VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Išmaniųjų elektros tinklų modeliavimas ir verifikavimas remiantis statistinio modelių patikrinimo metodais

Modelling and verification of smart grid electrical systems using statistical model checking methods

Magistro baigiamasis darbas

Atliko: Gabrielė Dėdinaitė (parašas)

Darbo vadovas: dr. Laibinis Linas (parašas)

Recenzentas: Karolis Petrauskas (parašas)

Vilnius – 2024

Turinys

Iliustracijų turinys

Santrumpų ir terminų žodynas

Išskirstytos sistemos – sistemos, kurių programinė įranga veikia integruotoje grupėje bendradarbiaujančių procesorių, kurie komunikuoja ir koordinuoja savo veiksmus, siųsdami pranešimus vieni kitiems [1].

Sistemų modeliavimas – procesas, kurio metu kuriamas abstraktus sistemos modelių rinkinys, kai kiekvienas modelis pristato skirtingą tos sistemos perspektyvą [2].

Išskirstytų sistemų verifikavimas – programinės įrangos kontekste, programinių sistemų verifikavimas įrodo arba paneigia funkcinių arba nefunkcinių reikalavimų korektiškumą (angl. *correctness*). Verifikavimas vyksta naudojantis matematiškai paremtais įrodymo metodais, pagal nagrinėjamos sistemos specifikaciją [3].

Statistinis modelis – matematinis modelis ir statistinės prielaidos naudojamos generuoti duomenis ir daryti prognozes apie ateitį [4].

Kiberfizinės sistemos – kompiuterinės sistemos, kurių mechanizmai yra kontroliuojami arba stebimi, naudojantis kompiuteriniais algoritmais. Kiberfizinėse sistemose fizinės ir programinės įrangos elementai yra stipriai sujungti ir gali veikti skirtingose erdvinės ir laiko skalėse, demonstruoja daug ir skirtingų elgesio būdų ir sąveikauja tarpusavyje būdais, kurie keičiasi atsižvelgiant į kontekstą [5].

Daiktų internetas – sistema sudaryta iš tarpusavyje susijusių prietaisų, mechaninių ar skaitmeninių mašinų, objektų, gyvūnų, kurie turi unikalius identifikatorius ir galimybę perduoti duomenis per tinklą, nereikalaujant žmogaus su žmogumi ar žmogus su kompiuteriu sąveikos [6].

Perteklinių būsenų skaičius „sprogimas“ (angl. *state explosion*) – situacija, kai būsenų ar skirtingų situacijų, kuriose sistema gali atsidurti jos veikimo metu, skaičius yra toks didelis, kad patikrinti kiekvieną tokią būseną yra praktiškai neįmanoma [7].

Išmanusis elektros tinklas (angl. *smart electrical grid*) – elektros tinklas leidžiantis dvipusį elektros ir duomenų srautą [8].

Išmanusis elektros skaitiklis (angl. *smart meter*) – skaitiklis (prietaisas, kuris renka elektros sunaudojimo informaciją), turintis kompiuterinę įrangą, kuri matuoja elektros sunaudojimą ir siunčia sunaudojimo informaciją elektros tiekimo įmonei [9].

Heterogeninės sistemos – sistemos, kurios yra sudarytos iš skirtingo tipo kompiuterinių vienetų. Paprastai kompiuteriniai vienetai yra kasdieninio naudojimo procesoriai veikiantys ant operacinės sistemos [10].

Dvipusis elektros tekėjimas (angl. *bi-directional power flow*) – elektros energijos tekėjimas į abi puses. Šis procesas yra aktualus, kai sistemos aktorius gali tiekti elektros energiją į sistemą ir naudotis sistemoje esančia energija (pvz. saulės baterijos ant privačių namų – vartotojas gali naudotis saulės elektra ir tiekti ją į sistemą esant pertekliui, arba esant elektros trūkumui naudotis sistemos tiekiama elektra) [11].

Verifikacija - ko nors autentiškumo patvirtinimas; patvirtinimo procedūra [12].

ĮVADAS

Nagrinėjamos temos aktualumas

Pastebima tendencija, kad programų sistemų vystymasis lėmė išskirstytų programų sistemų sudėtingumo augimą ir naujų tipų sistemų (tokių kaip kiberfizinės sistemos, išmanieji elektros tinklai, daiktų internetas ir kt.) atsiradimą [13]. Šios sistemos reikalauja didelių žmogiškųjų ir kitų įvairių resursų jų palaikymui. Taip pat jos naudojamos kritinėse infrastruktūrose ir jų klaidos gali kainuoti didelius finansinius nuostolius, nutraukti kritinių paslaugų tiekimą gyventojams, sukelti pavojų aplinkai, ekosistemoms ar net žmogaus sveikatai ir gyvybei. Tai ypač aktualu kalbant apie išmaniuosius elektros tinklus. Elektros tinklų gedimai, blogas informacijos perdavimas, nesugebėjimas balansuoti elektros poreikio ir paklausos dėl pateiktų klaidingų duomenų, priveda prie finansinių nuostolių, fizinės ir programinės įrangos gedimo, nesugebėjimo užtikrinti kokybiško elektros tiekimo ir kt. pasekmių darančių neigiamą įtaką visuomeniniam gyvenimui. Siekiant išvengti šių situacijų ir nuostolių reikia taikyti griežtesnius verifikavimo metodus kuriant šias sistemas [14].

Formalus išskirstytų programų sistemų reikalavimų verifikavimas remiasi matematiniais metodais. Modelių patikrinimas – plačiausiai naudojamas formalaus verifikavimo metodas [13]. Klasikinis modelių tikrinimas yra „mechaninis“. Visų sistemos būsenų patikrinimas vyksta remiantis pasiekiamumo grafu, leidžiančiu patvirtinti ar paneigti reikiamų sistemos savybių ar reikalavimų galiojimą ar tenkinimą. Pagrindinė metodo silpnybė – galimas tikrinamų būsenų skaičiaus „sprogimas“ (angl., *state explosion*), t.y., situacija, kai galimų tikrinamų sistemos būsenų kombinacijų kiekis yra per didelis kompiuteriniams resursams (t.y., įvyksta perkrova). Tuo tarpu statistinis modelio tikrinimo metodas yra modelių patikrinimo metodo atmaina, kuri nereikalauja perrinkti visų galimų sistemos būsenų ir grąžina rezultatą su tam tikru patikimumu (tikimybe), tuo pačiu statistiškai įvertinant sistemos savybių įvykdomumą [14]. Sistemos modelis tampa sistemos prototipu, kuris yra vykdomas daug kartų, su skirtingomis būsenų kombinacijomis, akumuliuojant statistinę informaciją apie sistemą.

Vienas iš įrankių, kuris yra skirtas naudoti statistinį modelio tikrinimo metodą, yra Uppaal. Uppaal yra integruotas programinis įrankis skirtas modeliavimui, validacijai ir verifikavimui naudojant realaus laiko sistemas [15]. Šis įrankis leidžia verifikuoti tokias sistemos savybes, kaip

pasiekiamumo (*angl. reachability*), laiko (*angl. timing*), aklaviečių nebuvimo (*angl. deadlock freeness*), kintamumo (*angl. invariant*), išreikštas kaip laiko logikos (*angl. temporal logic*) formulės. Visos šios savybės yra aktualios išskirstytų sistemų verifikavimui. Šis įrankis taip pat yra rinkos lyderis ir vienas iš lengviausiai suprantamų ir išmokstamų įrankių statistinių modelio tikrinimo srityje [16].

Darbo objektas

Šio darbo objektas – išmaniojo elektros tinklo sistema ir jos identifikuotų savybių modeliavimas ir verifikavimas statistinio modelių metodų patikrinimo metodais.

Darbo tikslas

Šio darbo tikslas – sukurti išmaniojo elektros tinklo sistemos modelį (prototipą), identifikuoti jo pagrindines savybes, gedimus ir gedimų sprendimus, išanalizuoti modelį statistinio modelių patikrinimo metodais, naudojant Uppaal įrankio aplinką ir pateikti analizės rezultatus.

Darbo uždaviniai

Darbo tikslui pasiekti iškelti šie uždaviniai:

1. Identifikuoti išmanaus elektros tinklo savybes, kylančias iš esminių sistemos reikalavimų, bei sistemos parametrus, nuo kurių tiesiogiai priklauso sistemos darbas.
2. Sukurti išmanųjį elektros tinklų modelį (prototipą) Uppaal modelių tikrintojo aplinkoje.
3. Sumodeliuoti kraštutinius elektros tinklo gedimus ir atvaizduoti juos modelyje.
4. Sukurtą išmanaus elektros tinklo modelį išanalizuoti statistiniais modelių patikrinimo metodais, identifikuotų savybių ir skirtingų parametrų reikšmių atžvilgiu.
5. Analizės pagrindu surasti silpnąsias sistemos veikimo vietas, bei palyginti skirtingas sistemos konfigūracijas.

Darbo praktinė vertė

Šiame magistro darbe, sukurtas išmaniojo tinklo modelio prototipas Uppaal modelių tikrintojo aplinkoje, identifikuotos sistemos savybės, kylančios iš esminių reikalavimų, bei sistemos parametrai, nuo kurių tiesiogiai priklausys sistemos darbas. Sukurtas modelis simuliuoja kraštutinius gedimo scenarijus, tų gedimų paiešką, atradimą ir sistemos darbą gedimo metu. Rezultatai išanalizuoti

statistiniais modelių pratikrinimo metodais, identifikuotų savybių ir skirtingų parametru reikšmių atžvilgiu. Analizės metu surastos silpnosios sistemos veikimo vietos ir palygintos skirtingos sistemos konfigūracijos. Šio darbo metu pademonstruota, kaip yra svarbu naudoti modelio patikrinimo metodus ir modelio tikrintojo įrankį dar sistemos kūrimo (dizaino) stadijoje. Rezultatai įrodo įrankio panaudojimo vertę esminių/kritinių išmaniųjų elektros tinklų savybių verifikavimo, silpnųjų sistemos veikimo vietų atradimo, skirtingų sistemos konfigūracijų, susijusių su skirtingomis sistemos parametrų reikšmėmis, palyginimo, siekiant surasti optimalią pagal turimus resursus ir kt. situacijose.

1. Analitinė dalis

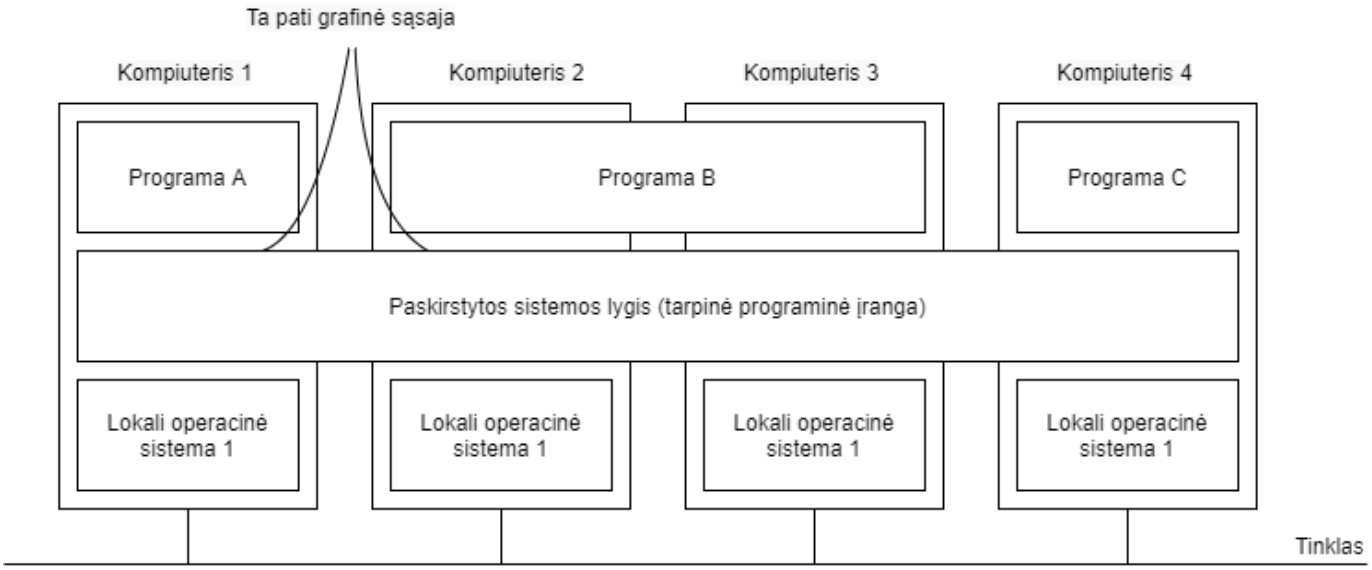
Analitinėje dalyje bus susipažinta su mokslinių šaltinių informacija, kuri yra susijusi su šio darbo tema. Siekiant įsigilinti ir labiau paaiškinti darbo temą, bus renkama ir analizuojama informacija susijusi su išskirstytų sistemų verifikacija. Ši analitinė darbo dalis bus sudaryta iš trijų dalių:

1. Kiberfizinių sistemų analizė. Šioje dalyje bus aptarta šių sistemų požymiai, charakteristikos, reikiamos esminės savybės, ir kaip tokių sistemų pavyzdys, išnagrinėta išmaniojo elektros tinklo sistema.
2. Metodų, kurie verifikuoja ir patikrina išskirstytas sistemas apžvalga. Šioje dalyje bus aptarta kiberfizines sistemas verifikuojantys metodai, skirtumai tarp modelių patikrinimo ir verifikavimo teoremų įrodymo būdo, modelio patikrinimo privalumai ir trūkumai, tikrinamų savybių išreiškimas laiko logika, laiko automatai ir statistinis modelių patikrinimas.
3. Uppaal įrankio aprašymas. Šioje dalyje bus analizuojamas Uppaal modelių patikrinimo įrankis, jo modeliavimo kalba, teoriniai pagrindai ir statistinis modelių tikrinimo metodas.

Šiuo darbu siekiama įsigilinti į modeliuojamos sistemos savybes ir charakteristikas, išanalizuoti naudojamus verifikavimo metodus ir susipažinti su Uppaal modelių patikrinimo įrankiu.

1.1. Išskirstytos sistemos ir elektros tinklai

Kompiuterinės ir programinės įrangos tobulėjimas suteikė galimybę lengvai sujungti didelį kiekį tinkle esančių kompiuterių, sukuriant sudėtingos kompiuterių architektūros sistemas. Tokios sistemos yra vadinamos išskirstytomis sistemos. Šios sistemos yra sudarytos iš autonomiškų kompiuterinės ir programinės įrangos dalių ir jų naudotojams atrodo, kaip vientisa sistema [1].



1 Pav. Išskirstytos sistemos architektūra realizuota tarpinės programinės įrangos lygyje, kuris tęsiasi per kelis kompiuterius **[1]**

Diagramoje (1 Pav. Išskirstytos sistemos architektūra realizuota tarpinės programinės įrangos lygyje, kuris tęsiasi per kelis kompiuterius) atvaizduojami keturi kompiuteriai su trejomis programomis, iš kurių programa B veikia naudodamasi kompiuterio 2 ir kompiuterio 3 resursais. Kiekvienas kompiuteris yra vienos sistemos dalis ir atlieka tam tikrą paskirtį šioje sistemoje. Šioje architektūroje komunikuoja ne tik fiziniai išskirstytos sistemos komponentai (per tarpinę programinę įranga), bet ir programinei sistemos elementai. Išskirstyta sistema naudotojui nerodo skirtumų tarp kompiuterių, kompiuterinės įrangos, lokalių operacinių sistemų ir atrodo, kaip vientisa sistema.

Tarpinė įranga turi panašumų su operacine sistema, nes užsiima resursų valdymu, koordinuoja programų veiklą ir kompiuterių resursus, efektyviai padalinant juos per tinklą. Tarpinė įranga taip pat atlieka šiuos veiksmus:

- Koordinuoja ryšį tarp sistemos komponentų;
- Užtikrina sistemos apsaugą;
- Suteikia apskaitos paslaugas;
- Tęsia sistemos darbą sistemos gedimų atveju (kompensuoja įvykusius gedimus) ir padeda komponentams atsigauti po klaidų.

Išskirstytos sistemos taip pat gali būti apibūdintos, kaip programinė įranga, kuri atlieka veiksmus pagal protokolų sąrašą, tuo siekiant koordinuoti procesorių darbą, kurie yra sujungti per komunikacinį tinklą [17]. Šioje sistemoje visi komponentai dirba kartu, atlikdami vieną užduotį dalimis arba mažą sąrašą glaudžiai susijusių užduočių. Kiberfizinės sistemos yra išskirstytų sistemų pavyzdys. Šios sistemos yra programinės įrangos ir fizinių komponentų integracija [5]. Sistemoje esantys prietaisai stebi ir kontroliuoja fizinius procesus. Šių prietaisų pagalba sistema reaguoja į nuolat atnaujinamą informaciją ir pakeičia sistemos procesus arba modifikuoja sistemos fizinę aplinką. Kiberfizinės sistemos veikia išskirstytų sistemų principu [18], jos yra hierarchinės, išskirstyto valdymo sistemos (turi išskirstytų sistemų bruožų).

Elektros tinklas yra išskirstyta sistema ir vienas iš pagrindinių iššūkių šiuolaikiniais laikais yra atsinaujinančių elektros šaltinių integraciją į sistemą. Nors šie šaltiniai mažina išmetamą anglies dioksido kiekį, jų nepastovumas reikalauja atidumo sistemos stebėjimo ir kontrolės atžvilgiu. Siekiant patenkinti šį poreikį svarbu integruoti realaus laiko elektros gaminimo ir sunaudojimo stebėjimo sistemą [19], kuri taip pat pagerina elektros energijos patikimumą, efektyvumą, padeda aptikti gedimus sistemoje ir juos paslėpti, sumažina elektros kainą iš vartotojo ir gamintojo pusės (sumažina elektros kiekio paklausos ir pasiūlos pertekliaus situaciją). Gebėjimas stebėti, dalintis ir kontroliuoti sistemos informaciją ir veiksmus yra kiberfizinių sistemų bruožas [20].

Toliau bus aptariama kiberfizinių sistemų požymiai, jų savybės ir išmaniojo tinklo sistemos ir jo charakteristikos.

1.1.1. Kiberfizinių sistemos

Šiame skyriuje aptarsime kiberfizinių sistemų požymius. Gilinantis į kiberfizines sistemas jos yra apibūdinamos, kaip sistemos su integruota programine įranga [20] (integruota į namus, prietaisus, transporto priemones, medicininius procesus ir t.t.) , kurios:

- Tiesiogiai surenka fizinius duomenis, naudojantis fiziniais jutikliais ir daro įtaką realaus pasaulio procesams, naudojantis prietaisais, kurie daro fizinę įtaką aplinkai.
- Įvertina ir saugo surinktus duomenis, aktyviai arba pasyviai sąveikauja su fiziniu ir skaitmeniniu pasauliu.
- Fizinius įrenginius sujungia vieną su kitu per tinklą, naudojantis skaitmeniniais komunikacijos įrenginiais.
- Naudojasi viešai prieinamais duomenimis ir paslaugomis (angl. *services*).
- Turi daugybę specialiai paruoštų žmogaus sąveikos sąsajų.

Šios sistemos gali išspręsti įvairaus sudėtingumo realaus pasaulio problemines situacijas, darydamos tiesioginę įtaką fizinei aplinkai. Kiberfizinių sistemų sprendžiamų problemų svarbumo aibė gali prasidėti nuo paprasto žmogaus kasdieninio gyvenimo situacijos ir baigtis visuomenei ar net planetai aktualiomis probleminėmis sritimis. Kiberfizinių sistemų įvairovė matoma apžvelgus šių sistemų pavyzdžius: buitiniai prietaisai prijungti prie interneto, išmanieji namai, transporto maršruto planavimo sistemos, medicinos procesų analizės ir planavimo sistemos ir t.t.

Kiberfizinių sistemų savybės:

- **Reaktyvus skaičiavimas** (*angl. Reactive Computation*) – reaktyvi sistema reaguoja į aplinkos pokyčius ir įvestį, parengdama aplinkai įtaką darantį atsaką.
- **Lygiagretumas** (*angl. Concurrency*) – skirtingų operacijų/procesų/algoritmų vykdymo išskirstymas keliems procesoriams ar kompiuteriams.
- **Fizinio pasaulio kontrolė grįžtamuoju ryšiu** (*angl. Feedback Control of the Physical World*) – valdymo sistema reaguoja į pokyčius fiziniame pasaulyje. Sistema nuolat reaguoja į atnaujinamą jutiklių informaciją ir atlieka veiksmus, kurie daro įtaką realiam pasauliui.
- **Realaus laiko skaičiavimas** (*angl. Real-Time Computation*) – realaus laiko duomenų apdorojimas yra kritiškai svarbus kiberfizinėms sistemoms.

- **Kritinis saugumo svarbumas** (*angl. Safety-Critical Applications*) – sistemos, kurių atsparumas kritinių klaidų atžvilgiu, turi didesnį prioritetą, nei kiti sistemos kūrimo tikslai (kūrimo kaina ar programos įrangos našumas) [22].

1.1.2. Išmaniojo elektros tinklo sistemos

Išmaniojo elektros tinklo sistemos terminas naudojamas apibūdinti elektros paskirstymo infrastruktūrą su naujausia kompiuterinės ir komunikacinis įrangos technologija, kuri masiškai nagrinėjama visame pasaulyje, siekiant ja pakeisti dabartinę sistemą [23]. Ši sistema gali užkirsti kelią pasikartojančioms klaidoms ir elektros dingimams, kurių rizika pakilo spartinant atsinaujinančių energijos šaltinių integraciją (vėjo, saulės ir geoterminės elektros generatoriai), bei gerinant elektrinių automobilių integraciją į visuomeninį transportą ir kasdieninį gyvenimą.

Išmaniojo elektros tinklo sistemos pastaruoju dešimtmečiu buvo labai tyrinėjamos ir analizuojamos, kuriant jų modelius ir prototipus. Mokslo tiriamieji darbai, kurie koncentruojasi į efektyvią energijos paskirstymo ir tiekimo sistemą, identifikavo du dalykus, kurie turi pasikeisti. Ateities infrastruktūra turi būti pasiruošusi priimti atsinaujinančius elektros šaltinius ir įdiegti dvipusį elektros energijos tiekimą. Siekiant integruoti paskirstytus atsinaujinančius elektros šaltinius ir paskirstytos energijos kaupimo įrenginį į tinklą, sistema turi tapti išmanesnė. Išmanusis elektros tinklas sugeba palaikyti tvarią ir patikimą elektros infrastruktūrą, kur kiekvienas sistemos narys (naudotojas, elektros tiekėjas ir operatorius) yra aktyvus realaus laiko energetikos rinkos dalyvis.

1.1.2.1. Išmaniųjų elektros tinklų charakteristikos

Federalinis energijos reguliavimo komitetas identifikavo šias pagrindines išmaniųjų elektros tinklų charakteristikas [24]:

- **Išmaniųjų skaitiklių infrastruktūra** *(angl. advanced metering infrastructure)* sistemos, kurių pagrindinis tikslas yra stebėti klientų skaitiklių duomenis. Šie skaitikliai leidžia dvipusį ryšį, kuris suteikia daug naujų privalumų ir galimybių, pvz.: namų ūkis gali tiekti savo duomenis elektros kompanijai ir tuo labiau optimizuojamas namų ūkio elektros sistemą, arba namų ūkis gali tapti elektros tiekėju ir tiekti elektrą į elektros tinklą.

- **Atsakas į paklausą** (*angl.* demand response) – laikinas pasikeitimas elektros rinkoje, reaguojant į pasikeitusius vartotojų poreikius. Dėl išmaniojo elektros tinklo lanksčios reakcijos į pasikeitusią paklausą, atsiranda galimybės:
    - Leisti vartotojui įsitraukti į rinką, pateikiant sunaudojimo ir kainos plano pasirinkimus;
    - Atverti rinką smulkaus elektros kiekio gamintojams, paskirstytiems generatoriams, pardavėjams ir pirkėjams;
    - Kontroliuoti maksimalaus elektros sunaudojimo sąlygas ir panaikinti elektros vogimo sąlygas;
    - Greitai reaguoti į elektros tinklo anomalijas;
    - Maksimizuoti gaunamą ir tiekiamą elektrą.
- **Elektrinės transporto priemonės** (*angl. electric vehicles*) – elektrinių automobilių inkorporavimas į išmanųjį elektros tinklą gali sukurti naujų problemų arba išspręsti tas problemas priklausant nuo šių automobilių krovimo laiko.
- **Plačios sistemos būsenos žinojimas** (*angl*. wide-area situational awareness) – šiuo atveju sistema stebi savo programinių ir fizinių komponentų darbą ir žino jų situaciją (veikimo stadijos, gedimai, anomalijos ir kt. situacijos galinčios paveikti sistemos darbą). Ši sistema atsako į šiuos klausimus:
    - Kokia yra elektros sistemos komponentų būsena? Kokia yra informacinės sistemos būsena?
    - Kokios yra elektros sistemos komponentų galimybės ir funkcionalumas? Kaip šie komponentai bendrauja su visa sistema?
    - Kaip kontrolės sistema paveiks bendrą sistemos būseną? Kokie yra parinkti sprendimai efektyviai valdyti sistemos darbą, pataisyti sistemos trukdžius ir atkurti nutrauktus elektros tinklo procesus?
- **Paskirstyti elektros resursai ir talpyklos** (*angl. distributed energy resources and storage*) – elektros kaupimo resursai, kurie yra naudojami elektros paklausos išaugimo atveju. Dabar labiausiai paplitusi elektros energijos talpykla yra hidroelektrinės.

- **Elektros paskirstymo sistemos valdymas** (*angl*. distribution grid management) – egzistuojanti elektros tiekimo ir paskirstymo sistema yra sudaryta iš šimtų elektros tiekėjų, tūkstančių paskirstymo transformatorių, tiekiančių elektrą milijonams klientų. Siekiant suvaldyti šį elektros paskirstymą naudojami lokalūs valdikliai, kurie prižiūri mažų sistemos dalių veikimą (rajonų ir regionų paklausą) ir dideli globalūs valdikliai prižiūrintys lokalių valdiklių darbą ir perduodantys rezultatus bendrajai sistemai. Siekiant maksimizuoti elektros paskirstymo proceso potencialą, didelis kiekis informacijos turi keliauti tarp elektros tinklo aplinkos, tinklo perdavimo, paskirstymo ir klientų sistemų.

1.2. Dažniausiai pasikartojančios problemos išmaniųjų tinklų sistemose

Reaguojant į tradicinių elektros tinklų negebėjimą patenkinti išaugusio elektros poreikio, buvo suprojektuotos šiuolaikinės išmaniųjų elektros tinklų sistemos. Tačiau šios sistemos nėra tobulos ir gali sugesti. Šie gedimai gali turėti didelių pasekmių, tad svarbu žinoti dažniausiai pasikartojančius gedimus, juos numatyti ir jiems pasiruošti. Šiame dokumente [25] yra identifikuoti 30 dažniausiai pasikartojantys gedimai išmaniuosiuose tinkluose ir būdai kaip tuos gedimus pašalinti ar kokių veiksmų imtis norint jų išvengti. Iš šių 30 gedimų išrinkti 5, kurių gedimai ir sistemos gebėjimas juos atpažinti ir į juos sureaguoti bus vėliau modeliuojami ir analizuojami, kartu vertinant tiriamos sistemos efektyvumą.

1.2.1. Ryšio nutrukimas tarp išmaniojo skaitiklio ir vietinio valdiklio

Šis gedimas atsiranda, kai išmanusis skaitiklis nebesiunčia elektros sunaudojimo informacijos ir ryšys nutrūksta tarp išmanaus skaitiklio ir vietinio valdiklio. Paprastai tai nutinka dėl tyčinio gadinimo iš kliento pusės. Ignoruojant šį gedimą sistema negalės išsiųsti kritiškai svarbių pranešimų laiku. Tai privestų prie išmaniųjų tinklų sistemos negebėjimo kritiškai įvertinti savo būsenos bei elektros pasiūlos ir paklausos balanso iškraipymo.

1.2.2. Tyčiniai gadinimai išmaniųjų jutiklių tinkle

Ši situacija įvyksta kai išmanusis skaitiklis yra modifikuojamas siųsti modifikuotus signalus į skaitiklio valdiklį. Šiuo atveju pasekmės yra panašios į prieš tai minėtas, taip pat apsunkinant tikslų

energijos sąnaudų duomenų kaupimo procesą, energijos pasiūlą ir skaitiklio patikimumo rodiklį (išmaniojo skaitiklio patikimumas).

1.2.3. Išsinchronizavę matavimai

Šis sutrikęs informacijos perdavimas įvyksta, kai išmanieji skaitikliai negali laiku perduoti elektros sunaudojimo ir generacijos informacijos, todėl informacija ateina vėluodama. Šie gedimai gali būti tyčinė žala iš vartotojo pusės arba sistemos klaida. Abejais atvejais gedimai iškreipia elektros paklausą ir pasiūlą, trukdant gedimų aptikimui ir išmaniųjų tinklų sistemos valdymo funkcijoms.

1.2.4. Sistemos negebėjimas surinkti tikslios informacijos

Ši situacija atsiranda, kai aukščiau minėti gedimai iškraipo elektros pasiūlos ir paklausos rodiklius. Tokiu atveju sistema nežino ar reikia įjungti papildomus energijos gaminimo rezervus ar mažinti elektros gaminimą. Išmaniojo tinklo efektyvumas priklauso nuo tikslių išmaniųjų skaitiklių duomenų tikslumo ir šių duomenų perdavimo laiku. Šiuo atveju sistema gali atlikti klaidingus pasirinkimus, tokius kaip pagaminti per daug ar per mažai elektros, apgadinant fizines sistemos komponentus ir padarant finansinę žalą vartotojams ir elektros gamintojams.

1.2.5. Atsinaujinančių elektros šaltinių nepastovumas

Atsinaujinantys elektros šaltiniai yra ekologiškas ir inovatyvus būdas generuoti elektros energiją, bet šių šaltinių generuojama elektra priklauso ne nuo žmogaus, o nuo gamtos jėgų. Dėl šios priklausomybės elektros generavimas gali svyruoti tarp visiškai neegzistuojančio ir pilnų pajėgumų tą pačią dieną ar net valandą. Ši situacija, kai nutrūksta elektros tiekimas, nutinka ir su vėjo jėgainėmis ir su saulės baterijomis, kai nebėra vėjo ar saulė yra užstojama debesų. Nepaisant to yra pastebėta, kad yra tam tikra priklausomybė tarp vėjo ir saulės elektros gaminimo, paprastai vienam elektros generavimo būdui negalint to daryti, kitas tiekia elektrą ir atvirkščiai. Neužtikrinant atsarginių elektros tiekimo šaltinių, elektros tinklas gali patirti pasiūlos trūkumą ir prastos kokybės elektrą ar net elektros tiekimo nutrūkimą.

1.3. Išskirstytų sistemų verifikavimo ir vertinimo metodai

Šiame skyriuje bus aptariamas modelių patikrinimas, modelių verifikavimo proceso žingsniai, modelių verifikavimo pranašumai prieš teoremų įrodymo verifikavimą, modelių patikrinimo privalumai ir trūkumai, modelių tikrinamos savybės išreikštos laiko logika, modelių patikrinimas ir laiko automatai, bei statistinis modelių tikrinimas.

1.3.1. Modelių patikrinimas

Modelių verifikavimo metodas – verifikacijos metodas, kuris naudoja algoritmus patikrinti kiekvieną įmanomą sistemos būseną [13]. Modelių verifikavimas veikia panašiai kaip kompiuterinė šachmatų programa, kuri sistemiškai patikrina visus įmanomus scenarijus. Naudojantis šiuo metodu, tikslingai galima patikrinti ar sistemos modelis atitinka tam tikrą reikalavimą. Tačiau yra palyginus sunku ištestuoti didelį kiekį būsenų scenarijų, neperkraunat naudojamų procesorių ir neperpildant turimos atminties. Naujausi modelių tikrinimo įrankiai gali ištestuoti ir išsaugoti nuo 10^8 iki 10^9 būsenų. Naudojantis išmaniausias algoritmais ir specializuotomis duomenų struktūromis galima pasiekti nuo 10^{20} ar net 10^{476} būsenų, kai bandoma ištestuoti specifinę problemą. Net subtilios klaidos, kurių galima nerasti naudojantis emuliacija, testavimu ir simuliacija, galimai gali būti atrastos naudojantis modelių tikrinimo ir analizės metu.

Modelių verifikavimo metodai yra paremti modelių gebėjimu apibūdinti galimą sistemos elgseną matematiškai tiksliu ir nedviprasmišku būdu. Neretai prieš pradedant bet kokią kitokio tipo verifikaciją, tikslus sistemos modeliavimas padeda atrasti neišbaigtumo, netikslumo ir dviprasmiškumo problemų reikalavimų specifikacijose. Tokios problemos paprastai atrandamos daug vėlesnėse dizaino stadijose. Vis dėlto reikia turėti omenyje, kad modelio tikrinimo metodo kokybė priklauso nuo modelio kokybės, todėl rezultatų kokybė irgi priklauso nuo modelio kokybės, o modelio kokybė priklauso nuo jo kūrėjo ar kūrėjų suvokimo ir patirties.

Modelio tikrinimo metodas dažniausiai tikrina savybės kokybinį aspektą, pvz.:

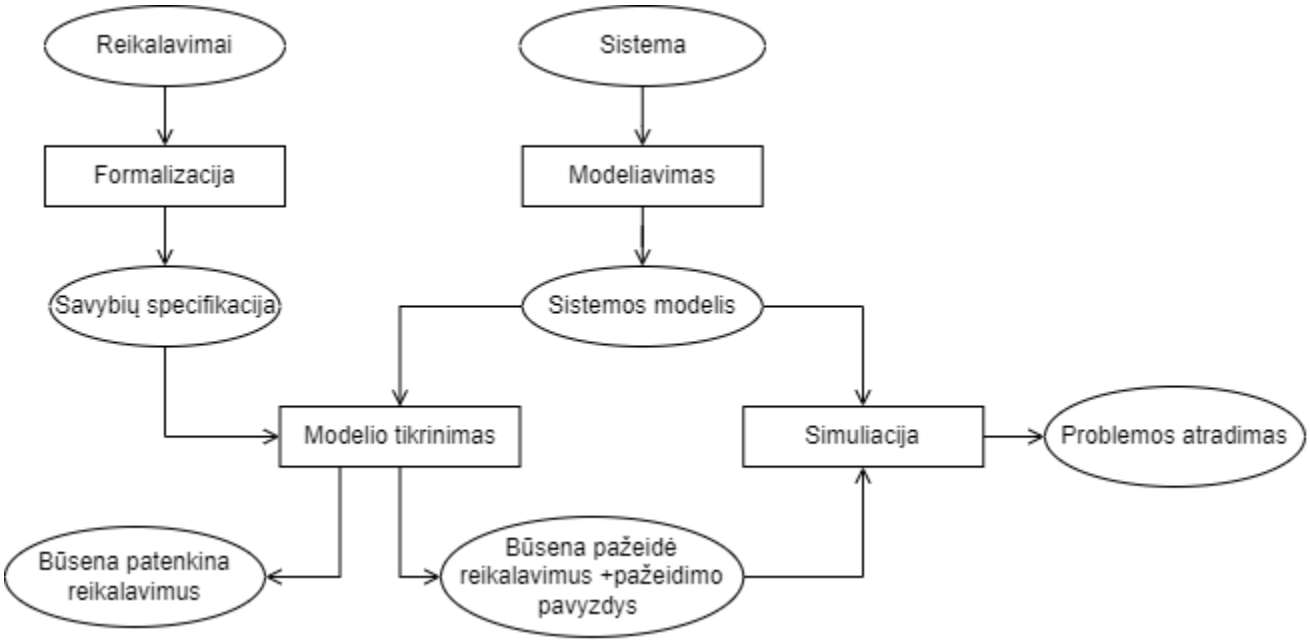
- Ar sugeneruotas teisingas atsakymas?
- Ar sistema gali pasiekti aklavietę, pvz.: kai dvi lygiagrečios programos laukia viena kitos ir stabdo sistemos darbą?

Taip pat modelio tikrinimo metodas gali padėti stebėti laiko savybes, pvz.:

- Ar sistemos aklavietė gali būti pasiekta per vieną valandą nuo sistemos pradžios?
- Ar išsiųstas atsakymas yra gaunamas per 8 minutes?

Modelių tikrinimas reikalauja tikslių ir nedviprasmiškų sistemos savybių apibūdinimo, norint jas kokybiškai ištestuoti. Kuriant tikslų sistemos modelį, lengvai pastebima dviprasmiškumai ir nenuoseklumai reikalavimų dokumentacijoje. Atradus problemines situacijas naudojantis šiuo metodu, galima padaryti pakeitimus reikalavimuose ir išvengti klaidų ir gedimų kuriant naują sistemą.

Sistemos modelis automatiškai sugeneruojamas iš modelio aprašymo, kuris paprastai būna parašytas įrankio modeliavimo kalba. Reikia turėti omenyje, kad savybių specifikacija aprašo ką sistema turi daryti ir ko ji neturėtų daryti, o modelio aprašymas specifikuoja, kaip sistema veikia. Modelio tikrintojas patikrina visas aktualias sistemos būsenas, norint patikrinti visos konfigūracijos atitinka tam tikrą savybę. Jei atrandama būsena, kuri pažeidžia savybės reikalavimus, modelio tikrintojas pateikia pavyzdį, kaip modelis pasiekia netinkamą būseną. Simuliacijos dėka, vartotojas gali pakartoti netinkamą scenarijų, taip gaunant reikiamos informacijos ir pamodifikuoti sistemos modelį reikiama linkme (**1 Diagrama.** Modelių tikrinimo ir klaidų ieškojimo procesas).



1 Diagrama. Modelių tikrinimo ir klaidų ieškojimo procesas [13]

1.3.2. Modelių verifikavimo proceso žingsniai

Modelių tikrinimo procesas [13] turi būti gerai struktūrizuotas, organizuotas ir suplanuotas. Toliau šiame skyriuje giliau išvardinsime ir išanalizuosime modelių verifikavimo proceso žingsnius. Šis procesas susideda iš šių trijų žingsnių:

1. Modeliavimas:

- Šioje fazėje kuriamas sistemos modelis, naudojantis modelių tikrintojo aplinkos kalba.
- Šiame žingsnyje paleidžiamos kelios simuliacijos, siekiant patikrinti modelio kokybę (ar modelis neturi klaidų, kurios iškreipia simuliacijos rezultatus)

2. Programos paleidimas – paleidžiamas modelio tikrintojas, kuris tikrina scenarijus ir modelio dalių veikimą.

3. Analizė – šioje dalyje analizuojamos problemos iškilusios per programos paleidimo žingsnį. Stebima, ar sąlyga įvykdyta, jei:

- Sąlyga įvykdyta – einame prie kitos sąlygos.
- Sąlyga neįvykdyta:
    1. Generuojami panašūs scenarijai ir analizuojama situacija;
    2. Koreguojamas modelis ar tikrinama sąlyga;
    3. Pakartojamas visas analizės procesas.
- Trūksta atminties? – sumažinama tikrinama modelio dalis ir bandoma iš naujo.

1.3.3. Modelių verifikavimo pranašumai prieš teoremų įrodymo verifikavimą

Teoremų įrodymas naudojamas matematikoje įrodyti programos savybes, naudojantis griežtais loginiais dėsniais. Pagrindinis skirtumas tarp teoremų įrodymo ir modelio tikrinimo programų verifikacijoje yra tai, kad teoremų įrodyme nereikia tikrinti kiekvienos įmanomos programos sistemos būsenos, kad verifikuoti jos savybes. Tuo pačiu teoremų įrodinėjimas gali aprėpti begalybę sistemos būsenų su sudėtingais duomenų tipais ir rekursija. Teoremų įrodymas tai gali pasiekti, nes jis dirba su matematiniais būsenų apribojimais, ne būsenų simuliacija. Teoremų įrodymo metodas yra labai tinkamas tikrinti sistemas, kurios turi kompleksines duomenų struktūras ir palyginus paprastą informacijos srautą. Nors teoremų įrodinėjimo metodas pilnai gali būti automatizuotas tik apribotais atvejais, brandi teoremų įrodinėjimo sistema gali pasiekti atitinkamą automatizacijos lygį.

Samprotavimas apie induktyvias struktūras (logikos medžius, sąrašus ir kt.) yra įmanomas naudojantis matematine indukcija, bet dažnai negali būti pilnai automatizuotas. Nepaisant to, kai kuriomis situacijomis tai yra pateisinama, nes tokio tipo analizė negali būti atlikta naudojantis modelių tikrinimu, bet vis tiek yra svarbi verifikavimo procese. Kai kuriais atvejais automatizacijos trūkumas gali būti toleruojamas dėl teoremų įrodymo metodo platesnių verifikavimo galimybių.

Nors teoremų įrodymo metodas turi savų privalumų lyginant su modelių tikrinimu (labiausiai dėl sistemų dydžio ir kiekio, kurioms šis metodas gali būti bent jau teoriškai pritaikytas, ir induktyvaus samprotavimo galimybių), jis taip turi savo trūkumų. Vienas iš didžiausių šio metodo trūkumų, kad juos suprasti ir pritaikyti reikia didelės formalių (t.y. pagrįstų matematika ir logika) metodų patirties ir papildomo formalizavimo bei verifikavimo laiko, todėl dėl šio aspekto teoremų įrodymo metodas sunkiai prigijo rinkoje, tuo tarpu modelių tikrinimo metodas yra daug populiaresnis [26].

1.3.4. Modelių patikrinimo privalumai ir trūkumai

Norint įvertinti modelių patikrinimo metodo praktinę vertę, reiktų įvertinti šio metodo stipriąsias ir silpnąsias vietas. Šioje knygoje [12] aptariami modelių patikrinimo privalumai ir trūkumai.

Modelių tikrinimo metodo privalumai:

- Šis verifikacijos metodas tinka testuoti įvairių sistemų dizainą, tame tarpe tokių kaip: įterptinės sistemos, programinės sistemos ar net kompiuterinės įrangos sistemos.
- Metodas gali verifikuoti modelio dalį individualiai, tai reiškia, kad galima susikoncentruoti į tas modelio dalis, kurios yra pačios svarbiausios ir gauti rezultatus neperkraunat kompiuterio atminties. Tai yra labai naudinga tuo atveju, kai trūkstant laiko ir kompiuterinių resursų. Ši savybė taip nereikalauja pilnos reikalavimų specifikacijos.
- Metodui negresia rizika, kad atradus klaidą, ji trukdys testavimui ar simuliacijai, ar kitaip trukdys verifikavimo darbą.
- Metodas pateikia diagnostinę informaciją, tuo atveju, jei kintamųjų užklausos yra nepabaigtos ar klaidingai aprašytos.

- Šis metodas yra labai paprastas ir nereikalauja iš tikrintojo nei aukštojo laipsnio išsilavinimo, nei gilios patirties.
- Metodas yra labai patrauklus industrijai: kelios kompiuterių įrangos kompanijos turi nuosavas modelių tikrinimo laboratorijas.
- Metodas labai lengvai integruojamas į jau egzistuojančius įrangos kūrimo ciklus, verifikacijos metodas lengvai išmokstamas ir empiriniai tyrimai rodo, kad tai priveda prie trumpesnio įrangos kūrimo proceso.
- Šis metodas turi tvirtą matematinį pagrindą: jis yra paremtas grafų teorija, duomenų struktūromis ir logika.

Kaip matome šis metodas tinka tikrinti plačiai programinių sistemų įvairovei. Šiame darbe mes orientuosimės į kiberfizines sistemas, kurioms šis metodas labai tinka. Taip pat šis metodas gali tikrinti ir sistemos dalį ir tai mums leidžia taupyti kompiuterio ir laiko resursus. Kitas modelių tikrinimo metodo privalumas yra, kad juo galima naudotis, kai nėra prieinama pilna reikalavimų specifikacija ir naudojantis gautais rezultatais ją papildyti ir pataisyti. Šis metodas taip pat teikia diagnostinę informaciją, nestabdo verifikacijos radus klaidą ir labai lengvai pritaikomas.

Metodo tikrinimo trūkumai:

- Šis metodas labiausiai tinkamas sistemoms, kurios orientuojasi į valdymo kontrolę ir mažiau tinka sistemoms, kurios orientuojasi į didelius duomenų kiekius.
- Šis metodas paprastai turi sistemos "būsenų skaičiaus sprogimo" problemą, kai susiduriama su sudėtingomis sistemomis, turinčių didžiulį būsenų skaičių, arba sistemos procesų modeliavime naudojama abstraktūs duomenų tipai ir duomenų struktūros, kurių efektyvus apdorojimas yra neįmanomas.
- Šis metodas verifikuoja sistemos modelį, bet ne pačią sistemą (produktą ar prototipą), tai reiškia, kad rezultatų kokybė priklauso nuo modelio kokybės. Papildomi testavimai reikalingi, kad rasti gamybos gedimus (aparatinei įrangai) ar kodavimo klaidas (programinei įrangai).
- Modelis patikrina tik nustatytus reikalavimus, tai yra negarantuoja, kad modelis bus pilnai ištestuotas. Tai reiškia, kad tos modelio dalys, kurios nebus patikrintos, gali turėti nepastebėtų trūkumų ir trukdyti sistemos pilnam verifikavimui.

- Šio metodo naudojimas reikalauja dalinės patirties bandant surasti atitinkamas abstrakcijas, kad gauti mažesnius sistemos modelius ir pateikti modelio dalis formalizuotoje logikoje, kuri bus naudojama.
- Šis modelio tikrinimo metodas negarantuoja teisingų rezultatų, kaip ir kiekvienas kitas įrankis; modelių tikrintojas gali turėti programinės įrangos klaidų.

1.3.5. Modelių tikrinamos savybės išreikštos laiko logika

Laiko logika yra formali teorija, leidžianti nurodyti ir verifikuoti reaktyvių sistemų savybes. Laiko logikos formulė aprašo būsenų sekų rinkinį, tenkinantį formalizuotas laiko savybes. Naudojama sistema patenkina savybę, jei visos galimos veiksmų sekos yra šiame rinkinyje.

Laiko logikos modelis yra neribota būsenų seka, kurioje kiekviena būsena skirtingame laiko taške gali turėti unikalias vertes. Laiko formulės yra įvertinamos naudojantis tokiomis būsenų sekomis, kartu su šių būsenų indeksų (*angl. index = 0,1,2,….*) i-tąją būsena [27].

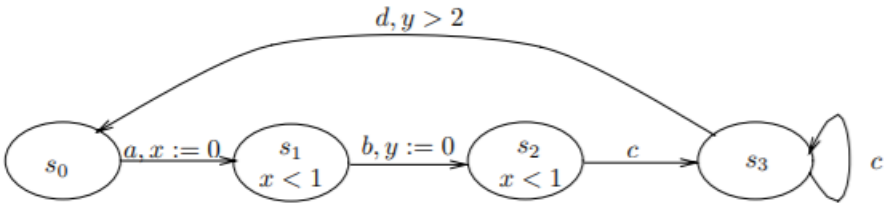
Modelių tikrinimas remiantis laiko logika atskleidžia modelių trūkumus. Šie trūkumai randami naudojantis gyvybingumo (angl., liveness), pasiekiamumo (angl., reachability) ir saugumo (angl., safety) sistemos savybėmis[28]. Neturint laiko logikos savo modelyje sunku pastebėti problemas, susijusias su užduočių atlikimu laiku. Nenaudojant laiko logikos, galima nepastebėti, kad modeliuose grafų viršūnės niekada nenaudojamos ir yra neaktyvios, taip pažeidžiant gyvybingumo reikalavimą, arba nepasiekia viena kitos ir netenkina pasiekiamumo reikalavimų, taip pat tuo atveju, jei modelyje yra galimybė sugadinti sistemos veikimą uždelsiant duomenų perdavimą taip, kad grafų viršūnės niekada nepakeičia savo būsenos, pažeidžiami saugumo reikalavimai. Viso to galima išvengti įtraukus laiko formules į savo modelį ir naudojant laiko logiką.

1.3.6. Modelių patikrinimas ir laiko automatai

Tradicinis modelių tikrinimo metodas, nepadengia realaus laiko modeliavimo[29] (neturi galimybės atvaizduoti savybių, kurios priklauso nuo laiko), ir dėl to netinka realaus laiko sistemų analizei, kurių tikslumas priklauso nuo užduočių vėlavimo ir trikdžių. Dėl šios problemos buvo sukurti laiko automatai, skirti modeliuoti realaus laiko sistemas. Laiko automatai gali lengvai

simuliuoti grafus su būsenų kaitomis ir laiko suvaržymais, naudojant daugybę realaus laiko laikmačių modelyje.

Išreikšti sistemų būsenų veikimą laiko sekos ribose, yra naudojami grafai su realaus laiko laikmačiais (*angl. real-valued clocks*). Grafo viršūnės vadinamos vietomis (*angl. locations*), o viršūnes jungiančios linijos – jungtys (*angl. switches*). Nors jungtys yra momentinės (*angl. instantaneous*), laiko reikšmė gali keistis vietose (grafo viršūnėse). Bet kokiu momentu, laikmačio reikšmė priklauso laiko sekai, nuo 0 iki užfiksuoto laiko skaičiavimo metu. Su kiekviena jungtimi, mes asocijuojame laikmačio reikšmę laiko sekoje, vadiname ją laiko invariantine savybe ir reikalaujame, kad laikas eitų vietoje tik tol, kol invariantas atitinka nurodytą nelygybę. Apžvelkime laiko automatų pavyzdį:



2 Pav. laiko automatas su dviem laikmačiais **[28]**

Laikmatis x nustatomas į 0, kiekvieną kartą, kai sistema keičiasi tarp s_0 ir s_1 a jungtimi. Invariantas (x<1) asocijuojasi su vietomis s_1 ir s_2, tai užtikrina, kad jungtis pažymėta c, kuri yra tarp s_2 ir s_3, nutinka per 1 laiko tarpą. Perkraunant dar vieną laikmatį y, kartu su b jungtimis iš s_1 į s_2 ir tikrinant vertę ant d jungties iš s_3 į s_0, užtikrina, kad užtrukimas tarp jungties b ir d yra visada didesnis nei 2. Tai yra svarbus privalumas, kai laiko automatas turi daugybę laikmačių, kurie gali būti nustatyti atskirai vienas nuo kito.

1.3.7. Statistinis modelių tikrinimas

Statistinis modelių tikrinimo metodas tikrina dizaino tikslumą, paleidžiant modelių simuliacijas ir kaupiant statistinius rezultatus, kurie kaupiami modelio tikrinimo metu, atsižvelgiant į tam tikros savybės specifikaciją. Šiuo metodu galima tikrinti sistemas, kurių specifikacijos yra per daug

sudėtingos, tikrinti paprastu modelių tikrinimo metodu. Todėl šis metodas plačiai naudojamas tokiose mokslinio tyrinėjimo srityje, kaip biologija ar programų kūrimas. Yra kelios šios sėkmės istorijos priežastys. Pirma būtų, kad šiuo metodu yra labai lengva naudotis ir suprasti (ypač lengva programų industrijoje dirbantiems programų sistemų inžinieriams ir tiems žmonėms, kuri nebūtinai užsiima moksline veikla, bet nori verifikuoti jiems svarbias sistemas). Antra priežastis yra, kad metodas reikalauja tik nedidelio (arba visiškai nereikalauja) papildomo modeliavimo ar specifikacijos papildymo. Verifikavimo procesui užtenka veikiančio sistemos modelio, kuris gali būti simuliuojamas ir tikrinamas atsižvelgiant į sistemos savybes ir reikalavimus. Trečia priežastis būtų, kad naudojantis statistiniu rezultatų generavimu, leidžiama numatyti nepastebėtų problemų galimybę, neperkraunant kompiuterio atminties [30].

1.3.7.1. Statistinio modelių tikrinimo metodo veikimas

Statistinio modelio tikrinimo metodas naudojasi statistika ir simuliuoja sistemą tol, kol sugeneruoja pakankamai pavyzdžių, kad pateiktų statistinius įvertinimus, kurie patvirtina specifikacijoje esantį reikalavimą arba jį paneigia. Statistiniai duomenys taip pat naudojami, klaidos atveju, norint nustatyti šios klaidos tikimybę. Pagrindinis skirtumas, lyginant su klasikiniu modelių tikrinimu: vietoje visų būsenų patikrinimo, sistemos modelis "vykdomas" (simuliuojamas) su laiko ar bandymų apribojimu, statistiškai apibendrinant gautus tikrinimo rezultatus.

1.3.7.2. Statistinio modelių tikrinimo metodo privalumai ir trūkumai

Šis metodas turi daugybę privalumų. Pirma, šis algoritmas iš sistemos reikalauja tik to, kad ją būtų galima simuliuoti (leisti iš sistemos gauti pavyzdinius statistinius duomenis, naudojantis nurodyta kompiuterio ir sistemos atmintimi). Tai reiškia, kad šis metodas gali būti pritaikytas didelį kiekį objektinių klasių turinčioms sistemoms, tokioms, kaip skaitinės kvantinės kompiuterinės sistemos, juodos dėžės sistemos ir begalinės būsenos sistemos. Antra, šis metodas gali leisti klasėms turėti didesnį skaičių objektų. Ir galiausiai šis algoritmas labai lengvai veikia lygiagrečiai, dėl to galima simuliuoti daug didesnes sistemas. Tais atvejais, kai modelio tikrinimo problema yra neapibrėžta ir per daug komplikuota, šis metodas yra vienintelis pasirinkimas. Šis algoritmas yra pagrindinis simuliavimo metodas naudojamas tokiuose įrankiuose, kaip Ymer, Prism ir Uppaal [14].

Vienas iš pagrindinių šio metodo trūkumų yra tokios dizaino klaidos, kurios turi labai žemą pasireiškimo šansą. Kadangi šios klaidos turi mažą pasirodymo šansą, jos gali nepatekti į generuojamą statistiką. Šios klaidos gali būti kritinės sistemos veikimui ir vėliau sukelti problemų klientams ir programos kūrėjams.

1.3.7.3. Statistinio modelių tikrinimo metodo įrankiai

Yra identifikuoti trys pagrindinei statistinio modelių tikrinimo metodo įrankiai: Uppaal, Prism ir Ymer. Palyginus šių įrankių trūkumus ir privalumus, tikimasi atrinkti šiam darbui patogiausią įrankį.

Prism įrankio pagrindiniai pranašumai yra tai, kad siekiant įrodyti savybę, įrankis gali sugeneruoti daugiau nei vieną savybės pasiekiamumo būsenų konfigūracijų aibę. Vienas iš didžiausių šio įrankio trūkumų yra šio įrankio sudėtingumas, nes norint juo naudotis reikia gilių tikimybių teorijos, Markovo grandinių ar kt. žinių, vien tam, kad kurti ir testuoti modelius [16].

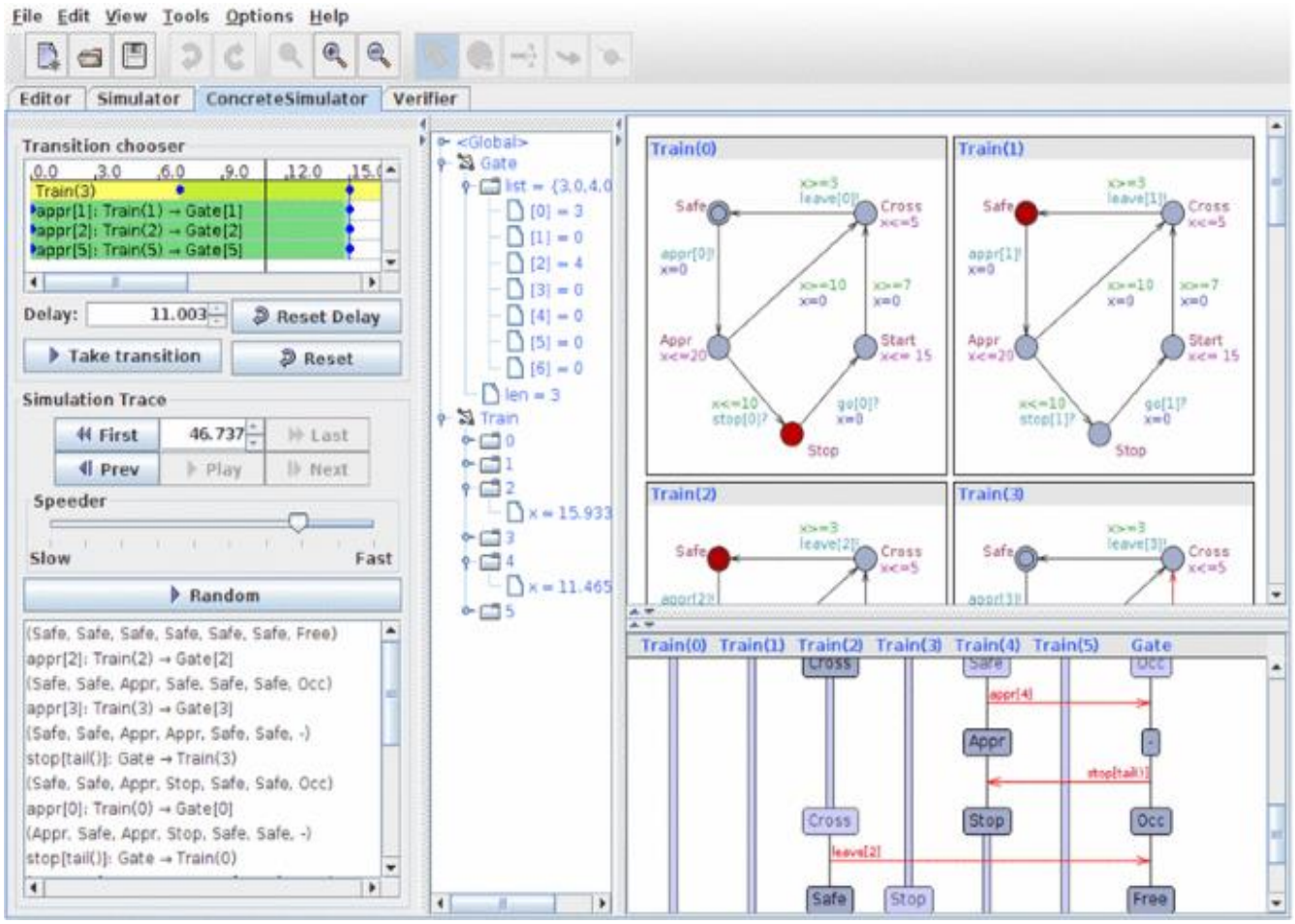
Ymer tikrinimo įrankiui trūksta prieinamos dokumentacijos, padedančios kurti ir tikrinti statistinius modelius. Taip pat Ymer trūksta grafinės sąsajos su vartotojo ir tai yra vienas iš lėčiausių statistinių modelių tikrinimo įrankių.

Uppaal yra statistinių modelių tikrinimo rinkos lyderis. Jis yra efektyvus, lengvai suprantamas ir juo lengva naudotis. Uppaal sulaukia atnaujinamų kas 6 mėnesius, turi tam tikrą specifikaciją ir grafinę sąsaja, kuri palengvina darbą. Dėl šių priežasčių pasirinktas šis įrankis.

1.4. Uppaal modelio tikrintojas

Šiame skyriuje aptarsime Uppaal modelių tikrintojo įrankį, aptarsime jo pagrindines dalis ir jo naudojamą modeliavimo kalbą.

Pagal oficialią Uppaal žiniatinklio svetainę [15] Uppaal yra integruotos aplinkos įrankis skirtas realaus laiko sistemų modeliavimui, validacijai ir verifikacijai. Šis įrankis buvo sukurtas, kaip kolaboravimo projektas tarp Aalborg universiteto pagrindinių kompiuterijos mokslo tyrimų skyriaus ir Informacinių technologijų departamento Upsala universiteto Švedijoje. Uppaal tinka modeliuoti sistemas, kurios gali būti sumodeliuotos, kaip nedeterminuotų procesų rinkinys su griežtu kontrolės centru ir realios reikšmės laikmačiais, komunikuojančiais per bendrus duomenų kanalus. Tipinės pritaikymo sritys apima realaus laiko valdiklis ir komunikacijos protokolus, kuriems laiko aspektai yra kritiški.



3 Pav. Uppaal grafinė sąsaja **[43]**

1.4.1. Uppaal dalys

Uppaal įrankis sudarytas iš trijų pagrindinių komponentų:

- Modeliavimo kalba – imperatyvi kalba su duomenų tipais. Ši kalba yra naudojama, kaip modeliavimo ir dizaino kalba skirta apibūdinti sistemą, kaip laiko automato tinklus, išplėstus su laikmačiais ir duomenų kintamaisiais.
- Simuliatorius – validacijos įrankis, kuris leidžia analizuoti galimą dinaminį sistemos veikimą, per ankstyvojo dizaino (ar modeliavimo) etapus.
- Modelio tikrintojas – įrankis, kuris gali patikrinti laiko ir pasiekiamumo (*angl. reachability)* savybes, tikrinant galimas sistemos būsenas, tai yra atliekant pasiekiamumo ir statistinę analizę.

Uppaal programos kūrimo metu buvo iškelti du pagrindiniai kriterijai: efektyvumas ir naudojimo paprastumas. Siekiant palengvinti modeliavimą ir klaidų ieškojimą, Uppaal modelių tikrintojas automatiškai generuoja diagnostinius duomenis, kurie paaiškina kodėl sistemos būsena tenkina arba netenkina sistemos reikalavimus. Modelių tikrintojo sugeneruoti diagnostiniai duomenys (angl., *error trace or counterexample*), gali būti automatiškai perkeliami į simuliatoriaus aplinką, kurioje atkuriamos probleminės situacijos, leidžiant atrasti kritines sistemos klaidas, vizualiai pakartojant scenarijus, kuriuose sistemos susidūrė su probleminėmis situacijomis. Naudojantis sukauptais duomenis, galima pamodifikuoti modelį ir ištaisyti dizaino klaidas, užkertant kelią šių klaidų pasikartojimą realiame gyvenime.

Uppaal privalumai:

- Grafinės sistemos redaktorius, leidžiantis kurti grafinį sistemos modelį.
- Grafinis simuliatorius, kuris gali grafiškai atvaizduoti sistemą ir kurti įrašus, kurie parodo modeliuojamos sistemos dinamišką veikimą, tai yra sistemos būsenų seką.
- Reikalavimų specifikacijos redaktorius, kuris veikia taip pat, kaip grafinė naudotojo sąsaja.
- Modelio tikrintojas skirtas automatinei sistemos savybių verifikacijai.

- Diagnostinių duomenų generacija, kuri yra naudojama tuo atveju, jei tam tikra realaus laiko sistema sutrinka. Tai yra skirta tam, kad duomenys būtų užkrauti ir grafiškai atvaizduoti, naudojantis simuliatoriumi [30].

1.4.2. Uppaal kalba

Modelių tikrinimo įrankis Uppaal yra sukurtas remiantis laiko automatų teorija ir jo modeliavimo kalba suteikia įrankiui papildomų funkcijų. Uppaal užklausų kalba yra TCTL (*angl. timed computation tree logic*) kalba, naudojama nurodyti, kokios sistemos savybės bus tikrinamos modelių tikrinimo procese. Šiame skyriuje pristatysime Uppaal modeliavimo ir užklausos kalbas [31].

1.4.2.1. Uppaal modeliavimo kalba

Laiko automatas yra baigtinių būsenų mašina (*angl. finite-state machine*) papildyta laikmačių kintamaisiais. Jis naudoja tankaus laiko modelį (*angl. dense-time model*), kuriame laikmačių kintamieji turi realiųjų skaičių vertes. Visų laikmačių reikšmės progresuoja sinchroniškai. Uppaal modeliavimo įrankyje sistema modeliuojama, kaip lygiagrečiai veikiantis kelių laiko automatų tinklas. Modelis taip pat yra papildytas diskrečiais kintamaisiais, kurie yra būsenos dalis. Šie kintamieji naudojami programavimo kalbose kuriant sistemą ir jie yra nuskaitomi, aprašomi ir dalyvauja paprastose aritmetinėse operacijose. Sistemos būsena yra apibūdinama laiko automatų išsidėstymu, laikmačių ir kintamųjų vertėmis.

Uppaal modeliavimo kalba praplečia laiko automatus šiais funkcionalumais:

- Vietomis (*angl. locations*) – vietos laiko automatuose grafiškai atvaizduojamos kaip apskritimai. Analizuojant laiko automatus kaip grafus, vietos yra šių grafų viršūnės. Viršūnės yra sujungtos jungtimis.
- Jungtimis (*angl. transitions*) – jungtys laiko automatų grafikuose jungia vietas. Šios jungtys gali būti:
    - Jungtys su sąlygomis (*angl. transitions with conditions*) – sąlygos apriboja jungtyje esantį kintamąjį tam tikros sekos ribose.
    - Jungtys su sargais (*angl. transitions with guards*) – jungtis veikia sistemos būsenoje, tik tuo atveju, jei nurodyta sargo sąlyga yra tenkinama toje būsenoje.

- Jungtys su būsenų atnaujinimu (*angl. transitions with state updates*) – jungtis, kurią aktyvuojant, atnaujinama sistemos būsena.

- Sinchronizuoti kanalai tarp komponentų/procesų (*angl. synchronization channels between components/processes)* – kanalai yra naudojami sistemų sinchronizavime. Tai atliekama pažymint jungtis sinchronizacijos žymėmis. Kanalas pažymėtas *e?* (kanalo_pavadinimas?) sinchronizuosis su kanalu e!(kanalo_pavadinimas!) [31]. Dvi jungtys skirtinguose procesuose, gali sinchronizuotis, jei abiejose procesuose esančios reikšmės atitinka sargų sąlygas ir *e?* jungtis sujungta su e!, sudarant kanalą [14].

- Būsenų kintamieji (*angl. state variables*) – būsenų kintamieji, kurie egzistuoja sekų ribose. Atliekant verifikaciją, jei sąlygos reikšmės nebeegzistuoja reikšmių sekos ribose, toliau esančios būsenos nebetikrinamos.

- Būsenų konstantos (*angl. state constants*) – konstantos negali būti modifikuojamos ir turi turėti sveikojo skaičiaus reikšmę. Deklaruojamos kaip const tipas. [31]

1.4.2.2. Uppaal užklausų kalba

Pagrindinis modelių tikrintojo tikslas yra verifikuoti modelį naudojantis reikalavimų specifikacija. Lygiai taip pat, kaip modelis, reikalavimų specifikacija turi būti formaliai aprašyta kompiuteriui suprantama kalba. Uppaal naudojasi supaprastinta TCTL (*angl. timed computation tree logic*) kalbos versija, kuri sudaryta iš kelio (*angl. path formulae*) ir būsenos formulių (*angl. state formulae*). Būsenų formulės apibūdinama individualias sistemos būsenas, kai tuo tarpu kelio formulės apibūdina sistemos modelio kelius. Kelio formulės apibūdina sistemos pasiekiamumo, saugumo ir gyvybingumo reikalavimus.

Uppaal užklausų kalba sudaryta iš:

- **Būsenų formulių** – išraiška, kuria naudojantis galima nustatyti sistemos būseną, nesinaudojant sistemos modeliu. Pavyzdžiui, tai gali būti labai paprasta išraiška, kaip i==7, kuri yra tenkinama būsenoje, kurioje i yra lygi 7. Uppaal aklavietė yra išreiškiama specialia būsenos formule. Būsenos formos pavyzdys: A[] not deadlock (sistemoje nėra aklaviečių).

- **Kelio formulių**, kurios yra išskirstomos į:

  - **Pasiekiamumo savybes** – pačios paprasčiausios savybės. Jomis siekiama išsiaiškinti ar duotos formulės būsena ϕ gali būti patenkinta, naudojantis bet kurioje iš pasiekiamų sistemos būsenų.

Pateikiant šią problemą klausimo forma, tai būtų: ar sistemoje egzistuoja kelias, pradedant nuo pradinės būsenos, kuriuo einant būsena ϕ bus patenkinta. Pavyzdžiui, kuriant modelį su siuntėju ir gavėju, yra logiška patikrinti ar siuntėjas iš viso išsiųs žinutę ir ar gavėjas ją gaus. Šios savybės negarantuoja protokolų korektiškumo (pvz.: kad žinutės bus priimta), bet jie patvirtina sistemos modelių pagrindinį veikimą. Uppaale ši savybė aprašoma kaip E<> ϕ.

- **Saugumo savybes** – savybės, kurios užtikrina, kad tam tikra nepageidaujam situacija niekada neįvyks. Pavyzdžiui sistema užtikrina, kad radioaktyvaus reaktoriaus modelyje, saugumo savybė atsako už reaktoriaus temperatūrą, tad ji niekada nepasieks rizikingo lygio, ir niekada nebus reaktoriaus perkaitymo ir sprogimo. Šios savybės variacija yra, kad „kažkas galimai niekada neįvyks“. Pavyzdžiui žaidžiant žaidimą yra išsaugota saugi būsena (*angl. save*), nuo kurios mes visada galime laimėti žaidimą, todėl yra galimybė nepralaimėti žaidimo. Uppaale savybės yra formuluojamos pozityviai, pvz.: kažkas gerai visada bus tiesa. Šiuo atveju ϕ yra būsenos formulė. Reikšmė ϕ turi būti teisinga, visose pasiekiamose būsenose su kelio formule A[] ϕ, kai tuo tarpu E[] ϕ reiškia, kad egzistuoja kelias, kuriame visada ϕ yra teisinga. Uppaale rašoma A[] ϕ ir E[] ϕ atitinkamai.
- **Gyvybingumo savybes** – savybės, kurios užtikrina, kad kažkas galiausiai įvyks. Pvz.: spaudžiant nuotolinio televizoriaus valdymo pultelio įjungimo mygtuką, galiausiai įsijungs televizorius. Pačia paprasčiausia forma gyvybingumas išreiškimas A◊ ϕ kelio formule, kuri reiškia, kad ϕ bus kažkada patenkinta. Uppaale ši savybė aprašoma A<> ϕ [31].

1.5. Panašūs darbai

Šioje dalyje aptarsime panašius mokslinius darbus susijusius su šia tema.

1.5.1. Išmaniųjų tinklų elektros valdymas naudojantis laiko automatais

Šis mokslinis darbas [32] yra artimiausias šio darbo temai ir įrankiui. Šio projekto metu buvo naudojama modifikuota Uppaal versija ir simuliuojamas išmanusis tinklas.

Darbo problema yra susijusiu su išmaniojo tinklo energijos paskirstymo planavimu. Didėjant priklausomybei nuo atsinaujinančių elektros šaltinių (saulės baterijos, vėjo jėgainės), pastovus ir nenutraukiamas elektros tiekimas yra sunkiai užtikrinimas. Paprastai siekiant išspręsti šią problemą,

išmanieji elektros tinklai reguliuoja tiekiamos elektros kiekį į rinką, naudoja baterijas, kontroliuoja elektros tiekimą iš neatsinaujinančių elektros gamybos šaltinių (dujos, nafta ir kt.). Šio darbo autorius siūlo pažiūrėti į problemą iš kitos pusės – leisti išmaniesiems tinklams kontroliuoti elektros naudojimą iš vartotojo pusės. Tam būtų naudojami pakraunami prietaisai, autorius šiuos prietaisus supaprastina iki elektromobilių ir pakraunamų baterijų. Šiuo darbu siūloma papildyti išmaniųjų tinklų sistemas, leidžiant joms kontroliuoti pakraunamų prietaisų krovimo ir naudojimo laiką, atsižvelgiant į elektros energijos perteklių ar trūkumą rinkoje.

Darbo tikslas – išsiaiškinti ar laiko automatai gali efektyviai ir taisyklingai modeliuoti ir simuliuoti scenarijus iš paklausos valdymo pusės. Fokusuojamasi į vartotoją ir energijos suvartojimą.

Norint atsakyti išsikeltus klausimus, autorius naudoja laiko automatus simuliuojant scenarijus iš paklausos pusės. Naudojantis Uppaal Stratego versija, planavimo algoritmas sukuria krovimosi ir elektros naudojimo tvarkaraščius kiekvienam namų ūkio prietaisui.

Skirtingai nuo šio darbo mūsų darbas fokusuojasi į išmaniojo tinklo klaidas ir sistemos reakciją į jas.

1.5.2. Sugedusių ir apgadintų išmaniųjų elektros skaitiklių aptikimas naudojant neuroninius tinklus

Šio darbo tikslas [33] yra surasti blogą informaciją tiekiančius išmaniuosius skaitiklius naudojantis neuroninių tinklų pagalba. Modelis renka skaitiklių paklausos informaciją ir tuo remiantis bando projektuoti tolimesnį elektros energijos panaudojimą. Realiai situacijai neatitinkant, tai ko tikimasi iš sistemos ir surinktų duomenų, atrandami apgadinti išmanieji skaitikliai.

Šio darbo idėja yra panaudota, kaip viena iš dalių padedančių aptikti išmaniojo elektros tinklo gedimus sistemoje.

Skirtingai nuo šio darbo, mūsų darbas fokusuojasi į išmaniojo tinklo klaidas ir sistemos reakciją į jas.

1.6. Analitinės dalies išvados

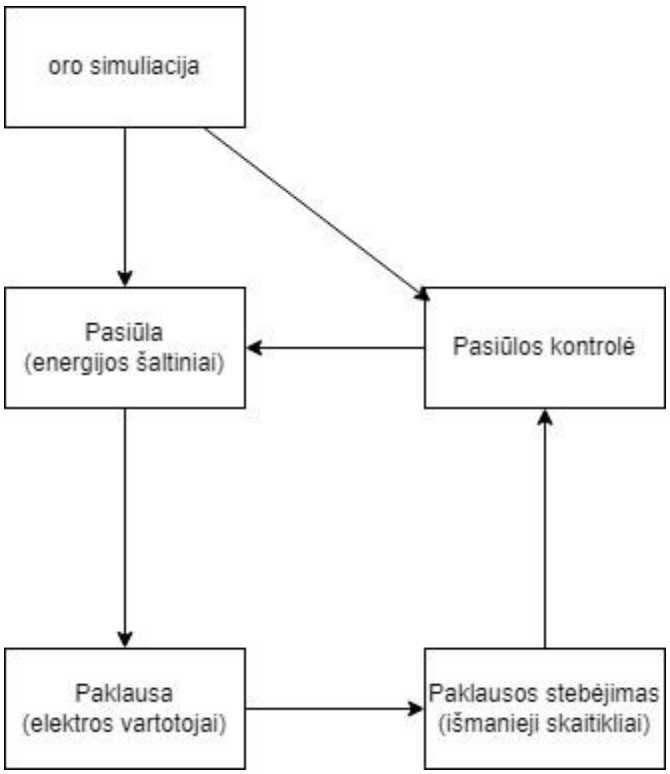
Atlikus mokslinių šaltinių analizę, buvo išanalizuotos kiberfizinės sistemos ir labiau įsigilinta į jų reikiamas savybes. Buvo apžvelgta ir labiau įsigilinta į išmaniojo elektros tinklo sistemas, jų charakteristikas ir dažniausiai pasikartojančias problemas, kurias modeliuosime šiame darbe. Toliau buvo analizuojami išskirstytų sistemų verifikavimo metodai. Buvo apžvelgtas modelių patikrinimo metodas, proceso žingsniai, pranašumai prieš teoremų įrodymo verifikavimą, modelių tikrinimo pranašumai ir trūkumai, bei tikrinamos savybės išreikštos laiko logika ir laiko automatai. Galiausiai buvo apžvelgtas statistinis modelių tikrinimas, jo privalumai ir trūkumai. Palyginti keli statistinio modelių tikrinimo metodo įrankiai. Tada pradėta analizuoti Uppaal įrankį, jo sudėtį, kalbą, statistinio modelio tikrinimo metodo pritaikymą, įrankio privalumus ir trūkumus. Paminėti į šį tyrimą panašūs ar susiję darbai.

Atlikus analitinės dalies analizę buvo giliau įsigilinta į darbo temą ir geriau suprasti, praktinėje darbo dalyje modeliuojamos sistemos požymiai, bei Uppaal įrankio funkcionalumas. Šios žinios toliau padės kitose magistrinio darbo dalyse.

2. Praktinė dalis

Praktinėje dalyje demonstruojamas išmaniojo elektros tinklo modeliavimas ir veikimas Uppaal sistemoje, išsikeltų reikalavimų testavimas ir rezultatai.

Šiame darbe modeliuojama sistema turi tam tikrą kiekį namų, kurie simuliuoja elektros energijos paklausą, o namai turi savo išmaniuosius skaitiklius, kurie gali sugesti ir būti pataisyti. Elektros pasiūlą simuliuoja elektros generatoriaus, vėjo ir saulės jėgainių modeliai. Vėjo ir saulės energijos reikšmės priklauso nuo oro simuliavimo modelio (2 Diagrama. Išmaniojo tinklo sistemos modelis).



2 Diagrama. Išmaniojo tinklo sistemos modelis

2.1. Praktinės dalies planas

Šiame darbe bus modeliuojama supaprastinta išmaniojo tinklo sistemos dalis su vartotojais ir tiekėjais. Taip pat sistemoje bus simuliuojamos keli dažniausiai pasikartojantys išmaniųjų tinklų

gedimai, aptarti ankstesniuose skyriuose (ryšio nutrūkimas tarp išmaniojo skaitiklio ir vietinio valdiklio, tyčiniai gadinimai išmaniųjų jutiklių tinkle ir kt.) ar situacijos (atsinaujinančių elektros šaltinių nepastovumas), analizuojamas jų pasikartojamas ir atsistatymo koeficientas. Šiuo darbu bus siekiama verifikuoti iškilusių problemų atsistatymo savybes ir analizuoti situacijas, kuriose sistema stengiasi patenkinti vartotojų poreikį naudojantis atsinaujinančiais elektros šaltiniais.

Išmaniojo tinklo sistema bus modeliuojama naudojantis Uppaal modelių tikrinimo įrankiu statistiniais modelio patikrinimo metodais. Sistema sudaryta iš modelių rinkinio, kurie simuliuoja pilną sistemos veikimą. Atliekant sistemos simuliaciją, modeliai sąveikauja vienas su kitu simuliuodami išmaniojo tinklo veikimą. Šie modeliai simuliuoja elektros tiekimą, vartojimą ir elektros tinklų gedimus susijusius su išmaniais skaitikliais ir elektros tiekimo pastovumu. Per šią simuliaciją suformuluotų užklausų forma yra tikrinama ar sistema atitinka tam tikras savybes.

Modeliuojama sistema sudaryta iš 12 modelių. Šie modeliai yra šabloniniai (*angl. generic*), pagal kuriuos galima sukurti reikiamą skaičių egzempliorių (sistemos komponenčių ar vykstančių procesų). Visi jie kartu modeliuoja laiką (laiko modelis), oro sąlygas (oro simuliacijos modelis), vartojimo pasiūlą ir paklausą (namų modelis, elektros tiekėjo modelis, saulės ir vėjo jėgainės), pasiūlos reguliavimą. Simuliacijoje naudojamas namo modelis, kuriuo remiantis yra sukurta 15 namo objektų su unikaliais elektros vartojimo poreikiais, vienas elektros tiekėjas, 20 saulės baterijų ir 5 vėjo jėgainės, kurios atvaizduotos kaip kintamieji, nes jų gedimai nėra modeliuojami. Modeliuojamoje sistemoje siekiama atvaizduoti realaus pasaulio sistemą ir sumodeliuoti situacijas, kurios iš šių sistemos genda, bei jų atsigavimo scenarijus ar situacijas, kuriose sistema persitvarko norint atitikti elektros paklausos poreikius.

Šiame darbe modeliuojami 5 situacijos, kuriose sistema susiduria su iššūkiais ir turi į juos reaguoti. Šios situacijos yra:

1. Ryšio nutrūkimas tarp išmaniojo skaitiklio ir vietinio valdiklio. Situacija, kai sistema praranda ryšį su namo išmaniuoju valdikliu ir negali surinkti tikslių duomenų. Tuo atveju sistema užfiksuoja gedimą ir skaičiuoja elektros suvartojimą pagal sukauptą istoriją, darant prielaidą, kad namo elektros suvartojimas, neturėtų drastiškai pasikeisti gedimo metu.

2. Tyčiniai gadinimai išmaniųjų jutiklių tinkle. Situacija, kai sistemą bandoma apgauti, pateikiant, kad namo elektros suvartojimas nukrito iki minimumo. Sistema aptinka tyčinį apgavimą ir pradeda skaičiuoti elektros suvartojimą pagal istoriją iki tol, kol sutvarkomas išmanusis skaitiklis.

3. Išsinchronizavę matavimai. Šioje situacijoje išmanusis skaitiklis nustoja rodyti tikslius duomenis ir rodo tam tikrą nukrypimą. Kitaip tariant įvyksta skaitiklio išsikalibravimas. Sistema aptinka klaidą, užregistruoja ir naudoja sukauptą elektros naudojimo istoriją apskaičiuoti galimą elektros sunaudojimą.

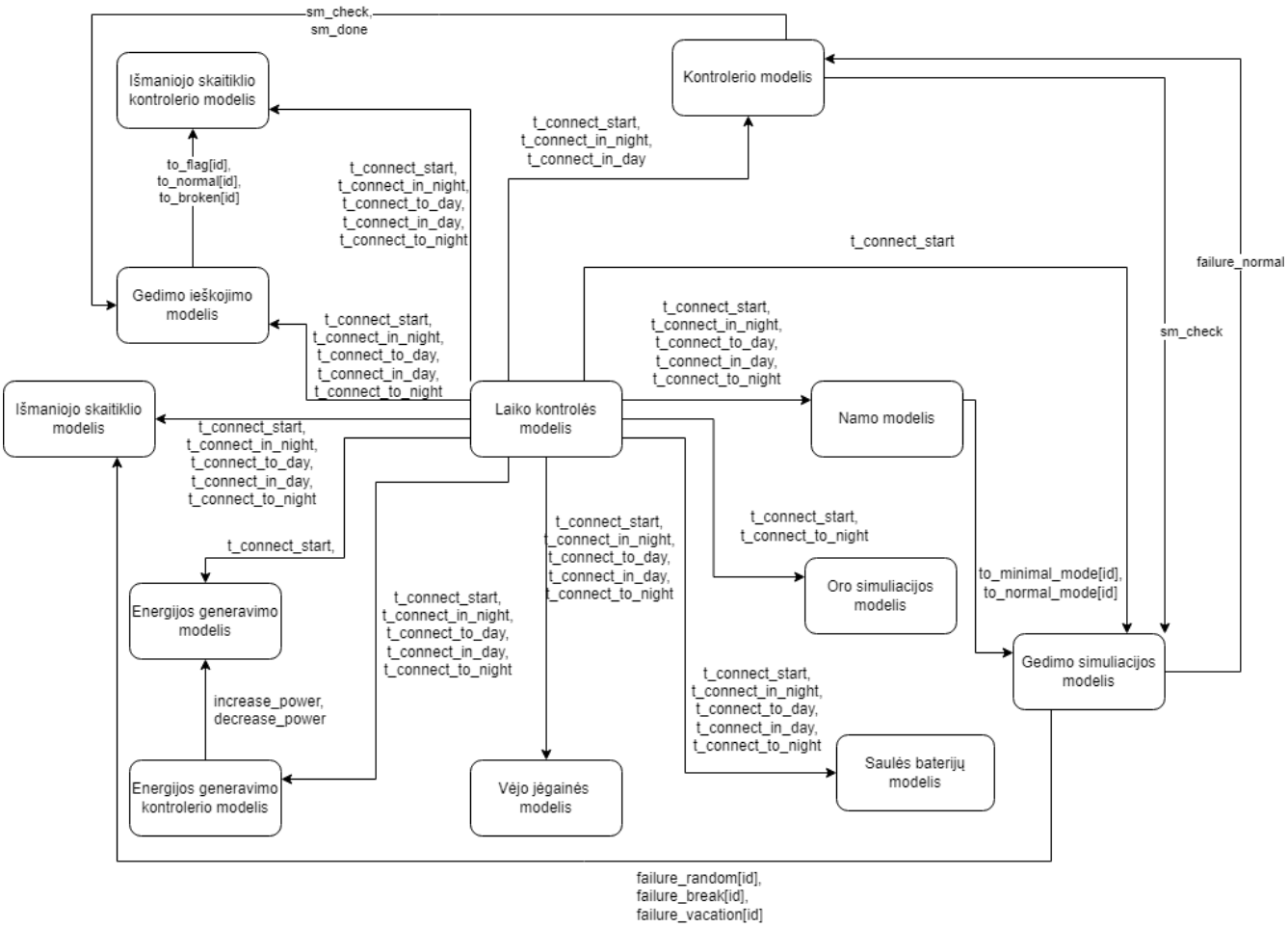
4. Sistemos negebėjimas surinkti tikslios informacijos. Dėl anksčiau išvardintų gedimų sistema gali klaidingai surinkti informaciją, todėl yra naudojamas istorijos kaupimo metodas, kuris padeda aptikti neatitikimus ir laikinai apskaičiuoja elektros sunaudojimą pagal sukauptą istoriją. Šis metodas padeda aptikti netikslumus ir juos užfiksuoti sistemoje.

5. Atsinaujinančių elektros šaltinių nepastovumas. Šioje situacijoje bandoma patenkinti visą sistemos paklausą naudojantis atsinaujinančiais elektros šaltiniais ir tikrinama ar tokia situacija yra reali.

2.2. Modeliai naudojami darbe

Šiame darbe yra modeliuojama išmaniųjų tinklų sistema, taip pat pateikiami jos sudedamųjų dalių modeliai, kurie atvaizduoja skirtingus procesus. Šie modeliai yra laiko kontrolės modelis, namo modelis, išmaniojo skaitiklio modelis, gedimo simuliacijos modelis, gedimo ieškojimo modelis, išmaniojo skaitiklio valdiklio modelis, valdiklio modelis, energijos generavimo kontrolės modelis, oro simuliacijos modelis, vėjo jėgainės modelis ir saulės baterijų modelis. Jų paskirtis bus aptariama žemiau esančiuose skyriuose, bet mes trumpai apžvelgsime jų bendrą veikimą. Sistemos struktūra ir komunikacija tarp jos komponenčių yra atvaizduota paveiksliuke (3 Diagrama. Išmaniojo elektros tinklo sistemos komponentai).

Svarbu paminėti, kad namo modelyje Namų elektros suvartojimą reikšmių atvaizduojantys duomenys yra paimti iš jau atlikto Australijos industrijų departamento eksperimento, renkant realius elektros sunaudojimo duomenis [34].



3 Diagrama. Išmaniojo elektros tinklo sistemos komponentai

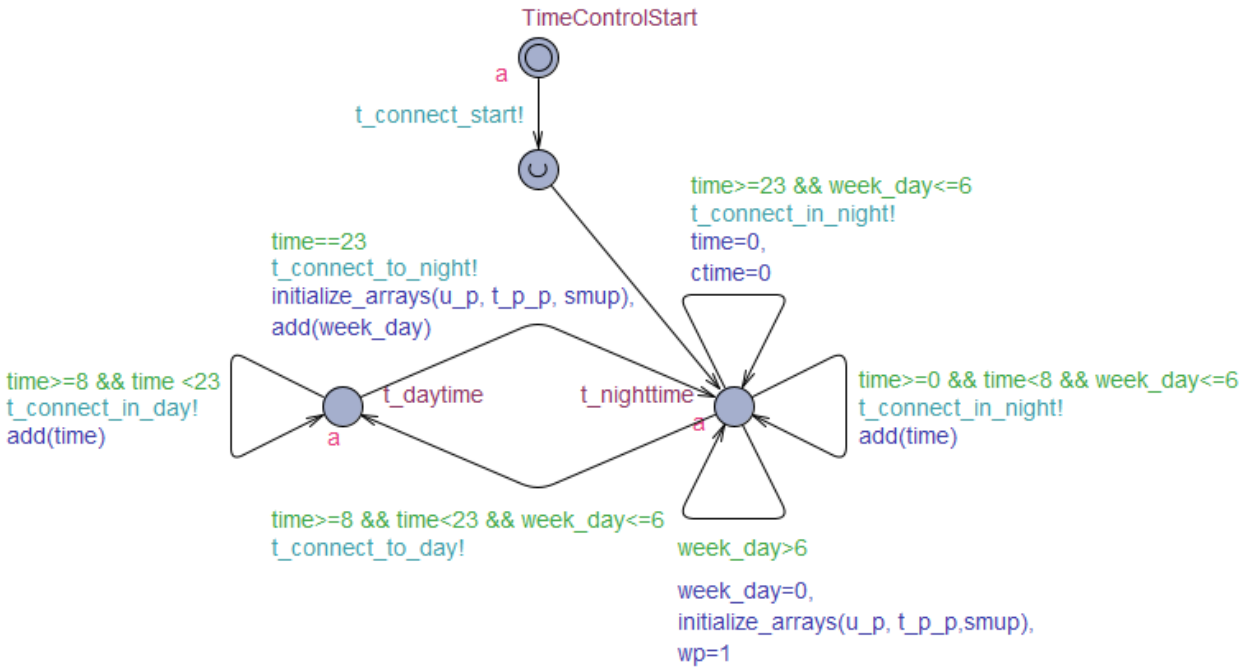
Laiko modelis reguliuoja, kad visų komponentų laikas būtų sinchronizuotas ir identiškas. Kaip pavyzdys yra namo modelis, kuris generuoja elektros sunaudojimo reikšmes pagal laiko modelio laiką. Už namo modelio reikšmių perdavimą sistemai yra atsakingas išmaniųjų skaitiklių modelis, kuris matuoja name elektros suvartojimą ir pateikia duomenis sistemai, išskyrus tuos atvejus, kai gedimo simuliacijos modelyje simuliuojamas gedimas. Gedimai iškraipo elektros suvartojimo paklausą, todėl gedimo ieškojimo modelis ieško šių gedimų naudodamasis sukaupta informacija apie namų elektros sunaudojimo istoriją ir lygindamas dabartinę reikšmę su tuo, kas tikimasi iš išmaniojo skaitiklio reikšmių. Atsiradus neatitikimui, sistema pažymi išmanųjį skaitiklį kaip nepatikimą (angl. *flagged*). Kol skaitiklis pažymėtas, kaip nepatikimas skaitiklis, išmaniojo skaitiklio valdiklis toliau generuoja reikšmes, kurios yra artimiausios realiai situacijai. Sistema

pažymi skaitiklį kaip nepatikimą tam tikrą laiko tarpą, jei valdiklio modelis sistemoje neužfiksuoja elektros dingimo, skaitiklis grąžinamas į normalią būseną ir nepatikimumo etiketė yra nuimama. Tuo atveju, jei valdiklio modelis užfiksuoja elektros dingimą, išmanusis skaitiklis pereina į taisymo būseną ir per sistemoje nurodytą laiką grįžta į normalią būseną (skaitiklis bus sutvarkytas fizinių asmenų per tam tikrą laiką).

Sistema taip pat palaiko elektros sistemos pasiūlą, t.y. modeliuoja energiją generuojančių modelių darbą. Reaguojant į esančią elektros paklausą yra keli būdai generuoti energiją: saulės baterijos, vėjo jėgainės ir elektrinė. Šie elektros šaltiniai yra atvaizduoti atitinkamais modeliais. Saulės baterijų ir vėjo jėgainių generuojama elektra priklauso nuo oro sąlygų, todėl naudojamas oro simuliacijos modelis, kuris simuliuoja vėją arba saulę. Nuo šio modelio priklauso atsinaujinančių elektros šaltinių efektyvumas. Kadangi minėti elektros šaltiniai neturi pastovumo generuojant energiją, papildomai egzistuoja elektros generavimo kontrolės modelis, kurio pagrindinis tikslas – kontroliuoti elektros generavimo modelio generuojamos elektros kiekį. Dabar pristatysime paminėtus sistemos modelius detaliau.

2.2.1. Laiko kontrolės modelis

Pirmas naudojamas modelis – Laiko Kontrolė (*angl. TimeControl*). Kadangi modeliuose naudojami duomenys yra priklausomi nuo paros valandinio laiko ir savaitės dienos, o sistemoje naudojamas laiko parametras laikmatis (*angl. real-valued clocks*) juda nepriklausomai nuo šios sistemos poreikių, tai yra gali judėti greičiau nei pasikeičia sistemos būsena, naudojamas modelis, kuris modeliuoja paros laiką ir savaitės dieną.



1 Modelis. Laiko kontrolės modelis

Šiame modelyje yra trys būsenos:

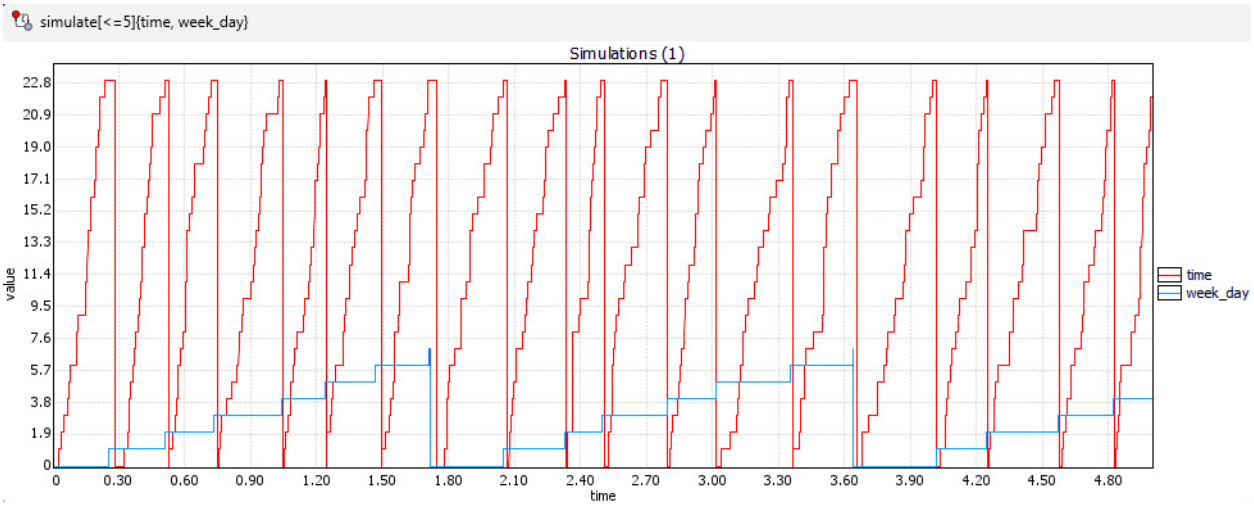
- Laiko kontrolės pradžia (*angl. TimeControlStart*) – pradinė modelio būsena, iš kurios išeina jungtis, kurioje yra transliuojantis kanalo kintamasis (*angl. broadcast channel*) „*t_connect_start*“. Šio kanalo sinchronizacija su kitais modelių kanalais signalizuoja, kad sistemos modeliai turėtų pradėti lygiagretų veikimą.
- Nakties laikas (*angl. t_nighttime*) – nakties laiko būsena, kuri yra naudojama simuliuoti nakties laiką. Su šia būsena yra susijusios šios jungtys:
    - Jungtis, kurioje laikmačio reikšmė tampa 0. Šios jungties kilpa aktyvuojama, jei laiko kintamasis – *time*, yra lygus arba mažesnis 23 valandoms, o savaitės dienos kintamasis – *week_day* yra mažesnis už 6 (savaitės seka yra [0;6], 0 – pirmadienis, 6 – sekmadienis). Šioje jungtyje yra naudojamas transliuojantis kanalas t_connect_in_night, kuris sinchronizuoja kitas modelių jungtis ir inicijuoja jų lygiagretų veikimą.
    - Jungtis, kurioje keičiama nakties laiko reikšmė. Laiko kintamojo reikšmė yra didinama kas valandą. Ši jungtis yra aktyvuojama, jei laiko kintamasis yra lygus arba daugiau už 0 ir mažiau už 8, taip pat jei savaitės diena yra mažesnė už 6. Šioje jungtyje taip pat yra

naudojamas anksčiau minėtas transliuojantis kanalas *t_connect_in_night,* kuris sinchronizuoja kitas modelių jungtis ir inicijuoja jų lygiagretų veikimą.

- Jungtis, kurioje savaitės dienos kintamais tampa 0, taip pat kituose modeliuose naudojami masyvai(*u_p* ir *smup* – masyvai kurie kaupia elektros pasiūlos istoriją) prilyginami 0. Ši kanalo jungtis aktyvuojama, jei savaitės kintamasis yra didesnis už 6. Taip pat kintamojo *wp* reikšmė tampa 1, signalizuojant, kad praėjo viena savaitė.
- Jungtis, kuri aktyvuojama, jei laiko kintamas yra didesnis arba lygus 8 ir mažesnis už 23, taip pat savaitės dienos kintamasis yra mažesnis už 6. Šioje jungtyje naudojamas *t_connect_to_day* kanalas, kuris sinchronizuoja kitų modelių perėjimą iš nakties būsenos į dienos būseną.

- Dienos laikas (*angl. t_daytime*) – dienos laiko būsena, kuri naudojama simuliuoti dienos laiką. Šioje būsenoje naudojamos šios jungtys:
  - Jungtis, kuriame keičiama dienos laiko reikšmė. Laiko kintamojo reikšmė yra didinama kas valandą. Ši kanalo jungtis yra aktyvuojama, jei laiko kintamasis yra lygus arba daugiau už 8 ir mažiau už 23. Šioje jungtyje yra naudojamas transliuojantis kanalas *t_connect_in_day*, kuris sinchronizuoja kitas modelių jungtis ir inicijuoja jų lygiagretų veikimą.
  - Jungtis, kuri aktyvuojamas, jei laiko reikšmė yra lygi arba didesnė 23. Šioje jungtyje naudojamas *t_connect_to_night* kanalas, kuris sinchronizuoja kitų modelių perėjimą iš dienos būsenos į nakties būseną.

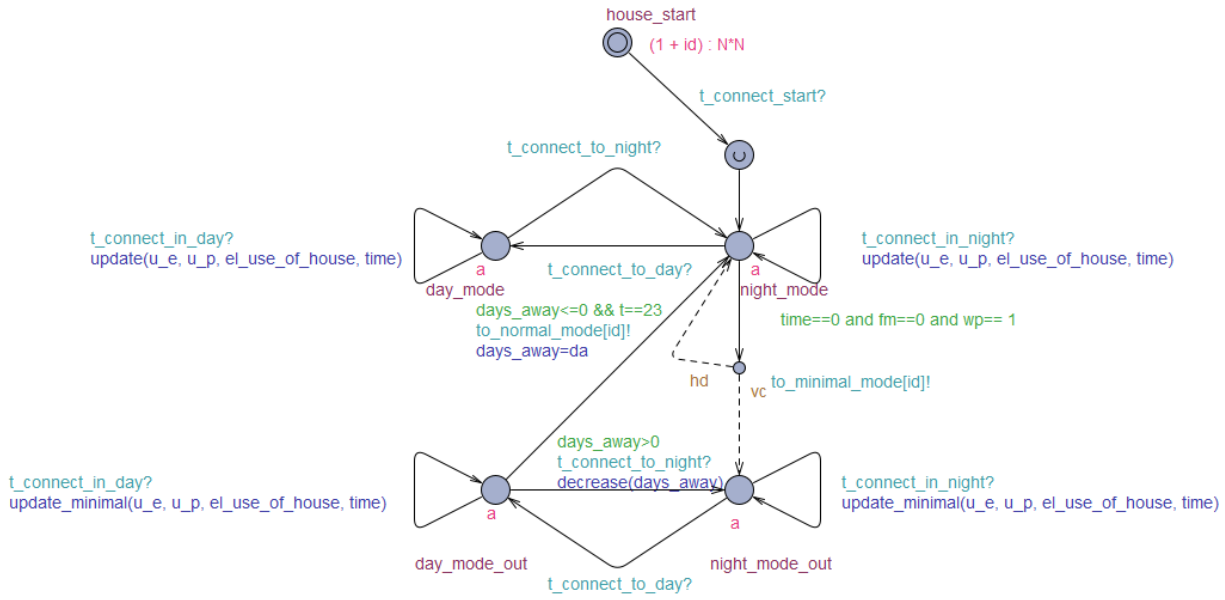
Šio modelio tikslas – sinchronizuotai kontroliuoti sistemoje esančius modelius, siekiant užtikrinti sinchronizuotą sistemos veikimą. Jame pagrindinė generuojama reikšmė – laikas ir savaitės diena (šių reikšmių generuojamas ciklas yra pavaizduotas4 **Diagrama**. Laiko ir savaitės generuojamos reikšmės sistemoje). Laiko ir savaitės kintamųjų reikšmės yra naudojami atvaizduojant namų modeliuose suvartojamą valandinį elektros kiekį.



4 Diagrama. Laiko ir savaitės generuojamos reikšmės sistemoje

2.2.2. Namo modelis

Antras naudojamas modelis – Namas (*angl. House*). Šis modelis yra naudojamas simuliuoti sunaudojamą elektros energiją sistemoje. Sistemoje gali egzistuoti kelios namo modelių kopijos; bendras jų kiekis yra apibrėžtas visos sistemos parametru N (namų skaičius simuliuojamoje sistemoje). Šis modelis, naudodamasis laiko modelio pateikiama informacija (*time*), generuoja namo sunaudojamas valandines elektros energijos reikšmes. Namo sunaudojama elektra gali skirtis priklausomai nuo namo *id* kintamojo.



2 Modelis. Namo modelis

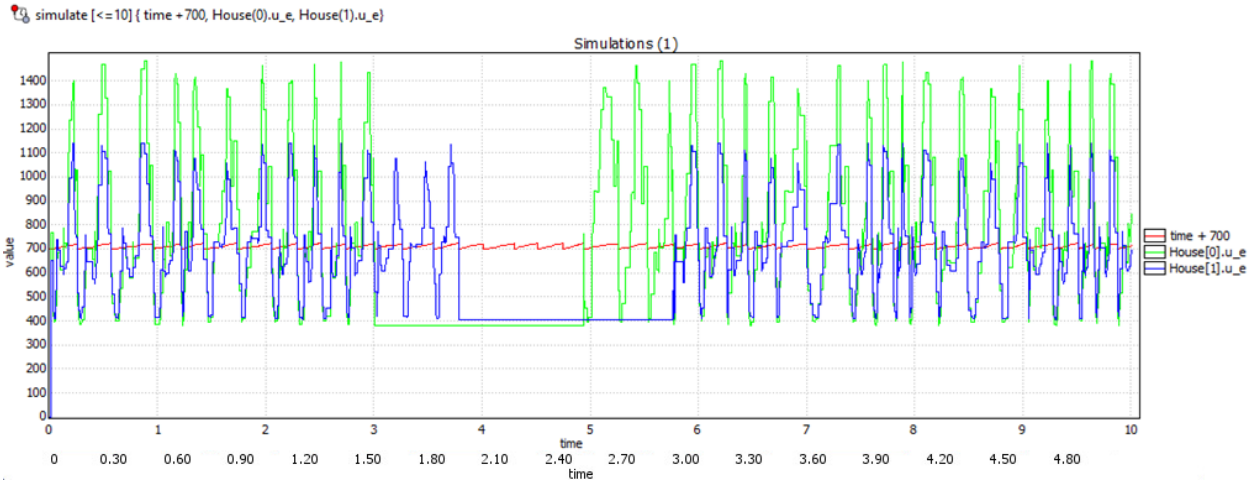
Šiame modelyje esančios būsenos:

- Namo būsenos pradžia – modelio pradžia, kuri sinchronizuojasi su laiko modeliu naudojantis *t_connect_start* transliuojančiu kanalu ir pradeda lygiagretų sistemos darbą.
- Nakties būsena (*angl. night_mode*)– šioje būsenoje esančios jungtys veikia tik nakties režimu. Su šia būsena yra susijusios šios jungtys:
  - Jungtis, kurioje apskaičiuojamas namo elektros sunaudojimas kilovatvalandėmis nakties režimu. Šioje kilpoje taip pat stebimas namo elektros individualus sunaudojimas ir bendras visų namų elektros sunaudojimas visoje sistemoje.
  - Jungtis, kurioje yra perėjimas į atostogų režimą (namas sunaudoja tik minimalų kiekį energijos). Tikimybė, kad namas pereis į šią būseną yra apibrėžtas *vc* ir *hd* kintamaisiais (*vc*-kokia tikimybė, kad namas pereis į atostogų režimą, *hd* – kokia tikimybė, kad namas liks dirbti normaliame režime), šie kintamieji padeda modeliuoti situacijas, kai sistema nežino ar namas perėjo į atostogų būseną.
  - Jungtis pereinanati iš nakties būsenos į dienos ir yra inicijuojama transliuojančio kanalo *t_connect_to_day* sinchronizacija.
- Dienos būsena (*angl. day_mode*) – šioje būsenoje esančios jungtys veikia tik dienos režimu. Su šia būsena yra susijusios šios jungtys:
  - Jungtis, kurioje apskaičiuojamas namo elektros sunaudojimas kilovatvalandėmis dienos režimu. Šioje kilpoje taip pat stebimas namo elektros individualus sunaudojimas ir bendras visų namų elektros sunaudojimas visoje sistemoje.
  - Jungtis pereinanati iš dienos būsenos į nakties ir yra inicijuojama su transliuojančio kanalo t_connect_to_night sinchronizacija.
- Atostogų nakties būsena (*angl. night_mode_out*) – šioje būsenoje pateikiama minimali namo valandinė reikšmė nakties režimu. Su šia būsena yra susijusios šios jungtys:
  - Jungtis, kurioje apskaičiuojamas namo elektros sunaudojimas kilovatvalandėmis nakties režimu, bet pateikiama minimali vertė. Kiekvienas namas gali pereiti į atostogų režimą individualiai.
  - Jungtis pereinanati iš nakties būsenos į dienos ir yra inicijuojama su transliuojančio kanalo t_connect_to_day sinchronizacija.

- Atostogų dienos būsena (*angl. day_mode_out*) – šioje būsenoje pateikiama minimali namo valandinė reikšme dienos režimu. Su šia būsena yra susijusios šios jungtys:
    - Jungtis, kurioje apskaičiuojamas namo elektros sunaudojimas kilovatvalandėmis dienos režimu, bet pateikiama minimali vertė. Šioje kilpoje taip pat stebimas namo elektros individualus sunaudojimas ir bendras visų namų elektros sunaudojimas visoje sistemoje. Kiekvienas namas gali pereiti į atostogų režimą individualiai.
    - Jungtis pereinanti iš dienos būsenos į nakties yra inicijuojama su transliuojančio kanalo t_connect_to_night sinchronizacija. Šioje būsenoje, taip pat sumažėja kintamojo days_away reikšmė, kuri nurodo, kiek atostogų dienų turi praeiti, kad namas grįžtų normalę veikimo būseną.
    - Jungtis grąžinanti namą į normalią dienos būseną. Ši jungtis aktyvuojama, jei days_away yra lygi 0 ir laiko (angl. *time* kintamojo) reikšmė yra lygi 23. Šioje jungtyje yra naudojamas transliuojantis kanalas to_normal_mode[id] (id – kintamasis, nurodantis namo numerį) kuris sinchronizuoja išmaniojo skaitiklio gedimo modelio jungtis ir inicijuoja jų lygiagretų veikimą.

Šiame modelyje pagrindinės generuojamos reikšmės yra namų elektros sunaudojimas kilovatvalandėmis. Modelis gali pereiti į atostogų režimą, kuriame namo suvartojama energija yra minimali (matomas tarpas5 Diagrama. Dviejų namų modelių generuojami elektros suvartojimo duomenys). Šie duomenys yra naudojami kituose modeliuose rinkti ir apdoroti elektros sunaudojimo duomenis, bei aptikti sugedusius išmaniuosius skaitiklius.

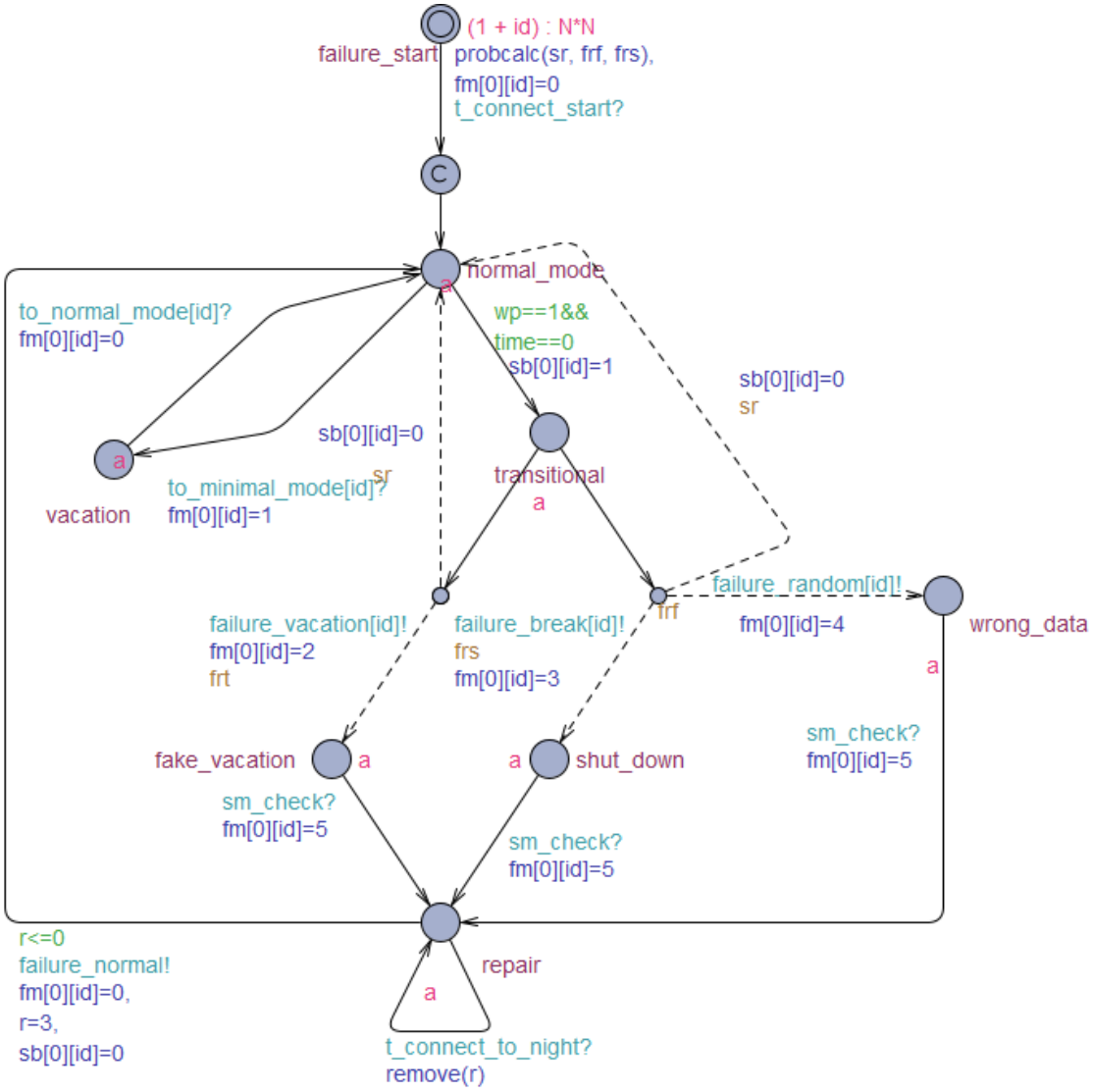
Svarbu paminėti, kad namo modelyje Namų elektros suvartojimo reikšmes atvaizduojantys duomenys (elektros valandinis sunaudojimo masyvas) yra paimti iš jau atlikto Australijos industrijų departamento eksperimento, renkant realius elektros sunaudojimo duomenis [34].



5 Diagrama. Dviejų namų modelių generuojami elektros suvartojimo duomenys

2.2.3. Gedimo simuliacijos modelis

Trečias naudojamas modelis – gedimo simuliacijos modelis. Šiame modelyje simuliuojamos keturios galimos situacijos, kurios gali daryti įtaką išmaniųjų skaitiklių duomenims: atostogos, suklastotos atostogos, skaitiklio atsijungimas ir skaitiklio duomenų iškraipymas. Atostogų būsena atsiranda namo modeliui perėjus į atostogų režimą (minimalų elektros sunaudojimo režimą), tuo tarpu kitos būsenos turi susietą su jomis tikimybę.



3 Modelis. Gedimų simuliacijos modelis

Šiame modelyje esančios būsenos:

- Sutrikimų pradžia (*angl. failure_start*) – modelio pradžia, kuri sinchronizuojasi su laiko modeliu naudojantis *t_connect_start* transliuojančiu kanalu ir pradeda lygiagretų sistemos darbą.
- Normali būsena (*angl. normal_mode*) – normali būsena, kai išmanusis skaitiklis nepatiria jokių gedimų. Su šia būsena yra susijusios šios jungtys:

- Jungtis, kuri perkelia modelį į tarpinę būseną. Norint patekti į tarpinę būsena *wp* kintamasis turi būti lygus 1 (šis boolean kintamasis yra lygus 1, kai praėjo savaitė, taip siekiant apmokinti sistemą, prieš leidžianti jai sugesti). Ši jungtis taip pat pakeičia *sb* kintamojo reikšmę, kuri indikuoja, kad modelis, kurio numeris yra numeis *id* gali būti sugedęs.
- Jungtis pereinanti į atostogų (*angl. vacation*) būseną. Ši jungtis suveikia, kai modelis namas pereina į atostogų būseną (naudojantis kanalu *to_minimal_mode[id]*). Šioje jungtyje *fm* reikšmė tampa 1, nurodant kad tai pirmo tipo „gedimas“ (sistemoje užfiksuojama kaip anomalija, bet nedingstant energijai, neturėtų niekas pasikeisti).

- Tarpinė būsena (*angl. transitional*) – tarpinė būsena tarp normalios būsenos ir gedimus apibūdinančių būsenų. Su šia būsena yra susijusios šios jungtys:
  - Jungtis iš kurios modelis pereis į normalią būseną arba suklastotų atostogų (*angl. fake_vacation*) Tikimybė, kad modelis pereis į šias būsenas yra apibrėžtos sr ir frt kintamaisiais (sr – tikimybė, kad modelis grįš į normalią būseną, frt – tikimybė, kad modelis pereis į suklastotų atostogų būseną). Perėjus į normalią būseną, sb kintamasis (sb – kintamasis naudojamas patikrinti ar skaitiklis perėjo į gedimo būseną) yra lygus 0, visais kitais atvejais – 1. Kintamasis *fm* nurodo kokį gedimą patiria modelis (*fm* reikšmė priklauso nuo būsenos į kurią būseną modelis perėjo, jei perėjimas yra į normalią būseną reikšmė išlieka lygi 0, jei pereinama į suklastotų atostogų – 2).
  - Jungtis iš kurios modelis pereis į išsijungimo (*angl. shut_down*) arba atsitiktinių duomenų būsenas (*angl. failure_random*). Tikimybė, kad modelis pereis į šias būsenas yra apibrėžtos *sr, frs* ir *frf* tikimybėmis (*sr* – tikimybė, kad modelis grįš į normalią būseną, *frs* – tikimybė, kad išmanusis skaitiklis pereis į išsijungimo būseną, *frf* – tikimybė, kad modelis pereis į atsitiktinių duomenų būsenas). Perėjus į normalią būseną, *sb* kintamasis (kuris naudojamas patikrinti ar skaitiklis perėjo į gedimo funkcija) yra lygus 0, visais kitais atvejais – 1. Kintamasis *fm*, nurodo kokį gedimą patiria modelis. Jei pereinant į normalią būseną reikšmė išlieka lygi 0, perėjus į išsijungimo – 3, perėjus į atsitiktinių duomenų – 4, netikrų atostogų – 2.

- Atostogų būsena (*angl. vacation*) ši būsena atsiranda, kai namo modelis pereina į atostogų režimą. Su šia būsena susijusios jungtys:

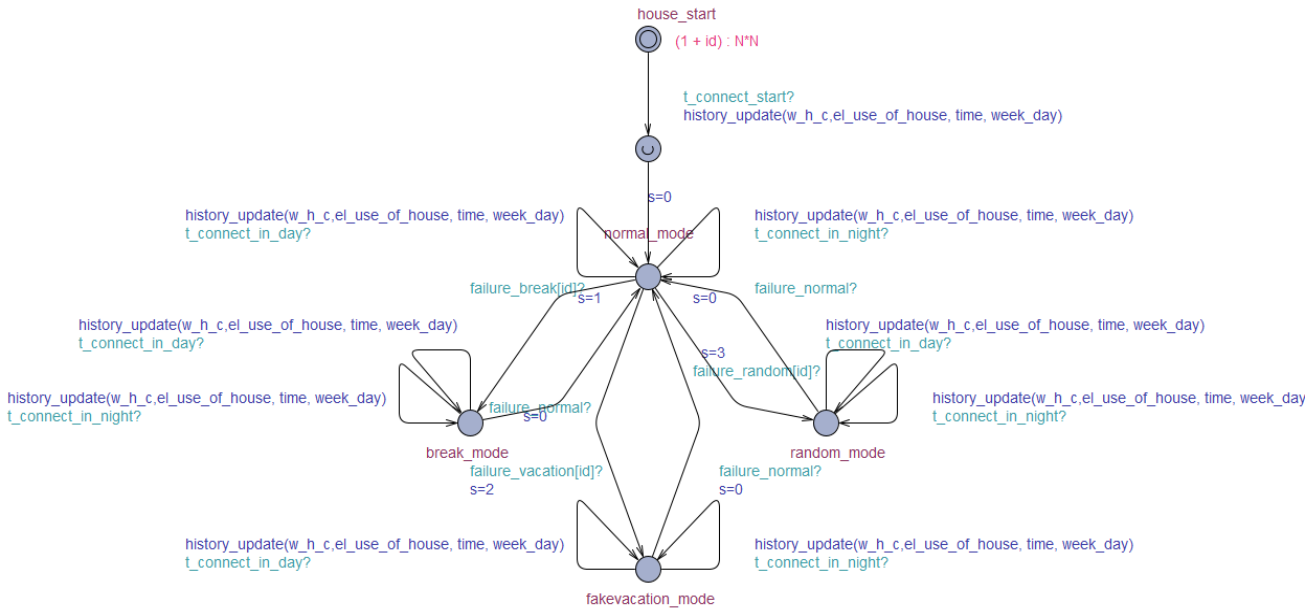
- Jungtis, kuri pereina į normalų režimą. Namo modeliui perėjus į normalų režimą, į šį režimą taip pat pereina ir gedimo modelis.

- Suklastotų atostogų būsena (*angl. fake_vacation*) – ši būsena pasiekiama, kai namas sunaudoja jam normalų kiekį energijos, bet išmanusis skaitiklis rodo, kad namas perėjo į atostogų būseną ir sunaudoja minimalų kiekį energijos. Su šia būsena susijusios jungtys yra:
  - Jungtis pereinanti į taisymo (*angl. repair*) būseną kartu su sinchronizavimu *sm_check?* kanalu (kanalas naudojamas valdiklio modelyje, jo liepimu visi trūkumus turintys modeliai, pereina į sąrašą skaitiklių, kuriuos reikia patikrinti). Reikšmė fm yra lygi 5, tuo siekiant parodyti, kad šis skaitiklis dabar yra taisomas.
- Išsijungimo būsena (*angl. shut_down*) – ši būsena pasiekiama, kai skaitiklis visai atsisako veikti (nėra siunčiamos reikšmės). Nuo atostogų būsenos skiriasi tuo, kad atostogų būsenoje siunčiami minimalios reikšmės, tuo tarpu šioje būsenoje, modelis negamina jokių reikšmių. Su šia būsena susijusios jungtys yra:
  - Jungtis pereinanti į taisymo (*angl. repair*) būseną kartu su sinchronizavimu *sm_check?* kanalu (kanalas naudojamas valdiklio modelyje, jo liepimu visi trūkumus turintys modeliai, pereina į sąrašą skaitiklių, kuriuos reikia patikrinti). Reikšmė fm yra lygi 5, tuo siekiant parodyti, kad šis skaitiklis dabar yra taisomas.
- Atsitiktinių duomenų būsena (*angl. failure_random*) – ši būsena pasiekima, kai skaitiklis pradeda siųsti reikšmes, kurios neatitinka namo sunaudojamų reikšmių (reikšmės yra per didelės arba per mažos). Su šia būsena susijusios jungtis yra:
  - Jungtis pereinanti į taisymo (*angl. repair*) būseną kartu su sinchronizavimu *sm_check?* kanalu (kanalas naudojamas valdiklio modelyje, jo liepimu visi trūkumus turintys modeliai, pereina į sąrašą skaitiklių, kuriuos reikia patikrinti). Reikšmė fm yra lygi 5, tuo siekiant parodyti, kad šis skaitiklis dabar yra taisomas.

Gedimų modelis negeneruoja reikšmių, jo pagrindinė paskirtis – modeliuoti gedimus, nuo šio modelio priklauso išmanaus skaitiklio, gedimų aptikimo ir valdiklio modelių darbas.

2.2.4. Išmaniojo skaitiklio modelis

Ketvirtas naudojamas modelis – išmaniojo skaitiklio modelis. Šiame modelyje simuliuojamas išmaniojo skaitiklio veikimas, tai yra namo duomenų perdavimas į sistemą ir lokalaus elektros sunaudojami istorijos kaupimas. Šiame modelyje yra keturios pagrindinės būsenos: normali būsena (*angl. normal_mode*), sugedimo būsena (*angl. break_mode*), suklastotų atostogų būsena (*angl. fakevacation_mode*) ir atsitiktinių duomenų būsena (*angl. random_mode*). Šios būsenos lemia, kokias reikšmes skaitiklis pateiks į sistemą. Jos pasiekiamos, kai gedimo simuliacijos modelis simuliuoja gedimą, tokiu atveju skaitiklis pereina į gedimo būseną ir pradeda teikti numatytas reikšmes.



4 Modelis. išmaniojo skaitiklio modelis

Šiame modelyje esančios būsenos:

- Išmaniojo skaitiklio pradžios būsena (*angl. sm_start*) – modelio pradžia, kuri sinchronizuojasi su laiko modeliu naudojantis t_connect_start transliuojančiu kanalu ir pradeda lygiagretų sistemos darbą.
- Normali būsena (*angl. normal_mode*) – šioje būsenoje reikšmės pateikiamos į sistemą sustampa su namo modelio generuojamomis reikšmėmis ir atnaujina lokalią išmaniojo skaitiklio duomenų istorija. Su šia būsena susijusios jungtys yra:

- Jungtis, kurioje atnaujima lokali išmaniojo skaitiklio elektros sunaudojimo istorija dienos metu ir sinchronizuojasi su laiko modeliu naudojantis t_connect_in_day transliuojančiu kanalu ir grįžta į pradinę normalią būsena. Sunaudojimo istorija atnaujinama kas valandinį sistemos vienetą.
- Jungtis, kurioje atnaujima lokali išmaniojo skaitiklio elektros sunaudojimo istorija nakties metu ir sinchronizuojasi su laiko modeliu naudojantis t_connect_in_night transliuojančiu kanalu ir grįžta į pradinę normalią būsena. Sunaudojimo istorija atnaujinama kas valandinį sistemos vienetą.
- Jungtis, kuri pereina į išsijungimo gedimo būseną. Gedimo modeliui simuliuojant šį gedimą, naudojantis *failure_break[id]* (*id* – skaitiklio ir namo numeris) transliuojančiu kanalu įvyksta perėjimas į išsijungimo gedimo būseną.
- Jungtis, kuri pereina į suklastotų atostogų būseną. Gedimo modeliui simuliuojant šį gedimą, naudojantis failure_vacation[id] (id – skaitiklio ir namo numeris) transliuojančiu kanalu įvyksta perėjimas į suklastotų atostogų būseną.
- Jungtis, kuri pereina į atsitiktinių reikšmių būseną. Gedimo modeliui simuliuojant šį gedimą, naudojantis *failure_random[id]* (id – skaitiklio ir namo numeris) transliuojančiu kanalu įvyksta perėjimas į atsitiktinių reikšmių būseną.

- Išsijungimo gedimo būsena (*angl. break_mode*) – ši būsena pasiekiama gedimų simuliacijos modeliui simuliuojant gedimo modelį. Esant šioje būsenoje, skaitiklis neperduoda į sistema jokių reikšmių, kai tuo tarpu namo modelis vis dar naudoja energiją. Sistemoje egzistuoja didesnė elektros paklausa, nei sistemai matoma paklausa. Šioje būsenoje veikia šios jungtys:
  - Jungtis, kurioje atnaujinama lokali išmaniojo skaitiklio elektros sunaudojimo istorija dienos metu ir sinchronizuojasi su laiko modeliu naudojantis *t_connect_in_day* transliuojančiu kanalu ir grįžta į pradinę gedimo būseną.
  - Jungtis, kurioje atnaujima lokali išmaniojo skaitiklio elektros sunaudojimo istorija nakties metu ir sinchronizuojasi su laiko modeliu naudojantis *t_connect_in_night* transliuojančiu kanalu ir grįžta į pradinę normalią būseną.

- Jungtis, kuri pereina į normalia būseną. Šis modelis sinchronizuojasi su gedimų simuliacijos modeliu ir praėjus laikotarpiui, per kurį išmanusis skaitiklis yra sutaisomas, grąžina modelį į normalią būseną.

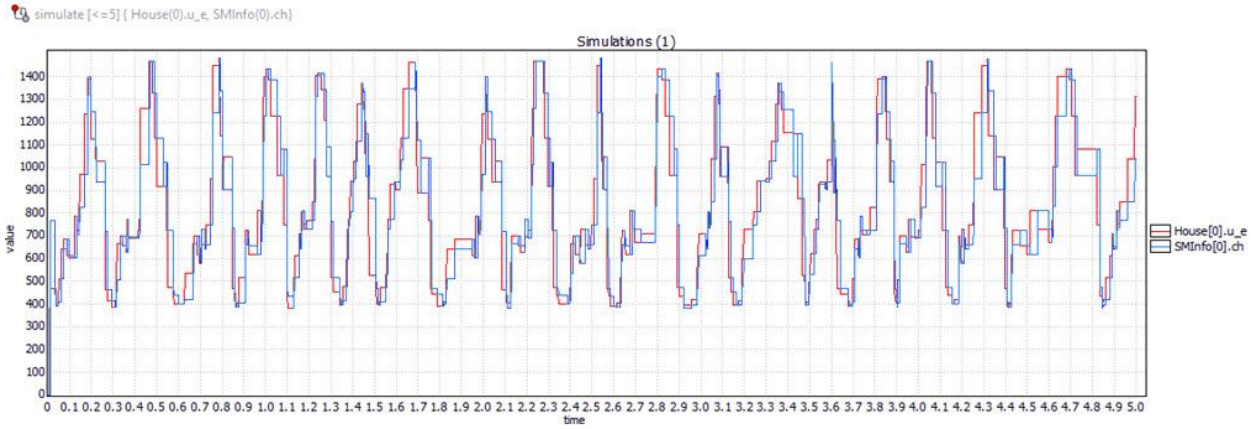
- Suklastotų atostogų būsena (*angl. fakevacation_mode*) – ši būsena pasiekiama gedimų simuliacijos modeliui simuliuojant atostogų klastojama. Esant šioje būsenoje, skaitiklis perduoda minimalias reikšmes į sistema, kai tuo tarpu namo modelis sunaudoja didesnį kiekį energijos negu pranešama sistemai. Sistemoje egsistuoja didesnė elektros paklausa, nei sistemai matoma paklausa. Šioje būsenoje veikia šios jungtys:
  - Jungtis, kurioje atnaujinama lokali išmaniojo skaitiklio elektros sunaudojimo istorija dienos metu ir sinchronizuojasi su laiko modeliu naudojantis *t_connect_in_day* transliuojančiu kanalu ir grįžta į atostogų klastojimo būseną.
  - Jungtis, kurioje atnaujinama lokali išmaniojo skaitiklio elektros sunaudojimo istorija nakties metu ir sinchronizuojasi su laiko modeliu naudojantis t_connect_in_night transliuojančiu kanalu ir grįžta į atostogų klastojimo būseną.
  - Jungtis, kuri pereina į normalia būseną. Šis modelis sinchronizuojasi su gedimų simuliacijos modeliu ir praėjus laikotarpiui, per kurį išmanusis skaitiklis yra sutaisomas, modelis grąžinamas į normalią būseną.

- Atsitiktinių reikšmių būsena (*angl. random_mode*) – būsena, kurioje sistemai perduodamos reikšmės neatitinka namo sunaudojimo elektros kiekio.
  - Jungtis, kurioje atnaujinama lokali išmaniojo skaitiklio elektros sunaudojimo istorija dienos metu, vyksta sinchronizacija su laiko modeliu naudojantis *t_connect_in_day* transliuojančiu kanalu ir grįžtama į atsitiktinių reikšmių būseną.
  - Jungtis, kurioje atnaujinama lokali išmaniojo skaitiklio elektros sunaudojimo istorija nakties metu, vyksta sinchronizacija su laiko modeliu naudojantis *t_connect_in_night* transliuojančiu kanalu ir grįžtama į pradinę atsitiktinių reikšmių būseną.
  - Jungtis, kuri pereina į normalia būseną. Šis modelis sinchronizuojasi su gedimų simuliacijos modeliu ir praėjus laikotarpiui, per kurį išmanusis skaitiklis yra sutaisomas, grąžina modelį į normalią būseną.

Šiame modelyje matome pasikartojančius procesus: elektros suvartojimo istorijos atnaujinimą dienos ir nakties metu ir grįžimą į normalią būseną. Pagrindinis skirtumas tarp būsenų yra kokie duomenys transliuojami į sistemą ir kaip jie yra išsaugomi. Normaliojo būsenoje, renkami ir saugomi duomenys atvaizduos tikslų sistemos vaizdą, kai tuo tarpu kitos būsenos iškreips tuos duomenis, pavyzdžiui visiško gedimo būsenoje, skaitiklis nerinks duomenų, bet bandys atnaujinti lokalią istoriją nulinėmis reikšmėmis, suklastotų atostogų būsenoje bus bandoma pateikti duomenis į sistemą minimaliomis reikšmes ir atnaujinti istoriją naudojantis klaidingomis reikšmėmis, o atsitiktinių reikšmių būsenoje bus pateikiamos atsitiktinės reikšmės, taip pat bandant iškreipti lokalios istorijos reikšmes.

Lokali istorija kaupiami saugant tris masyvus: savaitgalio valandinių reikšmių masyvas, dienos valandinių reikšmių masyvas ir minimalių reikšmių masyvas. Šie masyvai yra apibrėžti kintamuoju *w_h_c*. Darbo dienos laikotarpiu yra skaičiuojamas darbo dienos elektros sunaudojimo valandinis vidurkis, savaitgalio metu yra skaičiuojamas savaitgalio elektros sunaudojimo vidurkis ir yra nuolat stebimas minimalus namo sunaudojimo kiekis per valandą. Šių kintamųjų prasmė yra žinoti, kokių reikšmių reikia tikėtis iš sistemų ateityje ir aptikti gedimus naudojantis istorijos neatitikimu su realybe.

Šis modelis generuoja išmaniojo skaitiklio reikšmes kurios yra pateikiamos į sistemą (6 Diagrama. Namo ir išmaniojo skaitiklio generuojami elektros suvartojimo duomenys). Modelis gali pereiti į normalią, sugedusią, atsitiktinių reikšmių ir suklastotų atostogų būseną, bet normalioje būsenoje pateikiamos reikšmės atitinka name elektros sunaudojimo reikšmes.

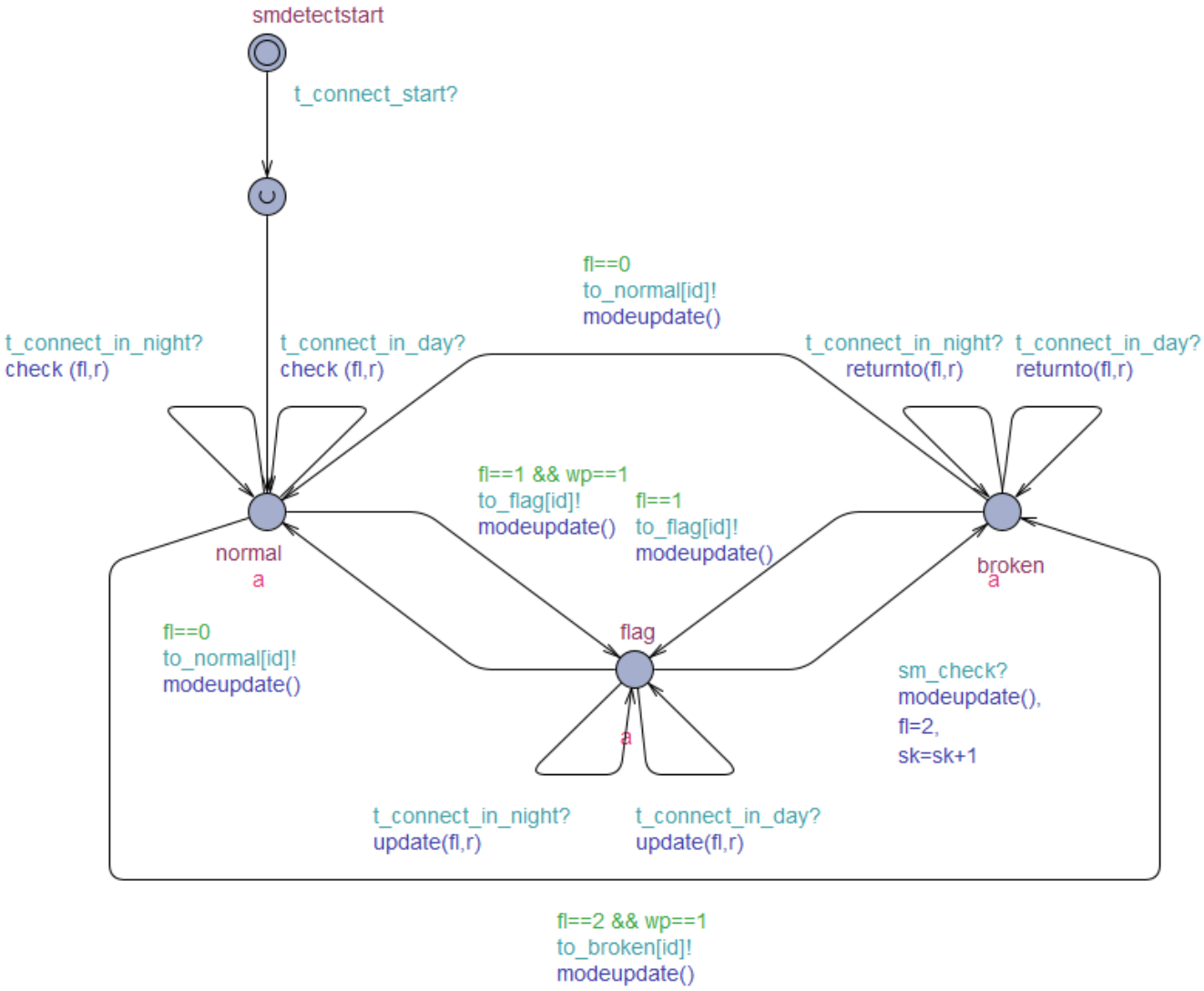


6 Diagrama. Namo ir išmaniojo skaitiklio generuojami elektros suvartojimo duomenys

Šioje diagramoje (6 Diagrama. Namo ir išmaniojo skaitiklio generuojami elektros suvartojimo duomenys) matome namo *House(0)* ir išmaniojo skaitiklio *SMInfo(0)* generuojamas reikšmes. Raudona linija reprezentuoja namo suvartojamą elektros kiekį, o mėlyna atvaizduoja namo elektros sunaudotas reikšmes, kurias užfiksavo išmanusis skaitiklis. Šio grafiko tikslas yra vizualiai atvaizduoti išmaniojo skaitiklio kaupiamų duomenų tikslumą.

2.2.5. Gedimo ieškojimo modelis

Penktasis modelis – gedimo ieškojimo modelis. Šis modelis stebi neatitikimus pateikiamose reikšmėse ir lygina jas su sukauptais duomenimis. Atradus duomenų neatitikimą sukauptai elektros energijos suvartojimo istorijai, išmanusis skaitiklis yra pažymimas kaip rizikingas (*angl. flagged*), sistemoje saugomos elektros suvartojimo istorijos nebeatnaujinamos ir yra laukiama signalo iš valdiklio, ar sunaudojamas elektros kiekis atitinka sistemai žinomą elektros sunaudojamą kiekį. Jeigu nustatoma, kad sunaudojamas elektros kiekis neatitinka žinomam elektros sunaudojimo kiekiui, išmanusis skaitiklis sistemoje pažymimas kaip sugedęs. Jeigu sunaudojamas elektros kiekis atitinka sistemoje žinomą sunaudojamą elektros kiekį, nuo išmaniojo skaitiklio nuima rizikingo namo etiketė ir toliau tęsiamas duomenų kaupimo darbas.



5 Modelis. gedimo ieškojimo modelis

Gedimo ieškojimo modelio būsenos:

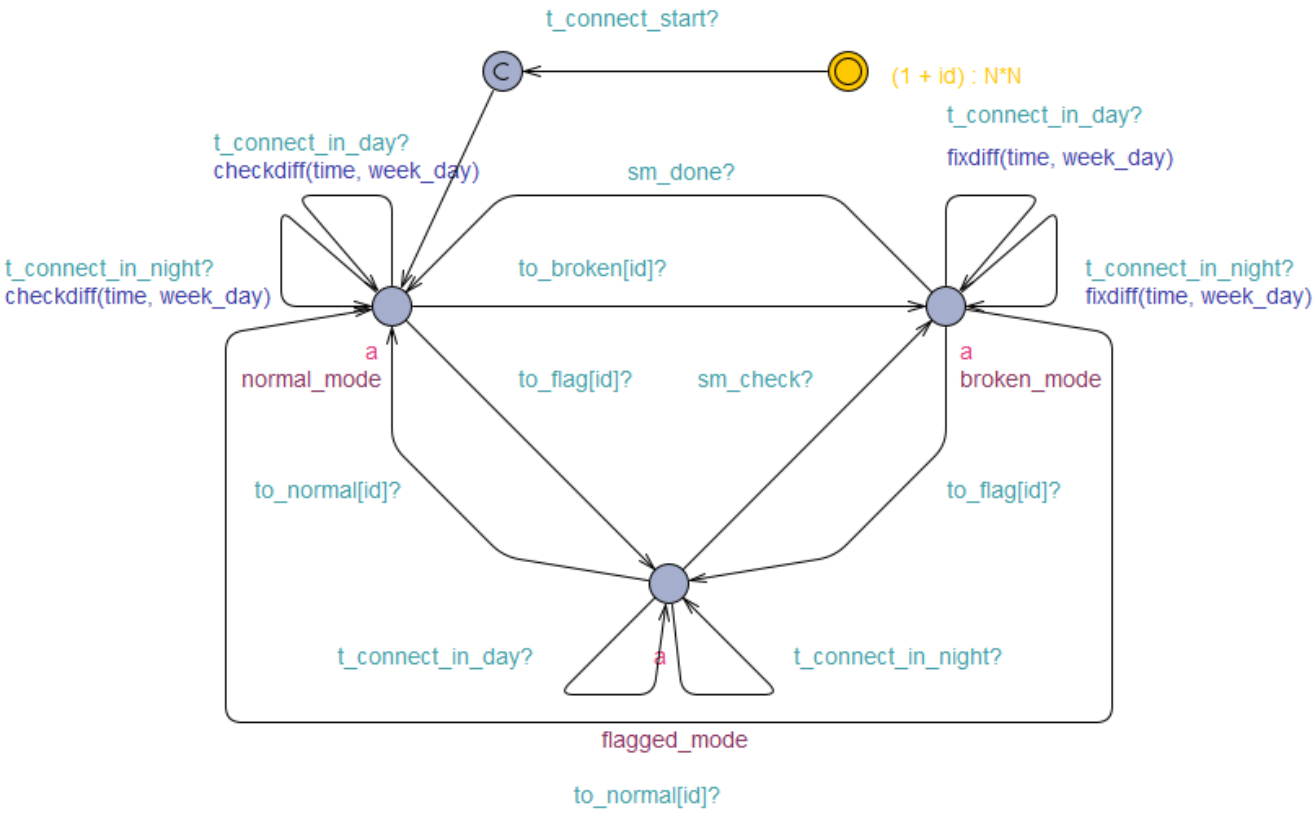
- Gedimo ieškojimo pradinė būsena (angl. *smdetectstart*) – modelio pradžia, kuri sinchronizuojasi su laiko modeliu naudojantis t_connect_start transliuojančiu kanalu ir pradeda lygiagretų sistemos darbą.
- Normali būsena (angl. normal) – būsena, kurioje sistema nerado jokių gedimų, bet nuolat tikrina ar išmaniojo skaitiklio reikšmės atitinka tokias reikšmes, kokių yra tikimasi, atsvelgiant į sukauptą elektros naudojimo istoriją. Šios būsenos jungtys:

- Dienos ir nakties jungtys, kurios naudojasi kanalais t_connect_in_day ir t_connect_in_night, tikrinti ar reikšmės atitinka tokias, kokių tikimasi.
- Jungtis, sujungianti normalią būseną su flag būsena, reiškianti kad galimai rasta klaida, bet dar reikia palyginti ar tai yra tik anomalija ar realus gedimas.
- Jungtis, sujungianti normalią būseną su broken būsena, tai reiškianti kad išmanusis skaitiklis paduoda nulines reikšmes ir šis skaitiklis sugedęs.

- Pažymėta būsena (angl. flag) – šioje būsenoje tikrinama, ar skaitiklio reikšmės vis dar neatitinka tai ko mes tikimės iš gaunamų reikšmių, skaičiuojamas laikas ir laukiama ar valdiklis perkels šį išmanųjį skaitiklį į sugedusią būseną. Jei nustatytas laikas praeina ir valdiklis neperkelia išmaniojo skaitiklio į sugedusią būseną, skaitiklis grąžinamas į normalią būseną. Šios būsenos jungtys:
  - Dienos ir nakties jungtys, kurios naudojasi kanalais t_connect_in_day ir t_connect_in_night, tikrinti ar reikšmės neatitinka tokių, kokių tikimasi.
  - Jungtis, kuri perkelia išmaniojo skaitiklio būseną į sugedusią būseną. Taip nutinka, jei valdiklis atranda elektros paklausos neatitikimų ir reikia tikrinti visus probleminius skaitiklius.
  - Jungtis, kuri perkelia iš pažymėtos būsenos į normalią, pasibaigus nustatytam laikui ir valdikliui neradus problemų elektros paklausoje.
- Sugedusi būsena (angl. *flag*) – šioje būsenoje esantis skaitiklis pažymimas sugedusiu ir bus šioje būsenoje tol kol jis bus pataisytas (sistemoje praeis nurodytas laikotarpis per kurį skaitiklis turi būti pataisytas).

2.2.6. Išmaniojo skaitiklio valdiklio modelis

Šiame modelyje pagrindinis tikslas yra tvarkytis su duomenų saugojimu kritinių situacijų metu, kai išmaniuoju skaitiklio informacija negalima pasitikėti arba jis nebesiunčia duomenų.



6 Modelis. Išmaniojo skaitiklio valdiklio modelis

Šio modelio būsenos yra identiškos gedimo ieškojimo modelio būsenoms:

- Išmaniojo skaitiklio valdiklio pradinė būsena (angl. *smdetectstart*) – modelio pradžia, kuri sinchronizuojasi su laiko modeliu naudojantis t_connect_start transliuojančiu kanalu ir pradeda lygiagretų sistemos darbą.
- Normali būsena, kurioje sistema nerado jokių gedimų, bet nuolat tikrina ar išmaniojo skaitiklio reikšmės atitinka tokias reikšmes, kokių yra tikimasi, atsvelgiant į sukauptą elektros naudojimo istoriją. Šios būsenos jungtys:
    - Dienos ir nakties jungtys, kurios naudojasi kanalais *t_connect_in_day* ir *t_connect_in_night*, tikrinti ar reikšmės atitinka tokias, kokių tikimasi.
    - Jungtis, sujungianti normalią būseną su pažymėta būsena (angl. *flagged_mode*), aktyvuojamas kanalas, kai sistema nusprendžia, kad išmaniojo skaitiklio modelio gaunamais duomenimis negalima pasitikėti.

- Jungtis, sujungianti normalią būseną su *broken* būsena, tai reiškia kad išmanusis skaitiklis paduoda nulines reikšmes ir šis skaitiklis yra sugedęs.

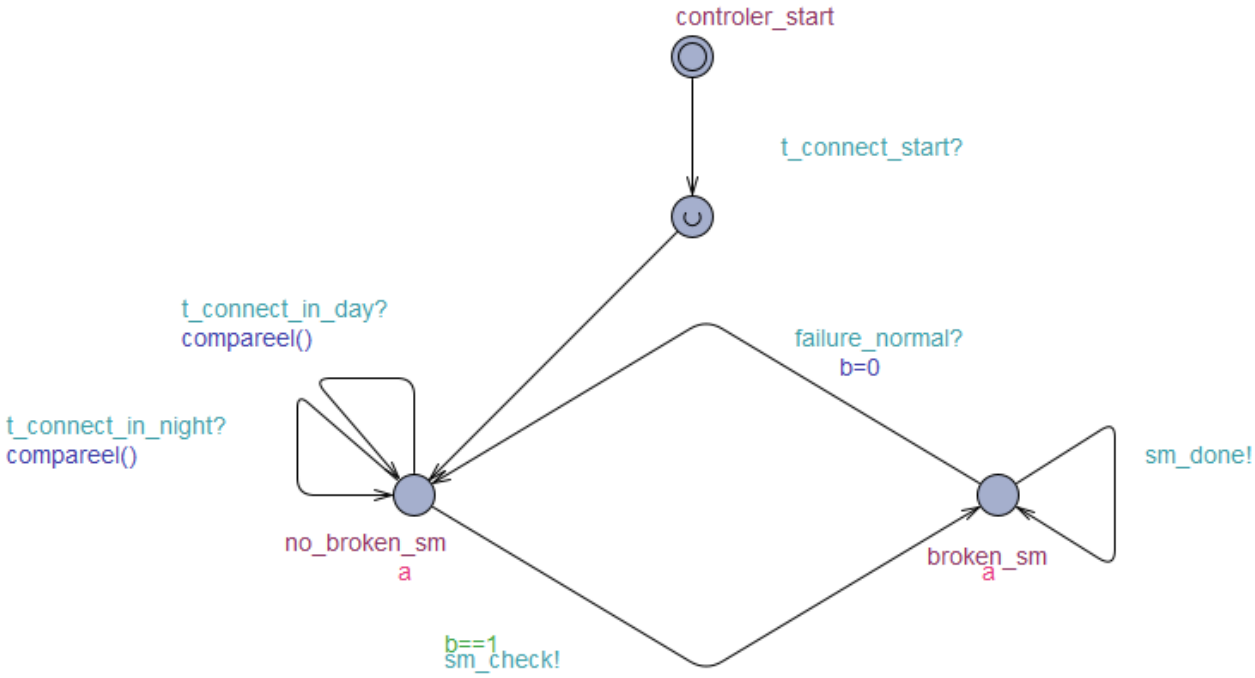
- Pažymėta būsena (angl. *flagged_mode*) – šioje būsenoje saugoma paros sunaudojimo masyvo vidurkio kopija, nedarant jai jokių pakeitimų, bet originalios istorijos reikšmės vis dar atnaujinamos gaunamomis reikšmėmis. Šios būsenos jungtys:
  - Dienos ir nakties jungtys, kurios naudojasi kanalais *t_connect_in_day* ir *t_connect_in_night,* jokie pakeitimai nevyksta sistemoje saugomai kopijai, bet vyksta originalaus paros elektros suvartojimo istorijos atnaujinimas.
  - Jungtis, kuri perkelia išmaniojo skaitiklio būseną į sugedusią būseną. Taip nutinka jei valdiklis atranda elektros paklausos neatitikimų ir reikia sureguliuoti.
  - Jungtis, kuri perkelia iš pažymėtos būsenos į normalią, pasibaigus nustatytam laikui ir valdikliui neradus problemų elektros paklausoje.

- Sugedusi būsena (*angl. broken_mode*) – šioje būsenoje esantis skaitiklis pažymimas sugedusiu ir šioje būsenoje tol kol jis bus pataisytas (sistemoje praeis nurodytas laikotarpis per kurį skaitiklis turi būti pataisytas), elektros paklausa bus apskaičiuojama remiantis istorijos duomenimis.

Išmaniojo skaitiklio valdiklio modelio tikslas yra pastebėti neatitikimus jo stebimame išmaniajame skaitiklyje. Išmaniojo skaitiklio valdiklis pats negali nustatyti, kad išmanusis skaitiklis sugedo, nebent išmanusis skaitiklis visiškai nustoja siųsti reikšmes. Šio modelio tikslas yra stebėti ar yra neatitikimų tarp išmaniojo skaitiklio generuojamų reikšmių ir sukauptos istorijos. Jei neatitikimai aptinkami, išmanusis skaitiklis pažymimas, kaip įtartinas (*angl. flagged mode*).

2.2.7. Valdiklio modelis

Šio modelio tikslas yra stebėti sistemoje esančią elektros paklausą ir pastebėti gedimus susijusius su sugedusiais, modifikuotais ar apgadintais išmaniaisiais skaitikliais. Tai realizuojama lyginant elektros energijos matavimus matuojant bendrą elektros sunaudojimą visoje sistemoje (šioje sistemoje matuojamas bendras tam tikros teritorijos elektros sunaudojamas kiekis) ir sumą individualių namų, kurie pateikia elektros sunaudojimo istoriją į sistemą. Atradus neatitikimą visi skaitikliai, kurie yra pažymėti kaip nepatikimi, yra perkeliami į sugedusių būseną.



7 Modelis. Valdiklio modelis

Šio modelio būsenos:

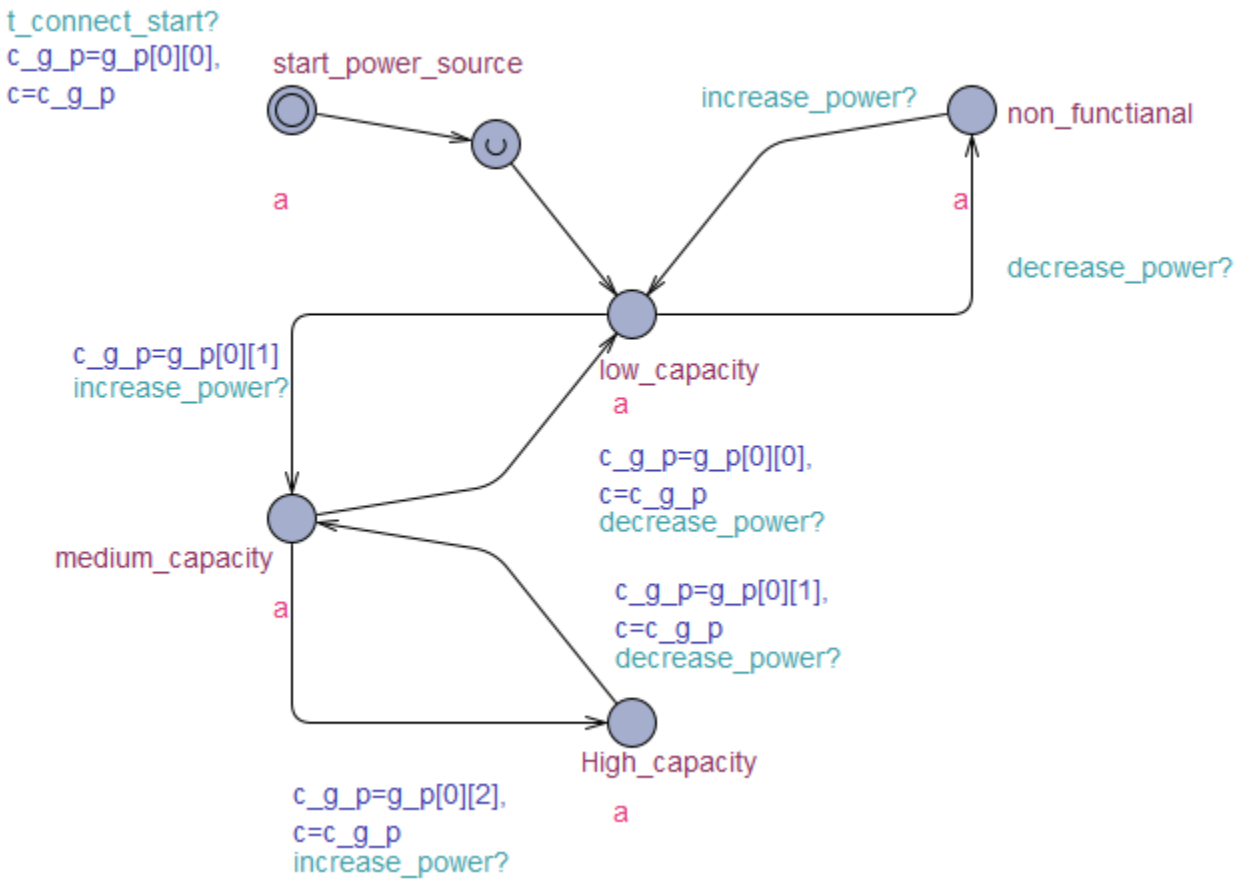
- Valdiklio pradžia (angl. *controler_start*) – modelio pradžia, kuri sinchronizuojasi su laiko modeliu naudojantis t_connect_start transliuojančiu kanalu ir pradeda lygiagretų sistemos darbą.
- Sistemoje neužfiksuota gedimų (angl. no_broken_sm) – sistemos būsena, kai nėra aptikta gedimų. Šioje būsenoje naudojamos jungtys:
  - Dienos ir nakties jungtys, kurios naudojasi kanalais t_connect_in_day ir t_connect_in_night, lyginančios skaitiklių pateikiamas reikšmes ir pamatuotas elektros sunaudojimo reikšmes.
  - Jungtis, pereinanti į gedimai užfiksuoti būsena. Ši būsena pasiekiama, kai sistema randa neatitikimus elektros paklausoje.
- Sistemoje užfiksuota gedimų (angl. broken_sm) – būsena, kuri pasiekiama užfiksavus gedimą. Šios būsenos jungtys:
  - Jungtis, kuri turi kanalą, kuris praneša apie šios būsenos pasiekimą kitiems modeliams.

- Jungtis, grąžinanti Valdiklį į normalią būseną, tuo atveju, kai išmanieji skaitikliai patikrinti ir pakeisti.

Šio modelio pagrindinis tikslas – stebėti ar teikiama elektra ir apskaičiuota paklausa sutampa. Jei yra neatitikimų, visi skaitikliai, kurie buvo pažymėti kaip nepatikimi naudojantis išmaniojo skaitiklio valdiklio modeliu, yra pažymi, kaip sugedę. Kitais žodžiais, išmaniojo skaitiklio valdiklio modelis atranda neatitikimus, o valdiklio modelis sprendžia ar nurodytus skaitiklius nurašyti, kaip sugedusiais.

2.2.8. Energijos generavimo modelis

Aštuntas modelis yra atsakingas už elektros gaminimą. Šiuo modeliu siekiama atvaizduoti vietinę elektrinę ir jos generuojamas elektros pasiūlos reikšmes.



8 Modelis. Energijos generavimo modelis

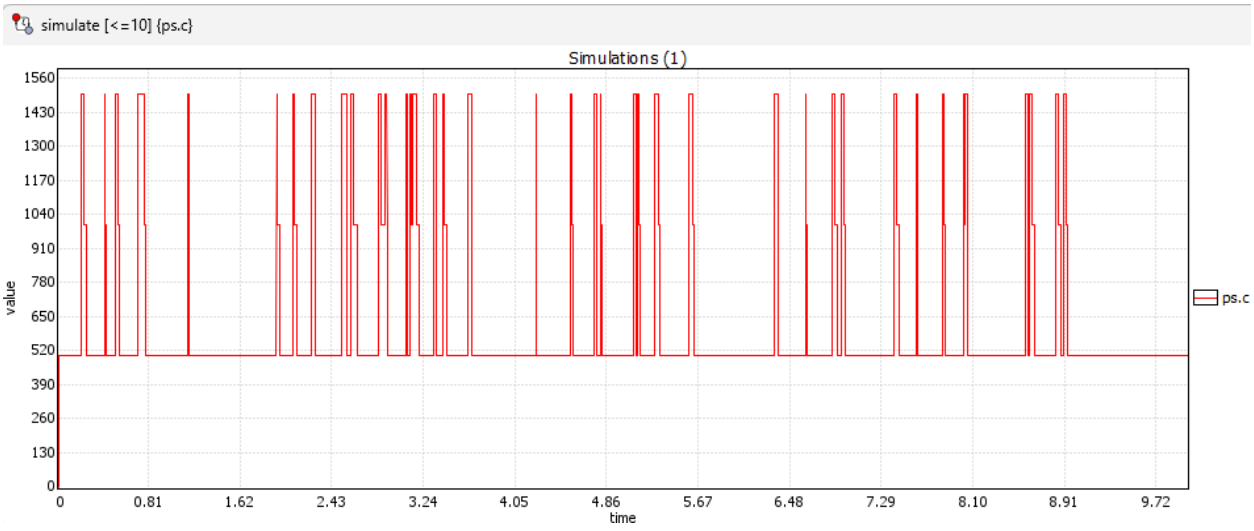
Elektros generatoriaus būsenos:

- Pradinė būsena (angl. *start_power_source*) – modelio pradžia, kuri sinchronizuojasi su laiko modeliu naudojantis t_connect_start transliuojančiu kanalu ir pradeda lygiagretų sistemos darbą.
- Nefunkcionali būsena – ši būsena pasiekiama, kai sistemoje esantis elektros perteklius yra per didelis ir gresia sistemos perkrova. Ši būsena yra susijusi su šia jungtimi:
    - Jungtis, kuri aktyvuojama *increase_power*? kanalu esant sistemos perkrovos nuslopimui ir pereina į minimalią elektros gaminimo būseną. Sistemoje padidėjo elektros poreikis, todėl elektrinė vėl įjungiama.
- Minimalios gamybos būsena – būsena, kurioje elektrinė pagamina mažiausią kiekį energijos. Su šia būsena yra susijusios šios jungtys:

- Jungtis, kuri pereina į nefunkcionalią būseną naudojantis *decrease_power*?, nes yra elektros energijos perteklius sistemoje.
- Jungtis, kuri pereina į vidutinės gamybos būseną naudojantis *increase_power*?, nes yra elektros energijos trūkumas sistemoje.

- Vidutinės gamybos būsena – būsena, kurioje elektrinė pagamina vidutinį kiekį energijos. Su šia būsena yra susijusios šios jungtys:
  - Jungtis, kuri pereina į minimalios gamybos būseną naudojantis *decrease_power*?, nes yra elektros energijos perteklius sistemoje.
  - Jungtis, kuri pereina į aukštos gamybos būseną naudojantis *increase_power*?, nes yra elektros energijos trūkumas sistemoje.
- Aukštos gamybos būsena– būsena, kurioje elektrinė pagamina didžiausią kiekį energijos. Su šia būsena yra susijusios šios jungtys:
  - Jungtis, kuri pereina į vidutinės gamybos būseną naudojantis *decrease_power*?, nes yra elektros energijos perteklius sistemoje.

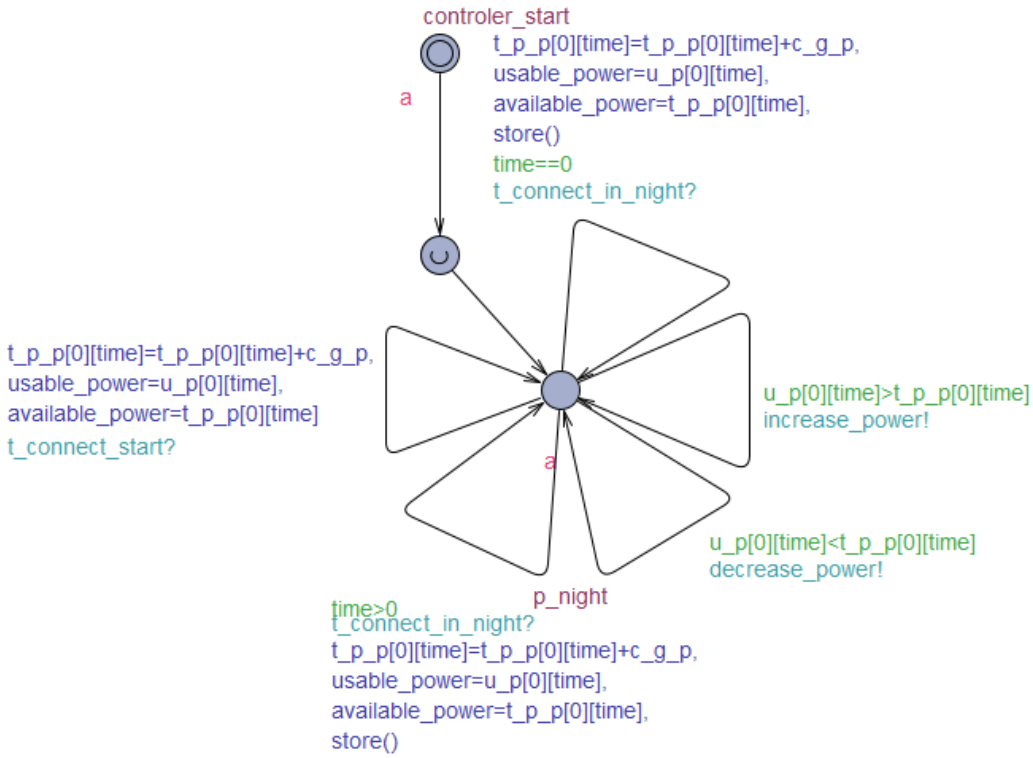
Šio modelio generuojamos reikšmės yra pagaminamos elektros kiekis per valandą, nesinaudojant atsinaujinančiais šaltiniais.



7 Diagrama. Energijos generavimo kontrolės modelio generuojamos elektros reikšmės

2.2.9. Energijos generavimo kontrolės modelis

Energijos kontrolės modelis reaguoja į esamą pasiūlą ir reguliuoja elektros gaminimo kiekį. Kadangi atsinaujinantys elektros šaltiniai nėra patikimi ir visiškai priklauso nuo gamtos jėgų, siekiant patenkinti paklausą, pasiūlos reikšmės atnaujinamos reguliuojant elektros energijos pagaminimo kiekį.



9 Modelis. Energijos generavimo kontrolės modelis

Šio modelio būsenos:

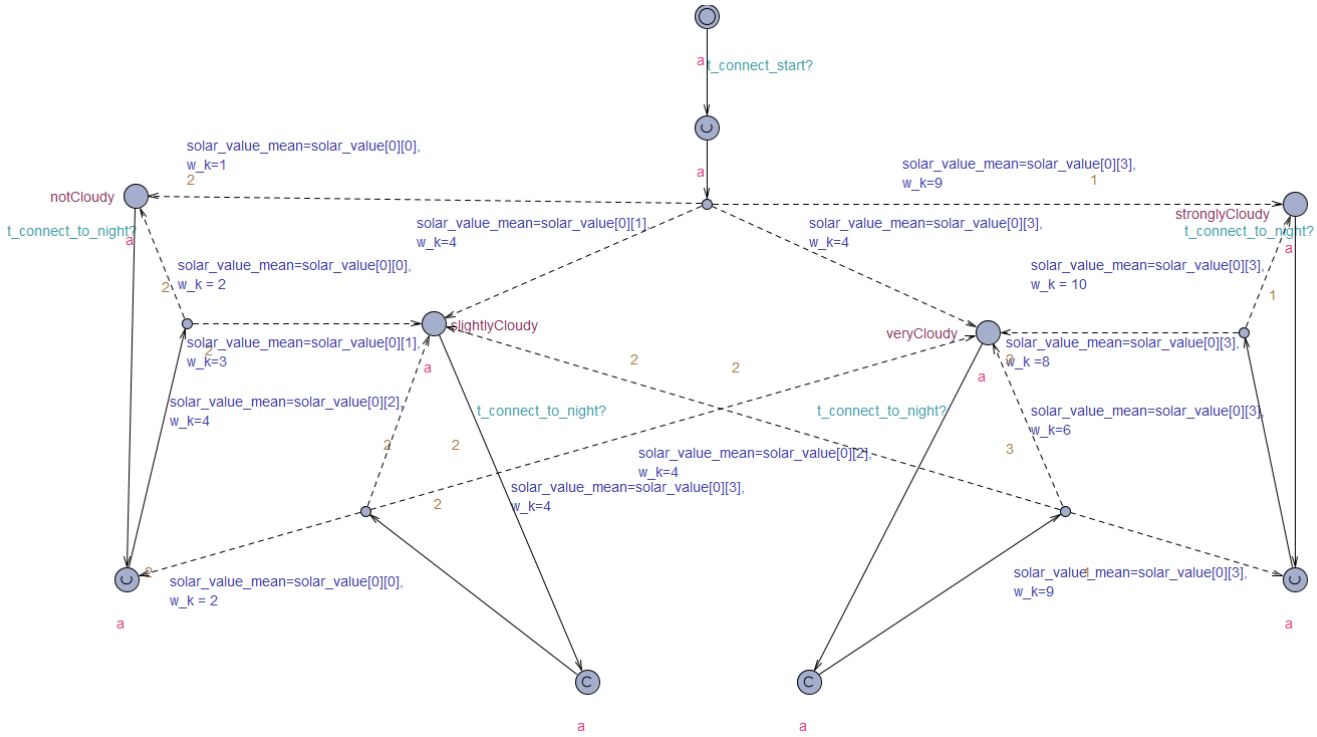
- Pradinė būsena (angl. start_power_source) – modelio pradžia, kuri sinchronizuojasi su laiko modeliu naudojantis t_connect_start transliuojančiu kanalu ir pradeda lygiagretų sistemos darbą.
- Aktyvioji būsena, kurioje tikrinama ar pasiūla nėra per didelė ar per maža, ir jei taip yra, atitinkamai sumažinama arba padidinama elektros gaminama energija. Su šia būsena susijusios šios jungtys:

- Dienos ir nakties jungtys aktyvuojamos t_*connect_in_night?* ir *t_connect_in_day? kanalais*, skaičiuojant bendrą elektros paklausą ir pasiūlą.
- Jungtys, kurios lygina pasiūlos ir paklausos reikšmes, ir atsižvelgus į elektros perteklių ar trūkumą, parenka ar sistema turi padidinti elektros gamybą (*increase_power!)* ar sumažinti elektros pasiūlą (*decrease_power!).*

Šiame modelyje negeneruojama jokių ypatingų reikšmių. Tai yra tik situacijos stebėtojas. Jis stebi elektros kiekį sistemoje ir sprendžia ar reikia didinti ar mažinti gaminamą elektros kiekį.

2.2.10. Oro simuliacijos modelis

Oro simuliacijos modelis simuliuoja aplinkos orą. Pagrindinis kintamasis – debesuotumas. Aptikta priklausomybė tarp debesuotumo, saulės energijos ir vėjo gaminimo, kuri yra išreikšta įvedus atitinkamas tikimybes. Didelis debesuotumas trukdo gaminti saulės energiją, bet paprastai gerai generuoja vėjo energiją



10 Modelis. oro simuliacijos metodas

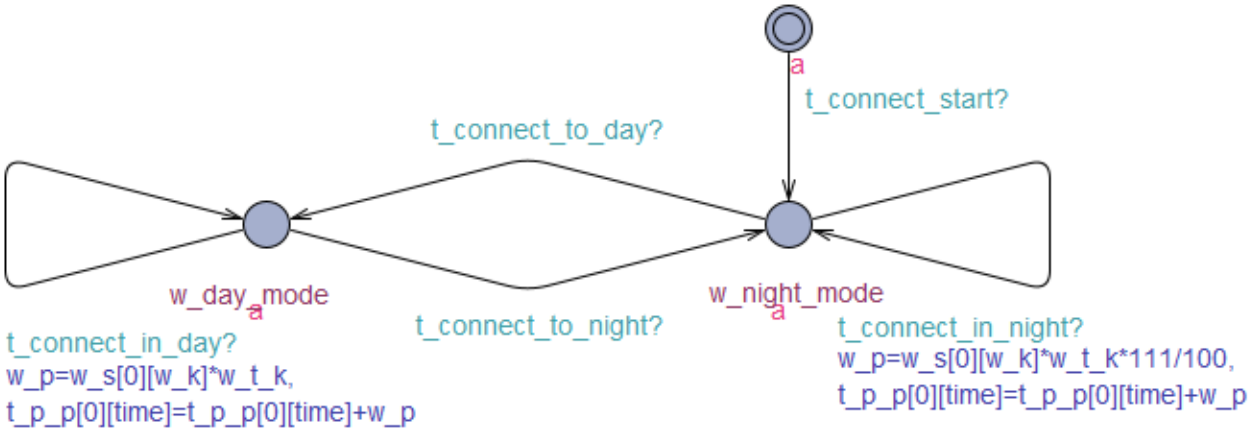
Šiame modelyje svarbiausios keturios būsenos:

- Visiškai nedebesuota;
- Truputi debesuota;
- Labai debesuota;
- Labai stipriai debesuota;

Šiose būsenos daro įtaką vėjo ir saulės elektros gaminimo intensyvumui. Saulės baterija gamins daugiau energijos prie giedro oro.

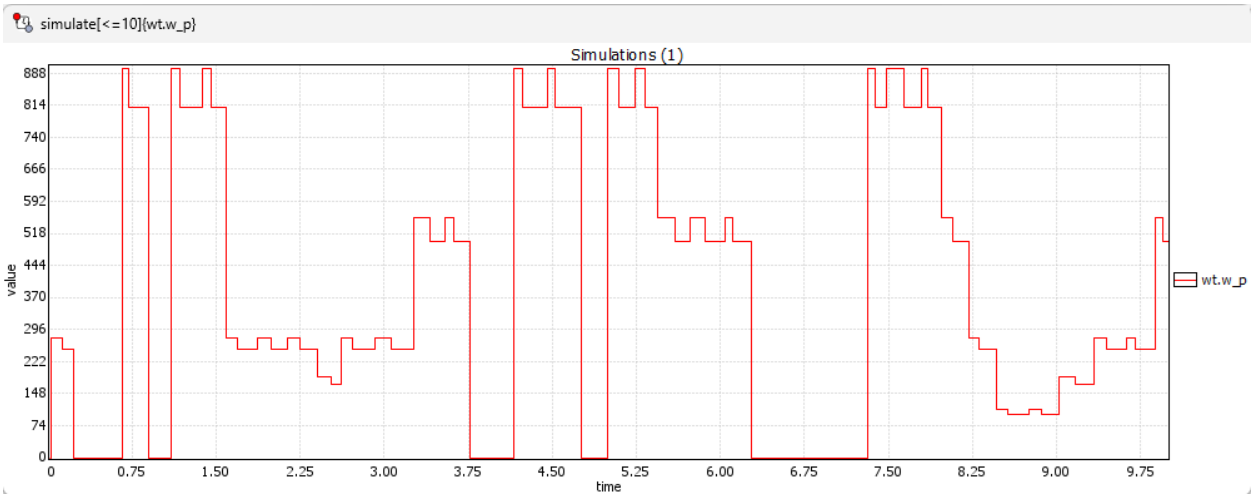
2.2.11. Vėjo jėgainės modelis

Vėjo reikšmės priklauso nuo debesuotumo esančio oro simuliacijos modelyje. Kuo daugiau debesuotumo, tuo didesnės vėjo stiprumo reikšmės.



11 Modelis. Vėjo jėgainės modelis

Vėjo jėgainė turi dvi būsenas: dienos ir nakties būsenos. Abi šios būsenos skaičiuoja pagaminamos elektros reikšmę.

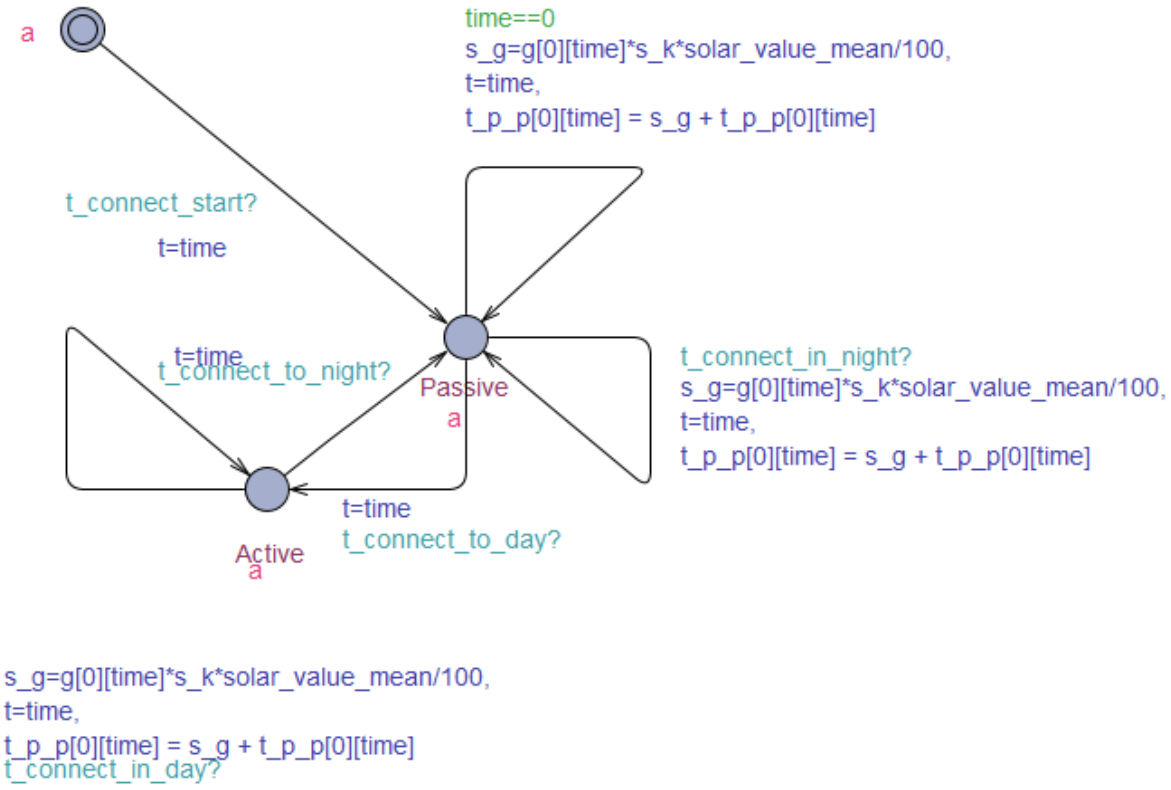


8 Diagrama. vėjo jėgainės reikšmių generacija

Yra pastebėta tendencija tarp generuojamos elektros naudojantis vėjo jėgainėmis ir debesuotumo [35]. Kuo didesnis debesuotumas, tuo didesnė tikimybė, kad vėjo jėgainė pagamins daugiau elektros. Panaši tendencija pastebima tarp vėjo generuojamos elektros ir nakties būsenos. Nakties metu yra didesnė tikimybė, kad vėjo jėgainių generuojama energija bus stabilesnė ir galingesnė, nes vėjas yra stipresnis nakties metu [35].

2.2.12. Saulės baterijų modelis

Saulės baterijos gamina saulės energiją. Taip pat kaip vėjo jėgainės, šių baterijų galingumas priklauso nuo debesuotumo oro simuliacijos modelyje. Kuo mažiau debesų, tuo daugiau saulės energijos yra pagaminama.

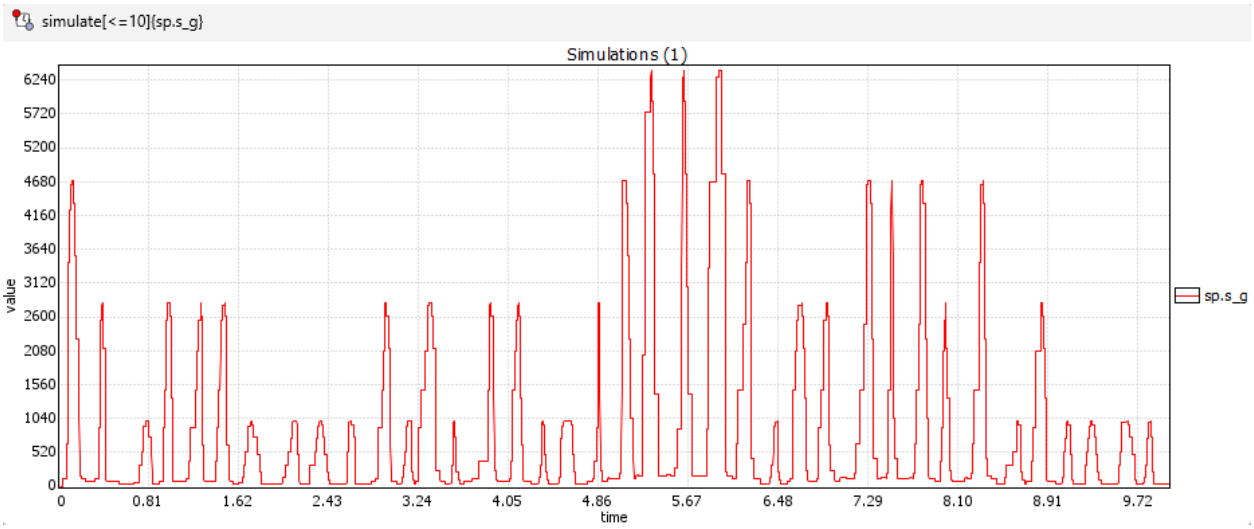


12 Modelis. Saulės baterijų modelis

Šis modelis turi dvi galimas būsenas:

- Aktyvi būsena – ši būsena yra dienos metu, kai galima gaminti saulės energiją. Su šia būsena susijusios šios jungtys:
  - Jungtis, sujungianti aktyvią būsena su pasyvia, naudojantis t_connect_to_night kanalu. tuo atveju, kai ateina naktis (saulės baterią nebegaminama elektros).
  - Jungtis, kuri skaičiuoja pagaminamą saulės energiją dienos metu.
- Pasyvi būsena – nakties būsena, kurioje taip pat vyksta sistemos matavimai, bet gaunamos reikšmės yra labai minimalios (nesugeba patenkinti jokių poreikių).
  - Jungtis, sujungianti pasyvią būsena su aktyvia, naudojantis t_connect_to_day kanalu. tuo atveju, kai ateina diena (saulės baterija pradeda gaminti elektrą).
  - Jungtis, kuri skaičiuoja pagaminamą saulės energiją(labai nedidelius kiekius).

Saulės baterijų generuojama elektros kiekis:



9 Diagrama. saulės jėgainės reikšmių generacija

Svarbu paminėti, kad saulės baterijų modelyje elektros pasiūlos reikšmes atvaizduojantys duomenys (valandinis saulės baterijos generuojamų reikšmių masyvas) yra paimti iš jau atlikto Australijos industrijų departamento eksperimento, renkant realius elektros sunaudojimo duomenis [34].

2.3. Modelių veikimo apibendrinimas

Modeliuojama sistema sudaryta iš 12 modelių. Pirmasis Laiko kontrolės modelis turi dienos ir nakties būsenas, jis naudojamas sinchronizuoti kitų modelių veikimą. Antras modelis yra namo modelis, kuris turi dienos, nakties ir atostogų dienos ir nakties būsenas. Trečias modelis yra gedimų simuliacijos modelis ir jo paskirtis yra simuliuoti gedimus. Yra penkios pagrindinės būsenos, kuriomis remiasi gedimų simuliacijos modelis. Pirmoji būsena yra normali būsena ir tol kol gedimų simuliacijos modelis yra šioje būsenoje, tol išmaniojo skaitiklio modelis, taip pat yra normalioje būsenoje ir nepatiria jokių gedimų. Antra būsena yra atostogų būsena, į kurią patenkama, kai namo modelis pereina į atostogų būseną. Namo modeliui grįžus į dienos ir nakties būsenas, gedimo simuliacijos modelis grįžta į normalią būseną. Kitos trys būsenos (suklastotos atostogos, išsijungimas ir atsitiktinių duomenų būsenos) simuliuoja išmaniojo skaitiklio gedimus ir į jas yra tam tikras nustatytas šansas patekti.

Ketvirtas modelis yra išmaniojo skaitiklio modelis ir jis turi keturias pagrindines būsenas, kurių aktyvumas priklauso nuo gedimo simuliacijos modelio (jei išsijungimo būsena yra aktyvi gedimo simuliacijos modelyje, ji bus aktyvi ir išmaniojo skaitiklio modelyje). Penktas modelis yra gedimo ieškojimo modelis ir jis turi tris būsenas: normali, pažymėta ir sugedusi. Šio modelio tikslas yra ieškoti gedimų ir jei yra randamas neatitikimas, tas modelio objektas pereina į pažymėta būseną. Šeštas modelis yra išmaniojo skaitiklio valdiklio modelis. Šio modelio tikslas yra skaičiuoti ir stebėti namo suvartojimo istoriją. Šis modelis turi tris būsenas (normali, pažymėta ir sugedus) ir į jas pereina, jei gedimo ieškojimo modelis pereina į tą būseną (jei pažymima būsena yra aktyvi gedimo ieškojimo modelyje, ji bus aktyvi ir išmaniojo skaitiklio valdiklio modelyje).

Septintas modelis yra valdiklio modelis, jo tikslas yra stebėti ar sistemoje nedingsta elektra. Jei randami neatitikimai iš būsenos „sistemoje nerasta gedimų“ pereinama į būseną „sistemoje rasti gedimai“, tuo atveju gedimo ieškojimo modelis pereina į sugedusią būseną. Aštuntas modelis yra energijos generavimo modelis, kuris turi tris būsenas (mažo, vidutinio ir aukšto pajėgumo būsena), tarp kurių keičiasi reaguodamas į energijos generavimo kontrolės modelį.

Devintas modelis yra energijos generavimo kontrolės modelis, jis turi tik vieną būseną, kurioje nuolat tikrina ar sistemoje yra energijos perteklius ar trūkumas sistemoje ir liepia energijos generavimo modeliui pasikeisti tarp būsenų. Dešimtas modelis yra oro simuliacijos modelis, nuo kurio priklauso debesuotumas. Šis modelis daro įtaką vėjo ir saulės jėgainių modeliams. Saulės baterijų ir vėjo modeliai turi dienos ir nakties būsenas. Saulės baterijos gamina daugiau elektros dienos metu, o vėjo jėgainės gamina daugiau nakties metu.

Modeliai sinchronizuoja savo aprašomus veiksmus Uppaal kanalų pagalba (žr. Išmaniojo elektros tinklo sistemos komponentai)

2.4. Rezultatai

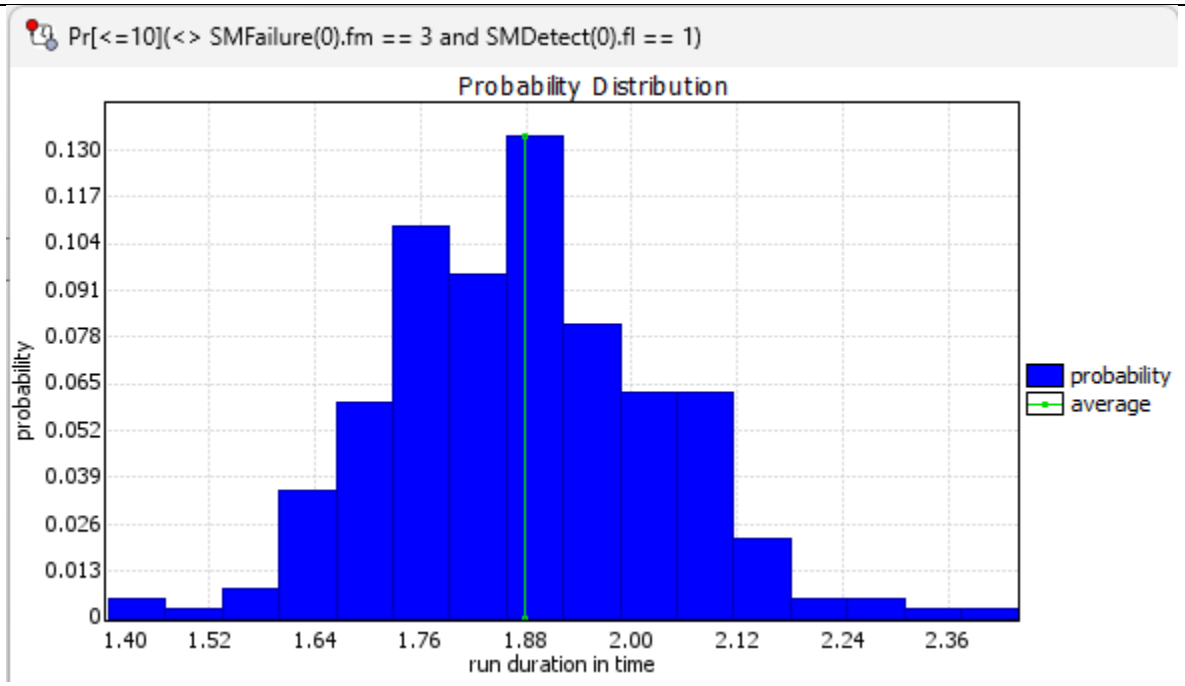
Naudojantis Uppaal užklausų kalba, buvo patikrinti tam tikri sistemos vykdymo scenarijai ir sistemos savybės. Kiekvienas scenarijus aptartas atskirame skyrelyje.

2.4.1. Ryšio nutrūkimas tarp išmaniojo skaitiklio ir vietinio Valdiklio

Šiuo atveju išmanusis skaitiklis nebesiunčia duomenų į bendrąją išmaniųjų tinklų sistemą. Tai reiškia, kad gaunamos nulinės reikšmės iš išmaniojo skaitiklio, o tuo tarpu namas vis dar naudoja tam tikrą kiekį elektros energijos.

Žemiau suformuluota Uppaal užklausa tikrina, kokia yra tikimybė (tiksliau, tikimybinis pasiskirstymas laiko prasme), kad gedimas, kai išmanusis skaitiklis nustoja siųsti reikšmes, bus atrastas naudojantis gedimų radimų modeliu ir bus perkeltas į pažymėtą sąrašą:

```
Pr[<=10](<> SMFailure(0).fm == 3 && SMDetect(0).fl == 1)
```



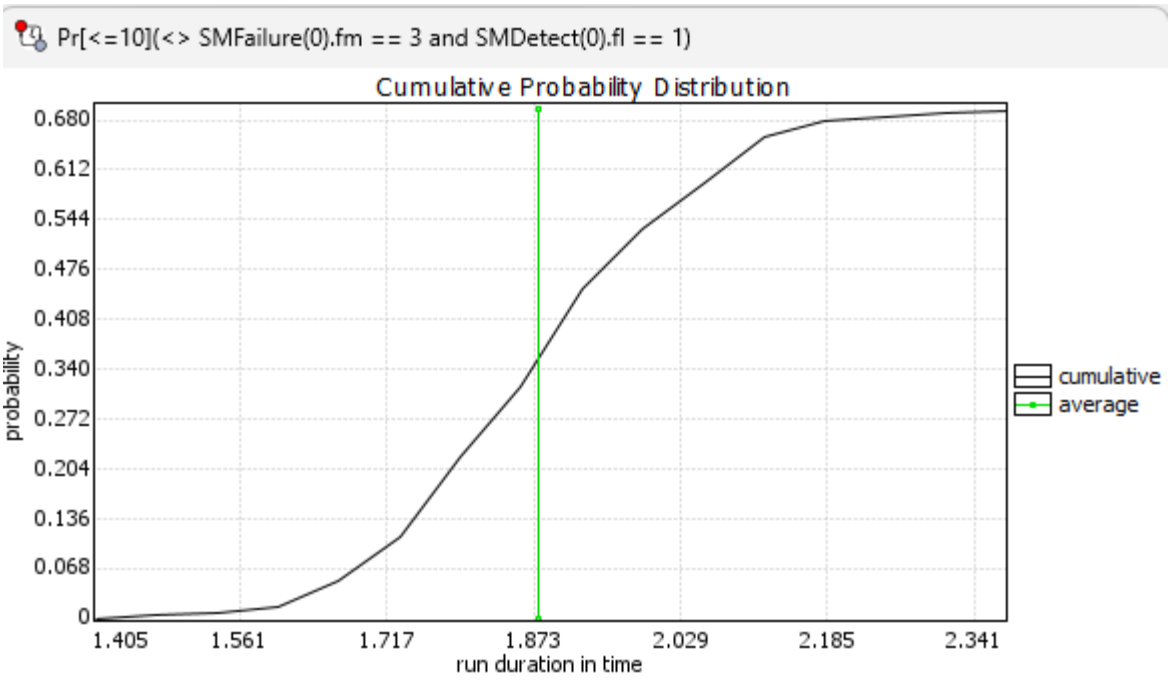
10 Diagrama. Paskirstyta tikimybių diagrama, tikrinanti, kad išmaniojo skaitiklio ryšio nutrūkimas gedimas bus užfiksuotas klaidų ieškojimo sistemoje

Naudodamiesi Uppaal statistinio modeliavimo įrankiu, mes sugeneravome šią diagramą (Paskirstyta tikimybių diagrama, tikrinanti, kad išmaniojo skaitiklio ryšio nutrūkimas gedimas bus užfiksuotas klaidų ieškojimo sistemoje). Joje atvaizduojama, kaip keičiasi nurodytos užklausos tikimybė (y ašis), sistemos simuliacijos vykdymo realiu laiku (x ašis). Kitais žodžiais, sistemos

reakcijos realaus laiko reikšmės yra susiejamos su to tikimybe Diagramoje taip pat matome žalia linija, kuri nurodo sistemos reakcijos laiko vidurkį per visą sistemos paleidimo laiką.

Šiame darbe išsikėlėme tikslą aptikti išmaniojo skaitiklio ryšio nutrūkimą su centrine sistema. Iš gautų rezultatų matome (diagramos), su kokia tikimybe (y ašis) sistemoje gedimo (fm == 3) ir gedimo aptikimo (fl==1) būsena sutampa, ir ieškojimo modelis pažymėjo šį skaitiklį, kaip nepatikimą (jei šios būsenos sutampa, skaitiklis pažymimas kaip nepatikimą).

Uppaal pateikė tikimybę, kad per visą sistemos veikimą, abiejų būsenų vietos bus aktyvios tuo pačiu metu, vidurkis yra 10% (žalia linija). Tai pateikiama diagramoje (10 Diagrama. Paskirstyta tikimybių diagrama, tikrinanti, kad išmaniojo skaitiklio ryšio nutrūkimas gedimas bus užfiksuotas klaidų ieškojimo sistemoje). Ši diagrama parodo tikimybę, kad šios dvi būsenos sutaps ir tikimybė yra tokia maža, nes sistemos veikimo metu modelių būsenos skiriasi (sistema gali užfiksuoti gedimą su kitu gedimo tipu, kurie bus paminėti vėliau). Tačiau ši diagrama parodo gedimo ir radimo dažnumą, įrodant, kad gedimas yra aptinkamas.



11 Diagrama. Kumuliacinė tikimybių diagrama, tikrinanti, kad išmaniojo skaitiklio ryšio nutrūkimas gedimas bus užfiksuotas klaidų ieškojimo sistemoje

Naudodamiesi Uppaal statistinio modeliavimo įrankiu, mes sugeneravome šią diagramą (Kumuliacinė tikimybių diagrama, tikrinanti, kad išmaniojo skaitiklio ryšio nutrūkimas gedimas bus užfiksuotas klaidų ieškojimo sistemoje). Šioje diagramoje matome, kaip tikimybė (y ašis), kad abi būsenos (sistemos gedimo ir gedimo radimo) auga per sistemos veikimo laiką (x ašis) ir koks vidurkis užfiksuotas

Šiame skyriuje yra analizuojama, kokia yra koreliacija tarp paklaidos, kuri naudojama bandant aptikti gedimą, gali skirtis tam tikrą procentinę dalį, ir kaip tai gali paveikti ryšio nutrūkimo tarp išmaniojo ir vietinio Valdiklio aptikimo tikslumo. Ši paklaida naudojama stebint namo elektros energijos sunaudojimo istoriją ir dabartinę elektros energijos sunaudojimo reikšmės skirtumą. Paklaida nurodo kokią procentinę dalį šios reikšmės skirtumas bus toleruojamas, iki kol tai bus pažymėta kaip klaida sistemoje.

Lentelėje pateikiami apibendrinti rezultatai iš (2.4.1 Ryšio nutrūkimas tarp išmaniojo skaitiklio ir vietinio Valdiklio) skyriaus:

paklaida	max	avg	min
5%	0,1	0,1	0,005
15%	0,11	0,08	0,002
25%	0,1	0,06	0,002
50%	0,11	0,01	0,001

1 lentelė Ryšio nutrūkimo tarp išmaniojo skaitiklio ir vietinio Valdiklio rezultatai manipuliuojant paklaidos parametrą

Matome, kad kuo didesnė paklaidos reikšmė, tuo vidutinės ir minimalios reikšmės pradeda kristi, indikuojant, kad gedimas yra sunkiau aptinkamas. Kuo didesnė paklaidos reikšmė, tuo didesnė tikimybė, kad gedimas bus sunkiau ir lėčiau atrandamas.

Šis gedimas yra lengviausiai aptinkamas iš visų gedimų, nes norint jį aptikti tiesiog reikia stebėti ar nenutrūko ryšys tarp skaitiklio ir Valdiklio. Jei ryšys nutrūko, skirtumas tarp sukauptos

istorijos ir gaunamų reikšmių yra palyginus didelis ir labai lengvai aptinkamas. Todėl skirtumas tarp gaunamų reikšmių, keičiant paklaidos reikšmę, yra palyginus mažas.

2.4.2. Tyčiniai gadinimai išmaniųjų jutiklių tinkle

Ši situacija nutinka, kai yra bandoma piktybiškai pakenkti išmaniojo elektros tinklo sistemai, jai paduodant blogus duomenis (Šiuo atveju yra pateikiami klaidingi duomenis, kad namas yra atostogų režime).

Žemiau pateikta formulė tikrina, kokia tikimybė, kad dviejų modelių būsenos (namo ir išmaniojo skaitiklio netikrų atostogų ir gedimo ieškojimo radimo būsena) bus aktyvios vienu metu.

```
Pr[<=100](<> SMFailure(0).fake_vacation and SMDetect(0).flag)
```

Kadangi sistema yra palyginus maža, klaidos radimo koeficientas yra 0.0981446, tai reiškia, kad situacija bus aptikta 98% laiko.

Šiame skyriuje yra analizuojama, kokia yra koreliacija tarp paklaidos, kuri naudojama bandant aptikti gedimą, gali skirtis tam tikrą procentinę dalį, ir kaip tai gali paveikti tyčinių gadinimų išmaniųjų jutiklių tinkle aptikimo tikslumo. Ši paklaida naudojama stebint namo elektros energijos sunaudojimo istoriją ir dabartinę elektros energijos sunaudojimo reikšmės skirtumą. Paklaida nurodo kokią procentinę dalį šios reikšmės skirtumas bus toleruojamas, iki kol tai bus pažymėta kaip klaida sistemoje.

Lentelėje pateikiami apibendrinti rezultatai iš (2.4.2 Tyčiniai gadinimai išmaniųjų jutiklių tinkle) skyriaus:

Namų kiekis	paklaida	max	avg	min
1	5	0,41	0,27	0,0
5	15	0,37	0,24	0,0
10	25	0,27	0,24	0,0
15	50	0,24	0,2	0,0

2 lentelė Tyčinių gadinimų išmaniųjų jutiklių tinkle rezultatų manipuliavimas naudojantis paklaidos parametru

Šioje lentelėje nurodoma, kaip dažnai pasikartoja gedimo ir gedimo radimo būsenos. Matome, kad kuo didesnė paklaidos reikšmė, tuo vidutinės ir maksimalios reikšmės pradeda kristi, indikuojant, kad gedimas yra sunkiau aptinkamas. Kuo didesnė paklaidos reikšmė, tuo didesnė tikimybė, kad gedimas bus sunkiau ir lėčiau atrandamas.

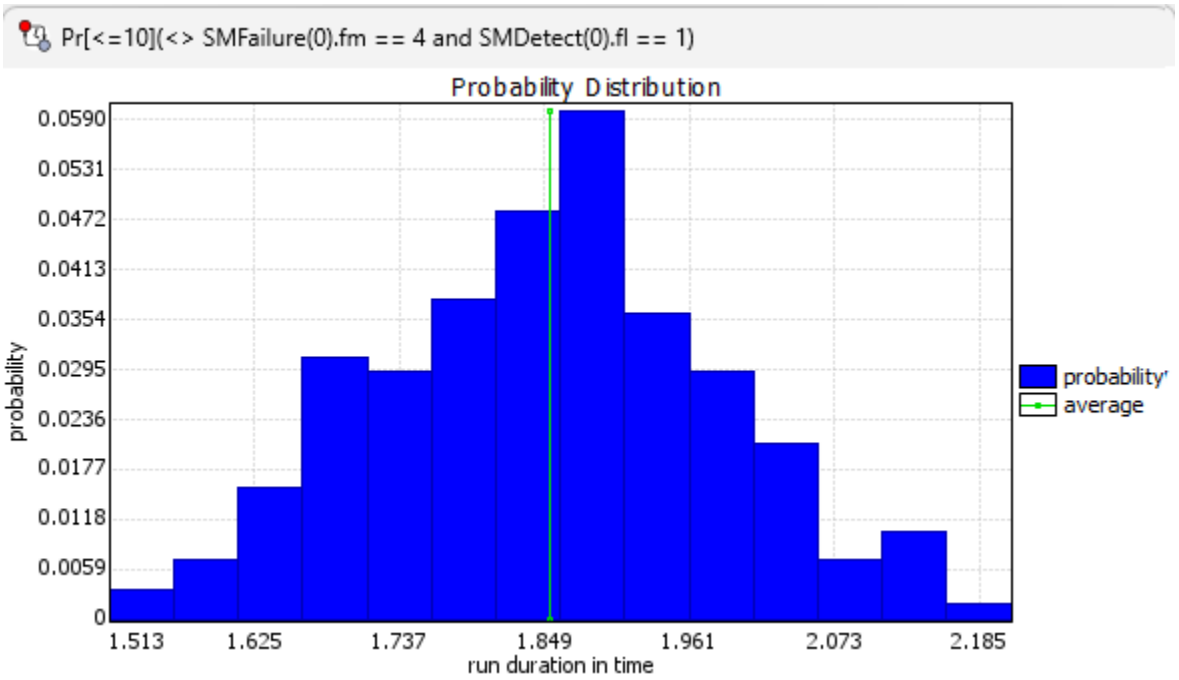
Šis gedimas yra sunkiausiai aptinkamas iš visų gedimų. Tai yra todėl, kad jis yra randamas tik tuo atveju, jei yra pastebimi neatitikimai suvartojamoje elektroje ir išmaniųjų elektros skaitiklių pateiktuose duomenyse. Šioje situacijoje skaitiklis teikia duomenis, kad elektra yra suvartojama, bet tik minimaliomis reikšmėmis (pateikiami duomenys atitinka atostogų režimą, kai namuose nėra elektros vartotojų ir name pajungti elektros prietaisai sunaudoja minimalų kiekį elektros). Sistemoje yra numatytos tokios situacijos, kai namas yra atostogų režime, tad šiuo atveju reikia atskirti tikras situacijas nuo netikrų. Sistema tikrina sunaudojamą elektros kiekį ir duomenis, kuriuos siunčia išmanieji skaitikliai, jei yra neatitikimas, namai esantys atostogų režime, pažymimi, kaip įtartini. Esant neatitikimams tarp sunaudojamo elektros kiekio ir išmaniųjų skaitiklių pateikiamų duomenų, šie skaitikliai yra žymimi kaip sugedę.

2.4.3. Išsinchronizavę matavimai

Žemiau pateikta formulė tikrina, kokia yra tikimybė, kad dviejų modelių būsenos (Gedimo simuliacijos modelio, atsitiktinės reikšmės būsena ir gedimo ieškojimo modelio pažymėjimo būsena) bus aktyvios vienu metu:

```
Pr[<=10](<> SMFailure(0).fm == 4 && SMDetect(0).fl == 1)
```

Naudodamiesi tikimybių pasiskirstymo diagrama (angl. probability distribtion) matome, kaip tikimybė keičiasi sistemos veikimo metu.



12 Diagrama. Paskirstyta tikimybių diagrama, tikrinanti, kad išmaniojo skaitiklio informaciniai nukrypimai, bus užfiksuoti klaidų ieškojimo sistemoje

Šiame darbe išsikėlėme tikslą aptikti iškreiptas išmaniųjų skaitiklių reikšmes. Iš gautų rezultatų matome, kad sistema aptiko gedimą ir gedimo ieškojimo modelis pažymėjo šį skaitiklį, kaip nepatikimą per visas modelio gedimo situacijas. Gedimų atradimas suintensyvėjo vidurį sistemos, vėliau ramiai leidžiantis prie nulio.

Uppaal pateikė tikimybę, kad per visą sistemos veikimą, abiem vietomis būti aktyvioms tuo pačiu metu, vidurkis yra 45% (reikšmė paimta iš 12 Diagrama. Paskirstyta tikimybių diagrama, tikrinanti, kad išmaniojo skaitiklio informaciniai nukrypimai, bus užfiksuoti klaidų ieškojimo sistemoje). Tai reiškia, kad šį gedimą sistemai yra lengva aptikti, bet sistemai yra palyginus sunku pažymėti šį skaitiklį, kaip sugedusią, nes naudojamas algoritmas pažymi skaitiklį kaip sugedus tik tuo atveju, jei yra tam tikras kiekis dingusios elektros kiekis, o šios elektros dingimo reikšmės per mažos, kad atkreipti sistemos dėmesį.

Šiame skyriuje yra analizuojama, kokia yra koreliacija tarp paklaidos, kurią stebint namo elektros energijos sunaudojimo istoriją ir dabartinę elektros energijos sunaudojimo reikšmę bandant aptikti gedimą, gali skirtis nuo tam tikro procentinio dydžio, ir kaip tai gali paveikti išsinchronizavusių matavimų aptikimo tikslumą.

Lentelėje pateikiami apibendrinti rezultatai iš (2.4.3 **Klaida! Nerastas nuorodos šaltinis.**) skyriaus:

paklaida	max	avg	min
5	0,65	0,65	0,004
15	0,55	0,55	0,003
25	0,56	0,53	0,002
50	0,54	0,23	0,001

3 lentelė Išsinchronizavusių matavimų rezultatų manipuliavimas naudojantis paklaidos parametru

Matome, kad kuo didesnė paklaidos reikšmė, tuo minimalios, vidutinės ir maksimalios reikšmės pradeda kristi, indikuojant, kad gedimas yra sunkiau aptinkamas. Kuo didesnė paklaidos reikšmė, tuo didesnė tikimybė, kad gedimas bus sunkiau ir lėčiau atrandamas.

Šis gedimas yra lengviau aptinkamas, nei anksčiau minėtas, nes jam nėra alternatyvios situacijos, kai gedimas gali būti blogai interpretuotas. Šio gedimo aptikimas vyksta stebint suvartojamos elektros kiekį ir skaitiklių raportuojamus duomenis. Jei perduodami duomenys neatitinka sukauptos istorijos, skaitikliai pažymimi, kaip įtartini. Atradus neatitikimus tarp perduodami duomenų ir sunaudojamos elektros, įtartini skaitikliai yra pažymimi kaip sugedę.

2.4.4. Sistemos negebėjimas surinkti tikslios informacijos

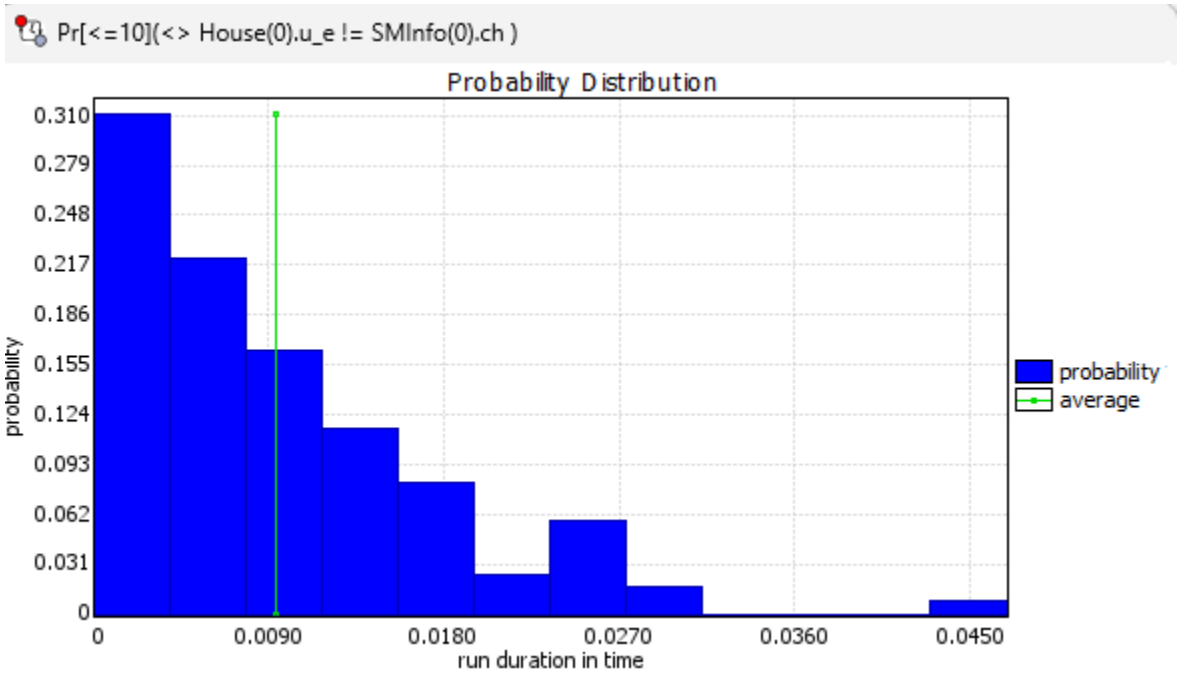
Dėl aukščiau išvardintų gedimų gali būti iškraipoma elektros paklausa. Tai gali sukelti finansinius nuostolius, tad svarbu atrasti metodus, kurie pataiso gedimų iškraipytas reikšmes, kad elektros įmonė galėtų pateikti sąskaitą klientui. Šiuo atveju mes tą padarysime kaupdami elektros sunaudojimo istoriją ir atsižvelgdami į tikimybę, kad šis sukauptas istorijos kiekis atitiks realų vaizdą. Gaunama istorija yra paleidus simuliaciją vienai savaitei be gedimų.

Anksčiau aptartuose gedimuose matėme, kad sistema aptinka gedimą ir jį sėkmingai užregistruoja. Tačiau ką daryti, kai sistema jį užregistruoja. ir turi nuspręsti kiek elektros namas sunaudoja ir kiek elektros bus tiekiamą į sistemą. Tam yra generuojami namo elektros suvartojimo istorija, pagal kurią nuspėjama kiek namas išnaudos elektros. Dėl gedimų išmaniojo skaitiklio reikšmės ne visada bus lygios namo suvartojimo energijai.

Užklausa, kuri randa tikimybę, kad namo elektros sunaudojimas nėra lygus skaitiklio generuojamiems duomenims(t.y. gedimai iškraipė paklausą). Užklausa yra suformuluota, taip:

```
Pr[<=10](<> House(0).u_e != SMInfo(0).ch )
```

Užklausos rezultatas yra atvaizduotas tikimybės diagramoje:



13 Diagrama. Paskirstyta tikimybių diagrama, tikrinanti ar skaitiklio surenkama informacija atitinka sukauptai elektros suvartojimo istorijai.

Iš diagramos matome, kad šios tikimybės vidurkis yra apie 11%.

Ši diagrama (13 Diagrama. Paskirstyta tikimybių diagrama, tikrinanti ar skaitiklio surenkama informacija atitinka sukauptai elektros suvartojimo istorijai.) parodo, kad nors išmanieji skaitikliai perduoda namų ūkio elektros suvartojimą sistemai, jie ne visada yra patikimi. Skaitikliai gali būti nepatikimi, ypač tais atvejais, kai įvyksta gedimas, kuris iškraipo teikiamus duomenis sistemai. Dėl to

matome, diagramoje atvaizduojamas situacijas, kai išmanusis skaitiklis stebėdamas sunaudojamą elektros energiją neatvaizduoja realios situacijos, vietoj to pateikdamas iškraipytas reikšmes.

Dėl šios priežasties. atradus gedimą, generuojamų reikšmių pakeitimas sukaupta istorija apie namo sunaudojamą energiją, pasitvirtino kaip tinkama praktika padedanti išvengti klaidų.

Šiame skyrelyje išanalizuosime skirtingų konfigūracijų rezultatus, kai rezultatas kinta atsižvelgiant į stebimų namų kiekį (N). Lentelėje stebime, kaip kinta tikimybės vidurkis, kad skaitiklyje esanti informacija, nesutaps su realia namo elektros sunaudojimo esančia informacija simuliacijos paleidimo metu.

N	max	avg	min
1	0,27	0,1	0,0
5	0,31	0,17	0,1
10	0,34	0,31	0,06
15	0,37	0,34	0,13

4 lentelė. Sistemos negebėjimo surinkti tikslios informacijos rezultatų manipuliavimas naudojantis namų parametru

Tikimybė, kad išmaniojo skaitiklio ir namo sunaudojama energija vienu momentu sutaps išlieka 90%. Simuliacijos metu matome, kad padažnėja atvejų, kai tikimybė (maksimali, vidutinė ir minimali reikšmė), kad šios reikšmės nesutaps, užkyla, įtraukiant kuo daugiau namų į simuliaciją.

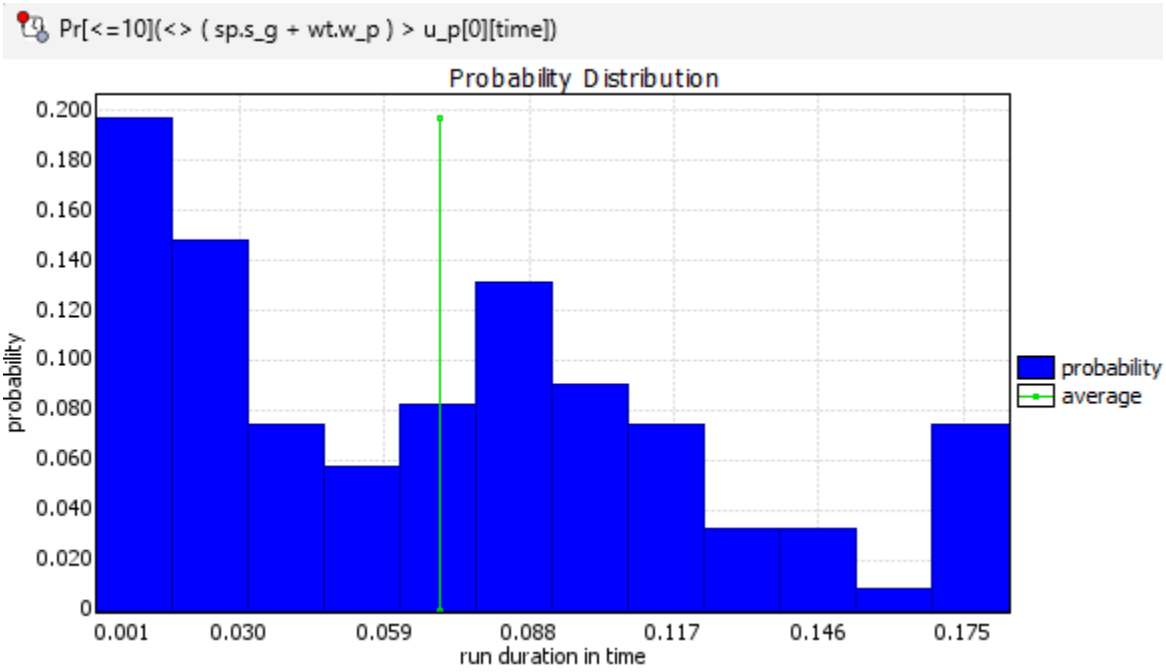
Anksčiau aptarti gedimai yra užfiksuojami tik tuo atveju, jei aptinkami neatitikimai tarp išmaniųjų skaitiklių pateikiamos informacijos ir bendros sunaudojamos elektros. Jų tikslumas priklauso ne tik nuo toleruojamos paklaidos, bet ir nuo kiek namų yra sistemoje. Kuo daugiau yra namų, tuo mažesnis yra gedimo aptikimo greitis. Todėl norint tikslumo, išmaniųjų skaitiklių Valdikliai turi stebėti atitinkamą kiekį namų.

2.4.5. Atsinaujinančių elektros šaltinių nepastovumas

Atsinaujinančių elektros šaltinių nepastovumas, apsunkina išmaniojo elektros tinklo pastangas padaryti kuo lankstesnę sistemą.

Žemiau suformuluota užklausa grąžina tikimybę, kad atsinaujinantys elektros šaltiniai patenkins vartotojų elektros paklausą:

```
Pr[<=10](<> ( sp.s_g + wt.w_p ) > u_p[0][time])
```



14 Diagrama. paskirstyta tikimybių diagrama, tikrinanti, atsinaujinančių šaltinių gebėjimą patenkinti sistemos elektros poreikį

Šiame grafike matome, kad atsinaujinantys elektros šaltiniai gali patenkinti elektros energijos paklausą, bet jų nepastovumas gali palikti sistemą be elektros šaltinio.

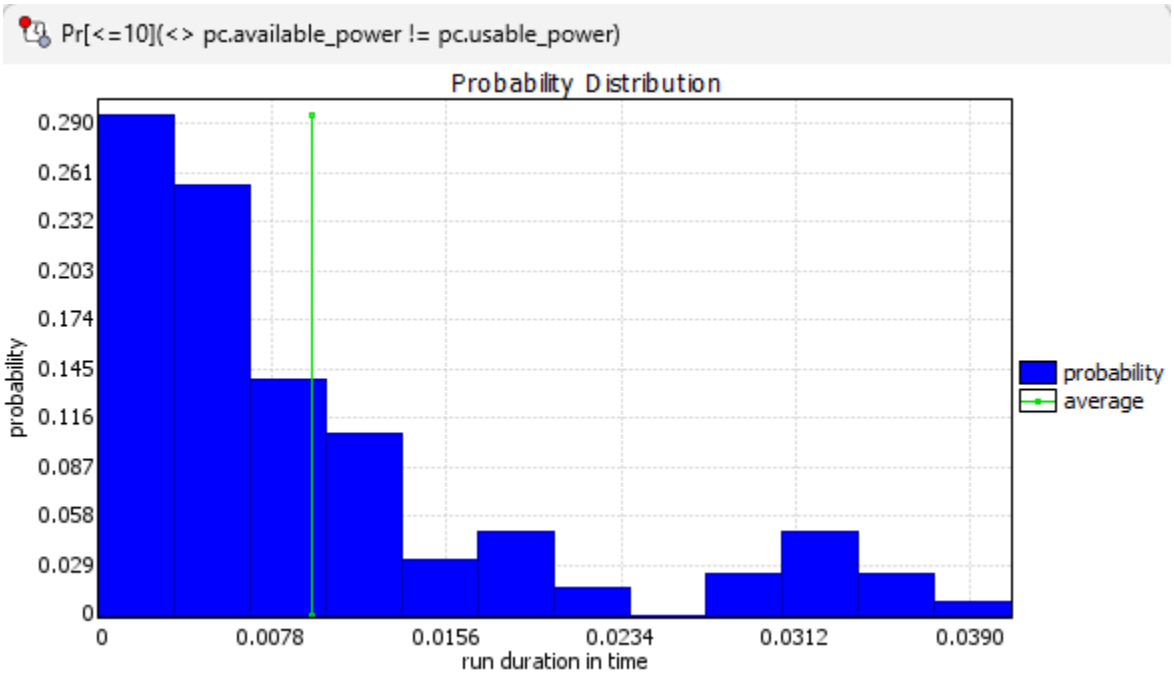
Vidutinė tikimybė, kad sistema sugebės patenkinti vartotojų poreikius yra 16%.

Dėl elektros sistemos trūkumų, yra įjungiama papildoma vietinė elektrinė, kuri padeda suvaldyti skirtumą. Iš to mes galime daryti išvadą, kad integravus atsinaujinančius elektros šaltinius, atsiranda tam tiktas nepastovumo faktorius, dėl kurio reikia lankstumo iš kitų sistemos komponentų.

Tikimybė, kad paklausos ir pasiūlos reikšmės nėra lygios:

```
Pr[<=10](<> pc.Prieiga_power != pc.usable_power)
```

Kaip matome ši reikšmė yra palyginus nedidelė (vidurkis apie 10%, kai kuriais atvejais tikimybė lygi 0):



15 Diagrama. Paskirstyta tikimybių diagrama, tikrinanti ar sistemos pasiūla nesugebės patenkinti paklausos

Užklausos rezultatai parodo, kad vidurkinė tikimybė, kad elektros paklausa bus nelygi pasiūlai yra labai maža (apie 10%), bet net šie mažyčiai nukrypimai rodo būtinybę investuoti į baterijos technologijas siekiant suvaldyti sistemos perkrovą ir į sistemą įtraukti energijos kaupimo įrangą, kuri padėtų valdyti energijos perteklius ir trūkumus.

Šiame skyrelyje analizuosime situaciją, kai netekus elektros tiekimo šaltinio, reikia pasikliauti tik vėjo ir saulės tiekiama elektra ir kokia tikimybė, kad tas poreikis bus patenkinamas.

Saulės ir vėjo pajėgumai priklauso nuo oro svyravimo. Todėl tikimybė, kad elektros paklausa bus patenkinta, svyruoja tarp maksimalios ir minimalios reikšmės sistemos simuliacijos metu. Maksimali ir minimali reikšmė skiriasi priklausant nuo saulės ir vėjo jėgainių kiekio ir santykio esančio sistemoje.

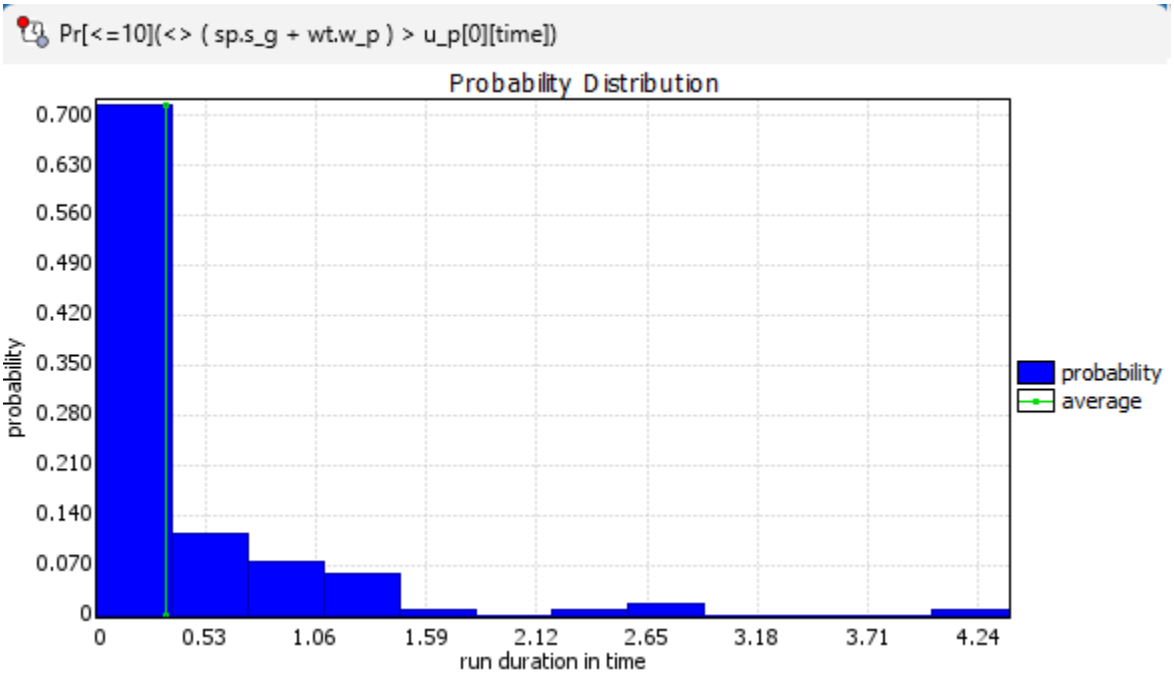
N	Saulės baterijos	Vėjo jėgainės	Max	Min	Avg
1	1	20	0,31	0,0	0,01
1	5	15	0,23	0,0	0,12
1	10	10	0,17	0,0	0,11
1	15	5	0,17	0,0	0,11
1	20	1	0,21	0,0	0,09

5 lentelė Atsinaujinančių elektros šaltinių nepastovumo rezultatų manipuliavimas naudojantis vėjo jėgainių ir saulės baterijų parametrais

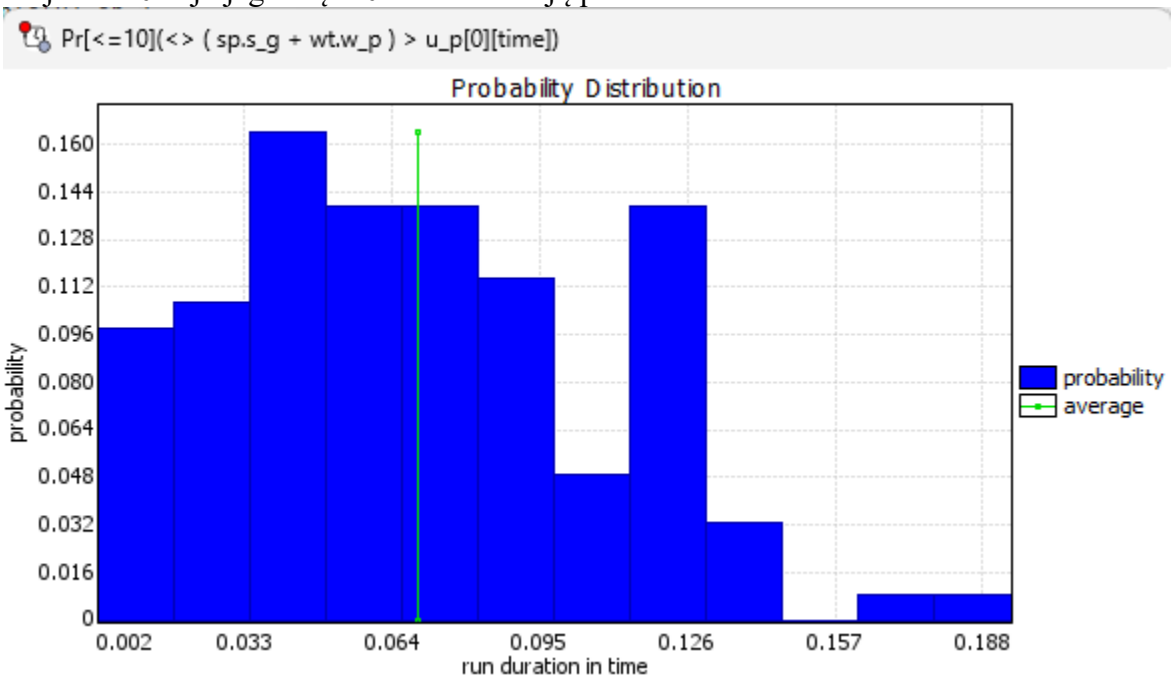
Iš lentelės matome, kad atsinaujinančių elektros šaltinių pajėgumas priklauso nuo saulės ir vėjo jėgainių santykio. Atlikto tyrimo metu matome, kad optimaliausias variantas yra 10 saulės baterijos ir 10 vėjo jėgainių vienam namui.

Kraštutiniai atvejai, kai saulės baterijų yra 20 kartų daugiau nei vėjo jėgainių arba priešingai, kai vėjo jėgainių yra daugiau, rodo efektyvumo kritimą vidurkio atžvilgiu. Iš to galime daryti išvadą, kad norint kuo įmanoma daugiau patenkinti paklausos poreikius, reikalingas tam tikras balansas tarp atsinaujinančių elektros šaltinių jėgainių.

Lyginant (16 Diagrama. Atsinaujinančių elektros šaltinių nepastovumo rezultatų manipuliavimas naudojantis 20 vėjo jėgainių ir 0 saulės baterijų parametrais) ir (17 Diagrama. Atsinaujinančių elektros šaltinių nepastovumo rezultatų manipuliavimas naudojantis 10 vėjo jėgainių ir 10 saulės baterijų parametrais) matome diagramas, kuriose tikrinama ta pati situacija su skirtingais parametrais. Pirmoje diagramoje matome situaciją, kai dominuoja tik vėjo jėgainės, o antroje – situaciją, kai yra tam tikras balansas tarp vėjo ir saulės jėgainių. Antroje diagramoje matomas tam tikras balansas, patvirtinant hipotezę kad balansas tarp vėjo ir saulės jėgainių yra efektyvesnis sprendimas, nei koncentravimasis į vieno tipo šaltinį.



16 Diagrama. Atsinaujinančių elektros šaltinių nepastovumo rezultatų manipuliavimas naudojantis 20 vėjo jėgainių ir 0 saulės baterijų parametrais



17 Diagrama. Atsinaujinančių elektros šaltinių nepastovumo rezultatų manipuliavimas naudojantis 10 vėjo jėgainių ir 10 saulės baterijų parametrais

Rezultatai

Taigi šiame darbe buvo identifikuotos esminės išmanaus elektros tinklo savybės, kartu su gedimais, kurie dažniausiai kelią problemą šių išskirstytų sistemų valdytojams. Tiriamos sistemos savybių ir gedimų analizė leido taip pat identifikuoti standartinius gedimų sprendimus (t.y., sistemos tinkamas reakcijas į gedimus) bei sistemos parametrus, darančius didžiausią įtaką modeliuojamai sistemai (namų kiekis, paklaidos toleravimas, saulės ir vėjo jėgainių kiekis sistemoje).

Buvo sukurtas išmaniojo elektros tinklo prototipas (modelis) Uppaal modelių tikrintojo aplinkoje ir sumodeliuoti kraštutiniai elektros tinklų gedimai, kurie buvo atvaizduoti šiame modelyje. Šie gedimai buvo išanalizuoti Uppaal įrankio aplinkoje naudojantis statistiniais modelių patikrinimo metodais, identifikuotų savybių ir skirtingų parametrų reikšmių atžvilgiu ir pateikti analizės rezultatai. Analizės pagrindu buvo surastos silpnosios sistemos vietos (koncentravimasis į vieną atsinaujinantį elektros šaltinį, stebimų namų perteklius, toleruojamos paklaidos dilema) ir palygintos skirtingos konfigūracijos.

Naudojantis Uppaal įrankiu buvo susidurta su keliais iššūkiais. Vienas iš jų buvo atminties trūkumas. Nors statistinis modelių tikrinimo metodas tikrina tik tam tikrą kiekį būsenų, kad patvirtinti savybę, sukurta sistema buvo per didelė, kad patikrinti tam tikras savybes. Norint su tuo susitvarkyti su tuo, reikėjo sumažinti modelį ir modifikuoti užklausas. Kitas iššūkis su kuriuo buvo susidurta, tai modeliavimo kalbos supratimas. Suprasti ir tikslingai panaudoti Uppaal užklausų kalba reikėjo daug pastangų, nes pateikiamą dokumentaciją buvo sunku suprasti. Nepaisant šių trikdžių, įrankis atliko savo paskirtį ir sistema buvo sumodeliuota, o užklausos buvo pateiktos.

Išvados

Atlikus mokslinių šaltinių analizę, šiame darbe buvo išnagrinėtos kiberfizinės sistemos, o dėmesys buvo sutelktas į jų svarbiausias savybes. Taip pat buvo išsamiai apžvelgtos išmaniojo elektros tinklo sistemos, jų charakteristikos ir dažniausiai pasikartojančios problemos, kurias modeliavome šiame darbe. Toliau buvo tirti ir išskirstytų sistemų verifikavimo metodai, įskaitant modelių patikrinimo būdus, jų pranašumus, palyginta su teoremų įrodymo verifikavimu, taip pat išanalizuota modelių tikrinimo privalumai ir trūkumai, atsižvelgiant į laiko logiką ir laiko automatus. Tada buvo nagrinėtas statistinis modelių tikrinimas, jo privalumai ir trūkumai, taip pat palyginti įrankiai, skirti šiam tikslui. Po to buvo pradėta detaliau analizuoti Uppaal įrankį, įskaitant jo sudėtį, kalbą, pritaikymą statistinio modelio tikrinimui, taip pat įrankio privalumus ir trūkumus. Be to, buvo paminėti darbai, kurie yra panašūs į šį.

Toliau šiame darbe buvo susipažinta su uppaal modelių kūrimo sistema ir patikrinti keli reikalavimai susiję su išmaniųjų tinklų reikalavimais. Giliau apžvelgta sukurti modeliai ir šių modelių tikimybės ištestuotos naudojantis verifikacijos metodais. Ištestuoti naudojami metodai ir jų patikimumas.

Pagrindinė darbe nagrinėjama sritis – gedimai išmaniojoje elektros tinklų sistemoje. Atrinkti 5 dažniausiai pasikartojantys gedimai ir sumodeliuota sistema perkelta į Uppaal išmaniąją aplinką. Šioje aplinkoje atlikta sistemos ir atrinktų gedimų simuliacija, bei sukaupti šių gedimų duomenys. Gedimai buvo išanalizuoti pagal atrinktus kintamuosius ir analizės rezultatai pateikti dokumente.

Pagrindiniai parametrai, pagal kuriuos buvo analizuota sistemos rezultatai – toleruojama gedimo paklaida ir namų skaičius sistemoje. Kuo didesnė paklaida arba kuo didesnis kiekis namų sistemoje, tuo mažesnė tikimybė, kad sistema grąžins tikslius rezultatus. Todėl svarbu atrinkti tokią paklaidą, kuri toleruos greitai reaguos į sistemos klaidas, bet neužskaitys neteisingų klaidų pasikeitus namo elektros vartojime. Taip pat buvo analizuojamos atsinaujinančių šaltinių produktyvumas atsižvelgus į saulės ir vėjo jėgainių kiekio parametrų reikšmės. Tyrimų rezultatai parodė, kad yra reikalingas balansas tarp šių jėgainių norint optimizuoti produktyvumą.

Santrauka

Šio magistrinio darbo tikslas yra sukurti išmaniojo elektros tinklo sistemos modelį (prototipą), identifikuoti jo pagrindines savybes, gedimus ir gedimų sprendimus, išanalizuoti modelį statistinio modelių patikrinimo metodais, naudojant Uppaal įrankio aplinką ir pateikti analizės rezultatus.

Uppaal yra statistinio modeliavimo įrankis skirtas modeliavimui, validacijai ir verifikavimui naudojant realaus laiko sistemas. Naudojantis Uppaal modeliavimo įrankiu buvo sumodeliuoti ir išanalizuoti penki dažniausiai pasikartojantys gedimai. Šie gedimai buvo išanalizuoti naudojantis statistiniais modelių patikrinimo metodais, identifikuotų savybių ir skirtingų parametrų reikšmių atžvilgiu. Analizės rezultatai parodė sistemos silpnąsias vietas, tokias kaip koncentravimasis į vieną atsinaujinantį elektros šaltinį, stebimų namų perteklius ir toleruojamos paklaidos dilemą.

Summary

The aim of this master's thesis is to create a model (prototype) of a smart electric grid system, identify its main features, faults, and fault solutions, analyze the model using statistical model checking methods with the Uppaal tool, and present the analysis results. Uppaal is a statistical modeling tool for simulation, validation, and verification of real-time systems. Using the Uppaal simulation tool, the five most frequent failures were simulated and analyzed. These failures were examined using statistical model checking methods for the identified properties and various parameter values. The results of the analysis highlighted the weak points of the system, such as the reliance on a single renewable electricity source, the excess number of monitored houses, and the tolerable error dilemma.

Šaltinių sąrašas

[1] M. van Steen, A. S. Tanenbaum, Distributed Systems 3rd edition (2017), Pearson Education, Inc, 2017.

[2] K. Lajos, S. Nóra, „Case study in system development - Notes," 2014. [Tinkle]. [Kreiptasi 17 Birželio 2021].

[3] A. Sanghavi, „What is formal verification?," *EE Times-Asia,* 2010.

[4] D. Cox, Principles of Statistical Inference, Cambridge University Press, 2006.

[5] US National Science Foundation, „Cyber-Physical Systems (CPS)," Directorate for Computer & Information Science & Engineering, 11 Kovo 2010. [Tinkle]. Prieiga: https://www.nsf.gov/pubs/2021/nsf21551/nsf21551.htm. [Kreiptasi 19 Birželio 2021].

[6] A. S. Gillis, „internet of things (IoT)," [Tinkle]. Prieiga: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT. [Kreiptasi 19 Birželio 2021].

[7] M. Grindal, Handling Combinatorial Explosion in Software Testing, Linköping, Sweden: Department of Computer and Information Science, Linköpings universitet, 2007.

[8] i-scoop, „Smart grids: what is a smart electrical grid – electricity networks in evolution," i-scoop, [Tinkle]. Prieiga: https://www.i-scoop.eu/industry-4-0/smart-grids-electrical-grid/. [Kreiptasi 29 Birželio 2021].

[9] Cambridge University Press, „Cambridge Advanced Learner's Dictionary & Thesaurus," [Tinkle]. Prieiga: https://dictionary.cambridge.org/dictionary/english/smart-meter. [Kreiptasi 15d. Birželio 2023].

[10] J.Lee, G.Jo, W.Jung, H.Kim, J.Kim, Y.-J.Lee, J.Park, „Chapter 2 - SnuCL: A unified OpenCL framework for heterogeneous clusters," įtraukta *Heterogeneous Computing System,* 2017.

[11] F. Naseri, „Application notes," [Tinkle]. Prieiga: https://www.researchgate.net/profile/Farshid_Naseri/post/How_Bi-directional_Power_Flow_can_be_achieved_in_Smart. [Kreiptasi 15 Birželio 2023].

[12] lietuviuzodynas.lt, „lietuviu zodynas," [Tinkle]. Prieiga:

https://www.lietuviuzodynas.lt/terminai/Verifikacija. [Kreiptasi 11 Kovas 2024].

[13] C. Baier and J. Katoen, Principles of Model Checking, Massachusetts Institute of Technology, 2008.

[14] A. Legay, A. Lukina, L. M. Traonouez, J. Yang, S.A. Smolka, R. Grosu, „Statistical Model Checking," įtraukta *Computing and Software Science*, Paryžius, Springer, Cham, 2019.

[15] Uppsala University, Aalborg University, „UPPAAL documentation," 2021. [Tinkle]. Prieiga: https://docs.uppaal.org/. [Kreiptasi 19 Birželio 2021].

[16] A. Naeem, F. Azam, A. Amjad, M. W. Anwar, „Comparison of Model Checking Tools Using Timed Automata - PRISM and UPPAAL," 18 Gegužė 2018. [Tinkle]. Prieiga: https://ieeexplore.ieee.org/document/8542231/authors#authors. [Kreiptasi 15 Birželio 2023].

[17] S. M. Thampi, „Introduction to Distributed Systems," 23 Lapkričio 2009. [Tinkle]. Prieiga: https://arxiv.org/abs/0911.4395. [Kreiptasi 18 Birželio 2021].

[18] T. Kovácsházy, Distributed architecture for real-time cyber-physical system, time-sensitive networks, IEEE, 2018.

[19] Y. V. Pavan Kumara, Ravikumar Bhimasingu, „ScienceDirect," 8 Gegužės 2015. [Tinkle]. Prieiga: https://core.ac.uk/download/pdf/82305687.pdf. [Kreiptasi 15 Birželio 2023].

[20] S. Karnouskos, „Cyber-Physical Systems in the SmartGrid," Gegužė 2011. [Tinkle]. Prieiga: https://www.researchgate.net/publication/224260726_Cyber-Physical_Systems_in_the_SmartGrid. [Kreiptasi 15 Birželio 2023].

[21] Dr. E. Geisberger, Dr. M. Vi. Cengarle, P. Keil, J. Niehaus, Dr. C. Thiel, H. J. Thönnißen-Fries, Cyber-Physical Systems Driving force for innovation in mobility, health, energy and production acatech POSITION PAPER, Federal Ministry of Education and Research, 2011.

[22] A. Rajeev, Principles of Cyber-Physical Systems, Massachusetts: The MIT Press, 2015.

[23] S. Patil, G. Zhabelova, V. Vyatkin, B. McMillin, „Towards Formal Verification of Smart Grid," *IEEE,* 2015.

[24] D. V. Dollen, „Report to NIST on the Smart Grid Interoperability Standards Roadmap (Contract No.

SB1341-09-CN-0031—Deliverable 10) Post Comment Period Version Document,“ Electric Power Research Institute, 2009.

[25] Z. Krivohlava, S Chren & B. Rossi, „Failure and fault classification for smart grids,“ [Tinkle]. Prieiga: https://energyinformatics.springeropen.com/articles/10.1186/s42162-022-00218-3.

[26] M. Ouimet, „Formal Software Verification: Model Checking and Theorem Proving,“ Massachusetts Institute of Technology, Cambridge, 2008.

[27] „Temporal logic,“ 1 Liepos 1998. [Tinkle]. Prieiga: https://web.archive.org/web/20170430084134/http://www-step.stanford.edu/tutorial/temporal-logic/temporal-logic.html. [Kreiptasi 6 Gruodžio 2021].

[28] P. Bouyer, U. Fahrenberg, K. Larsen, N. Markey, J. Ouaknine, J. Worrell, Model Checking Real-Time System, 2018.

[29] A. Rajeev, Timed Automata.

[30] Uppaal, [Tinkle]. Prieiga: https://uppaal.org/features/#statistical-model-checking. [Kreiptasi 6 Gruodžio 2021].

[31] G. Behrmann, A. David, K. G. Larsen, „Uppaal,“ 28 Lapkričio 2004. [Tinkle]. Prieiga: https://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf. [Kreiptasi 6 Gruodžio 2021].

[32] N. Geuze, „Energy Management in Smart Grids using Timed Automata,“ 19 Kovas 2019. [Tinkle]. Prieiga: https://essay.utwente.nl/77389/1/Geuze_MA_DACS.pdf. [Kreiptasi 19 kovas 2022].

[33] Cong Ma, Jiyuan Guo, Jian Liu, Yu Ren, „Research on the Strategy of Failure Recovery in Smart Grid,“ *Journal of Physics: Conference Series,* 2020.

[34] M. Ambrose, M. James, A. Law, P. Osman and S. White, „The Evaluation of the 5-Star Energy Efficiency Standard for Residential Buildings,“ 2013.

[35] Pim van Dorp, Anda Knol, Stephan R. de Roode, Harm J. J. Jonker, „Impacts of low stratocumulus clouds on offshore wind farm power,“ [Tinkle]. Prieiga: http://www.srderoode.nl/pubs/Wind_Energy_Clouds.pdf.

[36] ISO/IEC/IEEE, Systems and Software Engineering - System Life Cycle Processes, Geneva: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), 2015.

[37] T. Gawron-Deutsch, J. Widder, T. Wien, „Approaching Verification and Validation Challenges in Smart Grids," 2014.

[38] H. Song, D.B. Rawat, C. Brecher, Cyber-Physical Systems, 2017.

[39] A. Rotem-Gal-Oz, „Fallacies of Distributed Computing Explained," Sausio 2008. [Tinkle]. Prieiga: https://www.researchgate.net/publication/322500050_Fallacies_of_Distributed_Computing_Explained. [Kreiptasi 18 Birželio 2021].

[40] J. Y. Halpern, M. Vardi, „Model checking vs theorem proving," 1 Sausis 2000. [Tinkle]. Prieiga: https://www.researchgate.net/publication/2611876_Model_Checking_vs_Theorem_Proving_A_Manifesto#:~:text=Formal%20model%20checking%20aims%20at,(Halpern%20and%20Vardi%201991)%20.. [Kreiptasi 28 Rugpjūtis 2021].

[41] S. Wiesner, C. Gorldt, M. Soeken, K.Thoben, R. Drechsler, „Requirements Engineering for Cyber-Physical Systems," 2014.

[42] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu, „Review on Cyber-physical Systems," 2017.

[43] A. O. Otuoze, M. W.Mustafa, R. M. Larik, „Smart grids security challenges: Classification by sources of threats," 2017.

[44] P. Bulychev, A. David, K. G. Larsen, M. Mikučionis, D. B. Poulsen, A. Legay, Z. Wang, Statistical Model Checking for Priced Timed Automata, Kopenhagen, 2012.

[45] J.Harrison, „Theorem Proving for Verification," Portland, 2008.

[46] A. David, K. G. Larsen, A. Legay, M. Mikučionis & D. B.Poulsen, UPPAAL SMC tutorial, 2015.