computer-science/course-plan/sem-VII/object-oriented-modeling-and-design-10CS71.pdf [4.6.2023.]

[12] A. A. Oloufa, “Intuitive simulation modeling using object oriented constructs”, Proceedings of the 8th International Symposium on Automation and Robotics in Construction (ISARC 1991), ISSN 2413-5844, (pp. 727-736), Stuttgart, Germany, 1991.

[13] The Engineering ToolBox, “Units of Heat - BTU, Calorie and Joule” [online], 2004, dostupno na: https://www.engineeringtoolbox.com/heat-units-d_664.html [24.6.2023.]

SAŽETAK

Područje simulacije složenih uređaja predstavlja mnogo izazova u pogledu modeliranja njihove strukture i načina rada. Ovaj rad na primjeru perilice rublja demonstrira implementaciju objektno orijentiranog simulatora koji prikazuje proces pranja gdje se nečistoća s rublja postepeno uklanja uz djelovanje različitih sredstava za pranje. Razina detalja na kojoj je izveden simulator omogućuje opis funkcionalnosti svih važnih komponenti uređaja, njihovu međusobnu interakciju te njihovu sposobnost rukovanja tvarima i pojavama poput tekućine i elektriciteta. Takva izvedba simulatora omogućuje analiziranje pojava koje se odvijaju u sklopu svake pojedine komponente perilice rublja. Simulator u svakom trenutku prikazuje aktualne informacije o vanjskom i unutarnjem stanju perilice, stanju programa pranja, te stanju rublja i tekućine za pranje. Odabirom rublja, sredstava za pranje i postavki programa pranja mogu se provesti simulacije pranja i usporediti rezultati. Objektno orijentirana paradigma se pokazala pogodnom za izradu modularnog i proširivog simulatora perilice rublja.

Ključne riječi: modeliranje, objekt, perilica, simulacija, uređaj

ABSTRACT

Modeling, design and implementation of an object-oriented washing machine simulator

The field of simulation of complex devices presents many challenges in terms of modeling their structure and mode of operation. This thesis describes the implementation of an object-oriented washing machine simulator which shows the process of washing in which stains are gradually removed from laundry with the help of various additives. The level of detail at which the simulator is implemented enables functional description of all major components of the device, their coordinated activity and ability to handle physical substances and occurrences such as liquids and electricity. This kind of simulator implementation enables the analysis of events at the level of each individual component. The simulator continuously shows live information about the external and internal states of the machine, the state of the wash cycle, and the state of the laundry and washing liquid. By choosing laundry, adding detergent, and selecting wash cycle options, simulations can be run and their results compared. The object-oriented paradigm has shown to be useful in constructing a modular and extensible laundry machine simulator.

Keywords: device, laundry machine, modeling, object, simulation

ŽIVOTOPIS

Dominik Živko rođen je u Požegi 18.3.1999. godine gdje je završio osnovnu školu i srednju tehničku školu sa zanimanjem Tehničar za računalstvo. Godine 2018. upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Godine 2021. upisuje diplomski studij programskog inženjerstva na istom fakultetu. Tijekom pohađanja diplomskog studija obavlja stručnu praksu kao Android Developer u tvrtki COBE d.o.o. gdje ostaje zaposlen preko studentskog ugovora.

Posjeduje višegodišnje iskustvo samostalnog rada u programskim jezicima Java i Kotlin izrađujući desktop i mobilne aplikacije za Android platformu. Također posjeduje iskustvo i interes za rad u Unity okruženju, te osnovno poznavanje jezika C#, C++, Python i JavaScript. Pri radu se aktivno služi engleskim jezikom na naprednoj razini.

PRILOZI

Prilog 1. Izvorni kod programskog rješenja (*laundry-simulator-1.0.0-source.zip*).
Kod je također dostupan na GitHub repozitoriju:
https://github.com/dzivko1/laundry-simulator

Prilog 2. Instalacijska datoteka programskog rješenja za Windows operacijski sustav (*laundry-simulator-1.0.0.msi*).