SAVE/GTS-VLT: Visual Logic Tool for Geo-Temporal Specification and Verification of Safety Requirements in Smart IoT Systems

Sunghyun Lee and Moonkun Lee

Chonbuk National University 567 Baekje-daero Deokjin-gu Jeonju-si Jeonbuk 54896, Republic of Korea moonkun@jbnu.ac.kr

Abstract. Visual representation for operational requirements for Smart IoT Systems is desirable in process algebra, since it is more intuitive than textual representation. Further visual representation for safety requirements in the systems is more desirable in real-time logic since it reduces the complexity of verification of the requirements. However it is not well known that there are common logics for such visualization. In that purpose, this paper presents a visual logic, called GTS Visual Logic, to specify and verify the geo-temporal safety requirements for Smart IoT Systems specified with a process algebra, called dT-Calculus. The calculus is used to specify the operational requirements for the systems on some conceptual geographical space. Once they are specified, a set of simulations can be performed to construct all possible execution cases for the requirements, and a set of outputs are produced in terms of processes, their actions and interactions, and dependencies on the 2-dimentional geo-temporal space. Then the visual logic is used to specify and verify all the safety requirements for the systems in terms of dependencies, especially precedencies and conditions, among all the processes, their independent actions and synchronous interactions. For feasibility, a tool, called GTS-VLT, was developed on ADOxx as a basic component of the SAVE tool suite, which is the tool set to model Smart IoT Systems, in order to demonstrate the feasibility of the logic.

Keywords: GTS Visual Logic, dT-Calculus, process algebra, SAVE, VG-GTS, ADOxx

1 Introduction

One of the main objectives of Industry 4.0 may rely on Smart IoT Systems for automation with AI and Big Data [1], and process algebra may be considered to be one of the most suitable formal methods to model the systems because of their capability of representing each IoT and its behavior as a process and its actions or interactions [2]. Further process algebras are good for visualization of IoTs and their behaviors on some geographical space, since visual representation is more intuitive than textual representation [3]. There are some of process algebras that provide with the capability

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

of visual specification of operational requirements of the IoT systems [4][5][6], but there are only few formal methods that provide with the capability of visual specification of safety requirements of the IoT systems [7].

Note that, in general, the requirements for the IoT systems can be classified into two types of requirements: 1) operation and 2) safety requirements. Mostly the operation or operational requirements are specified with process algebra, and the safety requirements are specified with logic, especially, first-order logic [8].

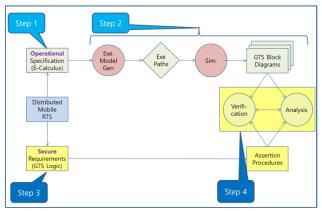


Fig. 1. A Dual Approach for Visualization of Requirements of IoT Systems

This paper presents an approach for visualization of safety requirements of the IoT systems with a visual logic, called *GTS Visual Logic* (GTS-VL), as shown in Fig. 1:

- 1) Firstly, operational requirements for the systems are specified with a process algebra, called *dT*-*Calculus*, with visualization capability, on some conceptual geographical space, shown in Step 1 of the figure.
- 2) Secondly, a set of simulations can be performed for all the possible execution cases of the operational requirements, and a set of output results are produced, which includes a set of processes, their actions and interactions, and dependencies in simulation time, represented on the 2-dimentional *geo-temporal space* (GTS), shown in Step 2 of the figure.
- 3) Thirdly, safety requirements for the systems are specified with GTS-VL with visualization capability on GTS, shown in Step 3 of the figure.
- 4) Finally, the safety requirements are verified with visual logic rules on the GTS with the requirements.

Note that GTS-VL is a first-order logic to represent all the processes, their independent actions and synchronous interactions, and, especially, dependencies among processes, actions and dependencies, visually on the space. It can reduce drastically the complexity of derivation and reduction steps of verification for the requirements over their textual representation. Note that the definition of the textual logic has been reported in [9].

In order to demonstrate the feasibility and applicability of the logic for the IoT systems, a tool, called GTS-VLT, has been developed on ADOxx as a basic component

of the SAVE tool suite, which is the tool set to model Smart IoT Systems. Fig. 2 shows the snapshot of the tool for visual verification of two simple safety requirements for an example. As noted in the figure, one of the main objectives of the visualization in the tool is WYSWYG: *What You See is What You Get*. The approach with the tool can be considered as one of the most innovative visual tools for visual specification and verification of the safety requirements for the IoT systems.

The paper is organized as follows. The visual definition of GTS-VL is described in Section 2. The GTS-VLT will be demonstrated with a simple example in Section 3. The method will be compared with other textual methods in Section 4. The SAVE tool set [5] will be briefly introduced in Section 5. Finally, conclusions and future research will be discussed in Section 6.

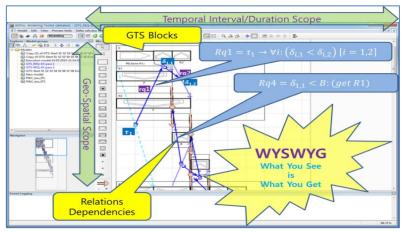


Fig. 2 Conceptual Entities in GTS-VL with an Example.

2 GTS Visual Logic

2.1 Geo-Temporal Space

GTS Visual Logic (GTS-VL) is a first-order logic defined in [9]. In the definition, System is defined as S = (P, I, C), where P, I, C are sets of processes, inclusion relations, and channels, respectively. Note that each P is defined as a sequence of timed actions defined in Fig. 3 [7]. Among the actions, communication and movement actions are synchronous interactions among processes as follows:

- 1) *Send/Receive*: Communication between processes, exchanging a message by a channel *r*.
- Movement request: Requests for movement. p and k represent priority and key, respectively.
- 3) Movement permission: Permissions for movement.

Note that *timed action* is an action with temporal properties of [r, to, e, d], where each represents *ready time, timeout, execution time*, and *deadline*, respectively. p and n are properties for periodic action or processes: p for period and n for the number of repe-

ti	ti	on.	
-	-		

$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	Action Timed action Timed process Priority Nesting Channel Choice Probabilistic choice Parallel Exception Sequence Discrete distribution Normal distribution Exponential distribution	$ \begin{array}{c} A ::= \emptyset \\ \mid r(\overline{msg}) \\ \mid r(msg) \\ \mid M \\ \mid C \\ M ::= m^p(k) P \\ \mid P m(k) \\ m ::= in \\ \mid out \\ \mid get \\ \mid put \\ C ::= new P \\ \mid kill P \\ \mid exit \end{array} $	Empty Send Receive Movement action Control action Movement request Movement permission In movement Out movement Get movement Put movement Create process Kill process Exit process
---	--	---	---

Fig. 3 Syntax of dT-Calculus

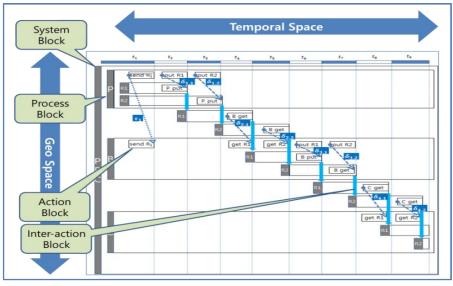


Fig. 4 Visual Definition of GTS with its Components

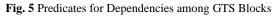
When a system is executed in a specific space in time, the system generates all the traces with the actions and interactions of the processes in the systems. These traces can be represented in its GTS as shown in Fig. 4. It consists of two dimensions: one for the geographical, and another for the temporal. There are three difference types of blocks: *System* Block (*S*), *Process* Block (*P*) and *Action* Block (*A*). By definition, a system contains processes, and a process contains actions. Further an interaction is represented as an *Interaction* Block (*I*) between two synchronous *Action* blocks of two different *Processes*.

2.2 Dependencies for Safety Requirements

Mostly the safety requirements imply the dependencies among processes, actions, and interaction block in GTS, with some additional conditions and predicates. Fig. 5

shows some of predicates in GTS Logic with visual representation on GTS. The temporal and geographical relations among action blocks are visually defined in Fig. 6 and 7. Note that each A_i implies *Action* Block, and t_j does of the temporal properties for the block defined in the previous section. All the detailed definitions of the spatial and temporal relations and the predicates are reported in the [9].

Predicates	Description	Definition	Visual Representation	
PCINQ(P,Q)	P can be in Q. Q cannot be in P	$\begin{aligned} &PCINQ(P,Q) \to \exists t \left[\exists A \left[P(A_p) = G\left(P(A_q) \right) \right] \\ & \land \neg \exists A \left[P(A_q) = G\left(P(A_p) \right) \right] \end{aligned}$	P - X - Q	
QCINP(P,Q)	P cannot be in Q. Q can be in P	$\begin{aligned} QCINP(P,Q) \to \exists t \left[\neg \exists A \left[P(A_p) = G\left(P(A_q) \right) \right] \right] \\ & \land \exists A \left[P(A_q) = G\left(P(A_p) \right) \right] \end{aligned}$	P	
$ABEFOREB(A_i, A_j)$	Action A _i should be performed before A _j in time.	$ABEFOREB \left(A_i, A_j \right) \rightarrow \left(A_i < A_j \right) \lor \left(A_i \leq A_j \right)$		
$AAFTERB(A_j, A_i)$	Action A _j should be performed after A _t in time.	$AAFTERB(A_j, A_i) \rightarrow (A_j \prec A_i) \lor (A_j \leq A_i)$	A _i) A _j	
$OI(t_1, Pred., t_2)$	Open interval	$(t_1, Pred., t_2)$	6 6	



Spatial Relation	Definition	Visual Representation
$A_i \sqsubset A_j$	$A_i \sqsubset A_j : (A_i \Subset A_j) \land (A_i \subset A_j)$	A A A
$A_i \sqsubseteq A_j$	$A_i \sqsubseteq A_j : (A_i \underline{\Subset} A_j) \land (A_i \subset A_j)$	
$A_i \equiv A_j$	$A_i \stackrel{\frown}{=} A_j : (A_i \stackrel{\frown}{=} A_j) \land (A_i \subset A_j)$	
$A_i \triangleq A_j$	$A_i \triangleq A_j : (A_i \cong A_j) \land (A_i \subset A_j)$	Aj Ai
$A_i ho A_j$	$A_i \rhd A_j : (A_i \prec A_j) \land (A_i \subset A_j)$	$ \begin{array}{c} a_{i} \\ a_{i} \\ a_{i} \\ a_{i} \end{array} $
$A_i \succeq A_j$	$A_i \trianglerighteq A_j : (A_i \preceq A_j) \land (A_i \subset A_j)$	$a_i^{i_1}$ A_i A_j $a_j^{i_1}$
$A_i \cong A_j$	$A_i \stackrel{{\scriptscriptstyle }}{=} A_j : (A_i \stackrel{{\scriptscriptstyle }}{=} A_j) \land (A_i \subset A_j)$	$ \begin{array}{c} a_{i}^{t_{b}} & a_{i}^{t_{b}} \\ a_{i}^{t_{b}} & A_{i} \\ a_{i}^{t_{b}} & a_{i}^{t_{b}} \end{array} $

Fig. 6 Visual Representation for Spatial Relations

Temporal Relations	Definition	Visual Representation
$A_i \prec A_j$	$orall t t ig[A_i \prec A_j ightarrow (extsf{t}_{ extsf{e}} < t_b') ig]$	$t_0 = t_0 = t_0$ $A_i = A_j$
$A_i \leq A_j$	$\forall t [A_i \leq A_j \rightarrow (t_e = t'_b)]$	es esté té A _l Aj
$A_i \leq A_j$	$\forall t \left[A_i \leq A_j \rightarrow (t_b < t_b') \land (t_e > t_e') \right]$	$\begin{array}{c} c_b & c_b' & c_c' & c_c' \\ \hline A_i & A_j \end{array}$
$A_i \cong A_j$	$\forall t \big[A_i \widehat{=} A_j \to (t_b = t_b') \land (t_e = t_e') \big]$	$t_{i}, t'_{i}, t_{e}, t'_{e}$ A_{i}, A_{j}
$A_i \Subset A_j$	$\forall t \big[A_i \Subset A_j \to (t_b > t'_b) \land (t_e < t'_e) \big]$	di da
$A_i \subseteq A_j$	$\forall t \big[A_i \subseteq A_j \to (t_b = t'_b) \land (t_e > t'_e) \big]$	$a_0, a_0' = a_0' = a_0'$ $A_j = A_j$
$A_i \ \overline{\sqsubseteq} \ A_j$	$\forall t \Big[A_i \ \overline{\Subset} \ A_j \rightarrow (t_b > t'_b) \land (t_e = t'_e) \Big]$	r_b r_b r_c r_c

Fig. 7 Visual Representation for Temporal Relations

3 PBC Example

This section demonstrates the applicability of GTS-VL to the IoT systems with a simple example, known as *Producer-Buffer-Consumer* (PBC).

3.1 Requirements

There are two types of requirements for the PBC example:

- 1) Operational Requirements:
 - *Producer* produces two resources, *R*1 and *R*2.
 - *Producer* stores the resources in *Buffer* in sequence.
 - Producer informs Buffer of the order of R1 and R2, or R2 and R1.
 - Consumer consumes the resources from Buffer in order.
 - The sequence of the consumption is informed to *Buffer* by *Consumer*.
- 2) Secure Requirements
 - The sequence should not be violated, since the first resource contains security information to decode the second resource.
 - The propagation between the first and the second should be less than 30 seconds.
 - The resources produced by *Producer* should be consumed by *Consumer* less than 5 minutes.

 $PBC = P[R1 \parallel R2] \parallel B \parallel C$

 $P = (PB(\overline{Send R1}).put R1.put R2 + PB(\overline{Send R2}).put R2.put R1).exit$

B = (PB(Send R1). get R1. get R2 + PB(Send R2). get R2. get R1). put R1. put R2. exit

C = get R1.get R2.exit

- R1 = P put. B get. B put. C get. exit
- R2 = P put. B get. B put. C get. exit

Fig. 8 Specification of the PBC Example in the Textual dT-Calculus.

3.2 dT-Calculus for Visualization

Fig. 8 shows the source code for the PBC example in the textual dT-Calculus, e-specifically for the operational requirements. The basic descriptions are as follows:

- 1) As shown in the code, there are three processes in the system BPC: P, B and C.
- 2) And there are two resource processes defined in *P* as child process: *R*1 and *R*2.
- 3) There are two channels:
 - i. *PB*: a communication channel between *P* and *B*.
 - ii. *CB*: a communication channel between *C* and *B*.
- 4) The *PBC* System operates as follows:
 - i. *PB* contacts with *B* through *PB*, nondeterministically, that is, with a choice operation (+), in order to send the resource in the sequence of *R*1 followed by *R*2, or of *R*2 followed by *R*1.
 - ii. Once the sequence is determined by the choice, P releases the resources in that sequence off the boundary of P, synchronously, with the synchronous passive movement operations between P and R, that is, the *put* R of P and the P *put* of R.

- iii. Once the resources are released off P by P, B gets the resource in B in that sequence of the release, synchronously, with the synchronous passive movement operations between B and R, that is, the *get* R of B and the B *get* of R.
- iv. Once the resources are moved into B by B, B releases the resource off B in the sequence R1 followed by R2, synchronously, with the synchronous passive movement operations between B and R, that is, the *put* R of B and the B *put* of R.
- v. Once the resources are released off *B* by *B*, *C* gets the resource in *C* in the sequence of the release, synchronously, with the synchronous passive movement operations between *C* and *R* :the *get R* of *C* and the *C get* of *R*.

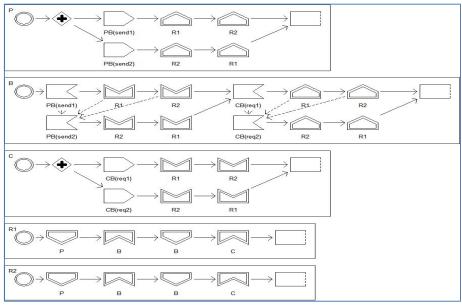


Fig. 9 The ITS Views of the PBC Example

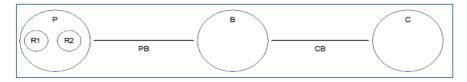


Fig. 10 The ITL View of the PBC Example

There are two forms of visualization for the example as follows:

- 1) ITS (*In-The-Small*) View: It is a process view to visualize the above description in 4). A set of the views for all the processes in the example is shown in Fig. 9.
- 2) ITL (*In-The-Large*) View: It is a system view to visualize the above description between 1) and 3). The view for the example is shown in Fig. 10.

Note that the views in the figures are the snapshots of the example for visual specification of the example with the tool developed by authors, namely, SAVE/GTS-VLT, on the ADOxx Meta-Modeling Platform. There are two ways of specifying the requirements in dT-Calculus:

- 1) Textual specification: The specification can be input to SAVE just as shown in Fig. 8, and ITL and ITS views are automatically generated by SAVE.
- 2) Visual specification: The requirements can be directly specified in the graphical editor for ITL and ITS views in SAVE.

Once the specification is done, SAVE generates all the possible execution paths of the system. Fig. 11 shows that there are four possible paths in the PBC example: One for the sequence of R1 and R2, another for that of R2 and R1, and two deadlock cases.

3.3 GTS Visual Logic and Safety Requirements

Fig. 12 shows the simulation output of the first path for the execution paths of the example shown from Fig. 11. All the elements of the GTS blocks are shown in the figure: *System, Process* and *Action* Blocks. Further Interactions are shown in the edges between two synchronous action blocks, as follows:

- τ: Communication

 τ₁ = (P:PB, (Send R1), B:PB(Send R1))
 τ₂ = (P:PB, (Send R2), B:PB(Send R2))

 δ: Movements

 δ_{1,1} = (P:put R1, R1: P put)
 δ_{1,2} = (P:put R2, R2: P put)
 δ_{2,1} = (B:get R1, R1: B get)
 δ_{2,2} = (B:get R2, R2: P get)
 δ_{3,1} = (B:put R1, R1: B put)
 δ_{3,2} = (B:put R2, R2: B put)
 - $\delta_{4,1} = (C:get R1, R1: C get)$ $\delta_{4,2} = (C:get R2, R2: C get)$

The figure also shows a couple of the safety requirements for the PBC example from Section 3.1. Note that the requirement edges are of the predicates shown in Fig. 5. The whole requirements are as follows:

- $Rq1 = \tau_1 \rightarrow \forall i: (\delta_{i,1} < \delta_{i,2}) [i = 1,2]$: After τ_1 , that is, the communication for exchange of the resources in the sequence of R1 and R2, all the movements actions of the resources, that is, $\delta_{i,j}$, must follow that sequence.
- $Rq2 = \tau_2 \rightarrow \forall i: (\delta_{i,2} < \delta_{i,2}) \ [i = 1,2]:$ Similarly, After τ_2 , that is, the communication for exchange of the resources in the sequence of *R*2 and *R*1, all the movements actions of the resources, that is, $\delta_{i,j}$, must follow that sequence.
- $Rq3 = \tau_1 \lor \tau_2 < \delta_{1,1} \land \delta_{1,2}$: After τ_1 or τ_2 , that is, the sequence of *R*1 and *R*2 is determined between *P* and *B*, both resources can be moved from *P* to *B*.
- $Rq4 = \delta_{1,1} < B: (get R1):$ Once R1 is moved off P by P, B can get it into B.
- $Rq5 = \delta_{1,2} < B$: (get R2): Once R2 is moved off P by P, B can get it into B.
- $Rq6 = \delta_{2,1} < B: (put R1):$ Once R1 is moved into B by B, B can put it off B.
- $Rq7 = \delta_{2,2} < B: (put R2):$ Once R2 is moved into B by B, B can put it off B.
- $Rq8 = \delta_{3,1} < C: (get R1):$ Once R1 is moved off B by B, C can get it into C.
- $Rq9 = \delta_{3,2} < C$: (get R2): Once R2 is moved off B by B, C can get it into C.

20

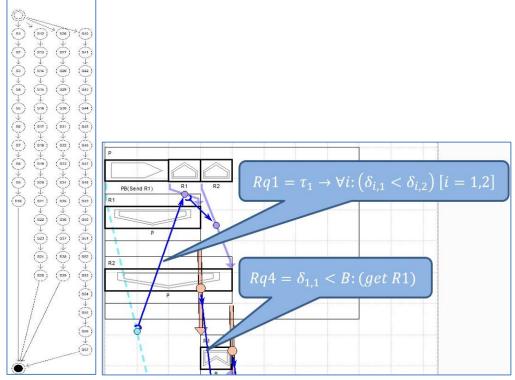


Fig. 11 Execution Paths

Fig. 12 Simulation Output on GTS

Note that the results of the verification of the requirements are automatically visualized in the figure as follows:

- Blue: A requirement edge in the figure is blue if it is satisfied.
- Red: A requirement edge in the figure is blue if it is not satisfied.

In case that a requirement is failed, the cause of the failure is listed in the workspace of the tool, which is the bottom section of the window of the tool in the figure

4 Comparative Analysis

GTS-VL is a first-order logic deal with space and time for dT-Calculus. Compared with other geo-temporal logics, it has some of advantages over them as follows:

- Temporal-based logics: *Linear Temporal Logic* (LTL)[10], *Computational Tree Logic* (CTL)[11], and *Real-time Logic* (RTL)[12].
 - LTL is a logic that can be used to analyze one time branch.
 - LTL uses time operators like *always*, *eventually* and *release* to represent time.
 - CTL is a logic that can be used to analyze multiple time branches.
 - CTL uses time operators like all, *exist*, *next*, *globally* to represent multiple branch times.
 - RTL specifies a system using actions and events.
 - RTL represents the time using formulas and operators (time, stop, state variable transitions, external events and global time).

- Disadvantages over GTS-VL:
 - These temporal-based logics have limitations to represent movements.
 - No visual capability to represent graphically specification and verification of systems because they are based on text representation.
- Spatial-based logics: *Region and Connection calculus* (RCC)[13] and *Cardinal Direction Relations* (CRD)[14].
 - RCC is a spatial logic that distinguishes each space by defining a relationship between each space.
 - CRD is a spatial logic based on coordinate system, and it is spatial logic that distinguishes each space according to coordinates.
- Disadvantages over GTS-VL:
 - The space of a process can be represented visually, but its mobility is not.
 - No visual capability to represent temporal properties of processes in specification and analysis of systems because they are based on textual representation only.

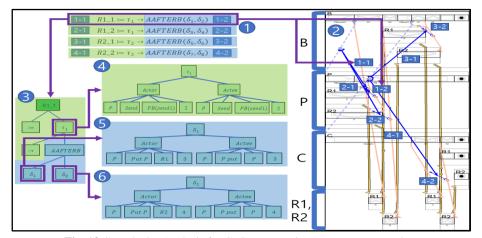


Fig. 13 Complexity Analysis for GTS-VL Requirements to Its Textual Form

Table 1 Reduction of Complexity from the Textual to the Visua	al
---	----

Textual Structure of Requiremt RI	Textal Complexity	Visual Complexity	Complexity
	(TC = $O + P + t_n + C$)	(# of Edges)	Reduction
$\begin{array}{l} R1_1:=\tau_1 \to AAFTERB(\delta_1, \delta_2) \\ \tau_1:=(P: Send, PB(send1), 2), (B: Receive, PB(send1), 2) \\ \delta_1:=(P: put P, R1, 3), (R1: P put, P, 3) \\ \delta_2:=(P: put P, R2, 4), (R2: P put, P, 4) \end{array}$	5+1+2+26=34	I+I=2	32

In order to demonstrate the advantages of the GTS-VL approach, we analyze the complexity of the analysis and verification process for GTS-VL to its textual representation. Fig. 13 shows the first requirement from the PBC Example in the GTS-VL on the GTS output of the simulation for the first execution path of the example, from Fig. 12. The right side of the figure is the syntax tree of the requirement in the textual representation, consisting of τ 's and δ 's, which are also structured in syntax trees of all the system, process and action blocks with temporal properties. Such trees generate severe complexity during analysis and verification processes of the requirement in

22

the form of textual representation. The left side of the figure is the final result that SAVE/GTS-VLT generates at the end of the analysis and verification process for GTS-VL over the simulation output on GTS. It drastically simplifies the complexity, as Table 1 shows. In general, it is well known that the visual method for communication of information is better than the textual method [15].

5 SAVE

SAVE is a suite of tools to specify and analyze the IoT systems with dTP-Calculus. It is developed on the ADOxx Meta-Modeling Platform. SAVE consists of basic five components: Specifier, *Execution Model Generator* (EMG), Simulation, Analyzer and Verifier. Specifier is a tool to specify the IoT systems with dT-Calculus, visually in the diagrammatic representations. EMG is a generator to construct all the possible execution paths for the system specified in Specifier. Simulation is the main engine to execute each execution path selected from the execution model in EMG. Analyzer and Verifier are tools to analyze and verify the safety requirements of the system specified in GTS-VL. The basic tool of SAVE/GTS-VLT consists of these two components. All the figures shown in the paper are the snapshots of SAVE/GTS-VLT generated for the PBC Example. The SAVE tool is an open SW that has been developed as a project within the Open Models Laboratory (OMiLAB) [16], an open environment for the conceptualization of domain-specific conceptual modeling languages [17]. The tool can be downloaded with a manual for the example [18].

6 Conclusion

This paper presented a visual method to specify and verify geo-temporal requirements for dT-Calculus, based on GTS-VL. Further SAVE/GTS-VLT was developed to demonstrate the feasibility of the method, based on the ADOxx Meta-Modeling Platform. With the tool, a small example, PBC, was selected for applicability of the method in steps by generating all necessary artefacts for the example: ITL and ITS views, GTS simulation output, GTS-VL requirements. The method with the tool may be considered to be one of the most innovative approaches to specify and verify the operation and safety requirements of Smart IoT Systems. The future research will include development of requirements analysis and verification methods for Smart IoT examples in field for Industry 4.0 in order to show its efficiency and effectiveness.

Acknowledgment

This work was supported by Basic Science Research Programs through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2010-0023787), Space Core Technology Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and Future Planning(NRF-2014M1A3A3A02034792), Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Educa-

tion(NRF-2015R1D1A3A01019282), and Hyundai NGV, Korea. and Research Fund from Chonbuk National University (2018~2019).

References

- 1. K. Rob. The real-time city? Big data and smart urbanism. GeoJournal. vol 79. Springer. 2014.
- Y. Choe, et al. Process Model to Predict Nondeterministic Behavior of IoT Systems. The 11th IFIP WG 8.1 working conference on the Practice of Enterprise Modelling (PoEM). 2018. pp.1-12.
- Y. Choe, et al. SAVE: an environment for visual specification and verification of IoT. IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW). 2016. pp.1-8.
- L. Cardelli, et al. Mobile Ambients. In International Conference on Foundations of Software Science and Computation Structure. Springer. 1998. pp.140-155.
- J. On, et al. A Study on Scheduler Based on CARDMI Process Algebra for Automated Control of Emergency Medical System. Proceedings of the Korean Information Science Society Conference. Korean Institute of Information Scientists and Engineers. 2008. pp.65-70.
- J. On. et al. A graphical language to integrate process algebra and state machine views for specification and verification of distributed real-time systems. IEEE 36th Annual Computer Software and Applications Conference Workshops. 2012. pp.218-223.
- 7. Y. Choe, et al. dT-Calculus: A Process Algebra to Model Timed Movements of Processes. International Journal of Computers. 2017. pp.53-62.
- Smullyan, Raymond R. *First-order logic*. Springer Science & Business Media. Vol 43. 2012.
- 9. Y. Choe, et al. A Dual Method to Model IoT Systems. International Journal of Mathematical Models and Methods in Applied Sciences. 2016. pp.201-219.
- 10. Clarke, et al. Design and synthesis of synchronisation skeletons using branching time Temporal Logic. Workshop on Logic of Programs. Springer. 1981. pp.52-71.
- 11. Huth, et al. Logic in Computer Science: Modelling and reasoning about systems. Cambridge university press. 2004.
- F. Jahanian, et al. Modechart: A specification language for real-time systems. IEEE Transactions on Software engineering. vol 20. 1994. pp.933-947.
- Cohn, et al. Qualitative spatial representation and reasoning with the region connection calculus. GeoInformatica. 1993. pp.275-316.
- Frank, et al. Qualitative spatial reasoning about distances and directions in geographic space. Journal of Visual Languages and Computing. vol 3. 1992. pp.343-371.
- Burkhard, et al. Learning from architects: the difference between knowledge visualization and information visualization. Eighth International Conference on Information Visualisation. IEEE. 2004. pp.519-524.

24

- 16. Bork, et al. An Open Platform for Modeling Method Conceptualization: The OMiLAB Digital Ecosystem. Communications of the Association for Information Systems. vol 44. 2019. pp.673-697. https://doi.org/10.17705/1CAIS.04432.
- 17. Karagiannis, et al. *Domain-specific conceptual modeling*. Springer International Publishing. 2016.
- 18. https://austria.omilab.org/psm/content/save/info