# Process Model to Predict Nondeterministic Behavior of IoT Systems

Yeongbok Choe and Moonkun Lee

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

**Abstract.** Process algebra is one of the best suitable formal methods to model IoT systems, supporting formal specification and analysis. However when IoT systems are under certain uncertainty, it is necessary to model their unpredictability based on the uncertainty. In other words, process algebra should provide specific features to model predictable behaviors to cover this kind of uncertainty, based on probability concept. There have been several process algebras with probability, such as, PAROMA, PACSR, etc. However they are not well suitable for the smart IoT with complex uncertainty, since they are simply based only on discrete model or exponential model. Consequently, they allows only simple or targeted probability to be specified and analyzed, and they only reveal simple or targeted behaviors of the IoT systems. In order to handle such limitations, the paper presents a new formal method, called dTP-Calculus, extended from the existing dT-Calculus with the discrete, normal, exponential, and the uniform probability models. It provides all the possible probability features for the smart IoT system with complex uncertainty. The specification of the modeling will be simulated statistically for the IoT systems, and further the simulation results will be analyzed for probabilistic properties of the systems. In order to demonstrate practicality of the approach, a tool set for the calculus has been implemented in the SAVE tool set, developed on the ADOxx Meta-Modeling Platform, including Specifier, Analyzer and Verifier. It can be considered as one of the most innovative methods with the practical tools.

**Keywords:** dTP-Calculus, process algebra, probability, SAVE, ADOxx

## 1 Introduction

Internet of Things (IoT) is one of the most important requirements for Industry 4.0. Especially, Industrial Internet of Things (IIoT) requires correctness and safety of system operations performed by people, in order to guarantee expected industrial production output, as well as to provide safety protection to the people [1]. In addition, it is necessary to provide the capability to predict a variety of system behaviors for the safety protection, detectable from probabilistic system analysis. In order to verify formally such safety requirements of the systems with respect to the characteristics of

IoT or IIoT, it is desirable to apply formal methods to the IoT or IIoT systems for formal specification and analysis.

Formal methods are used to verify various properties of the IoT systems, for example, communication protocols and security of the systems [2, 3, 4]. However there is lack of research to verify behavior of the systems in terms of movements of the IoT devices. In order to guarantee safety of the systems, verification of the behavior, as well as of security, is strongly required [5]. Therefore process algebra can be used to verify the IoT systems with the properties of distributedness, mobility and real-time.

Generally, it is very difficult to predict various system behaviors with process algebra because of its nondeterministic choice operations. In order to overcome the limitation, a new type of process algebras, such as, PAROMA [6] and PACSR [7], were defined based on the notion of probability. However these process algebras have limitations to specify and analyze very complex systems like IoT or IIoT, since PACSR is capable of specifying probabilistic choice operations in only one form of probability model, that is, discrete model and PAROMA is only based on exponential distribution model.

In the IoT systems, various behaviors cannot be predicted from the explicitly fixed or exponentially distributed only probabilistic branch of choice operations, since the systems behave differently according to different specification and requirements. Therefore it is necessary to apply various probabilistic models based on normal, exponential, or other distributions in order to predict various behaviors, instead of being on the fixed or exponential models.

In order to handle this kind of limitations, this paper proposes *dTP-Calculus*, a probabilistic process algebra extended from dT-Calculus [8] with a set of probability models, in order to specify and analyze probabilistic behaviors of the IoT systems. Note that dT-Calculus was originally designed by the authors to specify a variety of timed movements of processes on virtual geographical space.

Practically, in order to demonstrate the feasibility and applicability of the calculus to the IoT systems, a set of tools, known as SAVE, have been developed on the ADOxx Meta-Modeling Platform. SAVE consists of Specifier, Analyzer and Verifier. And an example, known as *Smart Emergency Evacuation System* (SEES), has been applied to SAVE for specification and analysis in the calculus. It can be considered one of the most practical tools applied to the IoT system for Industry 4.0.

The paper is organized as follows. The basic definition of dTP-Calculus is described in Section 2. The probabilistic models in the calculus are defined and analyzed with the example in Section 3, and the SAVE tool set [9] to model the calculus is described in Section 4. Finally conclusions and future research will be discussed in Section 5.

# 2 dTP-Calculus

## 2.1 Syntax

dTP-Calculus is a process algebra extended from existing dT-Calculus in order to define probabilistic behavioral property of processes on the choice operations. Note that dT-Calculus is the process algebra originally designed by the authors of the paper in order to specify and analyze various timed movements of processes on virtual geographical space. The syntax of dTP-calculus is shown in Fig. 1.

| | | | |
|---|---|---|---|
| $P ::= A$ | Action | $A ::= \emptyset$ | Empty |
| $\mid A^{p,n}_{[r,to,e,d]}$ | Timed action | $\mid r(\overline{msg})$ | Send |
| $\mid P^{p,n}_{[r,to,e,d]}$ | Timed process | $\mid r(msg)$ | Receive |
| $\mid P_{(n)}$ | Priority | $\mid M$ | Movement action |
| $\mid P[Q]$ | Nesting | $\mid C$ | Control action |
| $\mid P\langle r\rangle$ | Channel | $M ::= m^p(k)\ P$ | Movement request |
| $\mid P + Q$ | Choice | $\mid P\ m(k)$ | Movement permission |
| $\mid P\{c\} +_F Q\{c\}$ | Probabilistic choice | $m ::= in$ | In movement |
| $\mid P \parallel Q$ | Parallel | $\mid out$ | Out movement |
| $\mid P\backslash E$ | Exception | $\mid get$ | Get movement |
| $\mid A \cdot P$ | Sequence | $\mid put$ | Put movement |
| $F ::= D$ | Discrete distribution | $C ::= new\ P$ | Create process |
| $\mid N(\mu, \sigma)$ | Normal distribution | $\mid kill\ P$ | Kill process |
| $\mid E(\lambda)$ | Exponential distribution | $\mid exit$ | Exit process |
| $\mid U(l, u)$ | Uniform distribution | | |

**Fig. 1.** Syntax of dTP-Calculus

Each part of the syntax is defined as follows:

1) *Action*: Actions performed by a process.
2) *Timed action*: The execution of an action with temporal restrictions. The temporal properties of [*r, to, e, d*] represent *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 repetition.
3) *Timed process*: Process with temporal properties.
4) *Priority*: The priority of the process *P* represented by a natural number. The higher number represents the higher priority. Exceptionally, 0 represents the highest priority.
5) *Nesting*: *P* contains *Q*. The internal process is controlled by its external process. If the internal process has a higher priority than that of its external, it can move out of its external without the permission of the external.
6) *Channel*: A channel *r* of *P* to communicate with other processes.
7) *Choice*: Only one of *P* and *Q* will be selected nondeterministically for execution.
8) *Probabilistic choice*: Only one of *P* and *Q* will be selected probabilistically. Selection will be made based on a probabilistic model specified with *F*, and the condition for each selection will be defined with *c*.
9) *Parallel*: Both *P* and *Q* are running concurrently.

10) *Exception*: *P* will be executed. But *E* will be executed in case that *P* is out of timeout or deadline.
11) *Sequence*: *P* follows after action *A*.
12) *Empty*: No action.
13) *Send/Receive*: Communication between processes, exchanging a message by a channel *r*.
14) *Movement request*: Requests for movement. *p* and *k* represent priority and key, respectively.
15) *Movement permission*: Permissions for movement.
16) *Create process*: Creation of a new internal process. The new process cannot have a higher priority than its creator.
17) *Kill process*: Termination of other processes. The terminator should have the higher priority than that of the terminatee.
18) *Exit process*: Termination of its own process. All internal processes will be ter minated at the same time.

### 2.2 Probability

There are 4 types of probabilistic models to specify probabilistic choice as follows. Each model may require variables to be used to define probability:

1) Discrete distribution: It is a probabilistic model without variable. It simply defines specific value of probability for each branch of the choice operation. There are some restrictions. For example, the summation of the probability branches cannot be over 100%.
2) Normal distribution: It is a probabilistic model based on the normal distribution with the mean value of $\mu$ and the standard deviation of $\sigma$, whose density function is defined by $\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$.
3) Exponential distribution: This is a probabilistic model based on the exponential distribution with frequency of $\lambda$, whose density function is defined by $\lambda e^{-\lambda x}$.
4) Uniform distribution: This is a probabilistic model based on the uniform distribution with the lower bound $l$ and the upper bound $u$, whose density function is defined by $\begin{cases} \frac{1}{u-l} & (l \le x \le u) \\ 0 & \end{cases}$.

Once a model is defined, the conditions for the selection of the branches should be specified. There are differences in the specifications for the conditions in the models. In the discrete distribution, the values of the probabilities are specified directly in the condition as shown in Expression (1).

$$P\{0.7\}+_D Q\{0.3\} \tag{1}$$

In other cases, that is, other distribution models, such as, normal, exponential and uniform, a set of specific ranges are to be specified in the conditions. The following

Expressions (2), (3) and (4) are the examples for normal distribution, exponential distribution and uniform distribution, respectively.

$$P(v > 52) +_{N(50,5)} Q(v \leq 52) \tag{2}$$

$$P(v > 2.5) +_{E(0.33)} Q(v \leq 2.5) \tag{3}$$

$$P(v > 5) +_{U(3,7)} Q(v \leq 5) \tag{4}$$

During the specification, it is very important to check that the summation of the probabilities in the conditions on the braches should be less than or equal to 1. In the discrete distribution, the summation should be 1, since the values are specified directly. However, in other case, such as, normal, exponential and uniform distributions, the conjunction of all the ranges in the condition on the braches should be the set of the real numbers, since the ranges are specified in the conditions. Note that the restriction on is based on the facts that, if the summation is less than 1, it is possible for no branch to be selected, and, if it is greater than 1, it is possible for some branches to be selected at the same time, violating the notion of selection on the choice.

# 3 Example

This section demonstrates the applicability of dTP-Calculus to the IoT systems with an example, known as Smart Emergency Evacuation System (SEES).

## 3.1 Specification

Fig 2 shows the specification of the SEES example in dTP-Calculus. The processes in the example as follows:

1) *Control System*: The main process to control other processes in case of fire.
2) *Sensor*: The process to sense fires on *Stair A* and *Stair B*.
3) *Building*: The process to represent the building where the fire occurs. It contains all the related processes in the building, except *911*.
4) *Floor*: The process to represent the floors in the building. There are two floors: *1st Floor* and *2nd Floor*. And two persons, *P1* and *P2*, in *2nd Floor*.
5) *Stair*: The process to represent stairs. There are two stairs: *Stair A* and *Stair B*. A fire occurs at one of the stairs.
6) *Person*: The processes to represent the persons in the building: *P1* and *P2*.
7) *911*: The process to perform fire extinction and people rescue.

SEES performs its mission in order as follows:

1) A fire occurs on *1st Floor* or the *2nd Floor*.
2) *Sensor* detects the fire and sends a signal to *Control System*.
3) *Control System* informs *Person* of the fire and shows the escape route. And it sends the signal to *911*.
4) Each *Person* may get out of *Building* safely, or be confined on *2nd Floor*.

5) *Building* detects the escape of *Person*, and sends the information of the escaped to *Control System*.
6) *Control System* sends the information of the confined to *911*.
7) *911* enters *Building*, extinguishes the fire on *2nd Floor*, and rescues *Person*.

$$Sys \coloneqq Building[Control\ System|StairA[SensorA]||StairB[SensorB]\ |1st\ floor|2nd\ floor[P1|P2]]|\ 911;$$

$$Control\ System \coloneqq (CS(FireA)_{[0,-,1.3]} \cdot P1(\overline{StairB}) \cdot P2(\overline{StairB}))\backslash(CS(FireB) \cdot P1(\overline{StairA}) \cdot P2(\overline{StairA}))$$
$$\cdot CE(\overline{Call}) \cdot CS(P1)_{[0,-,1.7]}\backslash \mathrm{CE}(\overline{P1}) \cdot CS(P2)_{[0,-,1.4]}\backslash \mathrm{CE}(\overline{P2});$$

$$SensorA \coloneqq (SA(\overline{Fire})_{[0,-,1.2]} \cdot CS(\overline{FireA}))\backslash\emptyset_3;$$

$$SensorB \coloneqq (SB(\overline{Fire})_{[0,-,1.2]} \cdot CS(\overline{FireB}))\backslash\emptyset_3;$$

$$P1 \coloneqq (P1(StairB)_{[0,-,1.3]} \cdot (\emptyset.\boldsymbol{RC(P1).out\ 2nd.out\ Building\{v < 2.5\} +_{N(5,3)} out\ 2nd.in\ StairB.out\ StairB.in\ 1st.out\ 1st.out\ Building\{v \geq 2.5\}}))$$
$$\backslash(P1(StairA) \cdot (\emptyset.RC(P1).out\ 2nd.out\ Building\{v < 2.5\} +_{N(5,3)} out\ 2nd.in\ StairA.out\ StairA.in\ 1st.out\ 1st.out\ Building\{v \geq 2.5\}));$$

$$P2 \coloneqq (P2(StairB)_{[0,-,1.4]} \cdot (\emptyset.\boldsymbol{RC(P2).out\ 2nd.out\ Building\{v < 2.5\} +_{N(5,8)} out\ 2nd.in\ StairB.out\ StairB.in\ 1st.out\ 1st.out\ Building\{v \geq 2.5\}}))$$
$$\backslash(P2(StairA) \cdot (\emptyset.RC(P2).out\ 2nd.out\ Building\{v < 2.5\} +_{N(5,8)} out\ 2nd.in\ StairA.out\ StairA.in\ 1st.out\ 1st.out\ Building\{v \geq 2.5\}));$$

$$StairA \coloneqq (P1\ in_{[0,-,1.10]} \cdot P1\ out)\backslash\emptyset \cdot (P2\ in_{[0,-,1.3]} \cdot P2\ out)\backslash\emptyset;$$

$$StairB \coloneqq (P1\ in_{[0,-,1.8]} \cdot P1\ out)\backslash\emptyset \cdot (P2\ in_{[0,-,1.3]} \cdot P2\ out)\backslash\emptyset;$$

$$1st\ floor \coloneqq (P1\ in_{[0,-,1.11]} \cdot P1\ out)\backslash\emptyset \cdot (P2\ in_{[0,-,1.3]} \cdot P2\ out)\backslash\emptyset;$$

$$2nd\ floor \coloneqq P1\ out_{[0,-,1.8]}\backslash\emptyset \cdot P2\ out_{[0,-,1.3]}\backslash\emptyset \cdot 911\ in \cdot P1\ out_{[0,-,1.5]}\backslash\emptyset \cdot P2\ out_{[0,-,1.3]}\backslash\emptyset \cdot 911\ out;$$

$$Building \coloneqq (\underline{\boldsymbol{SA(\overline{Fire})\{0.5\} +_D SB(\overline{Fire})\{0.5\}}}) \cdot \left(P1\ out_{[0,-,1.13]} \cdot CS(\overline{P1})\right)\backslash\emptyset \cdot \left(P2\ out_{[0,-,1.3]} \cdot CS(\overline{P2})\right)\backslash\emptyset$$
$$\cdot 911\ in \cdot P1\ out_{[0,-,1.5]}\backslash\emptyset \cdot P2\ out_{[0,-,1.3]}\backslash\emptyset \cdot 911\ out;$$

$$911 \coloneqq \mathrm{Ce(Call)} \cdot CE(P1)_{[0,-,1.10]}\backslash((CE(P2)_{[0,-,1.3]} \cdot in\ Building \cdot in\ 2nd \cdot RC(\overline{P2}) \cdot out\ 2nd \cdot out\ Building)$$
$$\backslash in\ Building \cdot in\ 2nd \cdot out\ 2nd \cdot out\ Building)$$
$$\cdot (CE(P2)_{[0,-,1.8]} \cdot in\ Building \cdot in\ 2nd \cdot RC(\overline{P1}) \cdot RC(\overline{P2}) \cdot out\ 2nd \cdot out\ Building)$$
$$\backslash(in\ Building \cdot in\ 2nd \cdot RC(\overline{P1}) \cdot out\ 2nd \cdot out\ Building)$$

**Fig. 2.** SEES Specification

A fire occurs at *Stair A* or *Stair B* in *Building*, and each *Person* may or may not escape from *Building*. In SEES, three kinds of probabilistic choices are specified. Expressions (5), (6) and (7) are the probabilistic choices of *Building*, *P1* and *P2*, respectively - refer the underlined segments of the code in Fig. 2.

$$SA(\overline{Fire})\{0.5\} +_D SB(\overline{Fire})\{0.5\} \tag{5}$$

$$\emptyset \ldots \{v < 2.5\} +_{N(5,3)} out\ 2nd \ldots \{v \geq 2.5\} \tag{6}$$

$$\emptyset \ldots \{v < 2.5\} +_{N(5,8)} out\ 2nd \ldots \{v \geq 2.5\} \tag{7}$$

*Building* is of discrete distribution, and *P1* and *P2* are of normal distribution. The range of the choices in *P1* and *P2* are same, but the values of $\sigma$ are different. It implies that the escape and the non-escape, that is, confinement, of each *Person* are specified in the probabilistic choices, and different probabilistic values are applied to each *Person*.

### 3.2 Analysis

It is possible to analyze the probability of each occurrence of behaviors, that is, escaping or rescuing, by extracting the system behaviors from the SEES example. Even though it is possible to calculate the probability of each occurrence mathematically, in dTP-Calculus, the probability is obtained and analyzed by simulating a significant amount of trials for each occurrences based on the specifications.

From the specification, it is possible to analyze all the possible execution paths from the example, as Fig. 3 shows. There are total 8 possible paths, and no system fault, including deadlock, does occur in each cases.

Each path implies each possible system behavior, as shown in Table 1, based on the following meanings:

1) *Fire*: The location where the fire occurred.
2) *In*: *P1* and *P2*, confined in *Building*.
3) *Out*: *P1* and *P2*, escaped from *Building*.

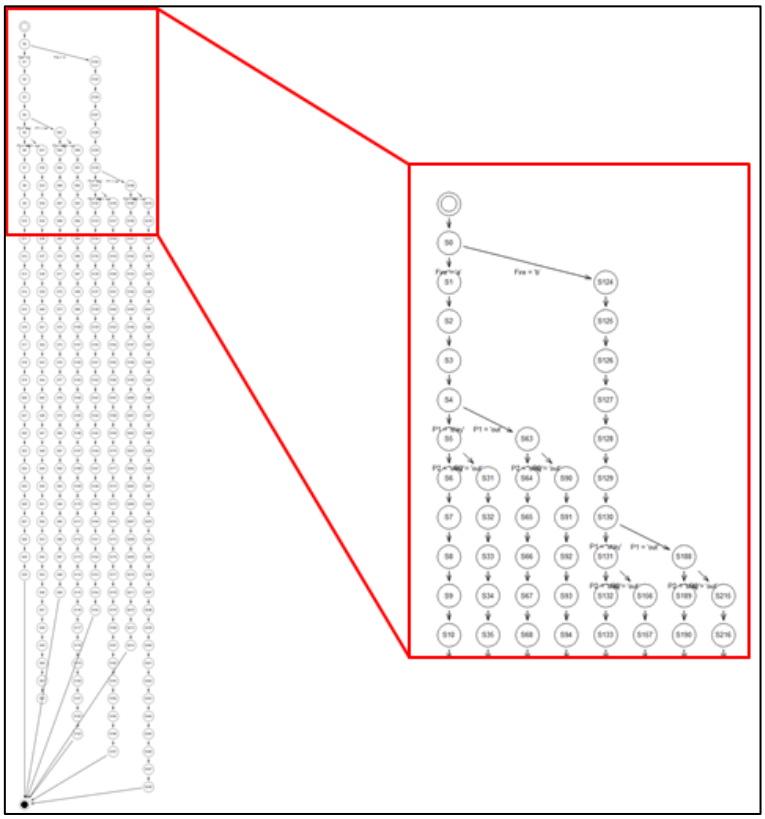

**Fig. 3.** Probabilistic Execution Paths of SEES

**Table 1.** Meaning of Paths

| | Path 1 | Path 2 | Path 3 | Path 4 | Path 5 | Path 6 | Path 7 | Path 8 |
|---|---|---|---|---|---|---|---|---|
| Fire | StairA | StairA | StairA | StairA | StairB | StairB | StairB | StairB |
| P1 | Stay | Stay | Out | Out | Stay | Stay | Out | Out |
| P2 | Stay | Out | Stay | Out | Stay | Out | Stay | Out |

Now it is possible to perform probabilistic analysis mathematically for each path. The probability in *Building* is specified directly in the condition of its probabilistic choice, but the probabilities in *P1* and *P2* are to be calculated from their distributions. According to the normal distribution density function, Expression (6) and (7) can be changed to Expression (8) and (9), respectively.

$$\emptyset \dots \{0.2023\} +_D out\ 2nd \dots \{0.7977\} \tag{8}$$

$$\emptyset \dots \{0.3773\} +_D out\ 2nd \dots \{0.6227\} \tag{9}$$

Consequently the probability for each path can be determined to be those shown in Table 2.

**Table 2.** Probabilities of Paths by Probability Function

| Path | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| % | 3.82 | 6.3 | 15.05 | 24.83 | 3.82 | 6.3 | 15.05 | 24.83 |

Even if the probabilities are determined, it is necessary to analyze if the system works according to the probabilities, in order to predict the execution of the system. For that purpose, simulation can be performed for each path of the execution. In our approach, Path Analysis of the ADOxx Meta-Modeling Platform [10] was utilized for the simulation. In simulation, it is possible to define a number of trials for the execution path in order to analyze its probability. Fig. 4 shows probabilistic analysis by simulation on the 8th path.

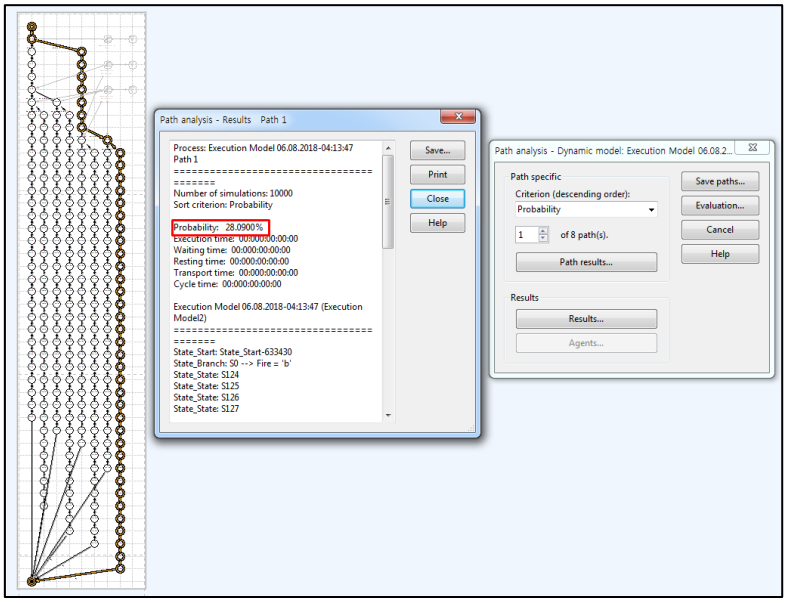


**Fig. 4.** Probability Analysis

Finally, Table 3 shows the result of the simulation in the tool. The table shows different values for different trails. The value of the probability changes with respect to the number of simulation trials. It can be noticed that the value of the probability becomes close to that of the probability in Table 2, as the number increases. Through the simulation, it can be checked whether the real system can execute properly according to the specified probabilities.

**Table 3.** Simulation Result

| Number of Simulation | Probability (%) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Path 1 | Path 2 | Path 3 | Path 4 | Path 5 | Path 6 | Path 7 | Path 8 |
| 1,000 | 3.5 | 6.2 | 14.2 | 25 | 3.5 | 7.5 | 14.4 | 25.7 |
| 1,000,000 | 3.79 | 6.32 | 15.04 | 24.86 | 3.82 | 6.32 | 14.98 | 24.87 |

## 4 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. Fig 5 shows the basic tools and system architecture of SAVE on ADOxx.

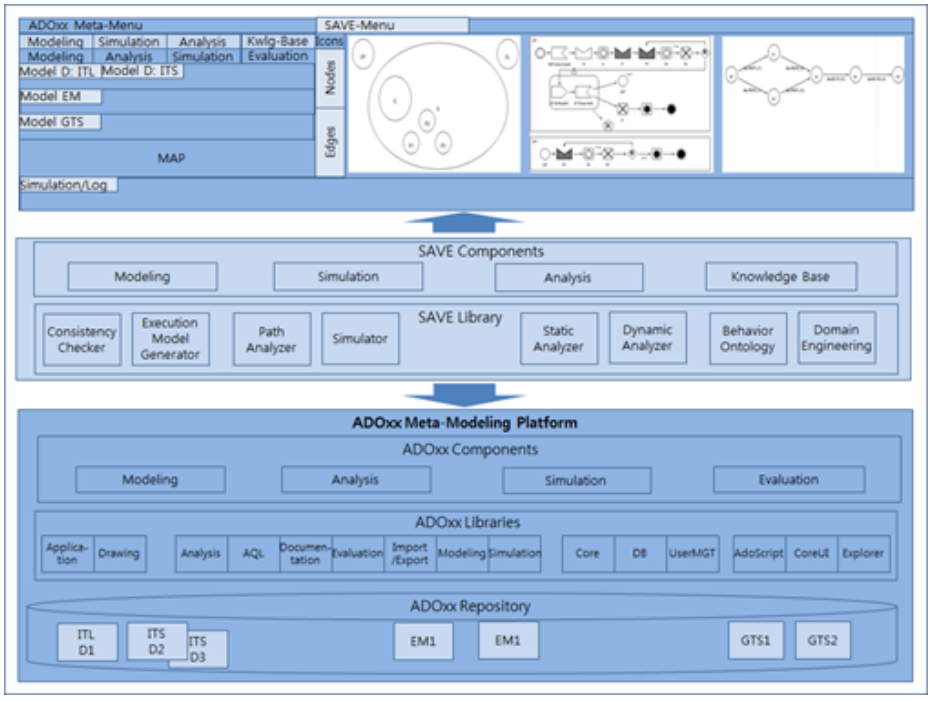


**Fig. 5.** SAVE Architecture

SAVE consists of the basic three components: Specifier, Analyzer and Verifier. Specifier, as shown in Fig. 6, is a tool to specify the IoT systems with dTP-Calculus, visually in the diagrammatic representations [11]. The left side of Fig. 6 is the In-the-Large (ITL) model, or system view, representing both inclusion relations among components of the system and communication channels among them. The right side of the figure is In-the-Small (ITS) models, or process view, representing a sequence of the detailed actions, interactions and movements performed by a process.

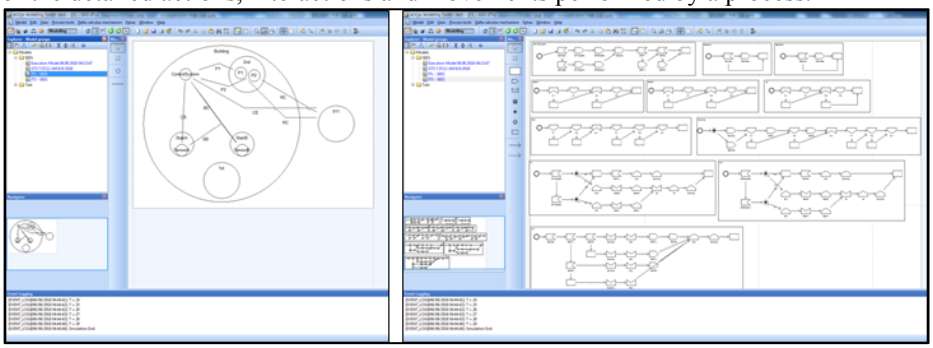

**Fig. 6.** Specification Tool of SAVE

Analyzer is a tool to generate the execution model from the specification in order to explore all the possible execution paths or cases, as the left side of Fig.7 shows in the form of a tree, and to perform trial-based simulation of each execution from the execution model in order to analyze probabilistic behaviors of the specified system.

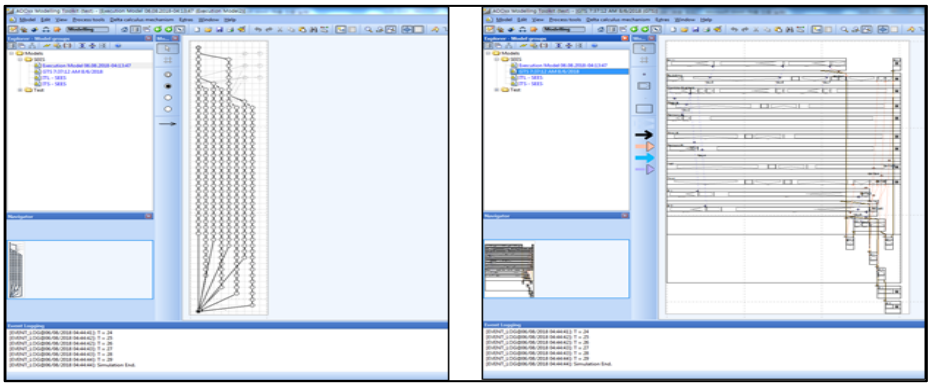

**Fig. 7.** Analysis and Verification Tool of SAVE

Verifier is a tool to verify a set of system requirements on the geo-temporal space generated, as output, from each simulation for all the execution paths or cases in the execution model, as the right side of Fig. 7 shows. This model allows both confirming the behavior and movements of the system and comprehending the security of the system by visualizing systems requirements and their verification results.

## 5 Comparative Study

The representative process algebras to specify probability properties of systems can be PAROMA [6] and PACSR [7]. These process algebras allow specifying various probability properties, but they have some limitations to the properties required by the IoT systems. PAROMA allows specifying exponential distribution probability model by using the λ parameter, at the time of defining location information of each agent. Further it is suitable to analyze systems consisting of geographically distributed agents by applying M2MAM (Multi-class, Multi-message Markovian Agent Models) [12]. However, the location information is simply a parameter used for communication, but mobility of the location cannot be expressed properly. PACSR is the process algebra to express resources and probability. It allows specifying three properties of resources, time and probability, as well as exceptional handling using time property, but only simple form of probability using discrete distribution is allowed.

However dTP-Calculus allows specifying various properties of geographical space, time and probabilities, suitable to the IoT environments. Geographical mobility, not just simple geographical information, can be expressed, and various types of time properties can be specified, too. More importantly, various probability properties can be specified with 4 kinds of probability models, and change of probability from change of the IoT environment can be more easily specified with probability density function, not with specific predefined probability. In addition, complex probability computation and simulation are automatically performed using the SAVE tool. The analysis of the IoT systems through the simulation increases prediction of nondeterministic behavior of the systems by showing whether the systems operate properly according to the specified probability or not.

## 6 Conclusion

This paper presented a probabilistic process algebra, known as dTP-Calculus, extended from dT-Calculus. It showed that the calculus allows 4 different types of probabilistic models in order to specify and analyze the IoT systems. It demonstrated that the calculus is capable of specifying and analyzing very complex probabilistic system behaviors, like IoT, based on the probabilistic models.

The paper, also, showed that a suite of tools, known as SAVE, has been developed on ADOxx in order to apply the calculus to real industrial examples in Industry 4.0. It also showed that SAVE can be used for real industrial examples based on different probabilistic cases in order to generate trial-based simulation output, not from the mathematical calculation. dTP-Calculus and SAVE can be considered as one of most innovative modeling methods and tools to specify and analyze very complex system behavior, like IoT, based on probability.

The future research will be development of requirements analysis and verification methods for probabilities, and be application of dTP-Calculus and SAVE to real the IoT examples 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), and Space Core Technology Development Program through the NRF(National Research Foundation of Korea) funded by the Ministry of Science, ICT and Future Planning(NRF-2014M1A3A3A02034792), and Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2015R1D1A3A01019282).

## References

1. Gregory Hale. Importance of IIoT safety and connectivity. https://www.controleng.com (2018).
2. Tabrizi, Farid Molazem, and Karthik Pattabiraman. Formal security analysis of smart embedded systems. Proceedings of the 32nd Annual Conference on Computer Security Applications. ACM (2016).
3. Aziz and Benjamin. A formal model and analysis of an IoT protocol. *Ad Hoc Networks 36.* Pp.49-57 (2016).
4. Diwan, Maithily, and Meenakshi D'Souza. A Framework for Modeling and Verifying IoT Communication Protocols. *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*. Springer, Cham, pp.266-280 (2017).
5. Ivan Lanese, Luca Bedogni, and Marco Di Felice. Internet of things: a process calculus approach. Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, pp.1339-1346 (2013).
6. Feng, Cheng, and Jane Hillston. PALOMA: a process algebra for located Markovian agents. International Conference on Quantitative Evaluation of Systems. Springer, pp.265-280 (2014).
7. Insup Lee, Anna Philippou and Oleg Sokolsky. Resources in process algebra. Departmental Papers (CIS) 337(2007).
8. Yeongbok Choe, Sunghyeon Lee and Moonkun Lee. dT-Calculus: A Process Algebra to Model Timed Movements of Processes. *International Journal of Computers*, volume 2, pp.53-62 ( 2017).
9. Y. Choe, W. Choi, G. Jeon and M. Lee. A Tool for Visual Specification and Verification for Secure Process Movements. eChallenges e-2015 Conference (2015).
10. H. Fill and D. Karagiannis. On the Conceptualisation of Modeling Methods Using the ADOxx Meta Modeling Platform. Proceedings of Enterprise Modeling and Information Systems Architectures 8 (1), pp.4-25 (2013).
11. Yeongbok Choe and Moonkun Lee. Algebraic Method to Model Secure IoT. *Domain-Specific Conceptual Modeling*. Springer, pp.335-355 (2016).
12. Cerotti, D., Gribaudo, M., Bobbio, A., Calafate, C.T., Manzoni, P. A Markovian agent model for fire propagation in outdoor environments. European Performance Engineering Workshop. Springer, Heidelberg, pp. 131–146 (2010).