

Process Algebra to Model Smart Distributed Mobile Real-Time IoT

Implementation Perspective for Cyber-Physical Systems

20 Sept 2019

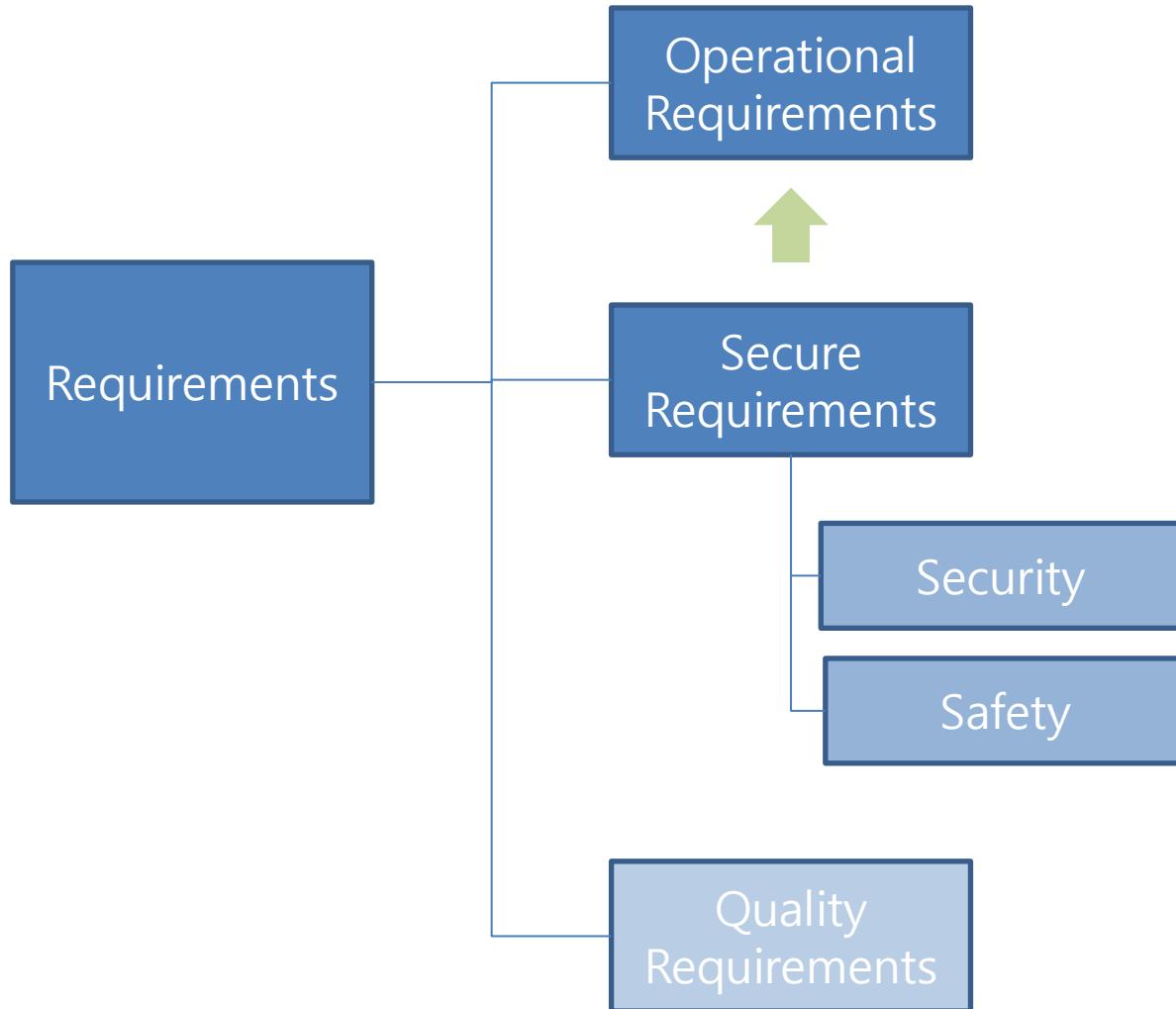
Prof. Moonkun Lee, Ph.D.

Division of Computer Science and Engineering
Chonbuk National University
Republic of Korea

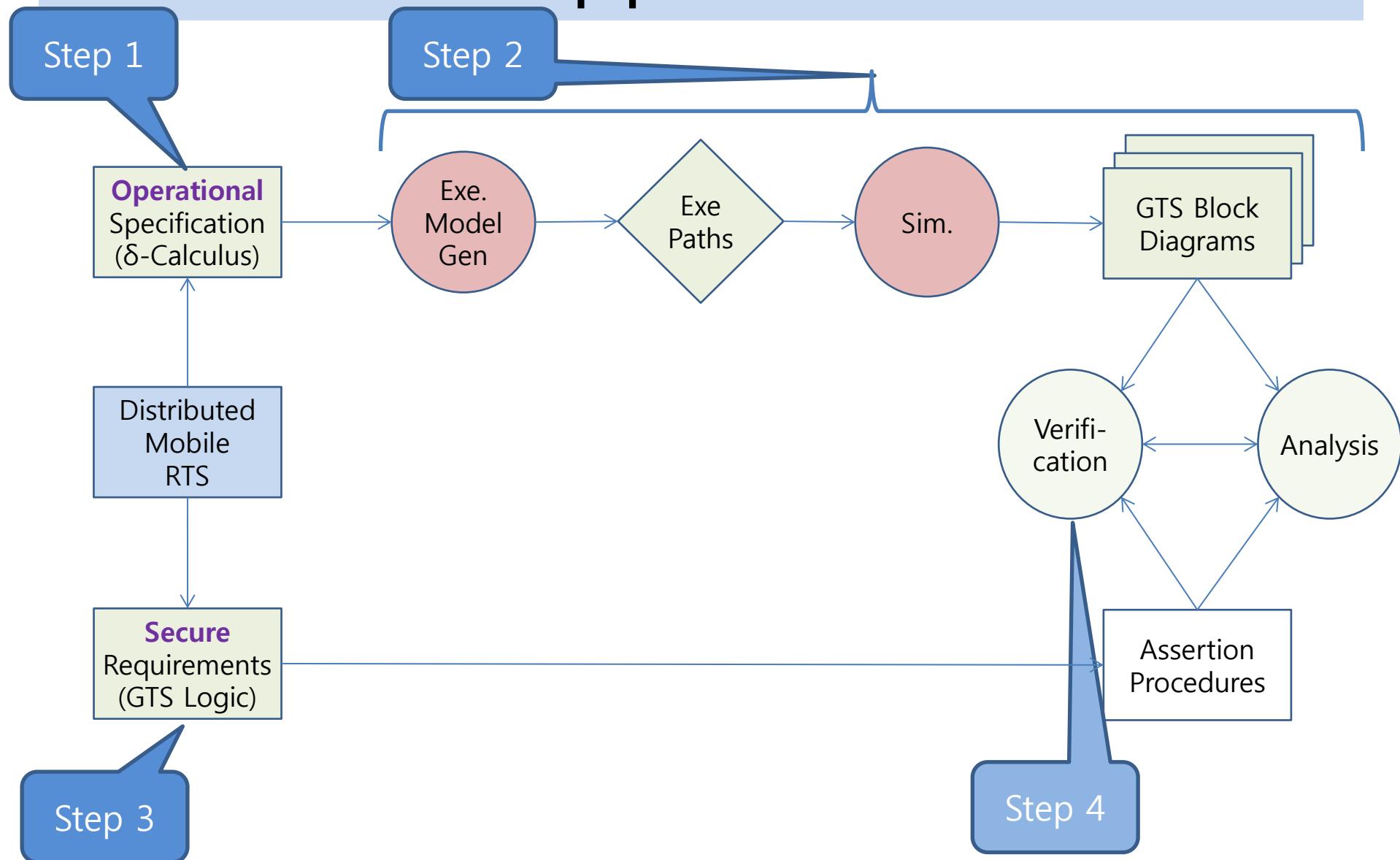
1. Motivation
2. **Modeling**
 - 1) Specification: δ -Calculus
 - 2) Verification: GTS Logic
 - 3) EMS Example
3. SAVE Tool
4. Cyber-Physical Systems Application
5. Summary
6. Discussion

2. **MODELING: SPECIFICATION & VERIFICATION**

Dual Approach



Approach

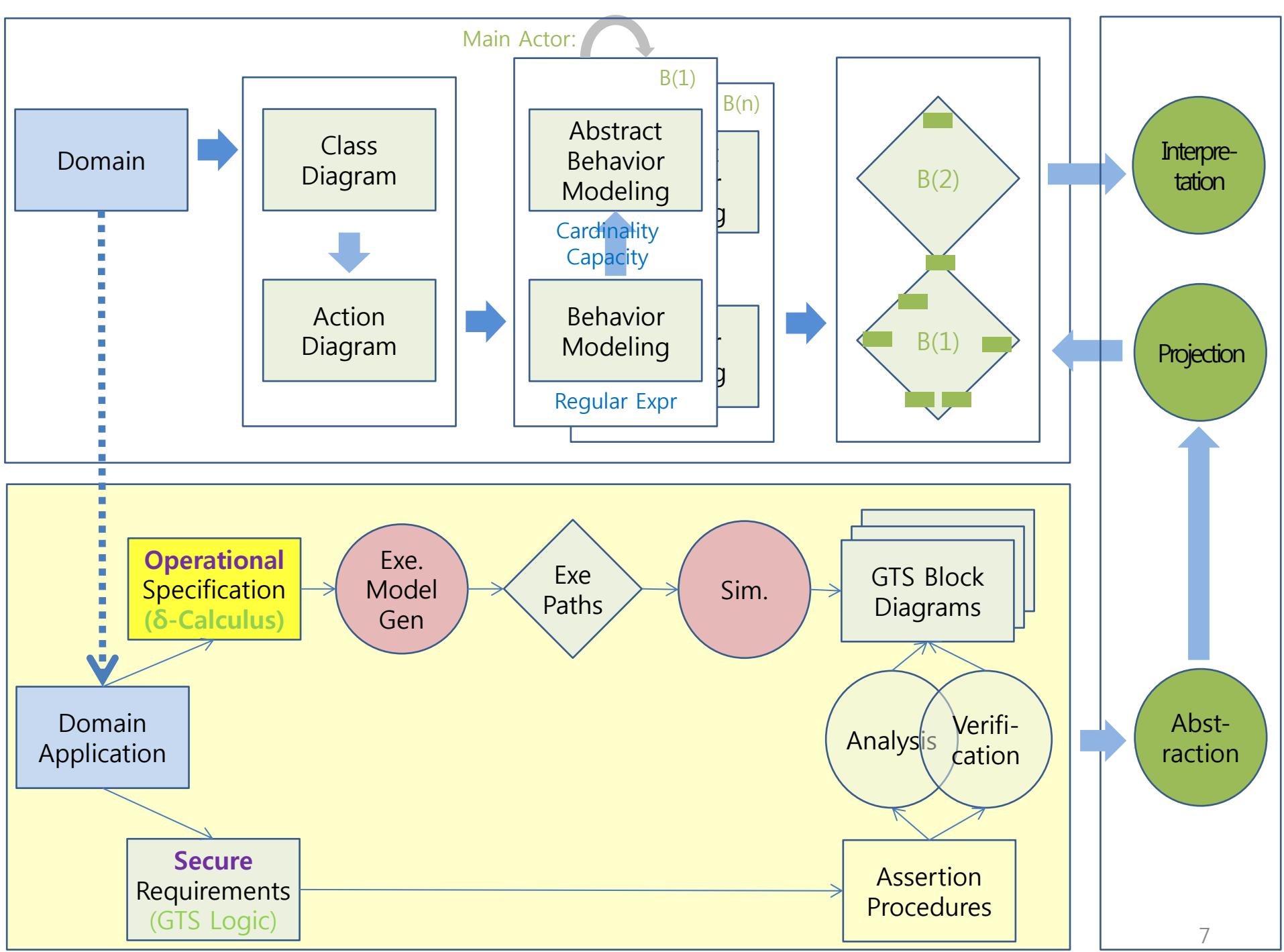


1. Motivation
2. Modeling
 - 1) Specification: δ -Calculus
 - 2) Verification: GTS Logic
 - 3) EMS Example
3. SAVE Tool
4. Cyber-Physical Systems Application
5. Summary
6. Discussion

2.1 δ -Calculus: Operational Specification

Step 1

Operational Requirements

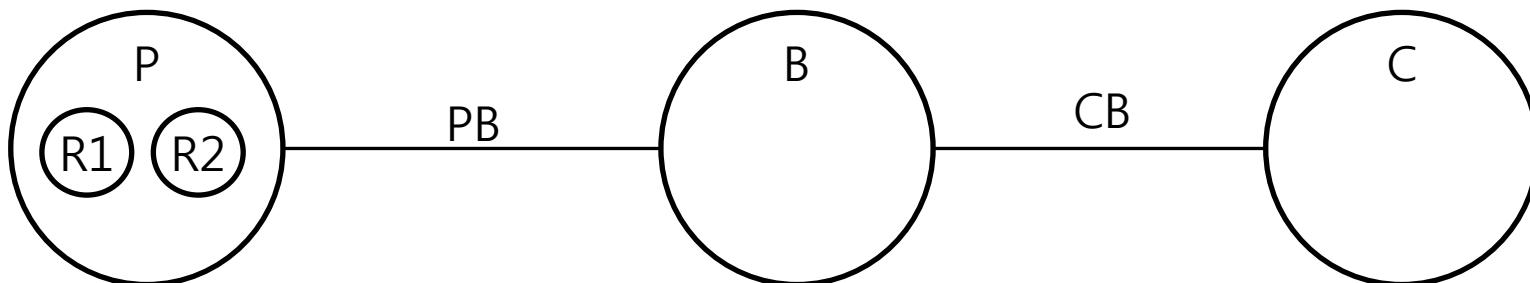


Requirements: PBC Example

- **Operational Requirements**
 - *Producer* produces two resources, R1 and R2.
 - *Producer* stores the resources in *Buffer* in order.
 - *Producer* informs *Buffer* of the order of R1 and R2, or R2 and R1.
 - *Consumer* consumes the resources from *Buffer* in order.
 - The order of the consumption is informed to *Buffer* by *Consumer*.

- Secure Requirements

- Security:
 - The order 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.
- Safety
 - The resources produced by *Producer* should be consumed by *Consumer* less than 5 minutes.

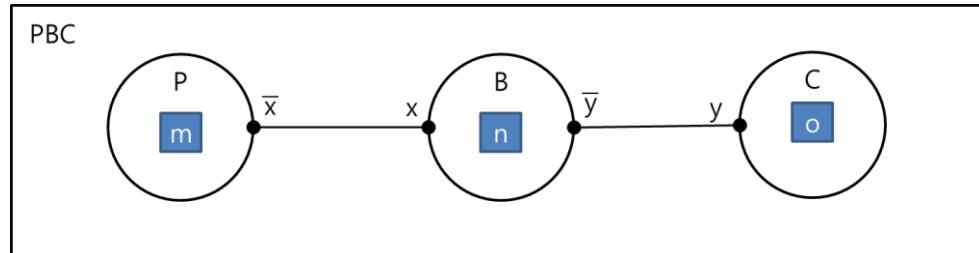


Specification: Process Algebra

$$\begin{aligned}
 1+2+3 &= 1+(2+3) \\
 &= 1+(3+2) \\
 &= 1+5 \\
 &= (1+5) \\
 &= 6
 \end{aligned}$$

- Specification:

- $PBC \stackrel{\text{def}}{=} P|C|B$
- $P \stackrel{\text{def}}{=} \bar{x}(m).P$
- $B \stackrel{\text{def}}{=} x(n).\bar{y}(n).B$
- $C \stackrel{\text{def}}{=} y(o).C$



- Execution:

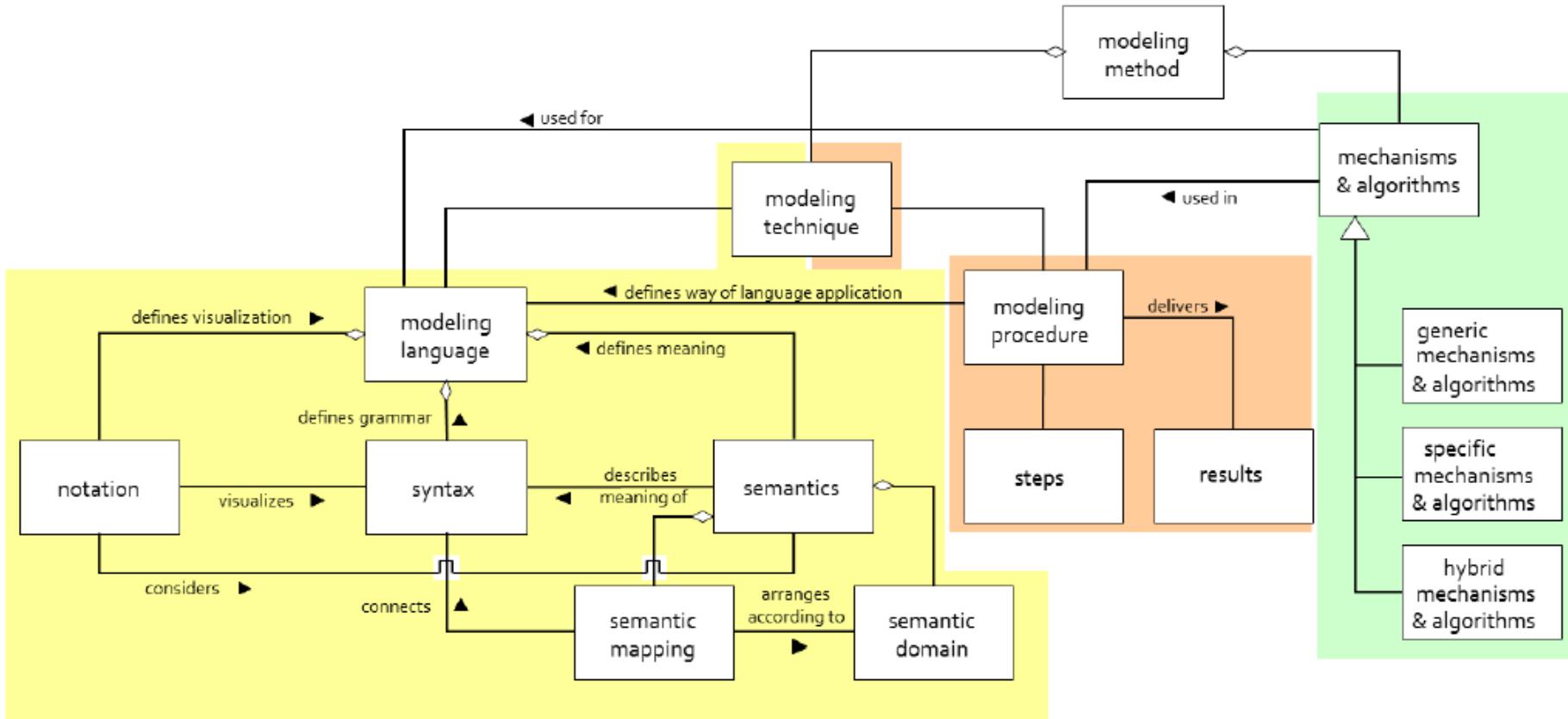
- $PBC \stackrel{\text{def}}{=} P|\textcolor{blue}{C}|B$ Commutative Law
- $= P|B|\textcolor{blue}{C}$
- $= (\bar{x}(m).P \mid \textcolor{blue}{x(m/n)}.\bar{y}(m/n).B) \mid y(o).C$
- $\rightarrow \quad \quad P \mid \quad \quad \quad (\bar{y}(m) \cdot B \mid \textcolor{blue}{y(m/o)}.C)$
- $\rightarrow \quad \quad P \mid \quad \quad \quad B \mid \quad \quad \quad C$

Associative Law

Binary; Synch

ADOxx Meta-Modeling Platform

$$\text{Method} = (T+MA) \cdot (L+P)$$



Karagiannis, D., Kühn, H.: „Metamodelling Platforms“. In Bauknecht, K., Min Tjoa, A., Quirmayer, G. (Eds.): Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2002, LNCS 2455, Springer, Berlin/Heidelberg, p. 182.

Process Algebra

System Representation

- System:
 - System: A set of processes.
 - Process: A sequence of actions or interactions.
 - Medium:
 - A set of ports or channels/buses for communication.
 - A set of gates or entrance for movements
- Operations
 - Process operations:
 - Parallel (|)
 - Choice (+)
 - Sequencing (;)
 - Actions:
 - Actions: Independent actions.
 - Communication
 - Movements
 - Usage of resources
 - Interactions: Dependent binary actions between two independent processes.
 - Communication
 - Movements
- Other properties;
 - Time
 - Probability: Choice (+)

Behavior Representation

- Labeled Transition System:
 - Transition rules
- Traces:
 - Process: A sequence of transitions.
 - System: A set of processes in transitions.
- Criteria:
 - Order of actions or interactions.
 - Properties related to the execution:
 - Time
 - probability
- Equivalence: Bisimulation
 - Strong bisimulation: Strong equivalence
 - Weak bisimulation: Observable equivalence

Language

- Syntax
 - Process
 - Action: Operation
 - Unary:
 - Asynchronous
 - Binary:
 - Synchronous
 - Asynchronous
- Semantics: Labelled transition systems
 - Operations: Transition rules
 - Unary:
 - Asynchronous
 - Binary:
 - Synchronous
 - Asynchronous
 - Laws
 - Operations
 - Precedency
 - Ordering
 - Equivalence

Example: CCS Syntax

$P ::= 0$	Empty process	A valid CCS process.
$ P_1 P_2$	Parallel Composition	The process P_1 and the process P_2 exist simultaneously.
$ P_1 + P_2$	Choice	Perform action a and continue as the process P_1 .
$ a.P_1$	Action Prefix	Proceed either as the process P_1 or the process P_2 .
$ P_1[b / a]$	Renaming	All actions named a to be renamed as b .
$ P_1 \setminus a$	Restriction	The process P_1 without action a .
$ A$	Process identifier	Use A to refer the process P_1 .

Example: CCS Semantics

$$\frac{\text{Premises}}{\text{Conclusion}} \quad [\text{Side Conditions}]$$

Nil	No transition	
Action Prefix	$\frac{}{a.P \xrightarrow{a} P}$	
Choice	$\frac{P \xrightarrow{a} P', Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} P'}$	Parallel Composition
	$\frac{P \xrightarrow{a} P', Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} Q'}$	$\frac{P \xrightarrow{a} P'}{P Q \xrightarrow{a} P' Q}$
Restriction	$\frac{P \xrightarrow{a} P'}{P \setminus L \xrightarrow{a} P' \setminus L} \left[a, \bar{a} \notin L \right]$	$\frac{P \xrightarrow{a} P', Q \xrightarrow{\bar{a}} Q'}{P Q \xrightarrow{\tau} P' Q'}$
Relabeling	$\frac{P \xrightarrow{a} P'}{P[f] \xrightarrow{f(a)} P'[f]}$	Process Variable
		$\frac{P \xrightarrow{a} P'}{c \xrightarrow{a} P'} \left[c \triangleq P \right]$

Example: CCS Basic Laws

Laws	
Choice: Commutative	$x + y = y + x$
Choice: Associative	$(x + y) + z = y + (x + z)$
Choice: Idempotent	$x + x = x$
Distribution of + over .	$(x + y).z = x.z + y.z$
Sequence: Associative	$(x.y).z = y.(x.z)$
Parallel: Commutative	$x \parallel y = y \parallel x$
Parallel: Associative	$(x \parallel y) \parallel z = y \parallel (x \parallel z)$

Producer-Buffer-Consumer Example

- Specification:

- $PBC \stackrel{\text{def}}{=} P|C|B$
- $P \stackrel{\text{def}}{=} \bar{x}(m).P$
- $B \stackrel{\text{def}}{=} x(n).\bar{y}(n).B$
- $C \stackrel{\text{def}}{=} y(o).C$

$$\begin{aligned} 1+2+3 &= 1+(2+3) \\ &= 1+(3+2) \\ &= 1+5 \\ &= (1+5) \\ &= 6 \end{aligned}$$

- Execution:

- $PBC \stackrel{\text{def}}{=} P|\textcolor{blue}{C}|B$ Commutative Law
- $= P|B|C$
- $= (\bar{x}(m).P \mid \textcolor{blue}{x(m/n).\bar{y}(m/n).B} \mid y(o).C)$
- $\rightarrow \quad \quad P \mid \quad \quad \quad (\bar{y}(m) \cdot B \mid \textcolor{blue}{y(m/o).C})$
- $\rightarrow \quad \quad P \mid \quad \quad \quad B \mid \quad \quad \quad C$

Associative Law

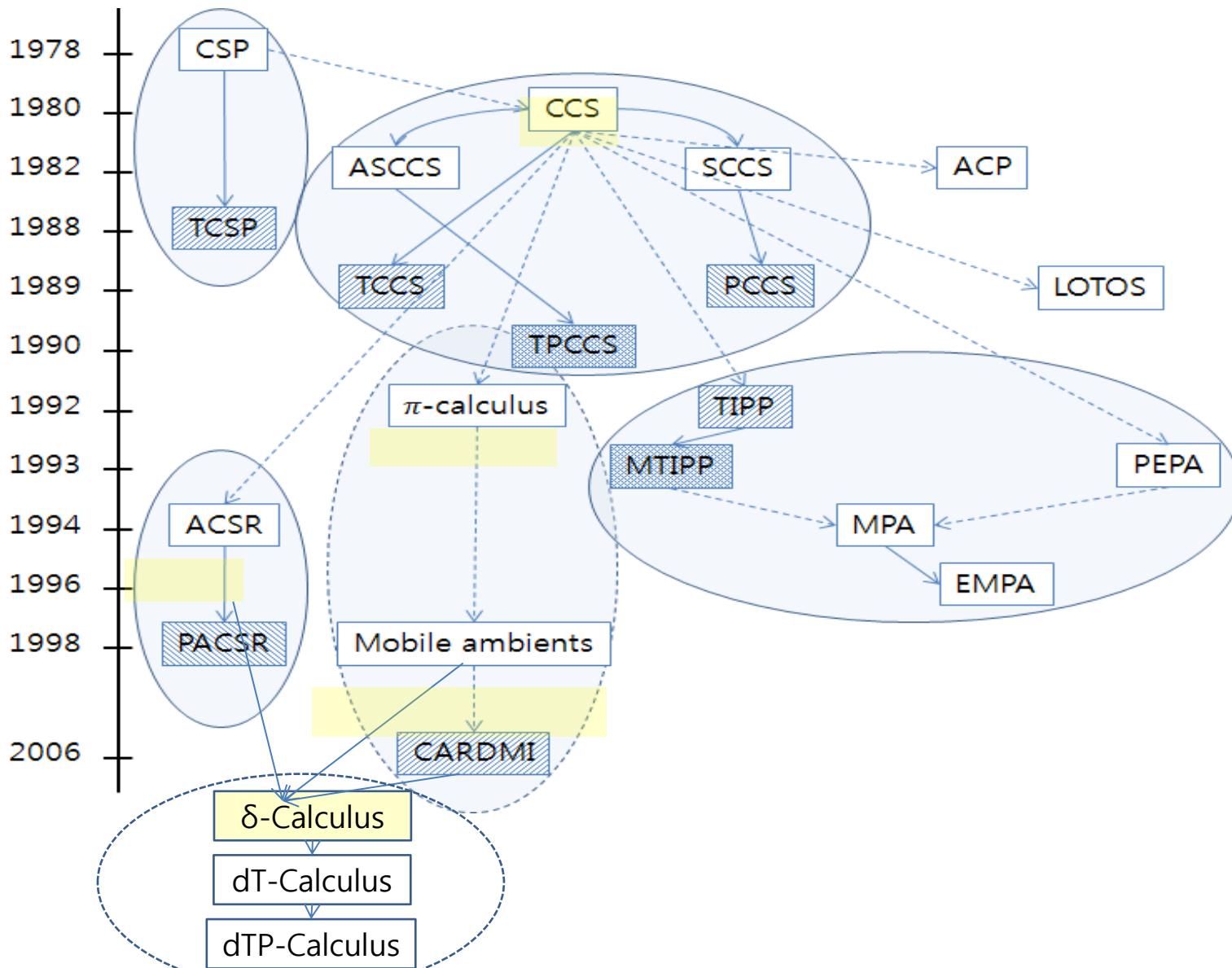
Binary; Synch

δ -Calculus

Overview

- A process calculus:
 - Based on ACSR, extended from CCS.
 - Extending its feature for movements on space.
- Purpose
 - Mathematical formalisms for describing and analyzing properties of concurrent computation with resource and priority, including movements over geo-temporal space.
- Main characteristics:
 - Space
 - Movement
 - Time
- Applications:
 - Distributed Mobile (Real-time) systems

Map



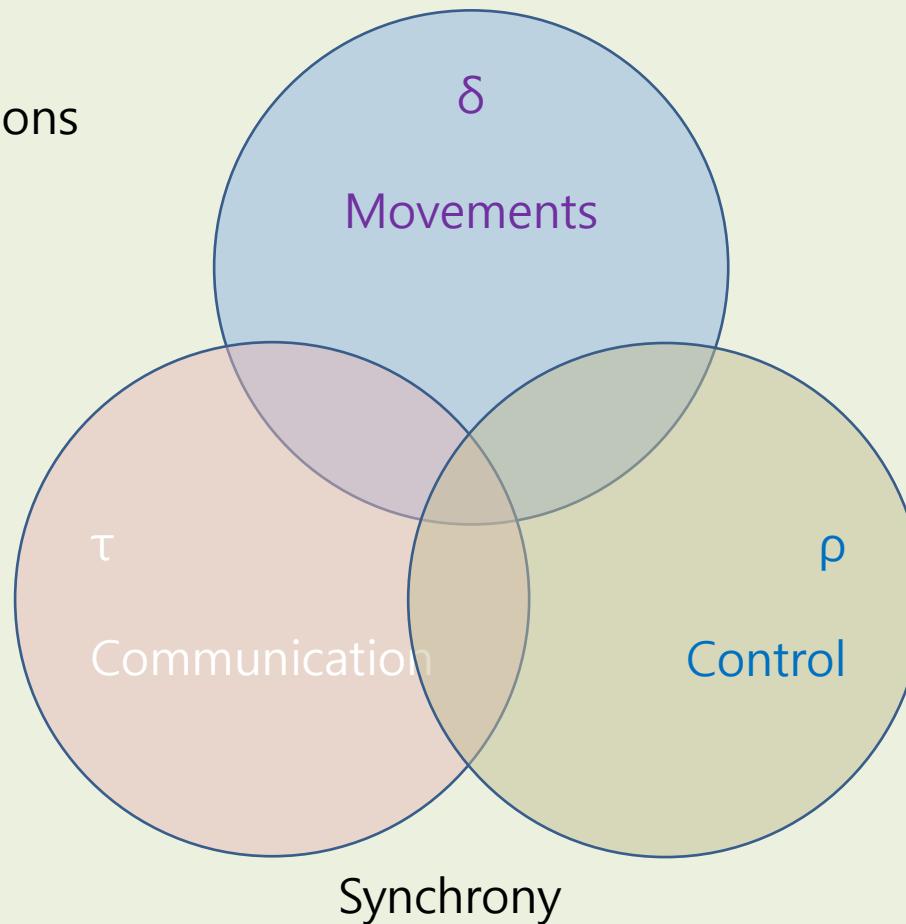
Main Characteristics

Process Algebra	Features
CCS	<ul style="list-style-type: none">ConcurrencyCommunication (τ)
π -Calculus	+ Channel Passing
Mobile Ambient	+ Asynchronous Movement
ACSR	+ Resource + Time + Priority
δ -Calculus	++ Space ++ Synchronous Movement (δ) ++ Synchronous Control (p)

δ -Calculus: Overview

Geographical Space

- Processes
 - Actions
 - Interactions
- Time
- Priority
- Scope



System & Process

- System $\mathcal{S} = \langle \mathcal{P}, \mathcal{I}, \mathcal{C} \rangle$, where
 - $\mathcal{P} = \{P_{1,r_1,\textcolor{blue}{T}_1}, \dots, P_{n_p,r_p,\textcolor{blue}{T}_{n_p}}\}$: A set of processes
 - $T_i = [t_l, t_u]; t_l \leq t_u$; Optional
 - $P \in \mathcal{P}; P = a_{1;\textcolor{blue}{T}_1} \circ \dots \circ a_{m_p;\textcolor{blue}{T}_{mp}}$: A set of actions
 - $T_i = [t_l, t_u]; t_l \leq t_u$; Optional
 - r_i : Priority
 - $\mathcal{I} = \{I_1, \dots, I_{n_i}\}$: A set of process inclusions
 - $I \in \mathcal{I}; c = (P, Q) = P[Q]; P, Q \in \mathcal{P}$
 - $\mathcal{C} = \{c_1, \dots, c_{n_c}\}$: A set of channels
 - $c \in \mathcal{C}; c = (P, Q); P, Q \in \mathcal{P}$

Actions

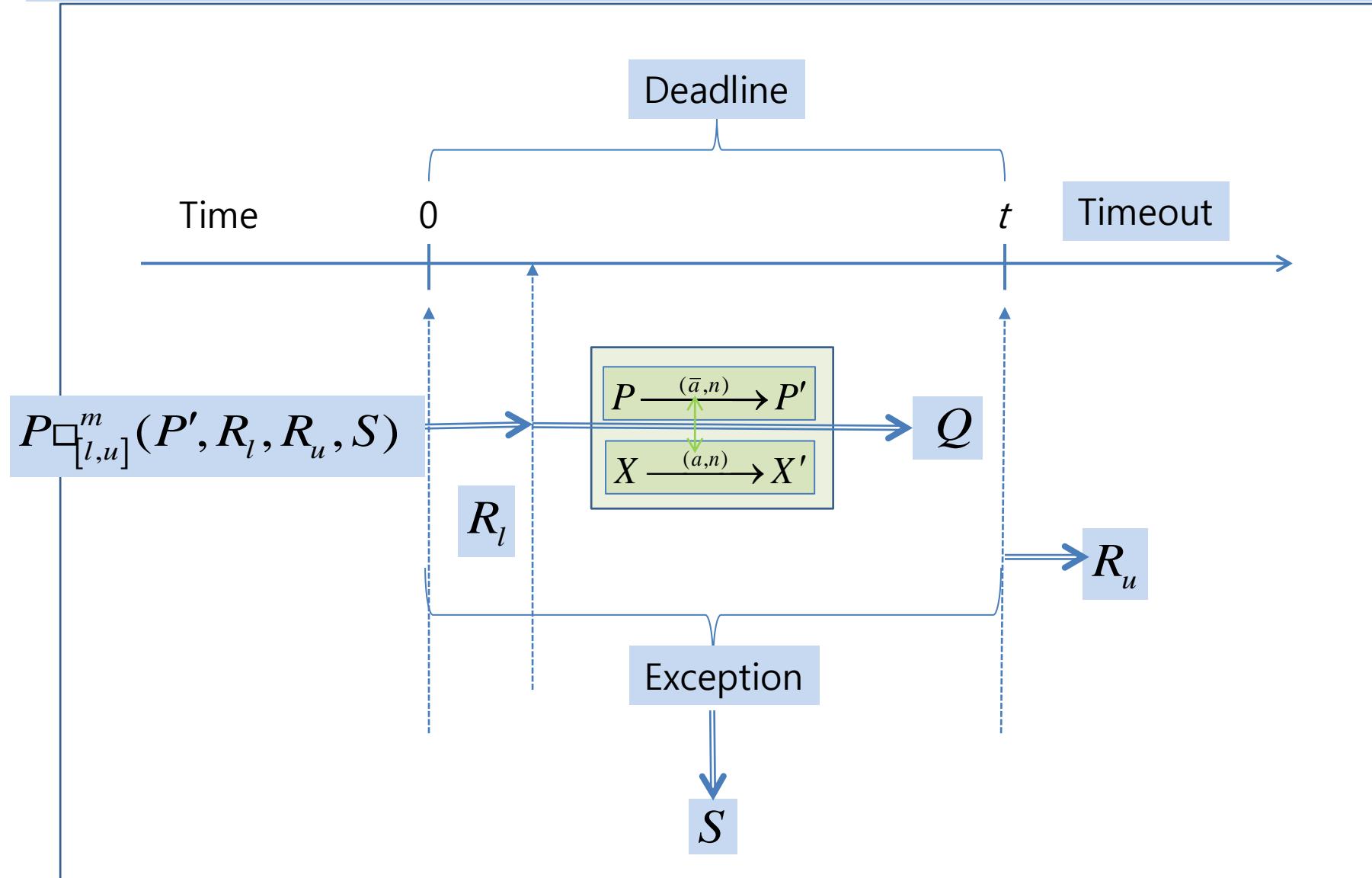
- $P \in \mathcal{P}; P = a_{1;r_1;T_1} \circ \dots \circ a_{m;r_m;T_m}$: A set of actions
 - $T_i = [t_l, t_u]; t_l \leq t_u$; Optional
- Type:
 - τ : Communication
 - Send:
 - Receive:
 - δ : Movement
 - Active
 - In:
 - Out:
 - Passive
 - Get
 - put
 - ρ : Control
 - Start
 - Terminate
 - Exit
 - Suspend
 - wake

Interaction		Action	Co-action
τ		$P: c(m);$	$Q: c(\bar{m});$
δ	Active	In	$P: (in Q);$
		out	$P: (out Q);$
	Passive	in	$P: (get Q);$
		out	$P: (put Q);$
ρ	Start	$P: (start Q);$	$Q.start;$
	Terminate	$P: (terminate Q);$	$Q.terminate;$
	Exit	$P: (exit);$	$P.terminate;$
	Suspend	$P: (suspend Q);$	$Q.suspend;$
	Wake	$P: (wake Q);$	$Q.wake;$

Autonomous

Heteronomous

Movement Scope



Syntax

$$P ::= nil \mid A \mid P_{(n)} \mid P[Q] \mid P\langle r_t \rangle \mid P + Q \mid P|Q \mid A \bullet P$$
$$A ::= \emptyset \mid r_t(\overline{msg}) \mid r_t(msg) \mid M$$
$$M ::= m_t^p(k) \mid P \mid P \ m(k)_t^p$$
$$m ::= in \mid out \mid get \mid put$$

Communication (τ) Semantics

Action	$\frac{-}{r(a) \bullet P \xrightarrow{r(a)} P}$	Choic eL	$\frac{P \xrightarrow{A} P'}{P + Q \xrightarrow{A} P'}$
ChoiceR	$\frac{Q \xrightarrow{A} Q'}{P + Q \xrightarrow{A} Q'}$	ParlL	$\frac{P \xrightarrow{A} P'}{P Q \xrightarrow{A} P' Q'}$
ParlR	$\frac{Q \xrightarrow{A} Q'}{P Q \xrightarrow{A} P Q'}$	ParCo m	$\frac{P \xrightarrow{A} P', Q \xrightarrow{\bar{A}} Q'}{P Q \xrightarrow{\tau} P' Q'}$
NestO	$\frac{P \xrightarrow{A} P'}{P[Q] \xrightarrow{A} P'[Q]}$	NestI	$\frac{Q \xrightarrow{A} Q'}{P[Q] \xrightarrow{A} P[Q']}$
Nest-Com	$\frac{P \xrightarrow{A} P', Q \xrightarrow{\bar{A}} Q'}{P[Q] \xrightarrow{\tau} P'[Q']}$		

Movement (δ) Semantics

In	$\frac{P \xrightarrow{\text{in } Q} P', Q \xrightarrow{P \text{ in}} Q'}{P Q \xrightarrow{\delta} Q'[P']}$	Out	$\frac{P \xrightarrow{\text{out } Q} P', Q \xrightarrow{Q \text{ out}} Q'}{Q[P] \xrightarrow{\delta} P' Q'}$
Get	$\frac{P \xrightarrow{\text{get } Q} P', Q \xrightarrow{P \text{ get}} Q'}{P Q \xrightarrow{\delta} P'[Q']}$	Put	$\frac{P \xrightarrow{\text{put } Q} P', Q \xrightarrow{Q \text{ put}} Q'}{P[Q] \xrightarrow{\delta} P' Q'}$
InP	$\frac{P_{(n)} \xrightarrow{\text{in}^p Q} P_{(n)}'}{P_{(n)} Q_{(m)} \xrightarrow{\text{in}^p Q} Q_{(m)}[P_{(n)}']}$ ($n > m$)		
OutP	$\frac{P_{(n)} \xrightarrow{\text{out}^p Q} P_{(n)}'}{Q_{(m)}[P_{(n)}] \xrightarrow{\text{out}^p Q} P_{(n)}' Q_{(m)}}$ ($n > m$)		
GetP	$\frac{P_{(n)} \xrightarrow{\text{get}^p Q} P_{(n)}'}{P_{(n)} Q_{(m)} \xrightarrow{\text{get}^p Q} P_{(n)}'[Q_{(m)}]}$ ($n > m$)		
PutP	$\frac{P_{(n)} \xrightarrow{\text{put}^p Q} P_{(n)}'}{P_{(n)}[Q_{(m)}] \xrightarrow{\text{out}^p Q} P_{(n)}' Q_{(m)}}$ ($n > m$)		
InN	$\frac{P \xrightarrow{\text{in } Q} P', Q \xrightarrow{P \text{ in}} Q'}{P Q[R] \xrightarrow{\delta} Q'[P' R]}$	GetN	$\frac{P \xrightarrow{\text{get } Q} P', Q \xrightarrow{P \text{ get}} Q'}{P[R] Q \xrightarrow{\delta} P'[R Q']}$

Control (ρ) Semantics

Start	$\frac{P \xrightarrow{\text{start } Q} P', (Q) \xrightarrow{Q.\text{start}} Q}{P \xrightarrow{\gamma} P[Q]}$
Terminate	$\frac{P[Q], P \xrightarrow{\text{terminate } Q} P', Q \xrightarrow{Q.\text{terminate}} ()}{P[Q] \xrightarrow{\gamma} P'}$
Exit	$\frac{P \xrightarrow{\text{exit}} P', P \xrightarrow{P.\text{exit}} ()}{P \xrightarrow{\gamma} ()}$
Suspend	$\frac{P[Q], P \xrightarrow{\text{suspense } Q} P', Q \xrightarrow{Q.\text{suspend}} Q_S}{P[Q] \xrightarrow{\gamma} P'[Q']}$
Wake	$\frac{P \xrightarrow{\text{wake } Q} P', Q_S \xrightarrow{Q.\text{wake}} Q'}{P[Q_S] \xrightarrow{\gamma} P'[Q]}$

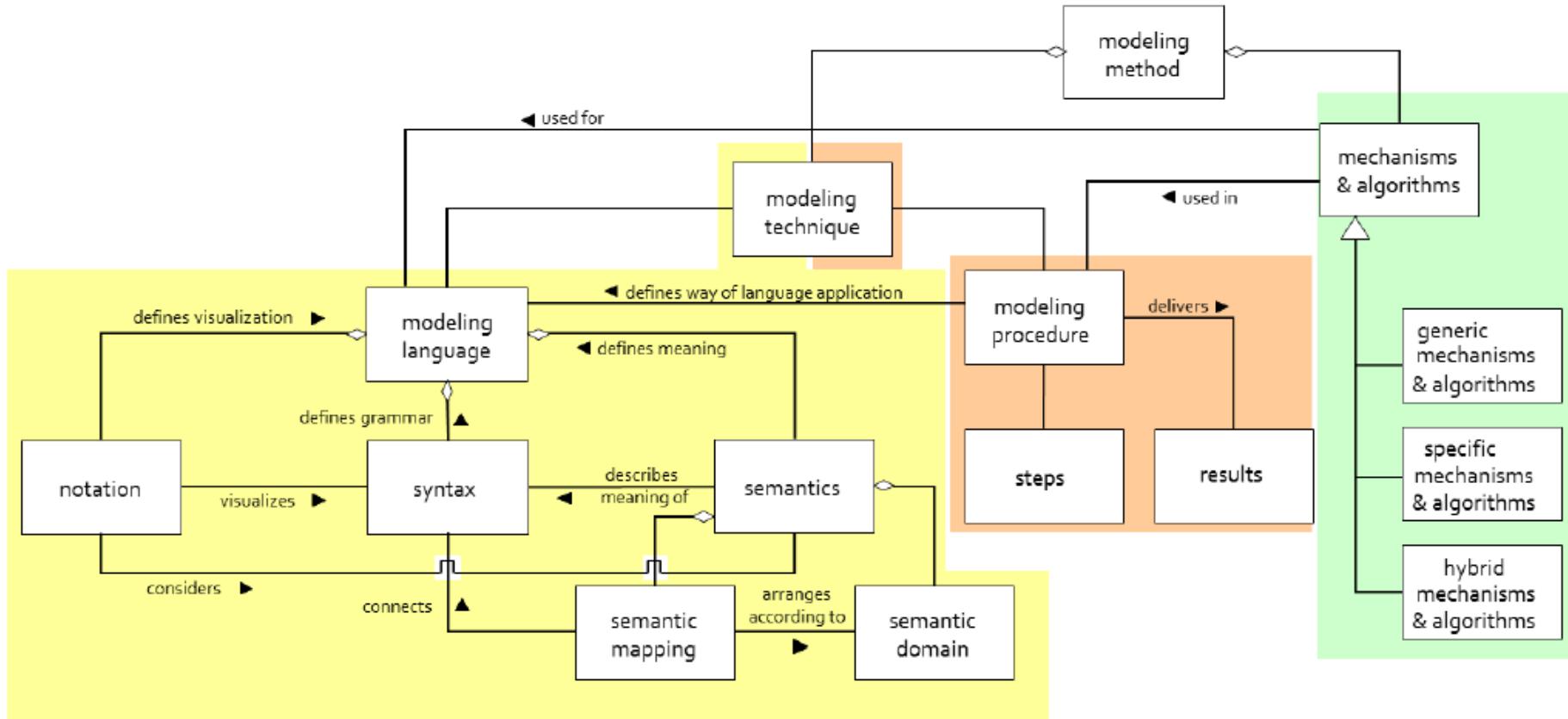
Laws

Choice(1)	$P + P \equiv P$	Choice(2)	$P + Q \equiv Q + P$
Choice(3)	$\begin{aligned} & (P + Q) + R \\ & \equiv P + (Q + R) \end{aligned}$	Parallel(1)	$P \mid nil \equiv P$
Parallel(2)	$P \mid Q \equiv Q \mid P$	Parallel(3)	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
Nesting(1)	$P[nil] \equiv P$	Nesting(2)	$R[P] + R[Q] \equiv R[P + Q]$
Distributive (1)	$\begin{aligned} & P \mid (Q + R) \\ & \equiv (P \mid Q) + (P \mid R) \end{aligned}$	Distributive (2)	$\begin{aligned} & (A_1 + A_2).P \\ & \equiv A_1.P + A_2.P \end{aligned}$

VISUALIZATION

ADOxx Meta-Modeling Platform

$$\text{Method} = (T+MA) \cdot (L+P)$$

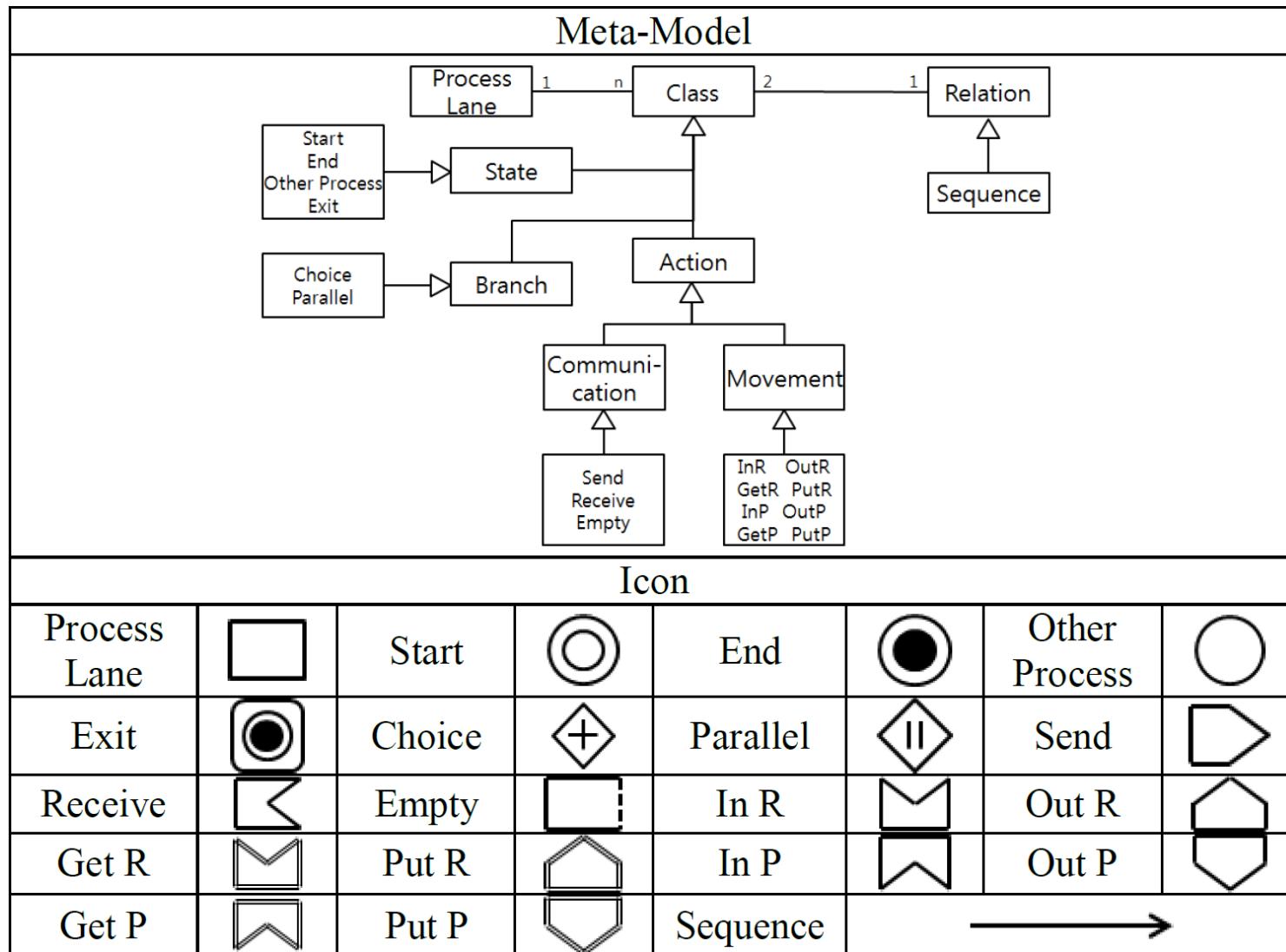


Karagiannis, D., Kühn, H.: „Metamodelling Platforms“. In Bauknecht, K., Min Tjoa, A., Quirmayer, G. (Eds.): Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2002, LNCS 2455, Springer, Berlin/Heidelberg, p. 182.

ITL Icons

Meta-Model	Icon	
<p>A UML Class Diagram fragment. At the top left is a box labeled "Class". To its right is a box labeled "Relation". A horizontal line connects them with a multiplicity of "2" on the "Class" side and "1" on the "Relation" side. Below "Class" is a box labeled "Process". An upward-pointing arrow connects "Process" to "Class". To the right of "Process" are two boxes: "Channel" and "Movement". A horizontal line connects "Channel" and "Movement". Above "Channel" and "Movement" is another box labeled "Relation". An upward-pointing arrow connects "Channel" and "Movement" to "Relation".</p>	Process	
	Channel	
	Movement	

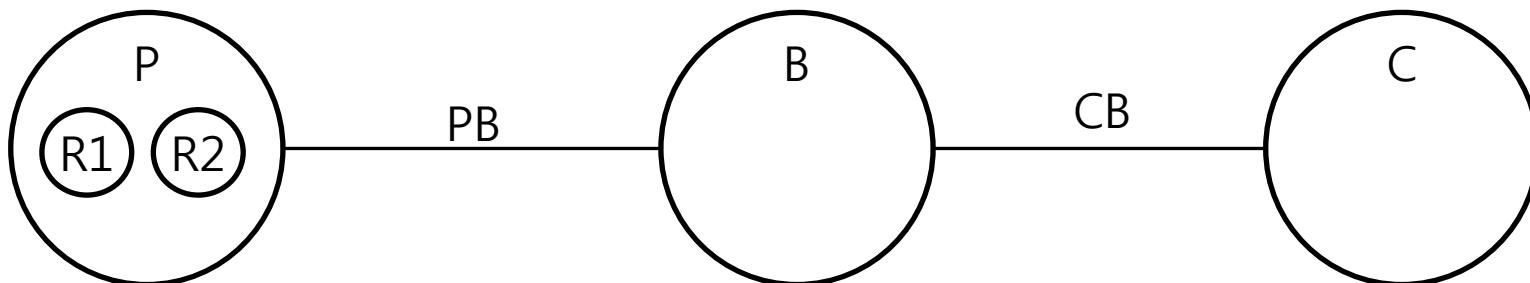
ITS Icons



EXAMPLE: PBC

Example: PBC

- Operational Requirements
 - *Producer* produces two resources, R1 and R2.
 - *Producer* stores the resources in *Buffer* in order.
 - *Producer* informs *Buffer* of the order of R1 and R2, or R2 and R1.
 - *Consumer* consumes the resources from *Buffer* in order.
 - The order of the consumption is formed to *Buffer* by *Consumer*.
- Secure Requirements
 - Security:
 - The order 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.
 - Safety
 - The resources produced by *Producer* should be consumed by *Consumer* less than 5 minutes.



PBC Example (0)

- δ -Calculus

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

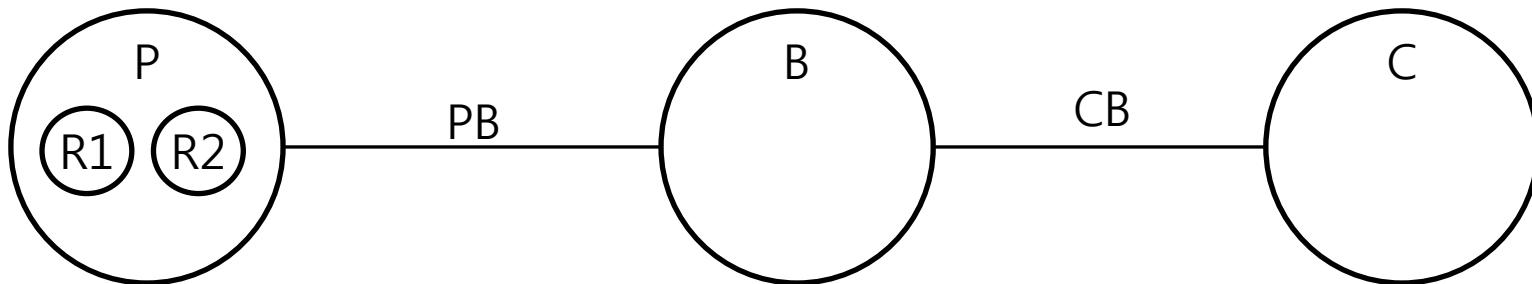
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



CASE 1

PBC Example (1) : τ_1

- δ -Calculus

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

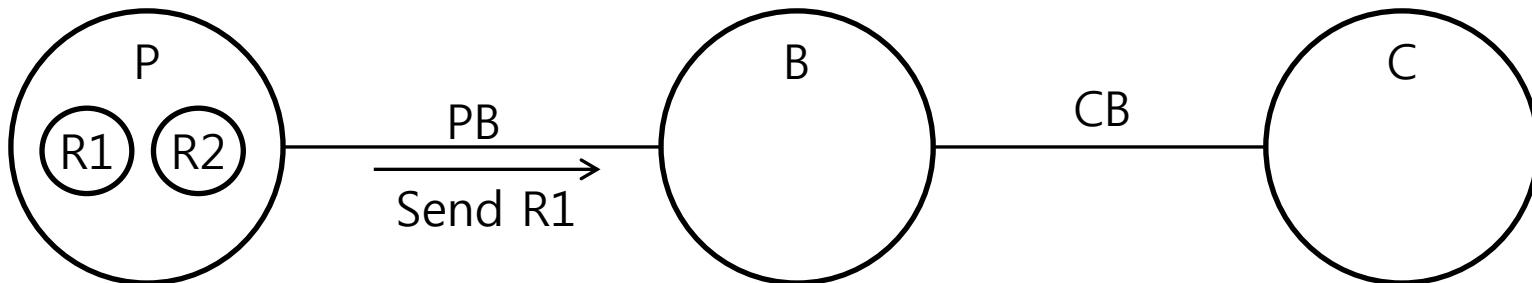
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



PBC Example (2) : $\delta_{1,1}$

- δ -Calculus

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

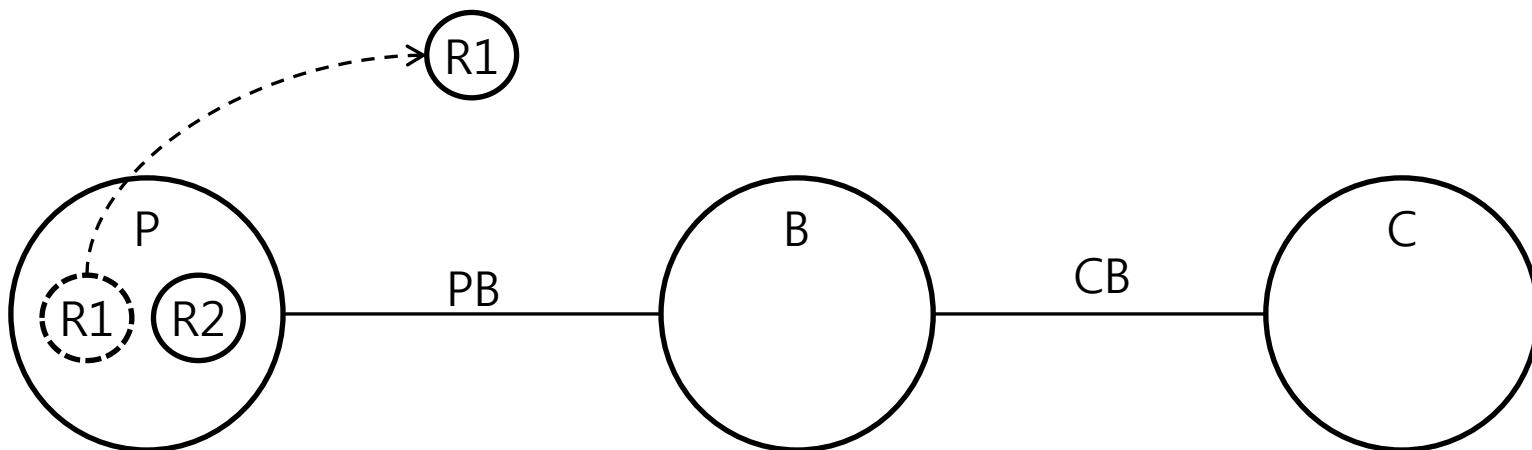
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



PBC Example (3) : $\delta_{1,2}$

- δ -Calculus

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

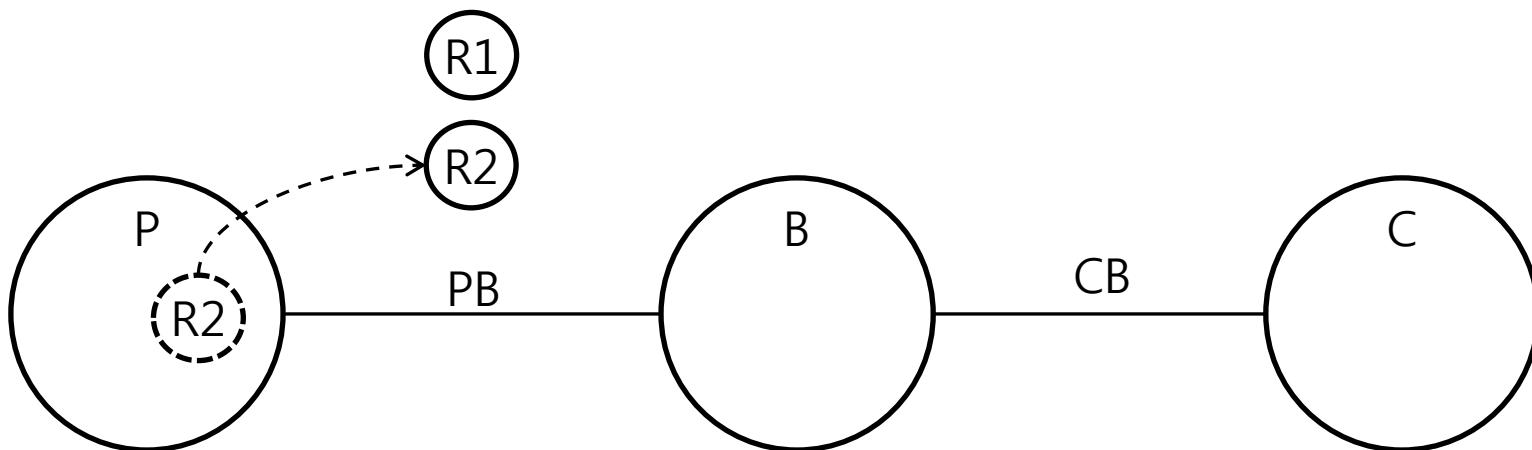
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



PBC Example (4) : $\delta_{2,1}$

- δ -Calculus

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

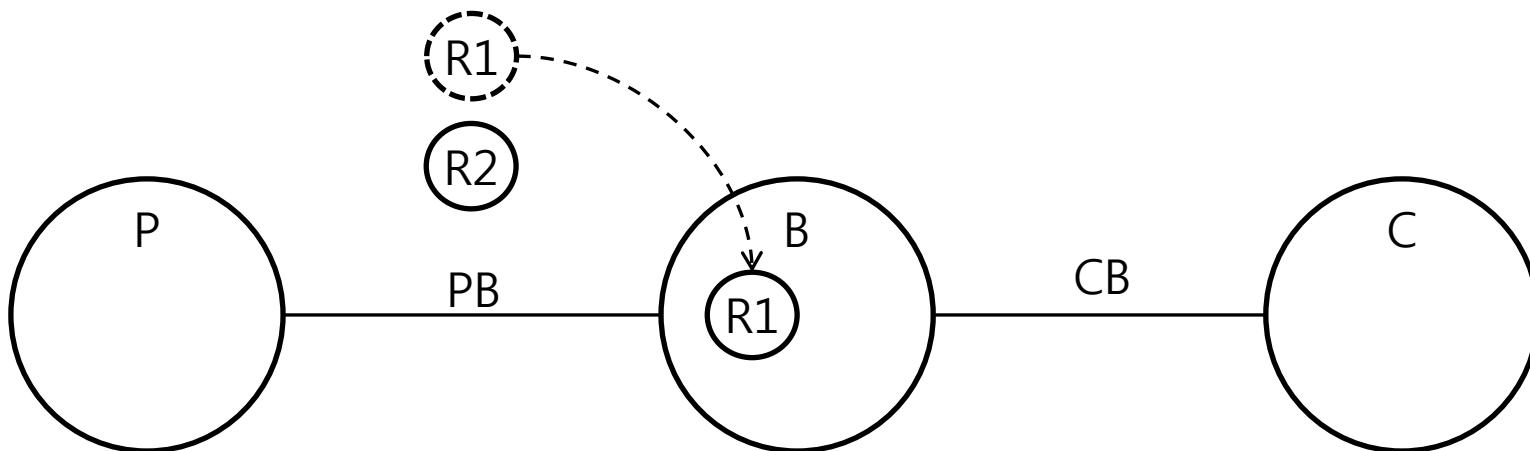
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

$$B = (PB(Send\ R1).\text{get}\ R1.get\ R2 + PB(Send\ R2).\text{get}\ R2.get\ R1).put\ R1.put\ R2.exit$$

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



PBC Example (5) : $\delta_{2,2}$

- δ -Calculus

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

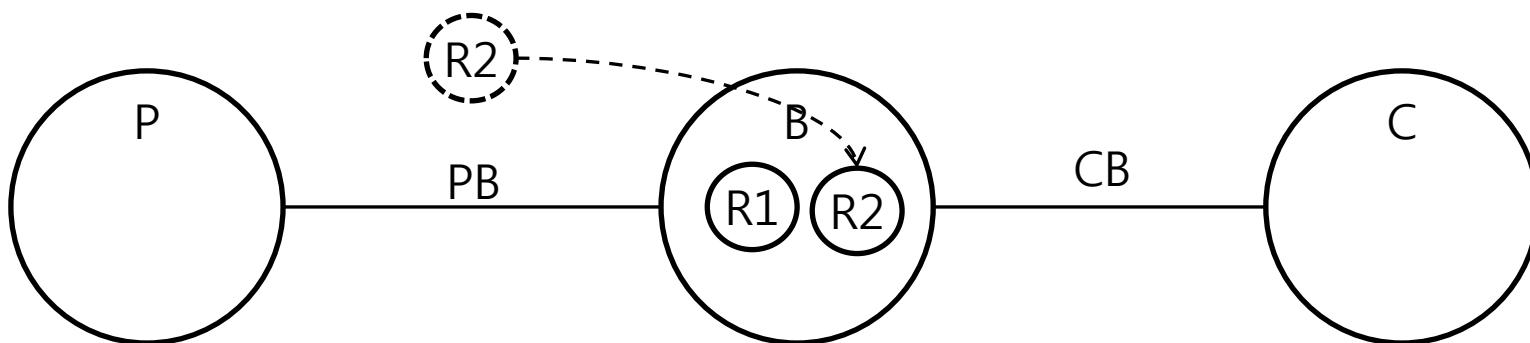
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



PBC Example (6) : $\delta_{3,1}$

- δ -Calculus

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

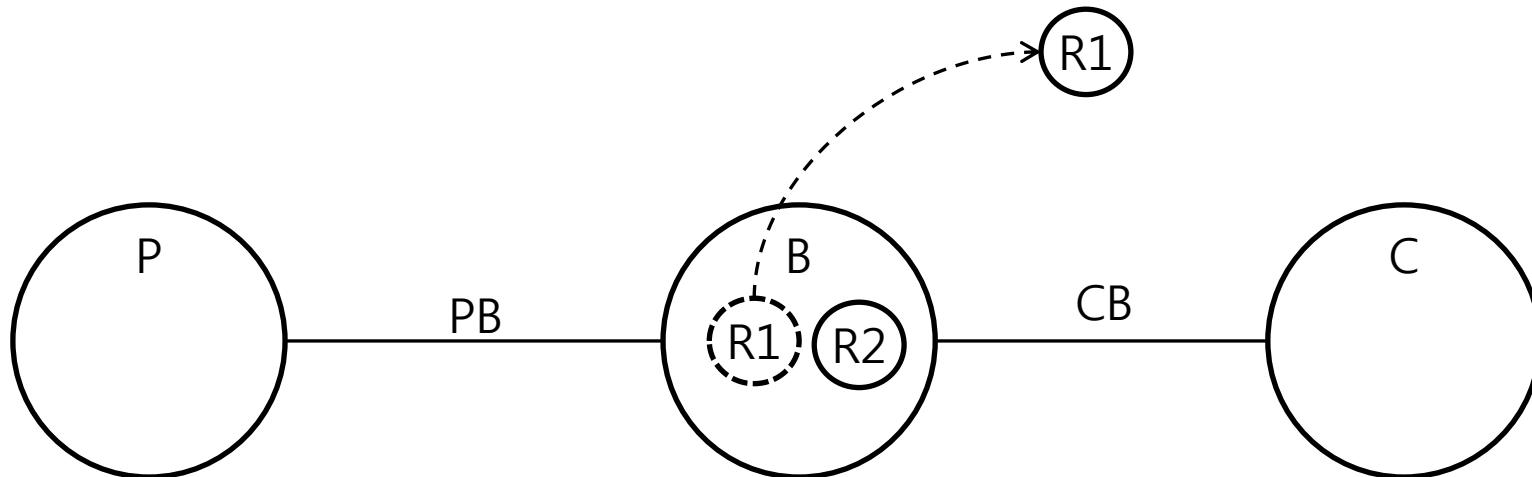
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



PBC Example (7) : $\delta_{3,2}$

- δ -Calculus

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

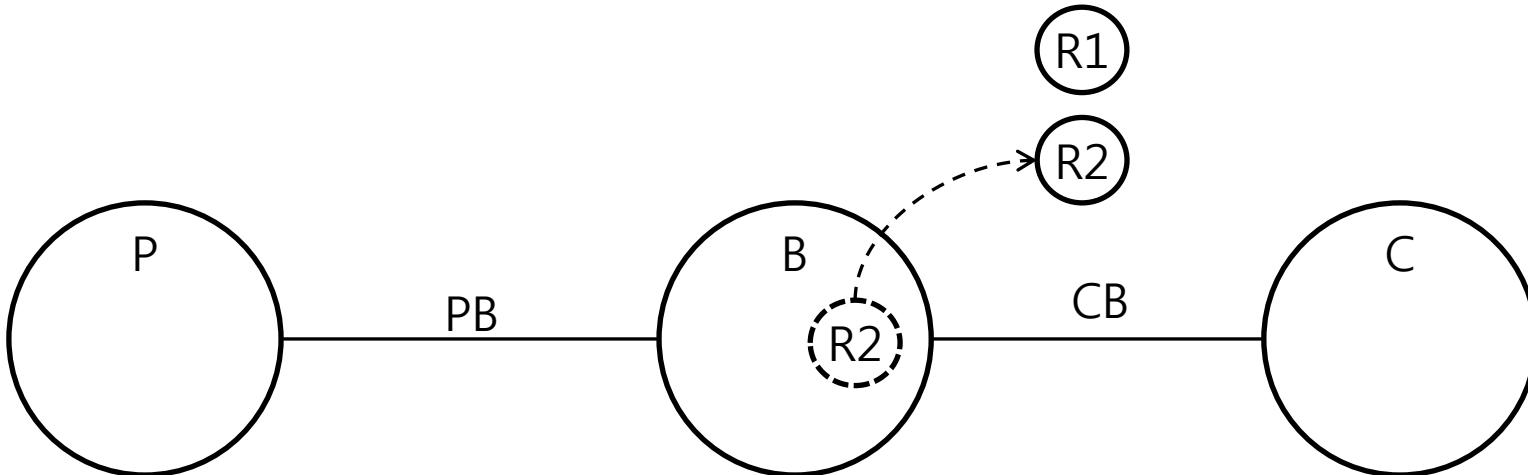
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

$$B = (PB(Send\ R1).get\ R1.get\ R2 + PB(Send\ R2).get\ R2.get\ R1).put\ R1.\textcolor{blue}{put\ R2}.exit$$

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.\textcolor{blue}{B\ put}.C\ get.exit$$



PBC Example (8) : $\delta_{4,1}$

- δ -Calculus

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

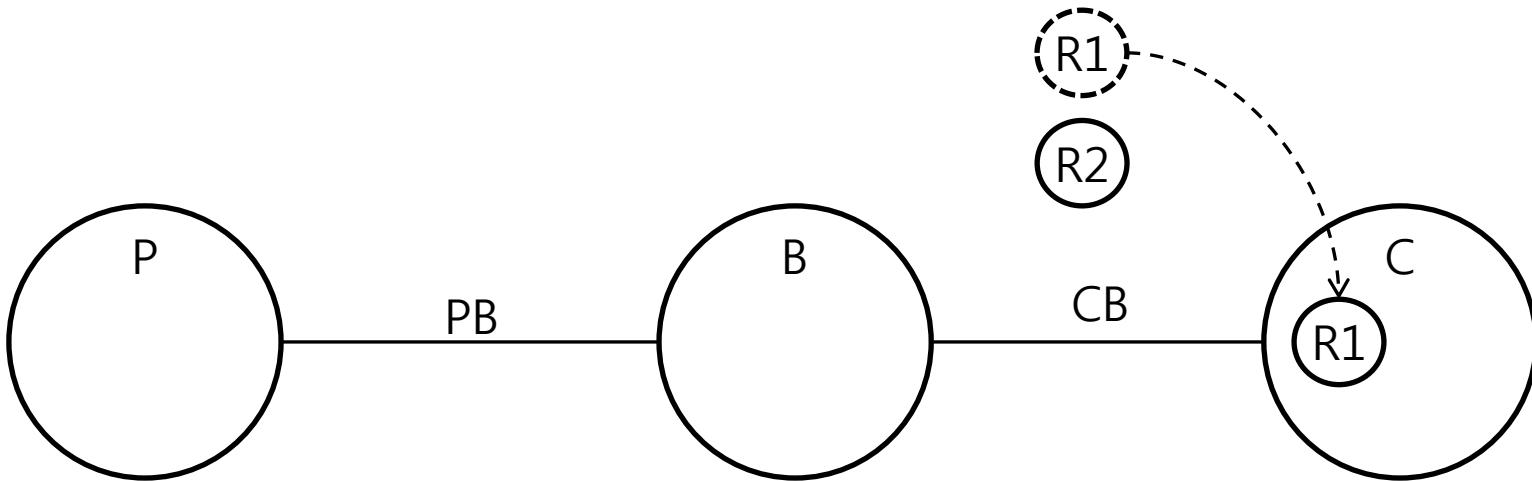
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = \text{get}\ R1.\text{get}\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



PBC Example (9) : $\delta_{4,2}$

- δ -Calculus

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

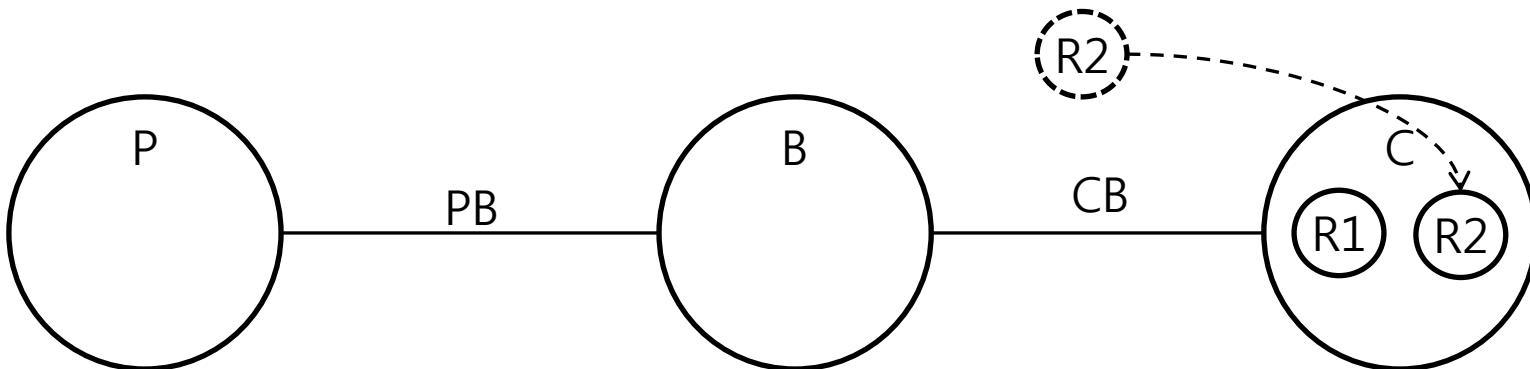
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.\textcolor{blue}{get\ R2.exit}$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.\textcolor{blue}{C\ get.exit}$$



PBC Example (10)

- δ -Calculus

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

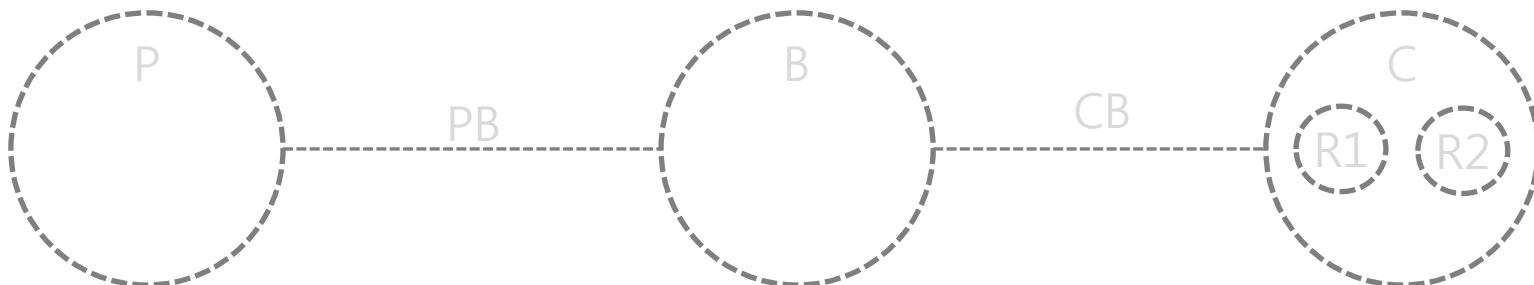
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



CASE 2

PBC Example (1) : τ_2

- δ -Calculus

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

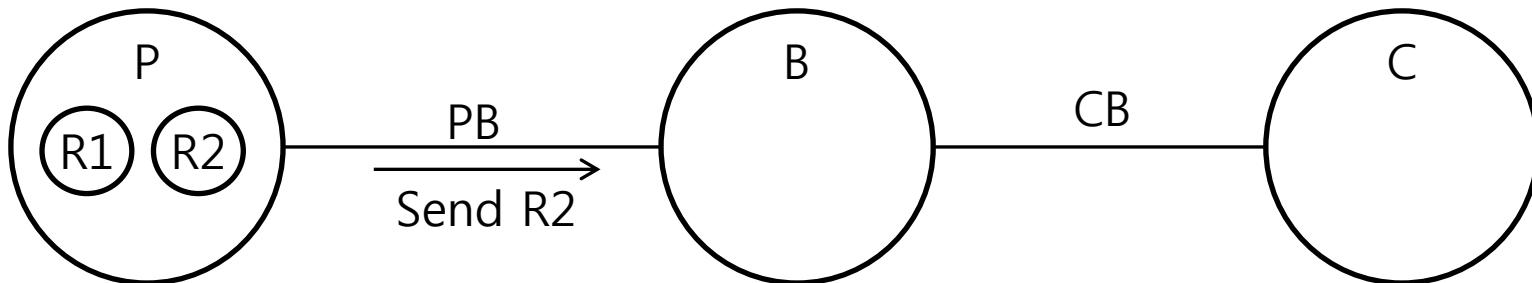
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



CASE 3: DEADLOCK

PBC Example (1) : τ_3

- δ -Calculus

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

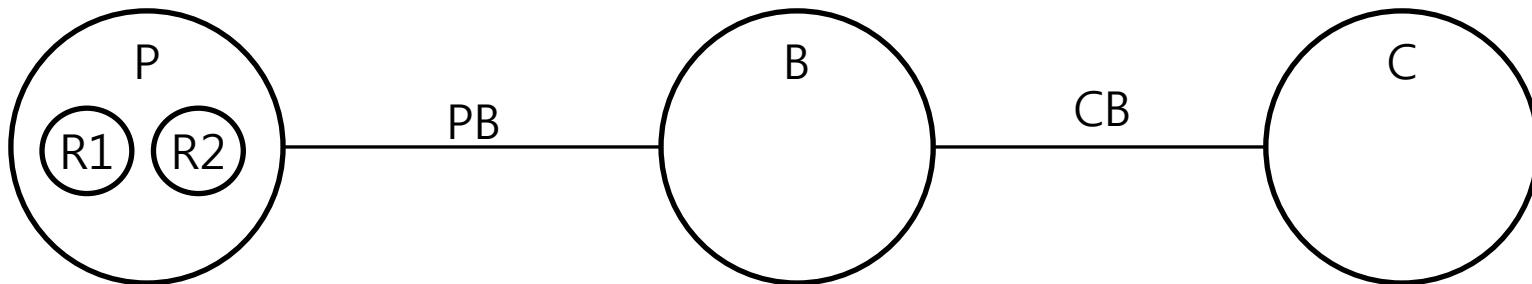
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



CASE 4: DEADLOCK

PBC Example (1) : τ_4

- δ -Calculus

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

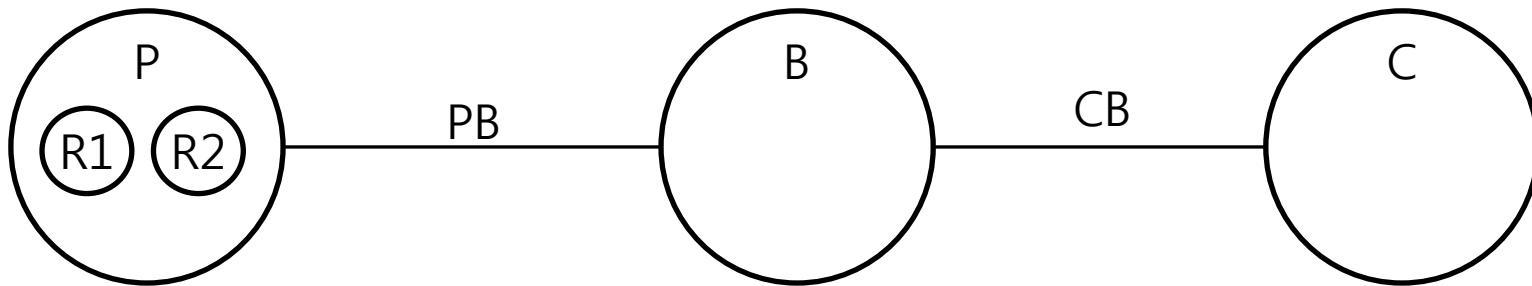
$$P = (PB(\overline{Send\ R1}).put\ R1.put\ R2 + PB(\overline{Send\ R2}).put\ R2.put\ R1).exit$$

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

$$C = get\ R1.get\ R2.exit$$

$$R1 = P\ put.B\ get.B\ put.C\ get.exit$$

$$R2 = P\ put.B\ get.B\ put.C\ get.exit$$



Temporal Properties

Interactions	Case 1	Case 2
$\tau_1 = P:PB, Send\ R1, B:PB(Send\ R1)$	0-1	X
$\tau_2 = P:PB, Send\ R2, B:PB(Send\ R2)$	X	0-1
$\delta_{1,1} = P:put\ R1, R1:P\ put$	1-2	2-3
$\delta_{1,2} = P:put\ R2, R2:P\ put$	2-3	1-2
$\delta_{2,1} = B:get\ R1, R1:B\ get$	2-3	3-4
$\delta_{2,2} = B:get\ R2, R2:P\ get$	3-4	2-3
$\delta_{3,1} = B:put\ R1, R1:B\ put$	4-5	4-5
$\delta_{3,2} = B:put\ R2, R2:B\ put$	5-6	5-6
$\delta_{4,1} = C:get\ R1, R1:C\ get$	5-6	5-6
$\delta_{4,2} = C:get\ R2, R2:C\ get$	6-7	6-7

ITL View

ADOxx: Modelling Toolkit (deltatest) - [P&C_tau_ITL (ITL)]

Model Edit View Process tools Delta calculus mechanisms Extras Window Help

Modelling

Explorer - Model groups

Models

- Execution model 14.05.2015-10:02:11
- GTS_REQ-EX pass 1
- GTS-REQ-EX pass 2
- New model
- P&C_tau_ITL
- P&C_tau_ITS

Mo... Mo...

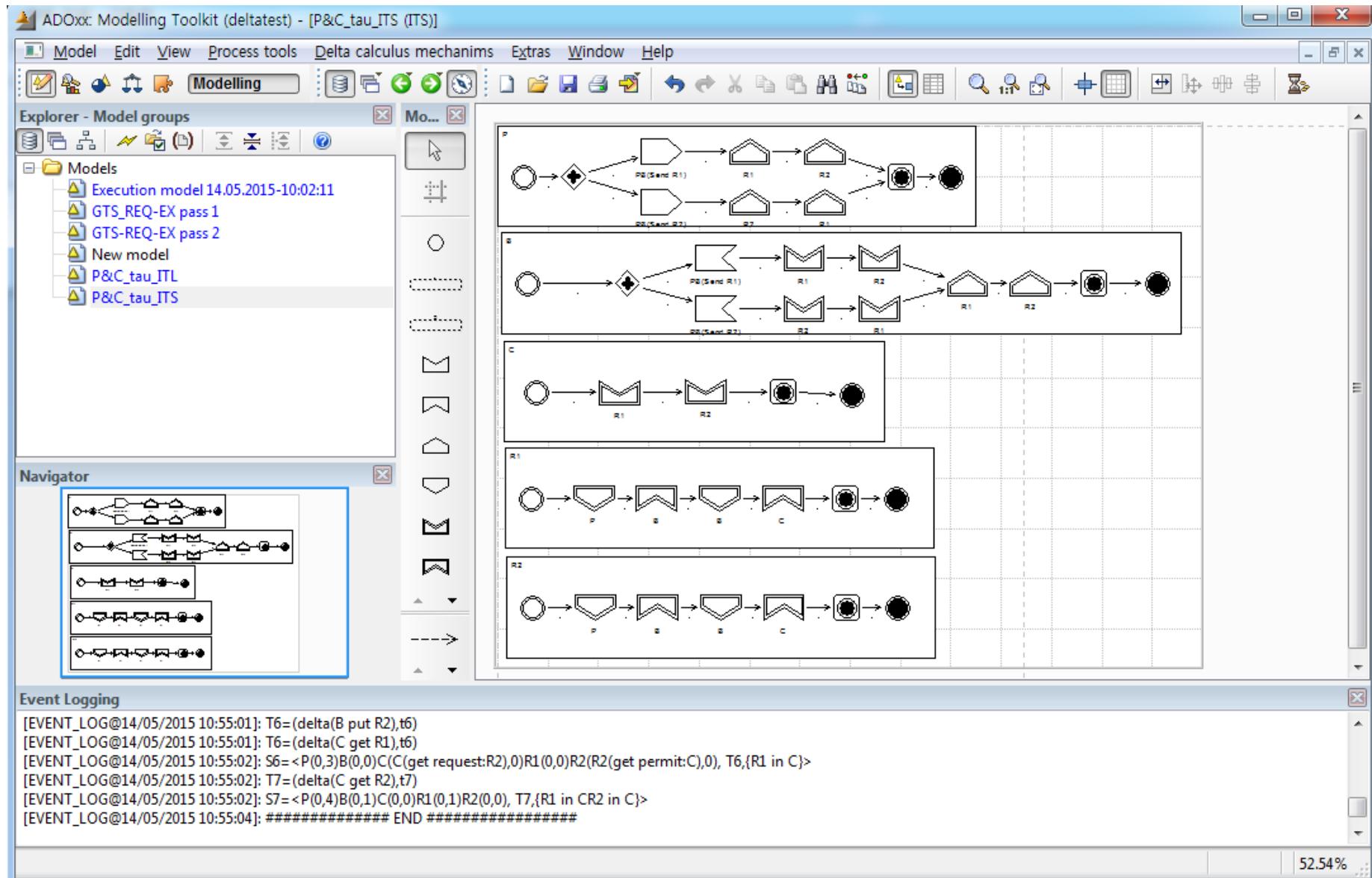
Navigator

Event Logging

```
[EVENT_LOG@14/05/2015 10:55:01]: T6=(delta(B put R2),t6)
[EVENT_LOG@14/05/2015 10:55:01]: T6=(delta(C get R1),t6)
[EVENT_LOG@14/05/2015 10:55:02]: S6=<P(0,3)B(0,0)C(get request:R2),0>R1(0,0)R2(R2(get permit:C),0), T6,{R1 in C}>
[EVENT_LOG@14/05/2015 10:55:02]: T7=(delta(C get R2),t7)
[EVENT_LOG@14/05/2015 10:55:02]: S7=<P(0,4)B(0,1)C(0,0)R1(0,1)R2(0,0), T7,{R1 in CR2 in C}>
[EVENT_LOG@14/05/2015 10:55:04]: ##### END #####
```

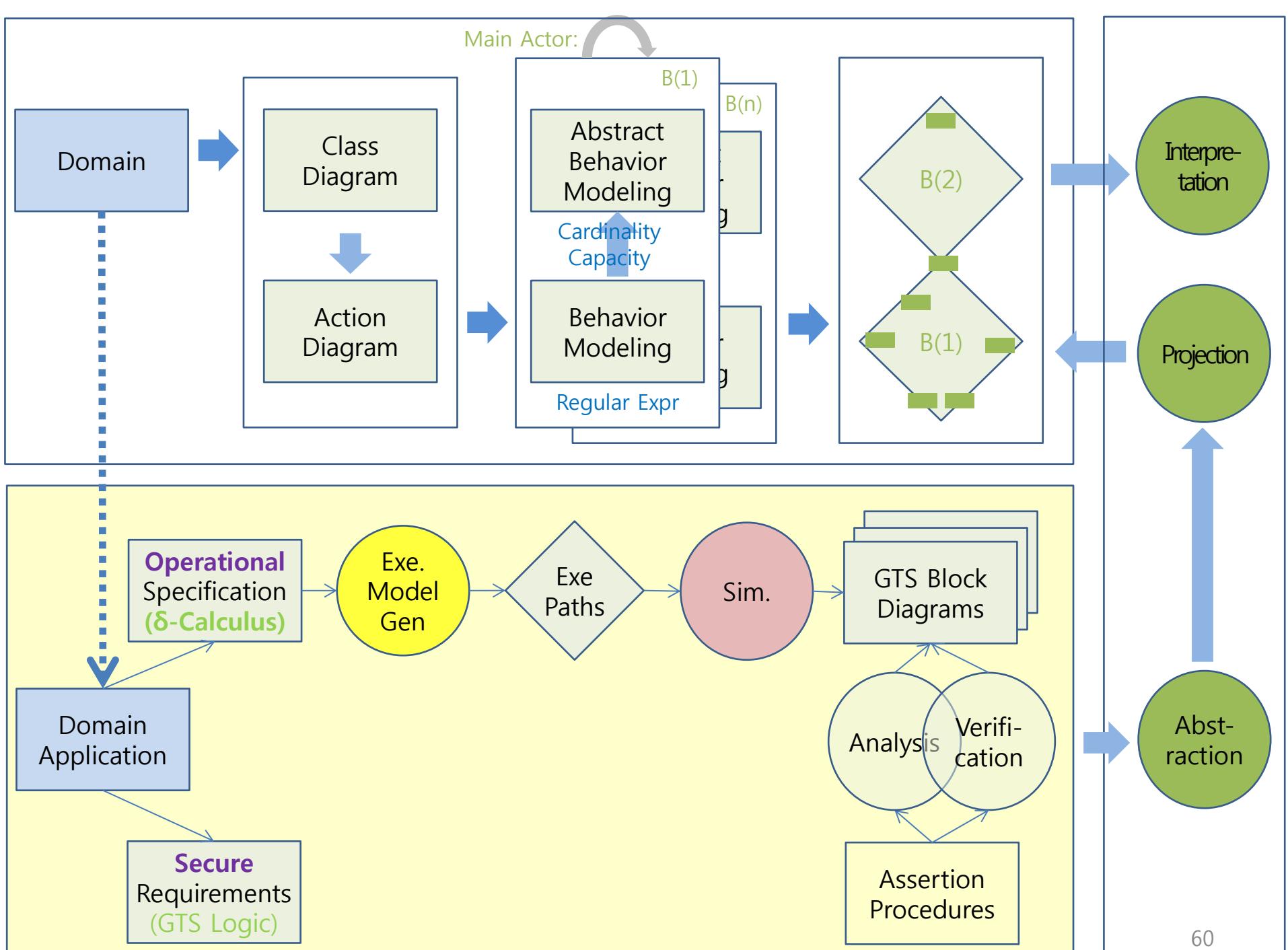
93%

ITS Views



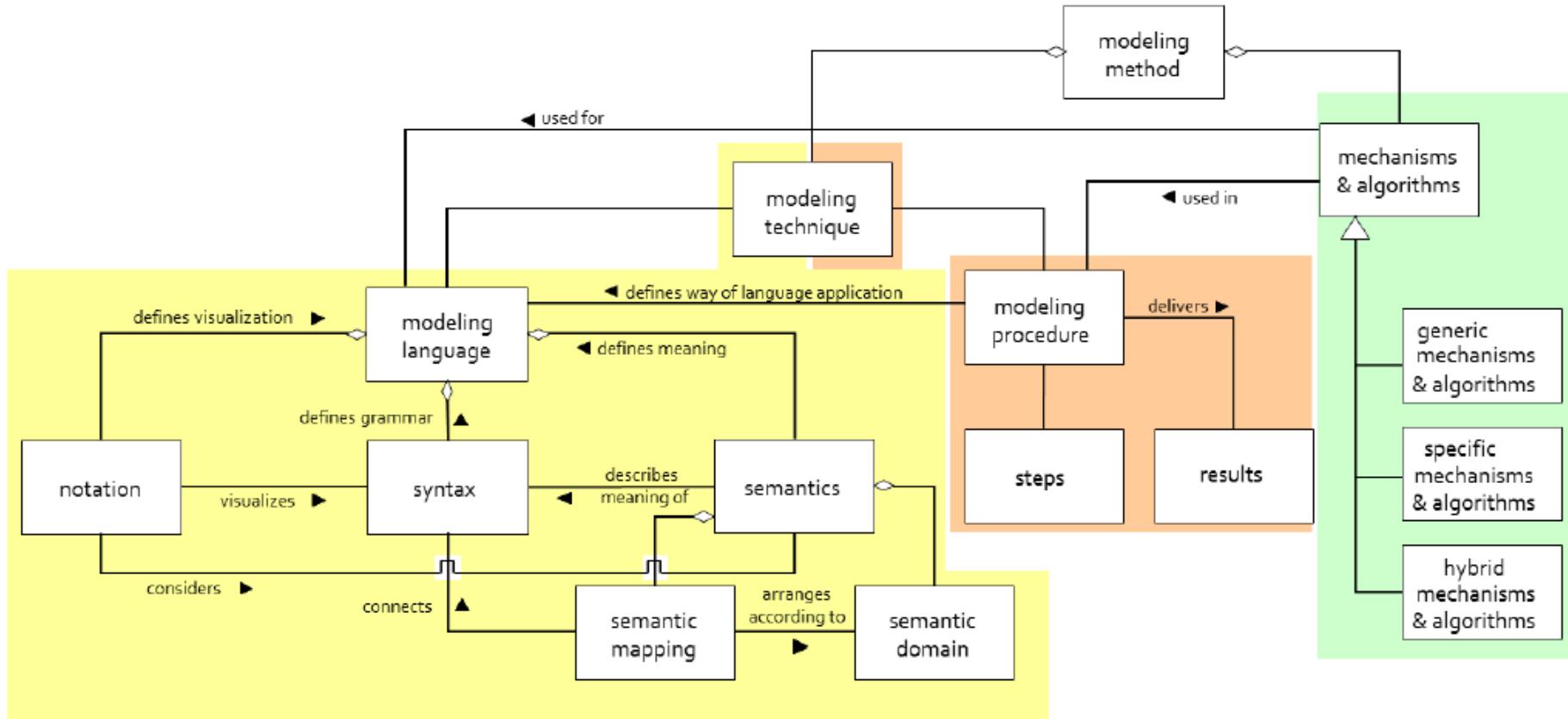
Step 2

Execution Model Generation



ADOxx Meta-Modeling Platform

$$\text{Method} = (T+MA) \cdot (L+P)$$



Karagiannis, D., Kühn, H.: „Metamodelling Platforms“. In Bauknecht, K., Min Tjoa, A., Quirmayer, G. (Eds.): Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2002, LNCS 2455, Springer, Berlin/Heidelberg, p. 182.

Execution Model

- Labelled Transition System
 - System States
 - Transitions
 - Global and local clocks.
- The process state s_j for Process P_i :
 - $s_{i,j} = \langle a_j, t_{i,j} \rangle$
 - $t_{i,j}$: the local time at the state.
- The system state:
 - $S_i = \langle s_{1,i_1}, s_{2,i_2}, \dots, s_{n,i_n}, T_i, I_i \rangle$
 - T_i : The global time at the system state S_i and
 - I_i is the inclusion relations of the processes for the system at the state.
- E.g.: $S = \langle \{P_1, P_2, \dots, P_n\}, C, I \rangle$
 - $S_0 = \langle s_{1,i_0}, s_{2,i_0}, \dots, s_{n,i_0}, T_0, I_0 \rangle$
 - $S_n = \langle s_{1,i_n}, s_{2,i_n}, \dots, s_{n,i_n}, T_n, I_n \rangle$

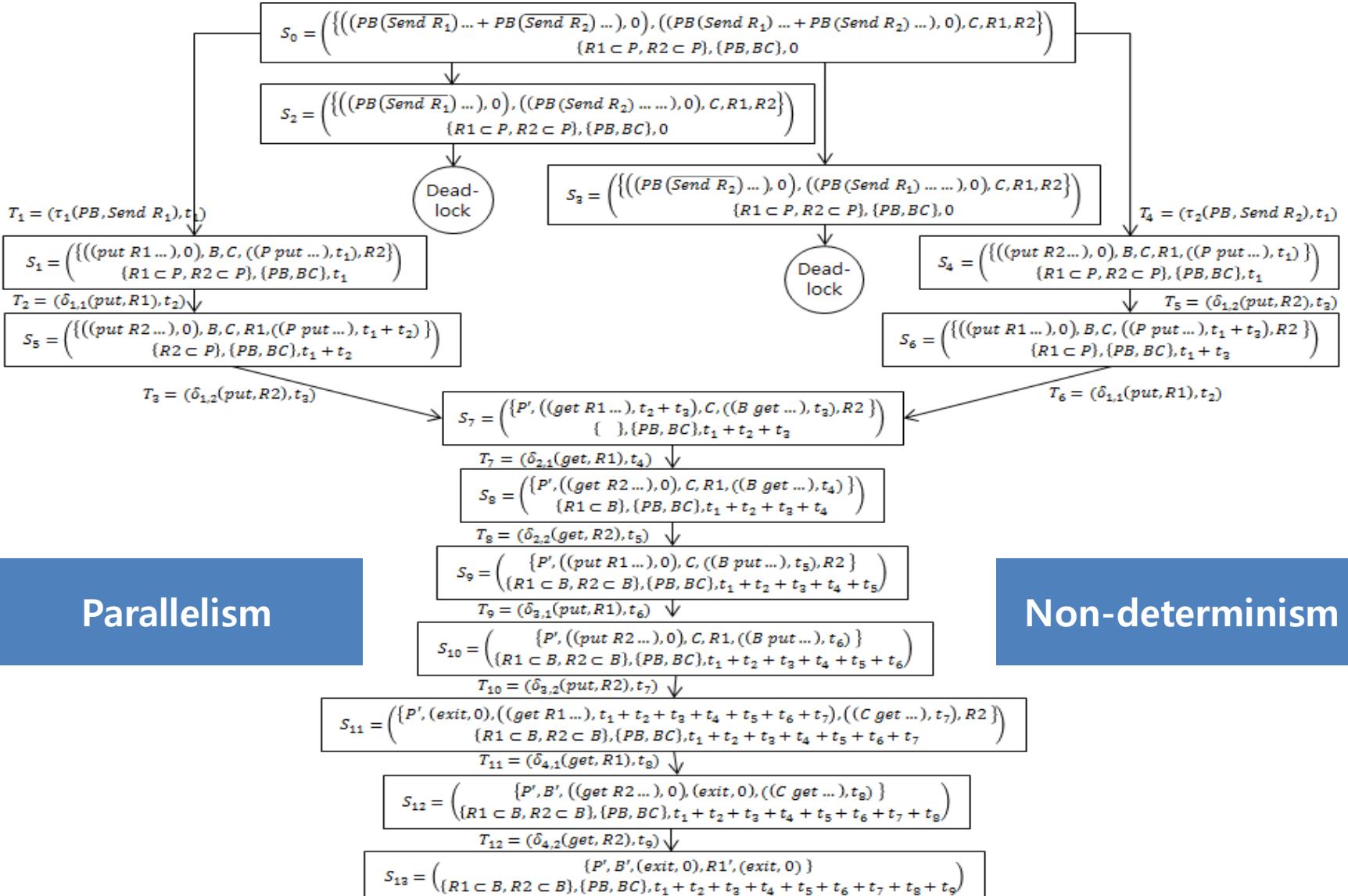
Communication (τ) Semantics

Action	$\frac{-}{r(a) \bullet P \xrightarrow{r(a)} P}$	Choic eL	$\frac{P \xrightarrow{A} P'}{P + Q \xrightarrow{A} P'}$
ChoiceR	$\frac{Q \xrightarrow{A} Q'}{P + Q \xrightarrow{A} Q'}$	ParlL	$\frac{P \xrightarrow{A} P'}{P Q \xrightarrow{A} P' Q'}$
ParlR	$\frac{Q \xrightarrow{A} Q'}{P Q \xrightarrow{A} P Q'}$	ParCo m	$\frac{P \xrightarrow{A} P', Q \xrightarrow{\bar{A}} Q'}{P Q \xrightarrow{\tau} P' Q'}$
NestO	$\frac{P \xrightarrow{A} P'}{P[Q] \xrightarrow{A} P'[Q]}$	NestI	$\frac{Q \xrightarrow{A} Q'}{P[Q] \xrightarrow{A} P[Q']}$
Nest-Com	$\frac{P \xrightarrow{A} P', Q \xrightarrow{\bar{A}} Q'}{P[Q] \xrightarrow{\tau} P'[Q']}$		

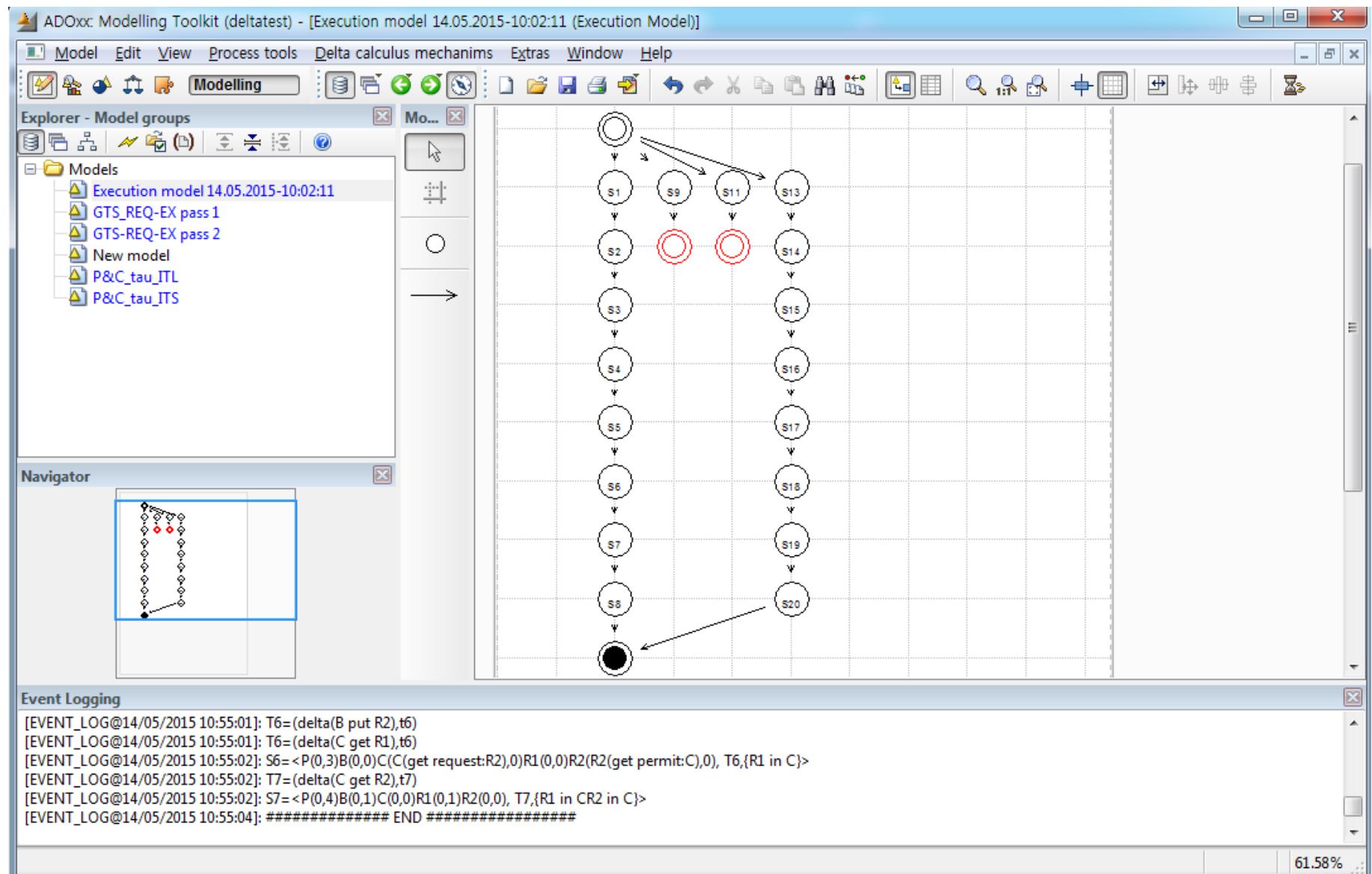
Movement (δ) Semantics

In	$\frac{P \xrightarrow{\text{in } Q} P', Q \xrightarrow{P \text{ in}} Q'}{P Q \xrightarrow{\delta} Q'[P']}$	Out	$\frac{P \xrightarrow{\text{out } Q} P', Q \xrightarrow{Q \text{ out}} Q'}{Q[P] \xrightarrow{\delta} P' Q'}$
Get	$\frac{P \xrightarrow{\text{get } Q} P', Q \xrightarrow{P \text{ get}} Q'}{P Q \xrightarrow{\delta} P'[Q']}$	Put	$\frac{P \xrightarrow{\text{put } Q} P', Q \xrightarrow{Q \text{ put}} Q'}{P[Q] \xrightarrow{\delta} P' Q'}$
InP	$\frac{P_{(n)} \xrightarrow{\text{in}^p Q} P_{(n)}'}{P_{(n)} Q_{(m)} \xrightarrow{\text{in}^p Q} Q_{(m)}[P_{(n)}']}$ ($n > m$)		
OutP	$\frac{P_{(n)} \xrightarrow{\text{out}^p Q} P_{(n)}'}{Q_{(m)}[P_{(n)}] \xrightarrow{\text{out}^p Q} P_{(n)}' Q_{(m)}}$ ($n > m$)		
GetP	$\frac{P_{(n)} \xrightarrow{\text{get}^p Q} P_{(n)}'}{P_{(n)} Q_{(m)} \xrightarrow{\text{get}^p Q} P_{(n)}'[Q_{(m)}]}$ ($n > m$)		
PutP	$\frac{P_{(n)} \xrightarrow{\text{put}^p Q} P_{(n)}'}{P_{(n)}[Q_{(m)}] \xrightarrow{\text{out}^p Q} P_{(n)}' Q_{(m)}}$ ($n > m$)		
InN	$\frac{P \xrightarrow{\text{in } Q} P', Q \xrightarrow{P \text{ in}} Q'}{P Q[R] \xrightarrow{\delta} Q'[P' R]}$	GetN	$\frac{P \xrightarrow{\text{get } Q} P', Q \xrightarrow{P \text{ get}} Q'}{P[R] Q \xrightarrow{\delta} P'[R Q']}$

Example: PBC

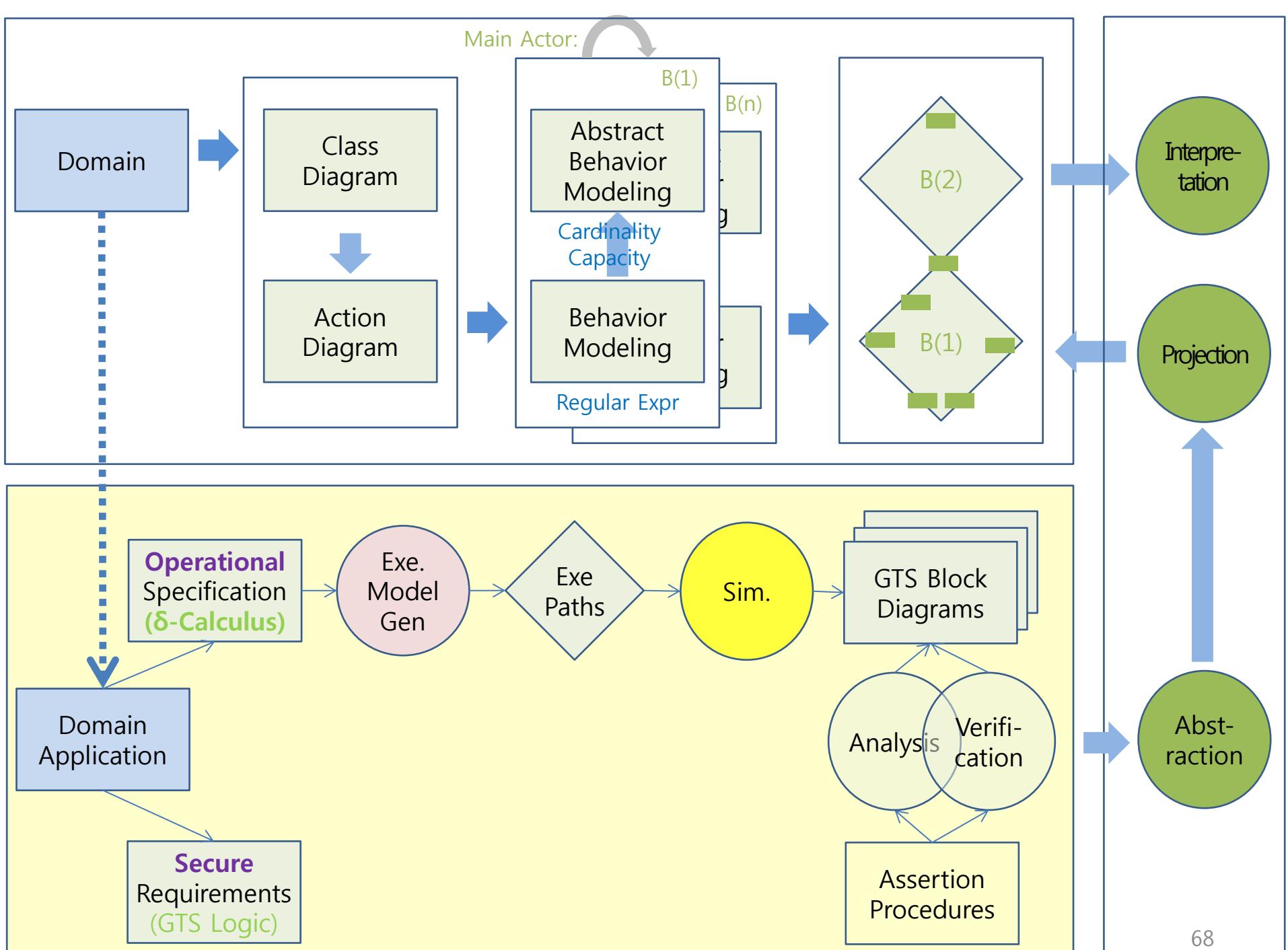


Execution Model: Paths for PBC



Step 3

Simulation



Algorithm

```
Simulation(M: Model)
{
    IF ((M ∈ Models) & (type (M) = ITL))
    {
        Select ITS model of M;
    }
    Else
    {
        Select ITL_M Model of M;
        Select ITS_M Model of M;
    }
    M_ITL = ITL_Loader(ITL_M);      // Load the ITL Model
    M_ITS = ITS_Loader(ITS_M);      // Load the ITS Model
    ITL_ITS_Mapper(M_ITL, M_ITS);  // Map the ITS to the ITL

    {
        P = Select_Simulation_Path(Execution_Model(M));
    } while (!P);

    PI = Load_Path_Info(P);
    Data = Sim_Init(); // Initialization for Simulation Environment

    //Search for Start node and set Start state for all processes
    Sim(Data);
}
```

```
Sim(Data) {
    while() {
        while (Search Branch) { // branch = choice or parallel
            Set the next node environment of Branch;
            IF (choice Node) {
                IF(∃ path info){
                    Select a path from path info;
                }
                ELSE {
                    Select a Branch after the select-path pop-up window;
                }
            }
        }

        If (All processes are terminated) {
            Exit();
        }

        For(Nodes of the current_node_list) {
            Action(node);
        }
        Update the current_node_list to the next state;
        Change the node state; //Standing, Ready, Waiting, Executing,
        Deadlock

        IF (Check Deadlock) {
            Exit();
        }
        Time++;
    }
}
```

Selecting a Path

ADOxx: Modelling Toolkit (deltatest)

Model Edit View Process tools Delta calculus mechanisms Extras Window Help

Modelling

Explorer - Model groups

Models

- Execution model 14.05.2015-10:02:11
- GTS_REQ-EX pass 1
- GTS-REQ-EX pass 2
- New model
- P&C_tau_ITL
- P&C_tau_ITS

Navigator

Event Logging

```
[EVENT_LOG@14/05/2015 10:55:02]: S6=<P(0,3)B(0,0)C(C(get request:R2),0)R1(0,0)R2(R2(get permit:C),0), T6,{R1 in C}>
[EVENT_LOG@14/05/2015 10:55:02]: T7=(delta(C get R2),t7)
[EVENT_LOG@14/05/2015 10:55:02]: S7=<P(0,4)B(0,1)C(0,0)R1(0,1)R2(0,0), T7,{R1 in CR2 in C}>
[EVENT_LOG@14/05/2015 10:55:04]: ##### END #####
[EVENT_LOG@14/05/2015 11:05:46]: ##### Simulation START #####
[EVENT_LOG@14/05 11:06:13]: ##### Simulation START #####
```

Diagram Area:

Execution Model 14.05.2015-10:02:11 (Execution Model)

P&C_tau_ITS (ITS)

Select Path

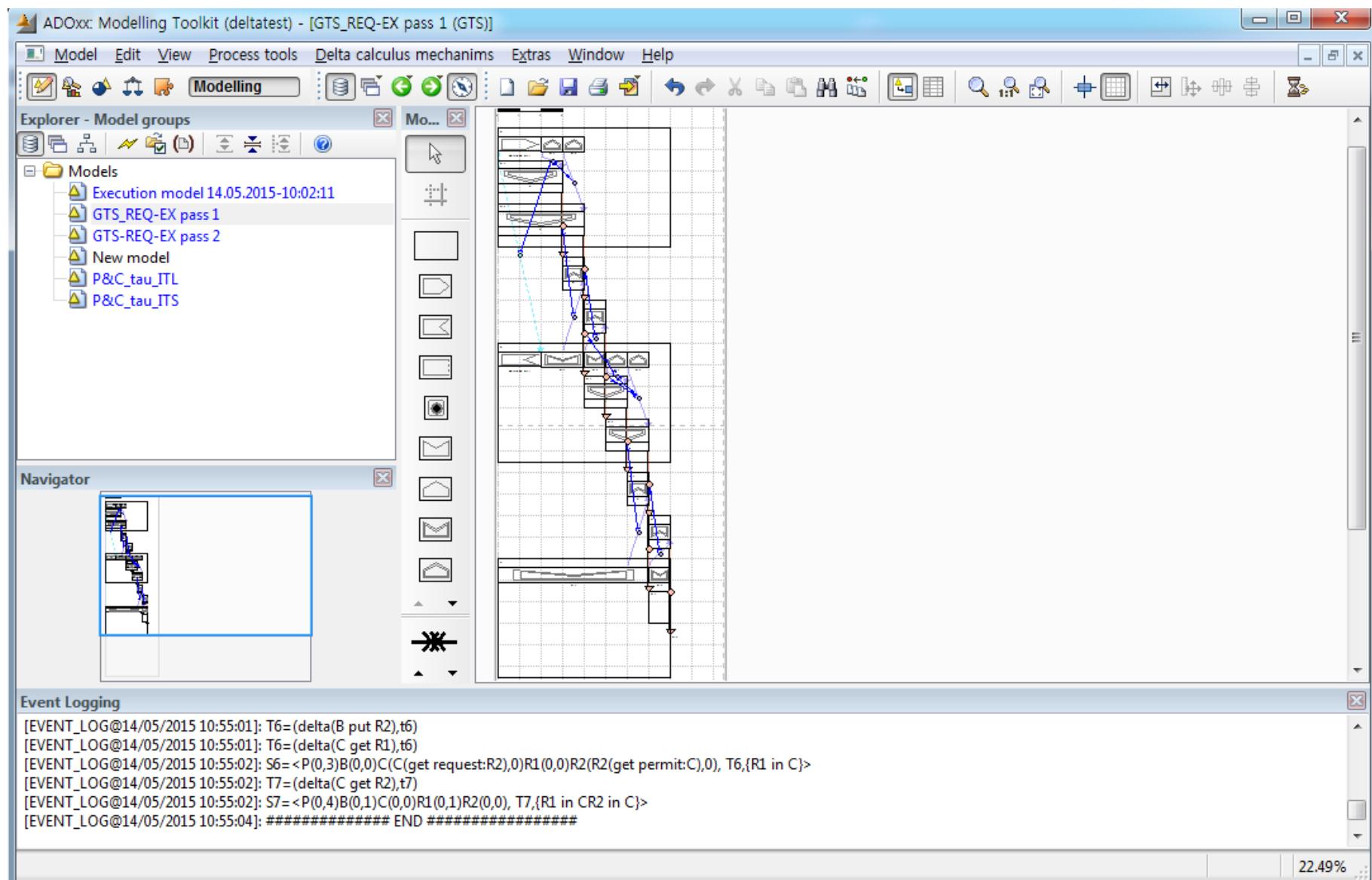
Choose simulate path:

- Start S1 S2 S3 S4 S5 S6 S7 S8 End
- Start S9 S10(Deadlock)
- Start S11 S12(Deadlock)
- Start S13 S14 S15 S16 S17 S18 S19 S20 End

Select Cancel

35.17%

Simulation



Selecting Another Path

ADOxx: Modelling Toolkit (deltatest)

Model Edit View Process tools Delta calculus mechanisms Extras Window Help

Modelling

Execution model 14.05.2015~10:02:11 (EXECUTION MODEL)

Explorer - Model groups

Models

- Execution model 14.05.2015-10:02:11
- GTS_REQ-EX pass 1
- GTS-REQ-EX pass 2
- New model
- P&C_tau_ITL
- P&C_tau_ITS

Navigator

Event Logging

[EVENT_LOG@14/05/2015 10:55:02]: S6=<P(0,3)B(0,0)C(C(get request:R2),0)R1(0,0)R2(R2(get permit:C),0), T6,{R1 in C}>
[EVENT_LOG@14/05/2015 10:55:02]: T7=(delta(C get R2),t)
[EVENT_LOG@14/05/2015 10:55:02]: S7=<P(0,4)B(0,1)C(0,0)R1(0,1)R2(0,0), T7,{R1 in CR2 in C}>
[EVENT_LOG@14/05/2015 10:55:04]: ##### END #####
[EVENT_LOG@14/05/2015 11:05:46]: ##### Simulation START #####
[EVENT_LOG@14/05/2015 11:06:13]: ##### Simulation START #####

Diagram (Main View): A state transition diagram with 20 states (S1-S20). States S9 and S10 are highlighted with red circles. State S20 is the final state, indicated by a double circle. Transitions are labeled with Greek letters (psi, phi, rho, sigma, tau, delta) and some with numbers (e.g., 1, 2, 3).

Diagram (Right Panel): A hierarchical state transition diagram for "P&C_tau_ITS (ITS)" showing four main parallel regions (a, b, c, d) with various internal transitions.

Select Path Dialog:

Choose simulate path:

- Start S1 S2 S3 S4 S5 S6 S7 S8 End
- Start S9 S10(Deadlock)
- Start S11 S12(Deadlock)
- Start S13 S14 S15 S16 S17 S18 S19 S20 End

Select Cancel

35.17%

Deadlock

ADOxx: Modelling Toolkit (deltatest)

Model Edit View Process tools Delta calculus mechanisms Extras Window Help

Modelling

Explorer - Model groups

Models

- Execution model 14.05.2015-10:02:11
- GTS_REQ-EX pass 1
- GTS-REQ-EX pass 2
- New model
- P&C_tau_ITL
- P&C_tau_ITS

Navigator

P&C_tau_ITS (ITS)

P&C_tau_ITL (ITL)

ADOxx: Modelling Toolkit (deltat...)

Error: Deadlock!!

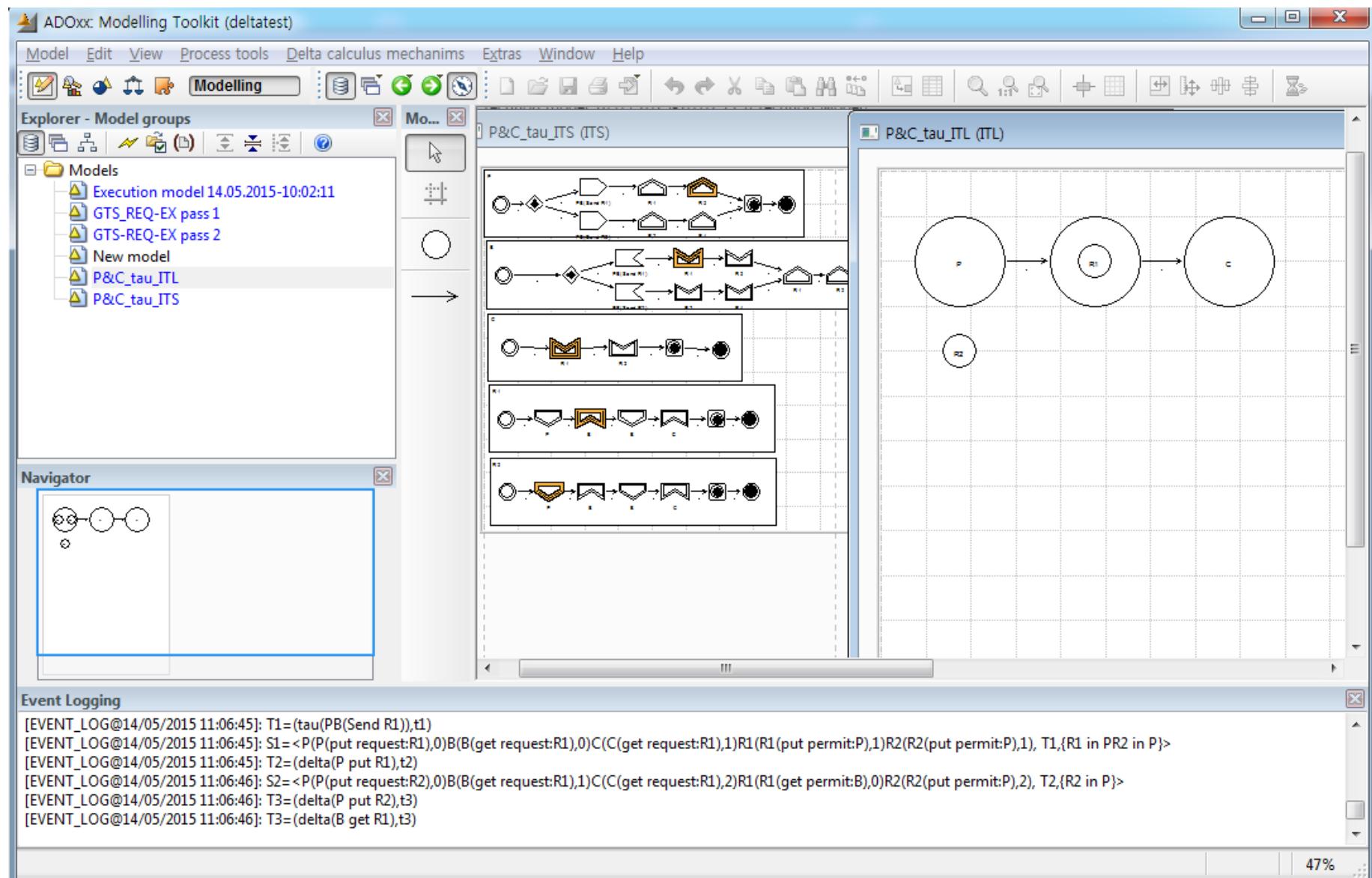
확인

Event Logging

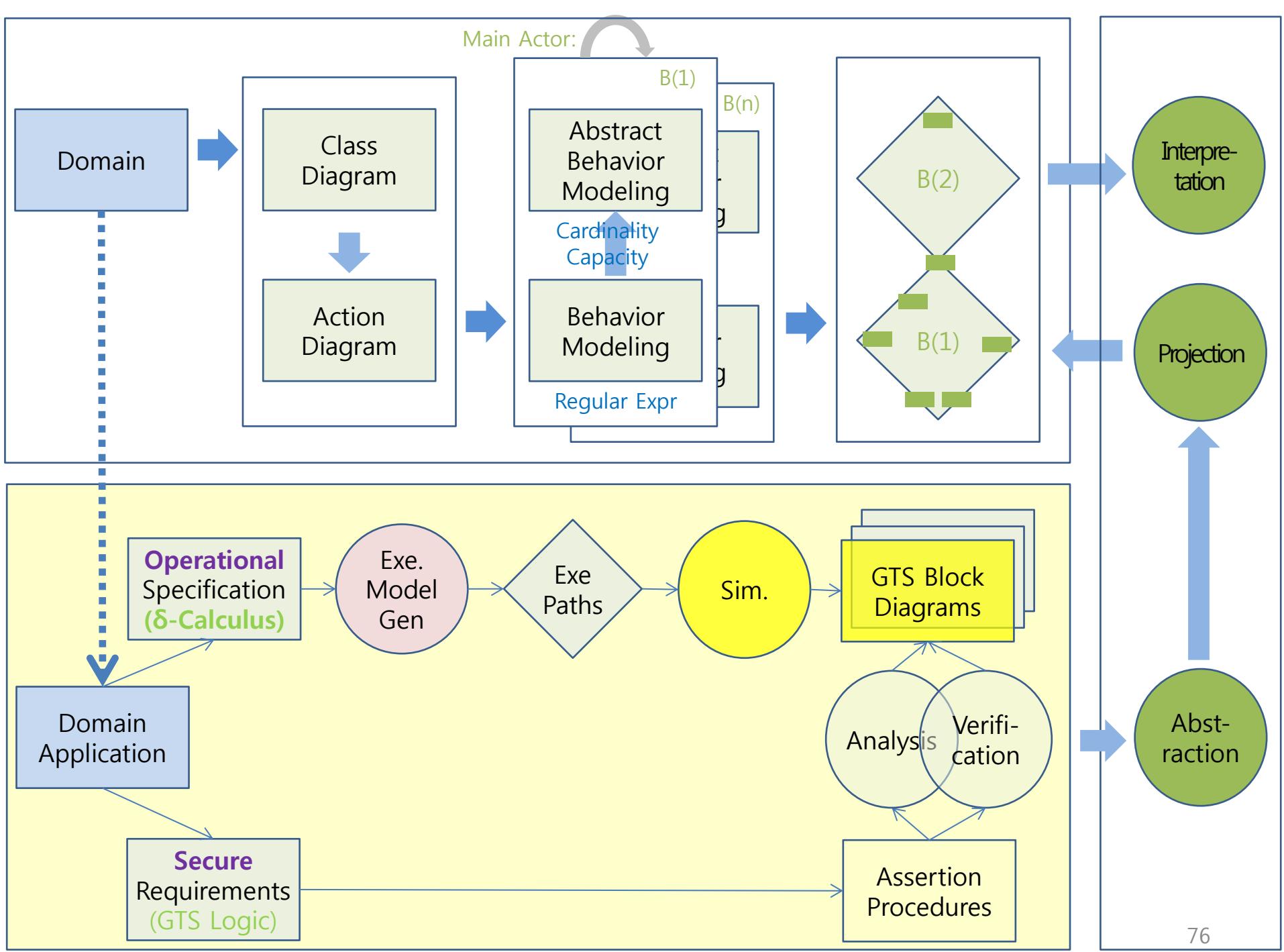
```
[EVENT_LOG@14/05/2015 11:06:48]: T7=(delta(C get R2),t7)
[EVENT_LOG@14/05/2015 11:06:48]: S7=<P(0,4)B(0,1)C(0,0)R1(0,1)R2(0,0), T7,{R1 in CR2 in C}>
[EVENT_LOG@14/05/2015 11:06:50]: ##### END #####
[EVENT_LOG@14/05/2015 11:07:15]: ##### Simulation START #####
[EVENT_LOG@14/05/2015 11:07:19]: S0=<P(PB(send:Send R1),0)B(PB(receive:Send R2),0)C(C(get request:R1),0)R1(R1(put permit:P),0)R2(R2(put permit:P),0), T0,{R1 in PR2 in P}>
[EVENT_LOG@14/05/2015 11:07:20]: S1=<P(PB(send:Send R1),1)B(PB(receive:Send R2),1)C(C(get request:R1),1)R1(R1(put permit:P),1)R2(R2(put permit:P),1), T1,{R1 in PR2 in P}>
```

47%

Deadlock State

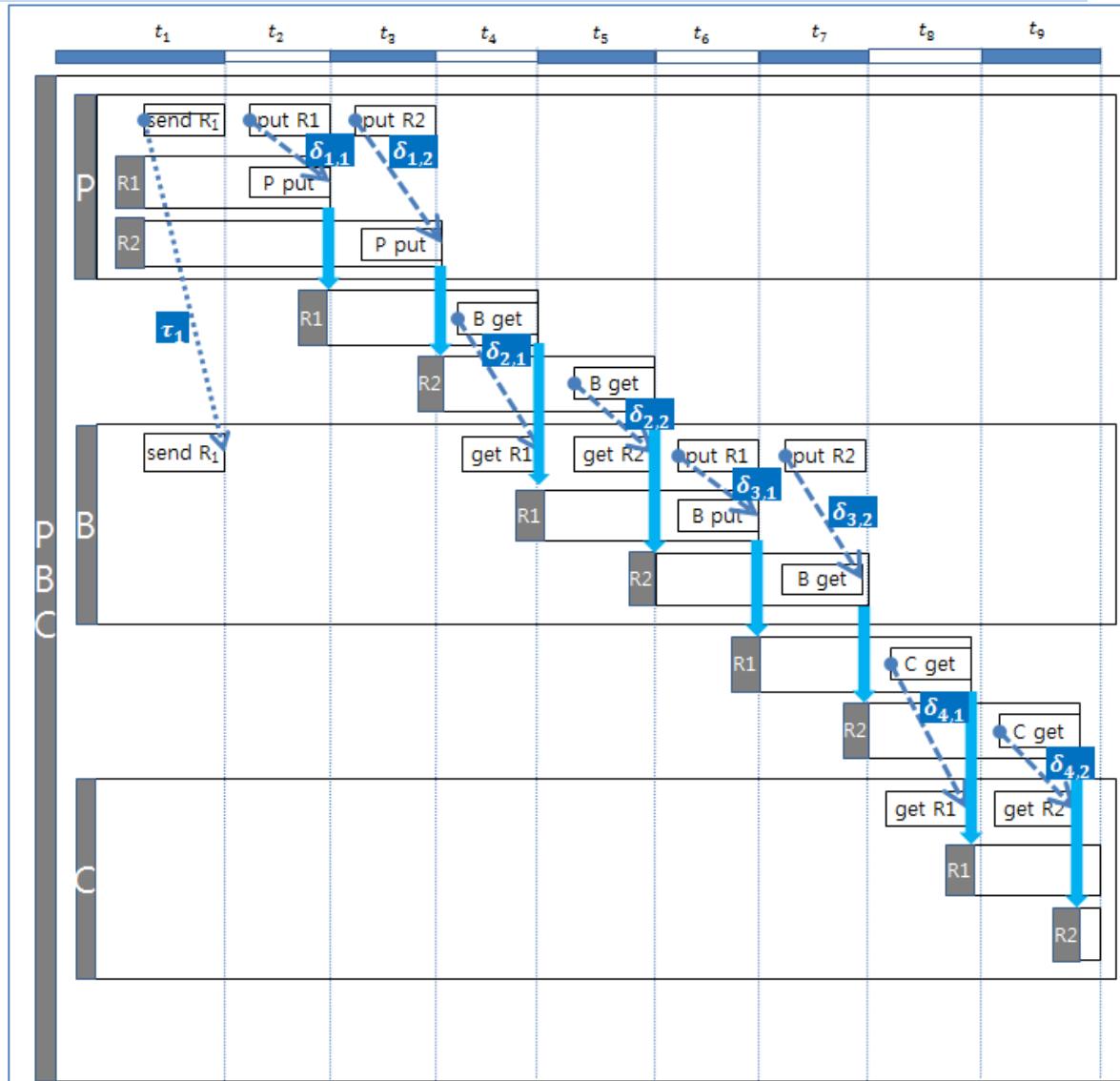


SIMULATION OUTPUT



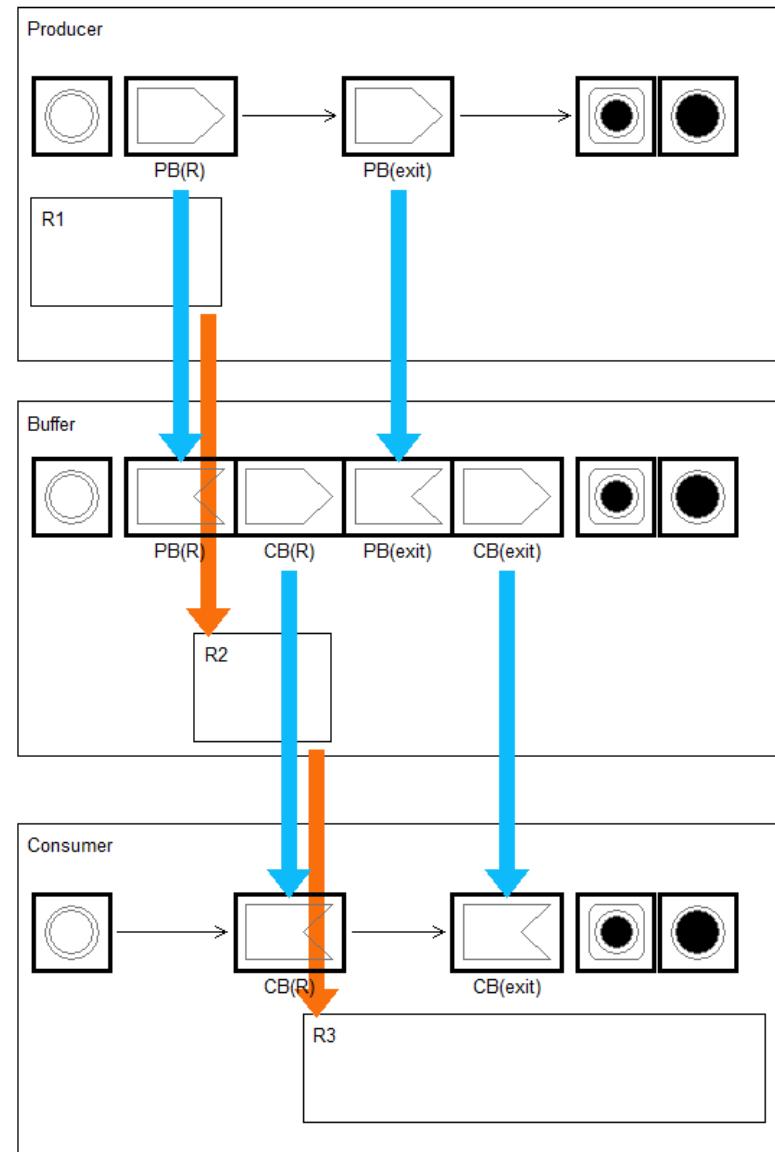
Blocks

- Block: Space x Time
 - System
 - Processes
 - Actions
 - Interactions
- Interactions
 - Communication
 - Movements
 - Control
- Properties
 - Channel: c_i
 - Message: m_i
 - Priority: n_i

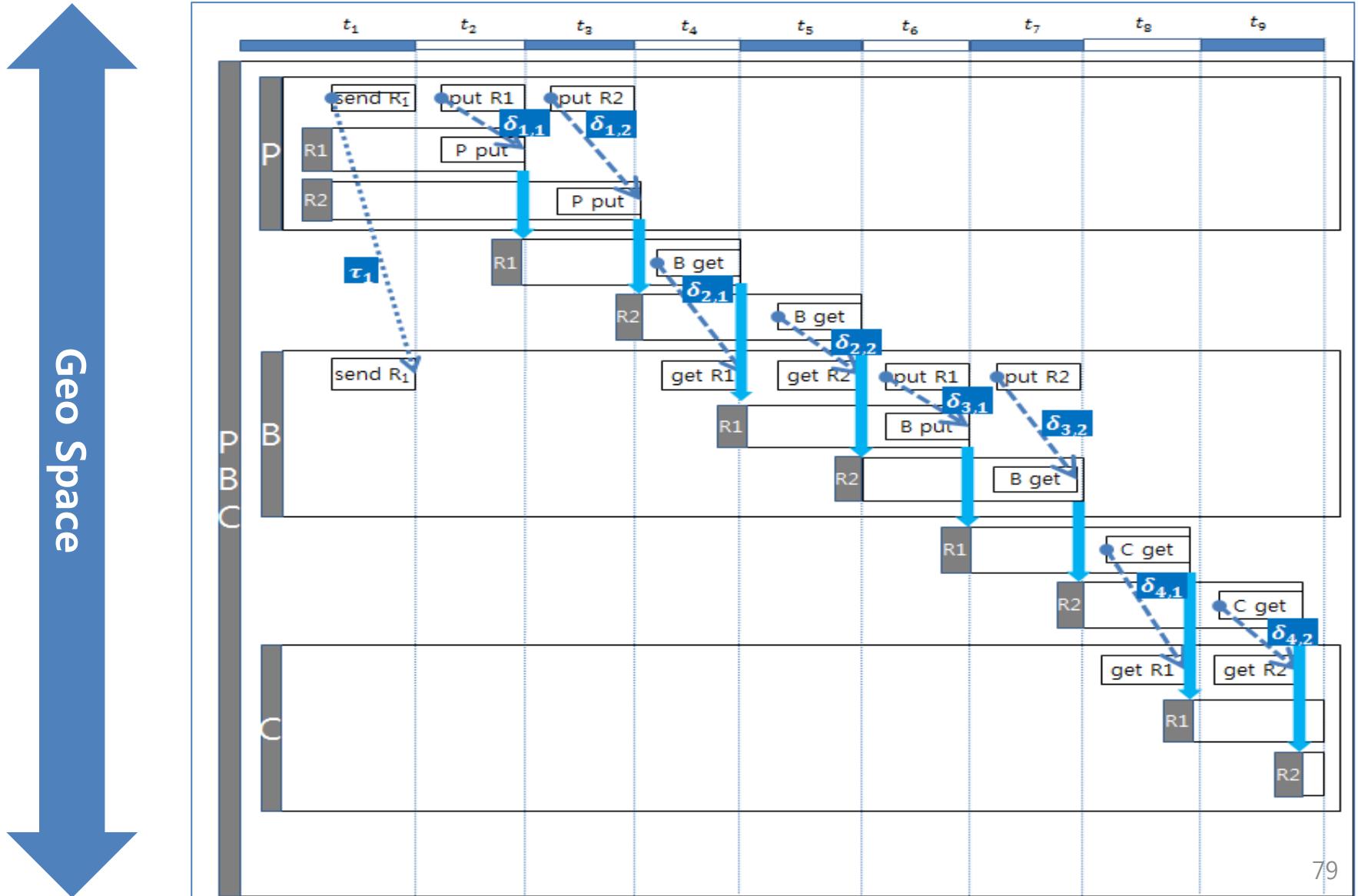


Blocks

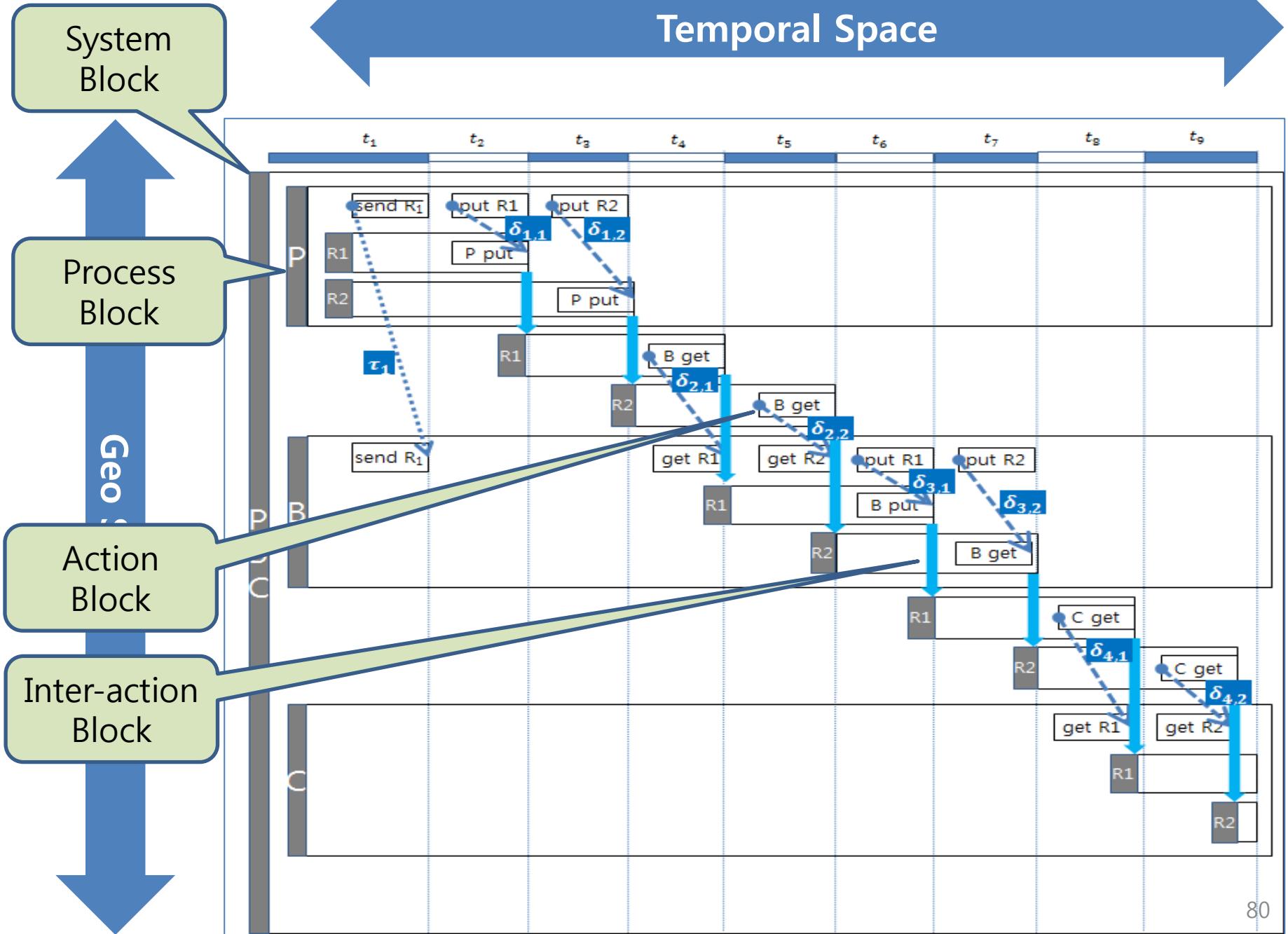
- Block: Space x Time
 - System
 - Processes
 - Actions
 - Interactions
- Interactions
 - Communication
 - Movements
 - Control
- Properties
 - Channel: c_i
 - Message: m_i
 - Priority: n_i



Temporal Space



Temporal Space



Properties: Restrictions

- Syntax
 - Actions in P cannot be overlapped in any space in any time.
 - If $P \parallel Q$, there is no overlap in space, but in time.
 - If $P[Q]$, there is an overlap of Q over P in space during a period of time.
- Semantics
 - τ for communication:
 - Q and P must be in the same time period.
 - δ for movements:
 - For active *in* movement, Q and P must be in the same space at the same time period.
 - For active *out* movement, Q must be in P or P must be in Q at the same time period.
 - For passive *in* movement, Q and P must be in the same space at the same time period.
 - For passive *out* movement, Q must be in P or P must be in Q at the same time period.
 - ρ for process control:
 - For *terminate*, *suspend* and *wake*, Q must be in P or P must be in Q at the same time period.
- Requirements: Dependencies among blocks
 - Inclusions/Precedencies/Duration
 - Conditionals/And/Or/Not

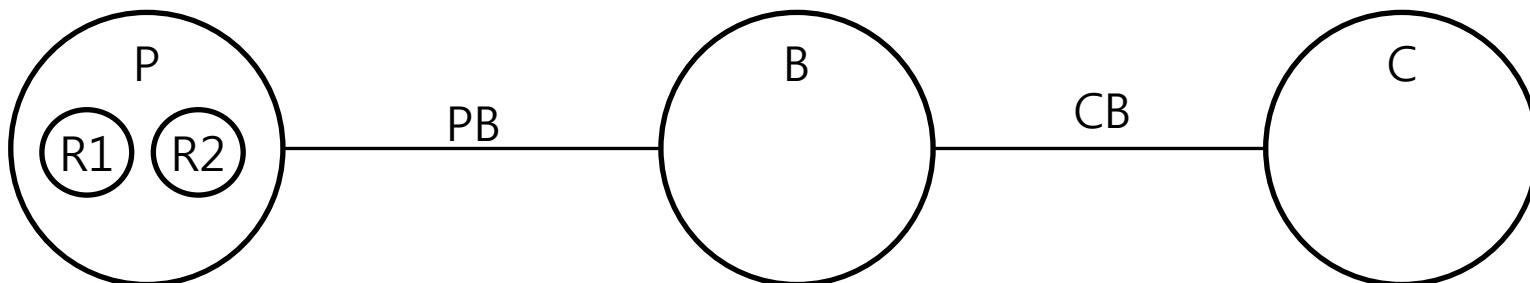
1. Motivation
2. Modeling
 - 1) Specification: δ -Calculus
 - 2) Verification: GTS Logic**
 - 3) EMS Example
3. SAVE Tool
4. Cyber-Physical Systems Application
5. Summary
6. Discussion

2.2

GTS Logic: Secure Specification & Verification

Requirements: PBC Example

- Operational Requirements
 - *Producer* produces two resources, R1 and R2.
 - *Producer* stores the resources in *Buffer* in order.
 - *Producer* informs *Buffer* of the order of R1 and R2, or R2 and R1.
 - *Consumer* consumes the resources from *Buffer* in order.
 - The order of the consumption is formed to *Buffer* by *Consumer*.
- **Secure Requirements**
 - Security:
 - The order 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.
 - Safety
 - The resources produced by *Producer* should be consumed by *Consumer* less than 5 minutes.



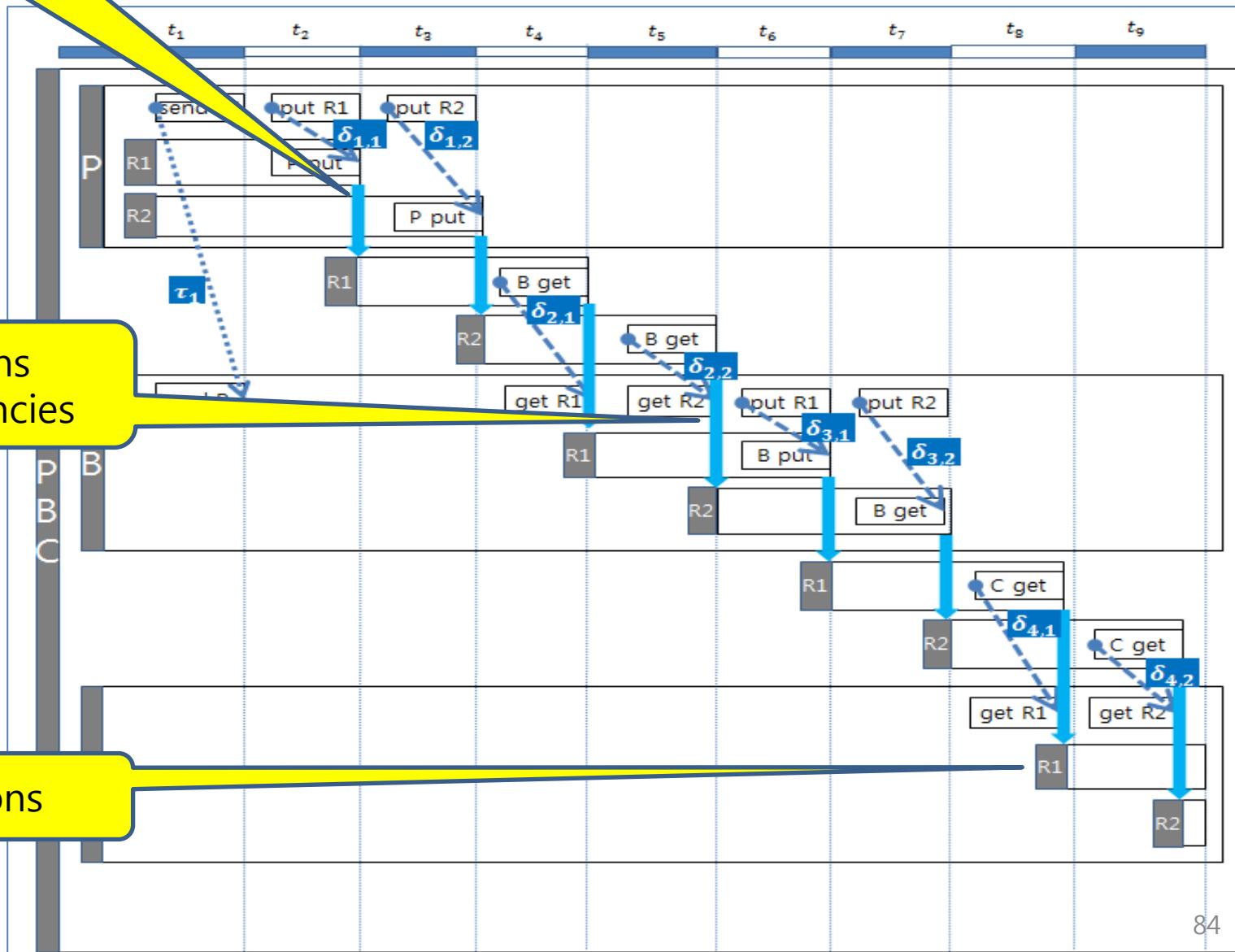
Temporal Space

Predicates

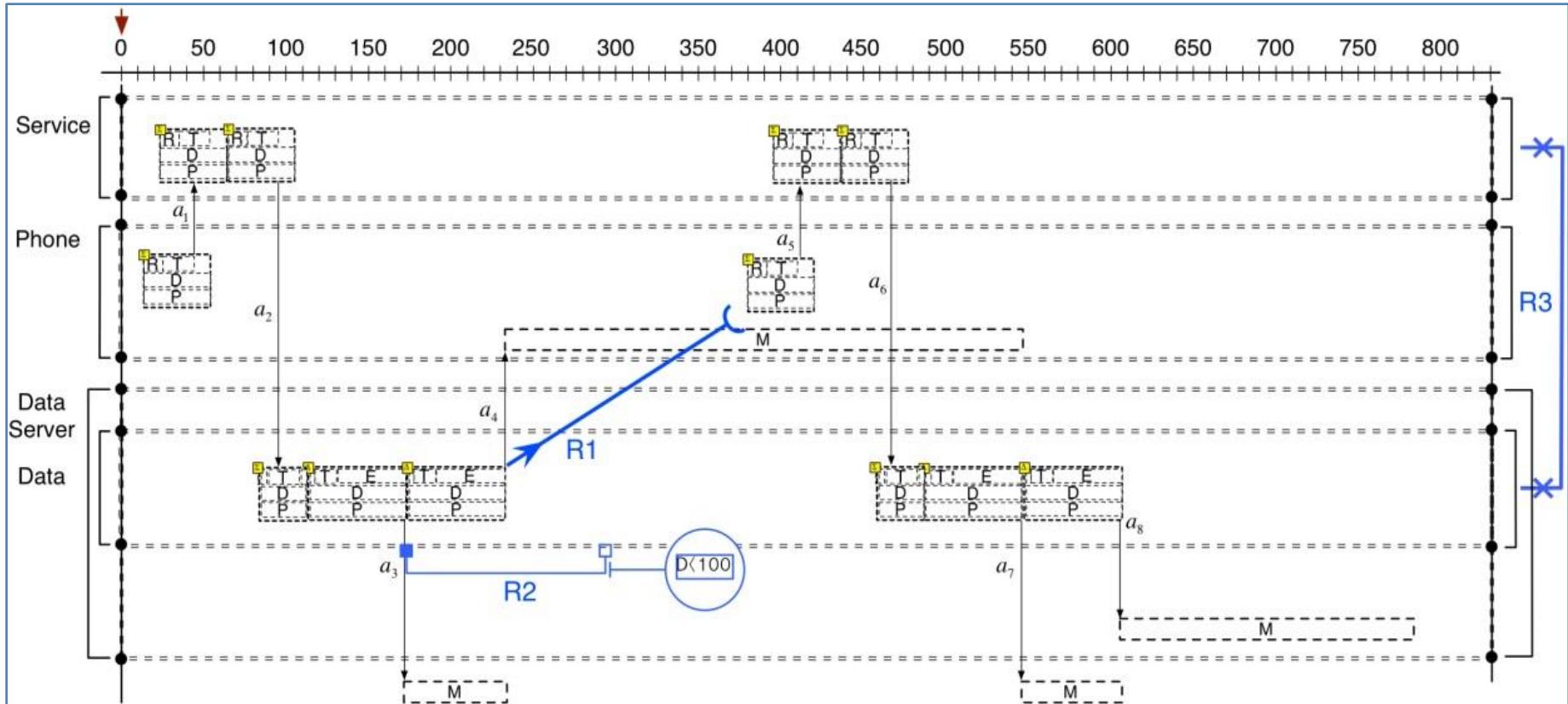
Relations
Dependencies

Space

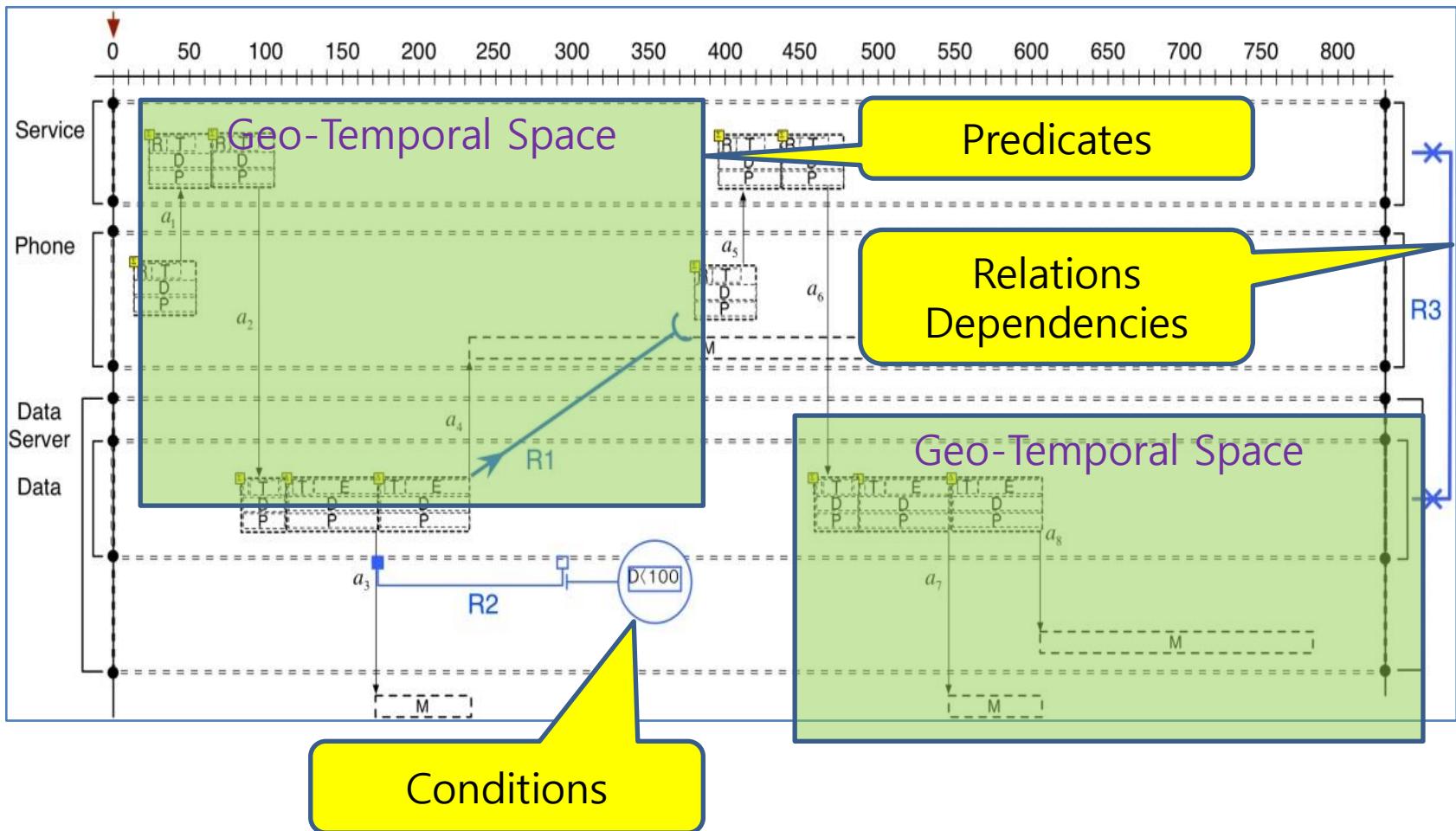
Conditions



Conceptual Example



Conceptual Example



GTS Visual Logic

	Syntax	Semantics	Syntax	Semantics
Geographical requirements	A o —— o B	A can be in B. B can be in A.	A ✗ —— ✗ B	A cannot be in B. B cannot be in A.
	A ✗ —— o B	A can be in B. B cannot be in A.	A —————← B	B should be in A, always.
	A o —— ✗ B	B can be in A. A cannot be in B.	A ➤———— B	A should be in B, always.
Temporal requirements	a →————(b	Action <i>a</i> should be performed before <i>b</i> in time.	a)————(b	Action <i>a</i> and <i>b</i> can be performed concurrently in the same time period.
	a)————→ b	Action <i>b</i> should be performed after <i>a</i> .	a +————←+ b	Action <i>a</i> and <i>b</i> cannot be performed concurrently in the same time period.
Interval	□————□	Open Interval	■————■	Close Interval
Conditions	—→ AUC	Condition for requirements: time, frequency, behavior, etc.		

Visual Logic: Geo (1)



- A can be in B. B can be in A
 - $((A \sqcap_g B) \sqcap_g A = (A \vee \emptyset)) \wedge ((A \sqcap_g B) \sqcap_g B = (B \vee \emptyset))$
- A
A diagram showing point A crossed out with a large red X. Point B is connected to A by a horizontal line segment, indicating A is not in B but B is in A.
- A can be in B. B cannot be in A
 - $((A \sqcap_g B) \sqcap_g A = \emptyset) \wedge ((A \sqcap_g B) \sqcap_g B = (B \vee \emptyset))$

Visual Logic: : Geo (2)



- - A cannot be in B. B can be in A
 - $((A \sqcap_g B) \sqcap_g A = (A \vee \emptyset)) \wedge ((A \sqcap_g B) \sqcap_g B = \emptyset)$
- - A cannot be in B. B cannot be in A
 - $((A \sqcap_g B) \sqcap_g A = \emptyset) \wedge ((A \sqcap_g B) \sqcap_g B = \emptyset)$



Visual Logic : Geo (3)

A ————— B

- - B should be in A, always.
 - $B \subset A$
- - A should be in B, always.
 - $A \subset B$

Visual Logic : Temporal (1)



- Action a should be performed before b in time.
• $(a < b) \vee (a \leq b)$



- Action a should be performed after b in time.
• $(b < a) \vee (b \leq a)$

Visual Logic : Temporal (2)



- Action a and b can be performed concurrently in the same time period.

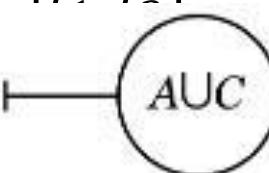
- $a \hat{=} b$



- Action a and b cannot be performed concurrently in the same time period.

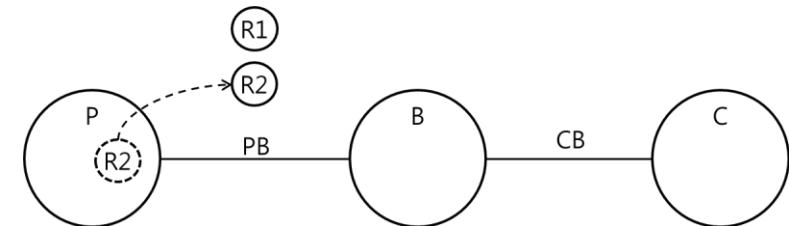
- $a \sqcap_t b = \emptyset$

Visual Logic: Interval & Conditions

- 
- - Open interval
 - (t_1, t_2)
- 
- - Close interval
 - $[t_1, t_2]$
- 
- - Conditions
 - Depends on conditions.

Interactions

- τ : Communication
 - $\tau_1 = (P: PB, (\overline{Send\ R1}), B: PB(Send\ R1))$
 - $\tau_2 = (P: PB, (\overline{Send\ R2}), B: PB(Send\ R2))$
- δ : Movements
 - $\delta_{1,1} = (P: put\ R1, R1: P\ put)$
 - $\delta_{1,2} = (P: put\ R2, R2: P\ put)$
 - $\delta_{2,1} = (B: get\ R1, R1: B\ get)$
 - $\delta_{2,2} = (B: get\ R2, R2: P\ get)$
 - $\delta_{3,1} = (B: put\ R1, R1: B\ put)$
 - $\delta_{3,2} = (B: put\ R2, R2: B\ put)$
 - $\delta_{4,1} = (C: get\ R1, R1: C\ get)$
 - $\delta_{4,2} = (C: get\ R2, R2: C\ get)$



Requirements: GTS Logic

$Rq1 = \tau_1 \rightarrow \forall i: (\delta_{i,1} < \delta_{i,2}) [i = 1,2]$

$Rq2 = \tau_2 \rightarrow \forall i: (\delta_{i,2} < \delta_{i,2}) [i = 1,2]$

$Rq3 = \tau_1 \vee \tau_2 < \delta_{1,1} \wedge \delta_{1,2}$

$Rq4 = \delta_{1,1} < B: (get\ R1)$

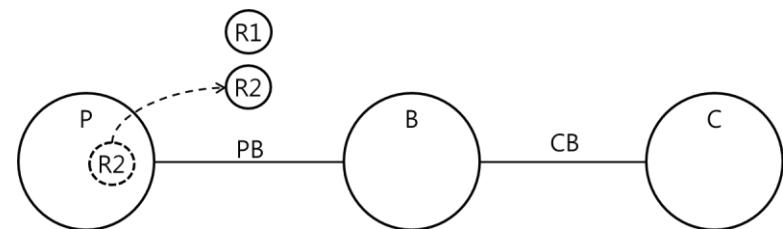
$Rq5 = \delta_{1,2} < B: (get\ R2)$

$Rq6 = \delta_{2,1} < B: (put\ R1)$

$Rq7 = \delta_{2,2} < B: (put\ R2)$

$Rq8 = \delta_{3,1} < C: (get\ R1)$

$Rq9 = \delta_{3,2} < C: (get\ R2)$



Requirements Verification

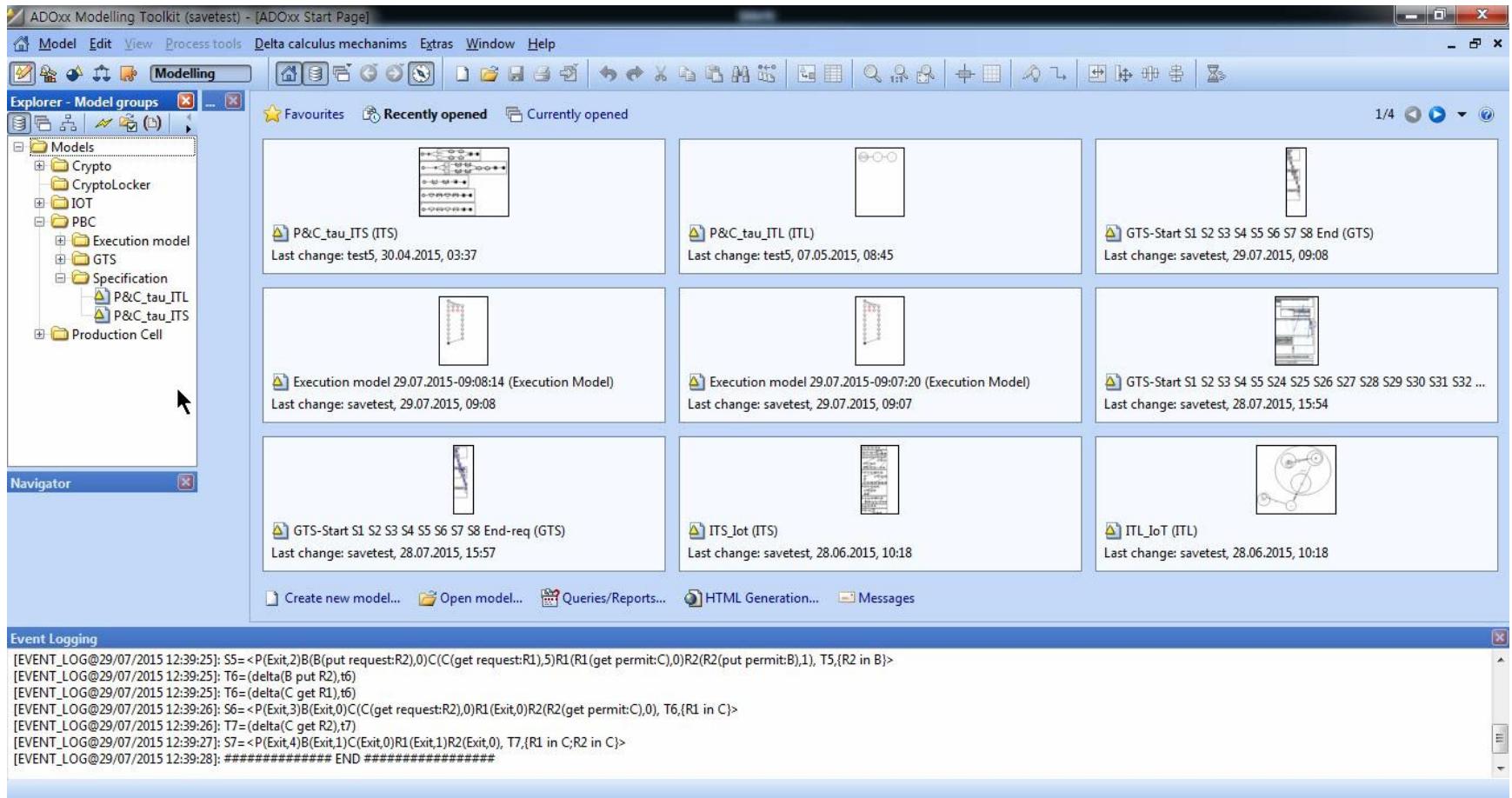
The screenshot shows the ADOxx Modelling Toolkit interface with a state transition diagram. The diagram consists of states labeled P, R1, and R2, connected by transitions. Annotations include:

- A blue callout bubble contains the requirement: $Rq1 = \tau_1 \rightarrow \forall i: (\delta_{i,1} < \delta_{i,2}) [i = 1,2]$. It points to a transition labeled τ_1 and two places labeled $\delta_{1,1}$ and $\delta_{1,2}$.
- A blue callout bubble contains the requirement: $Rq4 = \delta_{1,1} < B: (get\ R1)$. It points to a place labeled $\delta_{1,1}$ and a state labeled B.
- A large yellow starburst graphic with the text "WYSWYG" and "What You See is What You Get" is overlaid on the bottom right.

The interface includes a menu bar (Model, Edit, View, Process tools, Delta calculus/mechanisms, Extras, Window, Help), a toolbar, an Explorer panel showing model groups and files, a Navigator panel, and an Event Logging panel at the bottom.

SAVE

DEMO: PBC



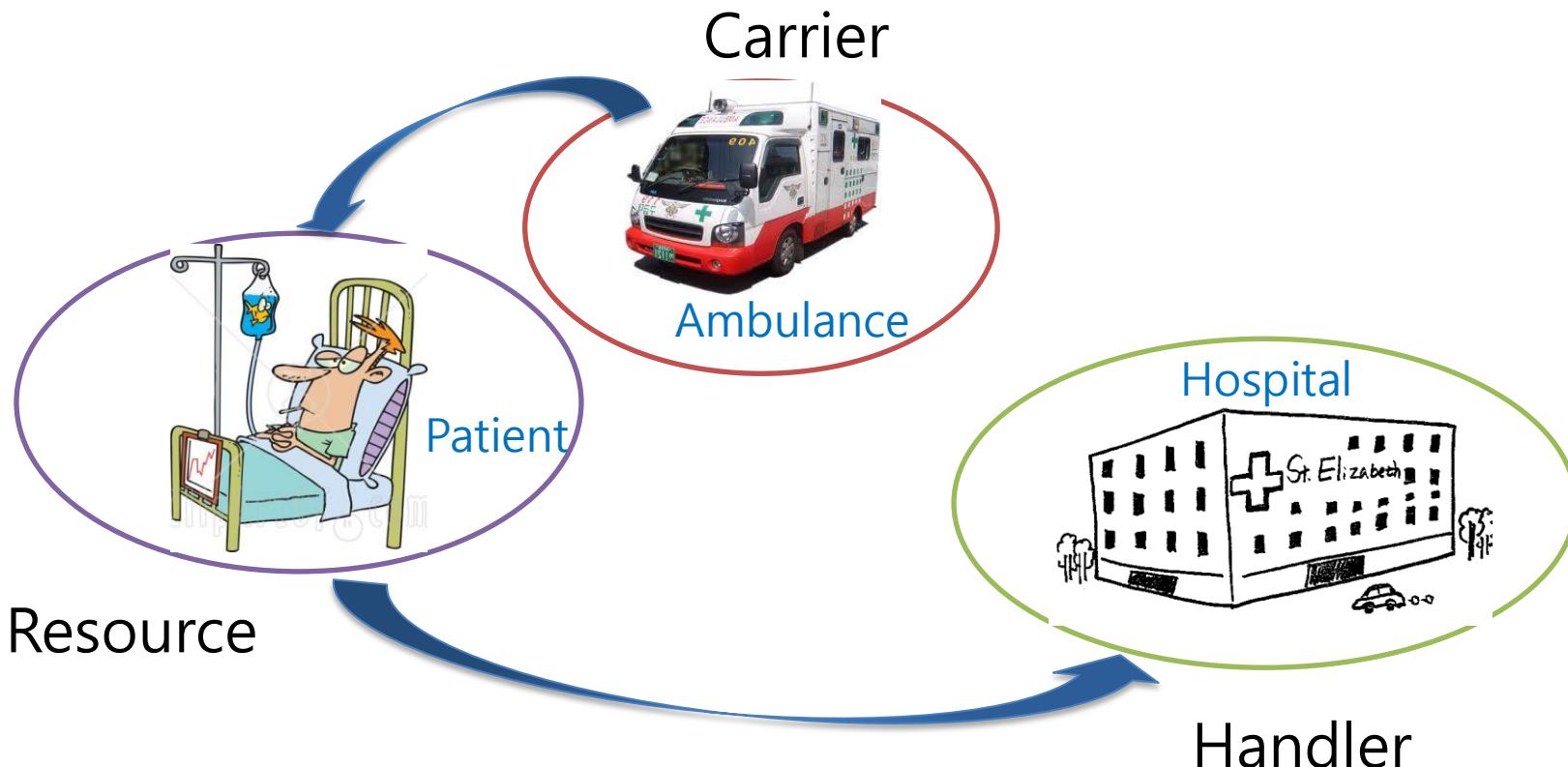
1. Motivation
2. Modeling
 - 1) Specification: δ -Calculus
 - 2) Verification: GTS Logic
 - 3) EMS Example**
3. SAVE Tool
4. Cyber-Physical Systems Application
5. Summary
6. Discussion

2.3

Example: Emergency Medical System (EMS)

Emergency Medical Systems

- Main theme
 - “Ambulances transport patients to hospitals.”

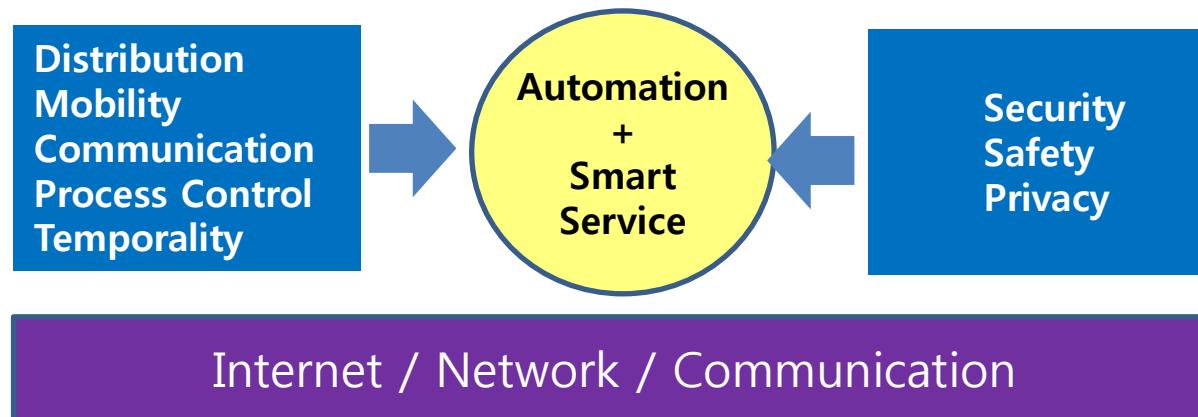


Operational Requirements

- Patients should call Control Center of their situations.
- The center orders 911 or Hospitals to handle the patients with Ambulances.
- Ambulances go to the locations where the patients are, and transport them to Hospitals.
- Ambulances follow orders from 911 or Hospitals.
- Ambulances call Hospitals to let Doctors to be prepared to treat Patients.

Smart EMS / IoT

- Operational Requirements
 - Patients should **call** Control Center of their situations.
 - The center **orders** 911 or Hospitals to **handle** the patients with Ambulances.
 - Ambulances **go to** the locations where the patients are, and **transport** them to Hospitals.
 - Ambulances **follow** orders from 911 or Hospitals.
 - Ambulances **call** Hospitals to let Doctors to be **prepared** to **treat** Patients.



Configuration: 35 Processes

- Main theme: System (1)
 - Control Center: 1
 - Carrier: 3 Ambulances
 - A in Hospital A
 - B, C in 911 Center
 - Sources: 5 Places
 - 4 Houses: A, B, C, D
 - 1 School: E
 - Resources: 8 Patients
 - 1 Patient w/ heart disease in House A
 - 2 Patient w/ high blood pressure in House B, C.
 - 1 Patient w/ diabetes in House D
 - 1 Patient w/ high blood pressure in School E
 - 2 Patient w/ food poisoning in School E
 - Destination: 3 Hospitals
 - A: 1 Lounge; 3 Doctors; 1 Surgery
 - B: 1 Lounge; 3 Doctors ; 1 Surgery
 - C: 1 Lounge; 2 Doctors ; 1 Surgery

Specification (1): 5

$CS = ((CALL(HAHBP).ORDER(\overline{HAHBP}).CALL(HDHD).ORDER(\overline{HDHD}))$
 $+ (CALL(HDHD).ORDER(\overline{HDHD}).CALL(HAHBP).ORDER(\overline{HAHBP}))).$

$CALL(SCFP1).CALL(SCFP2).CALL(SCHD).CALL(SCHBP).ORDER(\overline{ALL1}).ORDER(\overline{ALL2}).$
 $((CALL(HBHP).ORDER(\overline{HBHP}).CALL(HCHD).ORDER(\overline{HCHD}))$
 $+ (CALL(HCHD).ORDER(\overline{HCHD}).CALL(HBHP).ORDER(\overline{HBHP}))). \emptyset^\infty$

$911 = ORDER(HDHD).((AmbA(\overline{HD}).AmbA\ out.\ ORDER(ALL1).AmbB(\overline{SC}).AmbB\ out.\ ORDER(HCHD).AmbA(\overline{HC}))$
 $\oplus^1(AmbB(\overline{HD}).AmbB\ out.\ ORDER(ALL1).AmbA(\overline{SC}).AmbA\ out.\ ORDER(HCHD).AmbA(\overline{HC}))). AmbA\ in.\ AmbB\ in.\ \emptyset^\infty$

$HospitalA = ORDER(HAHBP).AmbC(\overline{HA}).AmbC\ out.\ CALL(Arrive1).HA(\overline{Ready1}).AmbC\ in.\ ORDER(ALL2).AmbC(\overline{SC}).$
 $AmbC\ out.\ ORDER(HBHP).AmbC(\overline{HB}).CALL(Arrive2).HA(\overline{Ready2}).HA(\overline{Ready3}).AmbC\ in.\ \emptyset^\infty$

$Hospital\ B = CALL(Arrive3).HB(\overline{Ready1}).((AmbA\ in.\ AmbA\ out)\oplus^1(AmbB\ in.\ AmbB\ out)).$
 $CALL(Arrive4).HB(\overline{Ready2}).((AmbB\ in.\ AmbB\ out)\oplus^1(AmbA\ in.\ AmbA\ out)).$
 $CALL(Arrive5).HB(\overline{Ready3}).((AmbA\ in.\ AmbA\ out)\oplus^1(AmbB\ in.\ AmbB\ out)).\emptyset^\infty$

$HospitalC = CALL(Arrive6).HC(\overline{Ready1}).HC(\overline{Ready2}).((AmbB\ in.\ AmbB\ out)\oplus^1(AmbA\ in.\ AmbA\ out)).\emptyset^\infty$

Specification (2): 3

$AmbA = ((AmbA(HD).out\ 911.in\ HouseD.get\ PHD2.out\ HouseD.CALL(\overline{Arrive3}).in\ HospitalB.put\ PHD2.\newline out\ HospitalB.AmbA(HC).in\ HouseC.get\ PHD1.out\ HouseC.CALL(\overline{Arrive3}).in\ HospitalB.put\ PHD1\newline out\ HospitalB) \oplus^1 (AmbA(SC).out\ 911.in\ School.get\ PFP1.get\ PFP2.get\ PHD3.out\ School.\newline ((CALL(\overline{Arrive4}).in\ HospitalB.put\ PHD3.out\ HospitalB.CALL(\overline{Arrive6}).in\ HospitalC.put\ PFP1.put\ PFP2.\newline out\ HospitalC) + (CALL(\overline{Arrive6}).in\ HospitalC.put\ PFP1.put\ PFP2.out\ HospitalC.CALL(\overline{Arrive4}).\newline in\ HospitalB.put\ PHD3.out\ HospitalB))).in\ 911.\emptyset^\infty)$

$AmbB = ((AmbB(SC).out\ 911.in\ School.get\ PFP1.get\ PFP2.get\ PHD3.out\ School.((CALL(\overline{Arrive4}).in\ HospitalB.\newline put\ PHD3.out\ HospitalB.CALL(\overline{Arrive6}).in\ HospitalC.put\ PFP1.put\ PFP2.out\ HospitalC) + (CALL(\overline{Arrive6}).\newline in\ HospitalC.put\ PFP1.put\ PFP2.out\ HospitalC.CALL(\overline{Arrive4}).in\ HospitalB.put\ PHD3.out\ HospitalB)))\newline \oplus^1 (AmbB(HD).out\ 911.in\ HouseD.get\ PHD2.out\ HouseD.CALL(\overline{Arrive3}).in\ HospitalB.put\ PHD2.\newline out\ HospitalB.AmbB(HC).in\ HouseC.get\ PHD1.out\ HouseC.CALL(\overline{Arrive3}).in\ HospitalB.put\ PHD1\newline out\ HospitalB)).in\ 911.\emptyset^\infty)$

$AmbC = AmbC(HA).out\ HospitalA.in\ HouseA.get\ PHBP1.out\ HouseA.CALL(\overline{Arrive1}).in\ HospitalA.put\ PHBP1.\newline AmbC(SC).out\ HospitalA.in\ School.get\ PHBP3.out\ School.AmbC(HB).in\ HouseB.get\ PHBP2.out\ HouseB.\newline CALL(\overline{Arrive2}).in\ HospitalA.put\ PHBP2.put\ PHBP3.\emptyset^\infty$

Specification (3): 16

$HouseA = AmbC \text{ in. } AmbC \text{ out. } \emptyset^\infty$

$HouseB = AmbC \text{ in. } AmbC \text{ out. } \emptyset^\infty$

$HouseC = ((AmbA \text{ in. } AmbA \text{ out}) \oplus^1 (AmbB \text{ in. } AmbB \text{ out})). \emptyset^\infty$

$HouseD = ((AmbA \text{ in. } AmbA \text{ out}) \oplus^1 (AmbB \text{ in. } AmbB \text{ out})). \emptyset^\infty$

$School = ((AmbB \text{ in. } AmbB \text{ out}) \oplus^1 (AmbA \text{ in. } AmbA \text{ out})). AmbC \text{ in. } AmbC \text{ out. } \emptyset^\infty$

$PHBP1 = CALL(\overline{HAHBP}). AmbC \text{ get. } AmbC \text{ put. } SurgeryA \text{ get. } DoctorA1(Surgery). \emptyset^\infty$

$PHBP2 = CALL(\overline{HBHBP}). AmbC \text{ get. } AmbC \text{ put. } SurgeryA \text{ get. } DoctorA2(Surgery). \emptyset^\infty$

$PHBP3 = CALL(\overline{SCHBP}). AmbC \text{ get. } AmbC \text{ put. } SurgeryA \text{ get. } DoctorA3(Surgery). \emptyset^\infty$

$PHD1 = CALL(\overline{HCHD}). ((AmbA \text{ get. } AmbA \text{ put}) \oplus^1 (AmbB \text{ get. } AmbB \text{ put})). SurgeryB \text{ get. } DoctorB1(Surgery). \emptyset^\infty$

$PHD2 = CALL(\overline{HDHD}). ((AmbA \text{ get. } AmbA \text{ put}) \oplus^1 (AmbB \text{ get. } AmbB \text{ put})). SurgeryB \text{ get. } DoctorB2(Surgery). \emptyset^\infty$

$PHD3 = CALL(\overline{SCHD}). ((AmbB \text{ get. } AmbB \text{ put}) \oplus^1 (AmbA \text{ get. } AmbA \text{ put})). SurgeryB \text{ get. } DoctorB3(Surgery). \emptyset^\infty$

$PFP1 = CALL(\overline{SCHD}). ((AmbB \text{ get. } AmbB \text{ put}) \oplus^1 (AmbA \text{ get. } AmbA \text{ put})). SurgeryC \text{ get. } DoctorC1(Surgery). \emptyset^\infty$

$PFP2 = CALL(\overline{SCHD}). ((AmbB \text{ get. } AmbB \text{ put}) \oplus^1 (AmbA \text{ get. } AmbA \text{ put})). SurgeryC \text{ get. } DoctorC2(Surgery). \emptyset^\infty$

$SurgeryA = DoctorA1 \text{ in. } PHBP1 \text{ get. } DoctorA2 \text{ in. } PHBP2 \text{ get. } DoctorA3 \text{ in. } PHBP3 \text{ get. } \emptyset^\infty$

$SurgeryB = DoctorB1 \text{ in. } PHD1 \text{ get. } DoctorB2 \text{ in. } PHD2 \text{ get. } DoctorB3 \text{ in. } PHD3 \text{ get. } \emptyset^\infty$

$SurgeryC = DoctorC1 \text{ in. } PFP1 \text{ get. } DoctorC2 \text{ in. } PFP2 \text{ get. } \emptyset^\infty$

Specification (4): 11

DoctorA1 = HA(Ready1).out RestroomA.in SurgeryA.DoctorA1(Surgery).Ø[∞]

DoctorA2 = HA(Ready2).out RestroomA.in SurgeryA.DoctorA2(Surgery).Ø[∞]

DoctorA3 = HA(Ready3).out RestroomA.in SurgeryA.DoctorA3(Surgery).Ø[∞]

DoctorB1 = HB(Ready1).out RestroomB.in SurgeryB.DoctorB1(Surgery).Ø[∞]

DoctorB2 = HB(Ready2).out RestroomB.in SurgeryB.DoctorB2(Surgery).Ø[∞]

DoctorB3 = HB(Ready3).out RestroomB.in SurgeryB.DoctorB3(Surgery).Ø[∞]

DoctorC1 = HC(Ready1).out RestroomC.in SurgeryB.DoctorC1(Surgery).Ø[∞]

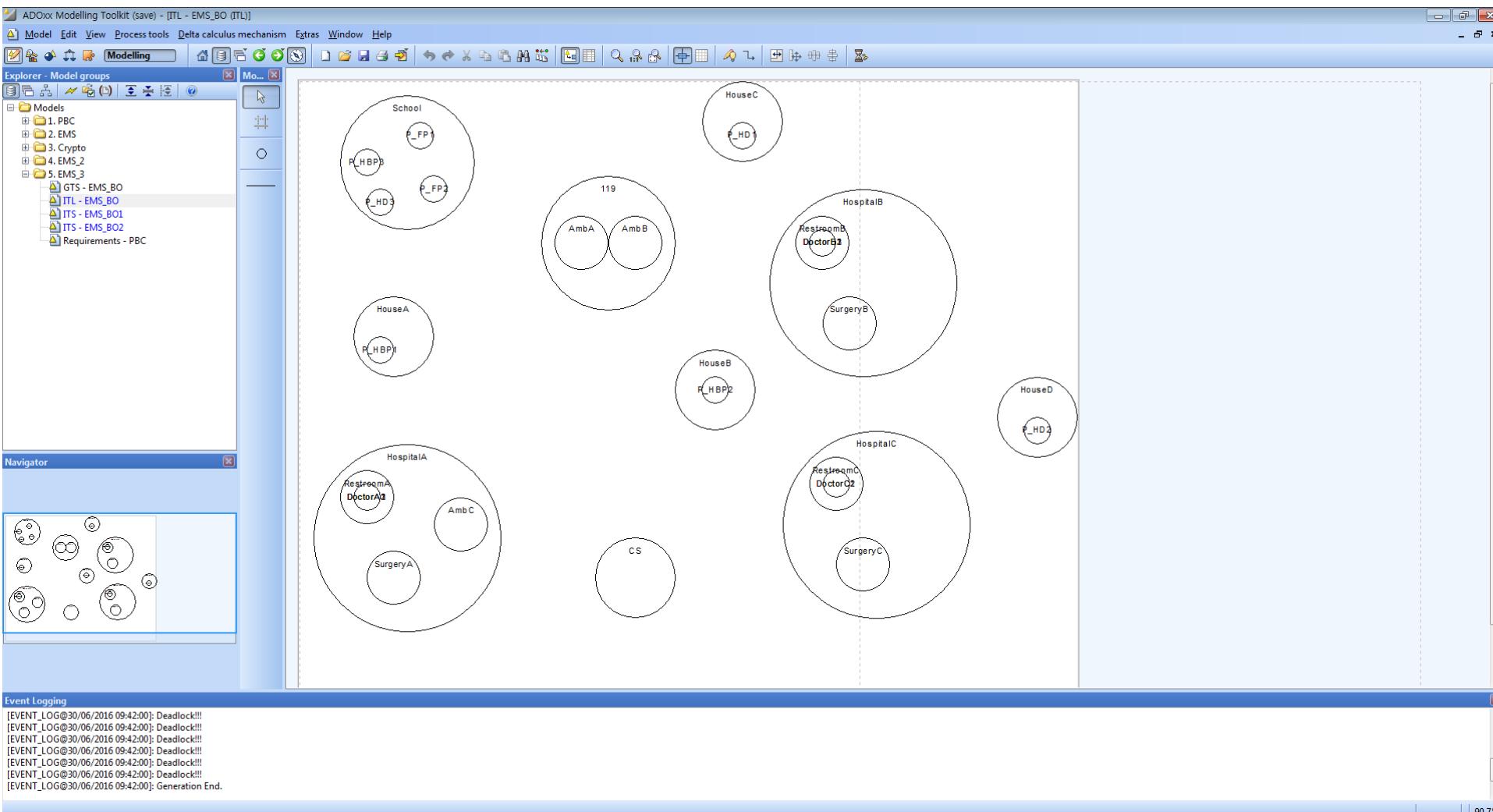
DoctorC2 = HC(Ready2).out RestroomC.in SurgeryB.DoctorC2(Surgery).Ø[∞]

RestroomA = DoctorA1 out. DoctorA2 out. DoctorA3 out. Ø[∞]

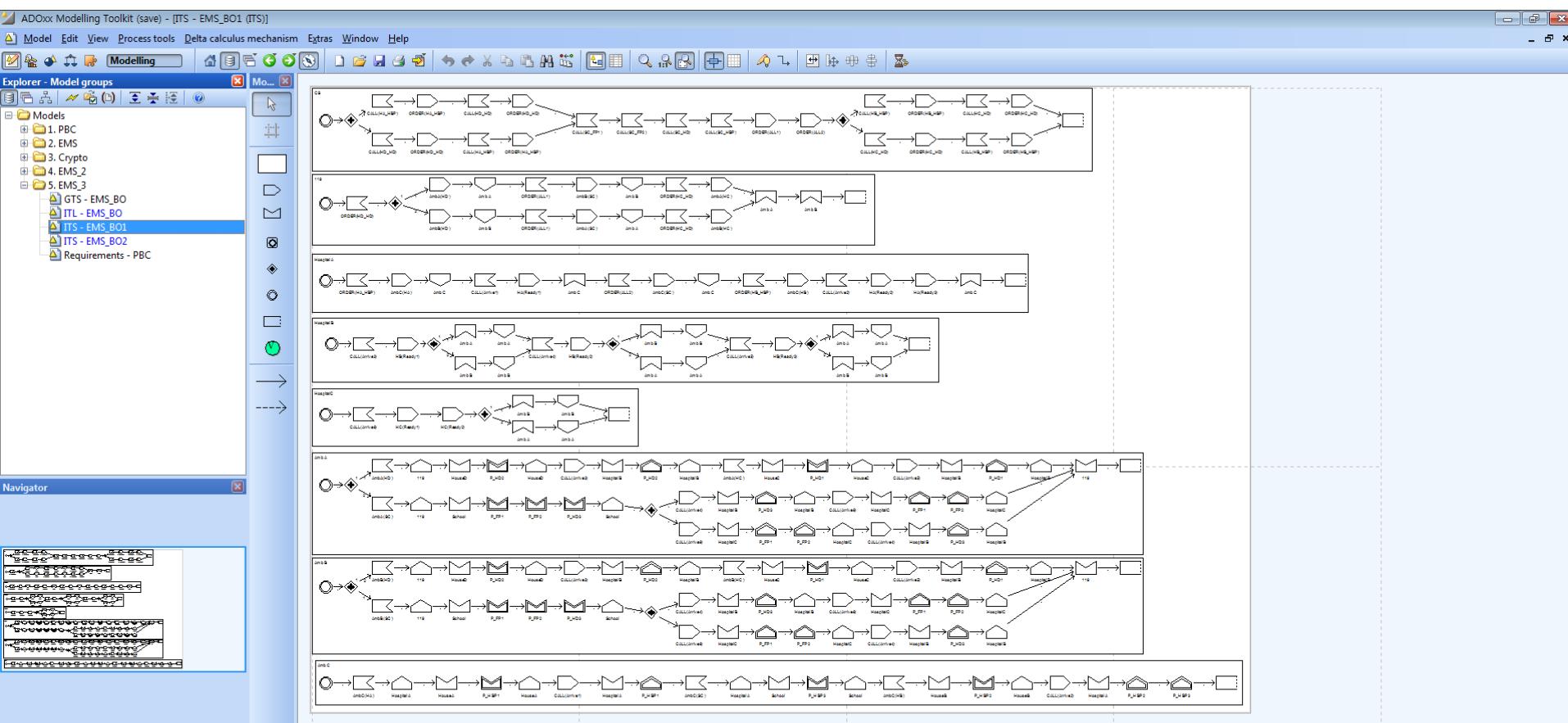
RestroomB = DoctorB1 out. DoctorB2 out. DoctorB3 out. Ø[∞]

RestroomC = DoctorC1 out. DoctorC2 out. Ø[∞]

ITL



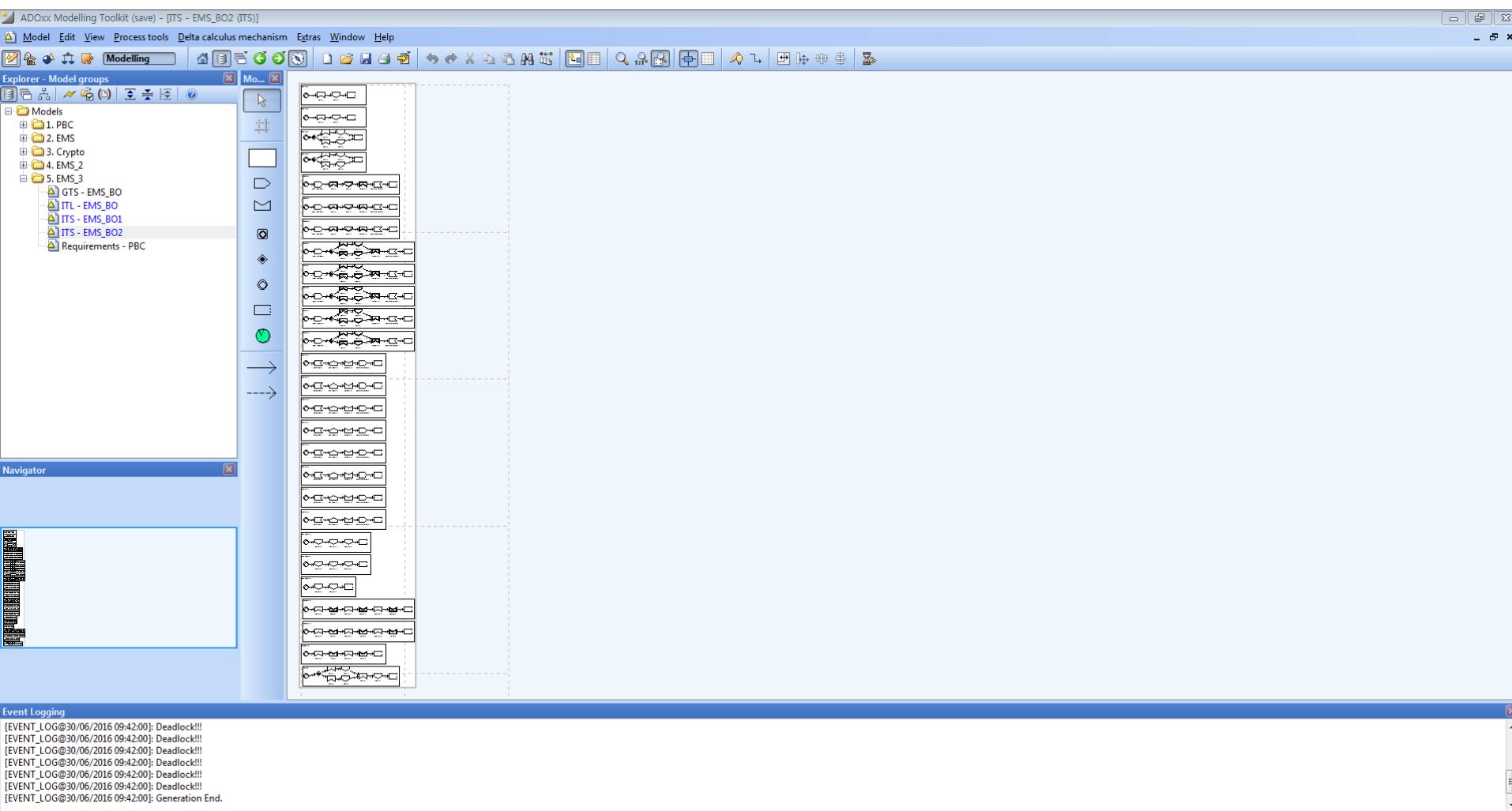
ITS (1)



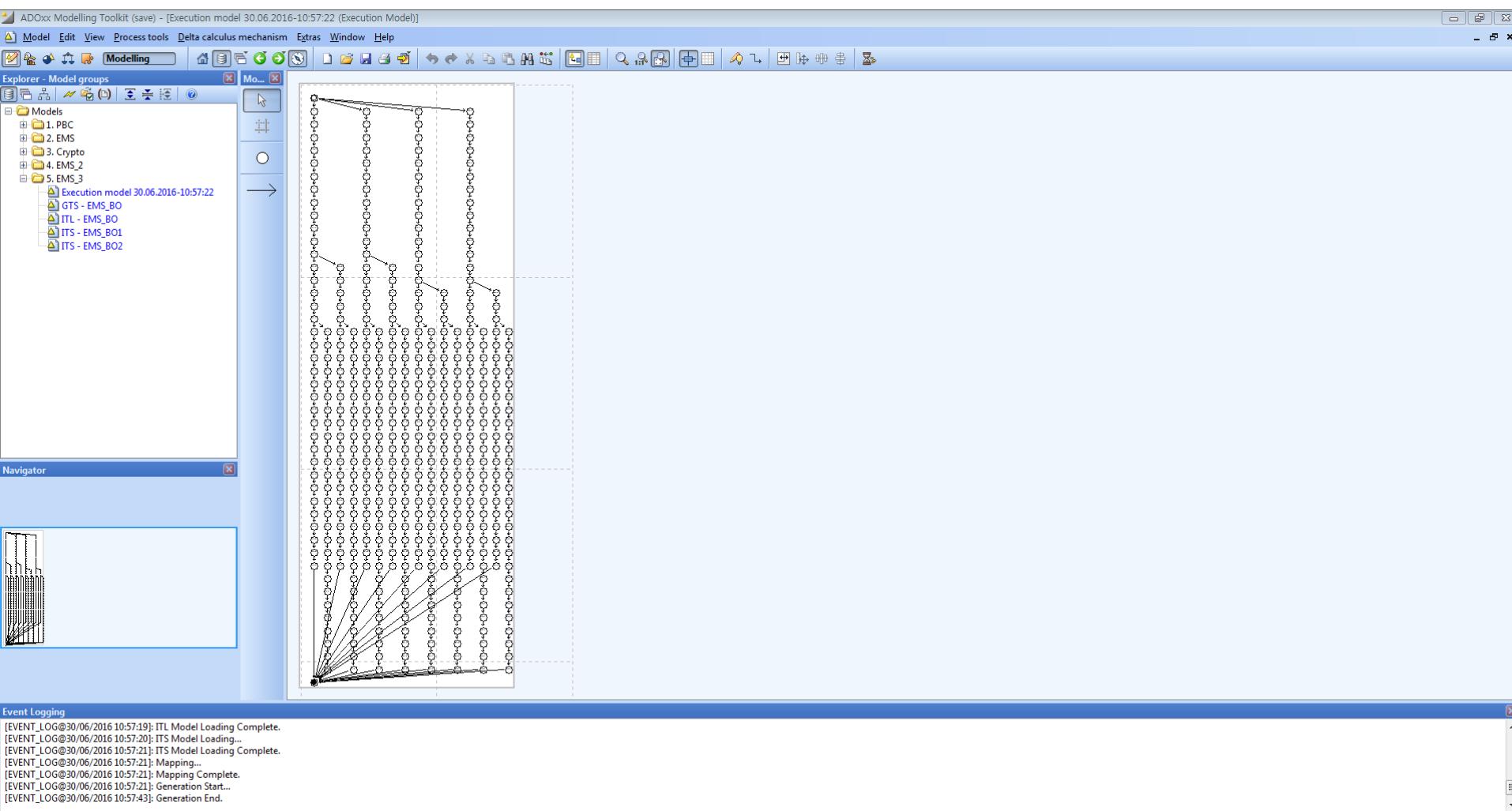
Event Logging

```
[EVENT_LOG@30/06/2016 09:42:00]: Deadlock!!!
[EVENT_LOG@30/06/2016 09:42:00]: Generation End.
```

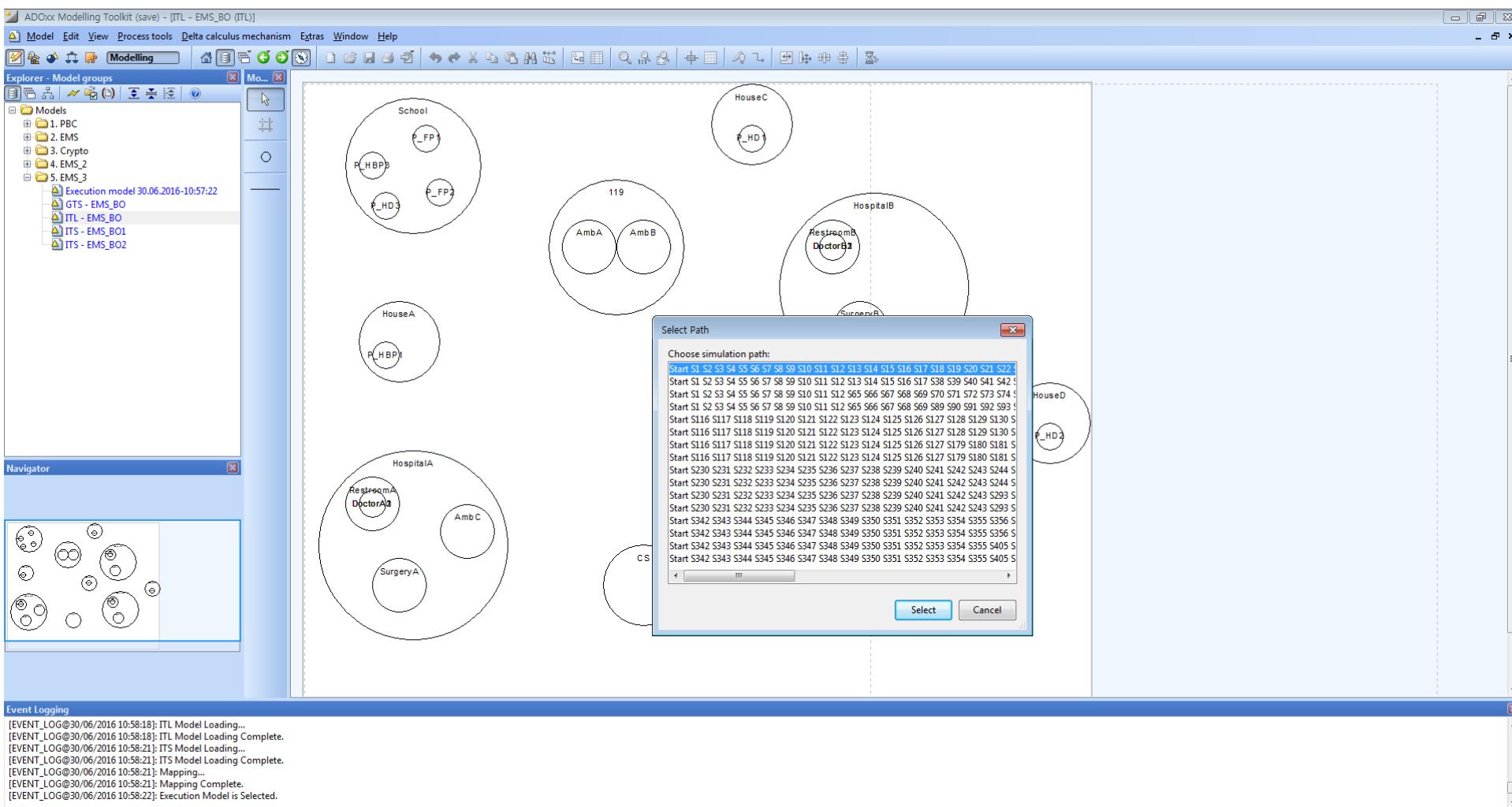
ITS (2)



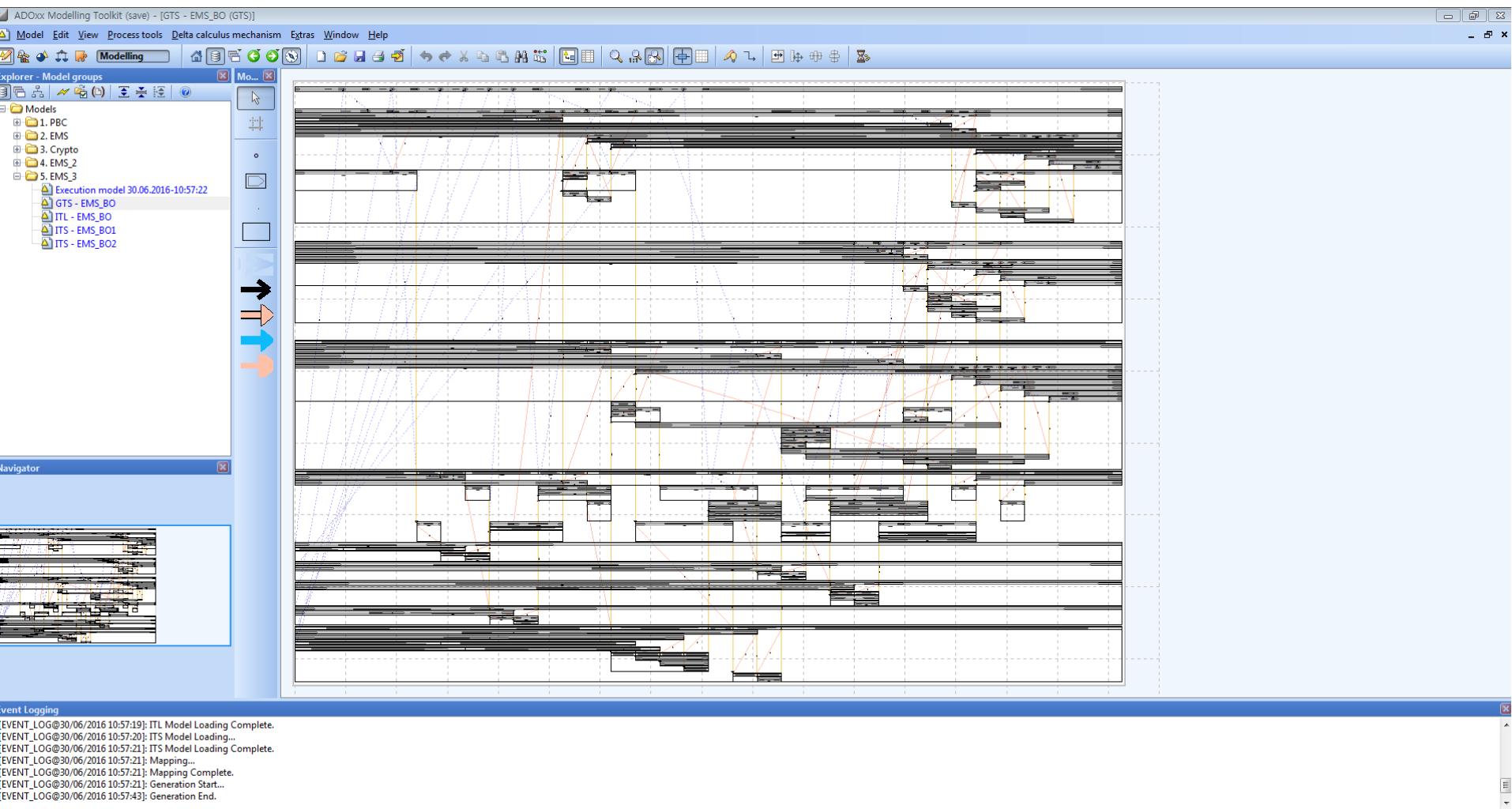
Execution Model



Simulation



GTS



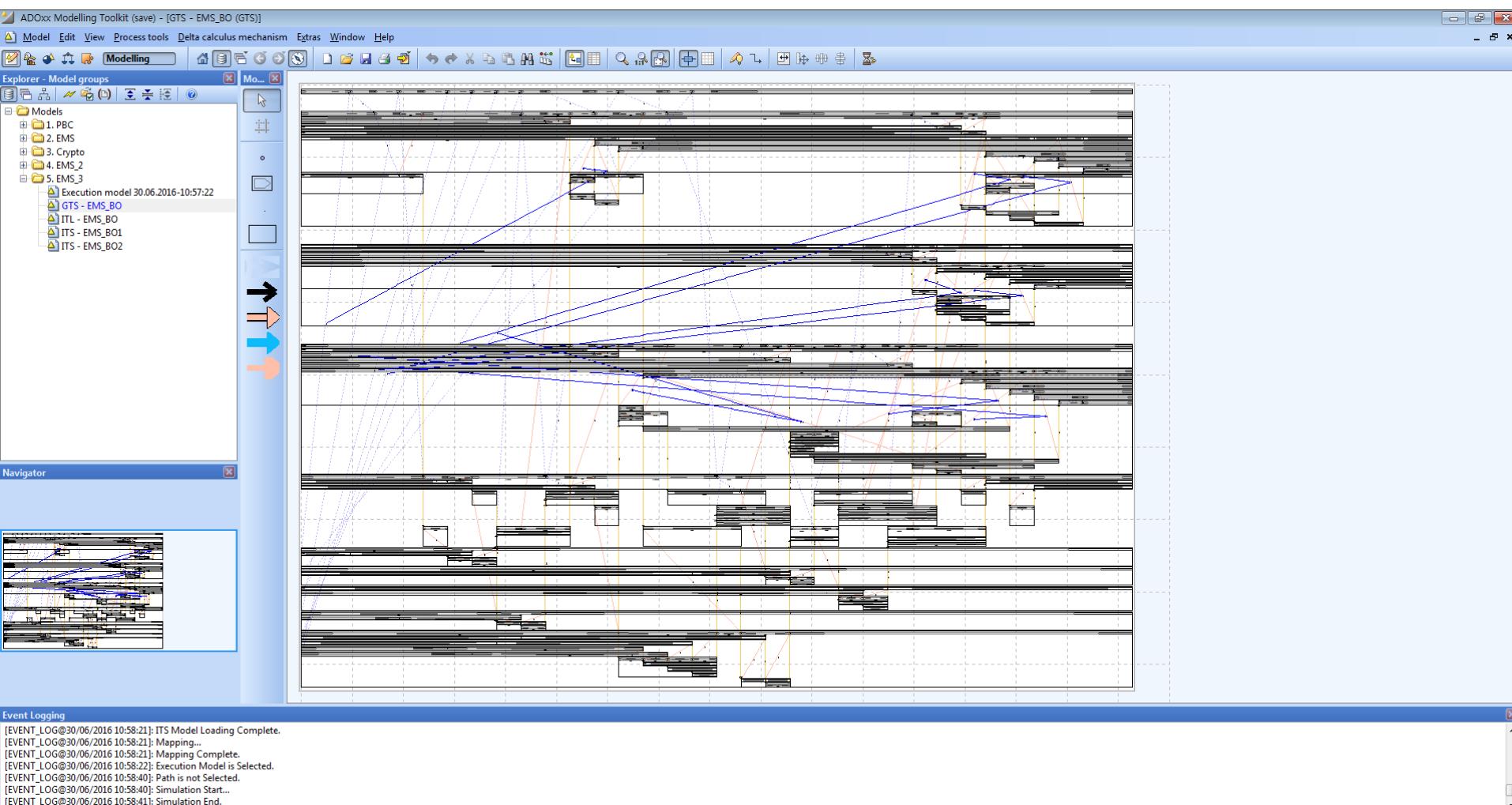
Secure Requirements (1)

- Patient HBP should be transported to Hospital A:
 - R1: $(P_HBP1 : CALL(\overline{HA_HBP}) \rightarrow (SurgeryA: get P_HBP1))$
 - R2: $(P_HBP2 : CALL(\overline{HB_HBP}) \rightarrow (SurgeryA: get P_HBP2))$
 - R3: $(P_HBP3 : CALL(\overline{SC_HBP}) \rightarrow (SurgeryA: get P_HBP3))$
- Patient HD should be transported to Hospital B:
 - R4: $(P_HD1 : CALL(\overline{HC_HD}) \rightarrow (SurgeryB: get P_HD1))$
 - R5: $(P_HD2 : CALL(\overline{HD_HD}) \rightarrow (SurgeryB: get P_HD2))$
 - R6: $(P_HD3 : CALL(\overline{SC_HD}) \rightarrow (SurgeryB: get P_HD3))$
- Patient FP should be transported to Hospital C:
 - R7: $(P_FP1 : CALL(\overline{SC_FP1}) \rightarrow (SurgeryC: get P_FP1))$
 - R8: $(P_FP2 : CALL(\overline{SC_FP2}) \rightarrow (SurgeryC: get P_FP2))$

Secure Requirements (2)

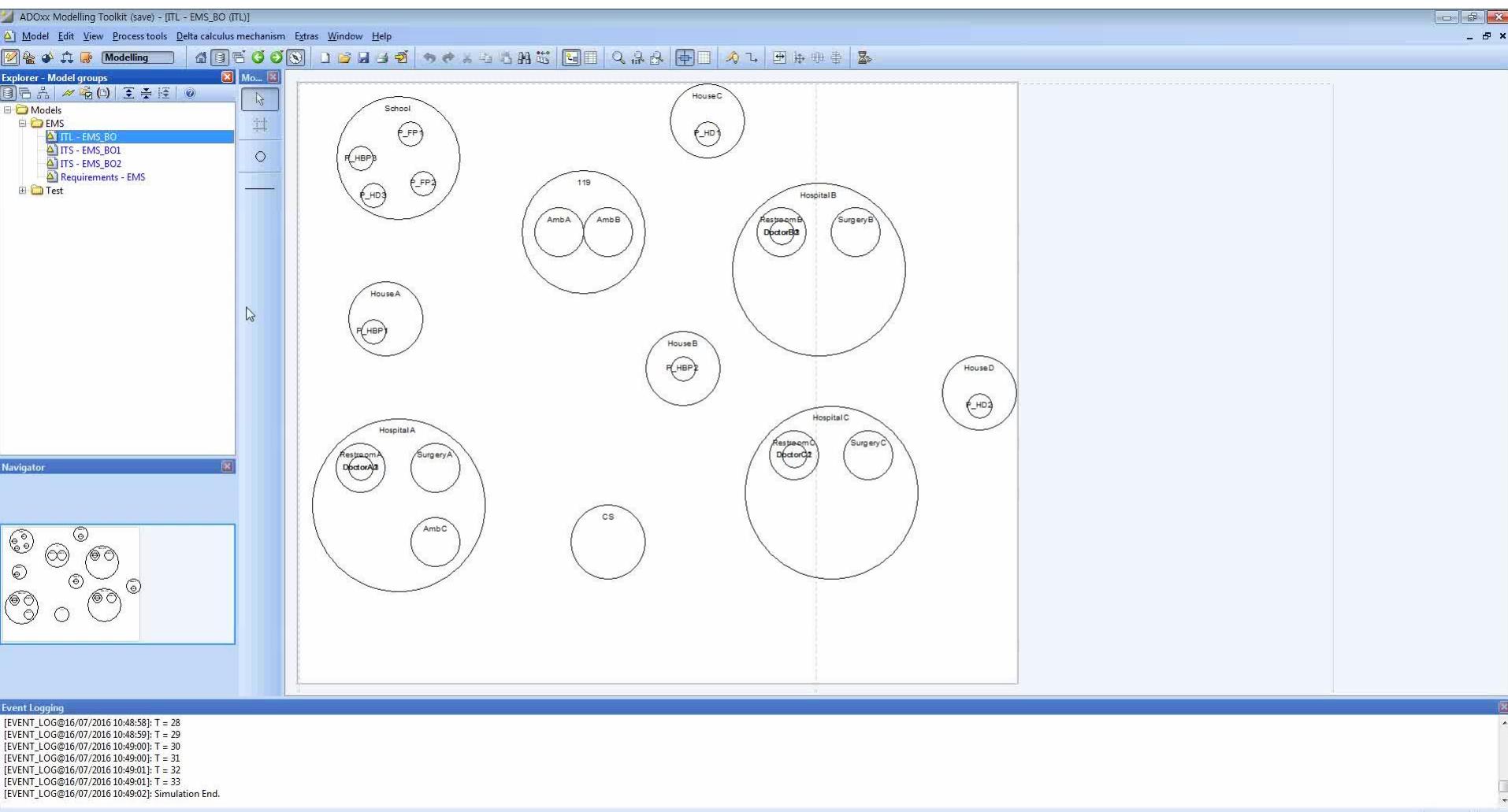
- Doctor should be ready before Patient's arrival for treatment:
 - R9: $(DoctorA1: in SurgeryA) < (SurgeryA: get P_HBP1)$
 - R10: $(DoctorA2: in SurgeryA) < (SurgeryA: get P_HBP2)$
 - R11: $(DoctorA3: in SurgeryA) < (SurgeryA: get P_HBP3)$
 - R12: $(DoctorB1: in SurgeryB) < (SurgeryA: get P_HD1)$
 - R13: $(DoctorB2: in SurgeryB) < (SurgeryA: get P_HD2)$
 - R14: $(DoctorB3: in SurgeryB) < (SurgeryA: get P_HD3)$
 - R15: $(DoctorC1: in SurgeryC) < (SurgeryA: get P_FP1)$
 - R16: $(DoctorC2: in SurgeryC) < (SurgeryA: get P_FP2)$

GTS Logic



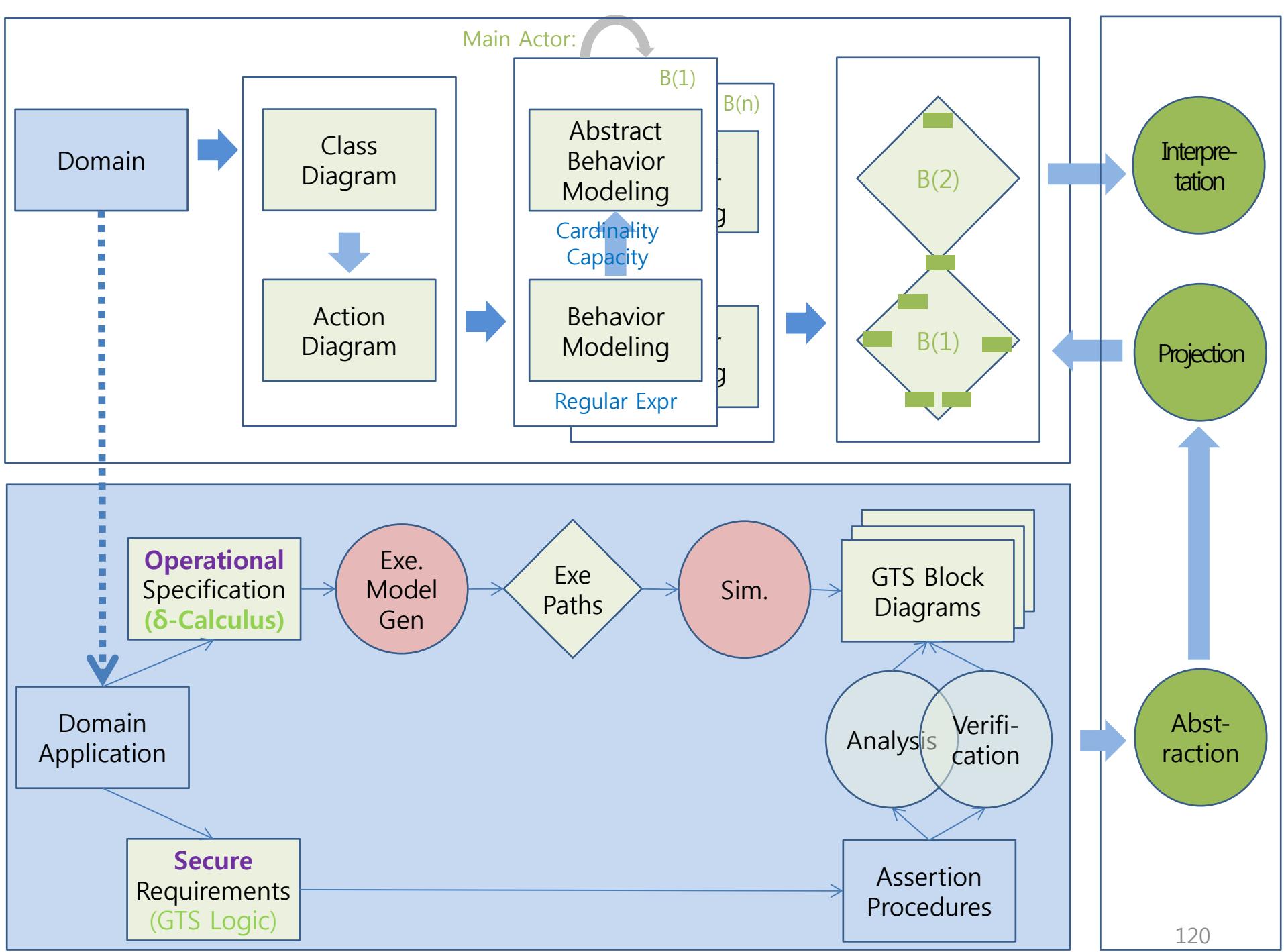
SAVE

DEMO: SAVE EMS

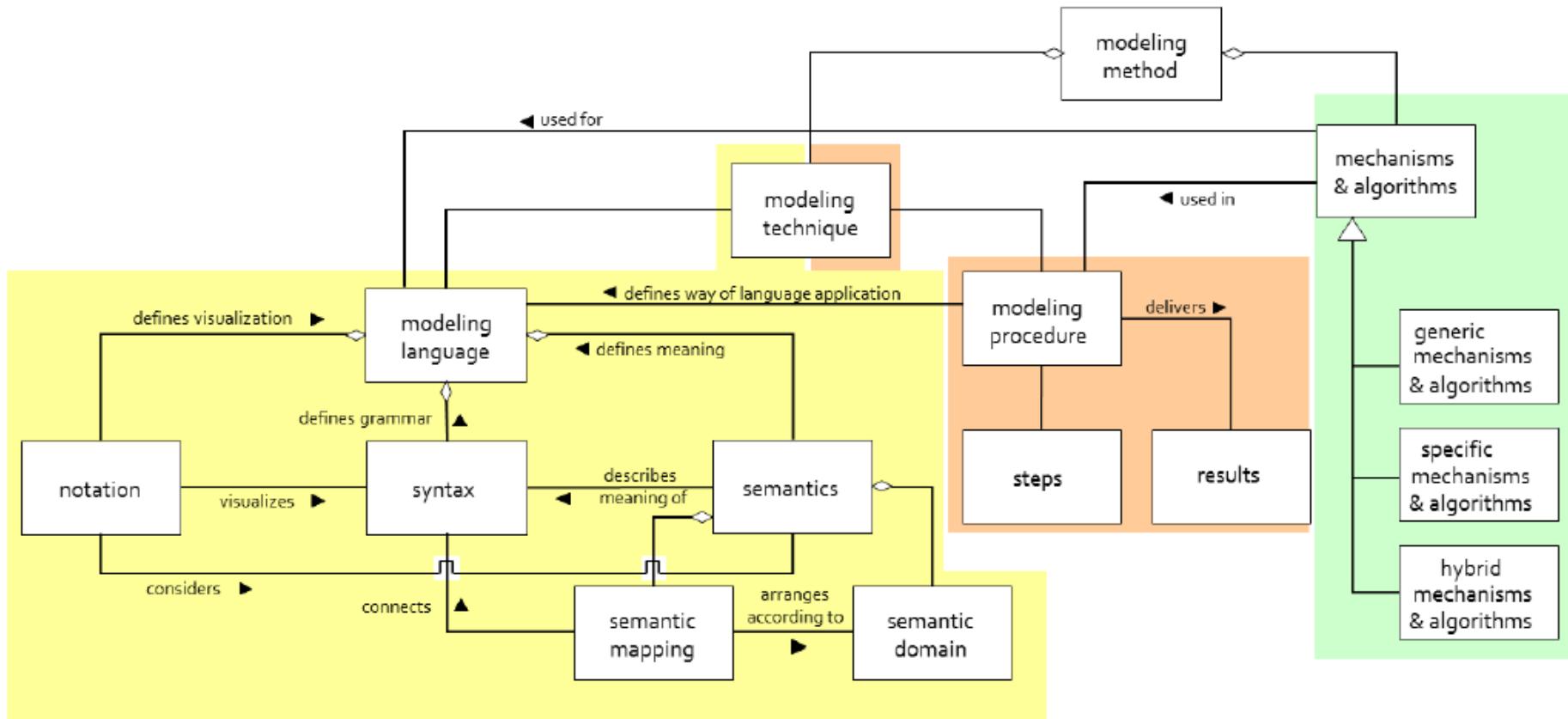


1. Motivation
2. Modeling
 - 1) Specification: δ -Calculus
 - 2) Verification: GTS Logic
 - 3) EMS Example
- 3. SAVE Tool**
4. Cyber-Physical Systems Application
5. Summary
6. Discussion

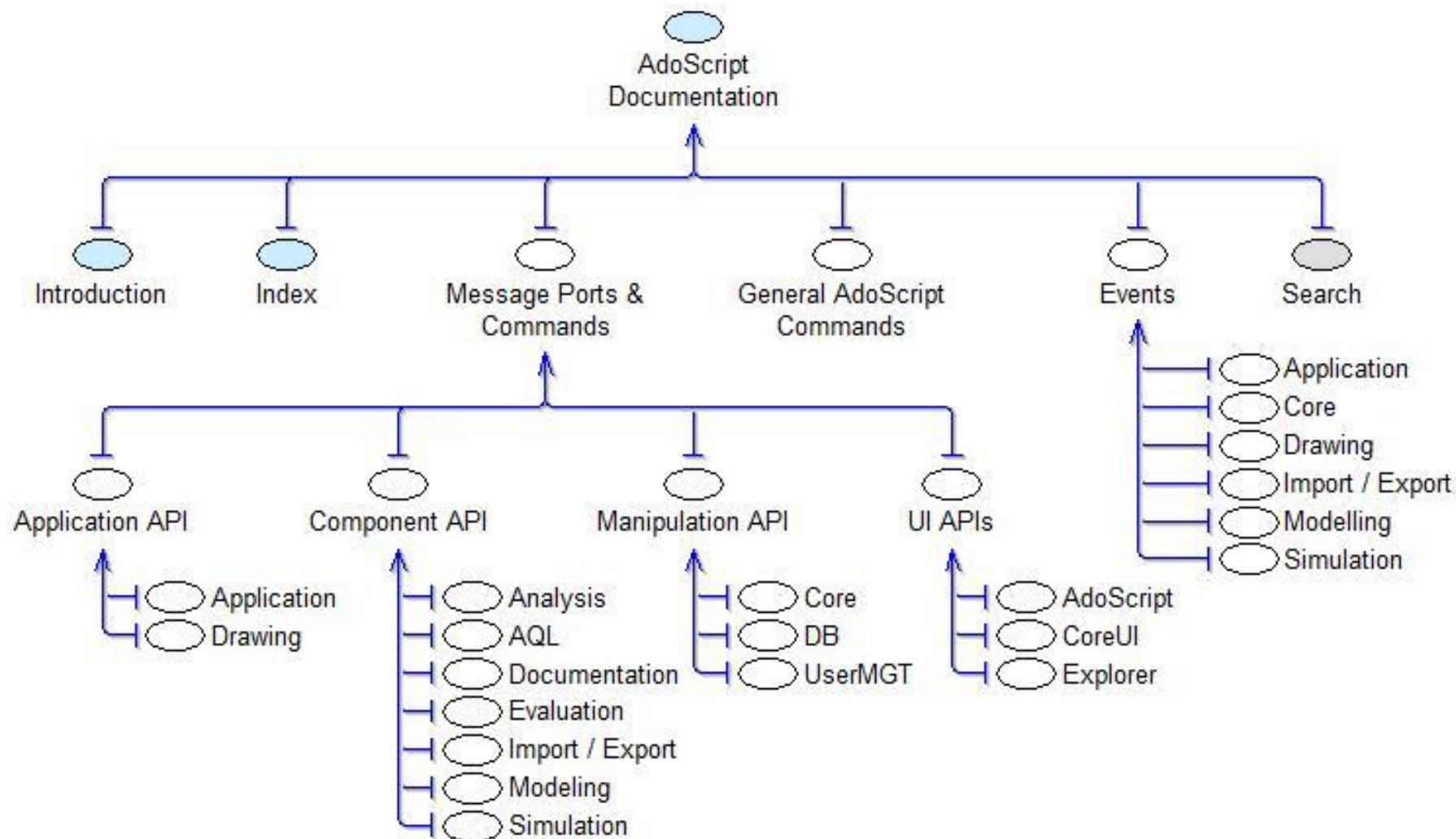
3. **SAVE TOOLS**



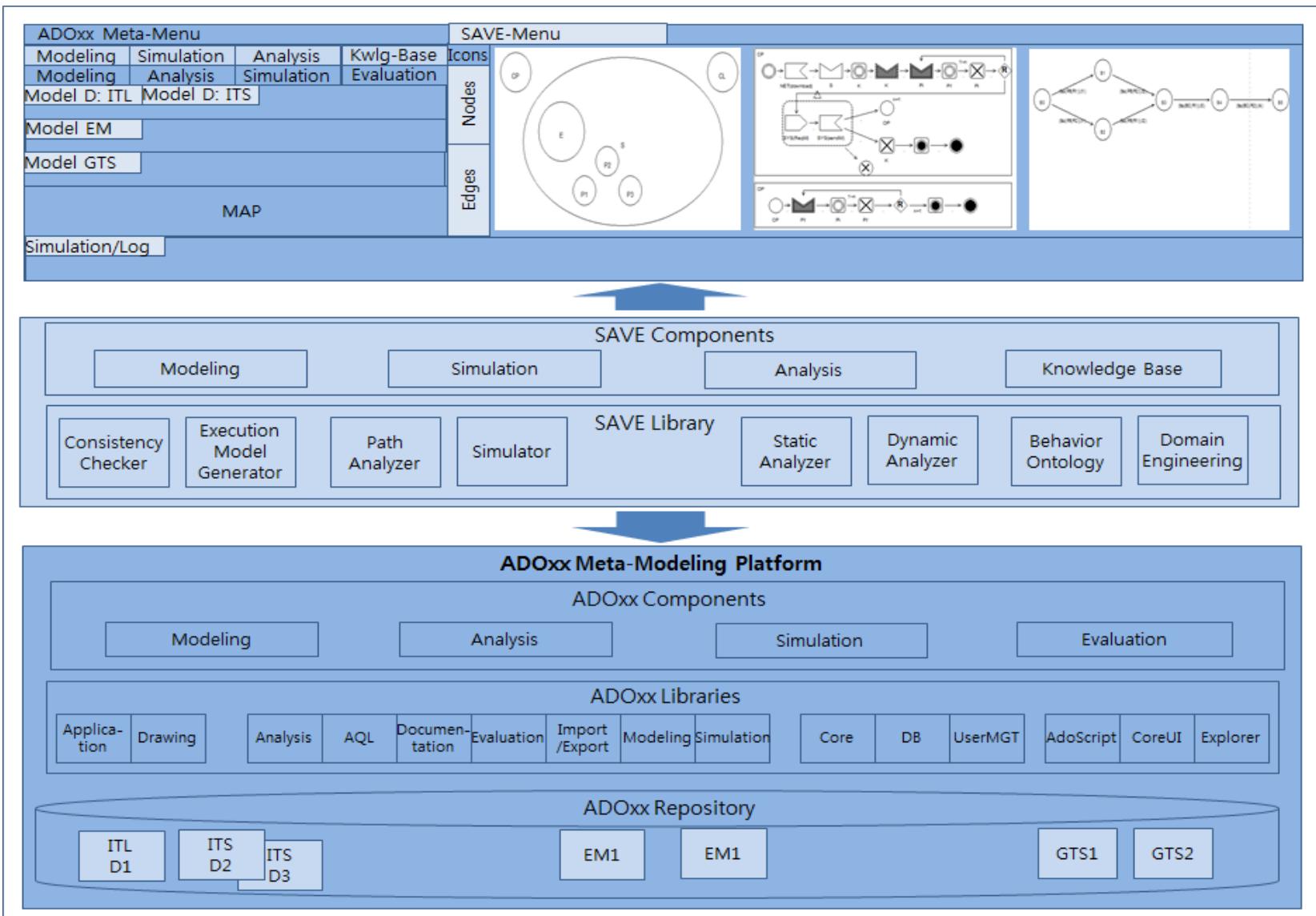
Generic Modelling Method Framework



ADOxx Library



Platform



Execution Model Generator

```

CC "CoreUI" MODEL_SELECT_BOX modeltype:(exModel) title:(“Select Execution Model”) boxtext:(“Please select execution model.”)
IF (endbutton = “ok”)
{
    SET sExModelIDs:(modelids)
    SET exModelID:(VAL sExModelIDs)
    CC “Core” LOAD_MODEL modelid:(exModelID)
    EVENT_LOG msgType:“EVENT_LOG” message: (“Execution Model is Selected.”)

    CC “Core” GET_CLASS_ID classname:“State_EX”
    SET nStateClassID:(classid)
    CC “Core” GET_OBJ_ID modelid:(exModelID) classid:(nStateClassID) objid
    SET startState:(objid)
    SET totalPath:(0)
    SETL pathString:(“”)
    SETL totalPathString:(“”)

    FIND_PATH start:(STR startState) path:(pathString)
    CC “AdoScript” LISTBOX entries:(totalPathString) toksep:”;” title:“Se
    IF (endbutton = “ok”)
    {
        SET select:(selection)
        #CC “AdoScript” VIEWBOX text:(select)
        EVENT_LOG msgType:“EVENT_LOG” message: (“Path is Selected.”)
    }
    ELSE
    {
        EVENT_LOG msgType:“EVENT_LOG” message: (“Path is not Selected.”)
        SET exModelID:(0)
    }
}
ELSE
{
    EVENT_LOG msgType:“EVENT_LOG” message: (“Execution Model is not Selected.”)
    SET exModelID:(0)

#Simulation Start
EVENT_LOG msgType:“EVENT_LOG” message: (“Simulation Start...”)

#Simulating...
SETG statecount:(0)
SETG pathcount:(0)
SETG choicemap:(map())
SETG choicemap[“all_list”]:(“”)
FOR pname in:(loadITS[“list”])
{
    SETG choicemap[loadITS[pname+“_name”]+“_list”]:(“”)
SIM_INIT mode:(“sim”) lcurrent:lcurrent
SETL current:(lcurrent)
SETL tempsync:(map())
SETL tempsync[“list”]:(“”)
SIM current:(current) mode:(“sim”) globaltime:(0) sync:(STR tempsync) syn
EVENT_LOG msgType:“EVENT_LOG” message: (“Simulation End.”)

}
ELSIF (acttype = “In”)
{
    SETL state:(loadITS[actid+“_state”])
    SETL ctime:(loadITS[actid+“_time”])
    SETL gtime:(loadITS[actid+“_gtime”])
    SETL dtime:(loadITS[actid+“_deadline”])
    #SETL ctime:(ctime + 1)
    #SETL gtime:(gtime + 1)

    #state and time check
    IF (state = “Waiting”)
    {
        SETL type:(loadITS[actid+“_attr1”])
        SETL process:(loadITS[actid+“_attr2”])
        SETL myprocess:(loadITS[actid+“_process”])

        IF (type = “Request”)
        {
            CHECK_MOVE current:(currentlist) move:(“In”) process:(process) myprocess:(myprocess) find:(“Permission”) moveresult:result
            SETL fidnid:(result)

            IF (fidnid != 0)
            {
                SETL change[“list”]:(tokunion(change[“list”], actid))
                SETL change[actid+“_mode”]:(“Executing”)
                SETG loadITS[actid+“_time”]:(-1)
                SETL synclist:(tokunion(synclist, actid))
            }
        }
        ELSE
        {
            CHECK_MOVE current:(currentlist) move:(“In”) process:(process) myprocess:(myprocess) find:(“Request”) moveresult:result
            SETL fidnid:(result)

            IF (fidnid != 0)
            {
                SETL change[“list”]:(tokunion(change[“list”], actid))
                SETL change[actid+“_mode”]:(“Executing”)
                SETG loadITS[actid+“_time”]:(-1)
                SETL synclist:(tokunion(synclist, actid))
            }
        }
    }
}
ELSIF (acttype = “Out”)
{
    SETL state:(loadITS[actid+“_state”])
    SETL ctime:(loadITS[actid+“_time”])
    SETL gtime:(loadITS[actid+“_gtime”])
    SETL dtime:(loadITS[actid+“_deadline”])
    #SETL ctime:(ctime + 1)
}

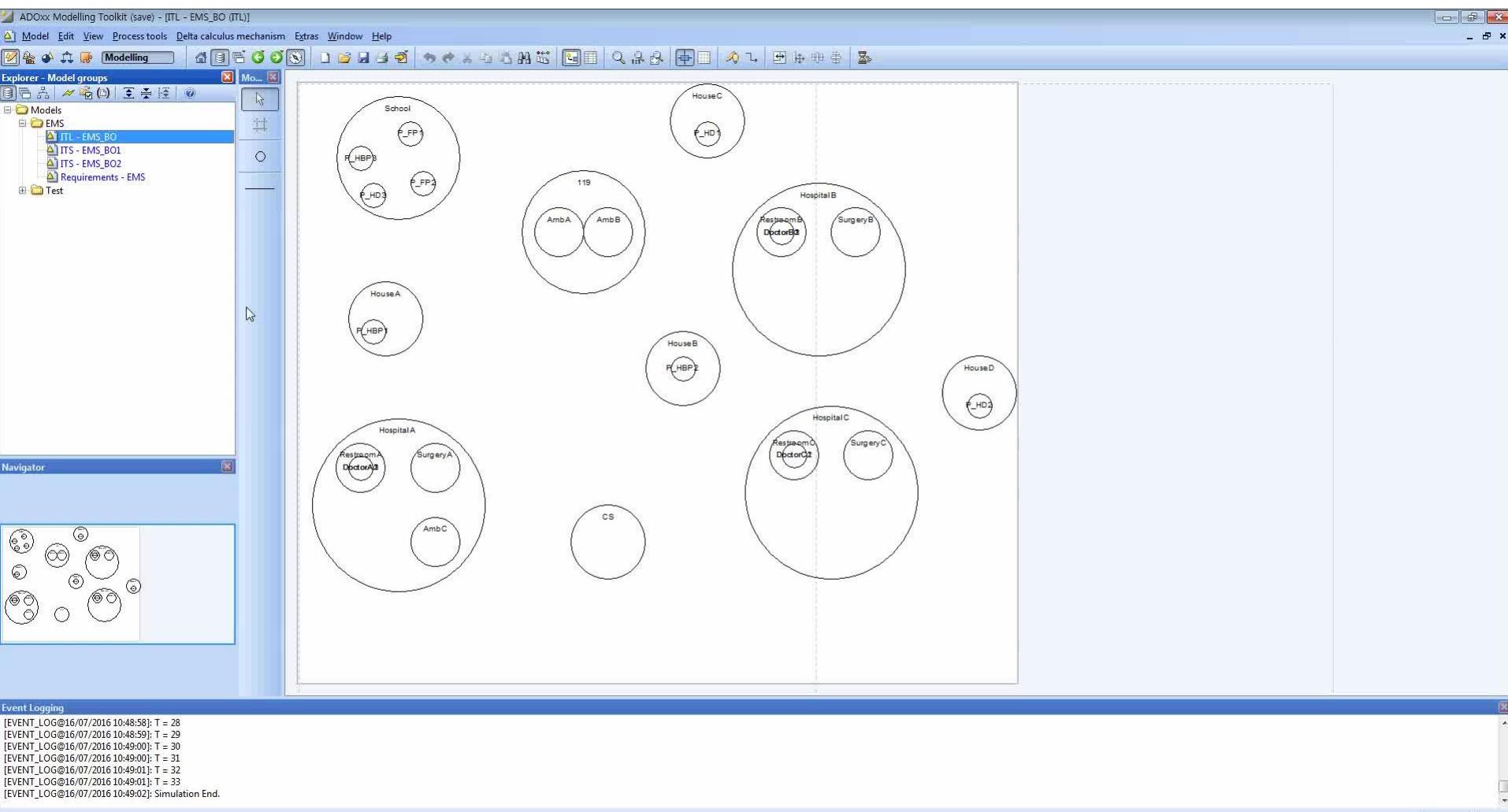
```

GTS Analyzer

```
1123 FOR i from:1 to:(sEndOfStateNum)
1124 {
1125     SETL state_num:(STR i)
1126     SETL t:(0)
1127     IF (nDebug = 1) {SET data10:(data10+state_num+": \t sEndOfStateNum \n")}
1128     #7.5
1129     FOR process in:(mD_Data[state_num+"_"+list])
1130     {
1131         IF (type(mD_Data[state_num+"_"+process+"_"+interactiontype]) != "undefined")
1132         {
1133             IF (nDebug = 1) {SET data14:(data14+state_num+": "+process+": "+"STATE: type:\t"+mD_Data[state_num+"_"+process+"_"+interactiontype])}
1134         }
1135         IF (type(mD_Data[state_num+"_"+process+"_"+interactionnum]) != "undefined")
1136         {
1137             IF (nDebug = 1) {SET data14:(data14+state_num+": "+process+": "+"STATE: num:\t"+mD_Data[state_num+"_"+process+"_"+interactionnum])}
1138         }
1139         IF (type(mD_Data[state_num+"_"+process+"_"+interactiontype]) != "undefined" AND type(mD_Data[process+"_"+a+b+"_AttrOrAte"]) != "undefined")
1140         {
1141             IF (type(mD_Data[process+"_"+a+b+"_AttrOrAte"]) != "undefined")
1142             {
1143                 IF (nDebug = 1) {SET data14:(data14+state_num+": "+process+": "+"STATE: AttrOrAte:\t"+AttrOrAte)}
1144             }
1145             # SET mD_Data[state_num+"_"+acteeprocess+"_"+interactiontype]:(sInterActionType) #내려온다
1146             # SET mD_Data[state_num+"_"+acteeprocess+"_"+interactionnum]:(sInterActionNum) #내려온다
1147             # IF (type(mD_Data[state_num+"_"+acteeprocess+"_"+interactiontype])!="undefined")
1148             # {
1149                 # IF (nDebug = 1) {SET data14:(data14+state_num+": "+process+": "+"STATE: AttrOrAte:\t"+AttrOrAte)}
1150             # }
1151         }
1152         IF (nDebug = 1) {SET data14:(data14+EndOfState"\n")}
1153     }
1154 }
1155 IF (nDebug = 1) {CC "AdoScript" FWRITE file:"C:/Users/default/Desktop/AH94BHT/Desktop/AD0xx_Debug_Text.dat"
1156
1157 SET data15:@""}
1158 FOR i from:1 to:(mSingleActionData["num"])
1159 {
1160     IF (type(mSingleActionData[STR i+"_AttrOrAte"]) != "undefined")
1161     {
1162         SETL t:(mSingleActionData[STR i+"_process"])
1163         SETL a:(mSingleActionData[STR i+"_interactiontype"])
1164         SETL b:(mSingleActionData[STR i+"_interactionnum"])
1165         IF (nDebug = 1) {SET data15:(data15+STR i+": "+t+": "+"STATE: AttrOrAte:\t"+t type: "+a+
1166         )
1167     }
1168     IF (nDebug = 1) {CC "AdoScript" FWRITE file:"C:/Users/default/Desktop/AH94BHT/Desktop/AD0xx_Debug_Text.dat"
1169
1170     }
1171
1172 FOR process in:(mAll_OF_Process["list"])
1173 {
1174     SETL prev_blockaddress:(" ")
1175     IF (type(mD_Data[process+"_InterActionList"]) = "undefined")
1176     {
1177         IF (nDebug = 1) {SET data07:(data07+GET_ERROR: " ")}
1178         IF (nDebug = 1) {SET data07:(data07+process+"\n")}
1179     }
1180     ELSE
1181     {
1182         IF (nDebug = 1) {SET data07:(data07+NO_ERROR: " ")}
```

```
751 IF (nDebug = 1) {CC "AdoScript" FWRITE file:"C:/Users/default/Desktop/AH94BHT/Desktop/AD0xx_Debug_Text.dat"
752 IF (nDebug = 1) {CC "AdoScript" FWRITE file:"C:/Users/default/Desktop/AH94BHT/Desktop/AD0xx_Debug_Text/int
753
754
755
756 IF (nDebug = 1) {CC "AdoScript" FWRITE file:"C:/Users/default/Desktop/AH94BHT/Desktop/AD0xx_Debug_Text.dat
757 #####
758 #####
759 # 7. 모든 프로세스 Obj의 Y축 크기 구하기
760 # 1) 모든 상태의 프로세스 포함관계 모두 구함. 모든 상태의 포함관계(block address)를 구함
761 # 2) 프로세스 포함관계 결과를 통해 프로세스의 Y축 크기 계산함
762 # 3) 프로세스 및 맥락의 X축 크기를 계산함. 동시에 Y축 크기도 계산함
763 # 4) 인터랙션의 x1 y1 x2 y2 구하기 (데스팅 후 진행)
764 # 5) 프로세스 포함관계 결과를 통해 프로세스 역선불록의 X축 크기 계산함
765 # 6) 프로세스 포함관계 결과를 통해 상급 역선불록의 X축 크기 계산함
766 # nlist: 모든 프로세스 정보
767 # slist: 수행된 프로세스 정보
768 #####
769
770 SET data05:(" ")
771 SET data06:(" ")
772 SET data07:(" ")
773
774 SET sAll_OF_Children:(map())
775 SET sAll_OF_Children["list"]:( "")
776 #####7.1 모든 상태의 프로세스 포함관계를 모두 구함
777
778 FOR i from:1 to:(sEndOfStateNum)
779 {
780     SETL state_num:(STR i)
781     SETL t:(0)
782     IF (nDebug = 1) {SET data05:(data05+state_num+": \t sEndOfStateNum \n")}
783     #7.1
784     FOR process in:(mD_Data[state_num+"_"+list])
785     {
786         IF (mD_Data[state_num+"_"+process+"_"+parent] = "")
787         {
788             IF (nDebug = 1) {SET data05:(data05+state_num+": "+root_"+process+": \t GET_Children \n")}
789             SET_ALL_OF_CHILDREN current_address:(root) next_address:(process) state_num:(state_num) sAll_OF_C
790         }
791     }
792
793 IF (nDebug = 1) {CC "AdoScript" FWRITE file:"C:/Users/default/Desktop/AH94BHT/Desktop/AD0xx_Debug_Text.dat
794
795
796 #####7.2 프로세스 포함관계 결과를 통해 7.1의 결과에 대한 프로세스의 Y축 크기 계산함
797 SET init_y:(1.0)
798 SET_ALL_OF_PROCESS_BLOCK_Y_POSITION current_address:(root) init_y:init_y sAll_OF_Children:sAll_O
799 IF (nDebug = 1) {CC "AdoScript" FWRITE file:"C:/Users/default/Desktop/AH94BHT/Desktop/AD0xx_Debug_Text.dat
800
801 #####7.5 프로세스 포함관계 결과를 통해 프로세스 역선불록의 X축 크기 계산함
802 #####7.6 프로세스 포함관계 결과를 통해 상급 역선불록의 X축 크기 계산함
803 SET data10:(" ")
804 SET data11:(" ")
805 SET data13:(" ")
806 SET prev_blockaddress:(map()) #프로세스 블록 계산
807 SET prev_sbblockaddress:(map()) #상급블록 계산
808 FOR i from:1 to:(sEndOfStateNum)
809 {
810     SETL state_num:(STR i)
811     SETL t:(0)
812     IF (nDebug = 1) {SET data10:(data10+state_num+": \t sEndOfStateNum \n")}
813     #7.5
814     FOR process in:(mD_Data[state_num+"_"+list])
```

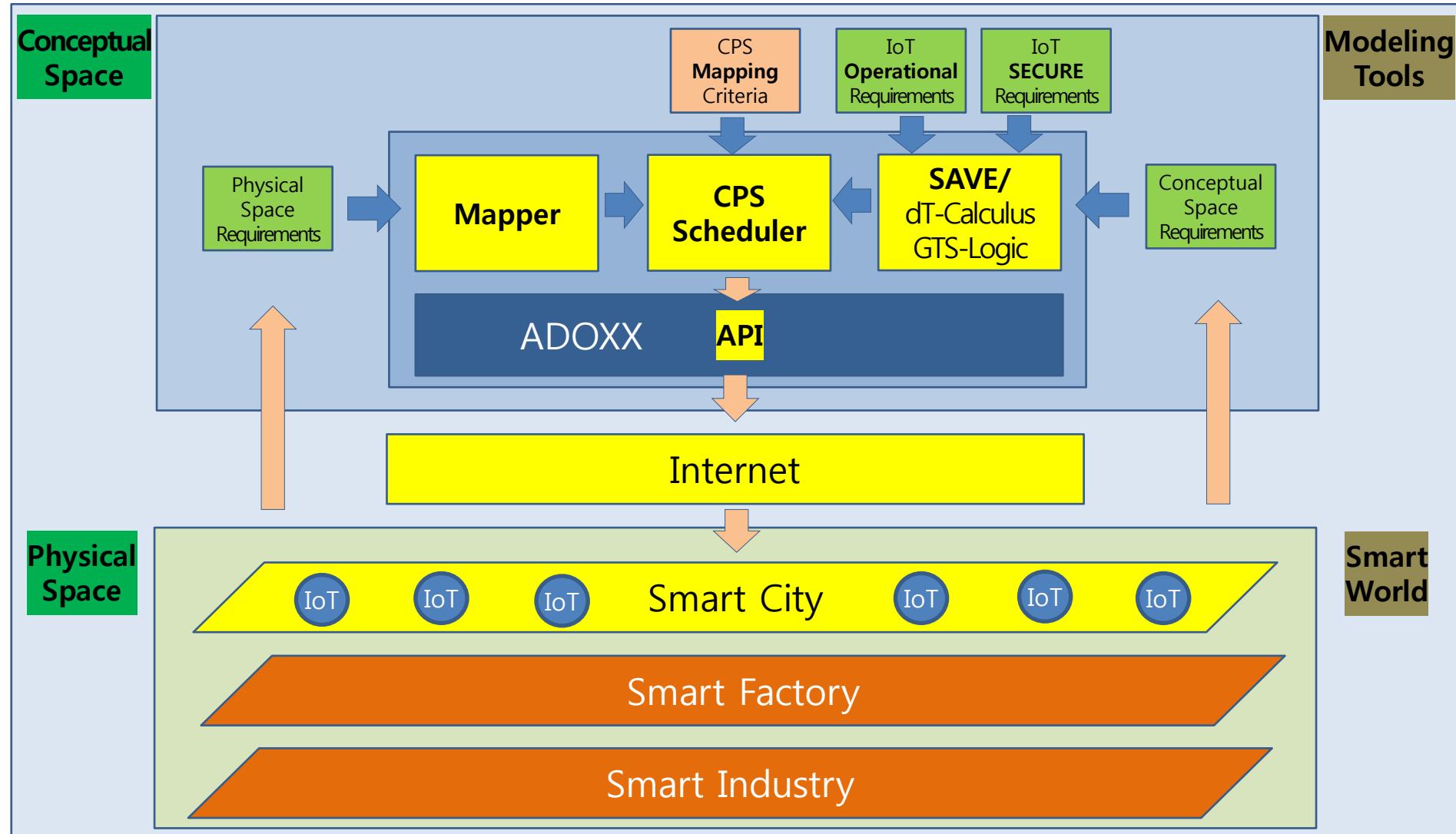
DEMO: SAVE EMS



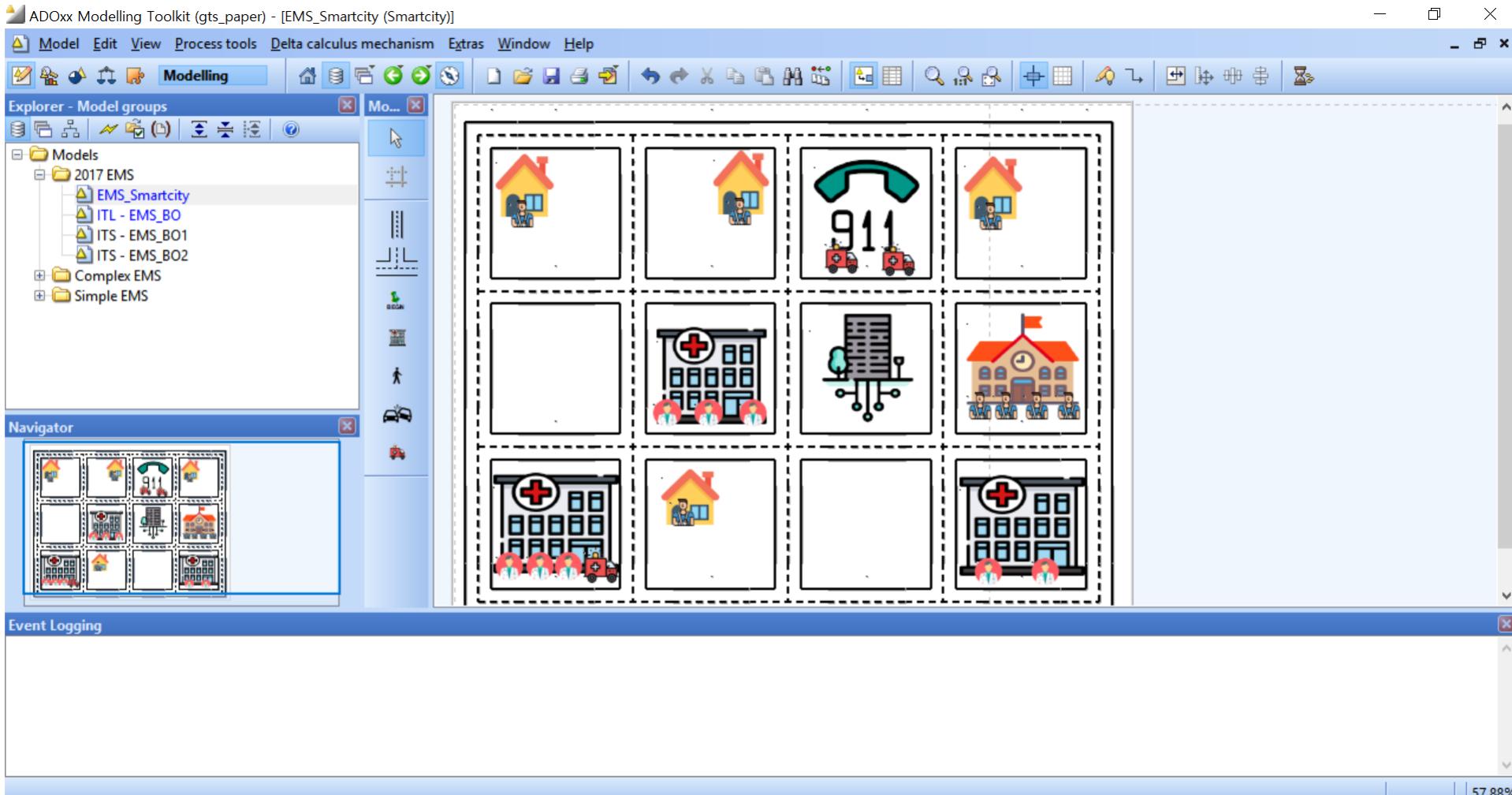
1. Motivation
2. Modeling
 - 1) Specification: δ -Calculus
 - 2) Verification: GTS Logic
 - 3) EMS Example
3. SAVE Tool
- 4. Cyber-Physical Systems Application**
5. Summary
6. Discussion

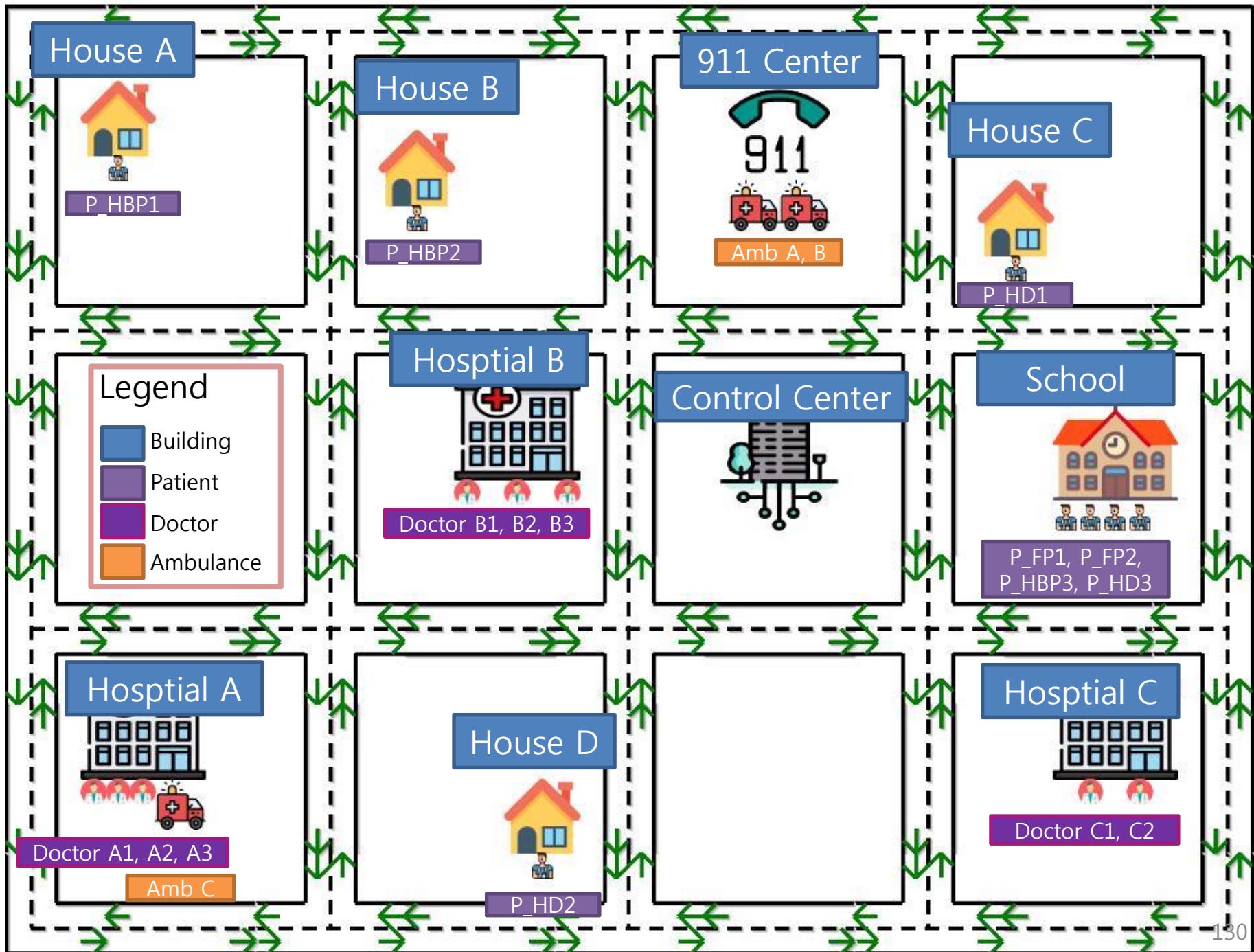
4. **CYBER PHYSICAL SYSTEMS ON MODELING**

Application: CPS

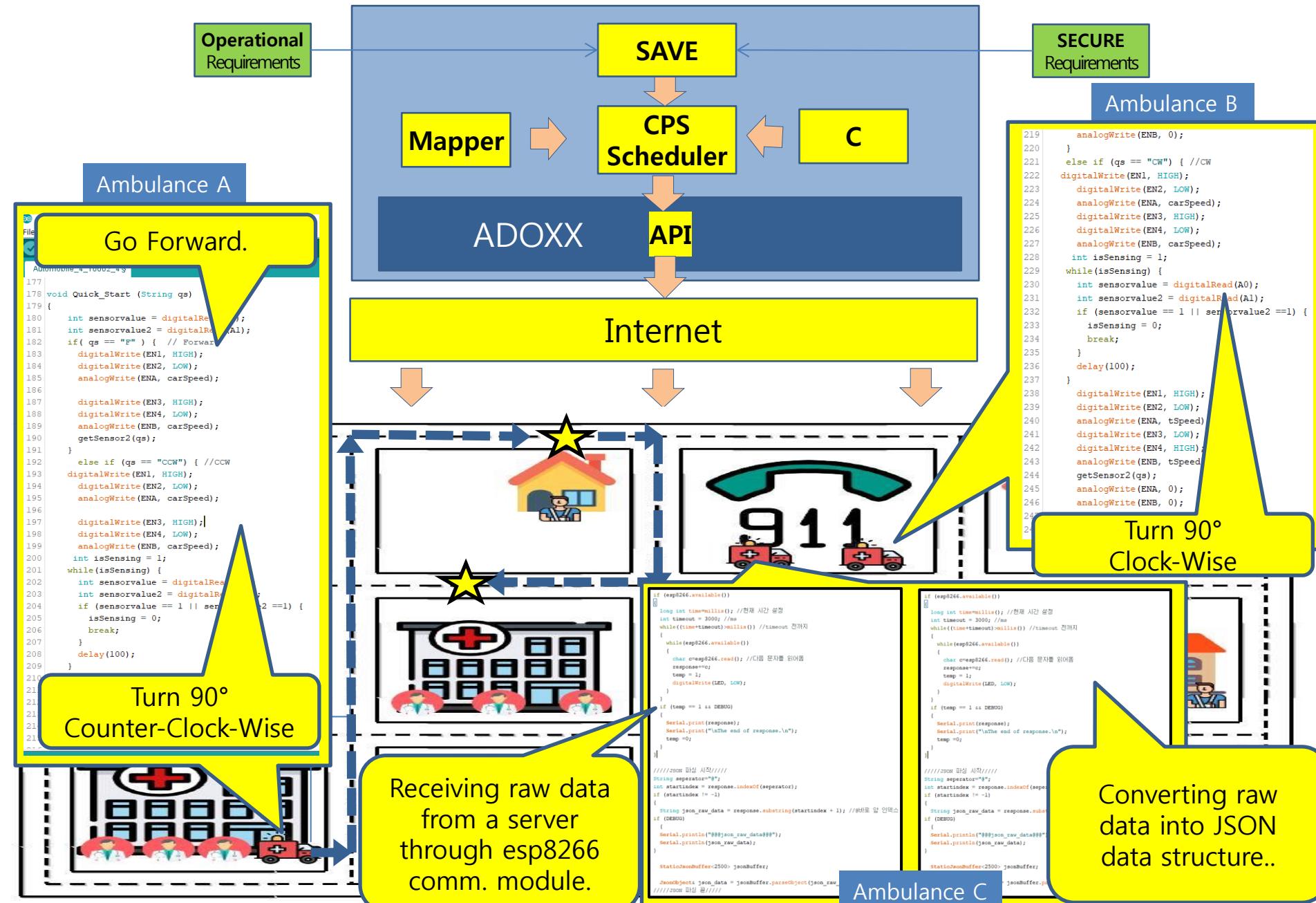


SAVE/Map

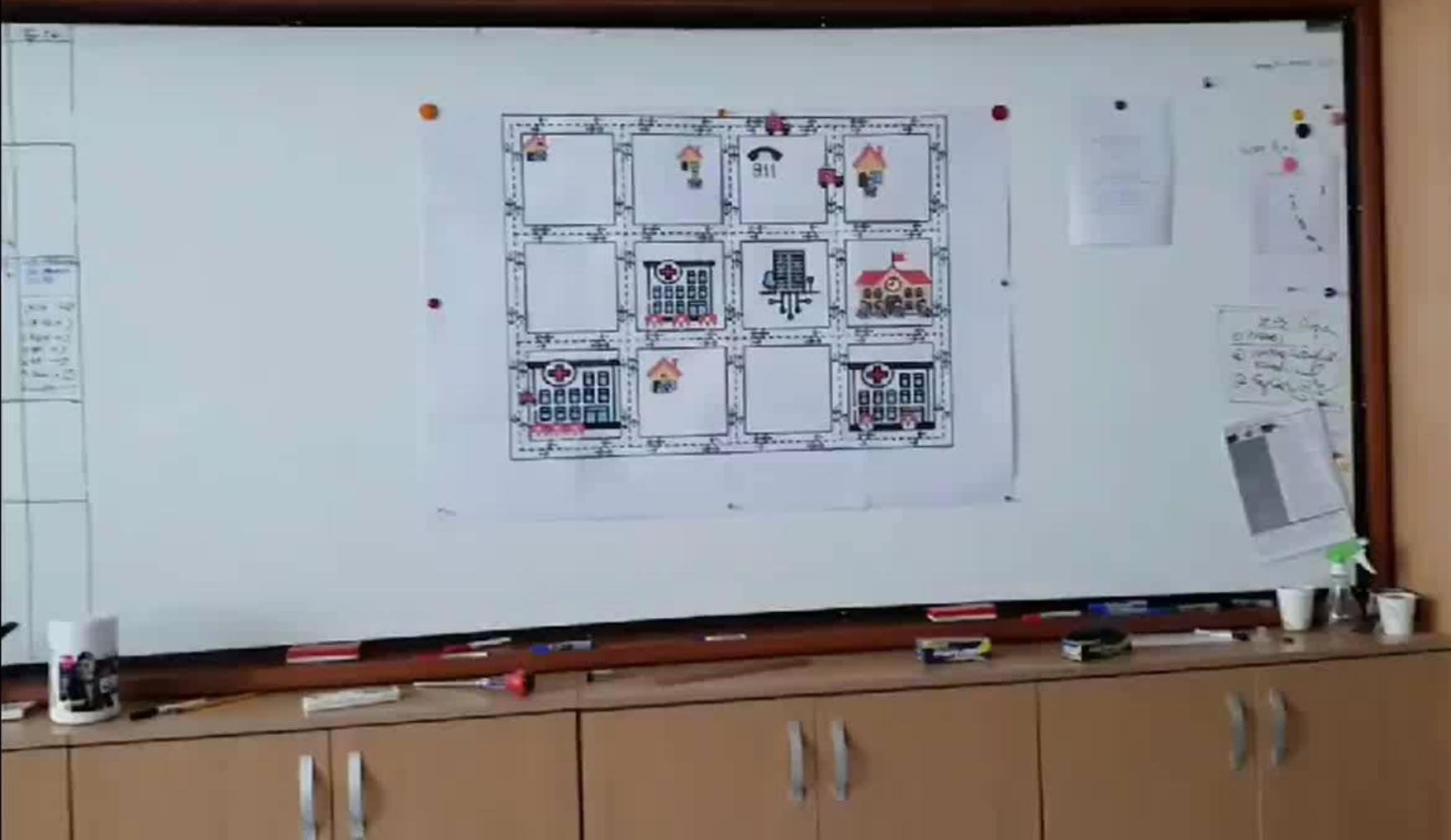




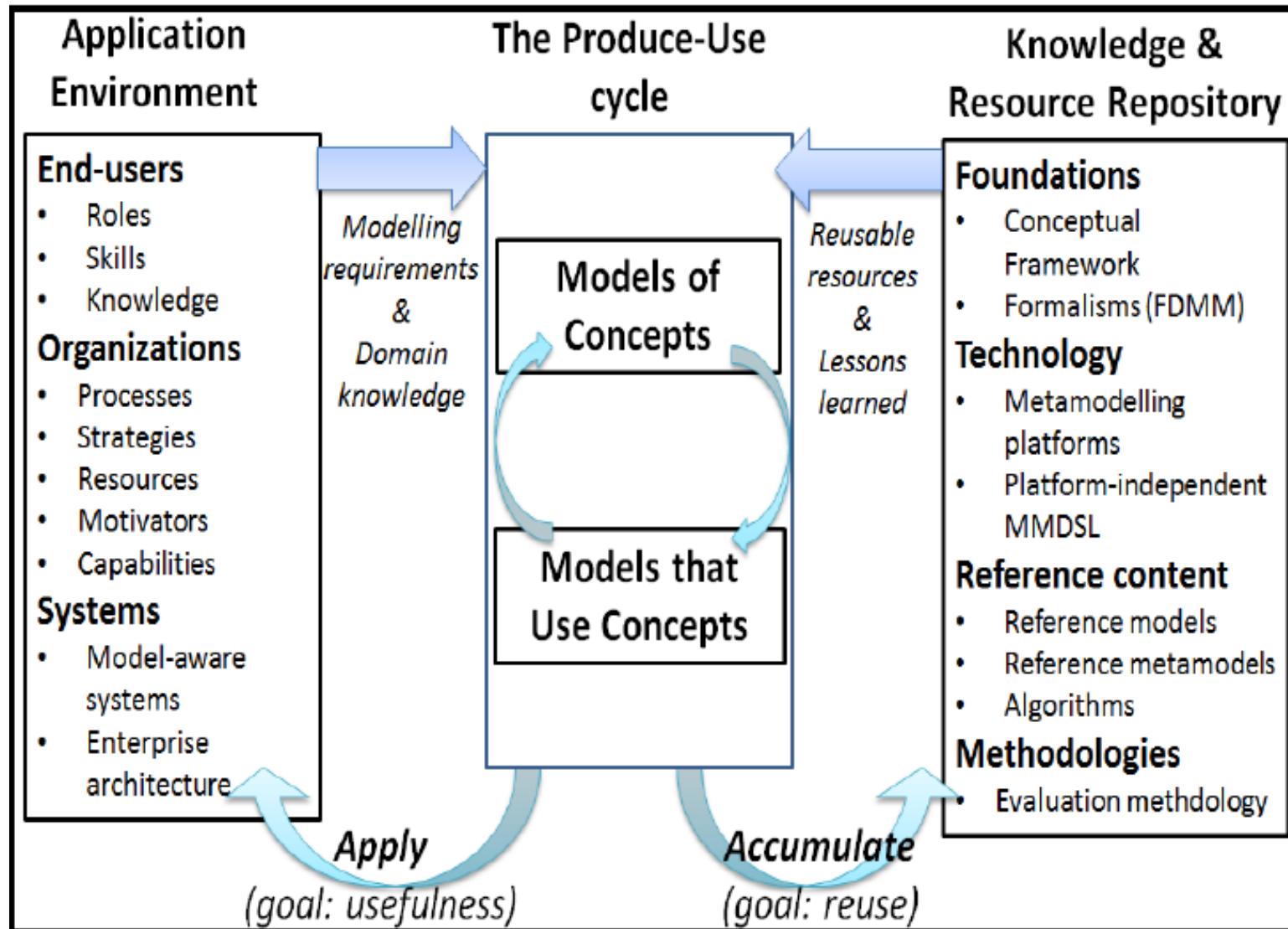
Distributed Real-time Mobile Systems: Scheduling



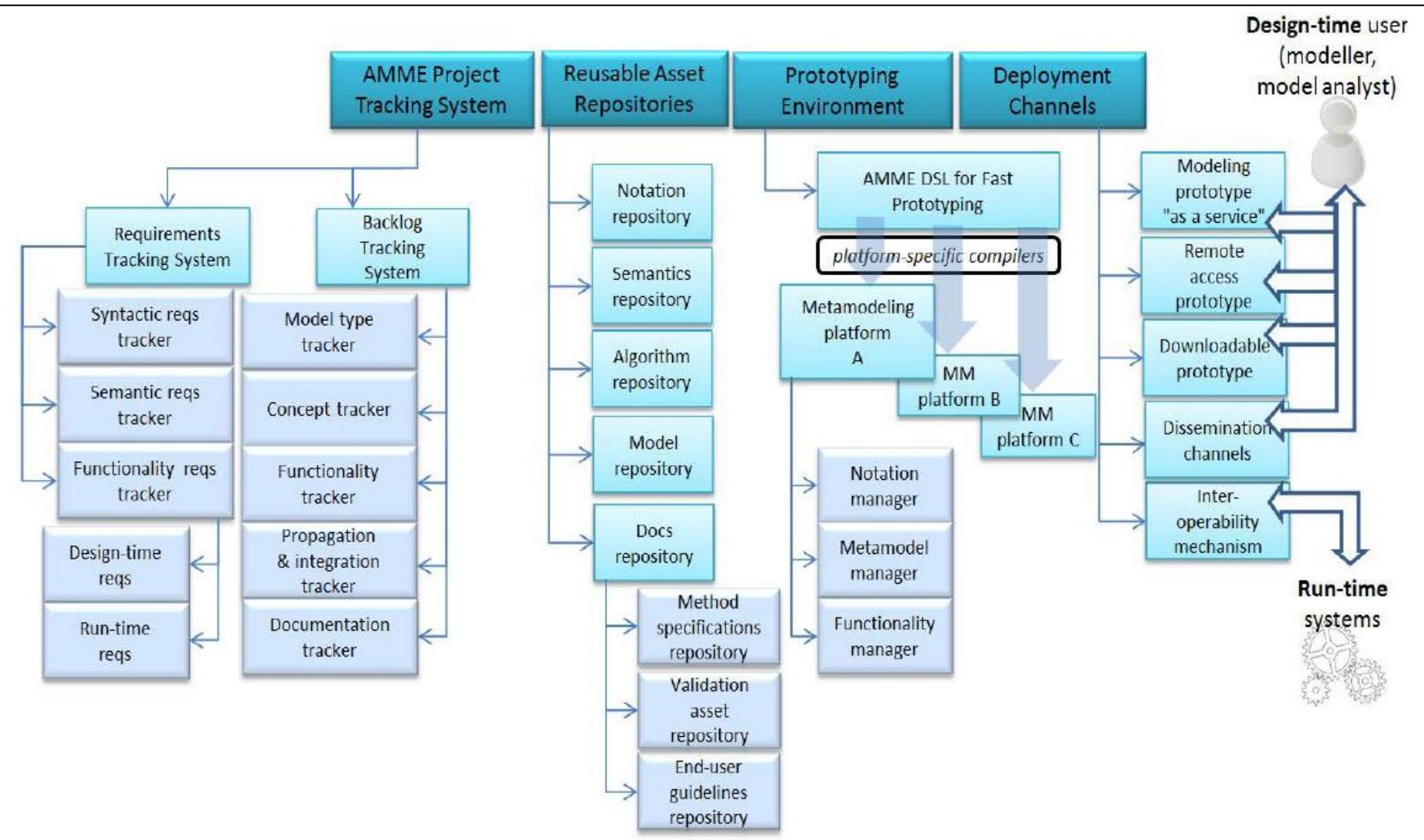
SAVE/EMS/CPS Example



The AMME Framework



AMME Architecture



Implementation Issues

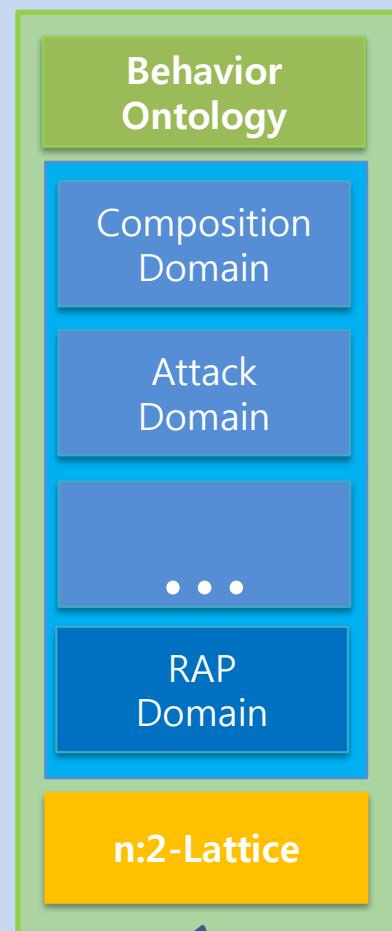
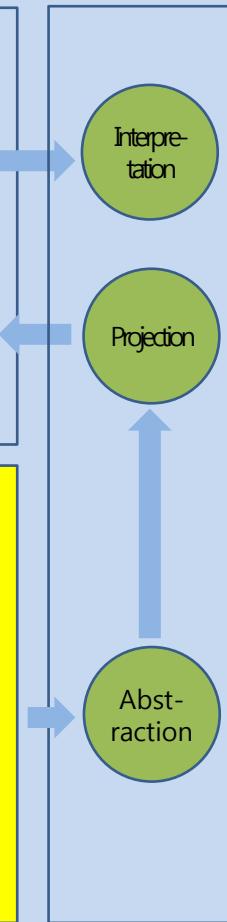
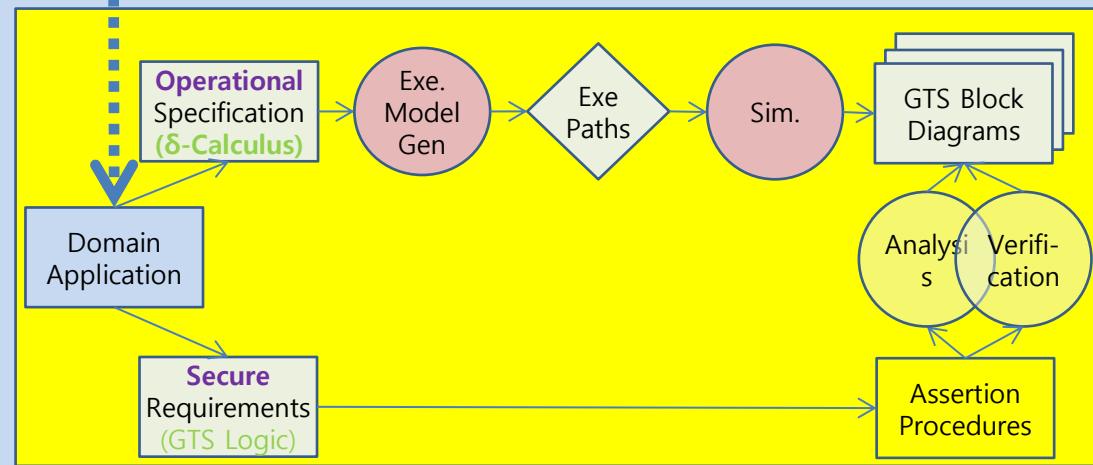
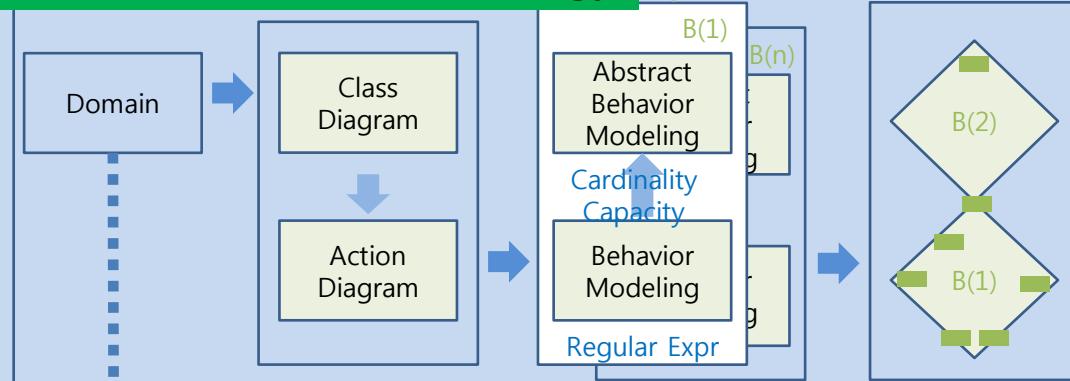
- Main Characteristics
 - Distribution
 - Real-time
 - Mobility
 - Secure: Safety, Security
 - Exceptional handling
 - Fault-tolerance
- Scheduling
 - Real-time
 - Monitoring
 - Autonomy
 - Security
 - Sensors

1. Motivation
2. Modeling
 - 1) Specification: δ -Calculus
 - 2) Verification: GTS Logic
 - 3) EMS Example
3. SAVE Tool
4. Cyber-Physical Systems Application
- 5. Summary**
6. Discussion

5. SUMMARY

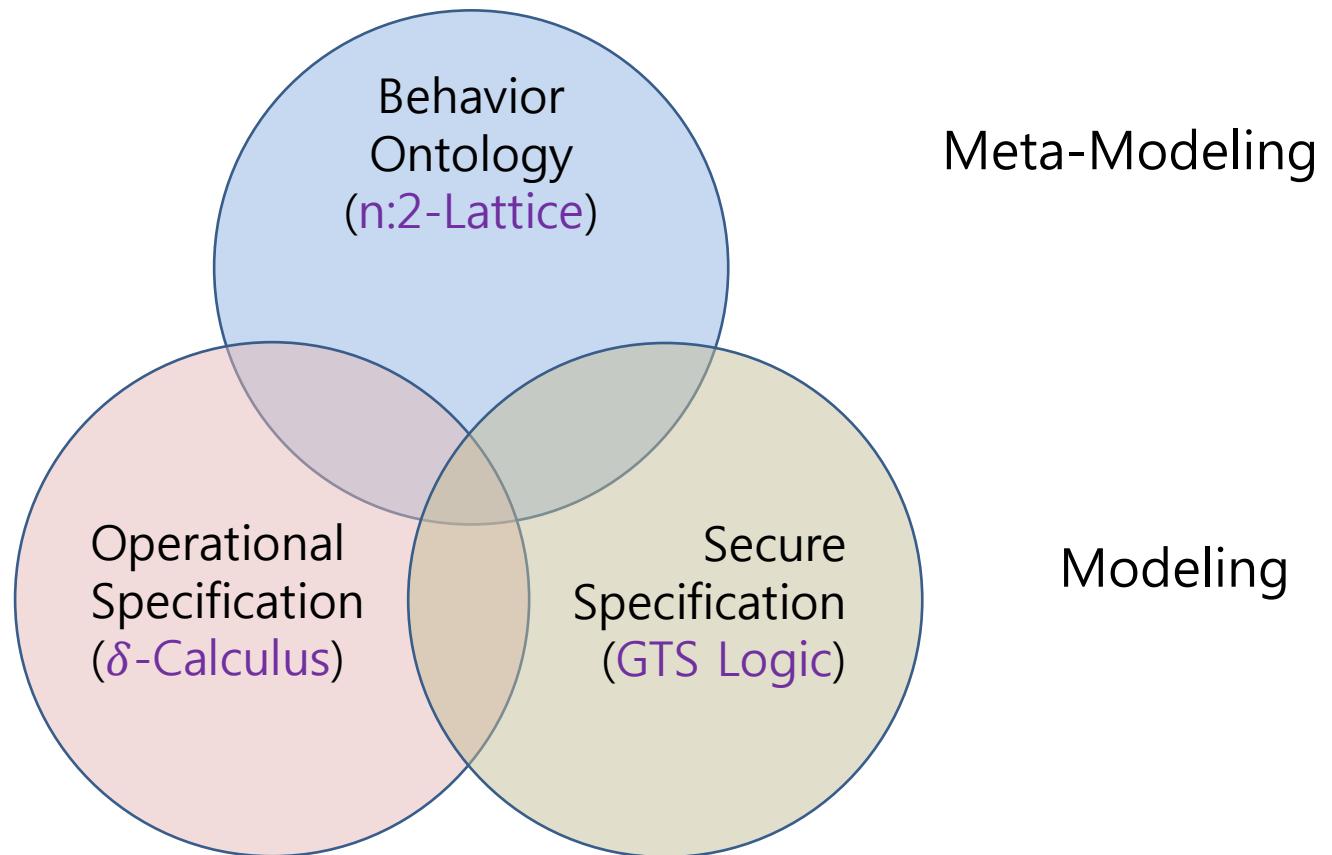
Approach: SAVE

PRISM for Behavior Ontology

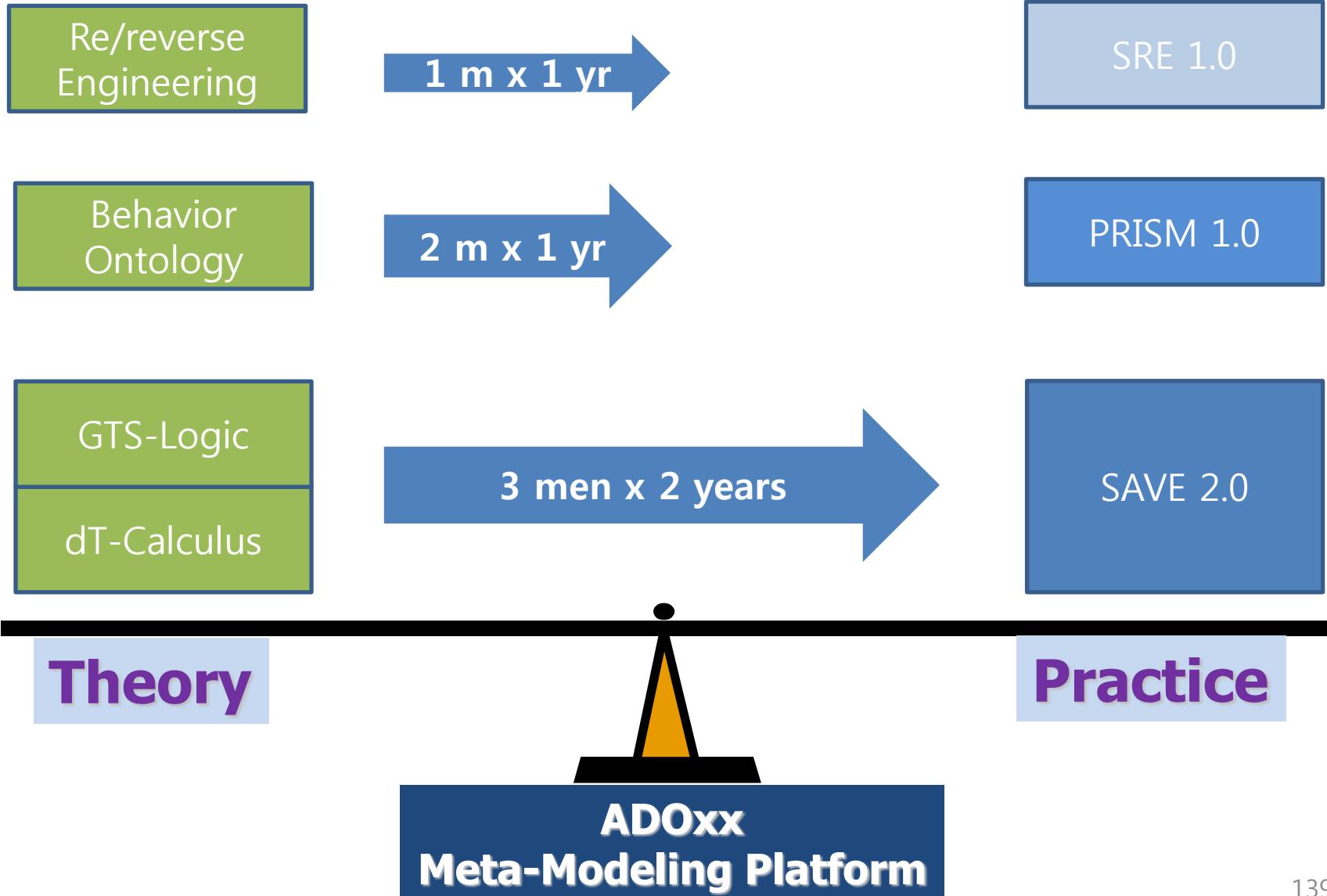


SAVE(Specification, Analysis, Verification, Evaluation)

Theories



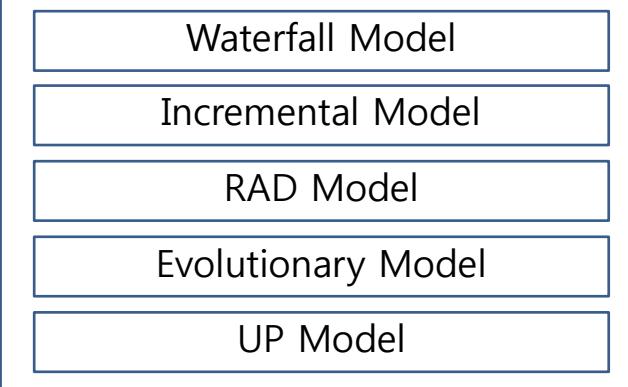
SW Engineering



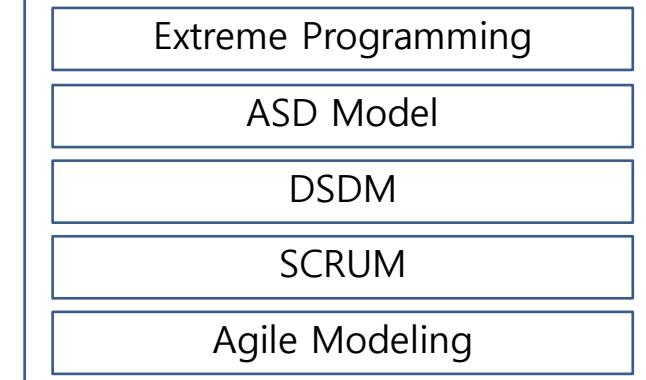
SW Engineering Process Models

R1

Prescriptive Process Model



Agile Process Model



R2

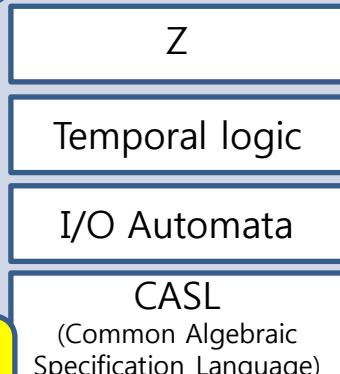
Generic Engineering Process



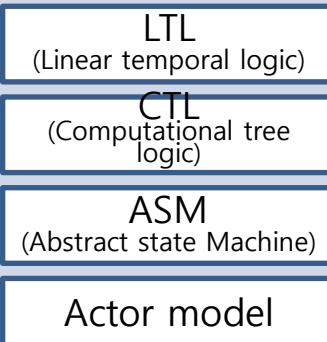
Modeling/Formal Methods

R3

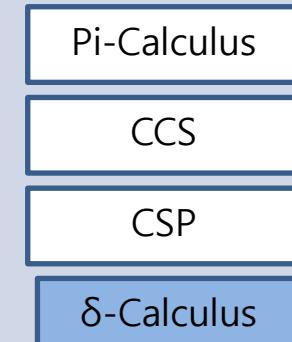
Logic



State Machine



Process Algebra



R4

TOOLS

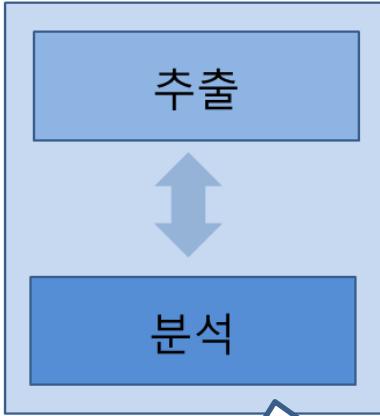
R5

Object-Oriented Paradigm

R1 & 2: FMM: Formal Method Model

SW Engineering Process Model

Requirements Extraction/Analysis



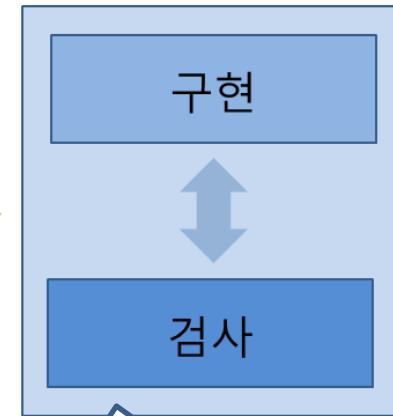
Modeling : Design/Specification

Specification & Verification

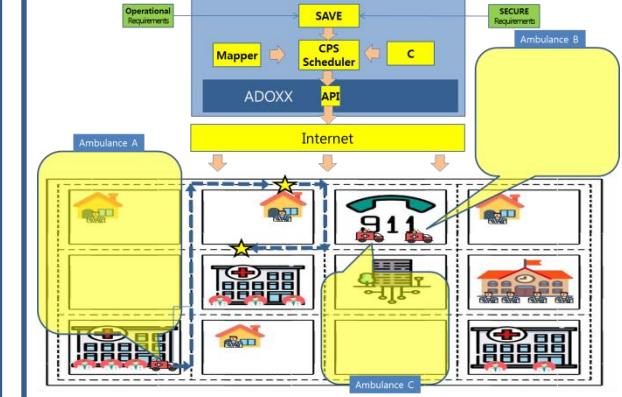
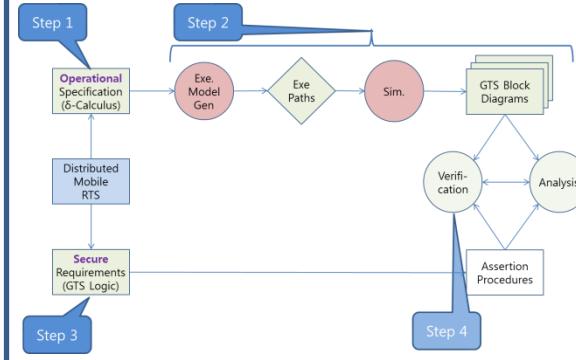
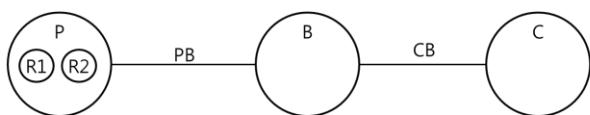


CPS Implementation

Implementation & Testing



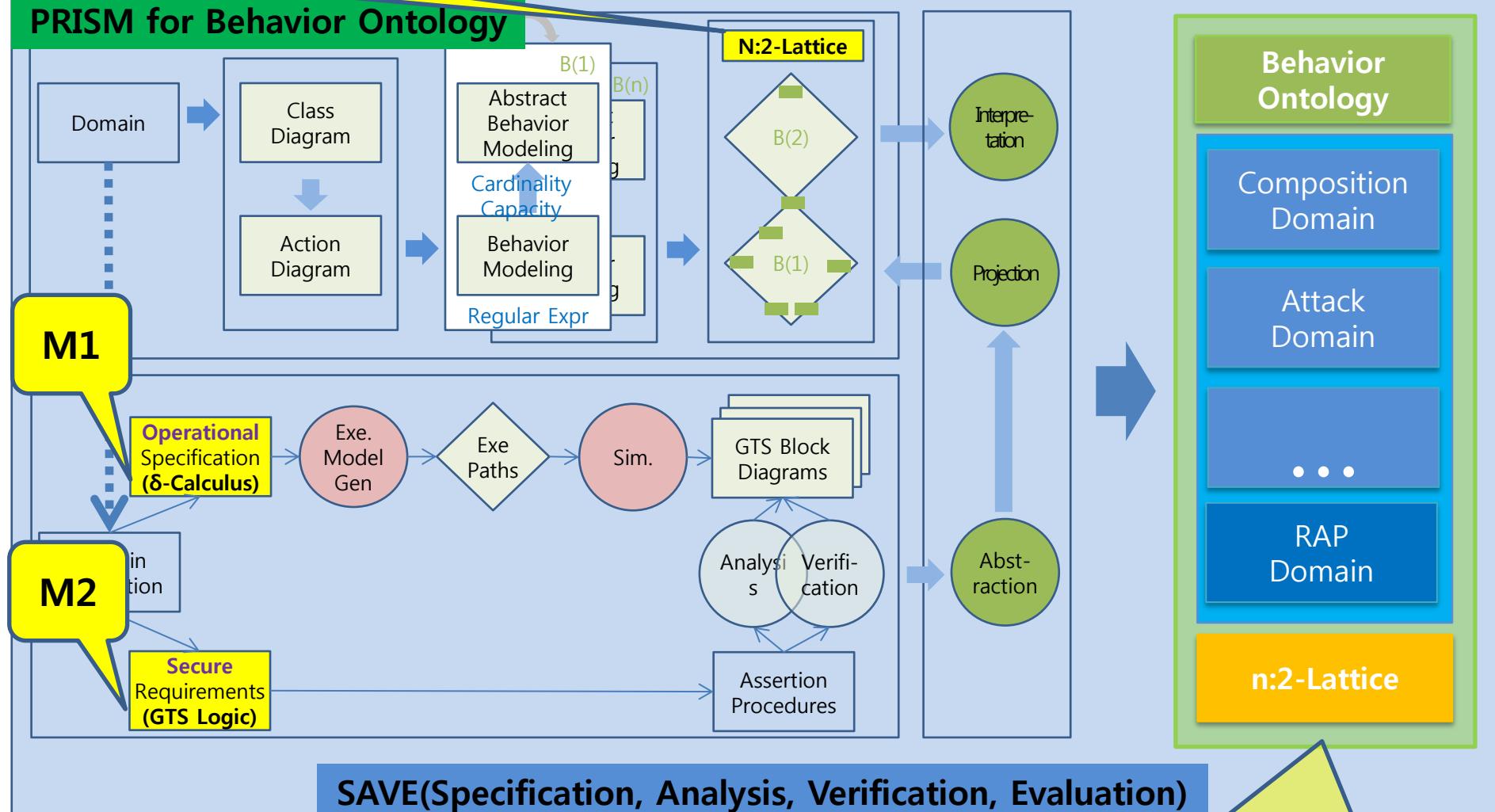
- Operational Requirements**
 - Producer produces two resources, R1 and R2.
 - Producer stores the resources in Buffer in order.
 - Producer informs Buffer of the order of R1 and R2, or R2 and R1.
 - Consumer consumes the resources from Buffer in order.
 - The order of the consumption is formed to Buffer by Consumer.
- Secure Requirements**
 - Security:**
 - The order 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.
 - Safety:**
 - The resources produced by Producer should be consumed by Consumer less than 5 minutes.



B3: Formal Methods and Mathematical Structures

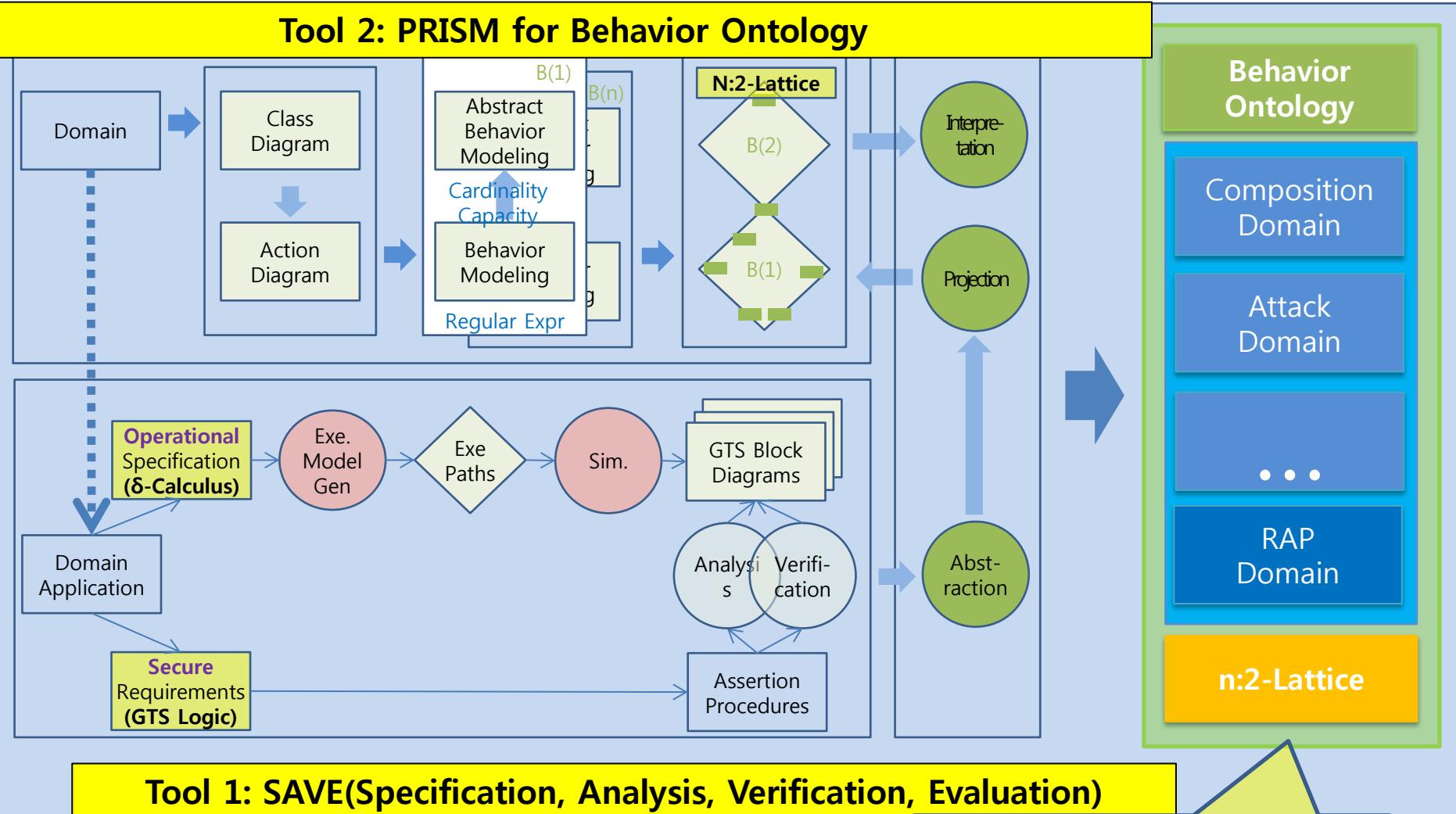
M3

PRISM for Behavior Ontology

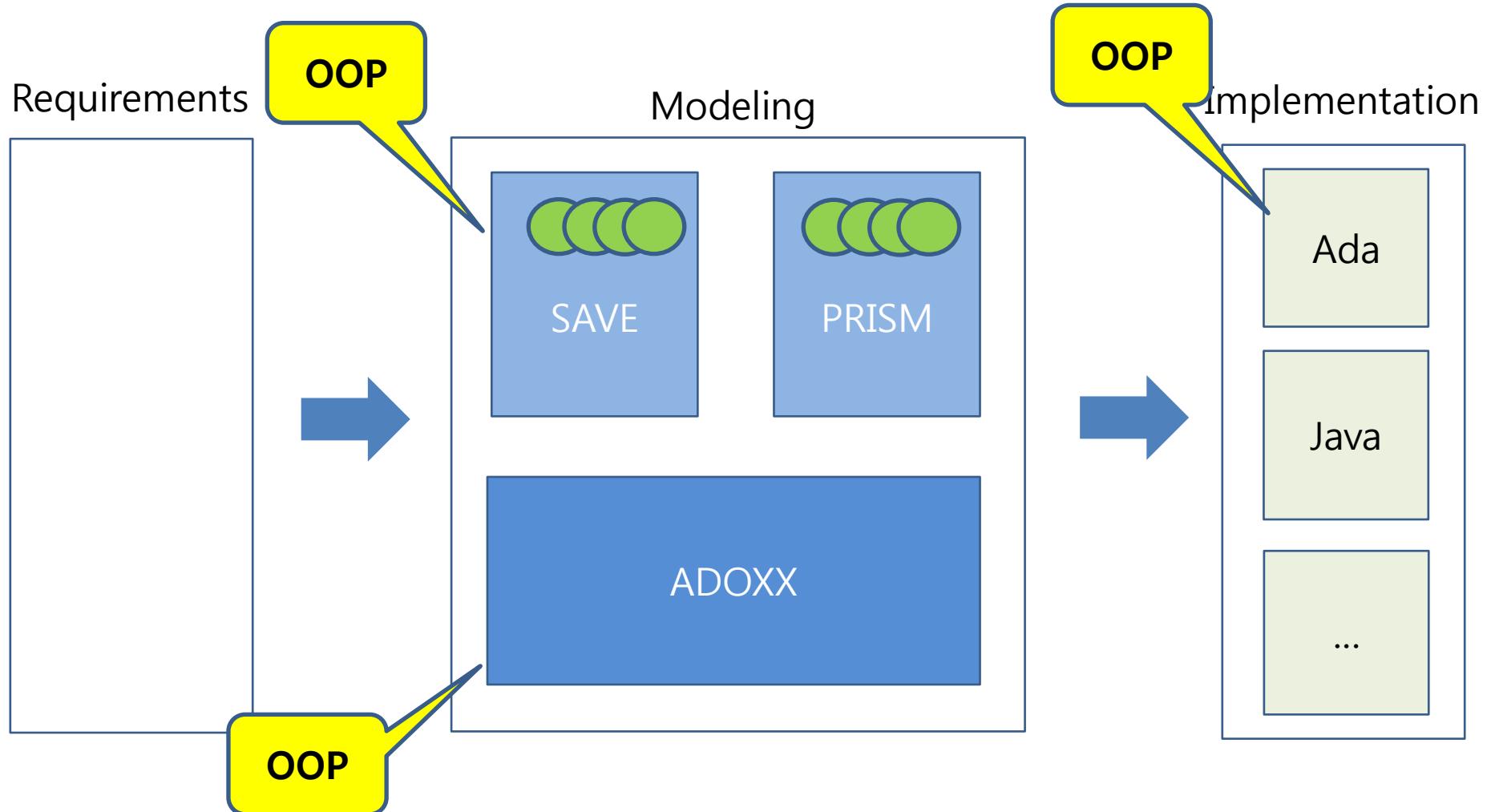


R4: SAVE & PRISM

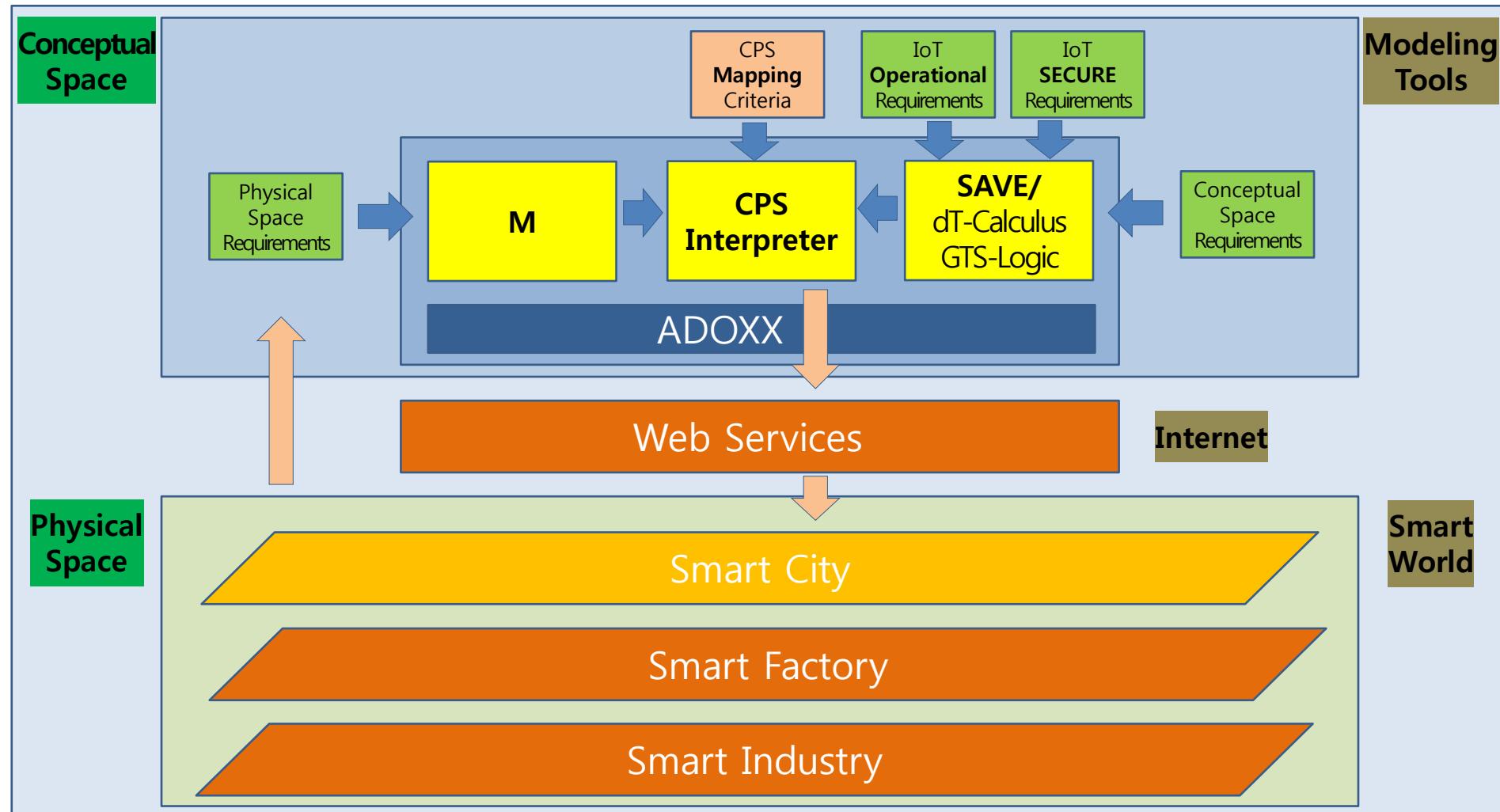
Tool 2: PRISM for Behavior Ontology



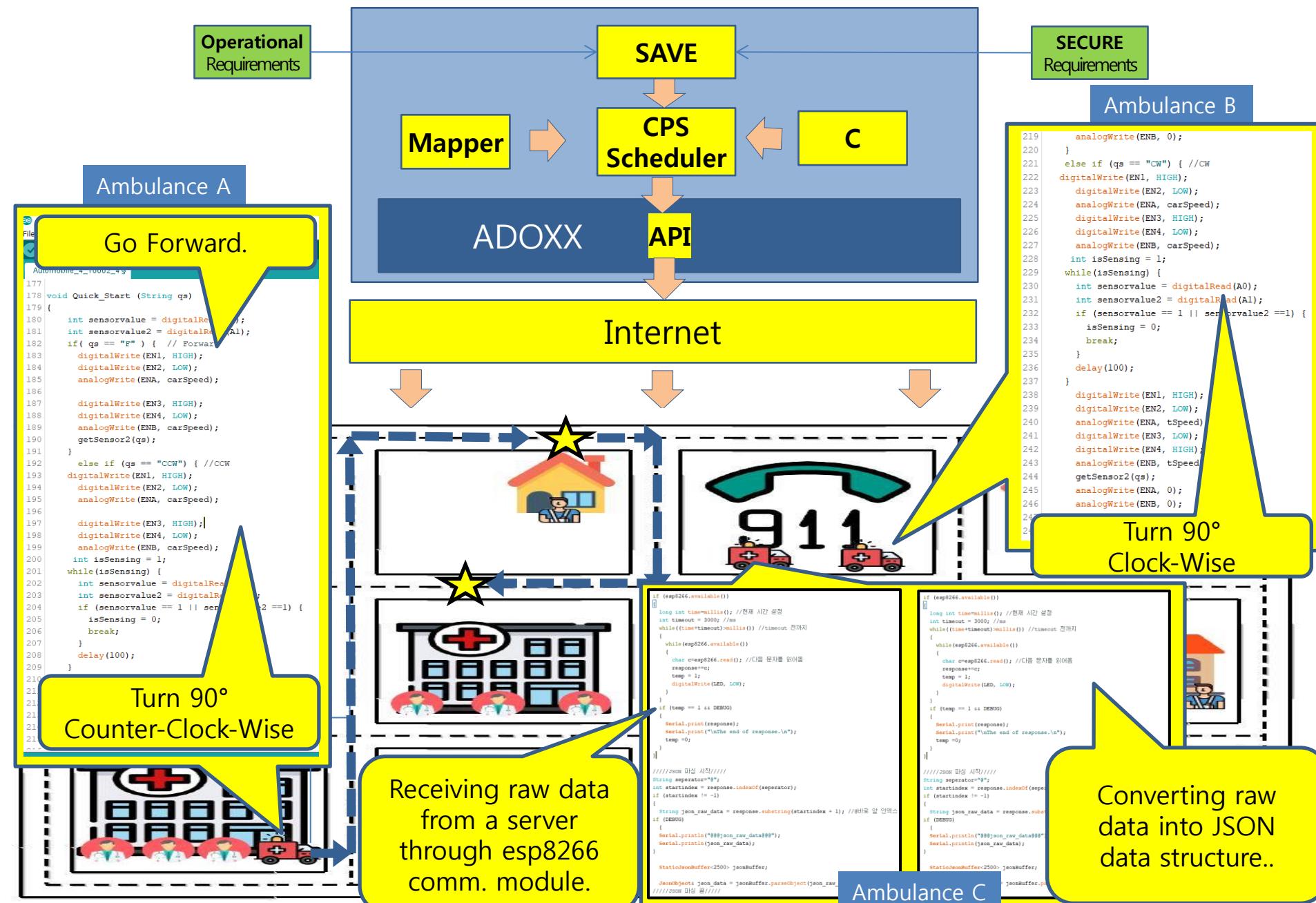
R5: Object-Oriented Paradigm



Application: CPS

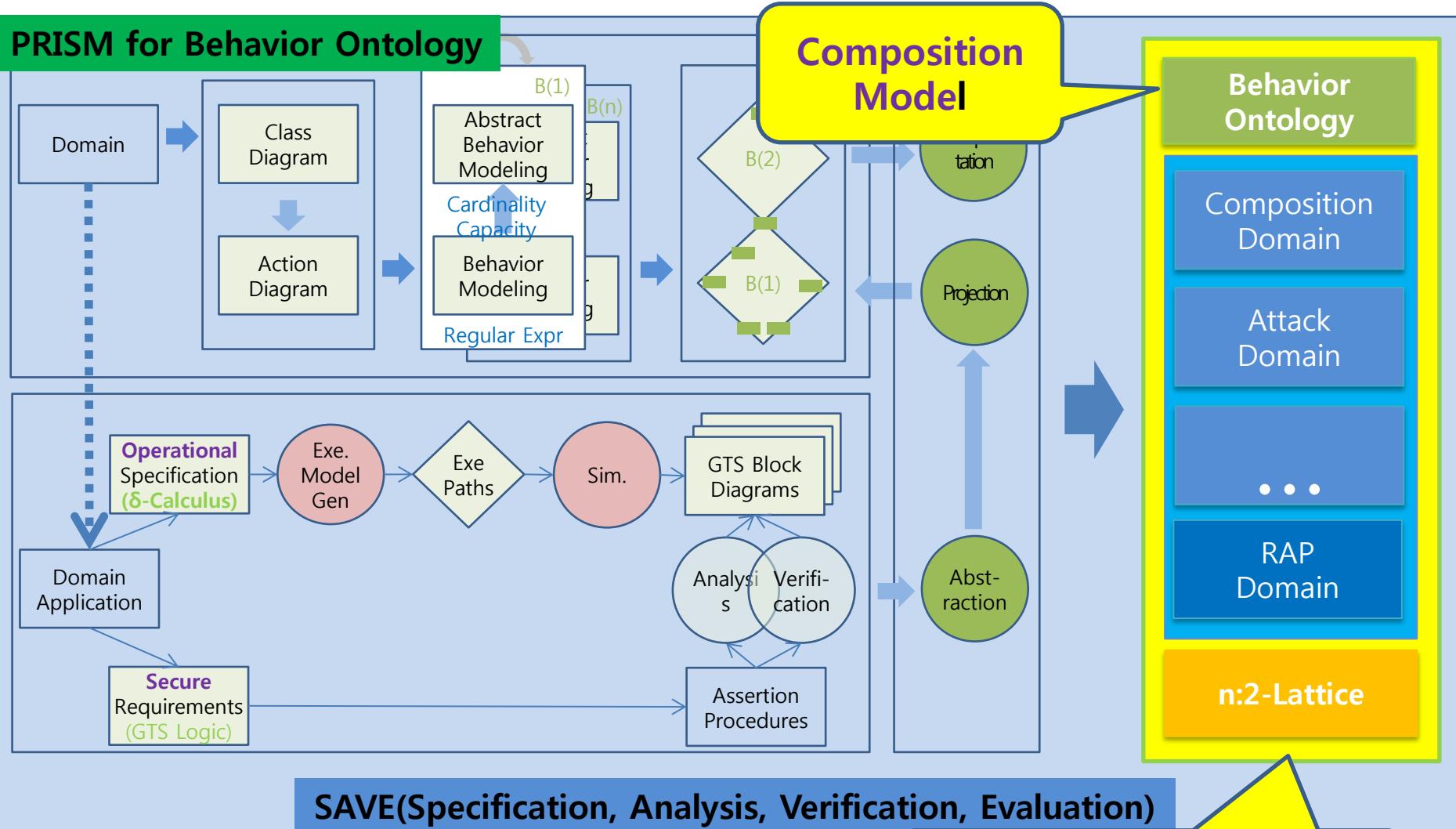


Distributed Real-time Mobile Systems: Scheduling

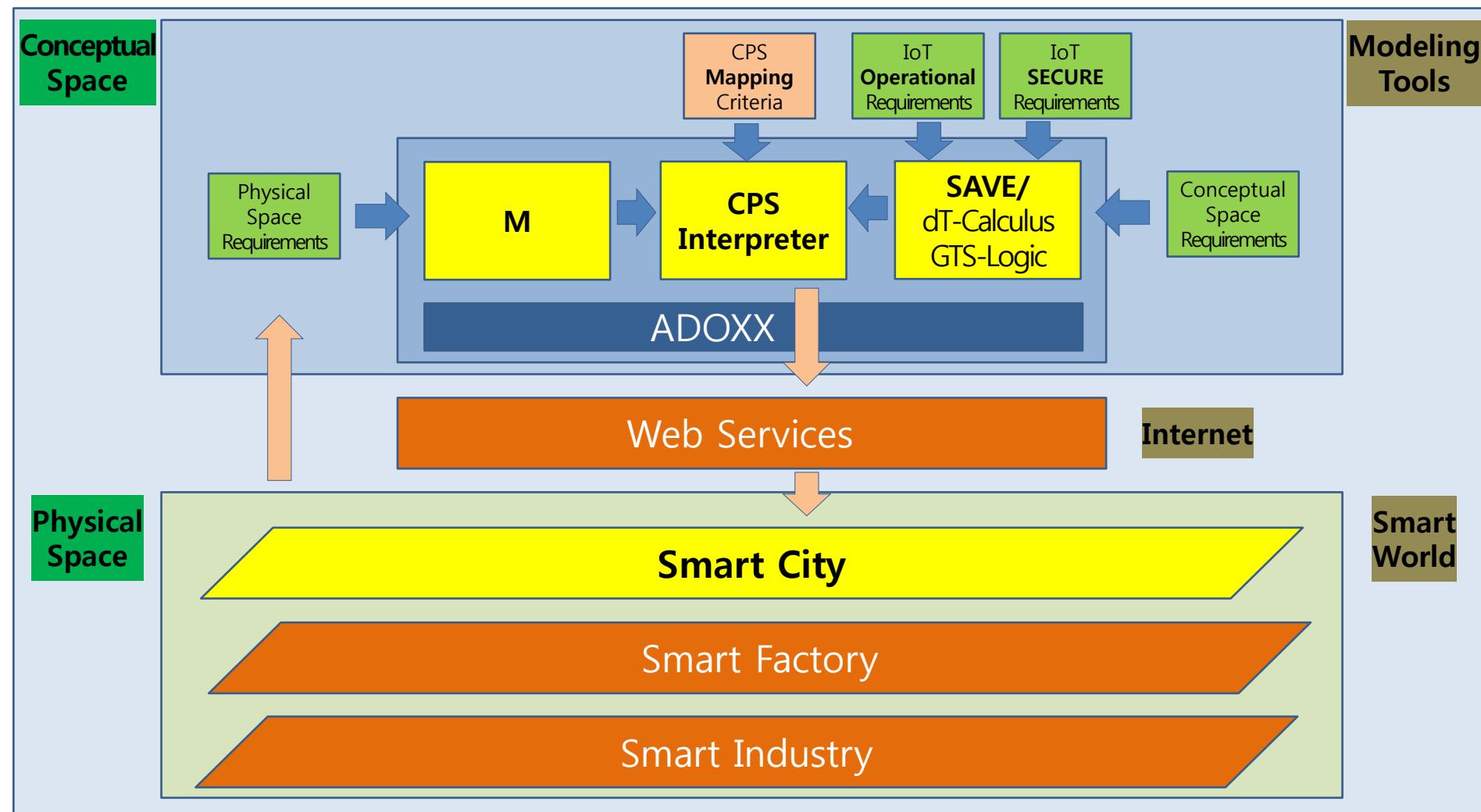


NEMO'19-A: Knowledge Engineering

PRISM for Behavior Ontology



NEMO'19-B: CPS/Smart City



1. Motivation
2. Modeling
 - 1) Specification: δ -Calculus
 - 2) Verification: GTS Logic
 - 3) EMS Example
3. SAVE Tool
4. Cyber-Physical Systems Application
5. Summary
- 6. Discussion**

6. **DISCUSSION**

OMiLAB-Rob Experiments

[Propose New](#)

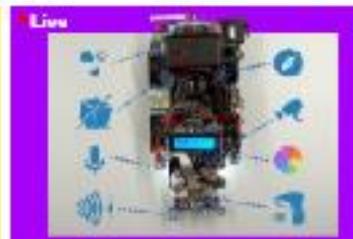
User Case: Interact with different real world objects
Approximation Problem: Items have a complex shape in 3 dimensions.



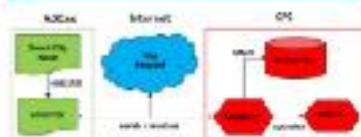
User Case: Assemble a product by incorporating components
Planning Problem: Reaching the goal state in a complex environment



User Case: Capture characteristics of a system without irrelevant details
Modeling Problem: Finding the right angle of minimum uncertainty



User Case: Enable new applications for users with assistive devices
Sensor Fusion Problem: Integrate sensor information in a convenient model



The aim of this project is the integration of physical and model environment within a Smart City Scenario. ADONet http requests are used to establish a connection between the smart city model and two MDRVs.



User Case: Self-identification within CPS sowie dessen Darstellung und experimentelle Möglichkeiten mittels ADONet



The Smart Drone Tourist Guide is an autonomous drone which accompanies a user on a sightseeing tour



User Case: Simulation of a delivery process in warehouses
Cross-logic Scenario Annotations using the GeMILIS modelling toolkit



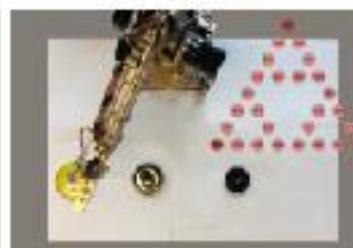
User Case: Learn and play a simple piano song.
Genetic Algorithm: Learning the keys



User Case: Consider a domain-specific modeling method
Metamodeling: Service-based ADONet

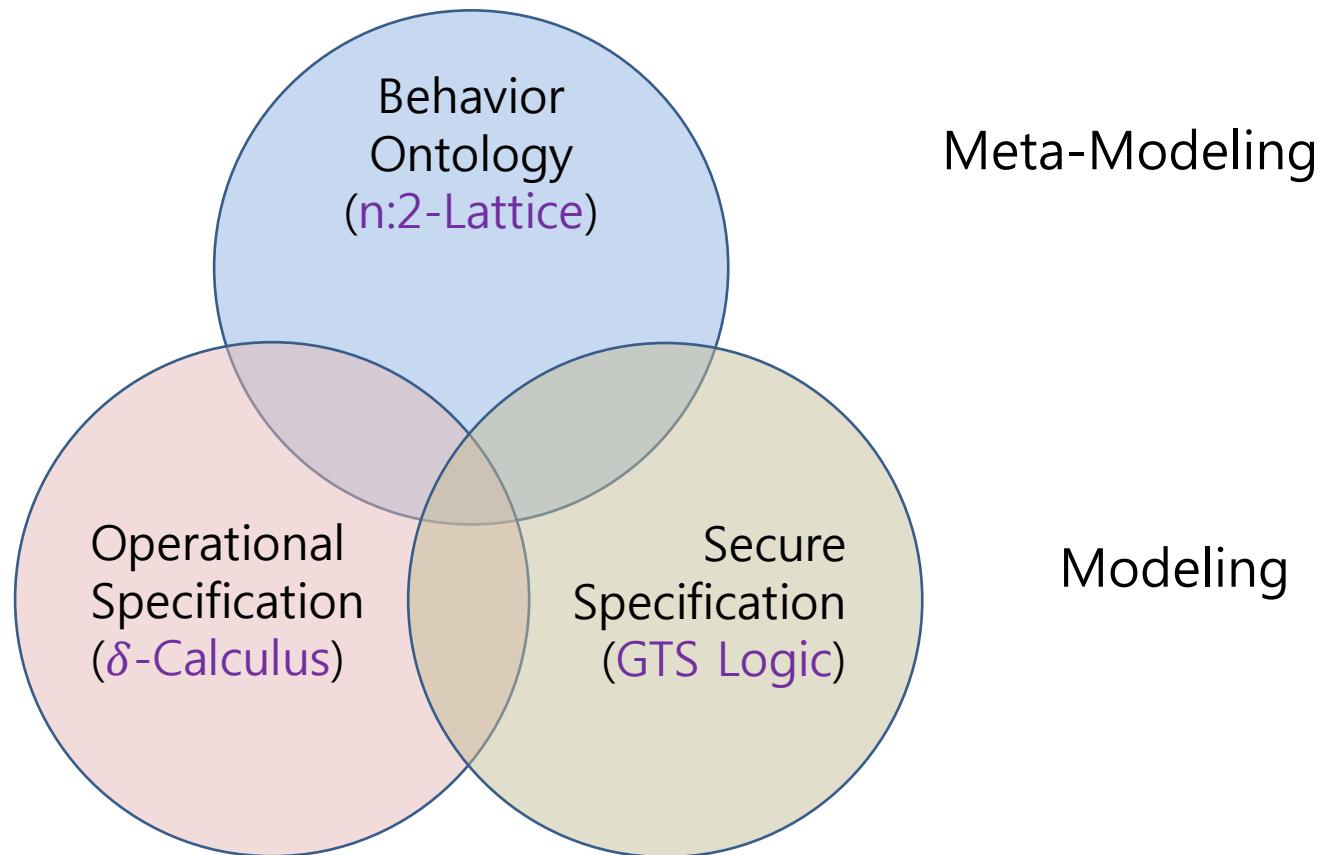


User Case: Mixing cocktail by receiving speed input
Integration: Combining different

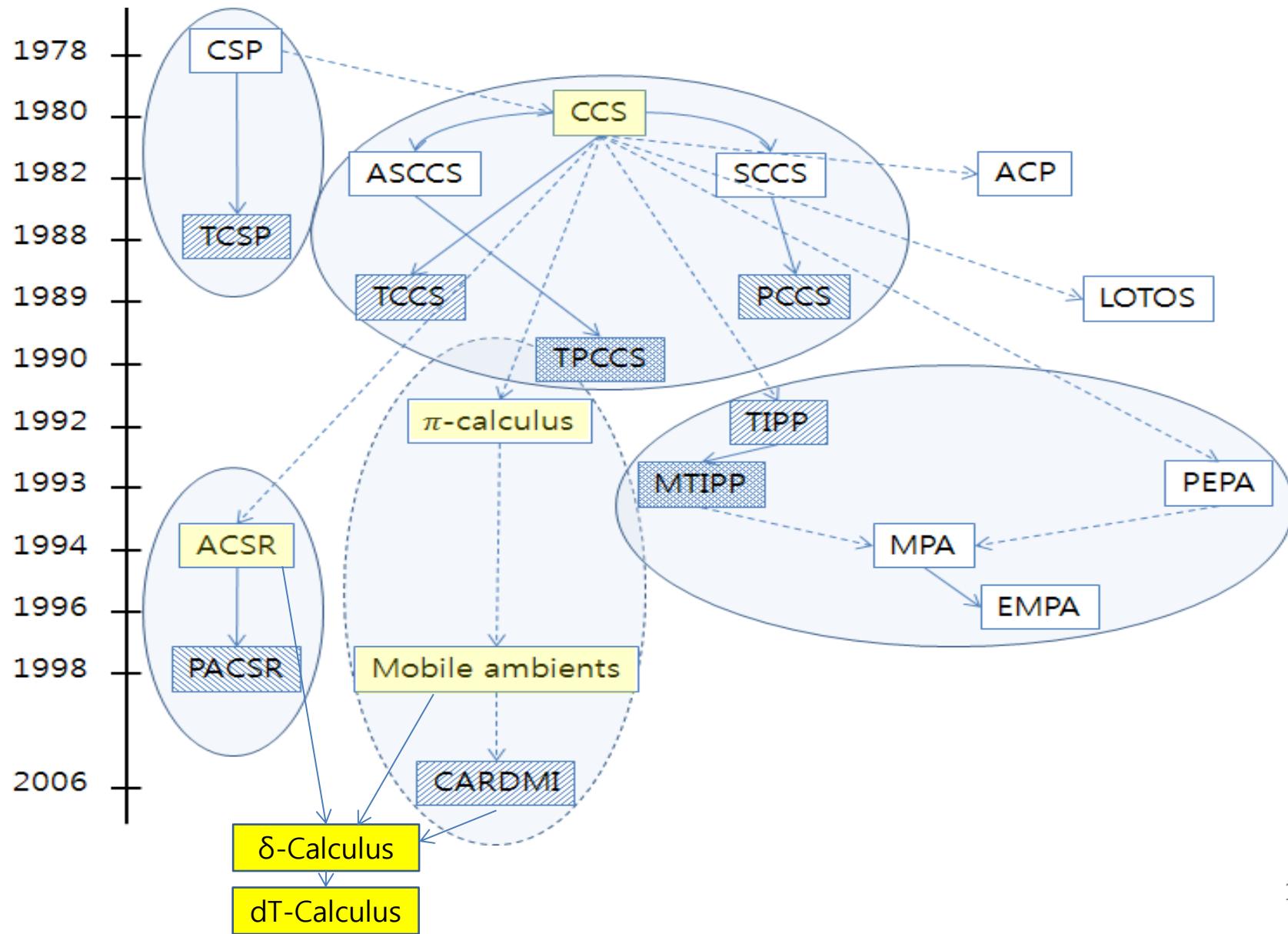


User Case: Tosses of items with these rules
Constraint Satisfaction: Formalize

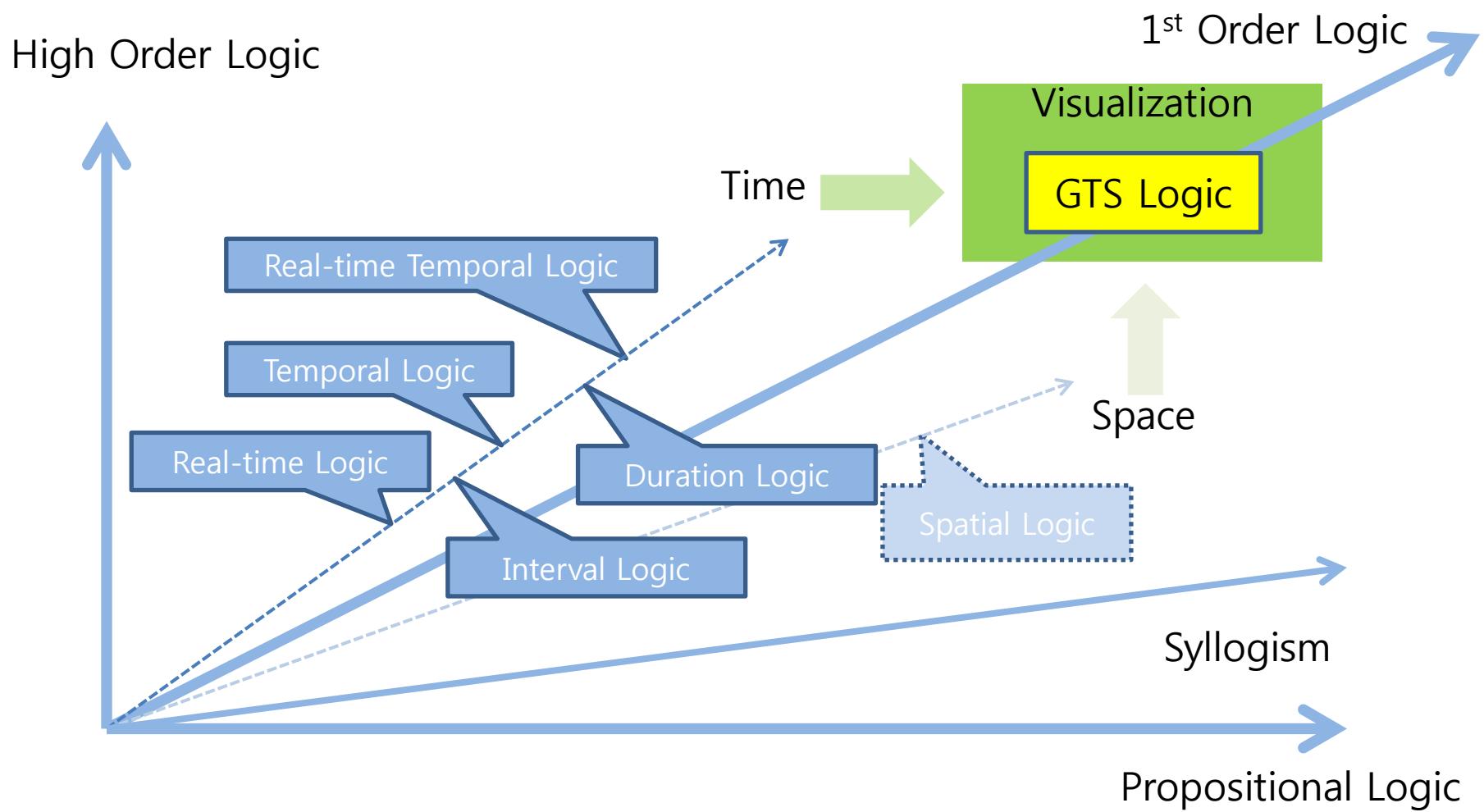
Theories



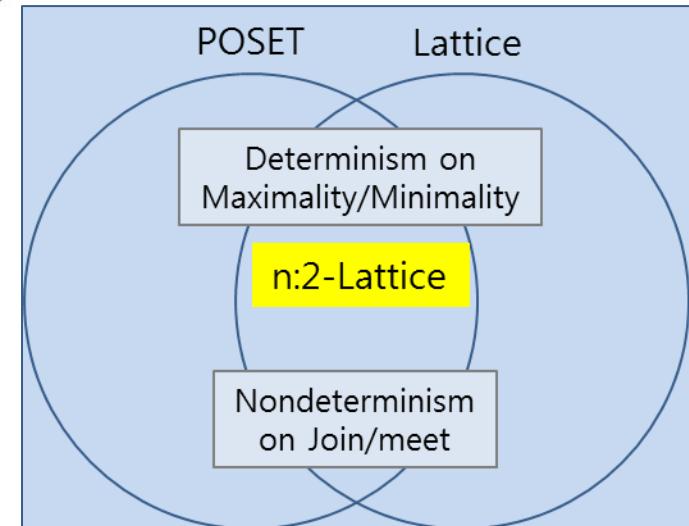
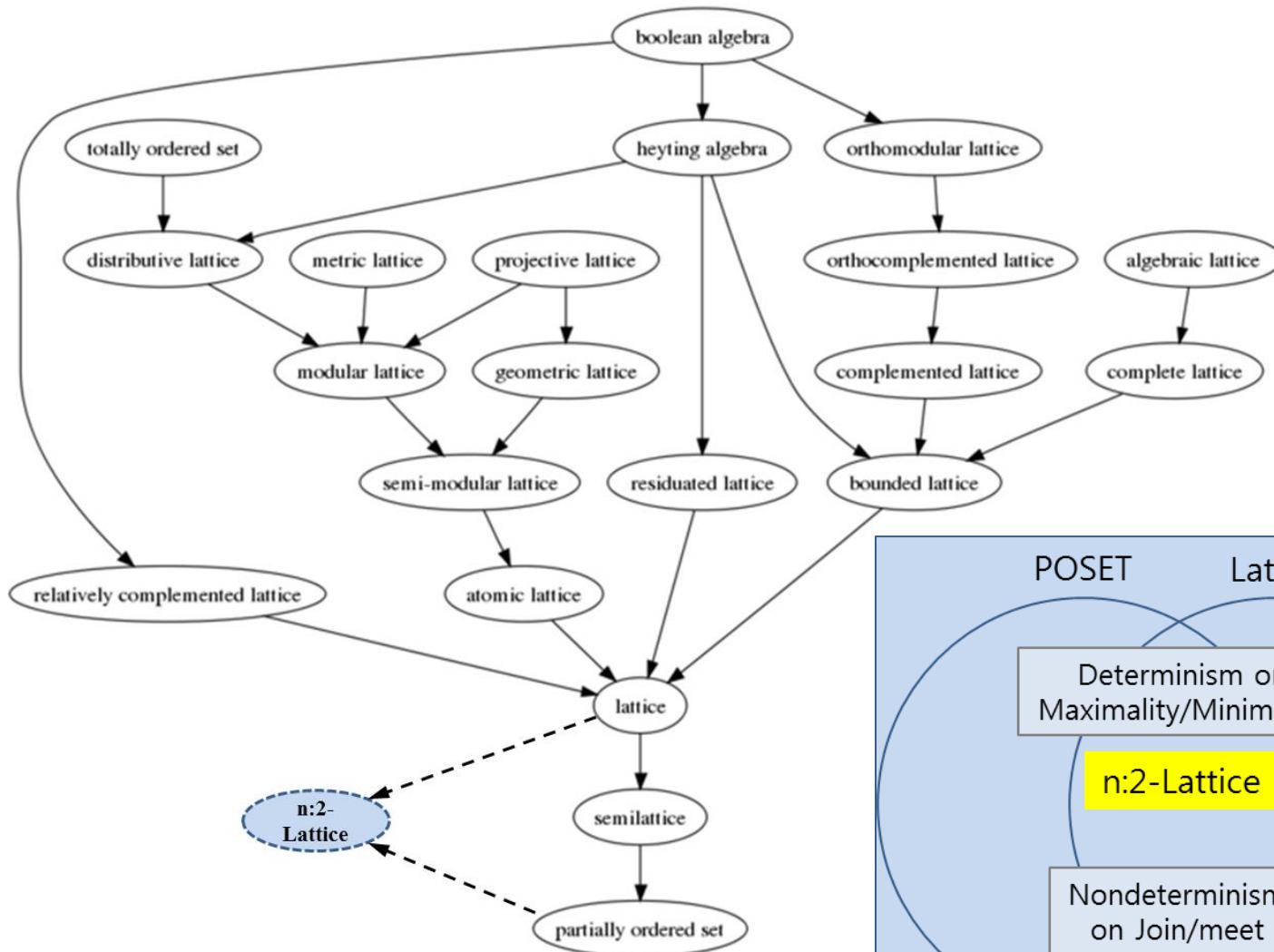
Map of Process Algebra



Dimension of Logic



Map of Lattices



Recent Publication

[Since 2012]

- **Books [Korean]**
 - M. Lee, *Formal Methods*, Chonbuk National University Press, Korea, Aug. 2018.
 - M. Lee, *Process ALgebra*, Ewha Press, Korea, Aug. 2015.
 - M. Lee, *Behavior Ontology*, Ewha Press, Korea, Aug. 2013.
- **Chapters:**
 - Y. Choe, M. Lee, Algebraic Method to Model Secure IoT., Edited D. Karagiannis, et. al., *Domain-Specific Conceptual Modeling*, Springer, 2016, Chapter.
 - S. Lee, Y. Choe, M. Lee, dT-Calculus: A Formal Method To Specify Distributed Mobile Real-Time IoT Systems. Edited . Jaydip Sen, *The Internet of Things*, ISBN 978-953-51-5945-2, inTechOpen, 2018.
- **Journals**
 - S. Lee, J. Song, Y. Choe, M. Lee, ADONIS: A Service Design and Certification Tool for Certification of Software Development Process in International Standard Organization, Journal of SOSS, 2018.
 - J. Song, M. Ramahni, M. Lee, Journal of KIISE, Modeling and Composition Method of Collective Behavior of Interactive Systems for Knowledge Engineering, Vol. 44, No. 11, pp. 1178-1193, Nov. 2017. Y. Choe, S. Lee, M. Lee, Process Algebra to Model Timed Movements of Processes in Distributed Real-Time Systems, Transactions on Information Science and Applications, Volume 14, 2017, pp. 89-101.
 - Y. Choe, S. Lee, M. Lee, dT-Calculus: A Process Algebra to Model Timed Movements of Processes The IARAS International Journal of Computers, April, 2017.
 - S. Lee, Y. Choe, M. Lee, "A Dual Method to Model IoT Systems", International Journal of Mathematical Models and Methods in Applied Sciences, 2016.05.
 - M. Lee, "A Reengineering Method from Procedural SW to Object-Oriented SW for SaaS in Cloud Computing", International Journal on Recent an Innovation Trends in Computing and Communication, 2016.03.
 - Y. Choe, M. Lee, A Process Algebra Construct Method for Reduction of States in Reachability Graph: Conjunctive and Complement Choices, Journal of KIISE, Vol. 43, No. 5, pp. 541-552, 2016. 5.
 - Y. Choe, M. Lee, A Process Algebra for Modeling Secure Movements of Distributed Mobile Processes, Journal of KIISE, Vol. 43, No. 3, pp. 314-326, 2016. 3.
 - J. On, Y. Choe, M. Lee, A Meta-Modeling for Security to Detect Attach Behavior, Journal of KIISE, Vol. 41, No. 12, Dec 2014, pp. 1035-1049.
 - J. H. Choi, J. On, S. Woo, G. Park, M. Lee, A Method for Personal Font Construction Using Template, Journal of KIISE: Practice and Reality, Vol. 29, No. 3, April 2012, pp. 336-340.
- **Conferences**
 - M. Rahmani, J. Song, M. Lee, PRISM: A Knowledge Engineering Tool to Model Collective Behavior of Real-Time IoT Systems, PoEM2017/PrOs2017, Lueven, Belgium, Nov. 2017.
 - J. Song, M. Rahmani, M. Lee, Behavior Ontology to Model Collective Behavior of Emergency Medical Systems, ER2017/AHA2017, Valencia, Spain, Nov. 2017.
 - Y. Choe, M. Lee, A Modeling Method for Smart Mobile Service with IoT Environment, ICServ2017, Vienna, Austria, July 2017.
 - Y. Choe, S. Lee, M. Lee, "dT-Calculus: A Process Algebra to Model Timed Movements of Processes", 19th International Conference on MATHEMATICAL and COMPUTATIONAL METHODS in SCIENCE and ENGINEERING, March 2017.
 - Y. Choe, S. Lee, M. Lee, SAVE: An Environment for Visual Specification and Verification of IoT, EDOC2016/ModTools2016, Vienna, Austria, Sept. 2016.
 - M. Lee, Composition Model for Cloud Services with Behavior Ontology, ICServ2016, Tokyo, Japan, Sept. 2016.
 - S. Lee, Y. Choe, M. Lee, A Dual Method to Model IoT Systems, INASE AMCME 2016, May 2016, Riga, Latvia.
 - Y. Choe, M. Lee, δ-CALCULUS: PROCESS ALGEBRA TO MODEL SECURE MOVEMENTS OF DISTRIBUTED MOBILE PROCESSES IN REAL-TIME BUSINESS APPLICATION, 23rd European Conference on Information Systems, 2015.05. Muenster, Germany.
 - Y. Choi, M. Lee, et. al, A Tool for Visual Specification and Verification for Secure process Movements, eChallenge e-2015, Vilius, Lithuania, Nov. 2016. The Best Paper.
 - W. Choi, Y. Choe, M. Lee, A Reduction Method for Process and System Complexity with Conjunctive and Complement Choices in a Process Algebra, 39th COMPSAC/MVDA, July 2015. Taiwan.
 - Y. Choe, M. Lee, A Lattice Model to Verify Behavioral Equivalences, Proceedings of UKSim-AMSS 8th European Modelling Symposium, Oct 2014.
 - J. Choi, M. Lee, A Calculus for Transportation Systems, Proceedings of 38th Annual IEEE Computer Software and Applications Conference Workshops, July 2014.
 - M. Lee, Y. Choe, Modeling and Analysis of Equivalences in Behavior Ontology, Proceedings of 4th European Conference of Computer Science, Oct 2013.
 - J. On, Y. Choe, M. Lee, An Abstraction Method of Behaviors for Process Algebra, Proceedings of IEEE 37th Annual Computer Software and Applications Conference Workshops, July 2013.
 - S. Woo, J. On, M. Lee, Behavior Ontology: A Framework to Detect Attack Patterns for Security, Proceedings of 7thInternational Symposium on Security and Multimodality in Pervasive Environment (SMPE-2013), March 2013.
 - J. On, S. Woo, M. Lee, Onion: a Visual Formal Method for Workflow Design in Cloud Computing, Proceedings of MSV12, July 2012.
 - J. On, S. Woo, M. Lee, A Graphical Language to Integrate Process Algebra and State Machine Views for Specification and Verification of Distributed Real-Time Systems, Proceedings of 36th IEEE International Conference on Computer Software and Applications Workshops, July 2012, pp. 218-223.

References (1)

- [ALUR] ALUR, Rajeev; HENZINGER, Thomas A. Real-time logics: Complexity and expressiveness. *Information and Computation*, 1993, 104.1: 35-77.
- [ARV08] Arvidsson, F.; Flycht-Eriksson, A. "Ontologies I" (PDF). Retrieved 26 November 2008.
- [BEN] ABDALLAH, Hanene Ben. *Graphical Communicating Shared Resources: A Language for the Specification, Refinement and Analysis of Real-Time Systems*. 1996. PhD Thesis. University of Pennsylvania.
- [BROS] BROSGOL, Ben; COMAR, Cyrille. DO-178C: A New Standard for Software Safety Certification. 2010.
- [BERG] BERGSTRA, J. A.; KLOP, Jan Willem. Algebra of communicating processes. *Mathematics and Computer Science, CWI Monograph*, 1986, 1: 89-138.
- [CES816] .M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. On Programming Languages and Systems*, 8(2):244-263, 1986.
- [DILL] DILLER, Antoni. *Z: An introduction to formal methods*. John Wiley & Sons, Inc., 1990.
- [DOWS] DOWSON, Mark. The Ariane 5 software failure. *ACM SIGSOFT Software Engineering Notes*, 1997, 22.2: 84.
- [GHEZ] GHEZZI, Carlo; KEMMERER, Richard A. ASTRAL: an assertion language for specifying realtime systems. In: *ESEC'91*. Springer Berlin Heidelberg, 1991. p. 122-146.
- [GRU93] Gruber, Thomas R. (June 1993). "A translation approach to portable ontology specifications" (PDF). *Knowledge Acquisition* 5 (2): 199–220. doi:10.1006/knac.1993.1008.
- [HARE] HAREL, David. Statecharts: A visual formalism for complex systems. *Science of computer programming*, 1987, 8.3: 231-274.
- [HOAR] HOARE, Charles Antony Richard. Communicating sequential processes. *Communications of the ACM*, 1978, 21.8: 666-677.
- [ISL] LEE, Insup; DAVIDSON, Susan; GERBER, Richard. Communicating Shared Resources: A Paradigm for Integrating Real-Time Specification and Implementation. *Foundations of Real-Time Computing: Formal Specifications and Methods*, 1991, 87-109.
- [ISO] LOTOS, I. S. O. A formal description technique based on the temporal ordering of observational behaviour. *ISO8807, 1XS989*, 1989.
- [JAHA1] JAHANIAN, Farnam; MOK, Aloysius K.. Modechart: A specification language for real-time systems. *Software Engineering, IEEE Transactions on*, 1994, 20.12: 933-947.
- [JAHA2] JAHANIAN, Farnam; MOK, Aloysius K.. Safety analysis of timing properties in real-time systems. *Software Engineering, IEEE Transactions on*, 1986, 9: 890-904.
- [JM86] F. Jahanian and A.K. Mok. Safety analysis of timing properties in real-time systems. *IEEE Trans. Software Engineering*, SE-12(9):890-904, Sept. 1986.
- [JONE] JONES, M., et al. Introducing ECSS Software-Engineering Standards within ESA. *ESA bulletin*, 2002, 132-139.

References (2)

- [KAR] Dimitris Karagiannis, Agile Modeling Method Engineering, Proceedings of the 19th Panhellenic Conference on Informatics, Pages 5-10 , Athens, Greece — October 01 - 03, 2015.
- [KEHO] KEHOE, Raymond. *ISO 9000-3.: A tool for Software Product and Process Improvement. Edition en anglais*. Springer, 1996.
- [KUHN] KÜHN, H. The ADOxx® Metamodelling Platform. In: *Workshop on Methods as Plug-Ins for Meta-Modelling, Klagenfurt, Austria*. 2010.
- [NASA] NASA TECHNICAL STANDARD, "SOFTWARE ASSURANCE STANDARD", National Aeronautics and Space Administration, 2004
- [MILN] MILNER, Robin. *Communication and concurrency*. Prentice-Hall, Inc., 1989.
- [OVER] OVERMYER, S. P. DoD-Std-2167A and methodologies. *ACM SIGSOFT Software Engineering Notes*, 1990, 15.5: 50-59.
- [LEE1] Lee, M., *An Environment for Understanding of Real-time Systems*, Ph.D. Thesis, The University of Pennsylvania, 1995.
- [LEE2] Lee, M., Prywes, N., Lee I., "Automation of Analysis, Simulation and Understanding of Real-Time Large Ada Software," The First IEEE International Conference on Engineering of Complex Computer Systems, IEEE Press, 1995.
- [LEE3] CLARKE, Duncan; LEE, Insup; XIE, Hong-Liang. VERSA: A tool for the specification and analysis of resource-bound real-time systems. *Technical Reports (CIS)*, 1993, 510.
- [PETE] PETERSON, James L. Petri nets. *ACM Computing Surveys (CSUR)*, 1977, 9.3: 223-252.
- [RADA] RADATZ, Jane; OLSON, Myrna; CAMPBELL, Stuart. Mil-std-498. *Crosstalk, the Journal of Defense Software Engineering*, 1995, 8.2: 2-5.
- [[SHOC]] SHOCK, Sticker. Estimating the real cost of modern fighter aircraft. *Defense-aerospace*, 2006.(2006-7)[2011-4]. <Http://www.defense-aerospace.com>, 2006.
- [SING] SINGH, Raghu. International Standard ISO/IEC 12207 software life cycle processes. *Software Process Improvement and Practice*, 1996, 2.1: 35-50.
- [SPC] SOFTWARE PRODUCTIVITY CONSORTIUM SERVICES CORPORATION. *Reuse-driven Software Processes Guidebook: SPC-92019-CMC, Version 02.00. 03*. Software Productivity Consortium Services Corporation, 1993..
- [SPIV] SPIVEY, J. Michael; ABRIAL, J. R. *The Z notation*. Hemel Hempstead: Prentice Hall, 1992.
- [WANG] WANG, Jiacun. *Timed Petri nets: Theory and application*. Dordrecht: Kluwer Academic Publishers, 1998.