International Journal of Electrical and Computer Engineering (IJECE)
Vol. 15, No. 1, February 2025, pp. 870~882
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i1.pp870-882

Timed concurrent system modeling and verification of home care plan

Acep Taryana[1,2], Dieky Adzkiya[1], Muhammad Syifa'ul Mufid[1], Imam Mukhlash[1]
[1]Department of Mathematics, Faculty of Science and Data Analytics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[2]Department of Electrical Engineering, Engineering Faculty, Universitas Jenderal Soedirman, Purwokerto, Indonesia

Article Info

Article history:

Received Dec 9, 2023
Revised Sep 20, 2024
Accepted Oct 1, 2024

Keywords:

Deadlock
Formalism
Home care plan
Synchronization
Verification

ABSTRACT

A home care plan (HCP) can be integrated with an electronic medical records (EMR) system, serving as an example of a real-time system with concurrent processes. To ensure effective operation, HCPs must be free of software bugs. In this paper, we explore the modeling and verification of HCPs from the perspective of scheduling data operationalization. Specifically, we investigate how patients can obtain home services while preventing scheduling conflicts in the context of limited resources. Our goal is to develop and verify robust models for this purpose. We employ formalism to construct and validate the model, following these steps: i) develop requirements and specifications; ii) create a model with concurrent processes using timed automata; and iii) verify the model using UPPAAL tools. Our study focuses on HCP implementation at a regional general hospital in Banyumas District, Central Java, Indonesia. The results include models and specifications based on timed automata and timed computation tree logic (TCTL). We successfully verified a concurrent model that utilizes synchronized counter variables and a sender-receiver approach to analyze collision constraints arising from the synchronization of patient and resource plans.

Corresponding Author:

Dieky Adzkiya
Department of Mathematics, Faculty of Science and Data Analytics, Institut Teknologi Sepuluh Nopember
Keputih, Sukolilo, Surabaya, East Java 60111, Indonesia
Email: dieky@its.ac.id

1. INTRODUCTION

Electronic medical records (EMR) systems are still developing, as information communication technologies are prioritized in developed and developing countries [1]–[3]. Various problems with EMR system implementation have been reported, such as those reported by Ebbers *et al.* [4] recording medical records can increase the burden on doctors. More specifically, in Indonesia: "the government has regulated the implementation of EMR systems, however, specific rules are needed to regulate the implementation it" [5], the development of it is currently concentrated on a single institution [6], Indonesia is in the process of developing it to replace manual systems with electronic alternatives or to establish a new system [1]. In 2019, 2.8% of hospital health facilities had already adopted it, and it was also reported that of those 78 hospitals, many still needed to implement it [7]. The challenges encountered in the implementation of it stem from issues related to infrastructure, standard operating procedures (SOPs), inadequate coding quality, system integration, hospital governance, service errors, and application bugs [8].

At a time when EMR systems implementation problems were still emerging, several researchers had written the results of their studies regarding the need for patient services from being provider-centered to being expanded to patient-centered [9]. These extensions are known as integrated home care (IHC) [10] and home

care plan (HCP) according to [11]. Both systems can be connected to the hospital medical record facilities, thus improving the adoption of EMR systems [12]. One example of a patient-centered service is a HCP. Gani *et al.* [11] state that the formulation for the specifications of home care plans is challenging for several reasons: care plans are inherently nonstructured processes that involve repetitive activities but are irregular and require complex temporal expressions.

HCP is an example of a real-time system or software with concurrent processes. Quality software must meet reliability, performance, and ease of use [13]. Errors and bugs are major software problems that can cause plane crashes, disrupt space shuttle missions, and even stop trading on the stock market. Ariane's rocket launch crashed due to a floating-point overflow, a bug that caused the crash [14]. The cause of the bug is a deadlock [15], [16]. Conditions such as deadlock, starvation, and inconsistency must be prevented as early as possible [15], [17].

Gani *et al.* [11] have discussed the formal modeling and verification of HCP problems, but it still needs to address the issue of deadlock conditions. As mentioned above, deadlocks cause software bugs that must be resolved. In HCP systems, deadlocks may occur if health services do not provide sufficient medical personnel to provide patient care at home.

The operation of an HCP system requires guarantees as a quality criterion in terms of development and operation. An HCP must be free of software bugs. In this paper, we discuss HCP modeling and verification from the perspective of scheduling data operationalization. How can patients obtain services at home to avoid scheduling conflicts with providers such as doctors and nurses? Each patient's business processes are unique and may be complex as they adapt to the patient's circumstances. Each proposed case requires a business process model to be checked to determine whether it can satisfy the request. The system must provide the best business process flow to serve patients. There is a lot of literature that has investigated improving business process modeling, for example, using petri nets [18] business process modeling and notation (BPMN), and unified modeling language (UML) [19], and conformance checking using graph databases [20]. However, researchers chose to use timed automata modeling for the HCP problem because timed automata can be formed through UML state-machine translation [21], which is helpful for automation development. The other advantage of timed automata is that they can model complex systems, unlike petri nets [22]. We propose a method to model and verify HCP. The study aims to develop and verify these plans so they are flexible for patients to create a home care plan and more accessible for the coordinator to control the collision and monitor the use of limited resources. Apart from being an example of real-time software, HCP can also be critical software [23] in health services because there are crucial aspects of the sequence of events and timing (not speed of performance) [24] and errors or bugs that can have a critical impact on patients. Based on the deadlock potential that causes bugs and critical systems, we conducted a study to contribute to the formal verification of the healthcare sector. The proposed solutions are as follows: i) formulate requirements and specifications; ii) model concurrent processes using timed automata; and iii) employ UPPAAL for verification. Note: UPPAAL is a formal modeling and verification tool for timed automata-based systems that allows us to verify properties using temporal logic [25].

This paper explores formal methods for modeling and verifying HCP. For this purpose, we model it using timed automata. The model encompasses a home care plan, a counter variable, and a medical staff automaton. We present two proposed verification methods. The first approach was modeled using a counter variable automaton, while the second used a medical staff automaton. We handle only postoperative and diabetic wound care cases. In three stages, we carried out our research study. Identifying a compatible healthcare system for implementing HCPs is the initial step. Home care health service systems are identified in the second stage. The HCP is formally modeled in the third stage. The final stage verifies the reachability, satisfaction, invariance, and certainty of the proposed automata with UPPAAL. The verification results are either satisfying or unsatisfying.

The rest of the paper is organized as follows: section 2 defines the formalism of the requirements and specifications, model, and UPPAAL verification. Section 3 discusses the study results, including the formulation of requirements and specifications of HCP, construction of home care patterns, modeling for limited resource synchronization, and verifying the automata of HCP using UPPAAL. Section 4 presents the study's conclusions.

2. METHOD

In this section, we discuss the formalization of HCP systems. The case study used was related to postoperative and diabetic wound care. Additionally, we have added a process synchronization aspect, a novelty for synchronizing some timed automata for home care plan systems, so that processes can avoid deadlocks [16]. The research design involves formalizing timed systems and UPPAAL verification.

2.1. Formalizing timed systems

In general, verification is a step to find a model (M) that meets the specification (ϕ) in a formal language,

$$M \vDash \phi \quad (1)$$

(read "M satisfies ϕ "). In this paper, we propose modeling a home care plan. Home care is a new service that supports EMR systems. Integrating home care services with EMR systems is critical because many urgent operations or processes threaten safety [26]. Home care is also a system that has timed requirements that can be clearly stated [11] and can be seen as a real-time system in which there are concurrent processes [27], [28] that require modeling using timed automata [11]. Proper home care involves modeling and preparing specifications that include the time aspect in the discussion. According to Gani *et al.*, timed automata can model an example home care system (as a M) [11].

In addition, specifications are prepared more precisely using timed computational tree logic (TCTL) [29]. The verification steps include i) developing requirements and specifications; ii) developing timed models; and iii) verifying the models. Each step is explained in subsections 2.1.1, 2.1.2, and 2.1.3.

2.1.1 Requirements and specifications

First, regarding the development of requirements and specifications, in the case of timed systems, we use TCTL-based specifications. An example of a timed requirement for a home care unit is as follows: According to Neal [30], postoperative wound care adapts to the location of the wound: i) the criteria for treating surgical wounds on the face are 3 to 5 days; ii) surgical wounds on the scalp and arms for 7 to 10 days; iii) surgical wounds on the chest, stomach, hands, and legs for 10–14 days; and iv) surgical wounds on the palms of the hands and feet for 14 to 21 days. Patients' home care requirements differ from those of hospital services. Home care is repetitive, unstructured (regular and irregular), and specific to certain patients. The specifications are determined based on time information expressed as a quadruplet (days, time ranges, period, duration) [11]. We will conduct a survey, collect data, and prepare requirements for home care services at one of the government hospitals in the Banyumas District.

2.1.2 Timed models

The second involves developing timed system models. The theory underlying the development of timed system specifications, specifically, using timed automata, is explained in more detail in a previous paper [31]. Gani *et al.* defined automata models as follows, especially for modeling a home care plan inside timed automata [11]. Definition 1 (Automata). Automata $A = (S, s_0, \Sigma, X, Inv, T, F, W, E, St)$ where*:*

- S is a finite set of the locations or states of the automaton*;* s_0 is the initial state*;* $F \subseteq S$ is the final state*;* $W \subseteq S$ is the set of waiting states*;* $E \subseteq S$ is the set of executed states*;* $St \subseteq S$ is start states*;*
- Σ is a finite set of transition labels including $\{\epsilon\}$*;*
- X is a finite set of clocks*;*
- $Inv: S \longrightarrow \phi(X)$, which associates an invariant in each state of the automaton*;*
- $T \subseteq S \times \Sigma \times \phi(X) \times 2^x \times S$ is a transition set.

Based on definition 1, we construct two automata to model patient care and nurse or physicians services. To model these two instances, several conditions need to be observed, such as scheduling and arrangements with limited resources e.g., nurses and physicians. Therefore, we discuss the synchronization settings for the resources, which are explained as follows.

The synchronization approach between processes in the model is implemented using two methods: i) one automaton uses shared global variables [32]–[34]; and ii) two automatons are synchronized (‖) using process synchronization: "sender!" and "receiver?" [25]. The first method uses a global variable to manage limited resources. Dinsdale-Young *et al.* [33] use the semaphore and counter variable to manage concurrency. Furthermore, Cicirelli and Nigro [32] state the semaphore using a counter variable in the Morris algorithm. The second method explains how the two automatons communicate synchronously. For example, a lamp automaton communicates with the user automaton. According to definition 1, the lamp automaton has the attribute S = {off, low, bright}, y clock is written as X = {y}, whereas the user automaton has the attribute S = {idle}. The behaviors of the automata are as follows: If the user presses a button in Figure 1 that is identified with "press!" as a sender (i.e., it synchronizes with "press?" as a receiver, the lamp is turned on, as shown in Figure 2). In addition, the user presses the button randomly at any time or does not press the button at all. The clock y of the lamp is used to detect whether the user is moving ($y < 5$) quickly or slow ($y \geq 5$). We use this concept to manage limited resources for the home care system, which will be discussed in the next section. The first and second synchronization methods are combined.

Furthermore, we also developed a reactive home care algorithm while creating a home care model using automata. The purpose of pseudocode is to enhance readers' comprehension of automaton behavior through code. This code includes sequential instructions that can be interpreted concurrently [35]. Therefore, this algorithm can solve a similar problem.

2.1.3. Model verification

A formal verification approach is a systematic method for ensuring that a system or model meets its specifications and properties. This approach uses mathematical logic, formalisms, and formal analysis tools [25], [29], [36], [37]. In general, the objective of formal verification is to prove or carefully check whether a system or model conforms to specified requirements. In (1), M represents the model, while ϕ is the properties or specifications.

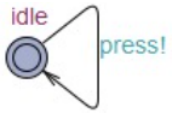

Figure 1. User automaton

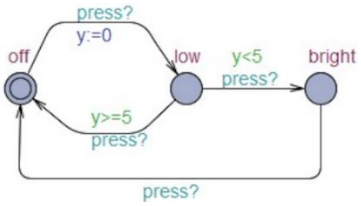

Figure 2. Lamp automaton

2.2. UPPAAL verification

Previous research that utilized UPPAAL included modeling and verification using UPPAAL [32] to formally verify deadlock-free as well as safety and response properties expressed in TCTL using an existing model checker, e.g., UPPAAL [38], facilitates automatic conversion of verified timed automata-based models (in UPPAAL) [39]. UPPAAL is a worldwide-famous model checking tool for timed automata [40]. UPPAAL timed automata take advantage of validation and verification support in the UPPAAL tool [41]. We use the following formula to check the model [25], [42]:

- $E <> \emptyset$ *(*Possibly $\emptyset$, *i.e.*, a state exists where $\emptyset$ holds*)*
- $A[]\emptyset$ *(*Invariantly $\emptyset$, equivalent to not $E <>$ not $\emptyset$*)*
- $E[]\emptyset$ *(*Potentially always $\emptyset$, *i.e.*, a state path exists over which $\emptyset$ always holds*)*
- $A <> \emptyset$ *(*Always eventually $\emptyset$, equivalent to not $E[]\emptyset$ not $\emptyset$*)*
- $\emptyset \longrightarrow \psi$ *(*$\emptyset$ always leads to ψ, equivalent to $A\,[](\emptyset$ implies $A <> \psi)$ *)*

3. RESULTS AND DISCUSSION

3.1. Requirements and specifications of home care plans

Survey results: a hospital in Banyumas District has developed home care particularly for treating diabetic and postoperative wounds. The service is simple: patients register for treatment via a specific WhatsApp, and then the nursing team approves the patient for the visit. One example of the requirements for patients participating in the HCP is as follows:

a. Diabetic wounds were treated daily for approximately 7 to 10 days. Wound treatment times can depend on patient availability. For example, Patient_1: every day from 1 to 10 October 2023, Patient_2: like Patient_1, but there are other restrictions; namely, visits are held every 08.00 am. Patient_3: Like Patient_2, there are exceptions, namely, except on 4 October 2023 at 12.30 noon.
b. Postoperative wound care adapts to the location of the wound: The criteria for treating surgical wounds on the face are approximately 3to 5 days, surgical wounds on the scalp and arms approximately 7–10 days, surgical wounds on the chest, stomach, hands, and legs for approximately 10 to 14 days, and surgical wounds on the palms of the hands and feet for approximately 14–21 days [30].

Specifically, regarding the specifications for diabetes and postoperative wound services, patient specifications were prepared based on the requirements statements explained in the introduction. The sentences arranged in a set of requirements have words that express temporality, such as "every day," "for ten days," and "from 08.00 to 20.00". Requirements also have different times than regular ones (except). Example: Patient_3 planned to treat diabetes wounds from 10/01/23 to 10/10/23 every morning except on 10/04/23 at 12.30. Examples of Patient_1 and Patient_2 are in Table 1.

The construction of the requirements sentence for postoperative wound care is presented in quadruplet form in Table 2. Example: Patient_3 planned treatment by receiving reinforcement dressing and daily injection activities except for 10/04/23. Each activity requires a treatment time of approximately 30 min.

Table 1. Specification of activities "diabetic wounds" for any patients

Patients	Activity	Days	Time ranges	Period	Duration
Patient_1	Changes in diabetic wound bandages	Every day	Morning	10/01/23-10/10/23	30
Patient_2	Changes in diabetic wound bandages	Every day	Morning, 08.00	10/01/23-10/10/23	30
Patient_3	Changes in diabetic wound bandages	Every day except (10/04/23)	Morning	10/01/23-10/10/23	30
		10/04/23	12.30	10/01/23-10/10/23	30

Table 2. Specification of activities "postoperative wounds" for any patients

Patients	Activity	Days	Time ranges	Period	Duration
Patient_1	Reinforce dressing	Every day	Morning	10/01/23-10/10/23	30
Patient_2	Reinforce dressing	Every day	Morning, 08.00	10/01/23-10/10/23	30
Patient_3	Reinforce dressing	Every day except (10/04/23)	Morning	10/01/23-10/10/23	30
	Injection	10/04/23	12.30	10/01/23-10/10/23	30

3.2. HCP pattern

HCP patterns can be categorized into three distinct types: daily care patterns, relative daily care patterns on specific dates, and absolute patterns on certain dates. The focus of this study is on the specifications and development of automata that utilize the first two categories of HCP patterns. These patterns will be elaborated upon in the subsequent subsection.

3.2.1 Every day pattern

This pattern represents a patient's need to visit the hospital daily in the morning, evening, or at certain times, such as 09:00. Table 3 specifies visits for patients with diabetes to receive medical procedures such as "diabetic wound care" and "injection". Figure 3 shows an everyday pattern automaton for treating patients with diabetes, which refers to the specifications in Table 1.

Table 3. Examples of "diabetic wounds"

Activity	Days	Time ranges	Period	Duration
Treating wounds	Every day	Morning	10/01/23-10/10/23	30
Injection	Every day	09:00	10/01/23-10/10/23	20

Wound treatment was planned every morning from 10/01/2023 to 10/10/2023, and injection activity was carried out every day starting at 09:00. Transition 1 explains that injection activity can begin at 9:00, and transition 2 explains that wound treatment activity begins after injection. Injection can be performed after wound treatment, as shown in transitions 5 and 7. The automata resets the day clock value (xd) every xd up to 1,440 minutes (one day). All possible schedules of injection and treating wound activities acceptable to the automata include "(treating wound, 480), (injection, 540), (injection, 1980), (treating wound, 2001), (Injection, 406620) " in a certain period.

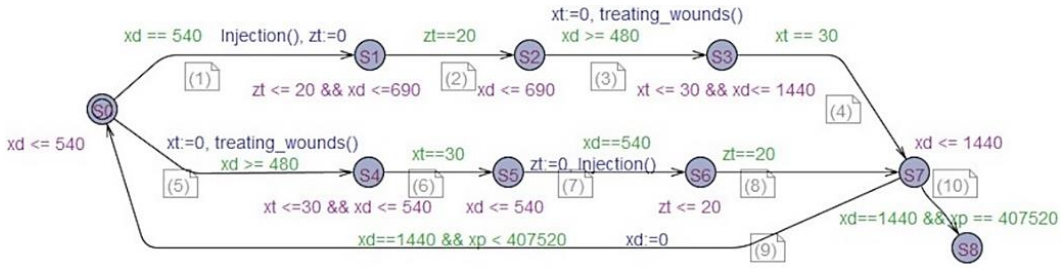

Figure 3. Diabetic automata for “every day pattern”

3.2.2. Relative days pattern with exception

During making a home care plan, the patient determines the day of the visit. However, some visits are forbidden on specific dates. For instance, postoperative patients may select a Thursday visit service for ten days (10/01/23 to 10/10/23) except for 10/04/23. Table 4 explains the specifications of the home care plan for postoperative activities, including reinforcing dressing and injection. Both activities were conducted within 60 minutes on Thursdays except for 10/04/23. Figure 4 shows the automata activity plan. Transitions 1, 2, 3, and 4 state Saturday, Sunday, Monday, and Tuesday, respectively. Transitions 5 and 8 execute injection and reinforce dressing activities, which last for a maximum of 60 minutes in locations $S1$ and $S2$. Transition 7 is a transition to accommodate the date exception "10/04/23". Transitions 10 and 11 state activities for Wednesday and Friday, respectively. Here, transition 12 represents an activity to reset the day and week clocks, and transition 13 resets the day clock. The final transition is transition 14, which represents the transition at the end of location $S4$.

Table 4. Examples of "postoperative wound care"

Activity	Days	Time ranges	Period	Duration
Reinforce dressing	Thursday except(10/04/23)	Morning	10/01/23-10/10/23	30
Injection	10/04/23	—	10/01/23-10/10/23	30

3.3. Limited resource synchronization

Home care practices lead to the joint use of limited resources. Home care may result in struggle or competition for resources. Limited resources for home care include home care medical equipment and health workers like nurses and doctors. To address this issue, we first model the limitation of nurses' resources. Due to the fact that healthcare workers' resources are limited, synchronization is necessary when patients use home care services. For example, five patients are planning home care with only two nurses available on the same day, at the same time, or at different times. Designing an automaton to handle resource contention problems is an exciting challenge. Tables 1 and 2 show the plans of many patients for home care services for diabetic and postoperative wound treatment.

Patients who are planning treatment must immediately receive confirmation of the availability of services according to their scheduled appointment. The timed automaton in Figure 4 regulates postoperative wound care. Automata still needs to discuss managing the limited availability of medical personnel to serve the many and varied patient care plans. When a patient plans home care, the automata must be able to check the possibility of a reliable schedule to avoid scheduling collisions.

To overcome the conflict in the use of medical staff resources, we propose two approaches. In the first approach, we design the automata as follows.

a. Increase the need for medical personnel to ensure a synchronization between home care services and the availability of medical personnel.
b. A global variable that detects medical use, for example, is the *counter* variable. It initially has the value of the number of medical staff, decreases by one if a patient has used it, and increases by one again if a patient has finished using it.

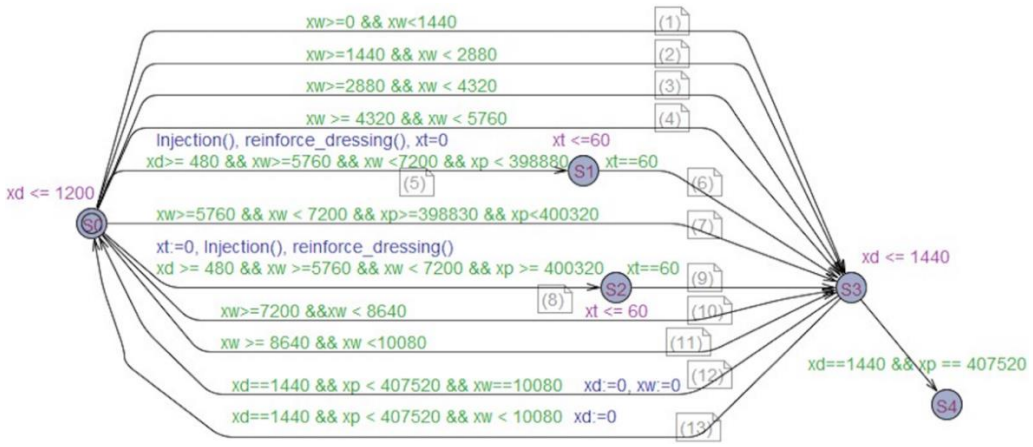

Figure 4. Postoperative wound care for "relative pattern"

The first automaton design based on the literature [34] guarantees correct sequential and temporal logic. This is a relatively simple strategy for managing the use of limited resources. In this proposal, only one automaton is arranged, and the availability of home care services is expressed as a global counter variable. Figure 5 explains the synchronization of care and availability of medical personnel services using the variable $counter$ with a maximum number of medical personnel of 2 people ($max = 2$). Home care services start from 08.00 to 11.00 every week. The variable xd represents the clock for the day, and xw represents the weekly clock.

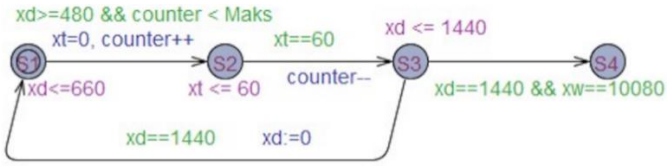

Figure 5. Synchronization of activity using a $Counter$ variable

The second approach uses a synchronization method between the home care automaton and a medical staff automaton. In this case, we discuss the maintenance activities for postoperative wound care, as in Figure 4, by adding sending "serve!" synchronization at transitions 5 and 8. Then, we create a new automaton to synchronize the availability of medical services in Figures 6 and 7, respectively. The addition of synchronization methods for source automata refers to the UPPAAL instructions [26] and the use of sender and receiver semaphores in UPPAAL [33].

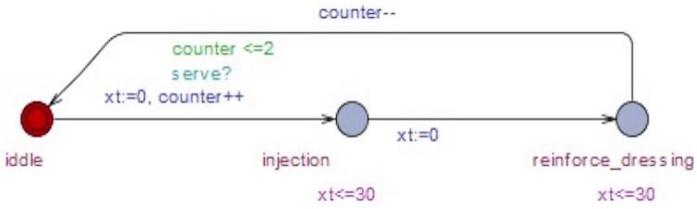

Figure 6. Medical staff automaton for postoperative wounds

The two approaches ensure that the temporal logic is guaranteed, that realizability is achieved, and that it can detect deadlocks. To describe the progress of the process, we developed an algorithm to explain the automaton synchronization process. Algorithm 1 contains an algorithm that demonstrates using the $Counter$ variable for concurrency in the home care automaton.

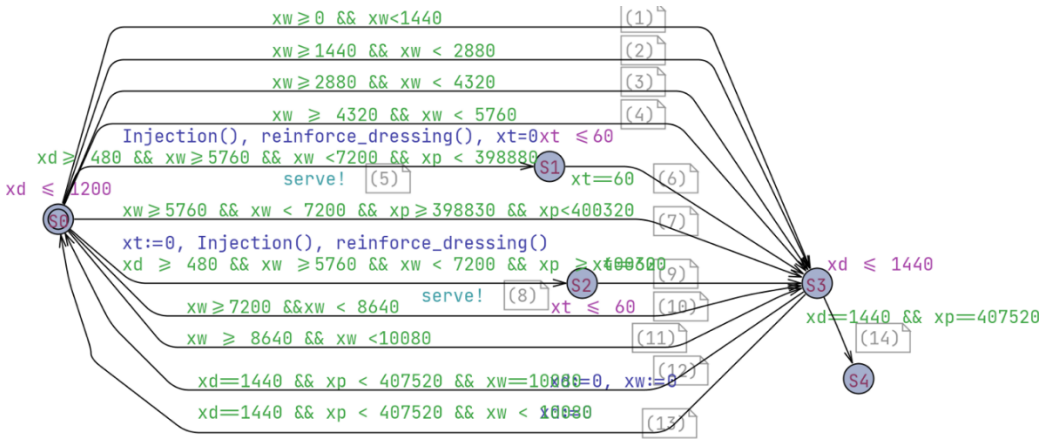

Figure 7. Automated home care plan automaton for postoperative wounds

Algorithm 2 presents the concurrency of the process formed from the two automata in Figure 6 and Figure 7. The two automata communicate using a serving channel to obtain permission to use "Injection" and "Reinforce_dressing" services from medical personnel's limited resources. The $Counter$ variable represents the available capacity of medical personnel.

Algorithm 1. Concurrency in a home care algorithm using "counter" variable

```
Program Homecare_1
{ S = {s1,s2,s3} with s1 the initial state and F ={s4} is final state. X ={xt, xd, xw} with xt,
xd, xw represents a duration clock, day clock, week clock, and a period clock in minutes,
respectively.}
{ Global Declaration For Variables, Constants, and Primitives }
        Counter : Integer,
        Max : Integer    {Max = 2}

Procedure Homecare_post_everyday()
{Local Variable Declaration}
          xt, xd, xw : Clock

{ Algorithm :}
   Repeat
         xd = 0
         s1 :
            While (xd < 480)
                      xd ++
            Endwhile
            While (xd < 660) AND Counter ≥ Max)
                       xd ++ or break {Optional because it is a non deterministic\}
             Endwhile
             xt = 0
             Counter ++
           s2:
             xt = 0
             While (xt < 60)
                     xt ++
              Endwhile
              Counter − −
            s3:
     Until (xd > 1440) AND (xw > 10080)
            s4:
{Main Program}

     Counter = 0          { Resources of nurses or doctors}
     Patient1=Homecare_post_everyday()       { Patient1 instantiation }
     Patient2=Homecare_post_everyday()       { Patient2 instantiation }
     Patient3=Homecare_post_everyday()       { Patient3 instantiation }
     Patient4=Homecare_post_everyday()       { Patient4 instantiation }
     Patient5=Homecare_post_everyday()       { Patient5 instantiation }
     Patient6=Homecare_post_everyday()       { Patient6 instantiation }
     Patient7=Homecare_post_everyday()       { Patient7 instantiation }
     Patient8=Homecare_post_everyday()       { Patient8 instantiation }


                                   End of the template
```

Algorithm 2. Concurrency in a home care algorithm using synchronization

```
Program  Homecare_2
S = {s0,s1,s2,s3} with S0 the initial state and F = {s4 } is final state. X = {xt,xd,xw,xp} with xt, xd,xw,xp
represents a duration clock, day clock, week clock, and a period clock, in minutes,
respectively.
{ Global Declaration  For Variables, Constants, and Primitives }
     serve : Chan  {Channel for synchronization}
Procedure Post_services()
{Local Variable Declaration}
      xt : Clock
 { Algorithm :}
     Repeat
        Idle :
          While (NOT serve?) OR (Counter > 1)
               xt = 0, Counter ++
           Endwhile
        Injection :
          xt = 0
```

```
          While (x_t ≤ 30)
               x_t++
           Endwhile
        Reinforce_dressing :
          x_t = 0
          While x_t ≤ 30)
               x_t++
           Endwhile
          Counter−−
     Until TRUE

Procedure Care_plan_post_except_syn()
{Local Variable Declaration}
      x_d, x_t, x_w, x_p : Clock

{ Algorithm :}
       x_p = 0
       Repeat
        x_w = 0
        Repeat
          x_d = 0
          Repeat
             s_0 :
               if (Condition_1 in Figure) then  transition_1
               if (Condition_2 in Figure) then  transition_2
               if (Condition_3 in Figure) then  transition_3
               if (Condition_4 in Figure) then  transition_4
               if (Condition_7 in Figure) then  transition_7
               if (Condition_10 in Figure) then  transition_10
               if (Condition_11 in Figure) then  transition_11
               if (Condition_5 in Figure) then
                   While (NOT serve)
                      Injection(), Reinforce_dressing()
                    Endwhile
                    s_1 :
                    x_t = 0
                   While (x_t < 60)
                      x_t++
               if (Condition_8 in Figure) then
                   While (NOT serve)
                       Injection(), Reinforce_dressing()
                    Endwhile
                       s_2 :
                         x_t = 0
                   While (x_t < 60)
                      x_t++
             s_3 :
          Until (x_d > 1440)
        Until (x_d > 1440 AND x_w > 10080 )
     Until (x_d > 1440 AND x_p > 407520 )
        s_4 :
{Main Program}
     S1=Post_Services()                                    {S1 instantiation }
     Patient5=Care_plan_post_except_syn()          { Patient5 instantiation }
     Patient6=Care_plan_post_except_syn()          { Patient6 instantiation }
     Patient7=Care_plan_post_except_syn().         { Patient7 instantiation }
     Patient8=Care_plan_post_except_syn()          { Patient8 instantiation }

                                    End of Template
```

3.4. Formal analysis of care plans using timed automata

Figure 8 depicts the instantiation of the eight patients using the template automata shown in Figure 5. Eight patients were planned to receive home care with a limited number of medical personnel (two people). The average medical service takes a maximum of 60 minutes. Similar to Figure 8, we also simulate the instantiation of 4 Care_plan_post_excep_syn automata consisting of Patient5, Patient6, Patient7, and Patient8 and a Post_services automaton, namely $S1$.

After describing and simulating the automata, the next step is to verify the properties of the proposed automata model. All modeling and verification steps use the UPPAAL tool [43]. We tried to check the model shown in Figures 6 and 7. Reachability checks are expressed as follows: $E <> (P5.S4)$ && $(P6.S4)$ && $(P7.S4)$ && $(P8.S4)$. All properties in Figure 9 are satisfied. Therefore, state $S4$ (final state) can be reached through transitions that pass through several states, and no deadlock occurs.

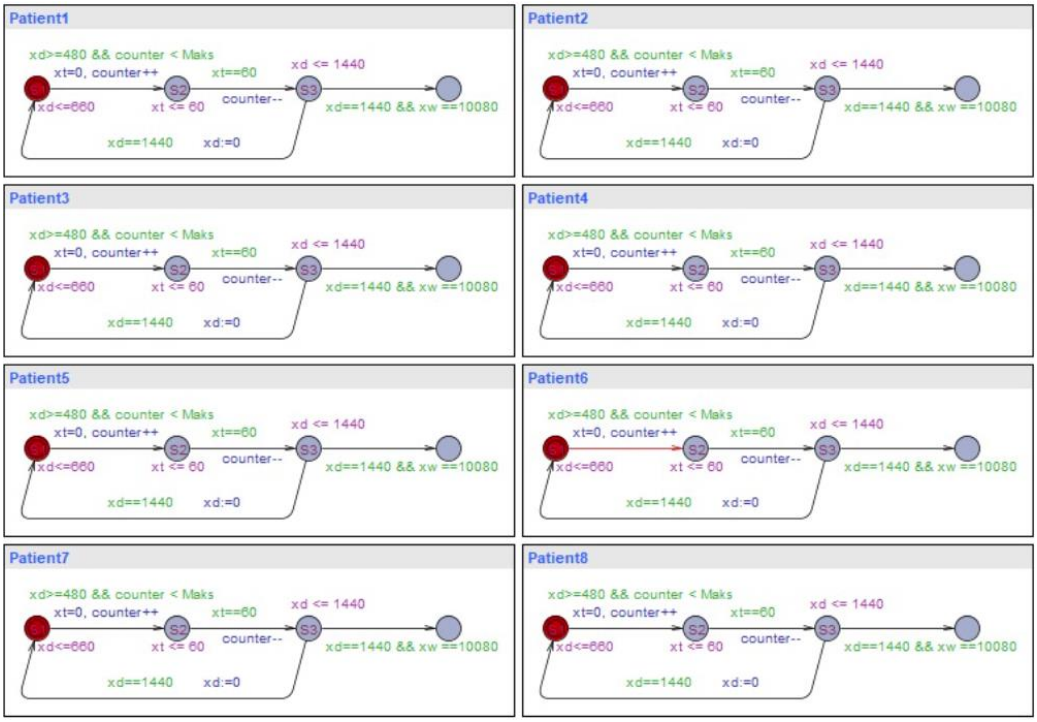

Figure 8. Automata instantiation for eight patients using synchronization counter variable

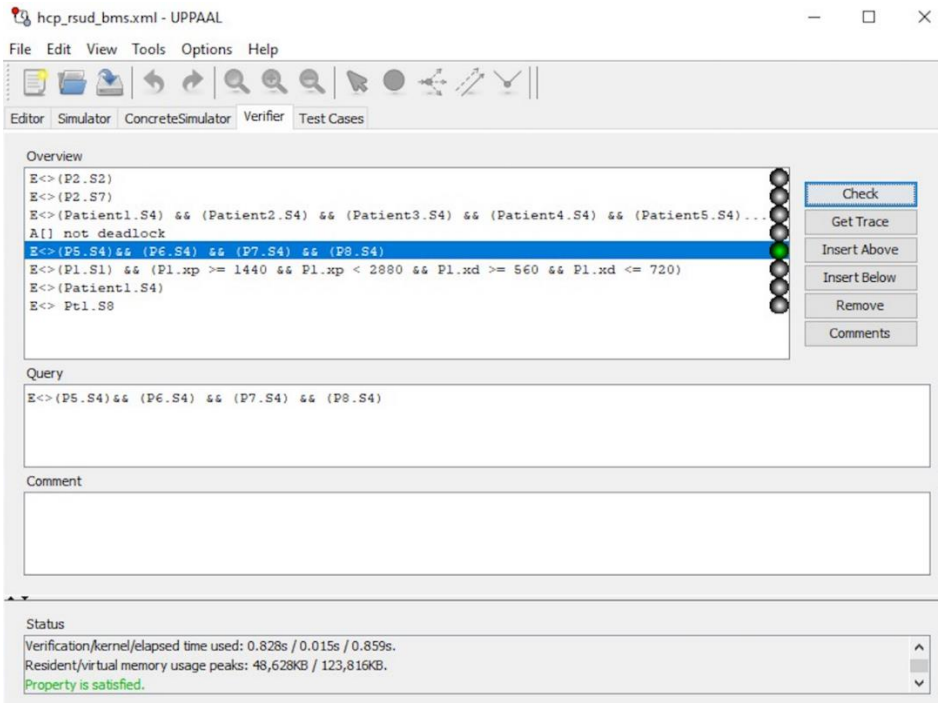

Figure 9. Reachability checking for the synchronization approach of two automata

4. CONCLUSION

The criticality of the home care system was addressed by proposing two approaches. The first approach uses a counter variable, and the second approach uses synchronization with a sender-receiver channel. The main focus of HCP discussed in this paper is diabetic and postoperative wound care. Modeling of patients who registered for home care is modeled using timed automata. The time automata are designed

according to every day patterns and relative patterns. The reachability, satisfaction, invariance, and certainty of the proposed automata are checked using UPPAAL. The result shows that all properties are satisfied. In addition, the proposed automata implement the concurrency concept. For future work, we will develop a model and the corresponding properties to identify underserved and unserved patients in model verification.

ACKNOWLEDGEMENTS

The authors thank the Directorate of Research, technology and community service, Ministry of Education, Culture, Research and Technology, Indonesian Government, for financial support through the Doctoral Dissertation Research scheme (main contract number 112/E5/PG.02.00.PL/2023 and researcher contract number 1926/PKS/ITS/ 2023).

REFERENCES

[1] N. Amalia, M. Z. A. Rustam, A. Rosarini, D. R. Wijayanti, and M. A. Riestiyowati, "The implementation of electronic medical record (EMR) in the development health care system in Indonesia," *International Journal of Advancement in Life Sciences Research*, vol. 4, no. 3, pp. 8–12, Jul. 2021, doi: 10.31632/ijalsr.2021.v04i03.002.

[2] M. C. Azubuike and J. E. Ehiri, "Health information systems in developing countries: benefits, problems, and prospects," *Journal of the Royal Society for the Promotion of Health*, vol. 119, no. 3, pp. 180–184, Sep. 1999, doi: 10.1177/146642409911900309.

[3] M. H. Alshammari, "Electronic-health in Saudi Arabia: a review," *International Journal of Advanced and Applied Sciences*, vol. 8, no. 6, pp. 1–10, Jun. 2021, doi: 10.21833/ijaas.2021.06.001.

[4] T. Ebbers, R. P. Takes, L. E. Smeele, R. B. Kool, G. B. van den Broek, and R. Dirven, "The implementation of a multidisciplinary, electronic health record embedded care pathway to improve structured data recording and decrease electronic health record burden," *International Journal of Medical Informatics*, vol. 184, pp. 1–7, 2024, doi: 10.1016/j.ijmedinf.2024.105344.

[5] T. S. Tilaar and P. L. S. Sewu, "Review of electronic medical records in Indonesia and its developments based on legal regulations in Indonesia and its harmonization with electronic health records (manual for developing countries)," *Daengku: Journal of Humanities and Social Sciences Innovation*, vol. 3, no. 3, pp. 422–430, Apr. 2023, doi: 10.35877/454RI.daengku1662.

[6] D. C. A. Nugraha and I. Aknuranda, "An overview of e-health in Indonesia: past and present applications," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 5, pp. 2441–2450, Oct. 2017, doi: 10.11591/ijece.v7i5.pp2441-2450.

[7] C. Saragih, C. Nafa Sari, B. N. Moch, and E. Muslim, "Adoption of electronic medical record in hospitals in Indonesia based on technology readiness and acceptance model," in *2020 The 6th International Conference on Industrial and Business Engineerin*, Sep. 2020, pp. 79–85, doi: 10.1145/3429551.3429565.

[8] Z. N. Indira, A. P. Widodo, and F. Agushybana, "Literature review: The effectiveness of electronic medical records (RME) on hospital service quality," *J-Kesmas: Jurnal Fakultas Kesehatan Masyarakat (The Indonesian Journal of Public Health)*, vol. 10, no. 1, pp. 57–64, Apr. 2023, doi: 10.35308/j-kesmas.v10i1.7278.

[9] N. C. Harahap, P. W. Handayani, and A. N. Hidayanto, "Barriers and facilitators of personal health record adoption in Indonesia: health facilities' perspectives," *International Journal of Medical Informatics*, vol. 162, Jun. 2022, doi: 10.1016/j.ijmedinf.2022.104750.

[10] M. Cingolani, R. Scendoni, P. Fedeli, and F. Cembrani, "Artificial intelligence and digital medicine for integrated home care services in Italy: opportunities and limits," *Frontiers in Public Health*, vol. 10, pp. 1–7, 2023, doi: 10.3389/fpubh.2022.1095001.

[11] K. Gani, M. Bouet, M. Schneider, and F. Toumani, "Using timed automata framework for modeling home care plans," in *2015 International Conference on Service Science (ICSS)*, May 2015, pp. 1–8, doi: 10.1109/ICSS.2015.36.

[12] J. Scholl, S. Syed-Abdul, and L. A. Ahmed, "A case study of an EMR system at a large hospital in India: challenges and strategies for successful adoption," *Journal of Biomedical Informatics*, vol. 44, no. 6, pp. 958–967, Dec. 2011, doi: 10.1016/j.jbi.2011.07.008.

[13] A. Z. Khan, S. Iftikhar, R. H. Bokhari, and Z. I. Khan, "Issues/challenges of automated software testing: a case study," *Pakistan Journal of Computer and Information Systems*, vol. 3, no. 2, pp. 61–75, 2018.

[14] G. Le Lann, "An analysis of the Ariane 5 flight 501 failure-a system engineering perspective," in *Proceedings International Conference and Workshop on Engineering of Computer-Based Systems*, 1997, pp. 339–346, doi: 10.1109/ECBS.1997.581900.

[15] J. C. Corbett, "Evaluating deadlock detection methods for concurrent software," *IEEE Transactions on Software Engineering*, vol. 22, no. 3, pp. 161–180, Mar. 1996, doi: 10.1109/32.489078.

[16] E. Kamburjan, "Detecting deadlocks in formal system models with condition synchronization," in *Electronic Communications of the EASST*, 2019, vol. 76, pp. 1–19.

[17] K. Tai, "Definitions and detection of deadlock, livelock, and starvation in concurrent programs," in *1994 International Conference on Parallel Processing (ICPP'94)*, Aug. 1994, vol. 2, pp. 69–72, doi: 10.1109/ICPP.1994.84.

[18] I. Mukhlash, W. N. Rumana, D. Adzkiya, and R. Sarno, "Business process improvement of production systems using coloured petri nets," *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 1, pp. 102–112, Mar. 2018, doi: 10.11591/eei.v7i1.845.

[19] F. L. Ayachi, H. B. Rahmouni, M. Ben Ammar, and H. Mahjoubi, "A reverse-engineering methodology for medical enhancement processes," *Procedia Computer Science*, vol. 164, pp. 714–723, 2019, doi: 10.1016/j.procs.2019.12.240.

[20] I. Waspada, R. Sarno, E. S. Astuti, H. N. Prasetyo, and R. Budiraharjo, "Graph-based token replay for online conformance checking," *IEEE Access*, vol. 10, pp. 102737–102752, 2022, doi: 10.1109/ACCESS.2022.3208098.

[21] F. Peres and M. Ghazel, "A proven translation from a UML state machine subset to timed automata," *ACM Transactions on Embedded Computing Systems*, vol. 23, no. 5, pp. 1–32, Sep. 2024, doi: 10.1145/3581771.

[22] S. Medina-Garcia, J. Medina-Marin, O. Montaño-Arango, M. Gonzalez-Hernandez, and E. S. Hernandez-Gress, "A petri net approach for business process modeling and simulation," *Applied Sciences*, vol. 13, no. 20, pp. 1–30, Oct. 2023, doi: 10.3390/app132011192.

[23] M. Madkour, K. Butler, E. Mercer, A. Bahrami, and C. Tao, "Semantic based model of conceptual work products for formal verification of complex interactive systems," *arXiv preprint arXiv:2008.01623*, pp. 1–11, 2020.

[24] M. Sirjani, L. Provenzano, S. A. Asadollah, M. H. Moghadam, and M. Saadatmand, "Towards a verification-driven iterative

development of software for safety-critical cyber-physical systems," *Journal of Internet Services and Applications*, vol. 12, no. 1, pp. 1–29, Dec. 2021, doi: 10.1186/s13174-021-00132-z.
[25] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *Lecture Notes in Computer Science*, vol. 3185, 2004, pp. 200–236.
[26] M.-F. Bouaziz, P. Marange, A. Voisin, and P. Jean-François, "Health checkup indicators-based safety criteria for operating sequences ranking of critical systems," *IFAC-PapersOnLine*, vol. 48, no. 21, pp. 808–813, 2015, doi: 10.1016/j.ifacol.2015.09.626.
[27] F. Reghenzani, Z. Guo, and W. Fornaciari, "Software fault tolerance in real-time systems: identifying the future research questions," *ACM Computing Surveys*, vol. 55, no. 14, pp. 1–30, Dec. 2023, doi: 10.1145/3589950.
[28] A. Burns and R. I. Davis, "Mixed criticality systems - a review," *Department of Computer Science, University of York, Tech. Rep.*, pp. 1–52, 2022.
[29] T. Vogel, M. Carwehl, G. N. Rodrigues, and L. Grunske, "A property specification pattern catalog for real-time system verification with UPPAAL," *Information and Software Technology*, vol. 154, pp. 1–18, Feb. 2023, doi: 10.1016/j.infsof.2022.107100.
[30] L. J. Neal, "Outpatient ACL surgery: the role of the home health nurse," *Orthopaedic Nursing*, vol. 15, no. 4, pp. 9–14, 1996.
[31] R. Alur, "Timed automata," in *Computer Aided Verification*, Berlin, Heidelberg: Springer, 1999, pp. 8–22.
[32] F. Cicirelli and L. Nigro, "Modelling and verification of starvation-free mutual exclusion algorithms based on weak semaphores," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, Oct. 2015, vol. 5, pp. 773–779, doi: 10.15439/2015F32.
[33] T. Dinsdale-Young, P. da R. Pinto, and P. Gardner, "A perspective on specifying and verifying concurrent modules," *Journal of Logical and Algebraic Methods in Programming*, vol. 98, pp. 1–25, Aug. 2018, doi: 10.1016/j.jlamp.2018.03.003.
[34] J. Baumeister, N. Coenen, B. Bonakdarpour, B. Finkbeiner, and C. Sánchez, "A temporal logic for asynchronous hyperproperties," in *Computer Aided Verification*, Cham: Springer International Publishing, 2021, pp. 694–717.
[35] S. Foster, K. Ye, A. Cavalcanti, and J. Woodcock, "Automated verification of reactive and concurrent programs by calculation," *Journal of Logical and Algebraic Methods in Programming*, vol. 121, pp. 1–39, Jun. 2021, doi: 10.1016/j.jlamp.2021.100681.
[36] S. E. Z. Soudjan, D. Adzkiya, and A. Abate, "Formal verification of stochastic max-plus-linear systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2861–2876, Oct. 2016, doi: 10.1109/TAC.2015.2502781.
[37] D. Adzkiya and A. Abate, "Modeling of railway networks using timed automata," *Applied Mathematical Sciences*, vol. 10, no. 49, pp. 2429–2436, 2016, doi: 10.12988/ams.2016.65208.
[38] A. David, M. O. Möller, and W. Yi, "Formal verification of UML statecharts with real-time extensions," in *Fundamental Approaches to Software Engineering*, 2002, pp. 218–232, doi: 10.1007/3-540-45923-5_15.
[39] M. Pajic, Z. Jiang, I. Lee, O. Sokolsky, and R. Mangharam, "From verification to implementation: a model translation tool and a pacemaker case study," in *2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium*, Apr. 2012, pp. 173–184, doi: 10.1109/RTAS.2012.25.
[40] K. Okano *et al.*, "A bounded model checker for timed automata and its application to LTL properties," in *Procedia Computer Science*, 2022, vol. 207, pp. 532–541, doi: 10.1016/j.procs.2022.09.108.
[41] M. Nobakht and D. Truscan, "An approach for validation, verification, and model-based testing of UML-based real-time systems," in *ICSEA 2013, The Eighth International Conference on Software Engineering Advances*, 2013, pp. 79–85, doi: 10.13140/RG.2.1.4021.8723.
[42] G. Behrmann, J. Bengtsson, A. David, K. G. Larsen, P. Pettersson, and W. Yi, "UPPAAL implementation secrets," in *Lecture Notes in Computer Science*, 2002, vol. 2469, pp. 3–22, doi: 10.1007/3-540-45739-9_1.
[43] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," in *Information and Computation*, Jun. 1994, vol. 111, no. 2, pp. 193–244, doi: 10.1006/inco.1994.1045.

BIOGRAPHIES OF AUTHORS

Acep Taryana received the S.Si. degree in mathematics from Padjadjaran University, Indonesia, in 1997 and the M.T. degree in informatics engineering from Institut Teknologi Bandung, Indonesia, in 2001. Currently, he is a student at the Department of Mathematics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. His research interests include software engineering, formal methods, verification, and real-time systems. He can be contacted at email: acep@unsoed.ac.id.

Dieky Adzkiya holds a PhD degree from Delft University of Technology, The Netherlands, in 2014. He is an assistant professor in the Department of Mathematics, Institut Teknologi Sepuluh Nopember, Indonesia. He was a postdoctoral researcher at the Delft Center for Systems and Control (DCSC), Delft University of Technology (TU Delft), The Netherlands, working with Manuel Mazo Jr. on networked control systems. His research interests include the analysis, verification, and control of max-plus-linear systems and their applications. In addition, he is currently the Head of the Postgraduate Study Program at the Department of Mathematics, Institut Teknologi Sepuluh Nopember. He can be contacted at email: dieky@its.ac.id.

Muhammad Syifa'ul Mufid received the doctoral degree from Oxford University, England, in 2021. He is an assistant professor in the Department of Mathematics, Institut Teknologi Sepuluh Nopember, Indonesia. His research interests include Latin squares, max-plus algebra, min-max-plus algebra, and computer science. He can be contacted at email: syifaul.mufid@its.ac.id.

Imam Mukhlash received the doctoral degree from Institut Teknologi Bandung, Indonesia, in 2010. He is an associate professor in the Department of Mathematics, Institut Teknologi Sepuluh Nopember, Indonesia. His research interests include computational mathematics, data mining, artificial intelligence, and software engineering. He can be contacted at email: imamm@matematika.its.ac.id.