MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

MODELAREA UNUI SISTEM DE CONTROL DISTRIBUIT PENTRU O ÎNCĂPERE FOLOSIND FLETPN

PROIECT DE DIPLOMĂ

Autor: **Robert Emanuel LUCACI**

Conducător ştiinţific: **SL.dr.ing. Octavian CUIBUS**

2024

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Vizat,

DECAN
Prof. dr. ing. Mihaela DÎNȘOREANU

DIRECTOR DEPARTAMENT AUTOMATICĂ
Prof. dr. ing. Honoriu VĂLEAN

Autor: **Robert Emanuel LUCACI**

Modelarea unui sistem de control distribuit pentru o încăpere folosind FLETPN

1. **Enunţul temei:** *Modelarea unui sistem distribuit cu regulatoare modelate cu rețele FLETPN pentru a controla și monitoriza temperatura unei încăperi*

2. **Conţinutul proiectului:** *Capitolul 1. Introducere, Capitolul 2. Studiu bibliografic Capitolul 3. Analiză, proiectare, implementare, Capitolul 4. Scenarii și testare aplicație, Capitolul 5. Concluzii, Capitolul 6. Bibliografie*

3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca*

4. **Consultanţi:**

5. **Data emiterii temei:** 01.10.2023

6. **Data predării:** 11.07.2024

Semnătura autorului _____

Semnătura conducătorului ştiinţific _____

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Declarație pe proprie răspundere privind

autenticitatea proiectului de diplomă

Subsemnatul(a) **Robert Emanuel LUCACI**, legitimat(ă) cu CI seria SM nr. 926039, CNP 5020321303915,

autorul lucrării:

Modelarea unui sistem de control distribuit pentru o încăpere folosind FLETPN

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializarea **Automatică și Informatică Aplicată (la Satu Mare),** din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie 2024 a anului universitar 2023-2024, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data

11.07.2024

Robert Emanuel LUCACI

(semnătura)

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

SINTEZA

proiectului de diplomă cu titlul:

Modelarea unui sistem de control distribuit pentru o încăpere folosind FLETPN

Autor: **Robert Emanuel LUCACI**

Conducător ştiinţific: **SL.dr.ing Octavian CUIBUS**

1. Cerinţele temei: Realizarea unui model al unui sistem capabil să controleze şi să monitorizeze reglarea temperaturii într-o încăpere

2. Soluţii alese: Soluţia include o aplicaţie software care modelează componentele de control prin reţele FLETPN

3. Rezultate obţinute: Se obţine un model al unui sistem distribuit capabil să controleze reglarea temperaturii şi să monitorizeze un set de performanţe

4. Testări şi verificări: Se testează reziliența aplicației prin conectarea/deconectarea regulatoarelor, se testează funcţionalităţile de bază implementate la nivelul interfeţei grafice, se verifică şi se analizează performanţele obţinute

5. Contribuţii personale: Realizarea aplicaţiei demonstrative care permite testarea componentelor prezente în modelul sistemului, posibilitatea de a urmări modificările de sistem prin vizualizarea diferitelor elemente precum grafice sau comenzi pentru a urmări comportamentul, implementarea unor modele matematice care să permită reglarea pe parcursul fiecărui anotimp

6. Surse de documentare: articole IEEE, cursuri/laboratoare SCD, surse web

Semnătura autorului _____

Semnătura conducătorului ştiinţific _____

Cuprins

1 Introducere

1.1 Context general

Reglarea temperaturii într-un spaţiu închis reprezintă un sistem esenţial pentru a asigura numeroase beneficii care influenţează în mod semnificativ calitatea vieţii şi bunăstarea persoanelor. Aceste sisteme prezintă astfel un rol esenţial care contribuie la îmbunătăţirea calităţii vieţii prin existenţa lor în diferite clădiri precum: locuinţe personale, spaţii medicale, clădiri comerciale, fabrici industriale cât şi o serie de diferite locaţii diverse utilizate în mediul cotidian, astfel gradul de răspândire al acestui tip de reglare motivează dorinţa de dezvoltare a lucrării curente.

În contextul climatic actual, reducerea consumului de gaz ca materie primă poate contribui semnificativ la protejarea mediului înconjurător şi, în consecinţă, costurile financiare personale se pot reduce semnificativ. Faptul că în ultimii ani s-au prevăzut creşteri însemnate la nivelul costurilor energetice, progresul tehnologic de a realiza un sistem de reglare devine cu atât mai relevant.

Conform unui studiu realizat în Germania, o treime din energia consumată este utilizată pentru încălzirea locuinţelor personale, fapt pentru care optimizarea acestui sistem poate duce la o reducere semnificativă a consumului de gaz pentru utilizarea acestuia în alte domenii de interes actual. Prin faptul că cel mai comun mod de a controla temperatura în cameră cu ajutorul unei valve termostatice acţionată manual, se justifică nevoia de inovare în domeniu[1].

În prezenta lucrare se propune utilizarea reţelelor Fuzzy Logic Enhanced Time Petri Nets(FLETPN) pentru a permite modelarea şi utilizarea regulatoarelor prezente în modelul centralei termice. Acestea se implementează sub structura unui sistem distribuit pentru a permite funcţionarea sistemului chiar şi în lipsa unui regulator din cele existente, fapt ce duce totodată la reducerea complexităţii fiecărei componente.

Structura lucrării prezente este secţionată sub forma următoare:

1. Introducerea are ca scop deprinderea subiectului abordat prin prezentarea contextului şi a necesităţii de dezvoltare a domeniului prin propunerea unui set de obiective
2. Studiul bibliografic urmăreşte analizarea tehnologiilor existente pentru a putea identifica problemele şi limitările actuale
3. Analiza, proiectarea şi implementarea prezintă realizarea detaliată a lucrării propuse prin oferirea informaţiilor necesare pentru a o reproduce
4. Concluziile constau în determinarea obiectivelor realizate în urma implementării şi prezentarea posibilelor direcţii de dezvoltare

1.2 Obiective

Obiectivul principal al proiectului constă în proiectarea şi testarea unui control al temperaturii realizat cu ajutorul unor regulatoare PID care contribuie la reducerea consumului de resurse prin utilizarea reţelelor FLETPN. Fiecare regulator ce intră în setul de componente al sistemului reprezintă un nod al sistemului distribuit, noduri care au capacitatea de a comunica între ele. Fiecare regulator este independent şi poate regla sistemul în mod independent, însă prin realizarea comunicării dintre acestea, ele pot să colaboreze cu scopul atingerii unei reglări mai eficiente.

Reţele Petri avansate permit implementarea unui sistem de control hibrid prin modelarea componentelor unei centrale termice care necesită utilizarea unor evenimente discrete la nivel de transmitere a semnalelor în acord cu ajustarea temperaturii care este un proces continuu în timp. Utilizarea acestor reţele împreună cu logica fuzzy provine din necesitatea de a implementa locaţii ce pot reprezenta reacţii sincrone precum compararea temperaturii actuale cu cea setată pe termostat cât şi reacţii asincrone cum ar fi decizia autonomă a termostatului care anticipă o scădere bruscă de temperatură şi, în consecinţă, poate începe încălzirea camerei în prealabil.

Un sistem distribuit permite divizarea obiectivelor şi a resurselor pentru a facilita şi îmbunătăţii implementarea sistemelor complexe. Principalul avantaj care justifică utilizarea acestui tip de implementare este dat de faptul că prin reducerea complexităţii, se previn substanţial posibilele defecte iar dacă unul dintre componente devine nefuncţional, serviciul oferit continuă să opereze în ciuda defecţiunii locale. Acest model de implementare a apărut odată cu necesitatea descentralizării sistemelor cu o capacitate computaţională extrem de ridicată care duce la diferite probleme, printre care: problema alocării resurselor şi apariţia timpului de răspuns ridicat.

Pentru realizarea proiectului se dorește implementarea unei centrale termice prin realizarea unui set de ecuaţii matematice care să descrie comportamentul fiecărei componente prezente. La nivel de control apar 3 regulatoare, fiecare cu o atribuţie diferită: un termostat pentru modelul camerei, un regulator pentru controlul temperaturii apei care circulă prin instalaţie şi un regulator care are ca scop gestionarea unui aer condiţionat.

O interfaţă de utilizator este necesară pentru a putea interacţiona în mod direct cu întreg sistemul prin ajustarea temperaturii cât şi să acorde posibilitatea de monitorizare a datelor şi analizarea acestora cu scopul de a identifica parametrii precum consumul mediu sau perioada maximă în care o componentă este activă.

1.3 Specificaţii

În acest subcapitol se evidenţiază cerinţele pe care sistemul trebuie să le îndeplinească. În primul rând se menţionează faptul că aplicaţia trebuie realizată sub forma unui model de sistem distribuit pentru a permite ca fiecare regulator să aibă posibilitatea să funcţioneze independent. În consecinţă, prin distribuirea sistemului, se

dorește realizarea unei rețele de comunicare care să permită regulatoarelor să funcționeze împreună pentru a putea regla modelul centralei termice.

În continuare se dorește identificarea componentelor care se află în sistem pentru a putea realiza ulterior un model matematic al fiecărei componente. Conform diagramei componentelor din Figura 1.3.1 se pot identifica intrările și ieșirile fiecărui element.

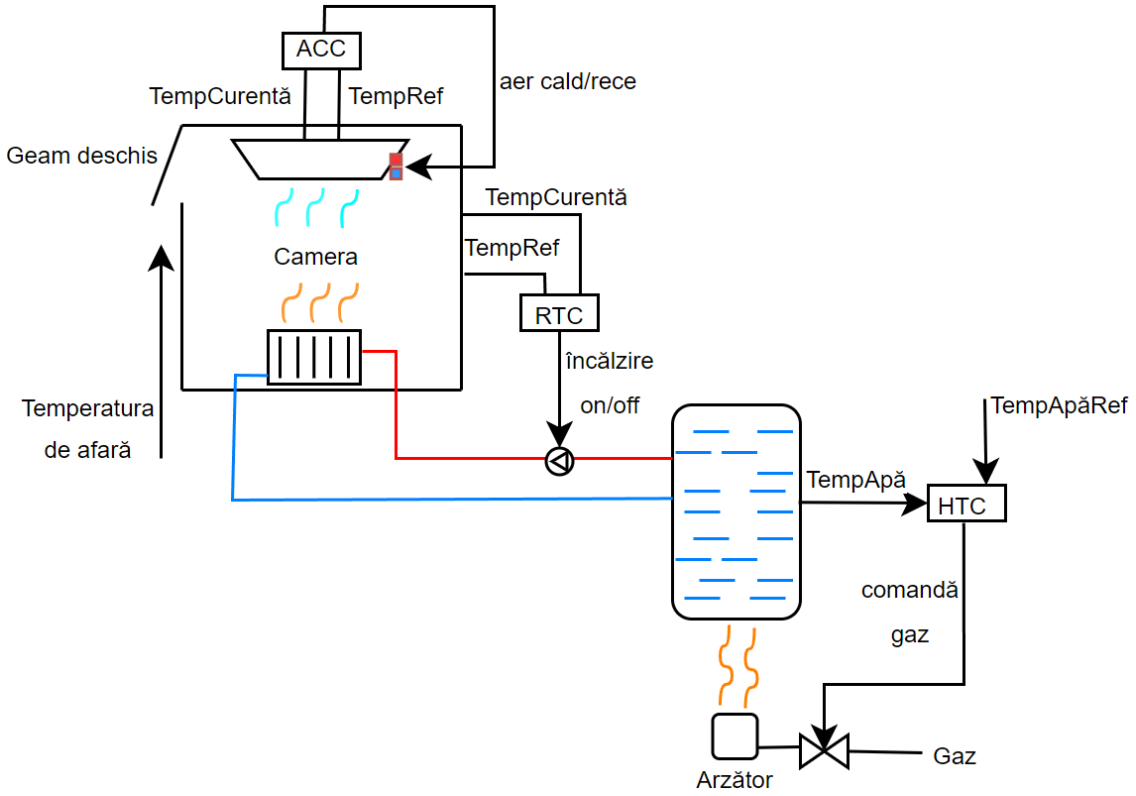

Figura 1.3.1. Diagrama componentelor

La nivelul diagramei de mai sus se observă apariția modelului asupra căreia se va realiza reglarea. Regulatorul RTC dorește să primească două valori: temperatura de referință dorită și temperatura curentă a camerei deoarce acesta va trebui să calculeze eroarea cu scopul de a determina dacă este necesară pornirea încălzirii. În cazul în care eroarea nu este nulă, se dorește ca agentul termic să fie circulat prin calorifer. Regulatorul HTC trebuie să realizeze reglarea temperaturii la nivelul boilerului, în funcție de eroarea calculată acesta trebuie să fie capabil să ofere la ieșire o comandă de gaz corespunzătoare care să acționeze asupra valvei care permite trecerea gazului de ardere. Regulatorul ACC se ocupă de reglarea aerului condiționat din încăpere și în funcție de eroarea măsurată se dorește ca acesta să comande dacă dispozitivul asigură aer rece sau aer cald. Totodată, se dorește ca sistemul să fie capabil să rejecteze perturbații exterioare precum deschiderea geamului din încăpere raportat la temperatura de afară.

Datorită apariției acestor elemente se remarcă necesitatea introducerii unei serii de regulatoare pentru a realiza controlul asupra sistemului dat, fapt pentru care se construiește o diagramă de reglare care să evidențieze schema de control împreună cu conexiunile corespunzătoare conform modelului prezentat în Figura 1.3.2.

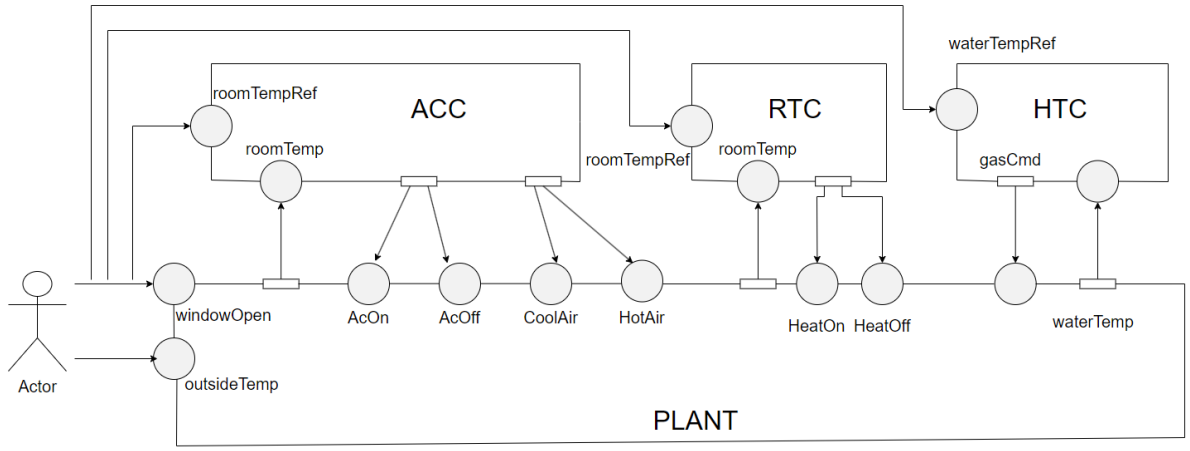

Figura 1.3.2. Arhitectura aplicaţiei

În această schemă se pot observa cele 3 regulatoare care intră în componenta sistemului şi care acţionează asupra centralei termice cu un set de comenzi. La nivelul regulatorului HTC se observă că acesta primeşte la intrare valoarea curentă a temperaturii apei, cât şi valoarea de referinţă la care se doreşte să se ajungă. Cu privire la ieşirea acestui regulator, se observă că se transmite o comandă de gaz care arde cu scopul de a încălzi temperatura apei până la valoarea dorită. În mod similar se poate observa ce intrări, respectiv ce ieşiri prezintă fiecare regulator, astfel se poate observa cu precizie rolul şi comanda pe care o are fiecare dintre acestea. Un alt set de intrări care este luat în considerare la nivelul acestei diagrame sunt perturbaţiile pe care sistemul trebuie să fie capabil să le rejecteze. Astfel, se observă de asemenea că se ia în calcul temperatura de afară şi posibilitatea ca un geam din încăpere este deschis.

Pentru ca utilizatorul să aibă capacitatea de a interacţiona şi monitoriza sistemul se ia în considerare crearea unei interfeţe grafice simple de utilizator care să permită reglarea parametrilor de intrare a regulatoarelor şi a scenariului de vreme. Deoarece sistemul se doreşte a fi realizat sub o formă distribuită, prin intermediul acestei interfeţe se poate alege pornirea sau oprirea fiecărui regulator în parte şi totodată se pot măsura şi analiza performanţele sistemului în timp real.

În urma proiectării şi implementării se anticipează un nivel de performanţă ridicat comparativ cu alte strategii comune de control termic şi se analizează ce limitări apar în urma realizării sistemului. Mai mult decât atât, se urmăreşte testarea aplicaţiei realizate pentru a se demonstra funcţionalităţile prezente şi pentru a se analiza nivelul de rezilienţă pe care această aplicatie o prezintă.

2 Studiu bibliografic

În scopul acordării unui context al proiectului prezentat în această lucrare, în acest capitol se realizează analiza conceptelor esențiale cu scopul de a studia necesitatea identificării aspectelor neexplorate în acest domeniu prin evaluarea studiilor deja existente în domeniul actual de cercetare. Lucrarea prezentă urmărește aplicarea și dezvoltarea unor concepte din cadrul sistemelor de control distribuit[2].

2.1 Logica Fuzzy

Logica reprezintă o componentă fundamentală în utilizarea gândirii raționale cu scopul realizării anumitor activități. În context matematic, logica permite utilizarea raționamentului prin valori aproximative pentru a putea judeca un incident în condiții neclare. În acest fel, logica fuzzy urmărește să explice și să permită utilizarea unei astfel de raționament neclar pentru a putea simula și utiliza sisteme de control[3]. Un exemplu relevant pentru acest proiect este atunci când o persoană intră într-o cameră: aceasta poate determina într-un timp scurt dacă atmosfera din incintă este caldă, moderată sau rece, fără a avea nevoie să ofere o valoare exactă a temperaturii. Într-o manieră similară, se utilizează variabile fuzzy care mapează setul de reguli utilizat la nivelul sistemului. Scopul acestora este de a reduce complexitatea prin reducerea numărului de variabile folosite și se pot opera logic conform condiției (2.1.1), unde „fuzzy_set1" este mulțimea valorilor pe care „input" le poate lua, iar „fuzzy_set2", mulțimea valorilor posibile pentru „acțiune" [4].

$$IF(input\ IS\ \{fuzzy\_set1\ \})THEN\ (acțiune)\ \ IS\ \{fuzzy\_set2\}) \qquad (2.1.1)$$

Setul de mărimi fuzzy poate fi ales diferit în funcție de aplicație și domeniul în care este utilizat, pentru aplicația curentă se alege mulțimea FS={NL, NM, ZR, PM, PL}, unde valorile sunt: Negative Long, Negative Medium, Zero, Positive Medium și Positive Long. Mulțimi similare pot fi alese precum FS={VHN,HN,MN,SN,ZR,SP,MP,HP,VHP}, unde terminii sunt VHP: Very High Positive, HP: High Positive, MP: Medium Positive, SP: Small Positive, ZE: Zero, SN: Small Negative, MN: Medium Negative, HN: High Negative, VHN: Very High Negative" [5], însă nivelul de complexitate al regulilor fuzzy cresc substanțial.

Pentru a exemplifica, se alege mulțimea FS={L, M, H}, unde L este Low, M este Medium și H este High, funcții care se reprezintă în acest exemplu în intervalul [-0.5,0.5]. Odată ce setul de reguli este ales, operația propriu-zisă de transformare a logicii umane sub forma unei variabile de tip fuzzy poartă numele de fuzzificare. Prin acest procedeu se realizează funcțiile de apartenență prezente în Figuria 2.1.1 pentru a determina gradul de apartenență al măsurii alese. Astfel, prin alegerea oricărei valori din interval, este posibilă determinarea corelării acesteia între două mărimi date de unde se observă conform ecuației (2.1.2).

$$U_M(x) + U_L(x) = 1 \qquad (2.1.2)$$

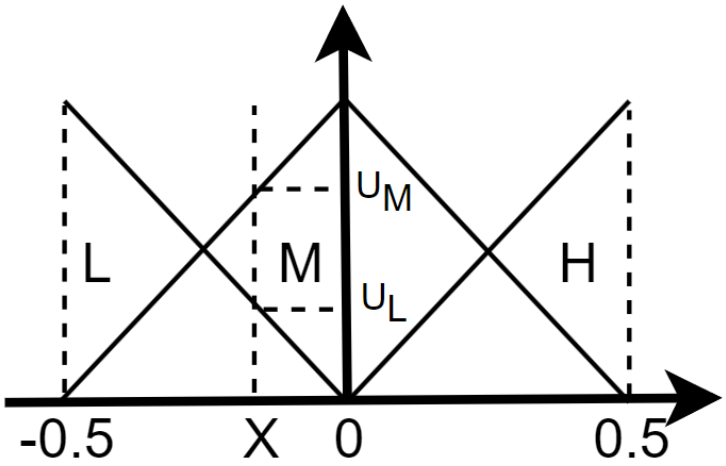

Figura 2.1.1. *Funcţii de apartenenţă*

În urma fuzzificării, scopul final este de a transforma rezultatele obţinute în valori crisp[6], care reprezintă opusul unei valori fuzzy în sensul că o astfel de valoare conţine un număr exact sau o stare clar definită. În esenţă, o valoare exactă precum 10 grade Celsius este o valoare de tip crisp iar „low" este o valoare neclară care poate fi interpretată diferit. Astfel, metoda inversă folosită pentru a realiza această conversie poartă numele de defuzzificare. Pentru a realiza acest procedeu se pot utiliza diferite metode de defuzzificare[7] iar în această lucrare se sugerează utilizarea Metodei Centrului de Greutate (MCG) deoarece aduce numeroase avantaje: este o metodă simplu de implementat fiindcă nu necesită calcule complexe, beneficiază de robusteţe prin faptul că sistemul nu este afectat uşor de zgomotul produs în datele de intrare şi este o metodă eficientă din punct de vedere computaţional. Se menţionează că nicio metodă nu este ideală iar alegerea acesteia depinde de avantajele şi limitările fiecărui sistem, însă utilizarea metodei MCG este justificată în această lucrare datorită simplităţii şi a utilizării minime de resurse computaţionale considerând că proiectul urmăreşte simularea unui control simplu de reglare a temperaturii într-o cameră. Astfel, se utilizează ecuaţia (2.1.3) unde x_i reprezintă valoarea nivelului fuzzy şi $u(x_i)$ este gradul de apartenenţă al acestei valori.

$$x = \frac{\sum_i x_i \cdot u_i(x_i)}{\sum_i u_i(x_i)} \tag{2.1.3}$$

Odată cu definirea conceptelor anterioare, se prezintă arhitectura generală a unui sistem de control fuzzy conform Figurii 2.1.2. După cum se poate observa, sistemul primeşte o valoare crisp la intrare care este fuzzificată pentru a putea fi utilizată de motorul de inferenţă[8], componenta care are scopul de a aplica regulile fuzzy asupra intrării cu scopul de a deduce informaţii noi. Logica pe care aceste motoare de referinţă le folosesc sunt în mod tipic reprezentate sub forma regulilor IF-THEN şi sunt adesea folosite în domeniul inteligenţei artificiale. În final, valoare rezultată este defuzzificată prin utilizarea metodei MCG pentru a primi o ieşire de tip valoare crisp. Valoarea finală este apoi utilizată de centrală pentru a realiza cerinţele impuse de regulator.

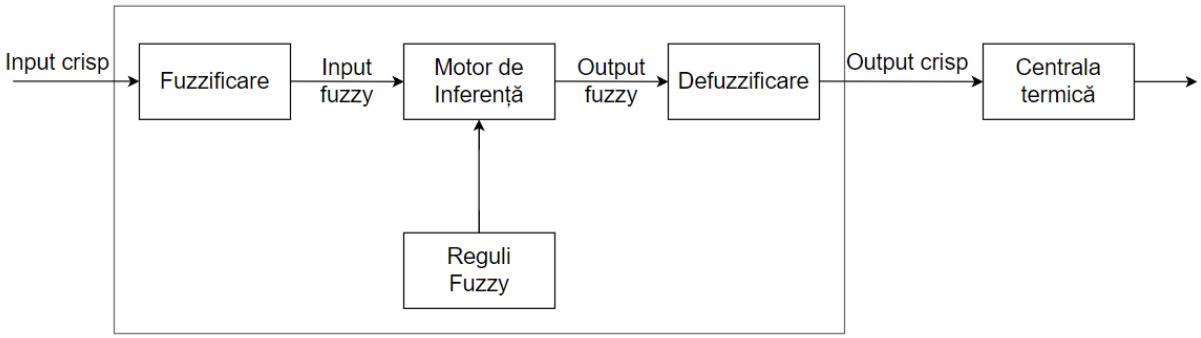

Figura 2.1.2. Arhitectura unui sistem de control de tip fuzzy

2.2 Rețele Petri

O rețea Petri [3][9] este un grafic conceput din două tipuri de componente, locații și tranziții, între care există arce de legătură. Scopul realizării unei astfel de rețele este de a reprezenta comportamentul sistemelor discrete distribuite. O locație, reprezentată grafic sub forma unui cerc, are rolul de a identifica disponibilitatea utilizării unei resurse care constă în apariția unei cantități de marcaje, cunoscute și sub numele de jetoane, în interiorul acesteia. Tranzițiile, redate sub formă dreptunghiulară, au obiectivul de a reprezenta acțiunile ce modifică starea curentă a sistemului. O astfel de tranziție poate fi activată doar în momentul în care locațiile acesteia de intrare au numărul necesar de jetoane corespunzător cu costul de transport al arcelor. Arcele sunt căile de transport ale jetoanelor între locații și tranziții iar acestea conțin un număr care semnifică costul transportului dintre cele două tipuri de elemente menționate anterior. În Figura 2.2.1 se poate observa un exemplu al folosirii elementelor unei rețele.

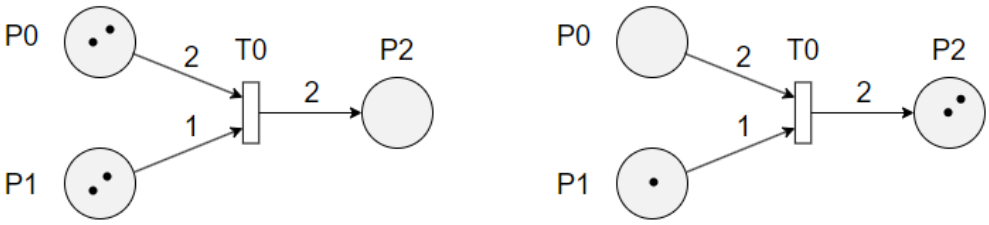

Figura 2.2.1. Exemplu de rețea Petri

Astfel o rețea Petri simplă poate fi definită sub forma ecuației (2.2.1) conform căreia P reprezintă o mulțime a locațiilor, T este mulțimea tranzițiilor, matricile pre și post semnalează existența arcelor cu costul aferent între o locație și o tranziție, respectiv între o tranziție și o locație iar M este un vector care arată numărul de jetoane din fiecare poziție.

$$PN = (P, T, pre, post, M) \tag{2.2.1}$$

O rețea Petri avansată în timp este o extensie a conceptului de rețele Petri standard deoarece conține caracteristici adiționale pentru a putea modela și analiza sisteme mult mai complexe. În literatură acest concept este cunoscut sub numele de Enhanced Time Petri Nets(ETPN) iar diferențele comparativ cu rețelele simple se pot observa din definiția (2.2.2) unde D reprezintă un set de valori naturale care marchează întârzierile care pot fi impuse pe tranziții.

$$PN = (P, T, pre, post, D, I, O, M) \quad (2.2.2)$$

Acest lucru înseamnă că fiecare tranziție t_i poate fi întârziată cu o unitate de timp d_i, caracteristică care permite implementarea unui sistem hibrid[10] deoarece în lucrarea curentă apare necesitatea de a utiliza un sistem care are atât elemente discrete cu privire la transmiterea de semnale de comandă, cât şi evenimente continue precum reglarea temperaturii care este dependentă de timp. Seturile de mulțimi I şi O reprezintă seturile de locații de intrare, respectiv de ieşire care sunt diferite față de locațiile normale prin faptul că se aşteaptă după o serie de evenimente, cum ar fi un eveniment care are loc în centrală.

O rețea ETPN capabilă să proceseze informație fuzzy poartă numele de Fuzzy Logic Enhanced Time Petri Net(FLETPN)[3][11]. Utilizarea acestor rețele provine din nevoia de a implementa sisteme hibrid, mai mult de atât, sistemul trebuie să fie capabil să răspundă la semnale asincrone cât şi sincrone pentru a putea reacționa la evenimentele discrete sau la modificările apărute la nivelul centralei termice. Conform definiției date de relația (2.2.3),o rețea FLETPN conține în plus următoarele elemente: coeficientul de greutate W aferent arcelor are rolul de a indica importanța regulii fuzzy, X este un set de variabile în domeniul real, FS este setul fuzzy, FLRS sunt regulile fuzzy, componenta Exp are rolul de a atribui fiecărei tranziții o expresie logică iar α, β, γ au rolul de a mapa diferite tipuri de variabile.

$$FLETPN = (P, T, pre, post, D, W, X, FS, FLRS, Exp, \alpha, \beta, \gamma, M) \quad (2.2.3)$$

În Figura 2.2.2 se află un model de rețea FLETPN care conține următoarele elemente noi în comparație cu o rețea Petri simplă: locații de intrare care reprezintă o valoare primită de la centrală, tranziții precum T0 şi T1 care conțin o regulă fuzzy, tranziția T2 este o tranziție întârziată cu un interval de timp şi tranziția de ieşire care are rolul de a trimite o comandă mai departe în sistem. Pe arcul dintre locația P5 şi tranziția de ieşire se poate observa o valoare care reprezintă coeficientul de greutate, pe restul arcelor dacă nu este specificat atunci fiecare arc are implicit valoarea 1.

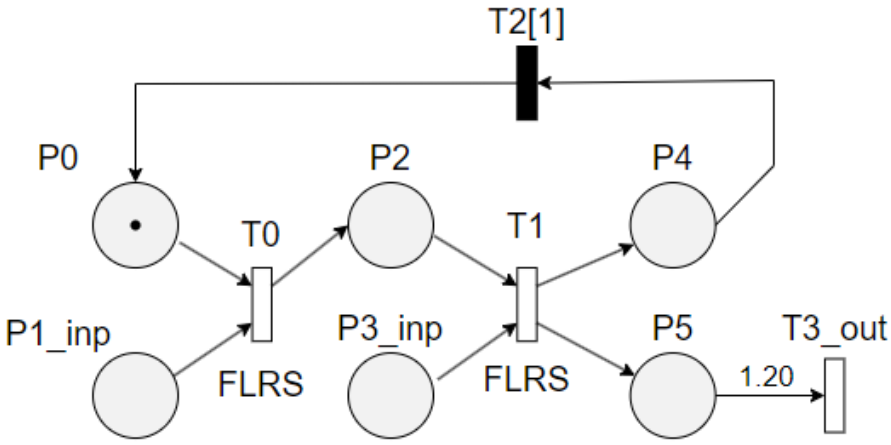

Figura 2.2.2. Exemplu de rețea FLETPN

2.3 Procese termice

Procesul termic reprezintă un fenomen fizic care semnalează transferul de căldură dintre două medii sau sisteme diferite. În contextul economic şi climatic actual se urmăreşte analizarea şi implementarea unor sisteme cât mai eficiente pentru a reduce semnificativ consumul de energie. Printre cele mai intâlnite sisteme de încălzire în Europa se află Centralele Electrice de Termoficare(CET)[12] care asigură transportul şi distribuţia energiei termice în special în mediile urbane deorece acolo există clădiri cu o densitate ridicată de locuitori. Alte alternative individuale constau în sisteme de încălzire prin arderea unor combustibili solizi precum lemnul, încălzirea cu gaze sau cu încălzire electrică [13]. În ultimii ani au apărut strategii de eficientizare a consumului precum boilere de condensare[14] care preiau vaporii de apă care în mod normal ar fi degajaţi în mediu şi ii reutilizează pentru a putea să îi transforme în căldură. Cu toate acestea, aceste sisteme nu au capacitatea de a lua în calcul diferiţi factori externi precum scăderile bruşte de temperatură sau frecvenţa utilizării geamurilor şi uşilor dintr-o încăpere.

La nivelul acestei lucrări se utilizează adesea procesul de histereză. Histereza este un fenomen cu caracter ireversibil care spune că un comportament al unui sistem nu depinde doar de stare anterioară, ci depinde şi de stările precedente pe care sistemul le suferă. Această funcţie de histereză este regăsită în majoritatea termostatelor[15], cât şi în diferite sisteme de climatizare cu aer condiţionat. Avantajul principal pe care îl prezintă acest fenomen este acela că, prin ajustarea setărilor unui termostat, se poate maximiza eficienţa energetică. Regulatoarele care reproduc fenomenul de histereză se numesc regulatoare bipoziţionale[16]. Acestea sunt folosite deoarece : când se ajunge la valoarea temperaturii dorită plus specificaţia de histereză(de exemplu 0.5 grade Celsius) atunci reglarea se opreşte, când se ajunge la valoarea temperaturii dorite minus specificaţia de histereză centrala porneşte din nou. Acest lucru este de dorit fiindcă o funcţionare neîncetată a centralei sau pornirea şi oprirea acesteia cu o frecvenţă ridicată poate duce nu doar la creşterea consumului, dar poate duce şi la uzarea echipamentelor.

Astfel un sistem de reglare al temperaturi implementat cu reţele FLETPN[17] este ales pentru a fi implementat în această lucrare deoarece acesta asigură o eficienţă energetică ridicată prin ajustarea dinamică a temperaturii cu ajutorul logicii fuzzy. Totodată, capacitatea de rejectare a perturbaţiilor externe şi posibilitatea de integrare cu alte sisteme de gestionare a clădirilor justifică importanţa dezvoltării unui asftfel de sistem.

2.4 Sisteme distribuite

Sistemele distribuite[18] acordă capacitatea de a gestiona şi implementa sisteme complexe, prin divizarea obiectivelor şi a resurselor pentru a facilita şi îmbunătăţi funcţionarea lor. Avantajul principal pe care îl prezintă un astfel de sistem este posibilitatea de a reduce nivelul de complexitate, prevenind astfel apariţia defectelor şi permiţând ca serviciul oferit să funcţioneze în continuare în ciuda unei defecţiuni la nivel local.

Apariţia acestui tip de sistem a apărut odată cu necesitatea de a descentraliza sisteme cu capacităţi computaţionale ridicate pentru a gestiona provocările aduse de un sistem centralizat complex, mai exact gestionarea resurselor computaţionale şi asigurarea unui timp de răspuns rapid, care poate prezenta o importanţă crucială în diferite domenii[19]. Prin împărţirea sarcinilor şi resurselor între mai multe componente sau noduri, sistemele distribuite permit o mai mare flexibilitate. deoarece sistemul poate creşte în dimensiune şi capacitate fără a fi necesară o revizuire completă a arhitecturii sau a infrastructurii.

Pe lângă aspectele tehnice, sistemele distribuite se bazează pe principii de descentralizare care permit nodurilor să opereze independent prin luare unor decizii locale în funcţie de nevoile specifice. Această abordare contribuie la reducerea dependenţei de un singur nod şi asigură astfel o robusteţe ridicată a sistemului.

2.4.1 Arhitectura client-server

Arhitectura client-server[20] este cel mai comun mod de proiectare a unei aplicaţii şi este alcătuit din două elemente esenţiale, mai exact clientul şi serverul. Clientul este componenta care iniţiază cereri cu scopul de a accesa o serie de servicii sau resurse disponibile din incinta serverului. Serverul este componenta care se ocupă ce procesarea cererilor primite de la clienţi şi transmite înapoi resursele cerute, procedeu care poate fi vizualizat în Figura 2.4.1.

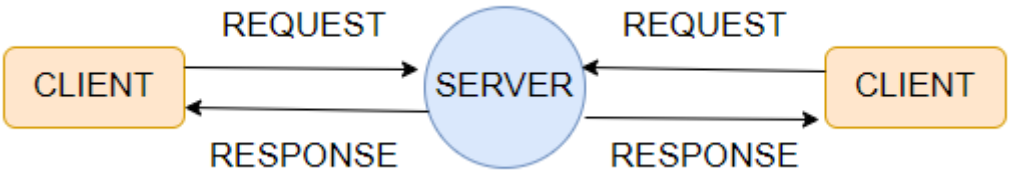

Figura 2.4.1. Arhitectura client-server

Acest tip de arhitectură este folosit deoarece aduce următoarele avantaje[21]:

- Clientul şi serverul pot fi implementaţi pe dispozitive diferite pentru a aduce o flexibilitate mai mare proiectului
- Prin faptul că sunt două elemente diferite, acestea nu necesită cunoaşterea detaliilor de implementare interne ale fiecăruia fiindcă interacţiunea dintre componente are loc prin intermediul unor protocoale
- Utilizare unei astfel de arhitecturi permite extinderea ulterioară prin adăugarea mai multor clienţi sau server în funcţie de noile cerinţe ale proiectului
- Poate aduce un nivel de securitate în plus prin crearea unei metode de autentificare pentru a se realiza conexiunea între client şi server

2.5 Principii OOP

Aplicaţia este realizată cu ajutorul limbajului Java care se bazează pe principiile Object Oriented Programming(OOP)[22]. Aceste principii sunt utilizate la nivel de

implementare a aplicaţiilor deoarece prin acest mod de gândire se realizează organizarea codului în aşa fel încât se permite reutilizarea acestuia cât mai uşor. Totodată, prin realizarea unui program care conţine obiecte ce interacţionează utilizatorul, urmărirea acestor principii permite ca proiectul să fie realizat eficient şi să fie uşor de înţeles.

Primul principiu care stă la baza OOP îl reprezintă abstractizarea. Acest proces permite ascunderea unor detalii din interiorul unei clase care nu sunt relevante. Principiul este foarte important la nivelul unei echipe deoarece dacă un programator primeşte o anumită cerinţă, restul echipei nu este interesată de modul de creare a noii funcţionalităţi, ci doar de folosirea acesteia.

Al doilea principiul îl prezintă încapsularea, proces care permite ascunderea datelor şi funcţiilor faţă de mediul exterior. La nivelul atributelor şi funcţiilor există astfel diferite tipuri de modificatori: public permite accesarea de oriunde, protected permite accesarea metodei/atributului din interiorul clasei principale şi al claselor derivate din aceasta şi modificatorul private semnifică faptul că proprietatea sau metoda este accesibilă doar în interiorul clasei.

Moştenirea este principiul care îi permite unei clase să extindă funcţionalitatea unei clase de bază prin utilizarea proprietăţilor acesteia. Astfel, o clasă derivată devine în consecinţă mai complexă prin faptul că poate aduce atribute şi metode noi care extind comportamentul. Prin utilizarea acestui principiu, se permite menţinerea clasei de bază fără ca aceasta să fie modificată, lucru care poate fi necesar în diferite situaţii la nivelul unui proiect.

Ultimul principiu din listă este polimorfismul. Prin acesta se realizează procesul prin care se crează mai multe copii ale aceleaşi metode, dar cu funcţionalităţi diferite. În limbajul Java există două tipuri de polimorfism: static şi dinamic. Polimorfismul static este cunoscut sub numele de supraîncărcare a metodelor şi are loc în momentul compilării. Această supraîncărcare permite existenţa a două metode cu acelaşi nume, dar cu parametrii diferiţi. Polimorfismul dinamic are loc în momentul rulării programului şi poartă numele de suprascriere a metodelor. În acest caz, numele clasei suprascrise trebuie să fie acelaşi, totodată, numărul şi tipul parametrilor este la fel.

2.6 Design patterns

La nivelul ingineriei software, un design pattern(şablon de proiectare) este o soluţie care poate fi reprodusă pentru rezolva diferite probleme ce pot apărea la nivelul realizării unei aplicaţii[23]. Aceste pattern-uri nu reprezintă o secvenţă de cod fixă ci mai exact descrie un şablon care ajută la rezolvarea problemei. La nivelul acestei aplicaţii se folosesc următoarele două tipuri de şabloane: client-server şi factory.

Un pattern de tip client-server[24] are rolul de a despărţi cele două componente pentru a putea trata separat partea de distribuire a datelor(preluată de server) faţă de interacţiunile utilizatorilor(realizate la nivel de client). Acest tip de şablon este adesea întâlnit la nivelul sistemelor distribuite deoarece avantajul principal este că permite unui număr mai mare de clienţi să interacţioneze simultan cu serverul prin transmiterea unor cereri de date. Şablonul client-server prezintă însă şi diferite dezavantaje precum:

necesitatea realizării unui management de comunicare complex între clienți cu serverul și conectarea unui număr foarte mare de clienți simultan poate crește mult timpul de răspuns al serverului.

Totodată, la nivelul acestei lucrări se utilizează și pattern-ul cunoscut sub numele de factory method[25]. Acest șablon are rolul de a furniza o interfață pentru a crea obiecte într-o superclasă, cu mențiunea că acesta le permite subclaselor să modifice tipul obiectelor care sunt create. Acest design este folosit ulterior în aplicație pentru a crea un client handler unic pentru regulatorul care se dorește a fi creat, această implementare este necesară deoarece fiecare regulator comunică diferit cu modelul plantului

3 Analiză, proiectare, implementare

În capitolul curent se urmăreşte prezentarea lucrării de la analizarea nevoii de a crea un sistem eficient pentru reglarea temperaturii într-o încăpere până la implementarea propriu zisă a acestui sistem prin detalierea paşilor care duc la dezvoltarea aplicaţiei. Proiectul curent este o aplicaţie demonstrativă care realizează controlul termic asupra unui model de cameră prin utilizarea reţelelor FLETPN pentru modelarea regulatoarelor cu scopul de a spori gradul de eficienţă al consumului şi de a permite sistemului să poată reacţiona şi rejecta factori externi.

Scopul realizării acestui sistem provine din necesitatea de a găsi alternative pentru a reduce consumul de agent termic în contextul în care resursele planetei sunt limitate şi în urma conflictelor actuale dintre state, s-a pus o tensiune asupra resurselor globale de energie.În consecinţă, costul de încălzire a unei locuinţe a crescut în mod semnificativ în ultimii ani, fapt ce justifică necesitatea de a aduce inovaţii în industrie. Totodată, resursele de gaz consumate pentru încălzirea locuinţelor pot fi relocate astfel la nivel global către dezvoltare spre alte domenii de interes uman.

Obiectivele luate în calcul în privinţa realizării unui astfel de tip de sistem sunt următoarele:

- realizarea unui sistem eficient de control termic prin evaluarea performanţelor obţinute în urma testării
- posibilitatea de a menţine temperatura dorită pe parcursul unui întreg an calendaristic prin adăugarea unui sistem de încălzire a apei cât şi prin implementarea unui aer condiţionat care să poată compensa la nevoie creşterea temperaturii dar şi să poată reduce temperatura din încăpere pe parcursul perioadelor calde de vreme
- existenţa multiplelor componente duce la nevoia de distribuire a sistemului pentru a putea realiza o simplificare a proiectului şi pentru a permite doar utilizarea regulatoarelor dorite în funcţie de nevoile curente
- crearea unei interfeţe grafice pentru a permite controlul asupra centralei cu componentele conexe şi monitorizarea performanţelor

Lucrarea este realizată în mediul de dezvoltare IntelliJ IDEA care permite crearea de software prin limbaj scris în Java. Adiţional, pentru a putea desena urmări comportamentul regulatoarelor realizate cu logică fuzzy, se utilizează un vizualizator şi executor[26] care este folosit în scopuri academice pentru a putea defini structura unei reţele Petri împreună cu tabelele de reguli fuzzy incluse în tranziţii. Acesta permite observarea comportamentului reţelelor şi interacţionarea cu acestea prin intermediul unei interfeţe grafice. Pentru realizarea unei interfeţe grafice cu care să interacţioneze utilizatorul se foloseşte Java Swing din intermediul mediului de dezvoltare ales. Reprezentările grafice prezente în lucrare sunt realizate cu ajutorul a diferite medii precum Pipe pentru realizare unor reţele Petri sau alte instrumente online.

3.1 Arhitectura aplicației

Pentru a realiza proiectarea se consideră o cameră care poate fi încălzită de un boiler de apă caldă sau poate fi răcită prin utilizarea unui aer condiționat. Cele două componente trebuie să asigure temperatura dorită din încăpere prin luarea în calcul al factorilor externi precum temperatura de afară. Arhitectura sistemul este modelată în Figura 3.1.1 pentru a vizualiza mai ușor componentele și pentru a observa care sunt intrările și ieșirile aferente.

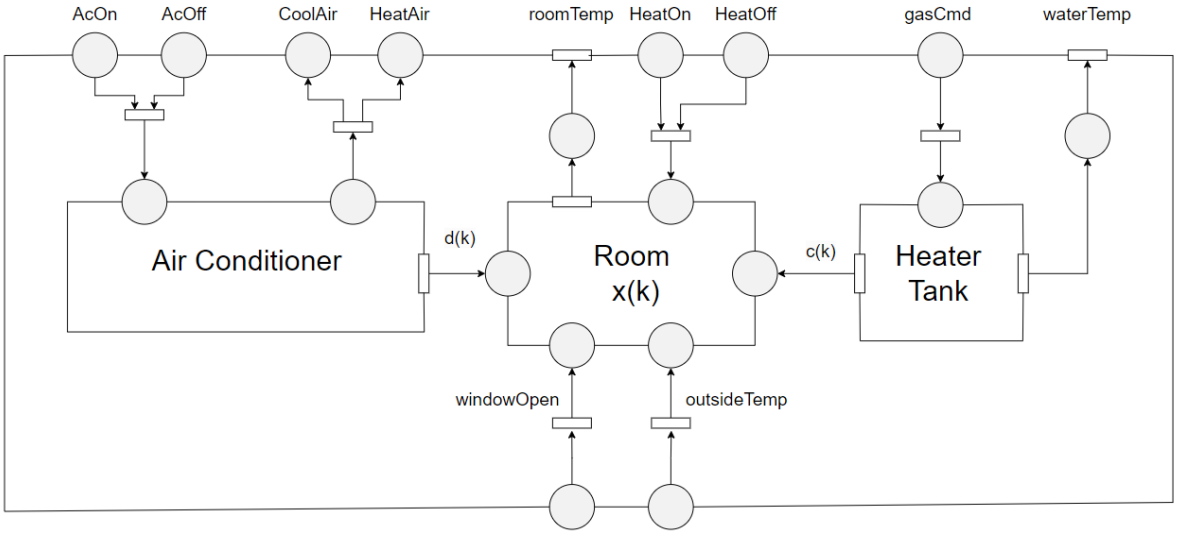

Figura 3.1.1. Diagrama componentelor cu sistemele de reglare

Componenta *Room* are rolul de a modela temperatura din cameră și este definită de următoarele porturi de intrare: *outsideTemp* indică temperatura de afară și *windowOpen* semnalează dacă o fereastră este deschisă, ambele intrări pot fi considerate ca fiind perturbații externe. Totodată, mai există două intrări, *c(k)* și *d(k)* care reprezintă căldura dată de boiler, respectiv temperatura asigurată de aerul condiționat. Ieșirea acestei componente este reprezentată de portul *roomTemp* care modelează temperatura sistemului. În urma analizări intrărilor și ieșirilor se poate crea modelul matematic pentru componenta *Room* care este dat de ecuația de stare (3.1.1).

$$t(k+1) = a * t(k) + b * c(k) + e * d(k) + f_1 * p(k) + f_2 * p(k) \qquad (3.1.1)$$

Elementul rezultat *t(k+1)* reprezintă temperatura la următorul moment de timp și este afectată de temperatura *t(k)* la momentul curent, căldura oferită de boilerul de apă *c(k),* temperatura aerului furnizată de aerul condiționat *d(k)* și cele două perturbații $f_1*p(k)$, respectiv $f_2*p(k)$, sunt afectate de temperatura externă *p(k),* unde prima perturbație este dată de deschiderea ferestrei și a doua este dată de răcirea prin pereții camerei. În ceea ce privește *a*, *b*, *e*, f_1 și f_2, aceștia reprezintă coeficienții fiecărui element menționat anterior. Intrarea *HeatOn/HeatOff* are rolul de a informa dacă boilerul este pornit sau oprit.

Componenta *HeaterTank* este boilerul care are rolul de a asigura o temperatură dată a apei pentru a o trimite prin sistemul de încălzire ca să asigure temperatura dorită în încăpere. Intrarea *gasCmd* specifică cât de tare este pornit debitul de gaz și ieșirile sunt *c(k)* care reprezintă căldura dată în cameră, respectiv *waterTemp* care are rolul de a

ajusta temperatura apei care este mai apoi circulată prin sistem. Reacţia de stare a componentei este prezentată în ecuaţia (3.1.2), unde $t_a(k+1)$ este temperatura apei la momentul următor, $t_a(k)$ temperatura apei la momentul curent, $g(k)$ reprezintă comanda de gaz şi $p_1(k)$ este o perturbaţia care ia în considerare faptul că transmiterea apei prin ţevi poate duce la o scădere mică de temperatură. În mod similar componentei anterioare, a_1, b_1 şi c_1 sunt coeficienţii aferenţi fiecărui element.

$$t_a(k+1) = a_1 * t_a(k) + b_1 * g(k) + c_1 * p_1(k) \quad (3.1.2)$$

Componenta *AirConditioner* este aerul condiţionat care în funcţie de temperatura din încăpere asigură aer rece sau cald pentru a asigura temperatura dorită. Porturile de comandă ale acestei componente sunt: *AcOn/AcOff* care are rolul de a porni sau opri aerul condiţionat şi *CoolAir/HeatAir* decide dacă aerul eliberat în cameră este rece sau cald. Portul de ieşire *d(k)* este temperatura pe care aerul condiţionat o aplică asupra camerei. Ecuaţia de stare (3.1.3) descrie comportamentul componentei *AirConditioner,* unde $t_c(k+1)$ este temperatura aerului la momentul următor, $t_c(k)$ reprezintă temperatura aerului la momentul curent şi $t_d(k)$ este diferenţa de temperatură dintre minimul, respectiv maximul de temperatură şi temperatura asigurată la momentul curent în cameră. În ecuaţie sunt prezenţi şi doi coeficienţi a_2 şi b_2 care modifică influenţa elementelor prezentate anterior.

$$t_c(k+1) = a_2 * t_c(k) + b_2 * t_d(k) \quad (3.1.3)$$

Pentru a putea controla cele trei componente prezente în sistem, este nevoie de crearea unui regulator pentru fiecare: Room Temperature Controller(RTC) pentru componenta Room, Heater Tank Controller(HTC) pentru boiler şi Air Conditioner Controller(ACC) pentru aerul condiţionat. Conform Figurii 3.1.2 prezente mai jos, componenta RTC menţine temperatura cât mai aproape de referinţa *roomTempRef* acordată de către actor, componenta HTC asigură încălzirea apei până la temperatura dorită şi ACC oferă temperatura aerului eliberat astfel încât să se ajungă la valoarea dorită.

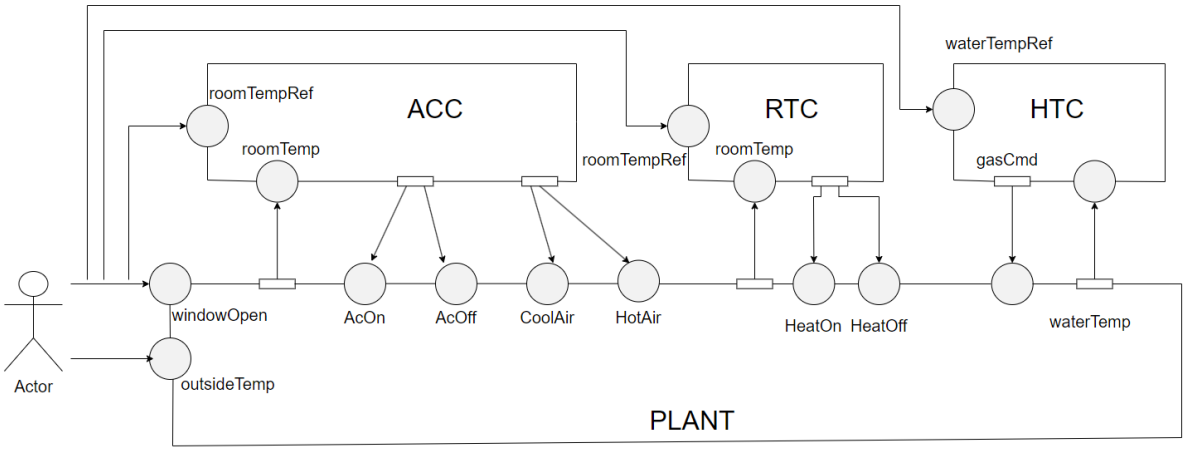

Figura 3.1.2. Diagrama de reglare a sistemului

În mod similar cu diagrama componentelor, regulatoarele au porturile de intrare şi ieşire similare pentru a putea realiza controlul adecvat al fiecărei componente din

sistemul de Plant. În continuare se explică realizarea fiecărui controller prin reţeaua Petri aferentă.

3.1.1 Componenta RTC

În figura 3.1.3 este realizat modelul Petri al regulatorului de temperatură al camerei RTC, unde fiecare locaţie şi tranziţie au o anumită semnificaţie:

- P1_inp şi P3_inp sunt locaţiile de intrare care primesc valoarea de referinţă a temperaturii din cameră *roomTempRef*, respectiv valoarea curentă *roomTemp*
- T0 are rolul de a citi valoarea referinţei pentru a o stoca în locaţia P2
- T1 realizează diferenţa dintre valoare de referinţă şi valoarea curentă a temperaturii pentru a calcula eroarea
- T2 este o tranziţie cu întârziere de un tact şi are ca scop reluarea ciclului
- T3 este un comparator care analizează dacă eroarea calculată anterior este NL, caz în care se introduce un jeton în locaţia P6, în caz contrar când eroarea este PL se adaugă un jeton în locaţia P7, acest fenomen are ca scop crearea unui proces de histereză prin pornirea încălzirii în momentul în care temperatura atinge pragul de *roomTempRef+E* şi oprirea acesteia când temperatura ajunge la valoarea *roomTempRef-E*, unde E reprezintă valoarea ponderii arcului dintre locaţia P5 şi T3

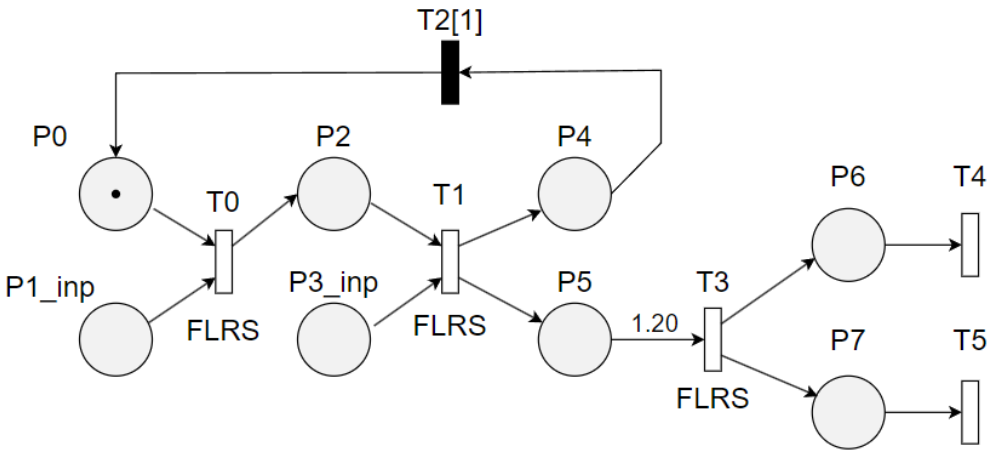

Figura 3.1.3. Componenta RTC: regulatorul temperaturii camerei

În cele ce urmează se prezintă regulile fuzzy prezente în tranziţiile din figura de mai sus. Pentru a putea realiza un tabel de reguli, este necesară determinarea numărului de intrări şi ieşiri ale acţiunii, pentru ca mai apoi să se modeleze comportamentul dorit. Pe parcursul întregii lucrări se încearcă utilizarea a maxim 2 intrări şi 2 ieşiri pe tranziţie pentru ca implementarea să fie mai simplă deoarece pentru o singură intrare tabelul de reguli este un vector linie, pentru 2 intrări se creează un tabel iar pentru 3 sau mai multe intrări este necesară reprezentarea în spaţiu a regulii respective. Pe parcursul acestei lucrări, regulatoarele prezente utilizează unele tranziţii în mod similar, fapt pentru care cele care se repetă sunt explicate o singură dată, după care se va realiza o simplă referire ulterioară a acestora.

Asftel că, în tranziţia T0 se află 2 intrări, fapt ce semnifică că se realizează Tabelul 3.1.1 cu o singură ieşire. Scopul acestei tranziţii este acela de a citi valoarea introdusă din locaţiile anterioare, fapt pentru care se începe prin a marca tabelul cu variabilele fuzzy din mulţimea FS = {NL, NM, ZR, PM, PL} atât pe linii cât şi pe coloane. Completarea regulii este simplă în acest caz deoarece se doreşte ca variabila de ieşire să reprezinte valoarea dată la intrare care este marcată pe coloană. De exemplu, dacă se doreşte citirea valorii ZR, atunci întreaga coloană a tabelului se marchează cu valoarea cerută.

Tabelul 3.1.1 Regula Fuzzy pentru tranziţia T0 de citire a valorii

Cititor valori	**NL**	**NM**	**ZR**	**PM**	**PL**
NL	NL	NM	ZR	PM	PL
NM	NL	NM	ZR	PM	PL
ZR	NL	NM	ZR	PM	PL
PM	NL	NM	ZR	PM	PL
PL	NL	NM	ZR	PM	PL

Tranziţia T1 prezintă două intrări cu două ieşiri şi aceasta realizează diferenţa dintre valoarea temperaturii de referinţă cu temperatura curentă din cameră pentru a calcula eroarea care este trimisă mai departe pe ieşiri. În acest caz, forma corespunzătoare a regulii este prezentată în Tabelul 3.1.2 care descrie comportamentul de diferenţiator al tranziţiei T1. Pe liniile şi coloanele tabelului s-au marcat valorile corespunzătoare fiecărei variabile fuzzy din mulţimea FS pentru a observa cu mai multă uşurinţă procedeul de calcul al ieşirilor.

Tabelul 3.1.2 Regula Fuzzy pentru tranziţia T1 de diferenţiere a intrărilor

Diferenţiator x_1/x_2	**NL=-1**	**NM=-0.5**	**ZR=0**	**PM=0.5**	**PL=1**
NL=-1	<ZR,ZR>	<NM,NM>	<NL,NL>	<NL,NL>	<NL,NL>
NM=-0.5	<PM,PM>	<ZR,ZR>	<NM,NM>	<NL,NL>	<NL,NL>
ZR=0	<PL,PL>	<PM,PM>	<ZR,ZR>	<NM,NM>	<NL,NL>
PM=0.5	<PL,PL>	<PL,PL>	<PM,PM>	<ZR,ZR>	<NM,NM>
PL=1	<PL,PL>	<PL,PL>	<PL,PL>	<PM,PM>	<ZR,ZR>

Pentru a înţelege modalitatea de calcul se reia în vizor funcţia de fuzzificare din Figura 3.1.4 care arată că fiecare variabilă fuzzy este aferentă cu o valoare din intervalul [-1,1], astfel că procedeul de marcare a variabilelor este următorul: NL=-1, NM=-0.5 şi analog pentru restul variabilelor. Astfel pentru a realiza diferenţa, se consideră cele două intrări x_1 pe linie, respectiv x_2 pe coloană şi se calculează diferenţa $d=x_1-x_2$. De exemplu, pentru linia 4, coloana 5 rezultatul este NM deoarece x_1-x_2=PM-PL=0.5-1=-0.5 care reprezintă variabila Negative Medium. Motivul pentru care ambele variabile de

ieşire prezintă acelaşi rezultat la diferenţă este pentru că această eroare este utilizată ulterior din locaţiile P4, respectiv P5.

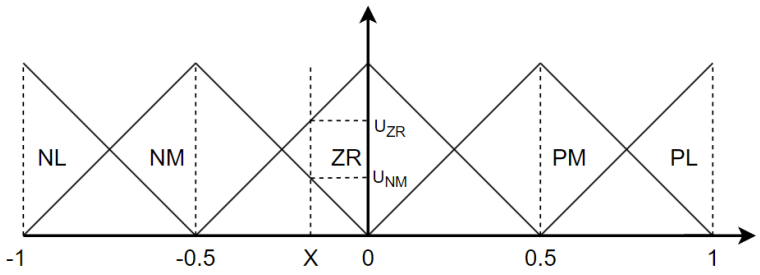

Figura 3.1.4. Funcţia de fuzzificare

Tranziţia T2 este singura din reţeaua Petri a regulatorului care are un tact de întârziere. Rolul acestei tranziţii este a transmite eroarea calculată din P4 în P0 pentru a relua ciclul de reglare atât timp cât nu s-a ajuns la valoarea dorită. Crearea unui tabel pentru această tranziţie este asigurat de utilitarul FuzzPVisual[20], iar tranziţia are tabelul realizat sub forma unei linii deoarece aceasta prezintă o intrare şi o ieşire.

Scopul tranziţiei T3 este acela de a realiza procedeul de histereză prin compararea intrării, care este eroarea, cu variabilele NL, respectiv PL. În funcţie de rezultatul obţinut, se injectează jeton fuzzy în una din cele două ieşiri. Tabelul 3.1.3 este regula pentru această acţiune are loc. Se menţionează utilizarea la ieşire a unei variabile noi FF care are ca scop marcarea faptului că ieşirea în acel punct nu este relevantă pentru funcţionalitatea dorită, acest lucru previne de altfel şi acumularea de jetoane. Spre exemplu, dacă eroarea este NL, se introduce jeton fuzzy ZR pentru prima locaţie, iar dacă aceasta are valoarea PL; jetonul este introdus în cealaltă locaţie.

Tabelul 3.1.3 Regula Fuzzy pentru tranziţia T3 cu rol de comparator

Comparator	**NL**	**NM**	**ZR**	**PM**	**PL**
Eroare	<FF,ZR>	<FF,FF>	<FF,FF>	<FF,FF>	<ZR,FF>

Ultimele două tranziţii din schemă T4, respectiv T5 sunt tranziţii de ieşire şi au ca scop preluarea jetonului fuzzy şi prezintă funcţionalitatea de a porni, respectiv de a opri căldura la nivelul centralei termice.

3.1.2 Componenta HTC

În Figura 3.1.5 este realizat modelul Petri al regulatorului de reglare a temperaturii apei din boiler HTC, unde semnificaţia porturilor şi tranziţiilor este:

- P1_inp ia valoarea de referinţă a temperaturii apei *waterTempRef*
- T0 stochează referinţa în tranziţia P2
- P3_inp citeşte valoarea curentă a temperaturii apei din boiler *waterTemp*
- T1 este un diferenţiator pe două canale care calculează eroarea la momentul curent dintre valoarea de referinţă şi valoarea curentă *err=waterTempRef-waterTemp*
- T3 este tranziţia cu întârziere care are ca scop reluarea ciclului

- T4 este un sumator dintre valoarea curentă, respectiv valoarea anterioară a comenzii către boiler
- T2 reprezintă portul de ieşire pentru comanda de gaz
- T5 are rolul de a actualiza valoarea comenzii către boiler cu o unitate de timp

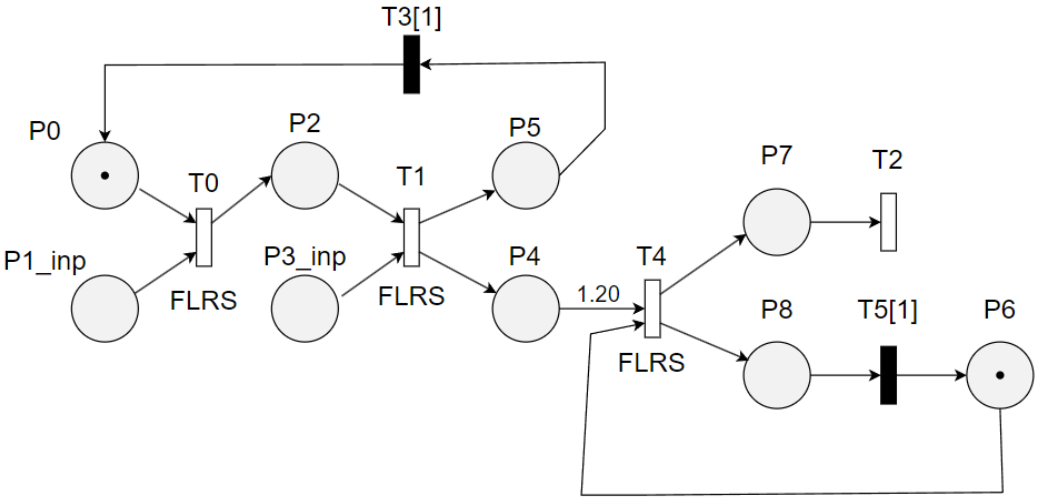

Figura 3.1.5. Componenta HTC: regulatorul apei din boiler

Tranziţiile T0, T1 şi T3 au funcţionalităţi similare cu tranziţii prezentate anterior, fapt pentru care se face doar referire la acestea: T0 este cititorul din Tabelul 3.1.1, T1 este un diferenţiator pe două canale conform Tabelului 3.1.2 şi T3 este tranziţia cu o intrare şi o ieşire care asigură reluarea ciclului după o unitate de timp în mod similar cu tranziţia T2 a regulatorului RTC. Totodată, tranziţia T5 acţionează în mod similar cu T3, deci singura tranziţie nouă pentru care este necesară explicarea creeri unei reguli este tranziţia T4.

Tranziţia T4 prezintă două intrări cu două ieşiri şi aceasta realizează suma dintre valoarea temperaturii de referinţă a apei cu temperatura curentă din boiler pentru a calcula eroarea care este trimisă mai departe pe ieşiri. În acest caz, forma corespunzătoare a regulii este prezentată în Tabelul 3.1.4 care descrie comportamentul de sumator al acestei tranziţii.

Tabelul 3.1.4 Regula Fuzzy pentru tranziţia T4 de însumare a intrărilor

Sumator x_1/x_2	**NL=-1**	**NM=-0.5**	**ZR=0**	**PM=0.5**	**PL=1**
NL=-1	<NL,NL>	<NL,NL>	<NL,NL>	<NM,NM>	<ZR,ZR>
NM=-0.5	<NL,NL>	<NL,NL>	<NM,NM>	<ZR,ZR>	<PM,PM>
ZR=0	<NL,NL>	<NM,NM>	<ZR,ZR>	<PM,PM>	<PL,PL>
PM=0.5	<NM,NM>	<ZR,ZR>	<PM,PM>	<PL,PL>	<PL,PL>
PL=1	<ZR,ZR>	<PM,PM>	<PL,PL>	<PL,PL>	<PL,PL>

Pentru a realiza operaţia de însumare, se consideră cele două intrări x_1 pe linie, respectiv x_2 pe coloană şi se calculează suma $s=x_1+x_2$. De exemplu, pentru linia 4, coloana 1 rezultatul este NM deoarece x_1+x_2=PM+NL=0.5-1=-0.5 care reprezintă variabila Negative Medium. Ambele variabile de ieşire prezintă acelaşi rezultat pentru că această eroare este utilizată ulterior din locaţiile P7, respectiv P8.

3.1.3 Componenta ACC

În Figura 3.1.6 se observă reţeaua Petri care modelează regulatorul aerului condiţionat ACC iar semnificaţia porturilor este următoarea:

- P1_inp şi P3_inp sunt locaţiile de intrare care primesc valoarea de referinţă a temperaturii din cameră *roomTempRef*, respectiv valoarea curentă *roomTemp*
- T0 are rolul de a citi valoarea referinţei pentru a o stoca în locaţia P2
- T1 realizează diferenţa dintre valoare de referinţă şi valoarea curentă a temperaturii pentru a calcula eroarea
- T2 este o tranziţie cu întârziere de un tact şi are ca scop reluarea ciclului
- T3 este un cititor care transmite eroarea mai departe în locaţiile P6 şi P7
- T4 este un comparator care analizează dacă eroarea calculată anterior este NL, caz în care se introduce un jeton în locaţia P6, în caz contrar când eroarea este PL se adaugă un jeton în locaţia P7 pentru a crea efectul de histereză
- T5 compară dacă eroarea dintre valoarea de referinţă şi cea curentă este mai mare sau mai mică decât 0 pentru a putea determina dacă se doreşte eliberarea de aer rece sau cald de la aerul condiţionat
- T6 şi T7 sunt tranziţiile de ieşire care acţionează la pornirea şi oprirea aerului condiţionat
- T8 şi T9 sunt tranziţii de ieşire care setează dacă aerul eliberat este cald sau rece

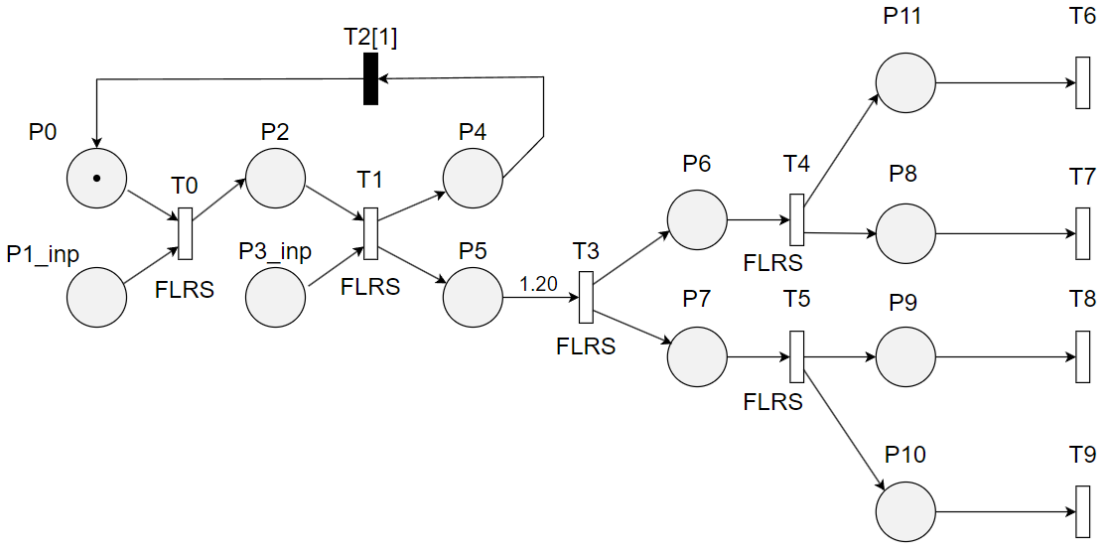

Figura 3.1.6. Componenta ACC: regulatorul aerului condiţionat

În Tabelul 3.1.5 de mai jos se descrie comportamentul tranziţiei T5 care are ca scop introducerea unui jeton in P9 atunci când eroarea este pozitivă pentru a urma să atribuie comanda de aer rece şi în caz contrar se injectează un jeton în P10 pentru a genera aer cald. După cum se poate observa, atunci când comanda este negativă se introduce pe prima intrare valoare PL pentru aer cald şi când comanda este pozitivă pe intrarea a doua se pune valoarea NL, în rest avem FF.

Tabelul 3.2.5 Regula Fuzzy pentru tranziţia T5 de setare aer cald sau rece

Comandă aer	NL	NM	ZR	PM	PL
Eroare	<PL,FF>	<PL,FF>	<FF,FF>	<FF,NL>	<FF,NL>

3.2 Conectarea serverului cu regulatoarele

La nivelul aplicaţiei este prezentă clasa Server care are rolul de a gestiona prezenţa regulatoarelor modelate în sistem prin pornirea unei interfeţe grafice cu care utilizatorul poate interacţiona. În Figura 3.2.1 este realizată diagrama de activităţi care are rolul de a facilita observarea comportamentului aplicaţiei bazat pe aspectele dinamice care au loc asupra sistemului.

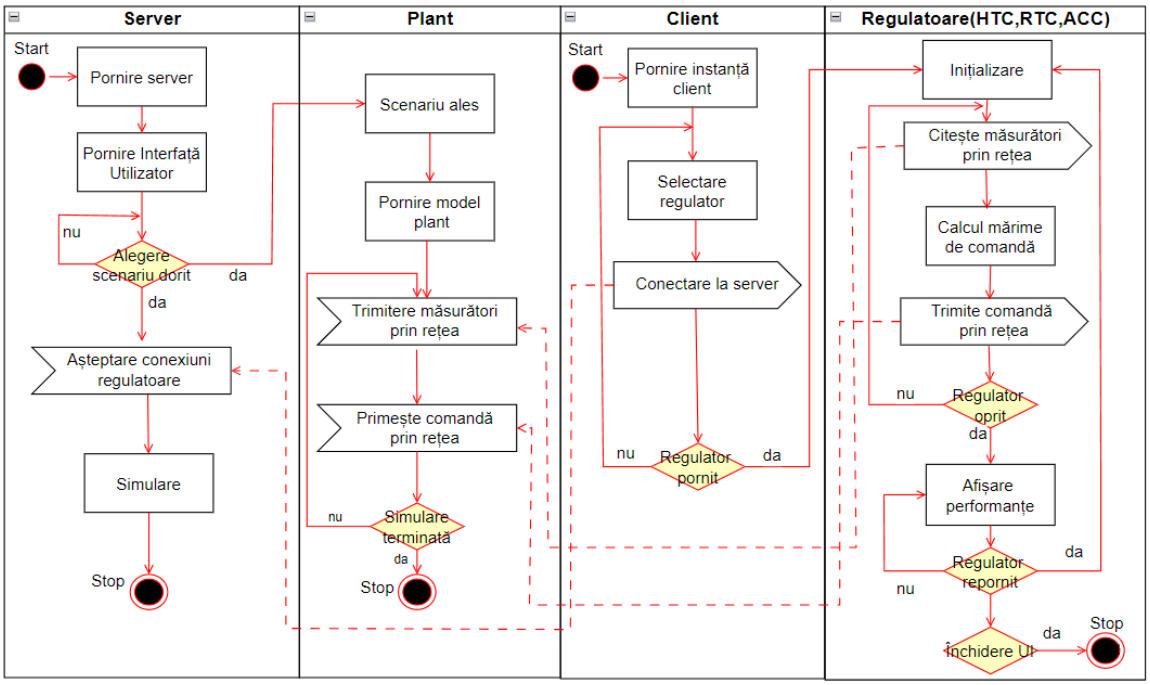

Figura 3.2.1. Diagrama de activităţi

În mod obişnuit, o acţiune este precedată de altă acţiune, fapt pentru care se urmăreşte comportamentul de la condiţia de start şi până la oprirea aplicaţiei marcată de butonul de stop. După cum se poate observa în schema de mai sus, pornirea serverului este primul pas realizat deoarece în urma acestei acţiuni se deschide interfaţa grafică a serverului unde utilizatorul poate vedea ce conexiuni se realizează cu clienţii. La nivelul acestei interfeţe grafice este prezent un element care permite selectarea scenariului dorit pentru a putea porni componenta centralei Plant. Odată ce modelul

Plantului este pornit, în momentul în care regulatoarele se conectează, această componentă trimite măsurătorile prin rețea și primește comanda de la regulator. După alegerea scenariului, serverul așteaptă ca regulatoarele să se conecteze pentru a putea realiza ulterior simularea.

Pentru a putea conecta regulatoare care să acționeze asupra centralei se pornește componenta Client care pornește o altă interfață grafică. La nivelul acestei interfețe grafice se află o serie de butoane denumite conform fiecărui regulator care deschid alte ferestre separate în momentul acționării butoanelor, acestea sunt unice pentru fiecare regulator în parte. Odată ce regulatorul dorit este selectat, acesta se conectează la server.

În momentul alegerii de pornire a unui regulator, în prealabil se realizează o instanță de client pentru a putea realiza o comunicare cu serverul. Fiecare client conectat este pornit pe câte un fir de execuție pentru că fiecare regulator care se pornește are un comportament și un mod de comunicare diferit cu serverul. De exemplu, un regulator HTC trimite o comandă de tip valoare reală care reprezintă comanda de gaz, regulatorul RTC are un semnal de tipul adevărat sau fals care semnifică pornirea/oprirea regulatorului iar elementul ACC are două tipuri de comenzi, fapt pentru care se justifică necesitatea de a crea această instanță de client.

Pornirea unui regulator este urmată de inițializarea acestuia prin setarea valorilor de intrare dacă este nevoie, după care acesta își realizează bucla de reglare atât timp cât este pornit. În cazul în care utilizatorul decide oprirea regulatorului, se afișează performanțele finale. În cazul în care se dorește repornirea unui regulator, acest lucru este posibil, caz în care se reiau pașii de la setarea valorii până la afișare performanțelor. În cazul în care se dorește oprirea sistemului, acest lucru se poate realiza cu ușurință prin închiderea interfețelor de client și server.

3.3 Conectarea între două componente cu jeton fuzzy

Pentru a se putea utiliza regulatoarele modelate de rețele FLETPN este nevoie ca valorile crisp să fie fuzzificate pentru ca acestea să poată fi interpretate și utilizate. Datorită acestui fapt, fiecare dintre cele 3 regulatoare prezintă în componenta clasei câte o funcție care se ocupă de fuzzificarea parametrilor care se află în locațiile de intrare a regulatoarelor. Astfel, pe parcursul traseului rețelei Petri se utilizează jetoane fuzzy care permit ca regulile fuzzy realizate și explicate în subcapitolul 3.1 să poată fi aplicate astfel încât să se respecte comportamentul dorit.

În constructorul regulatoarelor întrările sunt injectate cu un jeton fuzzy nul iar pe ieșiri se prezintă câte o funcție pentru fiecare tranziție de ieșire. Aceste funcții primesc ca parametru un element de tip FuzzyToken și au ca scop comunicarea cu centrala prin transmiterea comenzilor. Comanda dorită este trimisă mai departe la server prin intermediul socketului. La nivelul serverului se creează un obiect de tip ClientHandlerFactory care are rolul de a determina ce tip de regulator se conectează pentru a știi ce tip de client să creeze astfel încât să se poată realiza comunicarea corectă între server și regulatorul dorit.

Este necesară o comunicare diferită cu fiecare regulator deoarece acestea transmit comenzi diferite şi de diferit tip: la ieşirea regulatorului RTC se transmite dacă boilerul este pornit sau oprit prin comanda true/false, la componenta HTC pe ieşire se trimite valoarea reală a comenzii de gaz iar la ACC se trimit două tipuri de comenzi care semnalează dacă regulatorul e oprit/pornit sau dacă aerul eliberat este cald/rece. Pentru a realiza decriptarea mesajului care este trimis ulterior ca şi comandă la centrală se utilizează o funcţie care preia mesajul de la client şi îl interpretează. Acest procedeu de interpretare are loc în funcţia run a fiecărei din cele 3 clase deoarece, atât timp cât socketul este conectat se doreşte citirea continuă a comenzilor deoarece sistemul se modifică în timp real.

3.4 Structura aplicaţiei

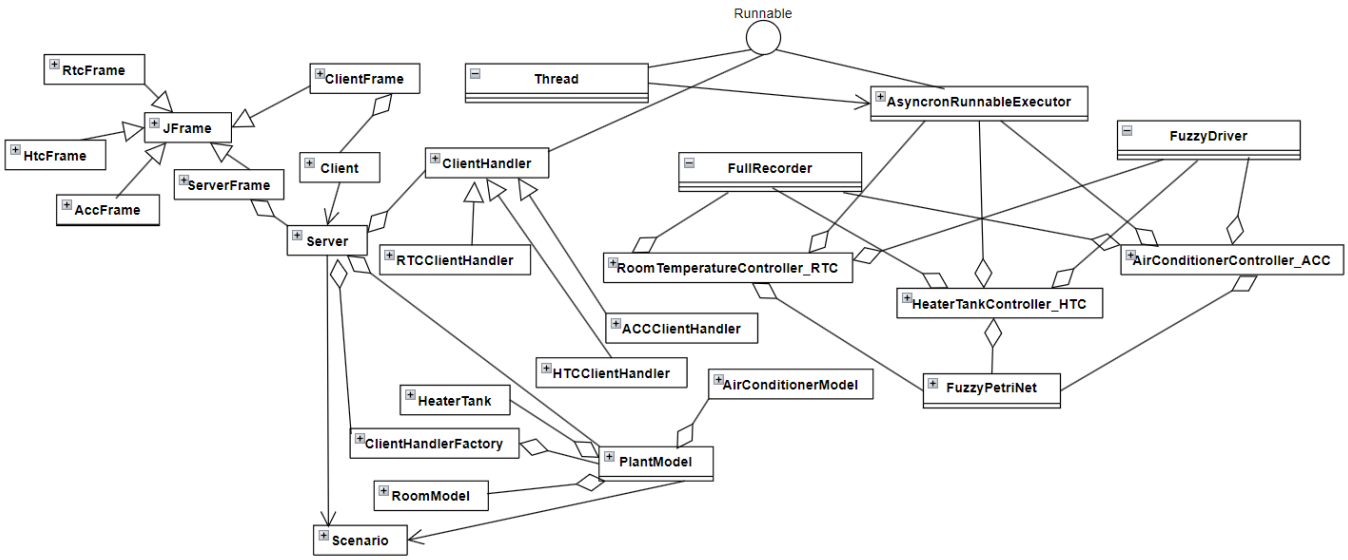

Figura 3.4.1. Diagrama de clase

În Figura 3.4.1 este prezentată diagrama simplificată de clase a lucrării. Aceste tipuri de diagrame reprezintă notaţii grafice care sunt folosite pentru a construi şi vizualiza cu uşurinţă sisteme orientate pe obiect. Diagrama de clase este realizată sub o convenţie cunoscută sub numele de UML (Unified Modeling Language)[27] şi permite descrierea structurii prin definirea claselor cu atributele şi metodele lor, cât şi marcarea relaţiilor dintre acestea. O clasă este un plan care permite realizarea de obiecte prin oferirea unor atribute care definesc acel obiect cât şi prin implementarea unui set de metode care reprezintă acţiunile pe care un obiect le poate efectua. Tipurile de relaţii ce pot fi folosite în realizarea conexiunilor dintre clase sunt prezentate în Figura 3.4.2.

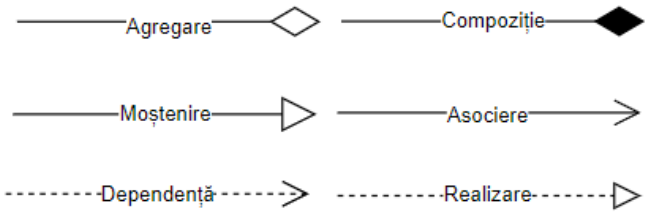

Figura 3.4.2. Tipuri de relaţii între clase

În structura aplicației după cum se poate observa se utilizează preponderent 3 tipuri de relații: agregarea, asocierea și moștenirea. Agregarea are ca scop crearea unui obiect mai complex prin asamblarea altor obiecte împreună. Diferența dintre agregare și compoziție constă în faptul că obiectele componente pot exista și după ce obiectul din care fac parte își încheie ciclul de viață. Asocierea reprezintă o relație structurală dintre două clase și poate fi explicată prin faptul că de exemplu un obiect dintr-o clasă poate obține accesul la un obiect dintr-o clasă diferită. Moștenirea este mecanismul de construire al unei clase derivate pornind de la o clasă existentă. În urma acestui procedeu, clasa derivată poate conține atribute noi sau metode diferite care permit modificarea comportamentului existent din clasa de bază. În următoarele paragrafe se explică în detaliu funcționalitatea fiecărei clase existente în proiect pentru a avea o mai bună înțelegere a modului de funcționare al aplicației.

O serie de clase prezente în diagramă nu necesită implementare deoarece acestea sunt utilizate din cadrul utilitarului FuzzPVisual[20] care este o unealtă utilizată în mediul academic pentru a studia funcționalitatea rețelelor FLETPN. Clasele care fac parte din acest utilitar sunt:

- FuzzyPetriNet care permite desenarea rețelei prin introducerea de locații, tranziții și arce
- AsyncronRunnableExecutor
- FullRecorder urmărește comportamentul rețelei
- FuzzyDriver are rolul de a seta intervalul de lucru și de a fuzzifica valorile introduse în locațiile de intrare și de a defuzzifica comenzile date la tranzițiile de ieșire a regulatoarelor

În continuare, acest capitol are scopul de a detalia funcționalitatea fiecărei clase din componenta sistemului pentru a justifica utilizare și crearea lor în contextul aplicației curente.

Runnable este o interfață care permite unei clase să fie executată în diferite instanțe simultan pe câte un fir de execuție diferit. Asftel se justică și utilizarea clasei Thread care este o clasă deja implementată la nivelul mașinii virtuale java. Pentru aplicații simple, de regulă programul funcționează pe un singur fir de execuție principal, însă necesitatea utilizării acestor elemente provine de la nevoia de distribuire a sistemului. Astfel, fiecare dintre cele 3 regulatoare prezente trebuie să ruleze în instanțe diferite deoarece comportamentul cât și efectul acestora asupra sistemului este diferit. Totodată, implementarea pe fire de execuție permite pornirea sau oprirea regulatoarelor în mod independent fără a afecta funcționalitatea celorlalte, lucru care prezintă libertatea de a utiliza doar setul dorit de regulatoare dintre cele prezente.

Următorul set de clase are ca scop modelarea componentelor existente în sistem: HeaterTank, RoomModel, AirConditionerModel și PlantModel. Primele 3 dintre acestea realizează modelul componentei numite prin asignarea unui set de atribute urmate de realizarea unei funcții constructor. La nivel de metode, fiecare clasă conține o funcție care are scopul de a actualiza sistemul, în această funcție se implementează ecuațiile de stare aferente fiecărui element, conform prezentării acestora în capitolul 3.1. Ca parametrii ai funcției, de exemplu clasa HeaterTank prezintă o variabilă de tip boolean

pentru a determina dacă boilerul este oprit sau pornit şi o variabilă de tip real care reprezintă comanda de gaz. Clasa AirConditionerModel prezintă următorii parametrii: o variabilă de tip real care dă valoarea aerul eliberat de dispozitiv şi două variabile de tip boolean care prezintă dacă componenta este oprită/pornită sau dacă aerul dat este cald/rece. În clasa RoomModel, această funcţie conţine toţi parametrii menţionaţi anterior cât şi alţi doi parametrii care prezintă perturbaţiile sistemului(o variabilă booleană pentru geamul deschis/închis şi o variabilă reală pentru temperatura din exterior) deoarece toate modificările sistemului trebuie să fie aplicate asupra camerei de referinţă pentru care se realizează controlul temperaturii. Aceste 3 clase conţin pe de altă parte şi o metodă de tip get care permite obţinerea unei valori necesare: temperatura aerului pentru aerul condiţionat, temperatura apei din boiler sau temperatura camerei. Componenta PlantModel are scopul de a modela centrala ca şi componentă principală, fapt pentru care în constructor toate componentele anterioare sunt create, fapt care justifică relaţiile de agregare dintre această clasă şi celelalte 3 clase menţionate anterior. În mare parte, funcţiile din această clasă sunt de tip get sau set pentru a permite obţinerea unor valori relevante sau modificarea acestora în funcţie de cerinţe. Totodată, cu ajutorul unei metode numită start se utilizează metodele de actualizare a sistemului atât timp cât centrala este pornită, mai exact cât timp aceasta primeşte date de la scenariul impus. În interiorul clasei se află de asemenea o funcţie care realizează un set de înregistrări pentru a urmări parametrii şi performanţele sistemului. Aceste înregistrări sunt de tip temperatură sau de tip comandă, acestea vor fi afişate ulterior la nivelul interfeţei grafice. În Figura 3.4.3 de mai jos este prezentată clasa HeaterTank.

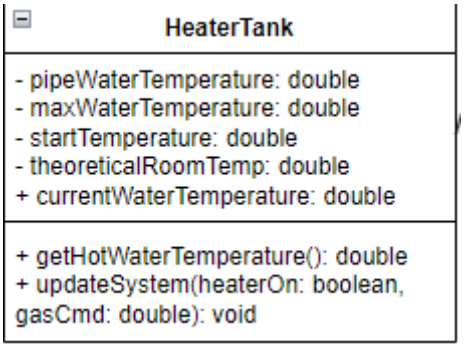

Figura 3.4.3. Clasa HeaterTank

În prima parte a chenarului se pot observa atributele clasei care crează obiectul de tip boiler iar în a doua parte sunt prezente metodele care reprezintă diferite acţiuni. Semnele de calcul precedate de atribute sau metode marchează dacă acestea sunt vizibile sau nu din exteriorul clasei curente: semnul + marchează parametrul ca fiind public, iar semnul – semnifică faptul că acel parametru este privat şi poate fi utilizat doar în interiorul clasei. Motivul pentru care se folosesc aceste elemente provine de la nevoia de a preveni suprascrierea nedorită a unor atribute din altă subclasă. Ca şi convenţie de scriere, se observă că iniţial este menţionat numele parametrului şi mai apoi este urmat de tipul de variabilă sau funcţie care îi exprimă comportamentul.

Clasa Scenario are ca scop realizarea unui set de scenarii modelate care să poată fi aplicate asupra centralei cu scopul de a urmări performanţele sistemului. Atributele

principale ale acestei clase sunt două liste de elemente care prezintă următoarele roluri: prima listă conține valori reale cu privire la temperatura exterioară iar a doua listă conține un set de valori de tip boolean care marchează dacă într-o anumită oră geamul modelat al camerei este deschis sau închis. Scopul acestei clase este de a injecta un set de valori în centrală pentru a urmări comportamentul îndelungat. La nivelul acestei clase sunt prezente mai multe tipuri de funcții: apar metode de tip get care obțin un parametru dorit, metode de tip Scenario care modelează un interval din zi în decursul unui anotimp dorit și totodată este prezentă o metodă care are rolul de a construi scenariul dat în funcție de lungimea listelor menționate anterior.

Elementele RoomTemperatureController_RTC, AirConditionerController_ACC și HeaterTankController_HTC sunt clasele care modelează cele 3 regulatoare care acționează asupra centralei termice. Rolul acestora este de a modela rețeaua Petri cu logică fuzzy, fapt pentru care atributele claselor sunt regulile fuzzy care o să fie ulterior introduse ca acțiune la nivelul tranzițiilor. Totodată aceste clase se folosesc de diferite elemente din utilitarul FuzzPVisual pentru a realiza reprezentarea grafică care facilitează vizualizarea comportamenului regulatorului în fiecare locație. Crearea efectivă a rețelei FLETPN a fiecărui regulator are loc în constructorul clasei, în locațiile de intrare se introduc valorile crisp care sunt fuzzificate și la tranzițiile de ieșire regulatoarele trimit prin socket la server o serie de mesaje de tip comandă. La nivel de metode, fiecare din aceste 3 clase conține o metodă de start care pornește un regulator pe un fir nou de execuție și funcția stop. În funcție de fiecare regulator, fiecare clasă are de asemenea un set de metode care fuzzifică intrările corespunzătoare: componenta RTC fuzzifică temperatura curentă și temperatura de referință a camerei, componenta HTC fuzzifică temperatura curentă și temperatura de referință a apei iar componenta ACC fuzzifică temperatura curentă și cea de referință a aerului dat.

Clasa Server se ocupă de pornirea interfeței grafice principale și are rolul de a comunica cu clienții care în acest context sunt regulatoarele existente. Cu ajutorul unei clase numită ClientHandler, serverul pornește fiecare regulator pe câte un fir de execuție diferit ca să permită ca acestea să funcționeze în paralel. Dat fiind faptul că fiecare regulator are un comportament diferit, clasa ClientHandler este o clasă abstractă a cărei funcționalitate este extinsă de clasele care o moștenesc: RTCClientHandler, HTCClientHandler și ACCClientHandler. Acestea din urmă implementează funcția run în mod diferit în funcție de mesajele pe care aceste componente le primesc prin socket de la cele 3 regulatoare.

Ferestrele de vizualizare sunt componentele cu care utilizatorul interacționează pentru a face modificări asupra sistemului sau pentru vizualizarea unui set de înregistrări sau performanțe. Fereastra principală este creată cu ajutorul clasei ServerFrame și este prima componentă cu care interacționează utilizatorul. La nivelul acestei ferestre se poate alege tipul de scenariu de temperatură care să fie utilizat la nivelul centralei pentru a avea un set de date de lucru. Odată ce scenariul dorit este ales se poate porni modelul centralei prin apăsarea butonului Pornire model centrală, iar pentru a închide aplicația se poate folosi butonul de Ieșire. De menționat este că odată ce s-a ales un scenariu meniul de selectare, acesta se oprește. Sub butoanele aferente se află un chenar în care se afișează diferite detalii precum dacă se conectează/deconectează

clienți la server. Totodată, la nivelul acestei interfețe, în zona de text apar performanțele fiecărui regulator conectat la centrală, precum ar fi: consumul mediu, timpul maxim în care centrala este pornită în mod continuu și multe altele. Structura ferestrei principale a serverului este prezentată în Figura 3.4.4 de mai jos.

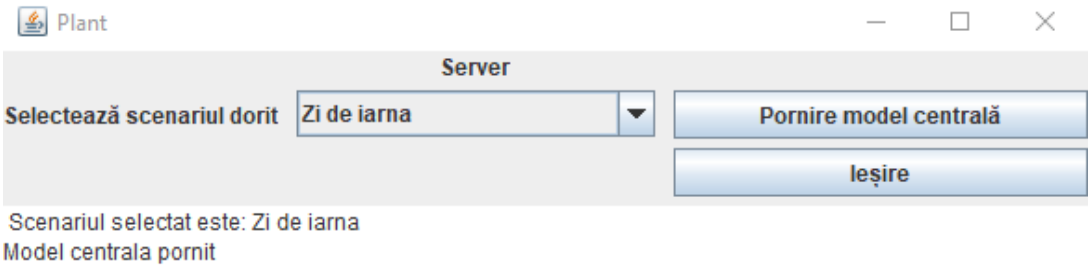

Figura 3.4.4 Fereastra principală a serverului

Clasa principală care se ocupă de regulatoarele din aplicație se numește ClientFrame. La nivelul acestei clase se prezintă o serie de butoane: un buton de Ieșire și 3 butoane cu numele aferent fiecărui regulator. Cele 3 butoane, la acționarea acestora, deschid fiecare câte o fereastră nouă, aferentă regulatorului dorit. Fereastra principală de Client este prezentată în Figura 3.4.5 de mai jos.

Figura 3.4.5 Fereastra principală a clientului

Ultimele 3 interfețe, RTCFrame, HTCFrame și ACCFrame conțin fiecare următoarele elemente: un buton de oprire/pornire a regulatorului, un câmp de introducere a unei valori de referință de intrare unde este cazul și un set de performanțe prezente la nivelul unui câmp de tip text. Pornirea unui regulator permite de asemenea accesarea unui set de grafice pentru a urmări intrările și ieșirile pentru a putea analiza dacă comportamentul impus este cel dorit. Pornirea/oprirea regulatorului este marcată pe nivelul butonului care capătă culoarea verde, respectiv roșu în funcție de caz. În Figura 3.4.6 de mai jos este reprezentată fereastra pentru componenta HTC iar pentru restul regulatoarelor fereastra de afișare este similară.

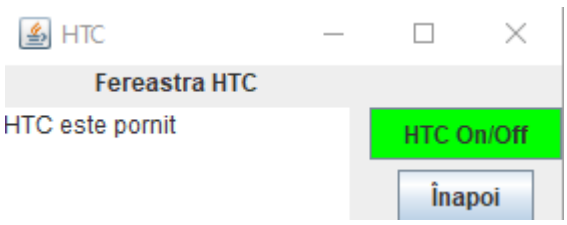

Figura 3.4.6 Fereastra regulatorului HTC

3.5 Implementare

Capitolul de implementare urmăreşte să dezvăluie o serie de secvenţe de cod relevante prezente la nivelul aplicaţiei. Scopul realizării acestui subcapitol urmăreşte aducerea unei explicaţii cât mai amânunţite cu privire la implementarea a diferite elemente sau simularea unui tip de comportament la nivelul sistemului. Totodată, existenţa acestui subcapitol permite facilitatea înţelegerii logicii din spate pentru a permite o implementare ulterioară sporită.

Realizarea reţelelor Petri la nivel de cod java este posibilă prin utilizarea utilitarului FuzzPVisual care permite modelarea şi monitorizarea acestor reţele realizate, în Figura 3.5.1 prezentată mai jos se explică o serie de comenzi.

```
net = new FuzzyPetriNet();
int p0 = net.addPlace();
p1RefInp = net.addInputPlace();
net.setInitialMarkingForPlace(p0, FuzzyToken.zeroToken());
int tr0Reader = net.addTransition( delay: 0, parser.parseTwoX0neTable(reader));
net.addArcFromPlaceToTransition(p0, tr0Reader, weight: 1.0);
net.addArcFromPlaceToTransition(p1RefInp, tr0Reader, weight: 1.0);
```

Figura 3.5.1. Comenzi realizare reţea Petri

În primul rând, se realizează crearea unei reţele Petri nouă, lucru care este posibil prin declararea la nivelul atributelor o variabilă numită net care este de tipul clasei FuzzyPetriNet, clasă componentă a utilitarului folosit. În continuare, în ordinea apariţiei elementelor, acestea se introduc rând pe rând.

Următoarele două linii de cod arată modul de creare a locaţiilor: p0 este o locaţie ce se află în interiorul buclei, fapt pentru care aceasta se poate crea ca şi variabilă de tip întreg în interiorul constructorului regulatorului iar p1RefInp este o locaţie de tip intrare. În cazul ulterior, locaţia de intrare este declarată în exteriorul constructorului ca şi variabilă globală deoarece într-o astfel de locaţie se doreşte introducerea unei valori crisp primite din exteriorul regulatorului. Această valoare crisp trebuie întâi convertită într-o variabilă de tip fuzzy cu ajutorul unor funcţii de tip setare valoare care sunt prezente la nivelul fiecărei clase de regulator.

Funcţia de setInitialMarkingForPlace permite introducerea în locaţii a jetoanelor fuzzy. Marcajele iniţiale sunt de regulă folosite în această aplicaţie atunci când locaţia respectivă este precedată de o tranziţie întârziată cu un tact de timp care impune necesitatea de a relua ciclul de reglare atât timp cât nu s-a ajuns la valoarea reglată dorită.

Crearea şi declararea elementelor de tip tranziţie se realizează la nivelul constructorului. Ca parametrii, funcţia de creare a unei tranziţii conţine valoarea dorită de întârziere cât şi tipul de regulă fuzzy care este folosită. În cazul prezentat mai sus, tranziţia tr0Reader nu este întârziată deoarece prezintă valoarea 0, iar regula pe care această tranziţie o execută este aceea de a citi valorile de intrare. Apariţia a două intrări şi o ieşire este semnalată de către funcţia parse care realizează analiza regulii. Crearea

tranzițiilor de ieșire se face în mod similar cu funcția addOuputTransition, singura diferență este că întârzierea nu mai este prezentă ca parametru, ci doar regula impusă.

Ultimele două linii de cod prezintă modalitatea de creare a arcelor între o locație și o tranziție. Parametrii de intrare sunt locația, respectiv tranziția între care se desenează arcul și o a treia valoare care semnifică costul de transmitere a jetonului mai departe. Unde nu este necesară o funcționalitate specială, costul arcelor este de regulă impus la valoarea 1. Cu ajutorul funcției addArcFromTransitionToPlace se realizează în mod similar desenarea unui arc, cu diferența că această metodă creează un arc de la tranziție la locație și nu invers..

Față de tranzițiile din interiorul rețelei, tranzițiile de ieșire au scopul de a transmite o comandă mai departe la centrală pentru a realiza reglarea dorită. În Figura 3.5.2 de mai jos se observă implementarea unei funcții pentru o astfel de tranziție.

```
net.addActionForOuputTransition(tr2Out, new Consumer<FuzzyToken>() {
    @Override
    public void accept(FuzzyToken tk) {
        try {
            osw.write(String.valueOf(tankCommandDriver.defuzzify(tk)));
        } catch (IOException e) { throw new RuntimeException(e); }}
});
```

Figura 3.5.2. Acțiune impusă pe o tranziție de ieșire

Această funcție are ca parametrii tranziția de ieșire, un jeton fuzzy de tip consumator și o metodă accept care preia marcajul fuzzy. În acest exemplu, această tranziție de ieșire corespunde cu comanda de gaz dată de regulatorul HTC. Valoarea este transmisă prin socket către server cu ajutorul variabilei osw care este de tip OutputStreamWriter. Pentru ca centrala să poată interpreta valoarea, jetonul este mai întâi defuzificat și apoi convertit sub forma unei variabile de tip text. Această comandă este introdusă într-o buclă de tip try catch care are scopul de a trata excepții, respectiv erori ce pot apărea la nivelul programului. Prin acest tip de buclă se previne oprirea bruscă a programului, acesta fiind capabil să își continue execuția.

La nivelul clasei Server, se alege în primul rând un scenariu dorit din cele existente și se pornește componenta care modelează centrala termică. Apoi, pentru a gestiona conectarea regulatoarelor, se crează un element de tipul ClientHandlerFactory care, prin determinarea tipului de regulator dorit, creează un ClientHandler care are scopul de a implementa o comunicare corectă între server și comenzile date de un regulator. În Figura 3.5.3 se exemplifică crearea unui astfel de element în funcție de regulatorul HTC.

```
if (controllerName.equals("HTC")) {
    setControllerName(controllerName);
    return new HTCClientHandler(socket, plant);
```

Figura 3.5.3. Crearea unui handler

Atât timp cât socket-ul serverului este pornit, în cazul conectării unui nou regulator se crează un astfel de ClientHandler care este pornit pe un nou fir de execuţie. Acest lucru este realizat pentru că fiecare regulator are un comportament şi un set de comenzi diferit, fapt ce duce la necesitatea funcţionării acestora în paralel, independent unul de celălalt. Regulatoarele conectate sunt ulterior introduse într-o listă pentru a reţine care dintre acestea sunt conectate la orice moment de timp dorit.

În Figura 3.5.4 prezentată mai jos are loc setarea comenzii primite de la regulator asupra modelului centralei termice. Pentru exemplificare, acest de comenzi este realizat pentru regulatorul HTC, celelalte două handlere ale regulatoarelor au o implementare similară, diferenţa principală fiind tipul de comandă care este transmis.

```
try {
    messageFromClient = bufferedReader.readLine();
while (socket.isConnected()) {
        double value = parseMessageFromClient(messageFromClient);
        plant.setHeaterGasCmd(value);
        double waterTemp = plant.getTankWaterTemperature();
        bufferedWriter.write( str: waterTemp + "");
    }} catch (IOException e) {
    e.printStackTrace();}
```

Figura 3.5.4. Setarea comenzii date de regulator

Implementarea evidenţiată mai sus are loc în interiorul funcţiei run al fiecărui handler. Handler-ul fiecărui tip de regulator extinde clasa ClientHandler, o clasă abstractă care implementează Runnable pentru a permite suprascrierea funcţiei run conform fiecărei cerinţe. Înainte de setarea comenzii date de regulator, se citeşte valoarea primită şi se reţine la nivelul unei variabile de tip mesaj. În momentul în care handler-ul este conectat la socket, această valoare este convertită la nivelul funcţiei parseMessageFromClient care o să fie explicată ulterior. În continuare, pentru acest exemplu, se setează ieşirea regulatorului HTC care reprezintă comanda de gaz şi se realizează o cerere de tip get la centrală pentru a obţine valoarea curentă a temperaturii apei care este modificată în urma setării comenzii de gaz. Tot acest procedeu este realizat în interiorul unui bloc de tip try catch pentru a trata erorile pe intrări şi ieşiri ca şi o excepţie. În cazul în care nu se poate realiza setul de comenzi, se aruncă excepţia şi se printează detalii cu privire la numărul liniei şi numele clasei în care are loc excepţia.

În Figura 3.5.5 este realizată funcţia de conversie a comenzii primite de la regulatorul HTC care este utilizată în funcţia run prezentată anterior.

```
private double parseMessageFromClient(String message){
    double value=0;
    try{
        value = Double.parseDouble(message);
    }catch (NumberFormatException e){}
    return value;}
```

Figura 3.5.5. Funcţia de conversie a comenzii pentru HTC

În acest exemplu are loc o conversie între un element text şi o valoare numerică reală după care se returnează valoarea convertită Această funcţie este în acest caz o funcţie de tip double, însă pentru alte regulatoare această funcţie este implementată diferit. Pentru regulatorul RTC, funcţia este de tip boolean deoarece pe ieşirea regulatorului comanda este de a opri sau a porni regulatorul. Pentru regulatorul ACC se foloseşte funcţia de tip String pentru că pe ieşirea regulatorului sunt prezente două tipuri de comenzi, una pentru pornirea/oprirea regulatorului şi una pentru alegerea aerului cald/rece.

3.5.1 Şabloane de proiectare folosite

La nivelul implementării acestei aplicaţii s-au folosit două tipuri de şabloane de proiectare: client-server şi factory. Acestea sunt adesea folosite pentru rezolva diferite probleme comune ce pot apărea la nivelul realizării unei aplicaţii. Cu ajutorul acestor şabloane se pot evidenţia atât componentele principale ale sistemului cât şi interacţiunea între acestea.

Şablonul client-server este utilizat cu scopul de a permite sistemului să suporte un volum ridicat de informaţii şi să permită extinderea ulterioară a acestuia în funcţie de priorităţile de dezvoltare. Astfel, la nivelul acestei aplicaţii, serverul are scopul de a comunica cu clienţii care, fiind reprezentaţi de regulatoarele sistemului, comunică şi reglează temperatura la nivelul modelului încăperii. Astfel, interacţiunea clienţilor cu serverul modifică în timp real comportamentul sistemului de reglare al temperaturii iar la nivel de clienţi, utilizatorul poate interacţiona cu interfaţa grafică pentru a decide ce regulatoare să fie puse în funcţiune.

Existenţa a mai multe regulatoare duce la nevoia de a crea o comunicare diferită între fiecare dintre acestea cu serverul. Un client handler este componenta care are rolul de a trata comunicarea între server şi un client, fapt pentru care este necesară utilizarea unei clase numită ClientHandlerFactory care este realizată cu ajutorul şablonului de proiectare factory. Astfel la nivelul acestei clase, în funcţie de numele regulatorului care se doreşte a fi conectat, se crează un client handler unic în funcţie de regulatorul dorit. În acest mod se asigură că de fiecare dată se apelează componenta care se ocupă de comunicarea corectă între server şi regulatorul aflat în cauză. Utilizarea acestui şablon facilitează totodată şi extinderea ulterioară a funcţionalităţii aplicaţiei deoarece, dacă la o implementare ulterioară se doreşte crearea unor noi regulatoare, codul rămâne uşor de înţeles.

4 Scenarii și testare aplicație

Capitolul curent urmărește să creeze și să analizeze o serie de teste aplicate la nivelul sistemului dat. Astfel, în urma testelor realizate se pot identifica funcționalitățile sistemului și se pot urmări performanțele obținute în urma utilizării regulatoarelor implementate. Principalele aspecte care se urmăresc sunt: confirmarea că sistemul este funcțional și răspunde cerințelor date de utilizator, determinarea că aplicația continuă să funcționeze în urma deconectării unui regulator și analizarea performanțelor pentru a constata dacă sistemul se comportă conform cerințelor impuse. Prin testarea aplicației urmând acești pași se confirmă că totul a fost implementat corect și aplicația este pregătită pentru a putea fi dezvoltată ulterior dacă se dorește acest lucru.

4.1 Scenarii de testare

În acest subcapitol se studiază utilizarea scenariilor pentru a vedea cum se modifică reglarea sistemului. Se consideră aplicația dată care urmărește să regleze temperatura unei camere prin capacitatea de a prelua date din mediul exterior și de a rejecta perturbații ce pot apărea la nivelul sistemului. Astfel, se creează câte un scenariu pentru un interval de zi într-un anume anotimp. Clasa care se ocupă de crearea acestui obiect de tip scenariu are ca atribute o listă de numere întregi care sunt temperaturi injectate, acestea fiind temperaturi medii de afară în fiecare oră într-un anotimp dat și o altă listă de elemente tip boolean care o să rețină pentru fiecare oră dacă geamul din incinta camerei este deschis sau închis. Aceste date introduse sunt alese pentru a demonstra pe un interval de timp care este eficiența dată de sistem, pentru o implementarea reală ulterioară aceste date injectate pot fi preluate de la un senzor de temperatură plasat în mediul exterior și un senzor ultrasonic care să determine dacă fereastra este deschisă, însă această aplicație are doar un rol demonstrativ, fapt pentru care datele sunt alese conform mediilor de temperatură anuale ale fiecărui anotimp și datele pentru acționarea asupra ferestrei sunt probabilități alese la intervale orare unde este firesc ca această acțiune are o mai mare șansă de a avea loc. Astfel, componenta scenariu conține mai multe funcții statice pentru o zi a tuturor celor 4 anotimpuri care inițializează cele două liste cu un set de câte 24 de valori. În mod normal însă o centrală termică nu este pornită pe parcursul întregii zile, fapt pentru care se pot crea și alte scenarii cu un set de valori mai mic decât 24 pentru a demonstra că este posibilă urmărirea performanțelor sistemelor pe un interval mai scurt de timp, fapt pentru care se introduce și un scenariu al unei dimineți de iarnă care utilizează sistemul doar timp de 4 ore. Pentru a demonstra și alte intervale de timp se pot crea noi funcții cu ușurință, singurul detaliu care trebuie luat în calcul este ca numărul de valori din lista de temperaturi să fie egal cu numărul de valori din lista de deschidere a geamului. Pentru a observa mai ușor scenariile folosite la nivelul aplicației, se poate urmări graficul de mai

jos prezent în Figura 4.1.1 unde fiecare scenariu este marcat cu o culoare marcată în legenda alăturată.

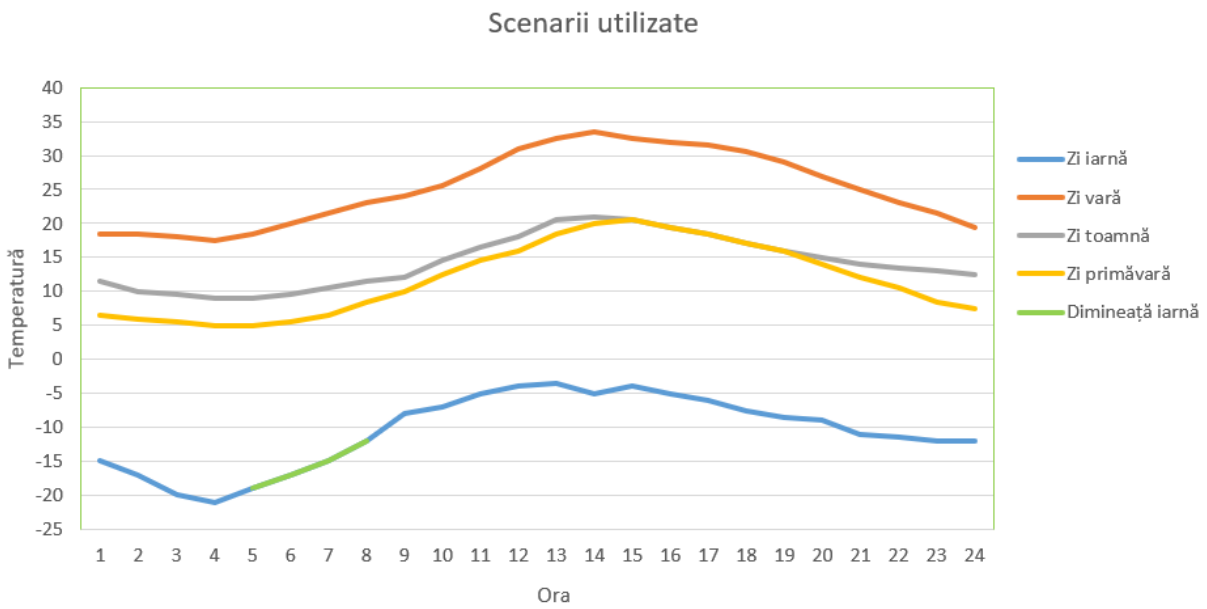

Figura 4.1.1. Scenarii utilizate

4.2 Testare efectivă

Elementele prezente la nivelul interfeței grafice sunt testate pentru a confirma faptul că sistemul este funcțional și receptiv la comenzile utilizatorului care se ocupă de testare. În primul rând, se pornește aplicația sub forma unei ferestre principale. La nivelul aceste ferestre se alege în primul rând un scenariu dorit asupra căruia se dorește testarea sistemului. Opțiunea de a selecta scenariul dorit este prezentată în Figura 4.2.1 de mai jos.

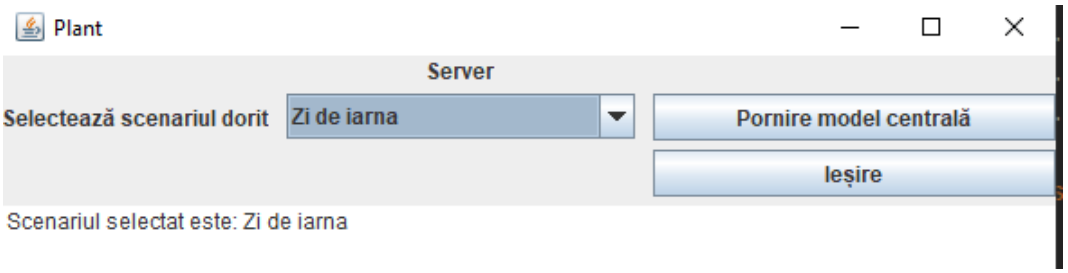

Figura 4.2.1. Selectarea scenariului

În urma alegerii scenariului dorit, următorul pas constă în pornirea unui set de regulatoare din cele prezente în sistem. Pentru a putea face acest lucru, la nivelul ferestrei principale se află un set de butoane al căror nume corespunde cu fiecare regulator. Prin acționarea unuia din cele 3 butoane, se deschide o nouă fereastră care poate conține: un câmp în care se setează o valoare de intrare(precum temperatura dorită în cameră), un buton Înapoi care permite reintoarcerea la fereastra principală, un buton care pornește/oprește regulatorul și totodată mai apar diferite grafice sau performanțe. În Figura 4.2.2 de mai jos se acordă ca exemplu accesarea ferestrei care se ocupă de regulatorul RTC.

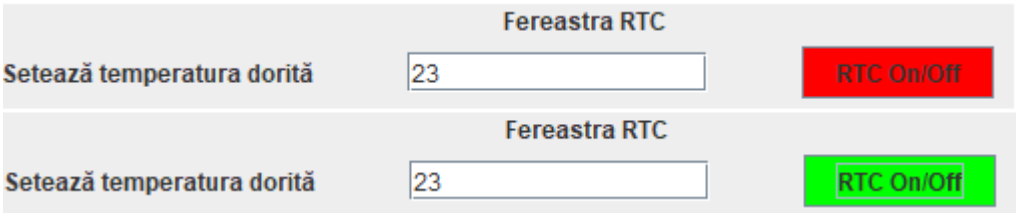

Figura 4.2.2.Pornire/oprire RTC și setare valoare temperatură

După cum se poate observa, la nivelul acestei ferestre se poate introduce de la tastatură valoarea dorită de reglare a temperaturii camerei și cu ajutorul butonului de sub câmp, se poate opri sau porni regulatorul. Culoarea roșie marchează faptul că regulatorul este oprit, iar buton capătă culoarea verde în momentul apăsării acestuia, fapt ce semnifică că regulatorul este pornit. În mod evident, dacă se dorește ulterior oprirea regulatorului, acest lucru este posibil prin simpla reapăsare a butonului, care o să își schimbe din nou culoarea în roșu pentru a semnala faptul că regulatorul este oprit. Oprirea și pornirea unui regulator este de asemenea semnalată la nivelul aplicației printr-o serie de mesaje afișate în cadrul unei zone de text. La nivelul acestor ferestre există și un buton numit Înapoi care permite închiderea ferestrei dorite.

La nivelul ferestrei principale există un ultim buton de Ieșire care permite oprirea aplicației. În urma acționării butonului, apare o fereastră de dialog care cere confirmarea utilizatorului pentru a se asigura că oprirea aplicației este dorită de către acesta. În Figura 4.2.3 se poate vedea această fereastră de dialog care este pornită în urma acționării butonului de ieșire din aplicație, dacă se selectează Yes atunci interfața grafică este oprită, în caz contrar, fereastra de dialog dispare.

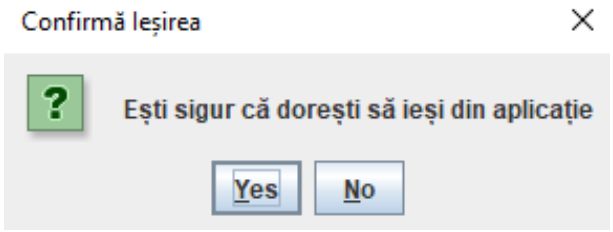

Figura 4.2.3. Confirmare oprire aplicație

4.3 Performanțe

În cele ce urmează, se urmărește simularea comportamentul sistemului în momentul pornirii regulatoarelor. Acest comportament în consecință prezintă un set de performanțe care este analizat la nivelul acestui subcapitol. Performanțele sistemului sunt foarte importante deoarece scopul inițial impus încă de la începutul lucrării este acela de a realiza un sistem capabil să regleze temperatura camerei cât mai eficient posibil. Astfel, prin atingerea unui set de performanțe bun se justifică nevoia de a se dezvolta acest domeniu de reglare cu regulatoare FLETPN pentru că se reduc costurile de întreținere care în consecință duc și la creșterea satisfacției potențialilor utilizatori.

Următoarele performanțe se consideră asupra unui scenariu de iarnă aplicat asupra centralei. Totodată, se consideră temperatura admisă în boiler la o temperatură maximă de 75 de grade Celsius pentru că o valoare mai mare la o implementare fizică

poate duce la distrugerea sistemului. La nivelul regulatorului RTC se alege valoare de 24 de grade Celsius ca fiind valoarea dorită la nivelul modelului camerei.

De menţionat este faptul că performanţele discutate în următoarele paragrafe sunt performanţele obţinute în cazul în care toate cele 3 regulatoare sunt pornite în scenariul unei zile de iarnă. Astfel, în continuare se explică pe rând ce performanţe aduce fiecare din cele 3 regulatoare prezente în momentul în care acestea sunt active.

Pentru regulatorul HTC, se prezintă următoarele detalii: temperatura maximă/ minimă, temperatura medie şi stabilitatea temperaturii. Stabilitatea temperaturii S_{Tapa} este un procent care arată variaţia temperaturii apei din boiler faţă de temperatura medie. Cu cât procentul este mai mic cu atât se indică o stabilitate mai mare. Pentru a putea determina acest parametru se utilizează ecuaţia (4.3.1) de mai jos.

$$S_{Tapa} = (T_{\max apa} - T_{\min apa})/T_{\text{mediu } apa} * 100 \qquad (4.3.1)$$

Se menţionează faptul că în ecuaţia utilizată anterior nu se ia în calcul temperatura minimă ca fiind 23 de grade deoarece aceasta este considerată temperatura la care se află apa în momentul în care centrala este pornită pentru prima dată. Performanţele se calculează în jurul valorii staţionare care în acest caz este temperatura medie. Astfel, se decide alegerea unei benzi de eroare de +-5%, care în acest caz încadrează sistemul în regim staţionar între +-3.65 de grade Celsius faţă de temperatura medie calculată. Toate aceste performanţe sunt vizibile în Figura 4.3.1 care apare la nivelul zonei de text din cadrul aplicaţiei server. După cum se poate obseva, variaţia de 4.99% este mică, fapt ce semnifică o stabilitate ridicată.

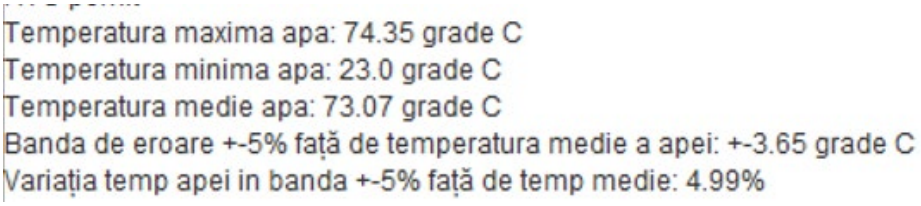

Figura 4.3.1. Performanţe HTC

Pornirea regulatorului RTC produce un set de informaţii precum temperatura minimă, maximă, respectiv medie atinsă la nivelul camerei, stabilitatea temperaturii faţă de temperatura medie cât şi alte performanţe cu privire la comportamentul centralei, printre care: raportul în care încălzirea este pornită, timpul maxim în minute care arată cât timp a fost pornită centrala în mod continuu, consumul total atins în urma acţionării scenariului ales şi consumul mediu utilizat într-un minut. Stabilitatea temperaturii din cameră se calculează utilizând ecuaţia (4.3.2) de mai jos.

$$S_{Tcamera} = (T_{max} - T_{min})/T_{mediu} * 100 \qquad (4.3.2)$$

Totodată, raportul de încălzire pornită semnifică perioada de timp în care boilerul este activ comparativ cu timpul total de funcţionare. Atât în acest caz, cât şi la nivelul stabilităţii temperaturii, un procentaj scăzut indică o eficienţă mai bună.

De asemenea, se poate lua în considerare timpul maxim în minute care arată cât timp a fost pornită centrala pentru a determina cât de eficient este sistemul. La nivelul performanţelor, se mai poate vizualiza atât consumul total cât şi consumul mediu într-

un minut pentru a asigura o bună monitorizare a consumului. Toate aceste înregistrări sunt prezentate mai jos în Figura 4.3.2.

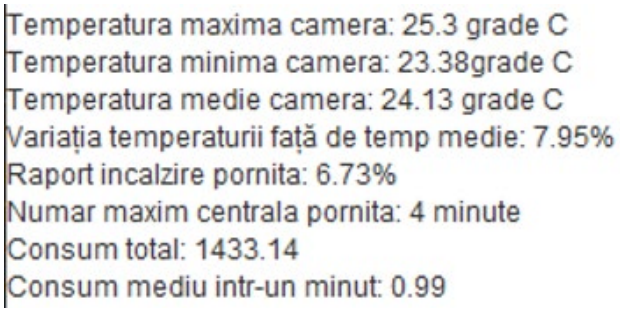

Figura 4.3.2. Performanţe RTC

Ultimul regulator ACC acţionează asupra temperaturii aerului eliberat, astfel că acesta afişează în consecinţă următoarele: temperatura maximă, minimă şi medie a aerului, cât şi eficienţa aerului condiţionat. Această eficienţă este calculată prin utilizarea ecuaţiei (4.3.3) de mai jos.

$$S_{Taer} = (T_{max\ aer} - T_{min\ aer})/T_{mediu\ aer} * 100 \qquad (4.3.3)$$

Limitele impuse la nivelul acestui regulator sunt că aerul eliberat nu poate fi mai cald decât o temperatură de 23 de grade Celsius sau mai rece decât 10 grade Celsius pentru că o implementare reală ar aduce astfel de limitări asupra sistemului. Astfel, în urma rulării scenariului pentru o zi de iarnă apar performanţele din Figura 4.3.3. După cum se poate observa, procentajul are o valoare foarte bună de doar 1.08%, însă în această situaţie performanţa este atât de bună deoarece toate cele 3 regulatoare funcţionează în acest scenariu simulat de iarnă, fapt pentru care regulatorul HTC se ocupă de majoritatea procesului de încălzire a camerei în acest caz.

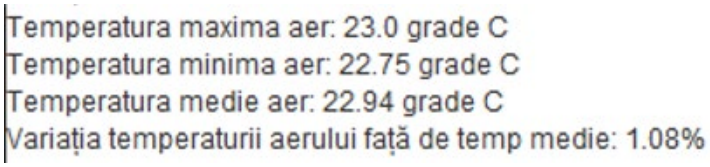

Figura 4.3.3. Performanţe ACC: scenariu de iarnă

Evident, dacă se rulează un alt scenariu performanţele sistemului diferă în funcţie de condiţiile date. La restul scenariilor, consumul scade deoarece nu este nevoie de o compensare la fel de mare asupra temperaturii camerei. Ca exemplu, în Figura 4.3.4 sunt prezentate înregistrări cu privire la temperaturile atinse de aerul controlat de regulatorul ACC. După cum se poate observa, deoarece în acest scenariu se doreşte răcirea camerei, temperatura minimă şi medie suferă schimbări în comparaţie cu valorile atinse în figura anterioară.

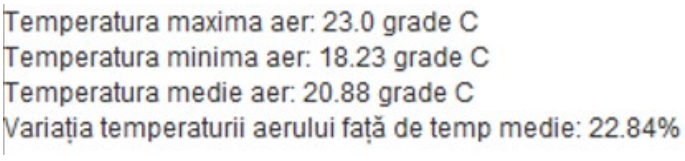

Figura 4.3.4. Performanţe ACC: scenariu de vară

În această situaţie variaţia temperaturii aerului faţă de temperatura medie are un procent mai ridicat, mai exact de 22.84%, deoarece în această situaţie doar regulatorul ACC poate contribui la răcirea camerei la temperatura dorită. În toate cazurile menţionate anterior, un procentaj scăzut al performanţelor explicate semnifică faptul că performanţele sunt bune.

În următoarele paragrafe se studiază graficele obţinute la nivelul regulatoarelor pentru a înţelege atât modul lor de funcţionare cât şi pentru a urmări şi determina calitatea performanţelor obţinute. Regulatoarele modelate de mai jos sunt realizate cu ajutorul utilitarului FuzzPVisual, astfel se poate monitoriza fiecare locaţie în parte pentru a determina dacă valorile la întrări sunt corecte şi dacă regulile fuzzy ale tranziţiilor sunt realizate corect. La nivelul interfeţei deschise se crează graficele care se prezintă ulterior în acest subcapitol, în cazul în care se doreşte vizualizarea comportamentului în interiorul unei locaţii se apasă click stânga, iar pentru deselectare se acţionează click dreapta.

În Figura 4.3.5 este afişat regulatorul HTC care este creat cu ajutorul clasei HeaterTankController_HTC, acest afişaj este realizat cu ajutorul utilitarului FuzzPVisual şi permite o vizualizarea facilitată a intrărilor şi ieşirilor regulatorului pentru a determina dacă comportamentul impus de regulile fuzzy de la nivelul tranziţiilor sunt implementate în mod corect.

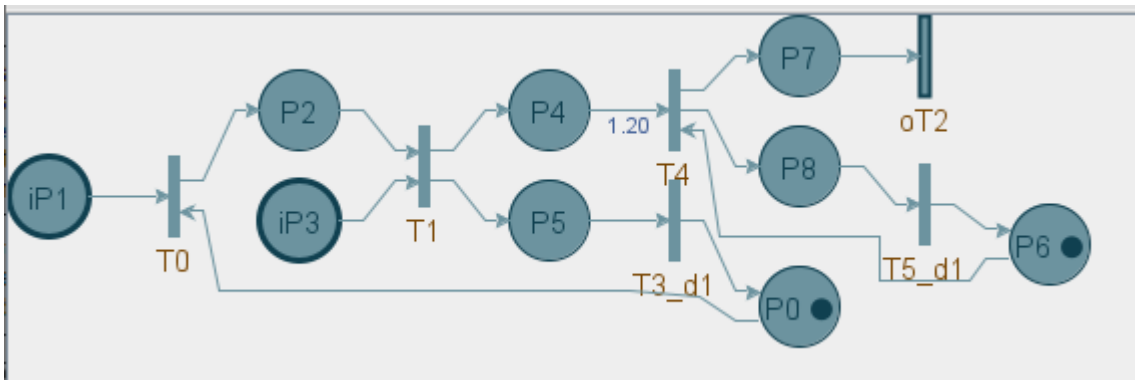

Figura 4.3.5. Regulator HTC modelat

La nivelul următoarelor grafice pe axa X este reprezentat timpul iar pe axa Y se află gradul de apartenenţă. În Figura 4.3.6 se observă că pe un grafic limitat în intervalul [-1,1] se realizează reprezentarea grafică a diferite elemente. Marcajul iP1 desenat cu roşu reprezintă temperatura de referinţă a apei din boiler la care se doreşte să se ajungă iar marcajul iP3 desenat cu albastru reprezintă temperatura reală curentă a apei din boiler. În tranziţia T1 este necesară realizarea erorii dintre aceste două valori, fapt pentru care regula fuzzy prezentă realizează operaţia de diferenţă aşa cum este explicat în subcapitolul 3.1. Astfel în locaţiile de ieşire a tranziţiei, precum în poziţia P5 marcată cu verde, se vede cu uşurinţă că acţiunea este corectă şi realizează calculul erorii. În primele secunde ale pornirii regulatorului se observă faptul că eroarea este mare, acest lucru este determinat de apariţia suprareglajului. Pentru a urmări mai uşor acest grafic cât şi următoarele grafice prezentate mai jos, fiecare va conţine o legendă care marchează culoarea valorilor parametrilor de intrare şi astfel ajută la identificarea mult mai rapidă a fiecărui element prezent.

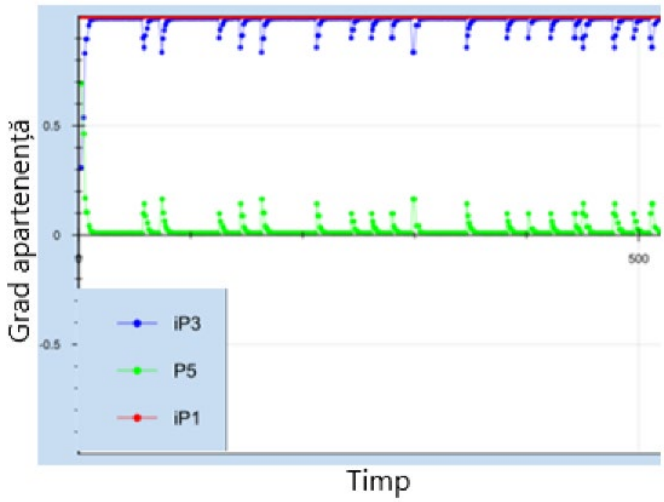

Figura 4.3.6. Intrările şi eroarea componentei HTC

În Figura 4.3.7 de mai jos apare aceeaşi eroare pentru ambele marcaje de culoare, însă se observă că acestea sunt decalate cu un tact de timp. Scopul prezentării acestui grafic constă în a demonstra că tranziţia T3_d1 realizează operaţia de întârziere cu un tact care este trimisă în tranziţia T0 pentru a relua ciclul de reglare atât timp cât sistemul este pornit şi nu s-a ajuns la valoarea dorită. Totodată, în această figură se observă mult mai uşor existenţa unui suprareglaj al erorii la începutul simulării, fapt ce este normal deoarece valoarea temperaturii apei din boiler la pornirea acestuia are o valoare mult mai mică decât valoarea finală la care se doreşte să se ajungă.

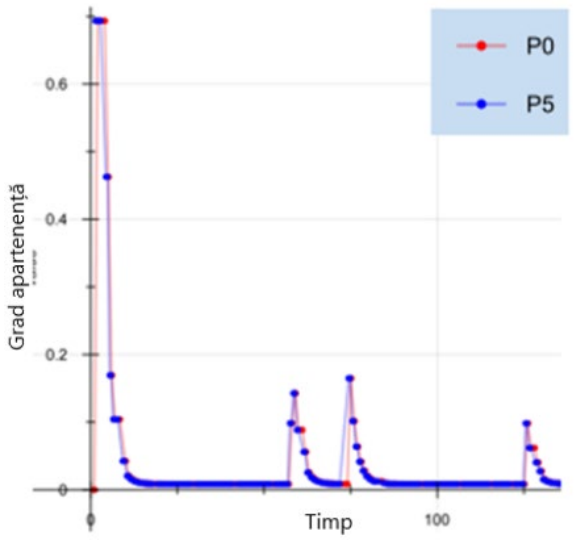

Figura 4.3.7 Întârziere de un tact

În Figura 4.3.8 este realizat modelul reţelei FLETPN pentru componenta RTC realizat la nivelul clasei RoomTemperatureController_RTC care utilizează clase din utilitarului FuzzPVisual. La fel ca şi la regulatorul anterior, se pot accesa graficele prezente la nivelul locaţiilor pentru a urmări comportamentul componentei RTC.

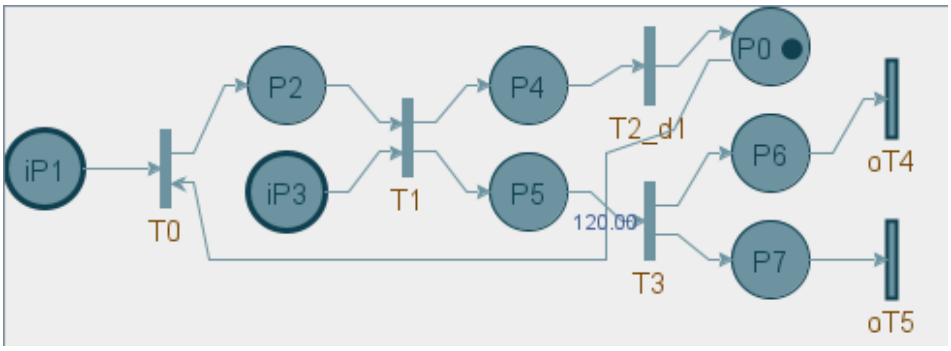

Figura 4.3.8. Regulator RTC modelat

La nivelul acestui regulator se află două locaţii de intrare: una primeşte valoarea de referinţă a temperaturii iar cealaltă primeşte valoarea curentă. La ieşirea acestui regulator se transmite o valoare booleană care comandă dacă sistemul de reglare a temperaturii se porneşte/opreşte. În mod similar ca şi la regulatorul anterior, există tranziţia T2_d1 care reprezintă întârzierea de un tact care reia calculul erorii atât timp cât aceasta nu este nulă.

În Figura 4.3.9 se pot observa următoarele elemente: temperatura de referinţă iP1 marcată cu albastru şi temperatura curentă a camerei iP3 marcată cu roşu. În tranziţia T1 are loc calculul erorii dintre cei doi parametrii prin realizarea operaţiei de diferenţă. Ieşirea acestei tranziţii este de asemenea prezentă la nivelul acestui grafic şi este marcată de locaţia P4 desenată cu mov.

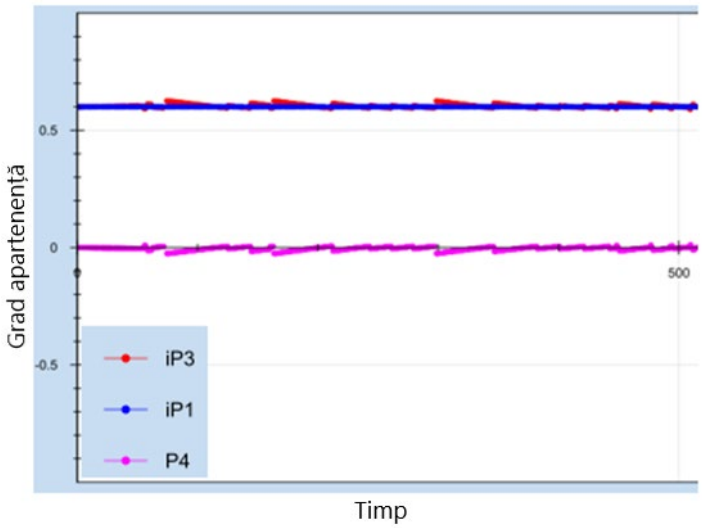

Figura 4.3.9. Intrările şi eroarea componentei RTC

La ieşirea regulatorului RTC se află comanda de oprire/pornire a componentei de încălzire a camerei fapt pentru care ieşirea prezintă două ramuri. În Figura 4.3.10, P6 este marcat cu culoarea neagră şi prezintă momentele de timp în care boilerul este oprit iar P7 este marcat cu roşu şi semnifică perioadele de timp în care componenta este pornită. Acest comportament de pornire/oprire este prezent şi la componenta ACC, fapt pentru care nu se reia explicarea acesteia ulterior.

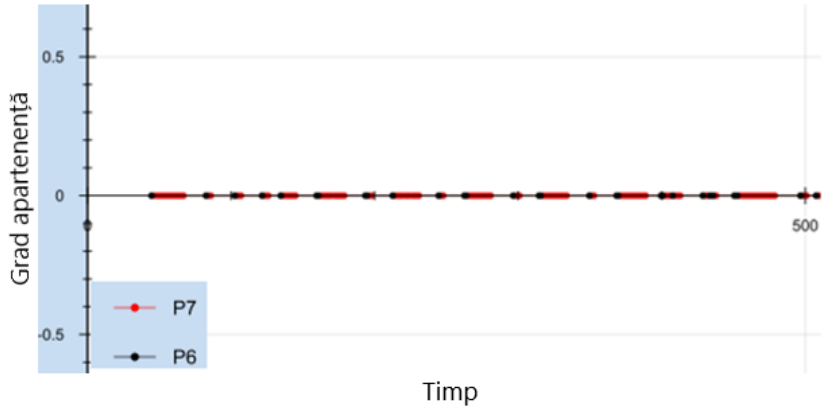

Figura 4.3.10. Comandă boiler oprit/pornit

Ultimul regulator din componenta sistemului este ACC și acesta este reprezentat mai jos în Figura 4.3.11.

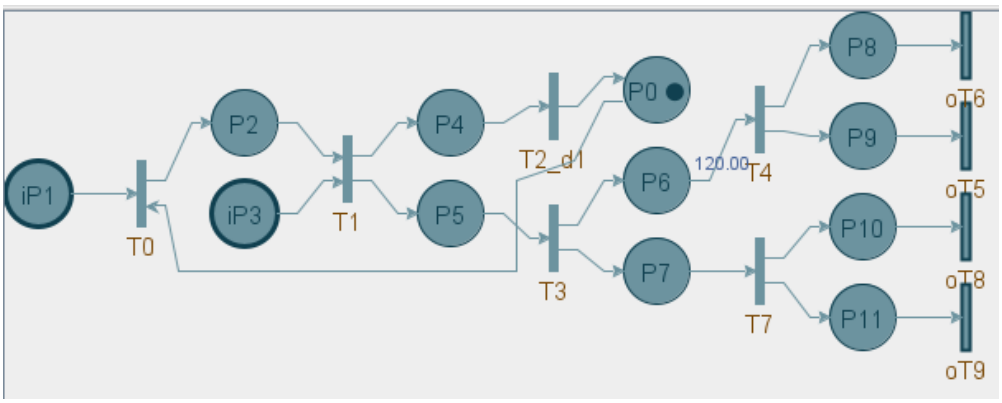

Figura 4.3.11. Regulator ACC modelat

La intrări se introduce valoarea temperaturii de referință și valoarea curentă iar eroarea se calculează în mod similar ca și la regulatoarele anterioare. La ieșire se prezintă 4 tranziții, 2 marchează prezența comenzii de oprire/pornire a regulatorului ACC în mod similar cu comanda prezentată în figura anterioară, iar al doilea set de tranziții ia decizia de a ejecta aer rece/cald de la componenta de aer condiționat, comportament care poate fi observat în Figura 4.3.12.

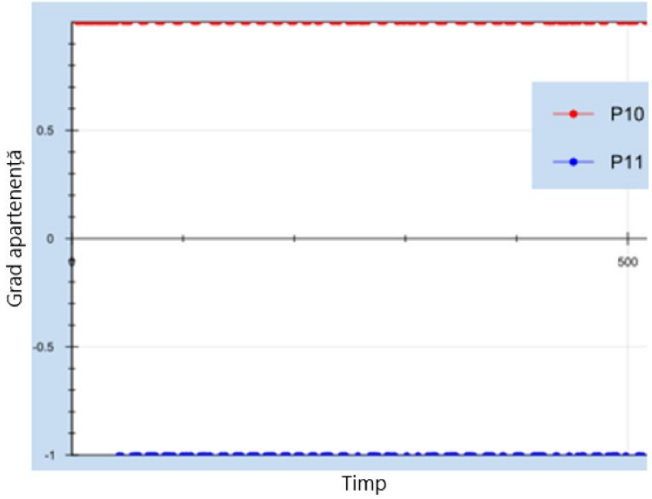

Figura 4.3.12. Comandă aer cald/rece

4.4 Rezilienţa aplicaţiei

Testarea rezilienţei are rolul de a asigura că, în ciuda unor modificări asupra aplicaţiei, aceasta este capabilă să îşi exercite funcţionalitatea în continuare. Dacă în urma unui factor extern, aplsicaţia îşi încetează funcţionarea în special în timpul perioadelor foarte reci, acest lucru duce la apariţia unei perturbaţii mari care necesită un consum mai ridicat de energie, astfel cu cât timpul de revenire la starea de funcţionare este mai mic, cu atât sistemul este mai eficient.

În acest ultim subcapitol se testează rezilienţa aplicaţiei la deconectarea unui regulator. Scopul urmărit este de a demonstra că sistemul este capabil să îşi continue funcţionarea în urma unui asemenea eveniment pentru a arăta că sistemul realizat poate funcţiona doar cu un anumit set de regulatoare.

Regulatoarele se pornesc la nivelul ferestrelor client prin acţionarea butoanelor de On/Off. În urma acţionării acestor butoane, serverul care se ocupă de gestionarea clienţilor afişează linii de text aferente comenzilor date în interiorul unui câmp de text prezent la nivelul ferestrei server. Atât timp cât serverul este pornit şi s-a selectat în prealabil scenariul de rulare pentru a porni modelul centralei, regulatoarele sunt capabile să se conecteze şi să se deconecteze ori de câte ori este nevoie. Afişare informaţiilor la nivel de fereastră server este prezentată în Figura 4.4.1 prezentată mai jos.

Figura 4.4.1. Fereastra Server

În funcţie de regulatoarele pornite, performanţele pot fi urmărite la nivelul ferestrei aferente. Existenţa celor două interfeţe client şi server permite utilizarea sistemului pe două dispozitive diferite, fapt ce marchează comportamentul unui sistem distribuit. Comunicarea între cele două se realizează cu ajutorul socket-urilor iar timpul de comunicare diferă în funcţie de caz: dacă ambele ferestre sunt pornite la nivelul aceluiaşi calculator, trasmiterea informaţiilor este realizată într-un timp mult mai scurt în comparaţie cu timpul de trasmitere în cazul în care aceste interfeţe ar rula pe dispozitive diferite.

5 Concluzii

5.1 Rezultate obţinute

Proiectul curent urmăreşte să aducă un aport în dezvoltarea sistemelor de control distribuite care să utilizeze reţele de tip FLETPN. Obiectivul propus la începutul acestei lucrări îl prezintă proiectarea unui sistem de reglare a temperaturii într-o încăpere pentru a sprijini reducerea consumului necesar de asigurare a confortului termic dorit. Utilizarea logicii fuzzy permite posibilitatea de a simula un comportament dinamic, capabil să răspundă la modificări externe pentru a realiza funcţia de control.

Modalitatea de implementare a aplicaţiei este realizată sub forma unui sistem distribuit pentru a permite posibilitatea de implementare ulterioară a noi componente care să comunice împreună. Avantajul principal constă însă în posibilitatea de a utiliza doar un set de regulatoare din cele prezente în funcţie de cerinţele impuse de utilizator. Acest aspect duce în consecinţă la reducerea complexităţii sistemului, la creşterea robusteţii aplicaţiei dar şi la oferirea unui tip de răspuns scurt. Prin realizarea acestui sistem care permite comunicarea dintre regulatoarele prezente, se poate realiza o gestionare mai simplă a funcţionalităţilor, acest lucru în consecinţă duce la creşterea robusteţii aplicaţiei.

Contribuţia principală adusă în domeniu este realizarea aplicaţiei demonstrative care permite testarea componentelor prezente în sistem. La nivelul acestei aplicaţii se remarcă prezenţa regulatoarelor fuzzy modelate astfel încât acestea să poată realiza corespunzător funcţia de control. La nivelul de luare a deciziilor, utilizatorul are posibilitatea de a alege scenariul de funcţionare cât şi numărul de regulatoare care să fie puse în funcţiune. Totodată, la nivelul regulatorului de tip termostat se poate alege temperatura de referinţă pe care să o urmărească sistemul.

O altă caracteristică importantă realizată la nivelul proiectului constă în posibilitatea de a urmări modificările de sistem. Prin intermediul aplicaţiei se pot vizualiza diferite elemente precum grafice sau comenzi de valori pentru a urmări comportamentul fiecărui element în parte şi pentru a putea analiza performanţele obţinute în urma rulării. În urma studierii acestor performanţe se constată că se obţine un set bun de performanţe, fapt pentru care se justifică necesitatea dezvoltării ulterioare în domeniu pentru a reduce costul de consum al unei centrale termice.

Un plus adus în domeniu cu această aplicaţie constă în faptul că spre deosebire de majoritatea sistemelor de încălzire realizată în această manieră, implementarea unui model matematic al unui boiler cât şi al unui aer condiţionat permite reglarea pe parcursul unui an întreg, nu doar pe un anumit anotimp. De regulă aceste sisteme există dar sunt adesea implementate separat, prin combinarea acestora însă se poate economisi la capitolul de implementare fizică, nefiind nevoie să se creeze un centru de comandă pentru două sisteme diferite. Mai mult decât atât, prezenţa unui aer

condiţionat care poate asigura şi aer cald este un factor adiţional care permite atingerea performanţelor minime în perioadele reci mult mai repede şi în cazul unei eventuale defectări a regulatorului care se ocupă de încălzirea apei, sistemul este în continuare capabil să realizeze reglarea într-o oarecare măsură.

Un alt avantaj adus la nivelul acestei aplicaţii constă în proiectarea acestuia în Java. De regulă un astfel de sistem este implementat în medii precum matlab dar prin utilizarea utilitarului FuzzPVisual, se facilitează posibilitatea de a dezvolta aplicaţia deoarece mediul Java prezintă un număr ridicat de framework-uri şi biblioteci. Totodată, maşina virtuală Java este independentă de platformă , fapt pentru care se permite rularea unui program pe orice sistem de operare.

În cele din urmă, contribuţia proprie generală se axează pe realizarea aplicaţiei demonstrative care permite urmărirea şi controlul unui sistem de reglare a termperaturii distribuit. Din cauza complexităţii aduse de un astfel de proiect implementat la scară reală, această lucrare se axează doar pe realizarea aplicaţiei software prin realizarea modelelor matematice ale componentelor existente şi crearea regulatoarelor aferente cu ajutorul reţelelor FLETPN. Mai mult de cât atât, se concep regulile fuzzy care acţionează la nivelul tranziţiilor şi se realizează în cele din urmă comunicarea dintre regulatoare şi centrala termică.

5.2 Direcţii de dezvoltare

Acest ultim capitol urmăreşte să ofere o serie de obiective care pot duce la dezvoltarea ulterioară a aplicaţiei. Prin realizarea acestui capitol, se facilitează substanţial urmărirea paşilor ce pot duce în viitor la completarea şi optimizarea mai în detaliu a sistemului de încălzire a unei camere realizat cu reţelele FLETPN.

Un prim pas de dezvoltare constă în realizarea unei implementări reale în care componenta software să comunice cu componentele hardware care în acest proiect sunt modelate în interiorul aplicaţiei. Acest lucru implică necesitatea de a interconecta componente precum aerul condiţionat sau boilerul care încălzeşte apa din calorifere pentru ca acestea să poată fi controlate de regulatoarele aferente.

La nivel de achiziţionare a datelor, apare nevoia utilizării a diferiţi tipi de senzori pentru a putea transmite regulatoarelor parametrii necesari. Astfel, este nevoie de o serie de senzori de temperatură pentru a achiziţiona: temperatura din cameră, temperatura apei din cadrul boilerului, temperatura vremii de afară şi eventual se pot adăuga şi senzori de temperatură la nivelul caloriferelor pentru a putea determina care este gradul de pierdere a nivelului temperaturii cauzat de transmiterea agentului termic pe ţevile de transport. Senzorii de temperatură pentru mediul extern şi cele puse la nivelul caloriferelor apar cu scopul de a permite transmiterea datelor la nivelul centralei ca aceasta să fie capabilă să rejecteze aceste perturbaţii. Tot la nivel de perturbaţii, se pot introduce şi alte tipuri de senzori precum senzorii ultrasonici care să fie amplasaţi pe uşile şi geamurile din încăpere, aceşti senzori fiind capabili să detecteze modificarea de stare. În cazul în care un geam sau o uşă este deschisă pentru o perioadă îndelungată

de timp, centrala trebuie să fie anunţată de această schimbare pentru a fi capabilă să compenseze la ajustarea temperaturii interne.

Într-o implementare fizică se doreşte nu doar reglarea temperaturii într-o singură cameră, ci în întreaga locuinţă. Datorită acestui fapt, se pot introduce în sistem mai multe camere a căror factor de încălzire diferă. De exemplu, o cameră mică se încălzeşte mult mai rapid decât o sufragerie, însă indiferent de cameră se doreşte ca procesul de încălzire să fie egal în ambele încăperi. Astfel, pentru fiecare cameră, se poate crea un model matematic şi se pot modela în aşa fel încât în funcţie de mărimea încăperii sau de faptul că o cameră este conectată cu un perete extesssrior. Prin introducerea acestor parametrii, centrala poate fi controlată în aşa fel încât în încăperea mai mare să se compenseze mai mult. Evident, un alt beneficiu al acestui tip de implementare constă în faptul că în funcţie de cerinţele utilizatorului, de multe ori nu se doreşte încălzirea tuturor camerelor din locuinţă deoarece controlul temperaturii asupra camerelor neutilizate duce la un consum în plus de care beneficiarul nu profită.

Datorită faptului că se utilizează reţele FLETPN, implementarea de noi elemente este posibilă. Astfel, se pot introduce alte tipuri de regulatoare care să ţină cont de factorii externi. De exemplu se poate crea un Light Intensity Controller(LIC) care, în funcţie de cât de prezent este pe cer soarele într-o anumită zi, centrala poate să compenseze în plus la ajustarea temperaturii. În manieră similară, există posibilitatea creeri unui regulator Wind Speed Controller (WSC) care în funcţie de cât de mare este viteza vântului, acesta să poată trimite un semnal către centrală. Atât apariţia soarelui cât şi prezenţa vântului pot într-o oarecare măsură să modifice temperatura de afară, fapt pentru care s-ar putea justifica implementarea unor astfel de regulatoare.

Această aplicaţie demonstrativă este accesibilă sub forma unei aplicaţii disponibile pe un calculator personal. O implementare mai bună poate fi realizarea unei aplicaţii de mobil pentru aceasta. Acest lucru ar fi mult mai practic deoarece majoritatea persoanelor deţin un telefon inteligent, dar numărul de persoane care deţin şi un calculator este puţin mai redus. Astfel utilizarea unui astfel de sistem ar fi mult mai atractivă pentru un potenţial client şi ar facilita posibilitatea de a monitoriza şi ajusta comportamentul sistemului de la distanţă.

Crearea unei aplicaţii pe mobil duce în consecinţă la necesitatea adăugării unui nivel de securitate asupra sistemului. Astfel, un prim strat poate consta în logarea utilizatorului la aplicaţie cu ajutorul unui set de credenţiale de tip utilizatori şi parolă. Mai mult decât atât, se poate include utilizarea aplicaţiei doar dacă telefonul este conectat la un Virtual Private Network(VPN). În ultimii se observă tot mai mult apariţia autentificării în doi factori, fapt pentru care un astfel de implementare ar spori semnificativ nivelul de siguranţă. La logare se poate cere un cod care este trimis de către server pe adresa de email asociată contului pentru a verifica că utilizatorul este întradevăr cel care doreşte să acceseze funcţionalităţile aplicaţiei.

Un ultim mod de a dezvolta aplicaţia poate fi conectarea sistemului cu o bază de date care să stocheze date precum: temperatura zilnică în fiecare oră, gradul de consum al energiei sau chiar cât de mult este deshisă uşa sau geamul într-o zi, eventual chiar şi o asociere cu intervalul de timp în care acestea două sunt deschise. Prin implementarea

acestei baze de date se poate realiza un control predictiv în funcţie de anotimp dar şi în funcţie de comportamentul utilizatorului asupra sistemului. Prin introducerea acestor date într-un sistem de Machine Learning ar fi posibilă o eventuală creştere graduală a performanţei sistemului cu cât se injectează mai multe date. Astfel, consumul raportat la fiecare anotimp ar scădea odată cu costurile de întreţinere.

În cele din urmă, se observă o gamă largă de direcţii de dezvoltare a aplicaţiei, fapt pentru care se doreşte o extindere a popularităţii în domeniu pentru a stimula crearea de noi tehnologii mai avansate nu doar la nivelul reglării temperaturii, ci în general la nivelul realizării de sisteme inteligente. În consecinţă, prin diseminarea acestui domeniu de lucru se poate garanta un viitor în care confortul este sporit şi mediul de viaţă este în general îmbunătăţit prin facilitatea diferitelor nevoilor umane.

6 Bibliografie

[1] N. Kopmann, R. Streblow and D. Müller, "Test of new control strategies for room temperature control systems fully controllable surroundings for a heating system with radiators," 2015 International Conference on Smart Cities and Green ICT Systems (SMARTGREENS), Lisbon, Portugal, 2015, pp. 1-6

[2] https://users.utcluj.ro/~ocuibus/

[3] Ross, T. J. (2010). Fuzzy Logic with Engineering Applications.pg. 134-135

[4] T. S. Letia and A. O. Kilyen, "Fuzzy logic enhanced time Petri Net models for hybrid control systems," 2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2016, pp. 1-6

[5] J. C. Mugisha, B. Munyazikwiye and H. R. Karími, "Design of temperature control system using conventional PID and Intelligent Fuzzy Logic controller," 2015 International Conference on Fuzzy Theory and Its Applications (iFUZZY), Yilan, Taiwan, 2015, pp. 50-55

[6] S. Y. Liao, H. Q. Wang and W. Y. Liu, "Functional dependencies with null values, fuzzy values, and crisp values," in IEEE Transactions on Fuzzy Systems, vol. 7, no. 1, pp. 97-103, 1999

[7] A. Pourabdollah, "Fuzzy Number Value or Defuzzified Value; Which One Does It Better?," 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 2020, pp. 1-6

[8] M. M. Pirbazari, A. Khoei and K. Hadidi, "Optimization of inference engine in CMOS analog fuzzy logic controllers," 2013 21st Iranian Conference on Electrical Engineering (ICEE), Mashhad, Iran, 2013, pp. 1-6

[9] T. Murata, "Petri nets: Properties, analysis and applications," in *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541-580, 1989

[10] T. S. Letia and O. Kilyen, "Enhancing the time Petri nets for automatic hybrid control synthesis," 2014 18th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 2014, pp. 621-626

[11] O. Cuibus and T. Leția, "Approaching Problems with Different Kinds of Petri Net Models," 2022 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2022, pp. 1-6

[12] https://en.wikipedia.org/wiki/Thermal_power_station

[13] https://www.viessmann.ro/ro/blog/cele-mai-populare-sisteme-de-incalzire-folosite-la-ora-actuala-in-europa.html

[14] A. Chulenyov, K. Agakhanova and D. Abramkina, "The Use of Condensing Boilers in Autonomous Heating Systems," *2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, Vladivostok, Russia, 2020, pp. 1-4

[15] https://www.quickshop.ro/ce-este-histereza-si-cum-reglezi-termostatul-a154

[16] https://eprofu.ro/docs/mm/12/sisteme-reglare-automată/fisa%20de%20 documentare% 2011%20-%20Regulatoare%20neliniare.pdf

[17] Z. Shi and C. Wang, "Application of fuzzy control in temperature control systems," *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, Harbin, China, 2011, pp. 451-453

[18] Andrews, Gregory R, Foundations of Multithreaded, Parallel, and Distributed Programming, Addison–Wesley, 2000, pp. 2-3,21-23,291-294

[19] B. S. Rawal, L. Berman and H. Ramcharan, "Multi-client/Multi-server split architecture," *The International Conference on Information Networking 2013 (ICOIN)*, Bangkok, Thailand, 2013, pp. 696-701

[20] B. Davey and A. Tatnall, "Tools for client server computing," Proceedings 1996 International Conference Software Engineering: Education and Practice, Dunedin, New Zealand, 1996

[21] https://www.simplilearn.com/what-is-client-server-architecture-article

[22] https://stackify.com/oops-concepts-in-java/

[23] https://sourcemaking.com/design_patterns/

[24] https://www.turing.com/blog/software-architecture-patterns-types

[25] https://refactoring.guru/design-patterns/factory-method

[26] https://github.com/AttilaOrs/FuzzP/tree/master

[27] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/