Adaptyvių kompiuterinių sistemų formalus modeliavimas ir verifikavimas taikant statistinį modelių patikrinimo metodą

Daniel Daukševič

Darbo vadovas: Prof. Dr. Linas Laibinis

Vilniaus universitetas, Matematikos ir Informatikos fakultetas,
Informatikos institutas, Didlaukio g. 47, LT-08303 Vilnius
daniel.dauksevic@mif.stud.vu.lt

Santrauka. Šiame straipsnyje yra aprašomas tyrimas, kurio metu buvo formaliai verifikuojama adaptyvi kompiuterinė robotų sistema. Sistemos modelis buvo sukurtas ir formaliai verifikuojamas taikant statistinį modelių patikrinimo metodą naudojant UPPAAL SMC įrankį. Darbe yra siūloma apjungti du populiarius sistemos organizavimo modelius – hierarchinį bei saviorganizuojantį. Tyrime yra analizuojamas laiko aspektas – tyrinėjamas sistemos reakcijos greitis, vidutinis užduoties atlikimo laikas. Tyrimo rezultatų pagrindų yra pateikiami sukaupti pastebėjimai ir rekomendacijos verifikuojamos sistemos kūrimui ir tolimesniems tyrimams.

Raktiniai žodžiai: adaptyvios kompiuterinės sistemos, statistinis modelių patikrinimas, formalus verifikavimas, UPPAAL SMC.

1 Įvadas

Moderniais laikais plačiai yra paplitusios kompiuterinės sistemos, gebančios atpažinti bei atitinkamai prisitaikyti prie aplinkos pokyčių arba darbo trikdžių. Tokiose sistemose, vadinamomis adaptyviomis (ang. *adaptive computer-based system*), galima išskirti lygiagrečiai vykdomas dvi veiklas – patį sistemos darbą (tikslo, kuriam sistema buvo sukurta ir eksploatuojama, siekimą) bei situacijos sekimą ir atitinkamų veiksmų (siekiant adaptuoti aplinkos pokyčių potencialią įtaką) atlikimą. Priklausomai nuo sistemų darbo specifikos, jų atliekamų veiksmų ir skaičiavimų korektiškumui gali būti taikomi griežti reikalavimai – būtini įrodymai, jog sistema visada veikia korektiškai, o atliekami skaičiavimai ir veiksmai atitinka numatytus apibrėžimus. Toks fundamentalus sistemos patikrinimas yra galimas formalaus verifikavimo pagalba [1].

DOI: https://doi.org/10.15388/LMITT.2023.2

Formalus verifikavimas (ang. *formal verification*) – algoritmo (kompiuterinės programos arba sistemos) korektiškumo ir atitikimo suformuluotiems reikalavimams įrodymas taikant formalius matematinius metodus. Įrodomos įvairios savybės – pradedant bendromis ir intuityviai suprantamomis charakteristikomis, pvz. aklaviečių (ang. *deadlock*) – situacijų, kai sistema dėl netenkinamų išankstinių sąlygų (ang. *precondition*) negali atlikti jokio veiksmo – nebuvimu, iki unikalių, savitų būtent tiriamąjam objektui ypatybių, apibrėžtų formaliame modelyje. Egzistuoja du pagrindiniai verifikavimo metodai – modelių patikrinimas ir teoremų įrodymas [2].

Šiame tyrime yra remiamasi modelių patikrinimo (ang. *model checking*) metodu – automatizuota tiriamo objekto veikimo analize būsenų perrinkimo ir patikrinimo būdu, atliekama siekiant nustatyti ar yra tenkinami apibrėžti reikalavimai, o darbas yra atliekamas korektiškai (pvz. nėra galimybės patekti į kritinę būseną) [3]. Metodo variacija – statistinis modelių patikrinimas – leidžia verifikuoti sudėtingesnes sistemas ir savybes. Verta pažymėti, jog tai yra pasiekiama pilną modelių patikrinimą pakeičiant tikimybiniu (statistiniu) deramo sistemos veikimo įvertinimu. Nepaisant to, statistinis modelių patikrinimas yra plačiai naudojamas įvairiuose tyrimų srityse, pvz. sistemų biologijoje, automobilių ir aviacijos elektronikoje ir programinės įrangos inžinerijoje [4], [5].

Tyrimui atlikti buvo pasirinkta UPPAAL integruota įrankių aplinka (ang. *integrated tool environment*) skirta realaus laiko (ang. *real-time*) sistemų formaliam modeliavimui, validavimui ir verifikavimui. Įrankis yra pritaikytas sistemoms, kurias galima modeliuoti kaip nedeterministinį procesų rinkinį su baigtine valdymo struktūra (ang. *finite control structure*) ir įprastai yra naudojamas komunikavimo protokolų ir sistemų, kuriose laiko aspektas yra ypatingai svarbus, verifikavimui [6].

Tyrimas susideda iš pasirinktos sistemos analizės (esminių ir papildomų, turinčių įtakos tyrimui, kompiuterinės sistemos reikalavimų bei verifikuotinų savybių ir parametrų identifikavimo), formalaus modeliavimo (objektą atitinkančio modelio kūrimo), formalaus verifikavimo (modelio korektiškumo bei apibrėžtų taisyklių tenkinimo įrodymo taikant formalius metodus – statistinį modelių patikrinimą) ir gautų rezultatų interpretavimo. Tyrimo metu yra analizuojamos adaptyvios kompiuterinės sistemos korektiškumas, duomenų integralumas bei laiko savybės – pilno darbo atlikimo arba komponentės gedimo per nurodytą laiko intervalą tikimybė. Tyrimo metu pasiekti rezultatai pagrindžia siūlomo sistemos organizavimo modelio tinkamumą bei sudaro pagrindą tolimesnei analizei.

2 Tiriamoji sritis

Daugiaagentė (ang. *multi-agent*) sistema – sistema, kurią sudaro tarpusavyje sąveikaujančių autonominių agentų aibė. Agentai turi savo lokalų aplinkos supratimą (vaizdą, ang. *local view*), kuriuo remdamiesi priima sprendimus. Dažnu atveju lokalus vaizdas neprilygsta realiai situacijai – pvz. dėl vėluojančio arba neegzistuojančio (sugedus atitinkamai sistemos komponentei) vaizdo atnaujinimo. Jeigu aplinka yra padalinta į atskiras dalis, o kiekvieno agento veiklos sritis apima tik vieną arba kelias tos aplinkos dalis, agentui užtenka pakankamai tikslaus supratimo, kokia situacija esti tik jam paskirtoje konkrečioje dalyje. Koordinuodami savo veiksmus, agentai paprastai bendrauja (komunikuoja) asinchroniškai.

Saviorganizuojanti daugiaagentė sistema (ang. *self-organising multi-agent system*, *MAS*) – decentralizuotų daugiaagenčių sistemų poaibis, kuriam priklauso sistemos su integruotu mechanizmu, leidžiančiu pakeisti sistemos struktūrą (konfigūraciją) ir darbo režimą. Tokios sistemos yra pilnai autonominės ir geba prisitaikyti prie aplinkos kaitos. Viena iš tokios sistemos komponenčių gali būti vien adaptyvumą užtikrinanti posistemė, nedalyvaujanti tiesiogiai sistemos užduoties sprendime, bet palaikanti adaptyvių sistemos komponenčių veikimą.

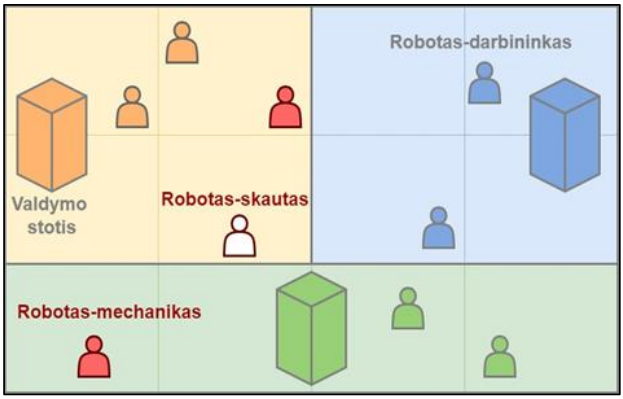
Kritinių sistemų klasei dalinai priskiriamos adaptyvios kompiuterinės sistemos pasižymi distributyvumu, klaidų ir gedimų toleravimu (ang. *fault tolerance*) – darbo tęsimu įvykus vienos (arba kelių) sistemos dalies gedimui (klaidos būsenai), prisitaikymu prie aplinkos pokyčių (ang. *adaptiveness*) bei dinamine rekonfigūracija (ang. *dynamic reconfiguration*) – automatiniu sistemos konfigūracijos (parametrų, nustatymų, struktūros ir t.t.) atnaujinimu realiu laiku, reaguojant į pasikeitusią situaciją.

Pagrindinis ir intuityviai suprantamas reikalavimas, keliamas bet kuriai adaptyviai kompiuterinei sistemai – tikslo pasiekimas nepriklausomai nuo aplinkos kaitos ar gedimų skaičiaus. Sistema turi gebėti tiksliai įvertinti situacijos pokytį bei atitinkamai į jį sureaguoti. Bendru atveju, sugedus pavienei sudėtinei daliai (agentui), adaptyvi kompiuterinė sistema nenutraukia darbo – gedimas yra atpažįstamas bei atitinkamai apdorojamas. Siekiama maksimaliai sumažinti neigiamą įtaką visos sistemos darbo rezultatui, todėl darbas yra paskiriamas iš naujo, o sugedusios komponentės resursai (pvz. susiję su valdymo stotimi robotai) yra paskirstomi tarp aktyvių dalių. Neveiksni dalis, priklausomai nuo jos tipo ir gedimo sudėtingumo, gali būti taisoma. Adaptyvi kompiuterinė sistema atlieka savo tiesioginį darbą bei

reaguoja į pokyčius arba savo sudėtinių dalių gedimus tol, kol gedimai leidžia sistemai fiziškai veikti be išorinio įsikišimo.

Šiame darbe formaliai modeliuojama ir verifikuojama teritoriją valanti robotų sistema. Adaptyvią daugiaagenčią kompiuterinę sistemą, kurios schema yra pateikiama 1 pav., sudaro keturių tipų agentai:

- Valdymo stotis – aukščiausio hierarchijos sluoksnio statinė komponentė. Kiekviena stotis turi aibę robotų-darbininkų, priklausančių tik jai. Taip pat, visos valdymo stotys turi nuosavą, lokalų, teritorijos ir situacijos įsivaizdavimą, kurio besiremdamos priima sprendimus;
- Robotas-darbininkas – judrus mechaninis aparatas. Kiekvienas robotas-darbininkas yra priskirtas vienai konkrečiai valdymo stočiai, iš kurios gauna darbą (sektoriaus kurį reikia išvalyti identifikacinį numerį). Gavęs užduotį, robotas-darbininkas vyksta į nurodytą sektorių ir ten atlieka valymą. Pilnai atlikęs darbą robotas-darbininkas siunčia pranešimą valdymo stočiai, su kuria yra susietas;
- Robotas-mechanikas – taisantis sugedusius robotus-darbininkus bei valdymo stotis mechanizmas. Sugedus robotui-darbininkui arba valdymo stočiai ir tai pastebėjus skautui, robotas-mechanikas gauna pranešimą. Gavęs pranešimą, robotas-mechanikas vyksta į nurodytą lokaciją bei taiso sugedusią komponentę;
- Robotas-skautas – žvalgo rolę atliekantis aparatas. Robotas tyrinėja teritoriją, atsitiktinai pasirinkdamas tikslą – komponentę, kurios būseną tikrins. Pastebėjęs roboto-darbininko arba valdymo stoties gedimą, robotas-skautas apie tai praneša robotui-medikui.



1 pav. Adaptyvios teritoriją valančios kompiuterinės sistemos schema.

Tiriamos adaptyvios kompiuterinės sistemos architektūrą sudaro dviejų organizavimo modelių – hierarchinio ir saviorganizuojančio – hibridas. Sistemoje egzistuoja hierarchija, tačiau ji tik dalinai griežta; dalis komponenčių hierarchijai nepriklausys visiškai ir yra sistemos kontekste autonomiški (sprendimus priima patys, savo nuožiūra).

Hierarchinės architektūros tipo sistemose kiekvienos komponentės pareigos yra aiškiai apibrėžtos. Nurodoma, kuri sistemos dalis priima sprendimus, priskiria užduotį kitai posistemei, o kuri – besąlygiškai atlieka gautus nurodymus ir praneša apie jų atlikimą.

Saviorganizuojančios architektūros sistemose jas sudarantys agentai (komponentės) veiksmų bei pareigų prasme yra tarpusavyje lygūs. Agentai negali nurodyti kitiems jų elgesio, o tik pasidalinti turima informacija. Remdamasis turimomis žiniomis bei nurodytomis taisyklėmis, agentas pats priima sprendimą, kuris tą akimirką atrodo teisingiausias. Vienas iš aktyviai tiriamų tokios architektūros modelių yra „ieškančių maisto skruzdžių“ (ang. *foraging ants*) modelis [7], [8].

Tyrime [9] buvo formaliai verifikuojama eismo greitkelyje stebėjimo sistema susidedanti iš išmaniųjų kamerų. Kameros yra tolygiai paskirstytos kelyje tam, kad būtų aprėptas maksimalus kelio plotas, ir turi aptikti eismo spūstis bei apie tai pranešti šviesoforų kontrolieriams ir vairuotojo pagalbos sistemoms. Esant didelei eismo spūsčiai (pastebimai daugiau nei vienai kamerai), ją filmuojančios kameros jungiasi į vieną bendrą organizaciją, kuri sudaro detalų spūsties apibrėžimą. Transporto priemonėms pajudėjus (nustojus egzistuoti spūsčiai) kameros atsiskiria nuo organizacijos ir tęsia darbą atskirai. Tokiu būdu dinamiškai yra kuriama trumpalaikė minimali hierarchija tarp kelių sistemos komponenčių.

Teritoriją valančios robotų sistemos idėja remiasi anksčiau atliktais tyrimais, kuriose panašios specifikos sistema buvo formaliai verifikuojama tiriant tikimybines charakteristikas PRISM modelių patikrintoju [10] bei funkcines savybes Event-B [11] ir TLA^+ [12] įrankiais. Tiriama sistema pasižymi nevienalyte (ang. *heterogenous*) architektūra – ją sudaro dviejų tipų agentai – stacionarios valdymo stotys bei judrūs robotai. Abiejų tipų robotai gali sugesti, todėl sistema turi būti pasiruošusi, įvykus gedimui, atlikti dinaminį rekonfigūravimą. Vienoje pakopoje esantys robotai yra tarpusavyje lygūs – todėl sistema neturi vieno prižiūrėtojo (ang. *supervisor*). Sistemos darbas yra koordinuojamas tarp valdymo stočių, joms tarpusavyje nuolat bendraujant ir besidalinant savo lokaliu situacijos vaizdu. Šiame tyrime ana-

lizuojamą sistemą galima laikyti nauja teritoriją valančių robotų sistemos versija, kadangi ji pasižymi hibridine architektūra bei yra papildyta naujomis komponentėmis.

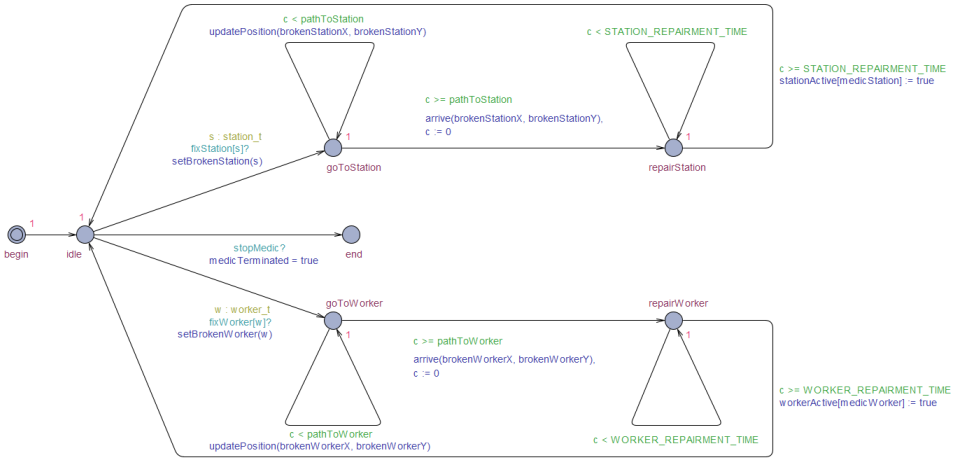
3 Formalus modeliavimas

Šiame mokslo tiriamajame darbe nėra iškeliamas tikslas pilnai formaliai verifikuoti konkrečią adaptyvią kompiuterinę sistemą, bet atlikti tyrimą, kuriuo metu ištirti bendras tokių sistemų aibei savybes ir reikalavimus, apjungti skirtingas architektūrines strategijas bei įvertinti junginio pranašumus ir trūkumus. Sulyg kiekviena iteracija kuriamas modelis bus tikslinamas siekiant tyrimo pabaigoje gauti detalų, bet priklausantį atitinkamai klasei adaptyvių kompiuterinių sistemų, modelį.

Teorinis pagrindas, dažnai naudojamas realaus laiko sistemų modeliavimui ir verifikavimui, yra laiko automatų teorija. Laiko automatas (ang. *timed automata*) – tai baigtinis elementų rinkinys, susidedantis iš padėčių (ang. *locations*), laikmačių (ang. *clocks*), veiksmų (ang. *actions*) [13]. UPPAAL modelis yra lygiagrečiai vykstančių procesų aibė. Kiekvienas procesas yra šablonas (ang. *template*) – laiko automatas, grafiškai vaizduojamas kaip grafas, kurio viršūnės yra lokacijos su jas jungiančiomis briaunomis. Briaunoms yra priskiriami: saugai (ang. *guard*) – loginės formulės, nuo kurių priklauso perėjimo briauna galimybė, modelio kintamųjų atnaujinimai, sinchronizacijos –dviejų arba daugiau procesų koordinavimas bei pasirinkimai (ang. *selections*) – nedeterministinis reikšmių iš nurodyto intervalo arba aibės priskyrimas atitinkamiems kintamiesiems [14].

Teritoriją valančios robotų sistemos modelyje buvo išskirti ir sumodeliuoti septyni šablonai, atvaizduojantys tiek fizines sistemos komponentes (valdymo stotis, robotus-darbininkus, robotus-žvalgus, robotus-mechanikus – šablonas yra pateikiamas 2 pav.), tiek pagrindinius procesus (resursų valdymą, komunikaciją, gedimus). Modeliuojant vieną pagrindinių keliamų sistemai reikalavimų – sėkmingą darbo atlikimą nepriklausomai nuo aplinkos kaitos – galima apibrėžti kitaip – kiekvienas sistemos veikimo scenarijus (vykdymas) galiausiai pasiekia būseną, žyminčią pilną užduoties atlikimą.

Sektorių užterštumas yra modeliuojamas skaitinės vertės masyvu. Kiekvienas masyvo elementas žymi konkretų sektorių, reikšmė nurodo kiek laiko vienetų užimtų valymo darbai. Elementui pasiekus nulinę reikšmę, atitinkantis sektorius yra laikomas išvalytu.



2 pav. Adaptyvios teritoriją valančios kompiuterinės sistemos roboto-mechaniko komponentės šablonas.

4 Formalus verifikavimas

Formalaus verifikavimo tyrimo dalį galima išskaidyti į dvi dalis – atliktą taikant tradicinį modelių patikrinimo metodą ir statistinį. Pirmoje tyrimo dalyje, verifikuojant pradinę sistemos versiją, buvo siekiama įrodyti, jog adaptyvios kompiuterinės sistemos modelio pagrindas atitinka sistemai keliamus reikalavimus – visada sėkmingai užbaigia darbą atitinkamoje būsenoje. Pradinis modelis buvo plečiamas bei verifikuojamas taikant statistinį modelių patikrinimą siekiant gauti ir išanalizuoti tikimybinius sistemos savybių įvertinimus.

4.1 Pilno darbo atlikimo patikrinimas

Pirmoje adaptyvios kompiuterinės sistemos formalaus verifikavimo dalyje buvo modeliuojama pradinė sistemos versija. Jos struktūrą sudarė dvi valdymo stotys, keturi robotai-darbininkai ir skambučio mechanizmas. Taikant modelių patikrinimo metodą, pradinė adaptyvios kompiuterinės sistemos modelio versija buvo formaliai verifikuojama. Siekiama įrodyti savybė buvo apibrėžta:

A[] (deadlock imply (station_0.end and station_1.end))

„Visada aklavietės pasiekimas reiškia abiejų valdymo stočių buvimą darbo užbaigimo būsenoje“. Savybės patenkinimas buvo sėkmingai įrodytas, kas leidžia teigti jog sistema nustoja veikusi (nebeatlieka jokių veiksmų) tik tada, kai pereina į darbo užbaigimo būseną.

4.2 Valdymo stočių lokalių vaizdų korektiškumo patikrinimas

Ypatingai svarbia analizuojamos adaptyvios kompiuterinės sistemos savybe taip pat yra duomenų integralumas. Korektiško sistemos darbo reikalavimą galima išreikšti taisykle jog valdymo stočių lokalaus vaizdas būtinai turi atitikti realią situaciją:

A[] (forall(s : station_t) forall(x : sector_t)
(localView[s][x] imply sectorClean[x]))

„Visada, jei pagal bet kurios valdymo stoties lokalų įsivaizdavimą pasirinktas sektorius yra išvalytas, tas sektorius iš tikrųjų turi būti išvalytas“. Situacija, kai pagal kurios nors valdymo stoties lokalų vaizdą sektorius yra išvalytas, nors iš tikrųjų toks nėra, yra laikoma grubia ir fundamentalia sistemos klaida. Sėkmingai įrodžius savybę galima teigti, jog pirminis adaptyvios kompiuterinės sistemos modelis veikia korektiškai.

Įrodžius pradinės sistemos modelio korektiškumą, modelis buvo tobulinamas. Sistemos konfigūracija buvo atnaujinta; ją nuo šiol sudaro 4 valdymo stotys, 12 robotų-darbininkų, 25 sektoriai, po vieną robotą-mechaniką ir robotą-skautą.

4.3 Tikimybinis sistemos pilno darbo atlikimo įvertinimas

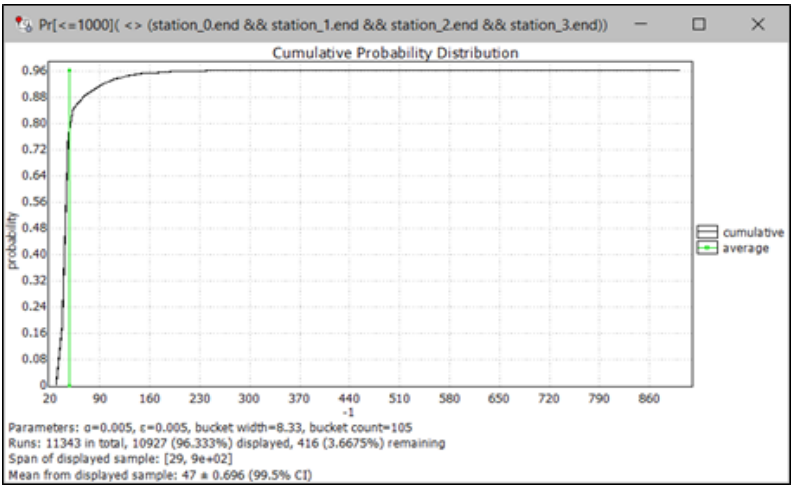
Naudojant UPPAAL SMC buvo siekiama įvertinti, su kokia tikimybe naujoje adaptyvios kompiuterinės sistemos modelio versijoje (konfigūracijoje) darbas bus užbaigtas pilnai atlikus užduotį (išvalius visą teritoriją) – t.y. kokia tikimybė, jog po 1000 laiko vienetų visos valdymo stotys bus darbo užbaigimo būsenoje *end*:

Pr[<=1000](<>(station_0.end && station_1.end &&
station_2.end && station_3.end))

Siekiant tikslesnių rezultatų, buvo modifikuoti skaičiavimų parametrai – tai padidino analizuojamų atvejų skaičių. Tai leido gauti tikslius rezultatus – su pasitikėjimo (ang. *confidence*) koeficientu lygiu 0,995. Iš sugeneruotos kumuliatyvaus tikimybių pasiskirstymo diagramos, pateikiamos 3 pav., galima

pastebėti, jog ties 130 laiko vienetu pasiekiama 0,96 sėkmingo darbo atlikimo tikimybė, kuri nežymiai didėja ilgėjant laiko intervalui. Greitas plynaukštės (ang. *plateau*) – taško, nuo kurio tikimybė didėja nežymiai – pasiekimas yra stabilios (prognozuojamos, neturinčios netikėtų funkcionalumo sutrikimų) sistemos požymis. Svarbu pabrėžti, jog diagramoje yra atvaizduojami 96,333% visų išanalizuotų atvejų (10927 iš 11343). Likę 416 truko ilgiau nei 900 laiko vienetai – ir dalis truko ilgiau nei 1000 laiko vienetų, todėl bendras tikimybinis savybės patenkinimo įvertinimas nesiekė 1.

Tikimybė, jog praėjus 1000 laiko vienetų nuo sistemos darbo pradžios visos valdymo stotys bus darbo užbaigimo būsenoje *end* buvo įvertinta tarp 0,959333 ir 0,969332. Tai, jog viršutinis tikimybės rėžis nesiekia 1, paaiškinamas faktu, jog 10980 patikrinimų metu buvo rastas vykdymo scenarijus, trukęs ilgiau nei 1000 laiko vienetų – toks scenarijus įmanomas pvz. valdymo stotims nuolat gendant. Pasiekta tikimybė yra pakankama, kad galima būtų teigti jog savybė yra patenkinama.



3 pav. Pilno sistemos darbo atlikimo kumuliatyvaus tikimybių pasiskirstymo diagrama.

4.4 Tikimybinis valdymo stočių gedimo galimybės įvertinimas

Tęsiant tyrimą buvo siekiama tyrinėti kitas sistemos savybes. Vienu iš tyrimo tikslų buvo pasirinkta tikimybė, jog per 1000 laiko vienetų (ankstesnis patikrinimas įrodė, jog tokio laiko intervalo užtenka norint analizuoti sistemos korektiškumą) bent vienas iš robotų-darbininkų suges. Užklausa:

Pr(<>[0, 1000] ([] [0,1] (worker_0.error or worker_1.error or worker_2.error or worker_3.error or worker_4.error or worker_5.error or worker_6.error or worker_7.error or worker_8.error or worker_9.error or worker_10.error or worker_11.error)))

leidžia įvertinti, jog per 1000 laiko vienetų bent vieno roboto-darbininko gedimo tikimybė tarp 0,294173 ir 0,394173 – t.y. apie kas trečią teritorijos valymą, kurį atlieka sistema, turėtų būti susidurta su roboto-darbininko gedimu.

5 Išvados

Verifikuojant pilną sistemos versiją buvo gauti sistemos pilno darbo atlikimo ir komponenčių gedimo tikimybiniai įvertinimai. Gauti rezultatai leidžia teigti jog dabartinis sistemos modelis atitinka iki šiol keliamus reikalavimus ir yra stabilus, t.y. prognozuojamas.

Remiantis atlikta mokslinės literatūros analize buvo pasiūlyta hibridinė daugiaagentinės sistemos architektūra, apjungianti hierarchinį ir saviorganizuojantį modelius. Taip pat buvo apibūdinti pagrindiniai reikalavimai, keliami tokio tipo sistemoms, charakteristikos, kuriomis jo pasižymi bei verifikuotinos savybės, kurios yra ir bus tiriamos. Pagrindinėmis sistemos charakteristikomis yra identifikuojami: distributyvumas, klaidų toleravimas bei dinaminė rekonfigūracija. Bendrai tokio tipo sistemoms keliami reikalavimai yra pilnas darbo atlikimas bei gebėjimas palaikyti sklandų darbą įvykus gedimams.

Atlikto tyrimo rezultatai leidžia teigti, jog hibridinės architektūros adaptyvios kompiuterinės sistemos modelis atitinka pagrindinius keliamus reikalavimus. Nepaisant naudojamo formalaus verifikavimo metodo, statistinio modelių patikrinimo, trūkumo – tikimybinis sistemos veikimo korektiškumo įvertinimas nėra pilnas – atlikta skaitinė sistemos charakteristikos analizė rodo rezultatus, pakankamus teigti jog sistema tenkina jai keliamus reikalavimus ir pasižymi reikiamomis savybėmis, kas sudaro pagrindą tolimesniems sistemos tyrimui ir vystymui. Tyrimas yra tęsiamas gilinantis į adaptyvios kompiuterinės sistemos laiko savybes (pvz. vidutinį reakcijos į roboto gedimą laiką), įvertinant įvairių sistemos konfigūracijos parametrų įtaką jos darbui.

Literatūra

[1] Hinchey, M., & Coyle, L. (2010, March). Evolving critical systems: A research agenda for computer-based systems. In 2010 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems (pp. 430-435). IEEE. DOI: 10.1109/ECBS.2010.56.

[2] Ouimet, M., & Lundqvist, K. (2007). Formal software verification: Model checking and theorem proving. Embedded Systems Laboratory Technical Report ESL-TIK-00214, Cambridge, USA.

[3] Baier C., Katoen J.-P. (2008) Principles of Model Checking. The MIT Press, Cambridge, Massachusetts. ISBN: 978-0-262-02649-9

[4] Bulychev, P., David, A., Guldstrand Larsen, K., Legay, A., Mikučionis, M., & Bøgsted Poulsen, D. (2012). Checking and distributing statistical model checking. In NASA Formal Methods: 4th International Symposium, NFM 2012, Norfolk, VA, USA, April 3-5, 2012. Proceedings 4 (pp. 449-463). Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-28891-3_39

[5] David, A., Larsen, K. G., Legay, A., Mikučionis, M., & Poulsen, D. B. (2015). Uppaal SMC tutorial. International journal on software tools for technology transfer, 17, 397-415.

[6] Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., & Yi, W. (1996). UPPAAL—a tool suite for automatic verification of real-time systems (pp. 232-243). Springer Berlin Heidelberg. DOI: 10.1007/BFb0020949

[7] Laibinis, L., Troubitsyna, E., Graja, Z., Migeon, F., & Hadj Kacem, A. (2014). Formal modelling and verification of cooperative ant behaviour in Event-B. In Software Engineering and Formal Methods: 12th International Conference, SEFM 2014, Grenoble, France, September 1-5, 2014. Proceedings 12 (pp. 363-377). Springer International Publishing. DOI: 10.1007/978-3-319-10431-7_29

[8] Bonabeau, E., Dorigo, M., Theraulaz, G., & Theraulaz, G. (1999). Swarm intelligence: from natural to artificial systems (No. 1). Oxford university press. DOI:10.1093/oso/9780195131581.001.0001

[9] Iftikhar, M. U., & Weyns, D. (2012). A case study on formal verification of self-adaptive behaviors in a decentralized system. DOI: 10.4204/EPTCS.91.4

[10] Tarasyuk, A., Pereverzeva, I., Troubitsyna, E., & Laibinis, L. (2013). Formal development and quantitative assessment of a resilient multi-robotic system. In Software Engineering for Resilient Systems: 5th International Workshop, SERENE 2013, Kiev, Ukraine, October 3-4, 2013. Proceedings 5 (pp. 109-124). Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-40894-6_9

[11] Pereverzeva, I., Troubitsyna, E., & Laibinis, L. (2012). Development of fault tolerant MAS with cooperative error recovery by refinement in event-B. DOI: 10.48550/arXiv.1210.7035

[12] D.Daukševič. Kompiuterinių sistemų verifikavimas taikant modelių patikrinimo metodą TLA+ Toolbox aplinkoje. Baigiamasis bakalauro darbas, Vilniaus Universitetas, Vilnius, 2021.

[13] Behrmann, G., David, A., & Larsen, K. G. (2004). A tutorial on uppaal. Formal Methods for the Design of Real-Time Systems: International School on Formal Methods for the Design of Computer, Communication, and Software Systems, Bertinora, Italy, September 13-18, 2004, Revised Lectures, 200-236. DOI: 10.1007/978-3-540-30080-9_7

[13] UPPAAL Documentation (2021) URL: https://docs.uppaal.org [Žiūrėta 2022-06-17]