

Multi-agent Simulation Programming Environment with Error-pruning Support based on Typical Pattern Description

Masanori Akiyoshi, Kota Itakura, Norihisa Komoda
Osaka University, JAPAN

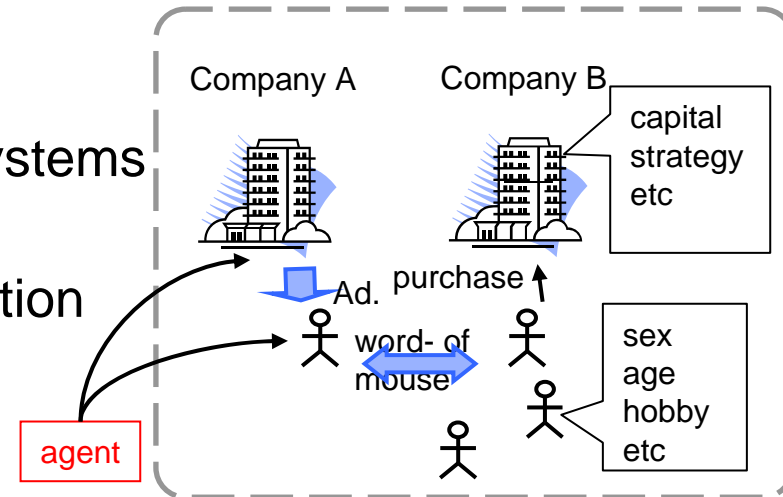
Contents

1. Research background
2. Outline of our research
3. Graph-based model editor
 - Event description
 - Event flow and event group
 - Source code generation
4. Model debugging
 - Error-inducing factors tree
 - Trimming the factors
5. Experiment
6. Conclusion

Research background

What is Multi-agent simulation (MAS) ?

- One of simulation methods for complex systems such as social system, economy system
- Behavior comes from autonomous interaction among elements



Design flow of MAS

Rough sketch by a designer

Model description

Consistent and understandable notation is not developed so far.

Program source code

Programming skill is necessary

Execution of simulation



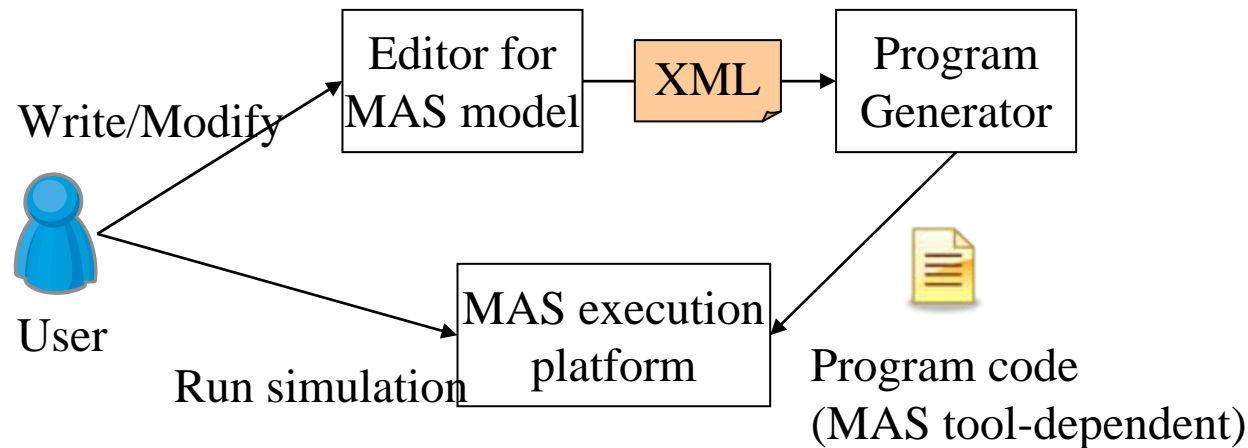
It takes a lot of time to execute MAS.

Outline of our research

Research purpose

- ✓Ease-of-use method for describing a target model
- ✓Reduce programming time

MAS development environment

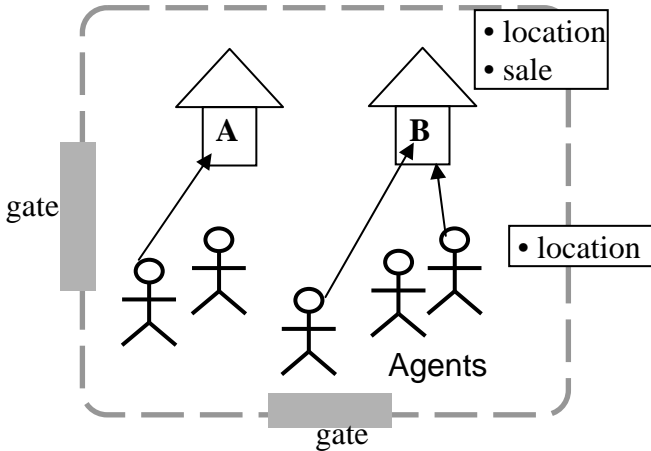


Problem to be tackled

- ✓How to reflect rough sketch of a user's MAS model based on the agent model description; **Attributes of an agent, Behavior of an agent, Mutual effects among agents**
- ✓How to debug the model description errors without source code debug

Graph-based model editor

Target Model (shopping mall)

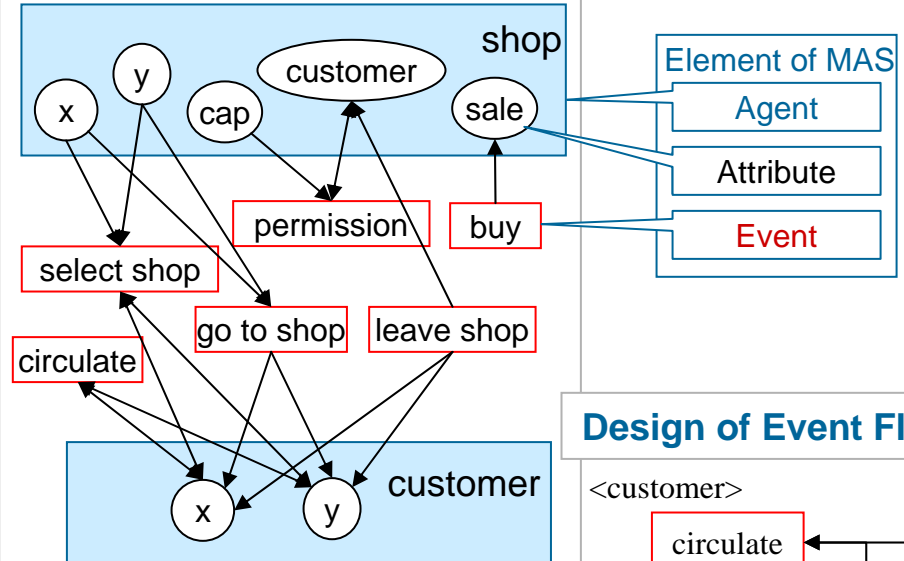


Distinctive feature of the notation

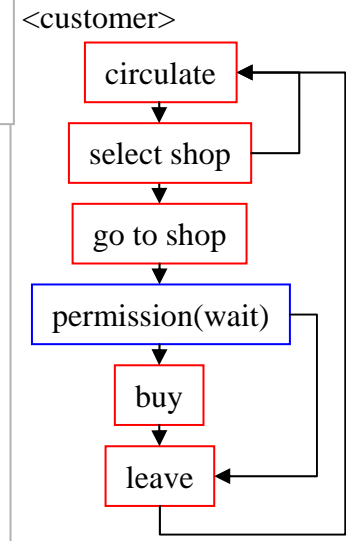
- ✓ Event description is based on typical action patterns of agents using **“interrogative (4W1H) and verbs”**.
- ✓ Mutual interactions are described with **“event flow diagram of agent events”**.

MAS Model Editor

Design of Model Elements



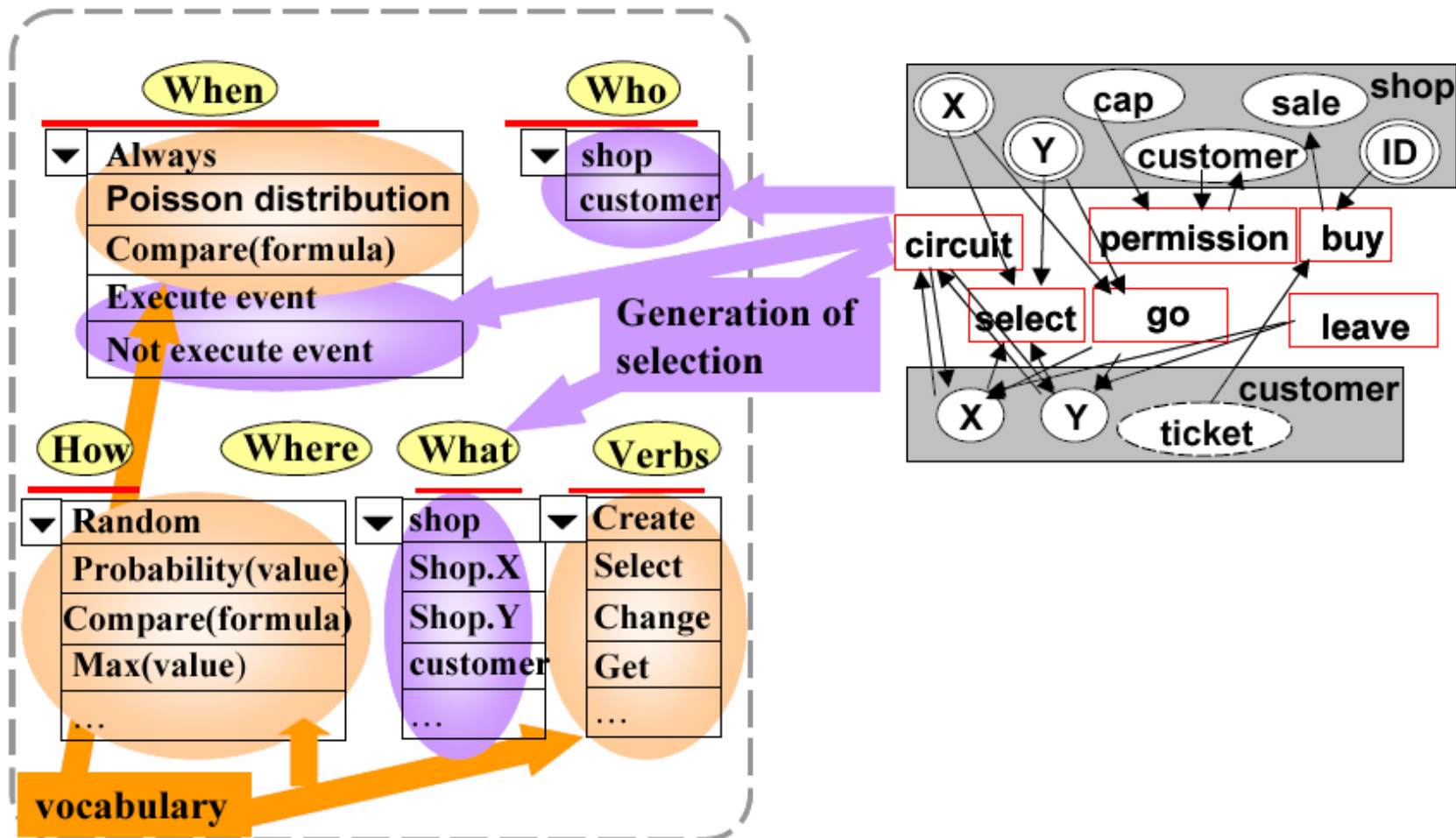
Design of Event Flow



Design of Event Contents

Event <buy>
When Always
Who shop
What my. sale
Verb Change
How Add(1)

Event description – typical action pattern



Without programming, it enables to design events along with narrative scenario of a user.

Event description – verbs and interrogates

verb

Create

agent creation

Delete

agent deletion

Make Decision

agent behavior firing/not-firing

Select One

interaction agent selection

Get

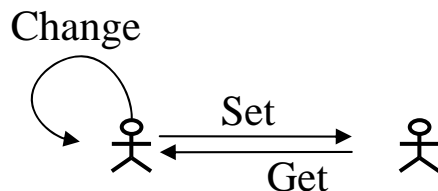
get attributes of an target agent

Set

set attributes of an target agent

Change

set own attributes



When

Always

Poisson distribution (λ)

interval based on poisson distribution

Compare(conditional description)

After acting

After not acting

Event description – interrogates

How

Probability(P)

act “verb” under P

Probability((P1,ID1) (P2,ID2)··)

act “Idxx action” under Pxx

Normal Add(value)

act “verb” after adding value

Int Random Add(minValue, max Value)

act “verb” after a random value
between minValue and max Value

Substitute(value)

act “verb” after substituting a value

Random Substitute(minValue, maxValue)

act “verb” after substituting random value
between minValue and max Value

Assign To(variableName)

assign acted results to variableName

Min(variableName)

act “verb” with minimum values of
variableName

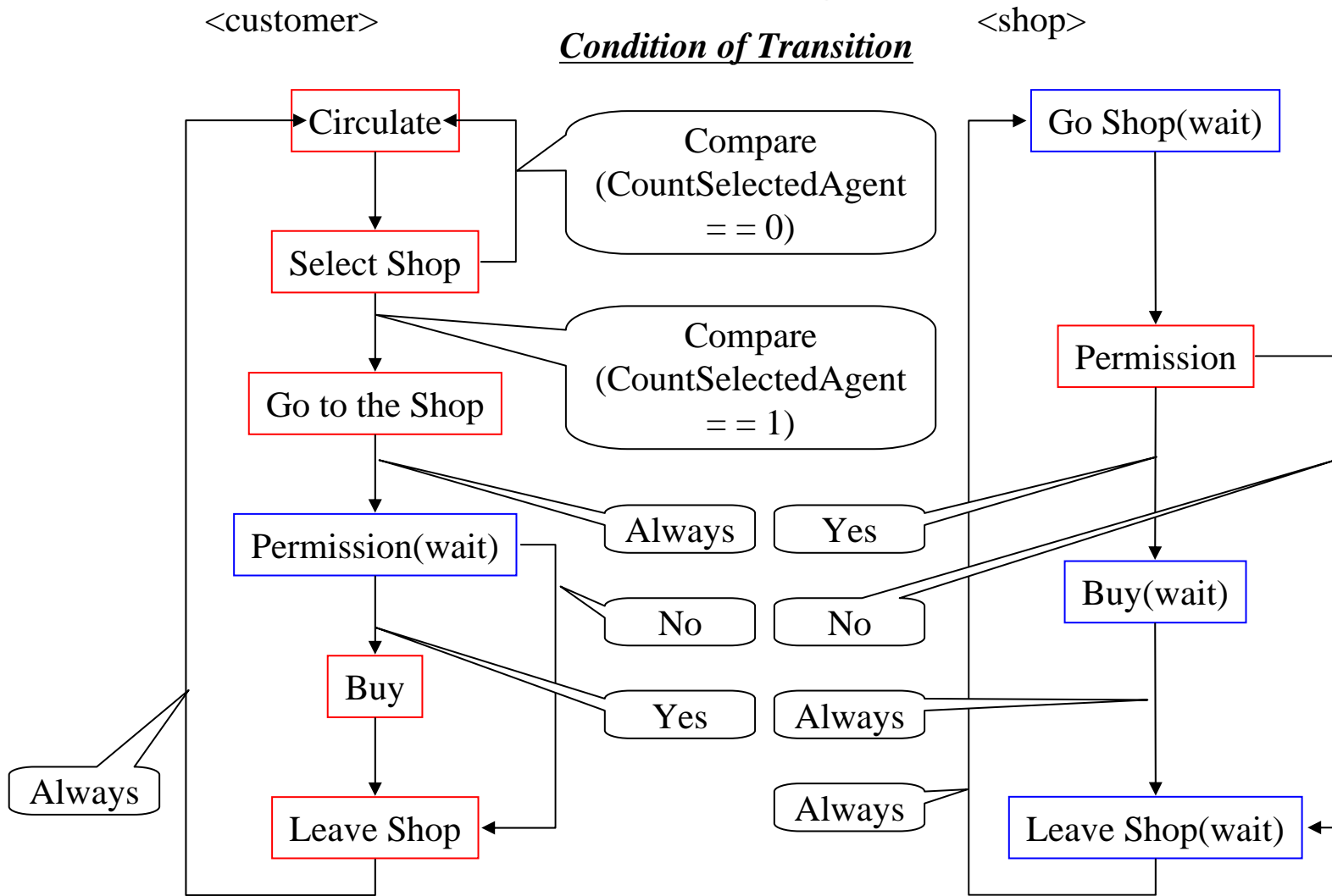
Max(variableName)

act “verb” with maximum values of
variableName

Compare(conditional description)

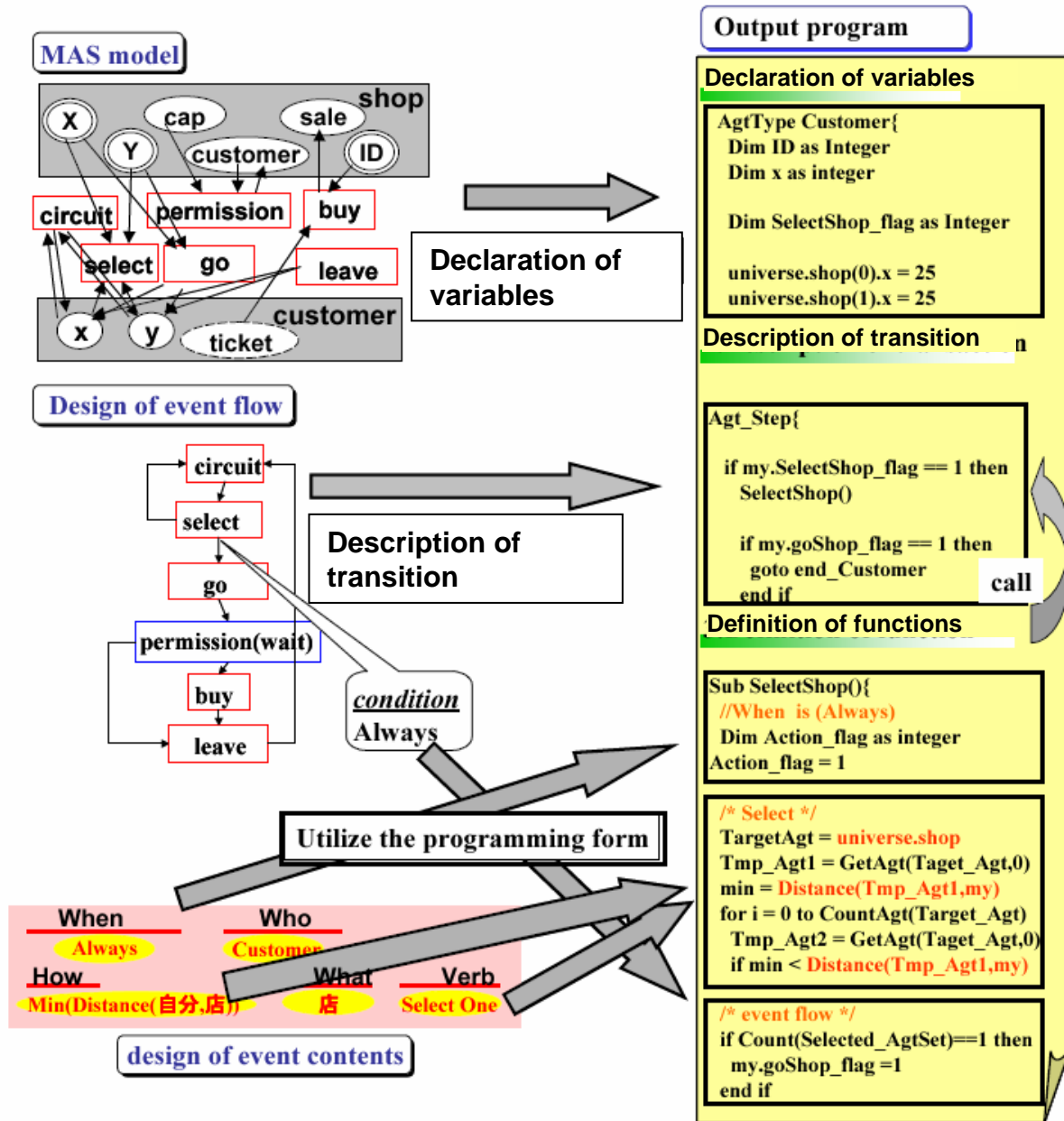
act “verb” if conditional description is true

Event flow and event group



“Event group” is designed as events are executed in the same simulation step. For instance, “select shop” and “go shop” are the same group.

Source code generation



Model debugging

What is model debug ?

- When detecting **unintended behavior** on execution, it is necessary to find causes in reverse manner and modify the model.

Examples of description error

- Mistake in the description of an event
- Loss of transition in an event flow
- Mistake of initial value setting

Propagation of effects

Finding of causes

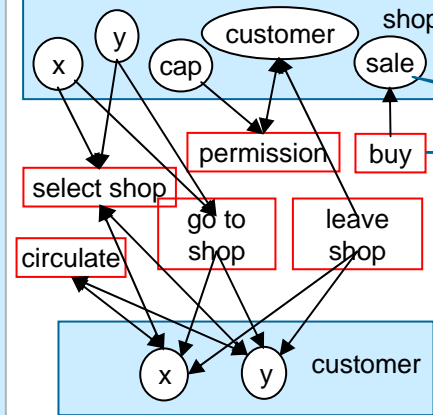
Unintended behavior

Why such findings are difficult ?

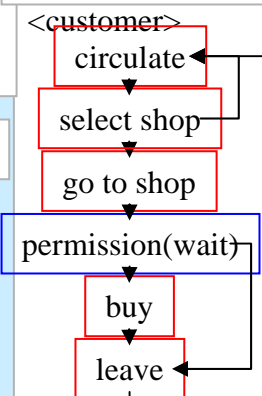
- MAS is not intrinsically traceable.
- MAS components are intertwined.

MAS Model Editor

Design of Model Elements



Design of Event Flow



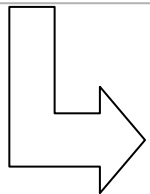
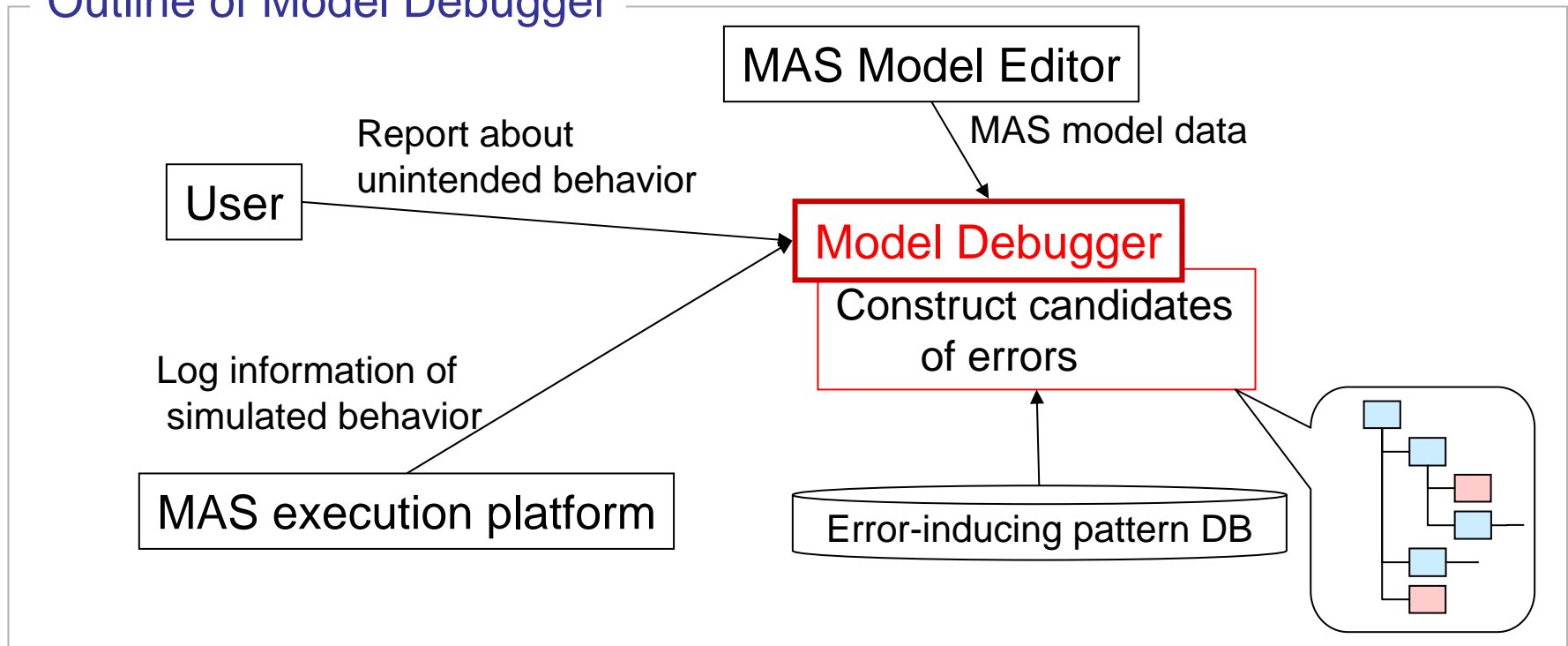
Design of Event Contents

Event <buy>
When Always
Who shop
What my. sale
Verb Change
How Add(1)

Model debugger with error-pruning

Candidates of errors are induced from unintended behavior.

Outline of Model Debugger

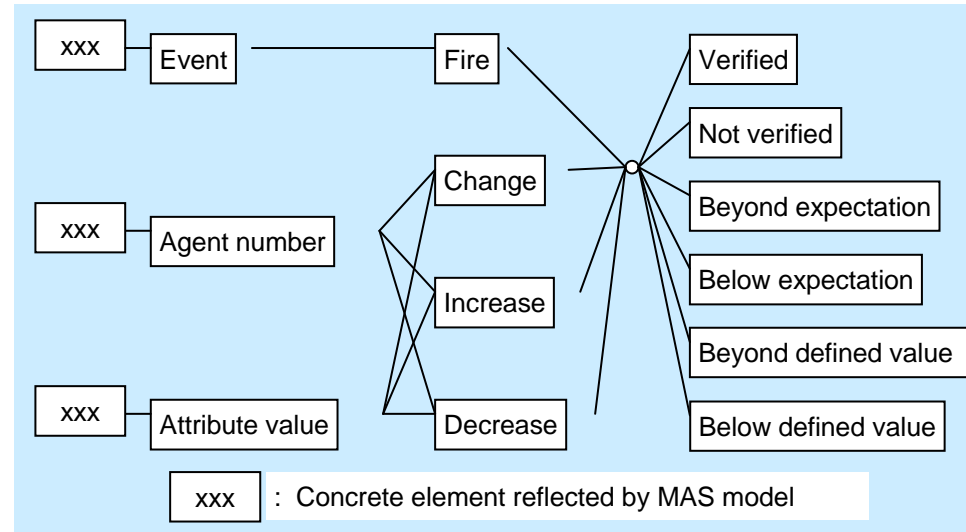


- **Report data format**
Typical report patterns for unintended behavior
- **Construction of error candidates**
Error-inducing factors tree and trimming the factors based on execution log

Error-inducing factors tree

Typical report patterns

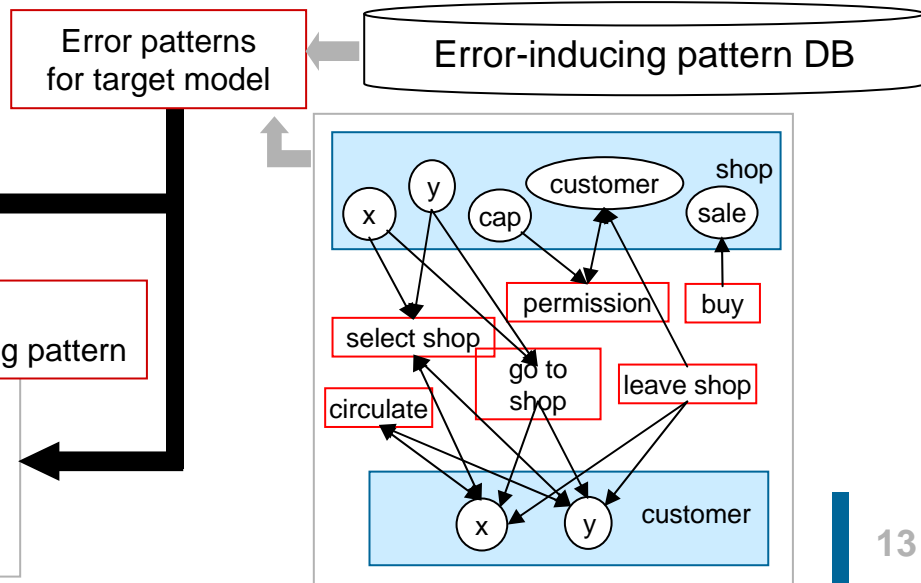
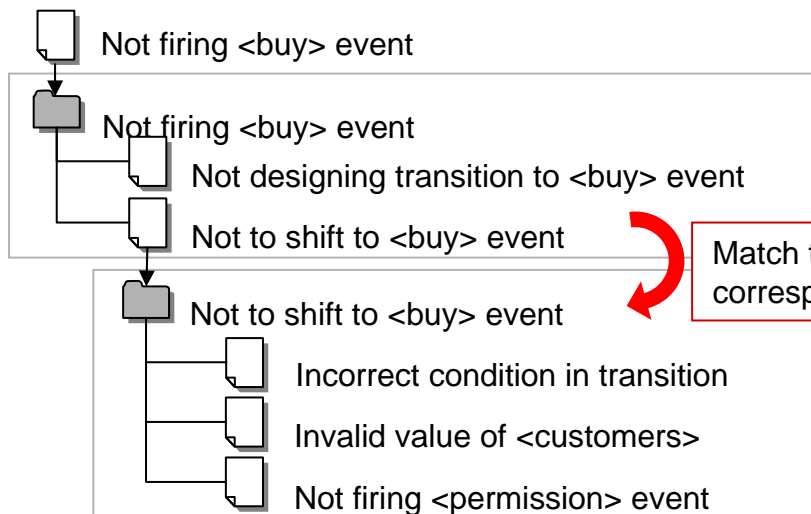
- Reports are done by report vocabulary related to event, agent and attributed of agents.



Generation of error-inducing factors tree

- Iterative matching with reports and error-inducing patterns

Report of unintended behavior



Trimming the factors

Tree elements are trimmed by using execution log such as **the number of event firing, the number of action firing, the attributes values** and so on.

The screenshot displays a software interface for error analysis, divided into three main sections:

- Error-inducing Factor Tree:** A hierarchical tree structure showing various error factors. The root node is "1:VarNodeBug". The tree includes nodes such as "[店] エージェントの [販売数] 変数が増えない", "[購買] イベントの Action0 が発火しない", and "[客数確認] イベントの Action0 が不正".
- Candidates of Error:** A list of error candidates, numbered 1 to 15. The list is filtered by "解析深度" (Analysis Depth) and "useFeedBack" / "withoutFeedBack". The selected candidate is "[客数確認] イベント Action0 の命令が不正".
- Contents of Selected Item:** A detailed view of the selected error candidate, titled "アクション命令の記述ミス" (Action Command Description Error). It shows the configuration for the "客数確認イベントアクションの編集" (Edit Customer Confirmation Event Action), including the condition "条件を満たしたとき (When)" set to "Always", the agent "エージェントが (Who)" set to "店", the rule "ルールに従って (How)" set to "Compare(自分の店内客数>自分の客数上限)", the target "対象を (What)" set to "購買させるか", and the action "何かする (verb)" set to "MakeDecision".

Experiment

Target model and scenario

- Customers arrive at the west gate or the south gate by some probability.
- Customers move in the shopping mall freely.
- Shop B passes coupons at the west gate and customers receive them at a constant rate.
- Customers go into the shop if distance between a customer and a shop is within a constant value.

.....

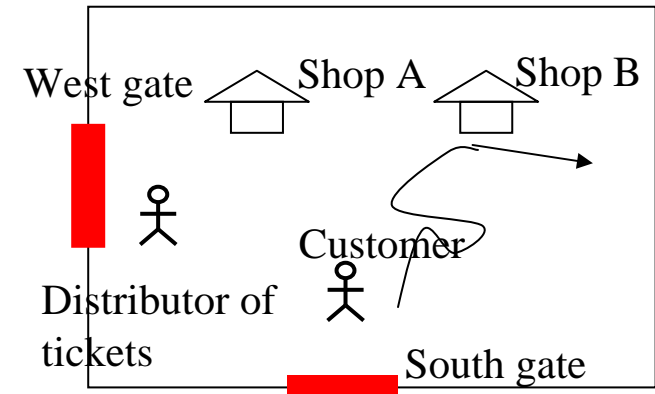
Model generation

A	Event description; artisoc(programming) and our description method (artisoc \Rightarrow our method)
B	Event description; artisoc(programming) and our description method (our method \Rightarrow artisoc)

Model debugging

3 errors are embedded in advance.

Shopping mall



Experimental results – model generation

examinee	artisoc		Our description method		
	Time (min)	Codes (line)	Time (min)	Pattern of action	Event flow
A	210	136	90	22	5
B	210	99*	100	21	5

* Examinee “B” cannot complete the programming codes.



It is effective to reduce the time for description.

In case of all the description of the target model (examinee “C”)

Agents	2 (shop, customer)
Num. of attributes	7 (shop), 9 (customer)
Num. of events	8
Num. of event flows	3
Description time	240 minutes



It is effective even if a user does not have any knowledge on programming.

Experimental results – model debug

	Contents of Error	Unintended behavior	With log	Without log
1	Reversed an inequality sign in condition	Event not firing	17	7
2	Loss of transition in an event flow	Agent not moving	25	6
3	Loss of an event description	Attribute not changed	15	8

(the number of candidate errors)

- Using log information helps to narrow the list **down to about 60%**.
- **4 minutes** to find errors assuming 30 seconds to check if a candidate is error, on the other hand debugging without our model debugger takes **40 minutes** to find errors.

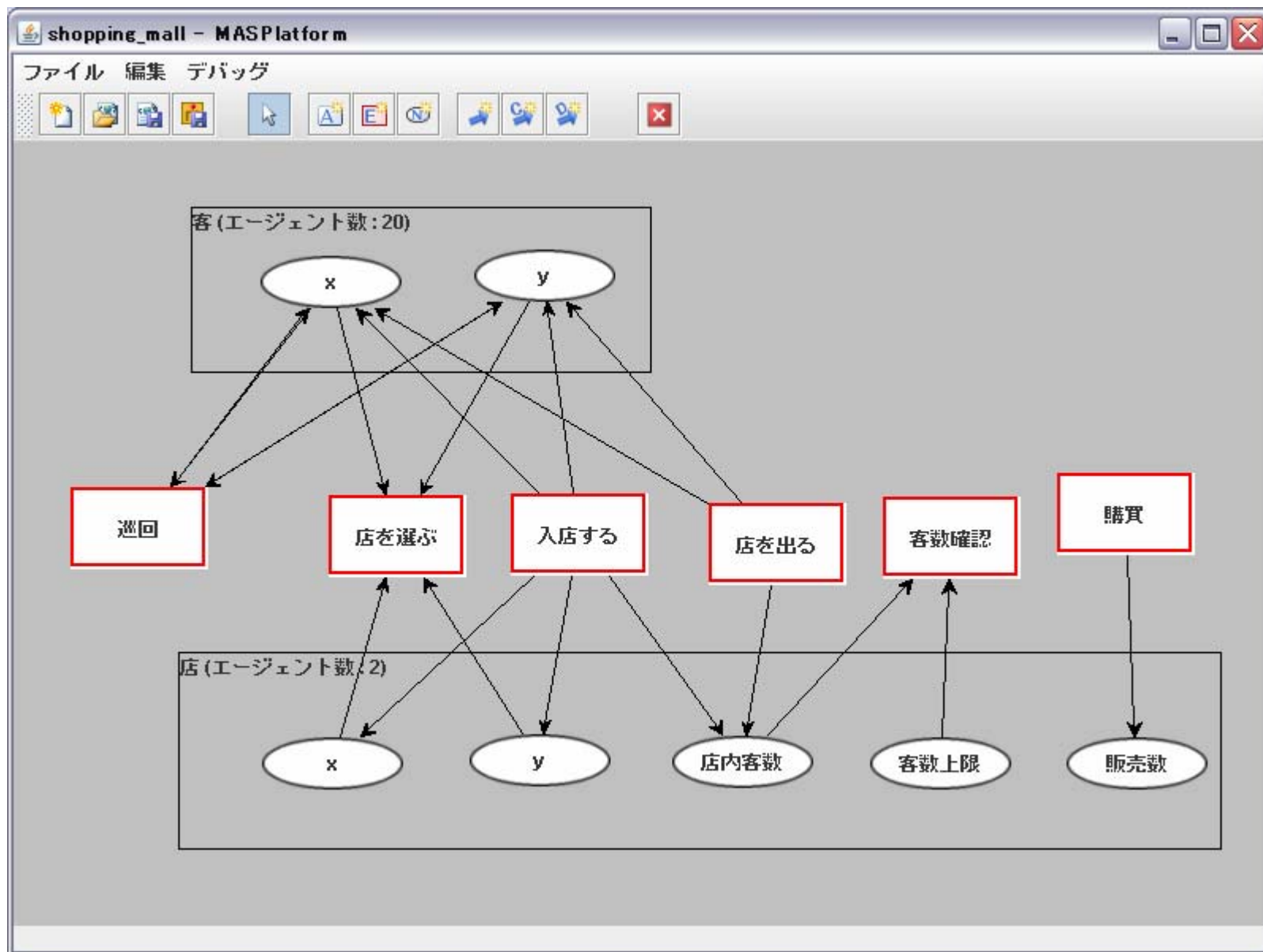


It is effective to reduce the time for debugging

Conclusion

- A description method for multi-agent simulation model utilizing typical action patterns of agents.
- A model debugger that generated error-inducing factors with typical report patterns of unintended behavior and execution log
- Through the experiments, our programming environment for MAS is effective to reduce the time compared to writing the source codes.

Screen shot – Model elements design



Screen shot – Typical report patterns

