Model Decomposition of Robustness Diagram with Loop and Time Controls to Sequence Diagrams Using Activity Groups

Kliezl Eclipse* and Jasmine Malinao

Division of Natural Sciences and Mathematics, University of the Philippines Tacloban College, Tacloban City, Leyte, Philippines
kpeclipse@up.edu.ph

Abstract. The Robustness Diagram with Loop and Time Controls is a workflow model for representing systems that can capture all three aspects of a workflow—resource, process, and case. It can be mapped to Class Diagrams – to extract the resource dimension of the system it represents, to Petri Nets – to extract both process and case dimensions thereof, and Sequence Diagrams - to extract both resource and case dimensions thereof. This paper proposes a mapping of RDLT into Sequence Diagrams that uses the concept of the input model's minimal activity and its activity group to ensure generating outputs that can capture a case management profile. The RDLT components and their mapped Sequence Diagram components are used in the proposed mapping. Such mapping can produce a set of Sequence Diagrams based on the number of objects, for checking arcs and referencing vertices. Furthermore, the mapping was used to demonstrate the partial mapping of the RDLT representation of the Chiller System to a set of Sequence Diagrams. Lastly, we validate the correctness of the decomposition by comparing the consistency of the activity profiles of the RDLT with the case management profiles of their corresponding set of Sequence Diagrams.

Keywords: Workflows, Robustness Diagram with Loop and Time Controls, Sequence Diagram, mapping, minimal activity, activity group, chiller system

1 INTRODUCTION

Workflows and Workflow Management Systems are utilized in business and scientific domains to analyze systems. A *workflow* is the automation of procedures where tasks are passed among participants based on specific rules to achieve an overall goal. The complexity of a workflow model depends on the captured workflow dimensions, which include the resource, process, and case [1]. *Work* specification defines cases with relevant processes, which profile both case and process dimensions [2]. An *activity* is the actual performance of a resource on a

* Corresponding author

J. Caro et al. (eds.), *Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2023)*, Atlantis Highlights in Computer Sciences 20,
https://doi.org/10.2991/978-94-6463-388-7_18

work specification, profiling all three dimensions [2]. *Case management* specifies a case within a system and the resources attributed to it, profiling the resource and case dimensions [3]. The Robustness Diagram with Loop and Time Controls (RDLT) is a tool for representing real-world complex systems such as *adsorption chillers* [2] and the *Philippine Integrated Disease Surveillance and Response (PIDSR) system* [5]. However, unlike other workflow models like UML Diagrams and Petri Nets, the model lacks automated tools that could support its implementation and model-to-model transformation. For example, the UML Class Diagrams, and Sequence Diagrams were previously used for the completeness of user requirements with the Robustness Diagram [6]. All these led to the partial mapping of RDLT to Class Diagrams to extract the resource dimension of the system it represents [3], Petri Nets to extract both process and case dimensions [3], and Sequence Diagrams to extract both resource and case dimensions [7]. To optimize the existing mapping of RDLT to Sequence Diagrams, this paper proposes a mapping of RDLT into Sequence Diagrams with the concept of minimal activity and activity group. The proposed mapping will be demonstrated with the RDLT representation of the chiller system.

1.1 Robustness Diagram with Loop and Time Controls

The *Robustness Diagram with Loop and Time Controls (RDLT)* [2] is an extension of the Robustness Diagram that captures all three workflow dimensions (See Figure 1).

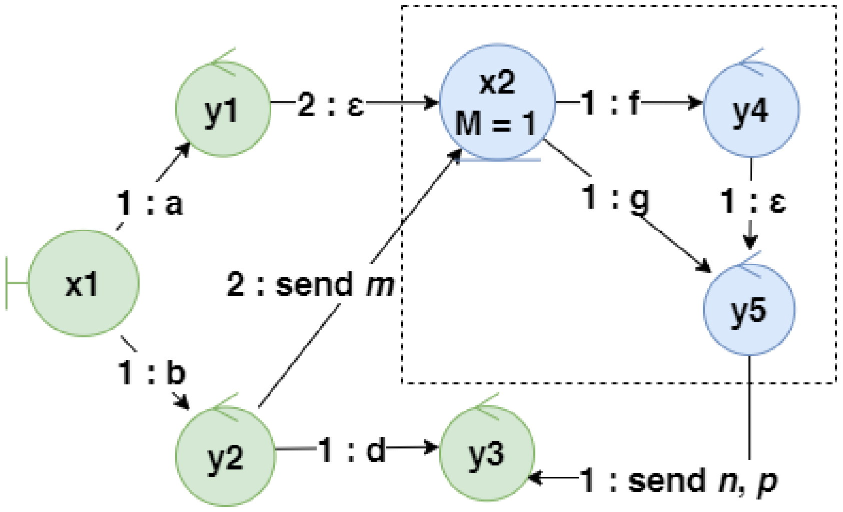


Fig. 1: RDLT (Based on [3])

Definition 1. ***RDLT*** *[2, 8]*
An RDLT is a graph representation R of a system that is defined as $R = (V, E, T, M)$ where:

- *V is a finite set of vertices where every vertex is either a boundary or entity object or a controller.*
- *E is a finite set of arcs such that no two objects are connected to each other. Furthermore, every arc (x, y) has following attributes:*
  - *$C : E \to \Sigma \cup \{\epsilon\}$ where Σ is a finite non-empty set of symbols and ϵ is the empty string. $C(x, y) \in \Sigma$ means that $C(x, y)$ is a condition that is required to be satisfied, e.g. input requirement or parameter [3], to proceed from x to y. Meanwhile, $C(x, y) \in \epsilon$ means that there is no condition imposed by (x, y) or signifies that x is the owner object of the controller y.*
  - *$L : E \to \mathbb{N}$ is the maximum number of traversals allowed on the arc.*
- *$T : E \times \mathbb{N}^n$ is a mapping such that $T((x, y)) = (t_1, ..., t_n)$ for every $(x, y) \in E$ where $n = L((x, y))$ and $t_i \in \mathbb{N}$ is the time a check or traversal is done on (x, y) by some algorithm's walk on R.*
- *$M : V \to \{0, 1\}$ indicates whether $u \in V$ is a center of a reset-bound subsystem (RBS). Given a vertex u such that $M(u) = 1$, an RBS is a substructure G_u of R that is induced by a center $u \in V$ and the set of controllers owned by u.*

  An arc $(x, y) \in E$ is said to be a bridge of G_u if and only if (1) x is not a vertex in G_u but y is. We then say that (x, y) is an in-bridge of y in G_u; or (2) x is a vertex in G_u but y is not. We then say that (x, y) is an out-bridge of y in G_u. A pair of arcs (a, b) and (c, d) are type-alike (with respect to y) if and only if (1) y is present in both arcs and (2) either both arcs are bridges of y of G_u or both are not.

An *activity profile S*, given a start vertex *s* and an end vertex *f*, can be extracted from an RDLT using an activity extraction algorithm in [2]. During the activity extraction, arcs are checked whether the maximum number of traversals is reached and whether they are unconstrained before traversal. In some cases, an RDLT can have multiple activity profiles. Given the RDLT in Figure 1 with *x1* as the start vertex and *y3* as the end vertex, one possible activity profile is $S = \{S(1), S(2), S(3), S(4), S(5)\}$, where $S(1) = \{(x1, y1), (x1, y2)\}$, $S(2) = \{(y1, x2), (y2, x2)\}$, $S(3) = \{(x2, y4)\}$, $S(4) = \{(y4, y5), (x2, y5)\}$, and $S(5) = \{(y2, y3), (y5, y3)\}$.

Definition 2. ***Reachability Configuration*** *A reachability configuration $S(t)$ in R contains the arcs at time step $t \in \mathbb{N}$. We call a set $S = \{S(1), S(2), \ldots, S(d)\}$, $d \in \mathbb{N}$, as an* ***activity profile*** *in R where $\exists (u, v) \in S(1)$ and $(x, y) \in S(d)$ such that $\nexists w, z \in V$ where $(w, u), (y, z) \in E$.*

In some algorithm's walk, an arc (x, y) is *unconstrained* [2] relative to every arc (v, y) where (x, y) and (v, y) are type-alike if at least one of the three conditions are satisfied: first is if the condition of every such (v, y) is either ϵ or the same as the condition of (x, y), second is if the number of times (x, y) has been checked/traversed is less than the number of times every such (v, y) was checked/traversed, where (x, y) and (v, y) have different Σ-conditions, or third is if for every such (v, y) has been traversed before, where (x, y) having ϵ-condition.

Checks and traversals in RDLT depend on arcs with their attributes (L, C, and T attributes) and the structure and relationships between vertices. RDLT supports connections of vertices to represent sequential flows, conditional flows and parallelisms, and iterations [2]. Figure 1 shows an AND-split from $x1$ to $y1$ and $y2$ and an AND-join from $y5$ and $y2$ to $y3$. If the C-values of the two arcs from the split were equal, it would show an OR-split. If the C-value of the arc was ϵ and the C-value of the other is in Σ, it would show a MIX-split. The same concept of the C-values applies to OR-join and MIX-join.

An RDLT has certain paths from an input vertex to an output vertex. Getting a contraction path [2], which is an abstraction of the sequences of vertices in reference to the input and output vertices, requires vertices to merge into one, resulting in a new vertex with the sequence of merged vertices as its name (See Figure 2).

Definition 3. ***Contraction Path*** *[2]*
Given a RDLT $R = (V, E, T, M)$ and its vertex-simplified RDLT $G_1 = (V_1, E_1, C_1)$, a ***contraction path*** *from x_1^1 to x_n in G_1 is a sequence $p = x_1^1 x_2 \dots x_n$, $n \leq |V_1|$, where a contraction is feasible on $(x_1^{i-1}, x_i) \in E_{i-1}$ in G_{i-1} resulting to $G_i = (V_i, E_i, C_i)$ for $i = 2, 3, \dots, n$, and $x_1^i \in V_i$ represents $x_1^{i-1} \in V_{i-1}$ and $x_i \in V_{i-1}$ whose arc (x_1^{i-1}, x_i) is contracted.*

The RDLT can capture all three workflow dimensions, which makes it a powerful tool for representing complex systems. One application of using the RDLT to model a real-world complex system is the RDLT representation of *adsorption chillers* [2, 4] and the *PIDSR system* [5].

1.2 Sequence Diagram

The *Sequence Diagram* [9] is a UML Behavioral Diagram that deals with the communication between resources via the sequence of exchanged messages. The *Object Dimension* [10] is the horizontal axis that shows the participants in the interaction. The *Time Dimension* [10] shows the order of proceedings down the page. A case management profile can be extracted from a Sequence Diagram with the process flow extraction of Sequence Diagrams in [6]. Given an input Sequence Diagram, the extraction outputs an updated Sequence Diagram with time-of-execution values next to the representation of arcs. A Sequence Diagram also uses *Sequence Fragments* [10] represented as a box that encloses a portion of the interaction within the diagram. A fragment has an operator in the top left corner of the box that indicates its type. Some types of fragments are *alternative (alt)*, *option (opt)*, *reference (ref)*, *loop*, *break*, and *sequence diagram (sd).*

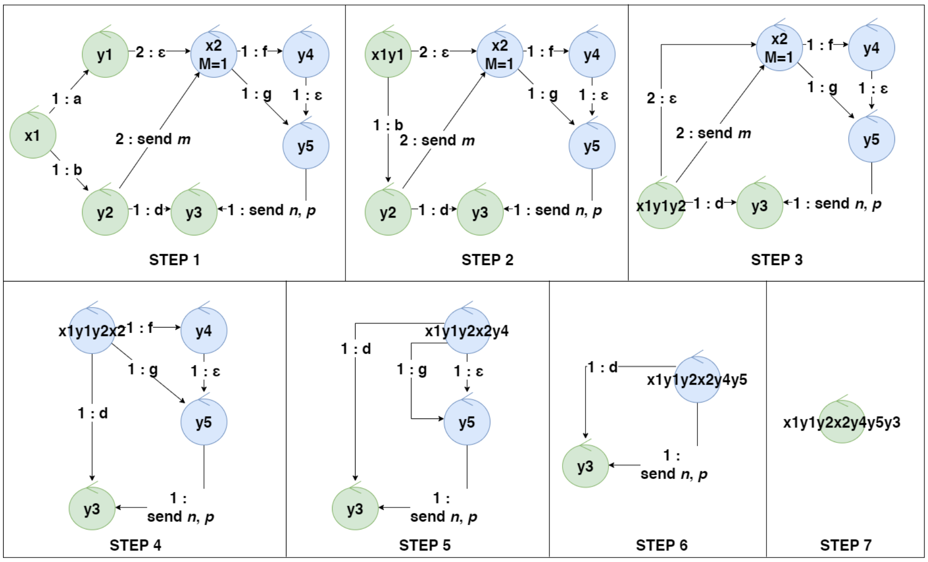


Fig. 2: Contraction Process of the RDLT in Figure 1

2 LITERATURE REVIEW

Literature in [3] explored the partial decomposition of RDLT to the design perspective of Class Diagrams. This included conditions in mapping RDLT components to class diagram components. This decomposition is limited to supporting associations between classes. They also introduced a partial mapping from RDLT to Petri Nets by considering 9 structures. They did not consider the limit L attribute in RDLT, however, which results in a mapped Petri Net being unable to limit the number of times a transition may fire. This work recommended the model decomposition of RDLT to Sequence Diagrams and to use their work for insights. Literature in [6] used the Robustness Diagram in the ICONIX paradigm as verification for User Requirements Traceability in Sequence Diagrams based on the pre-specified user requirements expressed in Use Case Texts (UCTs). The UCT is converted to its Robustness Diagram and Sequence Diagram, then both are compared by computing the *Requirements Traceability Value (RTV)* which determines the percentage of the consistency and completeness of representation between them. With RDLT being an extension of the Robustness Diagram, computing for the RTV can be adapted to validate the percentage of completeness of the components between RDLT and Sequence Diagrams once the proposed mapping is performed. Literature in [7] introduced a mapping from RDLT to Sequence Diagrams. The mapping can generate a set of Sequence Diagrams based on the number of objects with owned controllers, the number of controllers with multiple incoming arcs and with at least one outgoing arc, one to check whether an arc is unconstrained, one to update the T-values of an arc and its type-alike, and a start diagram to reference the source. This work recommended an extension of their mapping or a more optimal mapping with a lesser number of generated Sequence Diagrams. They also recommended a demonstration of the mapping with a real-world application of the input model.

3 METHODS

3.1 Minimal Activity

In some cases, an activity profile of an RDLT may be a subset of a larger activity profile. In this paper, an activity profile where no subset is another activity profile of the same RDLT is defined as the *minimal activity* S'. Some activities would reuse the components of S' to build a bigger activity profile. Such a set of activities is defined as an *activity group ActGr(S)*.

Definition 4. ***(Activity Group, Minimal Activity)***
Let $S = \{S(1), S(2), \ldots, S(k)\}$, $k \in \mathbb{M}$ *be an activity for the input pair* $[i, o]$ *of* V *in RDLT* R*. An* ***activity group*** *of* S *in* R *for* $[i, o]$*, denoted as* $ActGr(S)$*, is a set of activities in* R *where every* $S' = \{S'(1), S'(2), \ldots, S'(k')\}$ *in* $ActGr(S)$*,* $k' \in \mathbb{M}$*, where the following holds,*

1. $A \cap B \neq \emptyset$*, and,*
2. *without loss of generality, for every* $(x, y) \in B \backslash (A \cap B)$*, there exist* $(a, b) \in S'(j), (c, d) \in S'(j'), 1 \leq j, j' < k'$*, such that* $x \in \{a, b\}$*,* $y \in \{c, d\}$*,*

where $A = \bigcup_{j=1}^{k} S(j)$ *and* $B = \bigcup_{i=1}^{k'} S'(i)$*.*

$S_{min} \in ActGr(S)$ *is called a* ***minimal activity*** *for* $[i, o]$ *if* $\forall S' \in ActGr(S)$*,* $\bigcup_{j=1}^{k} S_{min}(j) \subseteq \bigcup_{i=1}^{k'} S'(i)$*.*

We call $P, Q \in ActGr(S)$ *as* ***sibling*** *activities in* R *for* $[i, o]$*. Furthermore,* $ActGr(S)$ *is a* ***maximal activity group*** *of* S *if there is no activity group* $ActGr'(S)$ *such that* $ActGr(S) \subset ActGr'(S)$*.*

Using a minimal activity guarantees that the input RDLT has at least one activity profile and that the resulting set of Sequence Diagrams has a case management profile. Unlike the mapping in [7], the proposed mapping excludes generating Sequence Diagrams that check whether an arc is unconstrained, thus having a lesser number of output Sequence Diagrams. The minimal activity also reflects the required and alternative paths in a Sequence Diagram to ensure that some algorithm's walk reaches the last message. Getting a contraction path from the source to the sink can help determine the minimal activity of an RDLT and the possible alternative paths.

3.2 RDLT and Sequence Diagram Components and Proposed Mapping

In this paper, the identified RDLT and mapped Sequence Diagram components in [7] are used, excluding the representation of an unconstrained arc. Algorithm

1 is a proposed mapping of an input RDLT R to its set of Sequence Diagrams SD. It generally follows a series of steps from determining a minimal activity S' of R to generating the required Sequence Diagrams.

Algorithm 1
Input: RDLT R
Output: Set of Sequence Diagrams SD mapped from R

1. Get Minimal Activity S' of I via Contraction Path.
2. Generate Sequence Diagrams for:
   (a) Every object o with owned controllers
   (b) $updateTValues$
3. For every object Sequence Diagram o:
   (a) Create a participant for object o.
   (b) Create a participant for objects u where an arc exists between an owned controller of o and an owned controller of u based on S'.
   (c) Add the note to the participant if an object is a center of an RBS.
   (d) Map sequence of arcs (x, y) in E and controllers from one object to another object based on S'.
      i. Use *par* fragment to start splits and end joins.
      ii. Use *loop* fragment if
         A. Controller x has a looping arc (x, y) where y is a controller and both x and y have the same owner,
         B. Controller x has a looping arc (x, y) where y is an object and y owns x, or
         C. Object x has a looping arc (x, y) where y is its owned controller.
         – If the loop may encounter two arcs in an AND-join again, use *alt* fragment which checks whether one of the arcs is checked. If one of the arcs is not checked, then use the *par* representation. Otherwise, use *alt* fragment to represent alternative unconstrained paths.
         – If the loop may encounter two arcs in a MIX-join again, use *alt* fragment which checks whether the ϵ-condition arc is checked. If it is not checked, then use the *par* representation. Otherwise, use *alt* fragment to represent alternative unconstrained paths.
      iii. Use ref fragment after arc (x, y) where y is an object and add y in fragment.
      iv. If the arc (x, y) is an out-bridge of, attach an additional note after reaching y to indicate the reset mechanisms in the Sequence Diagram.
   (e) Map sequence of arcs (x, y) in E and controllers that are not in S' on a separate fragment. Use the *opt* fragment if there is only one alternative path or an *alt* fragment if there is more than one alternative path. Add such a fragment before the execution of the required path.
4. Generate *start* Sequence Diagram.

(a) If the source is an object o, use its mapped participant and use the ref fragment.

(b) If the source is a controller c, map the sequence of arcs from the source to an object, then use the mapped participant of the object and end with the ref fragment.

– If there is no information on the controller source's owner or its previous tasks, abstract the owner as a participant $c(owner)$ and abstract the previous tasks as an asynchronous message $c()$.

Based on the algorithm, the unconstrained arc in the output Sequence Diagrams is instead represented with a synchronous message and an $updateTValues$ return message. Without these messages, the arc is automatically considered constrained.

3.3 Demonstration of Proposed Mapping

The adsorption chiller system shows reset mechanisms that RDLT can support. Additionally, an existing literature has established its RDLT representation [2]. This real-world application makes a good input to demonstrate the proposed mapping. In this paper, a portion of the RDLT representation is used as the input RDLT (See Figure 3). Using the input RDLT in getting its contraction path, Figure 4 reflects a minimal activity of the RDLT. After generating its output Sequence Diagrams, their activity and case management profiles are compared.

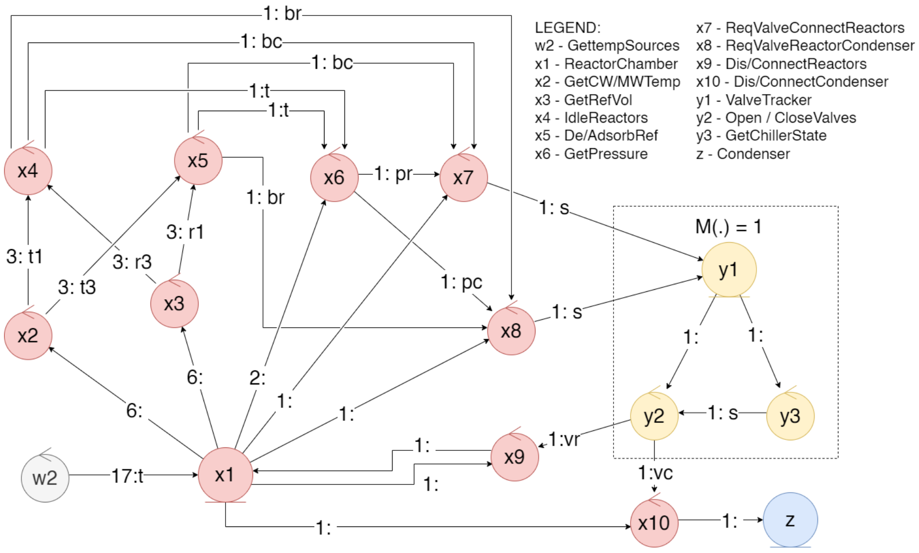


Fig. 3: A portion of the RDLT representation of Chiller System

4 RESULTS AND DISCUSSION

4.1 Demonstration of Proposed Mapping with RDLT Representation of Chiller System

After getting the contraction path from the source $w2$ to the sink z, the input RDLT R has minimal activity $S' = \{S'(1), S'(2), S'(3), S'(4), S'(5), S'(6), S'(7), S'(8), S'(9), S'(10)\}$ where $S'(1) = \{(w2, x1)\}, S'(2) = \{(x1, x2)(x1, x3)\}, S'(3) = \{(x2, x4)(x2, x5)(x3, x4)(x3, x5)\}, S'(4) = \{(x4, x6)\}, S'(5) = \{(x6, x7)(x4, x7)(x5, x7)\}, S'(6) = \{(x7, y1)\}, S'(7) = \{(y1, y3)\}, S'(8) = \{(y3, y2)\}, S'(9) = \{(y2, x10)\}$, and $S'(10) = \{(x10, z)\}$. Figure 4 reflects the minimal activity of the RDLT in Figure 3.

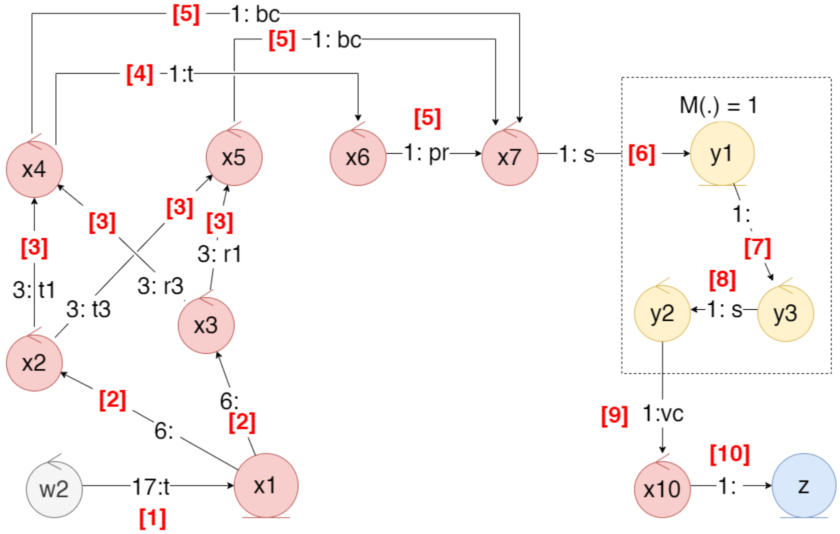
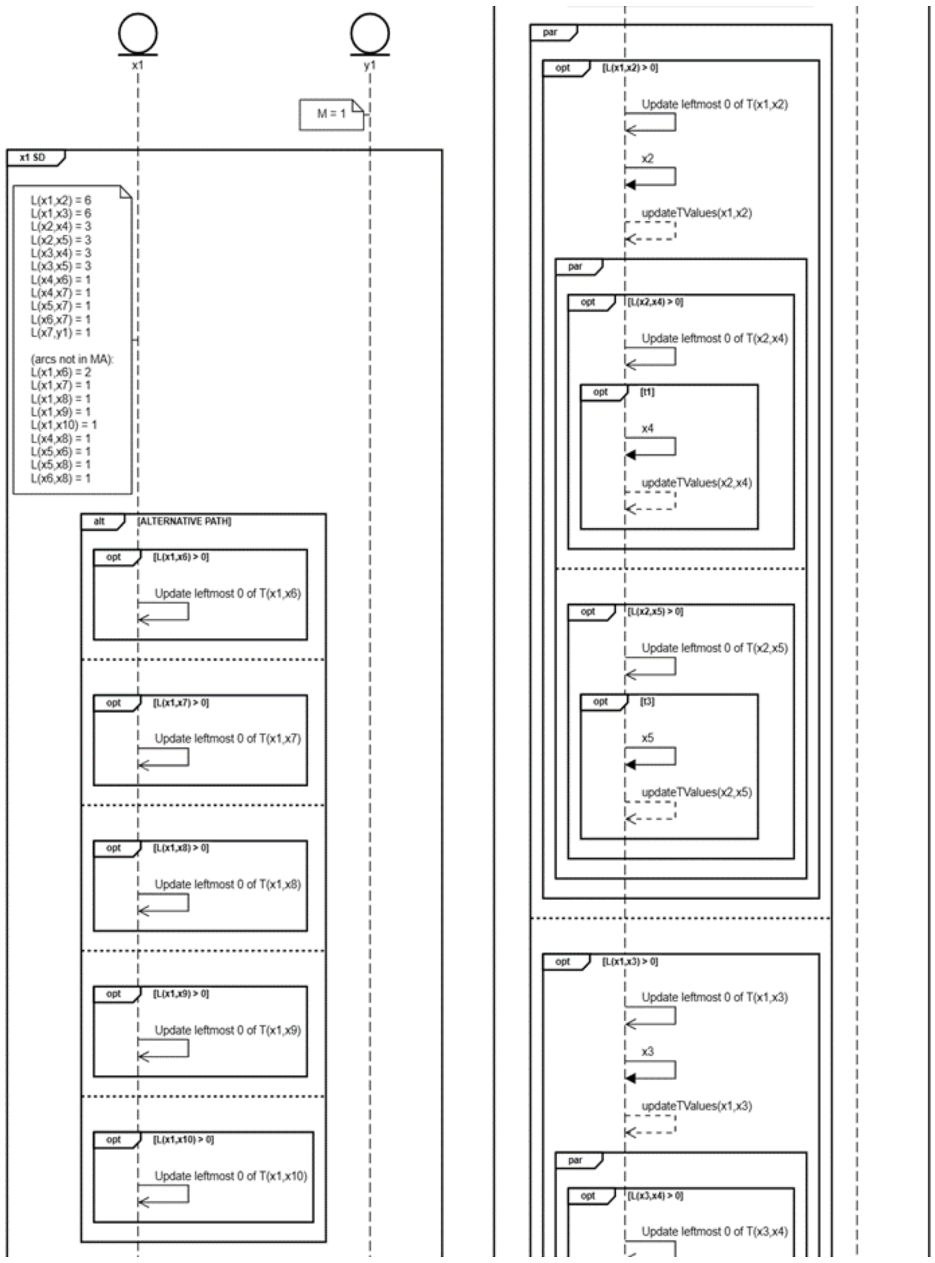


Fig. 4: Minimal Activity S' of the RDLT representation of Chiller System in Figure 3

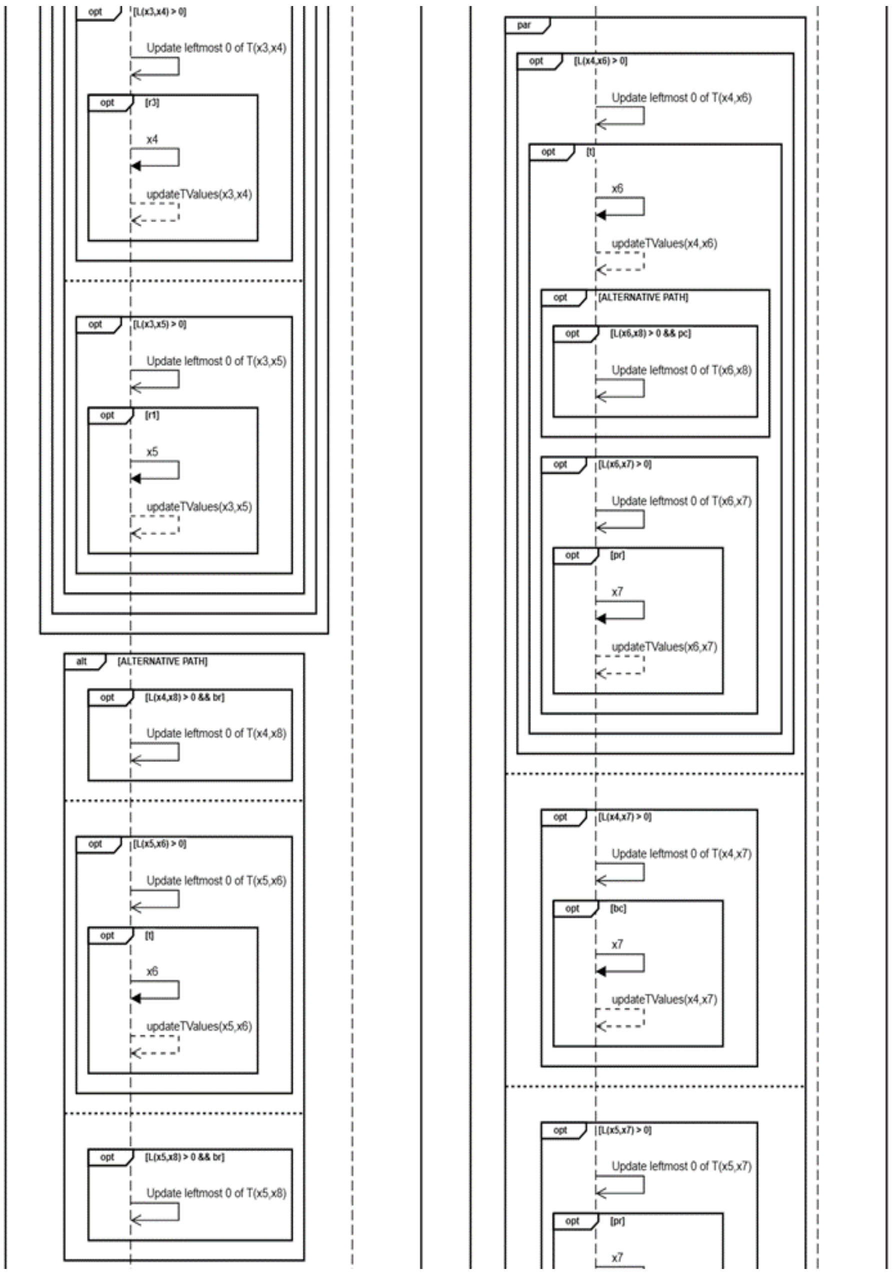
The proposed mapping generated Sequence Diagrams SD for objects $x1$ and $y1$ (See Figures 5, 6, 7, 8, and 9), the *updateTValues* Sequence Diagram for each traversed arc [7], and a *start* Sequence Diagram referencing the source controller $w2$ (see Figure 10).

In terms of length, the Sequence Diagrams for objects $x1$ and $y1$ extend longer compared to other generated diagrams. For object $x1$, Figures 5, 6, and 7 show the division of its Sequence Diagram into five (5) parts, with Part 1 showing the topmost portion of the $x1$ Sequence Diagram while Part 5 shows its bottom-most portion. Of all the generated Sequence Diagrams, the $x1$ Sequence Diagram is the longest. For object $y1$, Figures 8 and 9 show the division of its



(a) Part 1 **(b)** Part 2

Fig. 5: Generated *x1* Sequence Diagram (Parts 1 and 2)



(a) Part 3 **(b)** Part 4

Fig. 6: Generated *x1* Sequence Diagram (Parts 3 and 4)

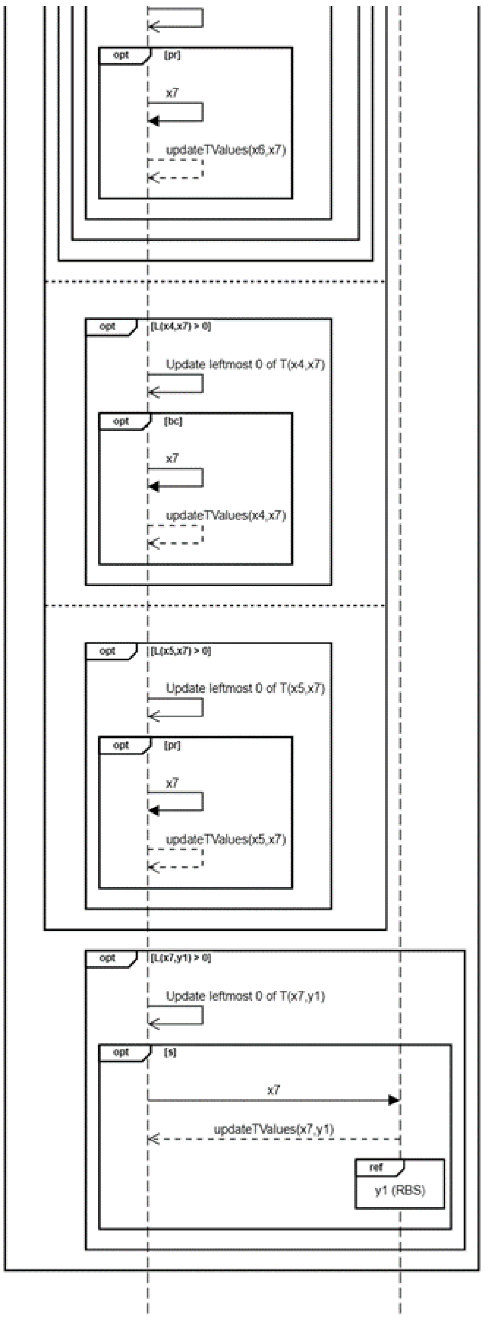


Fig. 7: Generated *x1* Sequence Diagram (Part 5)

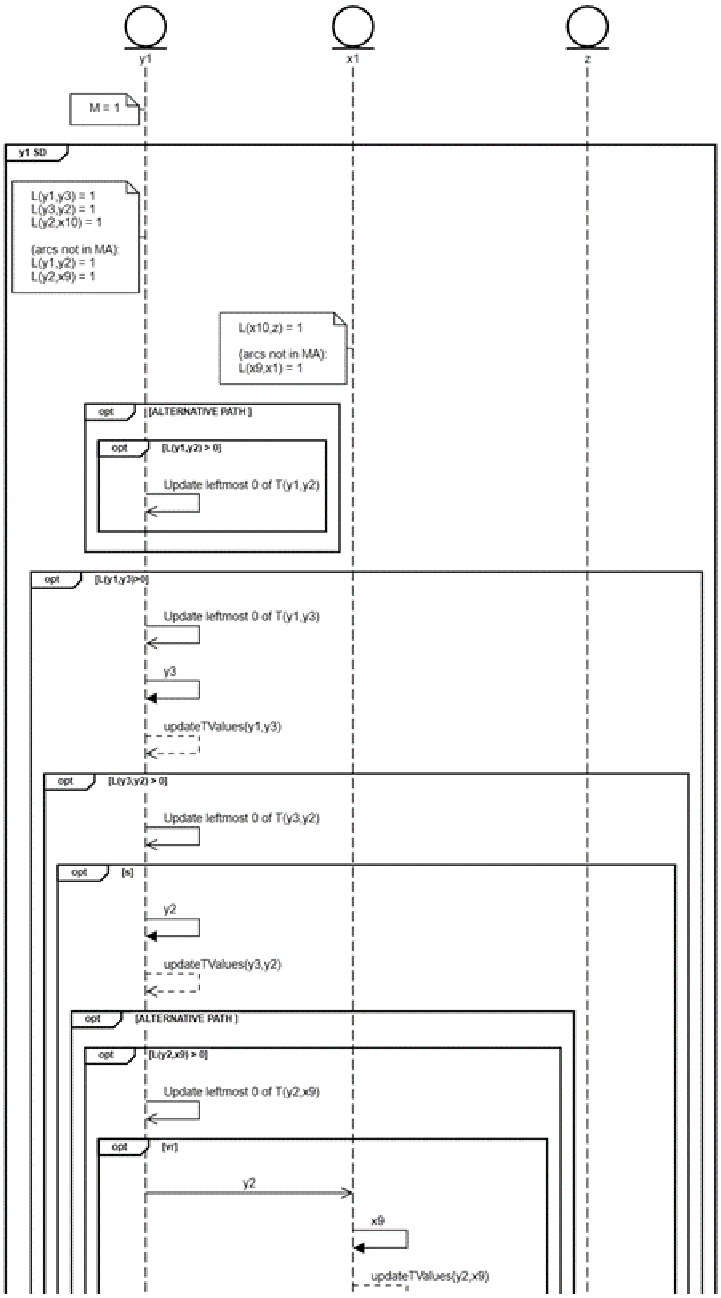


Fig. 8: Generated *y1* Sequence Diagram (Part 1)

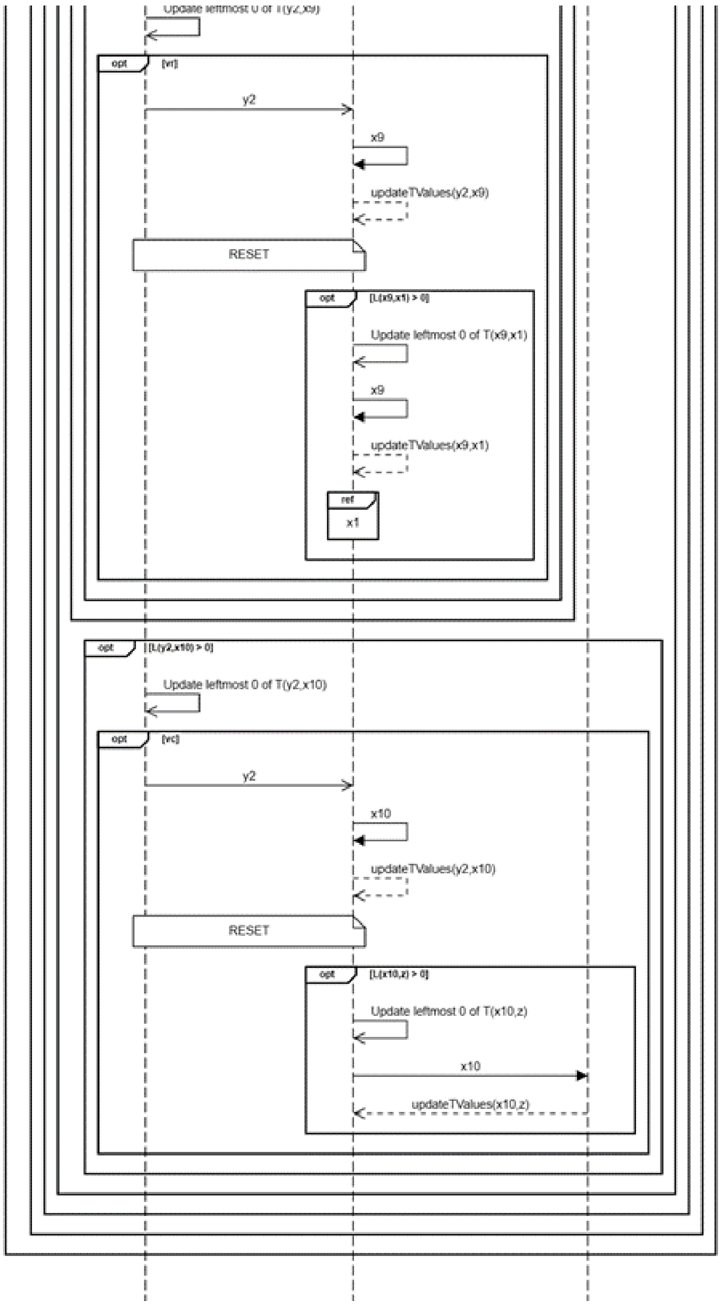


Fig. 9: Generated *y1* Sequence Diagram (Part 2)

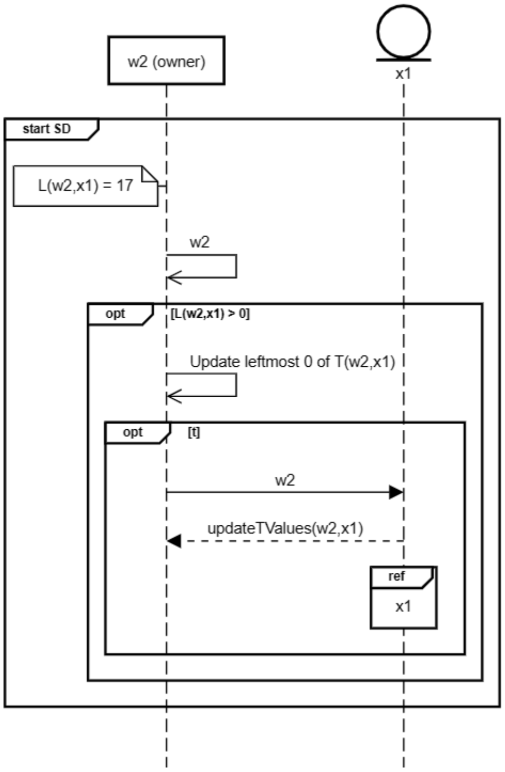


Fig. 10: Generated *start* Sequence Diagram for the RDLT in Figure 3

Sequence Diagram into two (2) parts, with Part 1 showing the topmost portion of the $y1$ Sequence Diagram while Part 2 shows its bottom-most portion.

Figure 11 shows the vertices and arcs not included in the minimal activity, which are represented as alternative paths in the generated diagrams. The alternative paths show the arcs from $x1$, $x4$, $x5$, and $x6$ in the $x1$ Sequence Diagram. The alternative paths show the arcs from $y1$, $y2$, and $x9$ in the $y1$ Sequence Diagram.

The figure highlights the alternative arcs captured in the generated Sequence Diagrams. Notice that $x8$ and its outgoing arcs, highlighted in red, were not mapped in the generated diagrams. This is due to the constrained alternative paths from $x8$'s ancestors, which means that $x8$ and its outgoing arcs are never reached. This example proves that the proposed mapping in reference to one minimal activity of an RDLT may lack some RDLT elements in the generated Sequence Diagrams.

Using the flow extraction in [6], the generated Sequence Diagrams have their updated version with the time of traversal for each T-value update at check and at traversal. Table 1 shows the summary of the time of traversal T of RDLT arcs as seen in the minimal activity profile and the $updateTValues$ return messages in

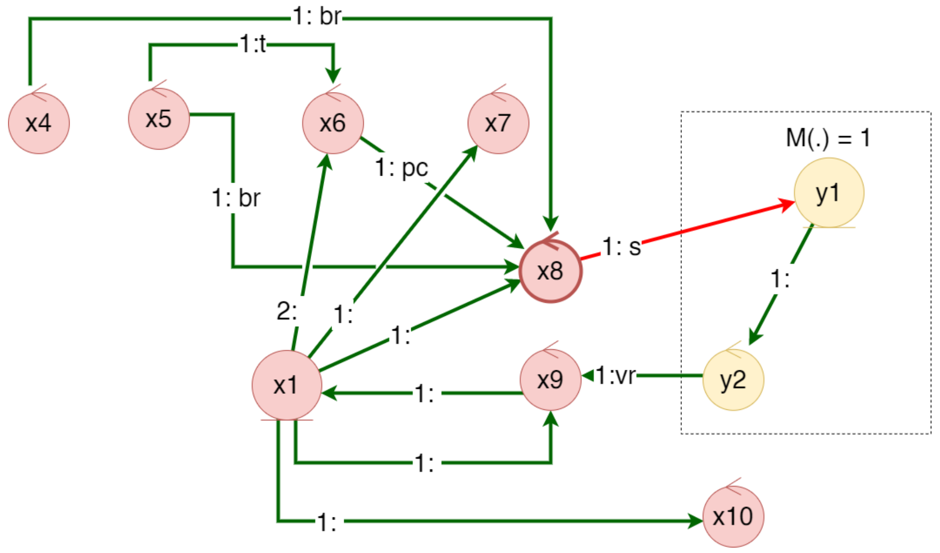


Fig. 11: Captured arcs not in the RDLT representation of Chiller System based on *S'*

the Sequence Diagram. From this summary, the traversal of Sequence Diagrams matches the sequential ordering of arc traversals in RDLT relative to the minimal activity *S'*.

4.2 Completeness of RDLT elements in generated Sequence Diagrams

Based on the demonstration of the proposed mapping, the elements of an input R are completely captured in the generated Sequence Diagrams if (1) there is exactly one activity S' in R which is minimal S', (2) R has at least one minimal activity S' and all other activities in R are elements of the activity group of S', or (3) for every minimal activity, all alternative paths are not involved with an AND-join. If the output Sequence Diagrams completely capture all elements of the input R or if assuming that some algorithm's walk does not pass any alternative path, then (1) the Case Management is consistent with the Activity, and (2) the time of traversal T in the case management is 100% consistent. The output Sequence Diagrams lack some elements from the input when (1) there is an AND-join in one of the alternative paths, and (2) the time of traversal T in Case Management is consistent until reaching an AND-Join in alternative paths. This is seen in the uncaptured elements after mapping the RDLT representation of the chiller system when there is an AND-join in one of the alternative paths vertex $x8$.

Table 1: Time of traversal T of RDLT arcs and their mapped Sequence Diagram *updateTValues* messages based on the RDLT in Figure 3

RDLT Arc	***UpdateTValues* in Sequence Diagram**	**T in Minimal Activity Profile**	**T in Case Management Profile**
$(w2, x1)$	$updateTValues(w2, x1)$	1	1
$(x1, x2)$	$updateTValues(x1, x2)$	2	2
$(x2, x4)$	$updateTValues(x2, x4)$	3	3
$(x2, x5)$	$updateTValues(x2, x5)$	3	3
$(x3, x4)$	$updateTValues(x3, x4)$	3	3
$(x3, x5)$	$updateTValues(x3, x5)$	3	3
$(x4, x6)$	$updateTValues(x4, x6)$	4	4
$(x6, x7)$	$updateTValues(x6, x7)$	5	5
$(x4, x7)$	$updateTValues(x4, x7)$	5	5
$(x5, x7)$	$updateTValues(x5, x7)$	5	5
$(x7, y1)$	$updateTValues(x7, y1)$	6	6
$(y1, y3)$	$updateTValues(y1, y3)$	7	7
$(y3, y2)$	$updateTValues(y3, y2)$	8	8
$(y2, x10)$	$updateTValues(y2, x10)$	9	9
$(x10, z)$	$updateTValues(x10, z)$	10	10

4.3 Algorithm Analysis

Theorem 1. *Algorithm 1 has a Time Complexity of $O(n^2)$ where n is the number of vertices in RDLT R.*

Proof. Let R be an input RDLT. Step 2 of the proposed mapping goes through vertices $n \in V$ to check for objects. Step 3 of the mapping goes through all arcs $(x, y) \in E$ in relation to objects and owned controllers to be included in the generated Sequence Diagram. In an RDLT, the maximum number of arcs in E is n^2 where n is the number of vertices. In the worst-case scenario, the proposed mapping goes through n vertices for generating Sequence Diagrams and n^2 arcs for mapping them in the Sequence Diagrams. Overall, the maximum running time is $O(n^2)$.

Theorem 2. *Algorithm 1 has a Space Complexity of $O(n^2)$ where n is the number of vertices in RDLT R.*

Proof. Let R be an input RDLT. Step 2 of the proposed mapping generates a Sequence Diagram for each object with owned controllers. Step 3 of the mapping generates an *updateTValues* Sequence Diagram via the *updateTValues* return message for every traversed arc. In an RDLT, the maximum number of arcs in E is n^2 where n is the number of vertices. In the worst-case scenario, the proposed mapping generates n Sequence Diagrams for objects and n^2 Sequence arcs for every arc. Overall, the maximum space complexity is $O(n^2)$.

5 CONCLUSIONS

This paper was able to provide a mapping of RDLT to Sequence Diagrams which generally performs the following steps: getting the minimal activity of the RDLT and generating Sequence Diagrams based on the minimal activity. Based on the proposed mapping, an input RDLT R can generate a set of Sequence Diagrams SD for every object with its owned controllers, for the updating of T-values after arc traversal, and a *start* diagram to reference the source. Additionally, the proposed mapping was demonstrated with a portion of the RDLT representation of the chiller system as the input. From this demonstration, it was determined that not all RDLT elements can be mapped with respect to the input's minimal activity, and the case management and the time of traversal are 100% consistent with the activity if the input elements are completely mapped or an algorithm's walk does not pass any alternative path. The Time and Space Complexity of the mapping is $O(n^2)$.

For future work, we recommend an additional validation of the mapping with the metric on the Requirements Traceability Computation of Sequence Diagrams and Robustness Diagrams to get the percentage of captured components between two models. The metric can also be adapted to fully support traceability between Sequence Diagrams and RDLT. The current mapping could also be extended so that output Sequence Diagrams are not dependent on an activity profile (minimal or not) of an RDLT or to resolve the uncaptured vertices and arcs not part of the minimal activity. This is based on the results of the demonstration of the mapping with the RDLT representation of the chiller system. Suggestions would be a novel mapping, preferably a mapping with a lower time and space complexity, or an additional checker for AND-joins in the existing mapping.

References

1. Hollingsworth, D.: Workflow Management Coalition The Workflow Reference Model. WFMC-TC-1003, v. 1.1. (1995)
2. Malinao, J.: On Building Multidimensional Workflow Models for Complex Systems Modelling. Fakultät für Informatik (Pattern Recognition and Image Processing Group) Institute of Computer Graphics and Algorithm, Technische Universität Wien, Vienna, Austria (2017)
3. Yiu, A., Garcia, J., Malinao, J., Juayong, R.: On model decomposition of multidimensional workflow diagrams. Proceedings of the WCTP (2018)
4. Rezk, A.: Theoretical and Experimental Investigation of Silica Gel/Water Adsorption Refrigeration Systems. Univ. of Birmingham (2012)
5. Lopez, J.C.L., Bayuga, M.J., Juayong, R.A., Malinao, J.A., Caro, J., and Tee, M.: Workflow models for integrated disease surveillance and response systems. Theory and Practice of Computation, 2021 Taylor and Francis Group, London, ISBN 978-0-367-41473-3. (2020)
6. Malinao, J., Tiu, K., Lozano, L. M., Pascua, S., Chua, R., Magboo, M. S. and Caro, J.: A Metric for User Requirements Traceability in Sequence, Class Diagrams, and Lines-Of-Code via Robustness Diagrams. In: Workshop on Computation: Theory and Practice 2012, Proceedings in Information and Communications Technology 7 (2013)

7. Eclipse, K., and Malinao, J.: Model Decomposition of Robustness Diagram with Loop and Time Controls to Sequence Diagrams. In: Kabassi, K., Mylonas, P., Caro, J. (eds) Novel & Intelligent Digital Systems: Proceedings of the 3rd International Conference (NiDS 2023). NiDS 2023. Lecture Notes in Networks and Systems, vol 784. Springer, Cham. (2023)
8. Malinao, J., and Juayong, RA: Classical Soundness in Robustness Diagram with Loop and Time Controls, Philippine Journal of Science, Vol. 152 (6B): 2327-2342, ISSN 0031 - 7683 (2023)
9. Object Management Group. OMG Unified Modeling Language (OMG UML) Version 2.5 (2015)
10. Visual Paradigm, What is sequence diagram? https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/