Rapid Simulation Evaluation from Scenario Specifications for Command and Control Systems

Ray Paul

DoD OSD NII

Real-Time Distributed Network-Centric Warfare

- System modeling and simulation from the system requirement important for network-centric warfare
- System modeling and simulation are usually expensive for real-time distributed network-centric warfare
- Many specification and simulation packages are available such as SDL that can model and simulate the target system from requirements. But they are often design-oriented.
- Our scenario-driven system engineering approach provides a way to perform system modeling and simulation rapidly based on system requirements.

Comparisons Between SDL and Scenario ACDATE Model

Comparison		ACDATE	SDL/TTCN
Equivalence	Essence	High-level system description	
	Approach	Requirement engineering	
Difference	Fundamental Technique	Scenario oriented	Object oriented
	Intuitive Feature	More intuitive	Less intuitive
	Code Generation	Partial code	Complete code
	Components	Actor, Condition, Data, Action, Timing, Event	Structure, Communication, Behavior, Data, Inheritance
	Simulation	Non-real code based simulation	Real code based simulation
	Testing	Test cases generation	Test script generation
	UML Relation	Class diagram, Sequence diagram	UML compatible
	MDA Support	Unavailable	Available
	Goal	Ensure no errors in requirements	Generate real time applications
	V&V Support	Convenient to support V&V	Not focus on this

SDSE and Command & Control Systems

- Future Command and Control (C2) systems need to operate within an integrated grid-based network-centric environment (GIG) that allows rapid decision development and evaluation to meet the challenges of modern agile warfighting.
- A Scenario-Driven System Engineering (SDSE) approach is proposed to develop, evaluate, and test C2 systems
- Once system scenarios are specified, the system can be simulated without any programming and thus saves significant effort and time.

SDSE Features

- Compatible with the Service-Oriented Architecture (SOA) to develop trustworthy systems
- Can be used to specify and analyze system behaviors in an SOA.
- Core: scenario specification and analyses based on ACDATE model.

ACDATE Model

- Actors – An actor is either an external user, system or device, or an internal system, device, component or object;
- Conditions – A condition is a predicate used to trigger an action;
- Data – Attributes of actor, and presenting the semantic of condition, event and action
- Actions – Specified by the trigger event, guard condition, the way to change the status of actors, and sent event(s) to some actors
- Timing – A semantic statement about the relative or absolute value of time or duration
- Events – External/internal significant occurrences that may trigger action(s)

A Sample Scenario

```
using ACTOR:Alarm
using ACTOR:Horn
using ACTOR:DriverDoor
using ACTOR:PassengerDoor
using ACTOR:Trunk

if ( CONDITION:DriverDoor.DriverDoorIsLocked || CONDITION:PassengerDoor.PassengerDoorIsLocked )
then
{
    do ACTION:Alarm.TurnOnAlarm
    do ACTION:Horn.MakeHornBeepOnce(DATA:Horn.HornStatus)
}
else
{
    do ACTION:Horn.MakeHornBeepThreeTimes(DATA:Horn.HornStatus)
}
```

Actors

Condition

Data

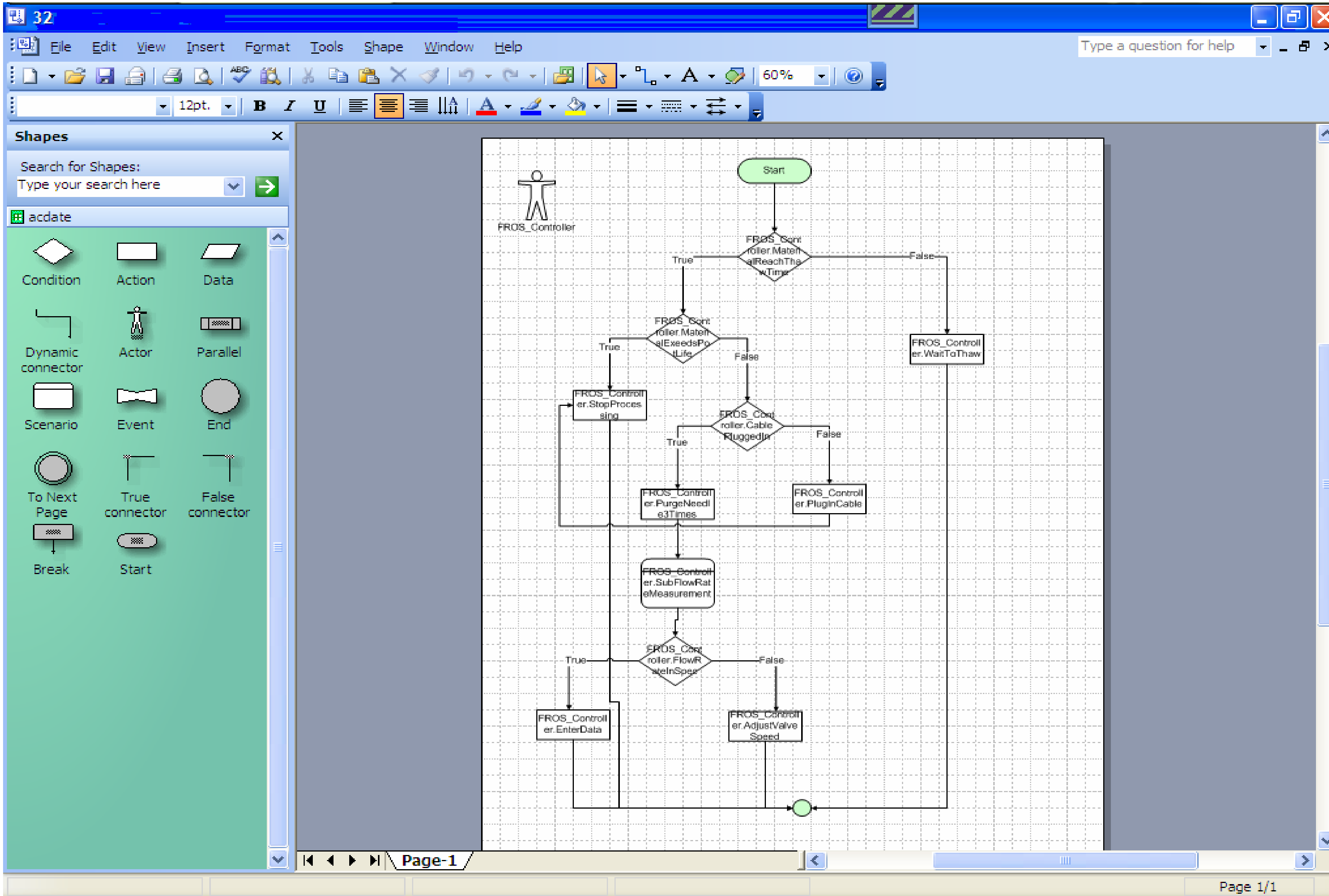
Action

A scenario “*when driver door is locked and passenger door is locked, if remote controlled is pressed unlock, then the driver door is open*” can be specified using Scenario Specification Language as above

Analyses based on ACDATE/Scenario Model

- Based on the ACDATE/Scenario model, a variety of static and dynamic (via simulation) analyses can be performed:
  - Completeness and consistency analysis
  - Performance evaluation
  - Safety analysis
  - Behavior analysis
  - Policy specification and enforcement
- The SDSE is to be integrated and supported by an automated tool E2E that is currently being used in several experimental projects by US Navy

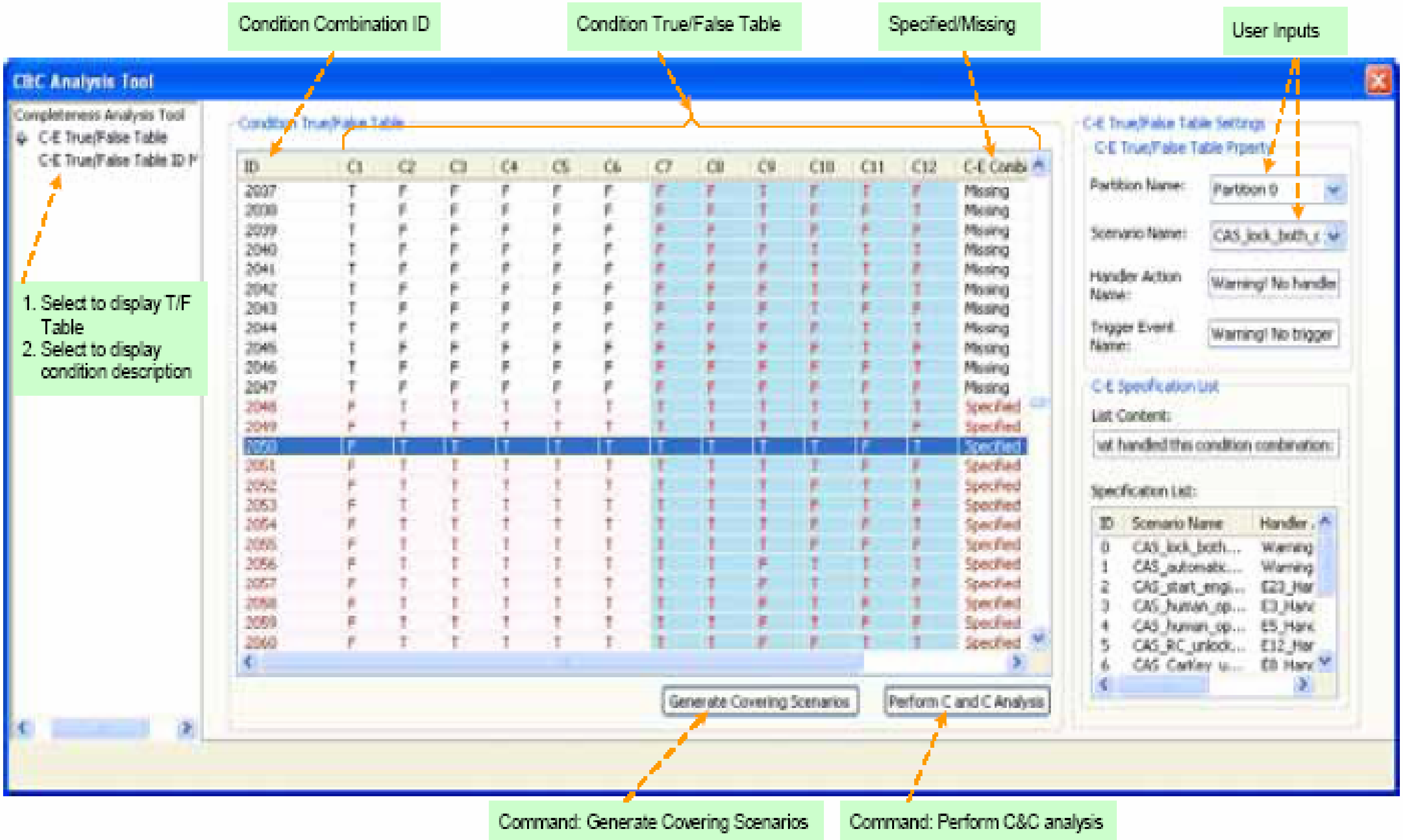
Scenario Tool Input Interface



Static C&C Analysis

- Once the system ACDATE/Scenario model is ready, one can easily using our automated tool to perform completeness and consistency analysis to see if there is any problem in system modeling
- Static C&C analysis can discover a large amount of incompleteness and inconsistency problems that are hard for engineers to detect

Static C&C Analysis Tool

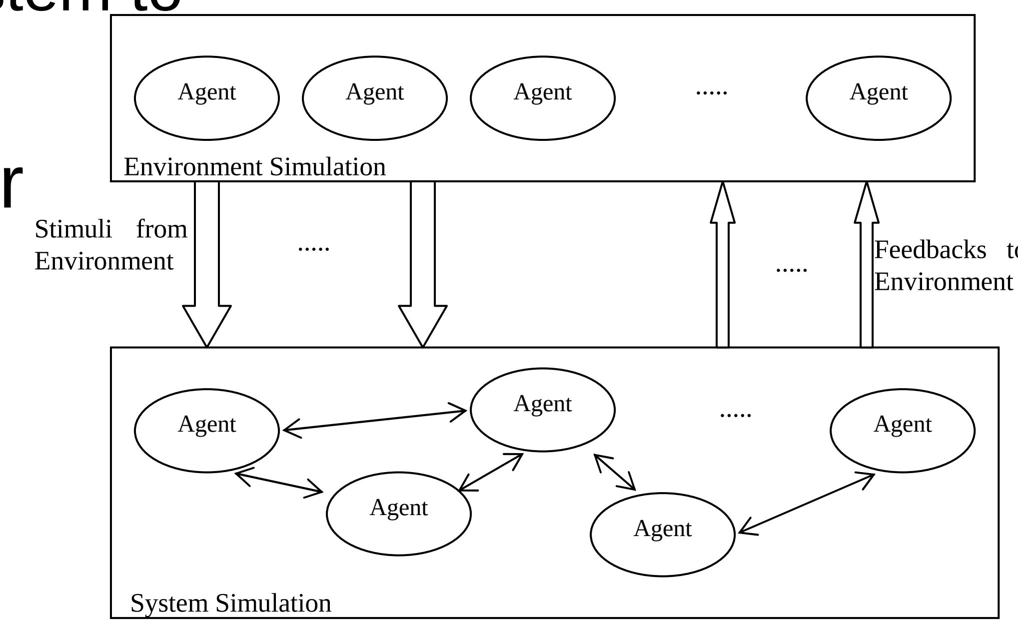


Experiment Results of Static C&C Results

Systems	Type	Experimental/ Industrial	# of Scenarios	# of Conds.	# of Missing C-E Combination	# of Covering Scenarios	Whole/ Partial System
Car Alarm Systems	Centralized application.	Experimental	13	12	547	40	Whole
Banking System	Distributed application		23	2	0	0	Whole
ICS	Distributed real-time application.	Industrial	140	15	396800	29	Whole
RCS	Distributed real-time application.		54	10	1792	4	Partial
LDRS	Distributed real-time application.		10	19	1966080	8	Whole
Communication Processor	Embedded distributed application		31	33	$1.29 * 10^{11}$	27	Partial

Scenario-Based Simulation Architecture

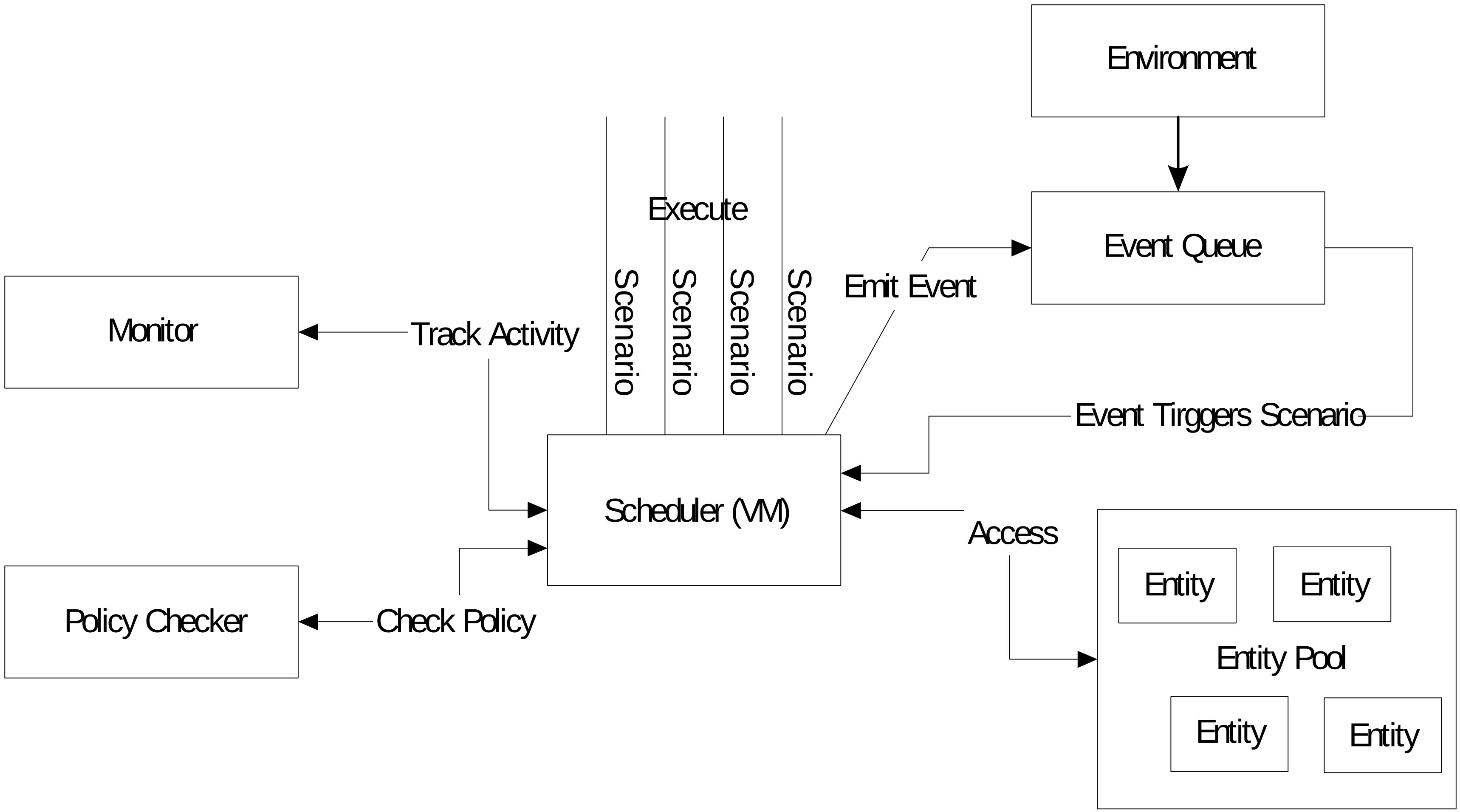
- Scenario-based simulation is divided into two major parts
  - Environment Simulator
    - The behavior of the environment
    - Impact of the system to its environment
  - System Simulator
    - Target system behavior



Rationale for Separating Environment And System Simulation

- It offers the opportunity to observe the behavior of the *system under testing* (SUT) under different loads by varying the environment simulator
- Robustness, reliability and scalability of the system can be determined by generating various inputs to drive the system
  - Generating an incorrect input can evaluate the system's robustness
  - Generating the input according to the operational profile will determine the system's reliability
  - Generating inputs of various sizes to determine the system scalability

Simulation Engine Architecture



Simulation Code Generation

- The simulation code is generated based on the scenario specification, which includes the ACDATE definition and scenario description
- Each ACDATE model element will be translated to an object with the attributes defined in the specification
- Instrumentation code will be inserted to the objects to interface with the monitor and policy checker
- Each scenario will be translated to a procedure that is basically a sequence of operations on the ACDATE objects or emitting events.
- With these automated simulation code generation, previously Excel-spreadsheet based real-time distributed network-centric C2 systems can be simulated without any additional programming effort saving significant effort.

Simulation Code Generation – Sample Scenario

```
using ACTOR:Alarm
using ACTOR:Horn
using ACTOR:DriverDoor
using ACTOR:PassengerDoor
using ACTOR:Trunk

if ( CONDITION:DriverDoor.DriverDoorIsLocked || CONDITION:PassengerDoor.PassengerDoorIsLocked )
then
{
    do ACTION:Alarm.TurnOnAlarm
    do ACTION:Horn.MakeHornBeepOnce(DATA:Horn.HornStatus)
}
else
{
    do ACTION:Horn.MakeHornBeepThreeTimes(DATA:Horn.HornStatus)
}
```

Actors

Condition

Data

Action

A scenario “*when driver door is locked and passenger door is locked, if remote controlled is pressed unlock, then the driver door is open*” can be specified using Scenario Specification Language as above

Simulation Code Generation Example – Generated Code

```
scenario_5 = function(co_routine_name, platform)  // a scenario
coroutine.yield();                                // interface to scheduler ...
if (condition_11.Eval() || condition_17.Eval() ) //condition evaluation
then
    action_10:before_do(co_routine_name, platform);  // interface to policy
                                                      //checker embedded here
    action_10_dummy_func(co_routine_name, platform);  // turn on alarm
     action_10:after_do(co_routine_name, platform);
     timer[platform] = timer[platform] + unit;       // advance and record system
                                                      // time ...
     action_17: action_10:before_do(co_routine_name, platform);
     action_17_dummy_func(co_routine_name, platform);   // beep once
     action_17:after_do(co_routine_name, platform);
     timer[platform] = timer[platform] + unit;
else
    action_20: action_10:before_do(co_routine_name, platform);
     action_20_dummy_func(co_routine_name, platform);   // beep three times
     action_20:after_do(co_routine_name, platform);
     timer[platform] = timer[platform] + unit;
end
```

Sample Simulation Result

Execution Sequence

Event Handlers Invoked

Event Being Processed

Event Generated

…

[30] Event E1_ReceiveFireOrder is Generated.

[31] Event E13_ReceiveFireOrder with Event Instance ID of 12 Arriving at 30

Timing Information

[31] Begin Doing Action Action13_ReceiveFireOrder

[32] Condition Vig3_ChancellorsvilleCanShoot is Evaluated to Be False

[33] Begin Doing Action Action3_MakeDecision

[34] Condition Conditon27_DecideRejectMission is Evaluated to Be True

…

Condition Evaluated

Action Performed

Dynamic Analyses Performed Based on Simulation

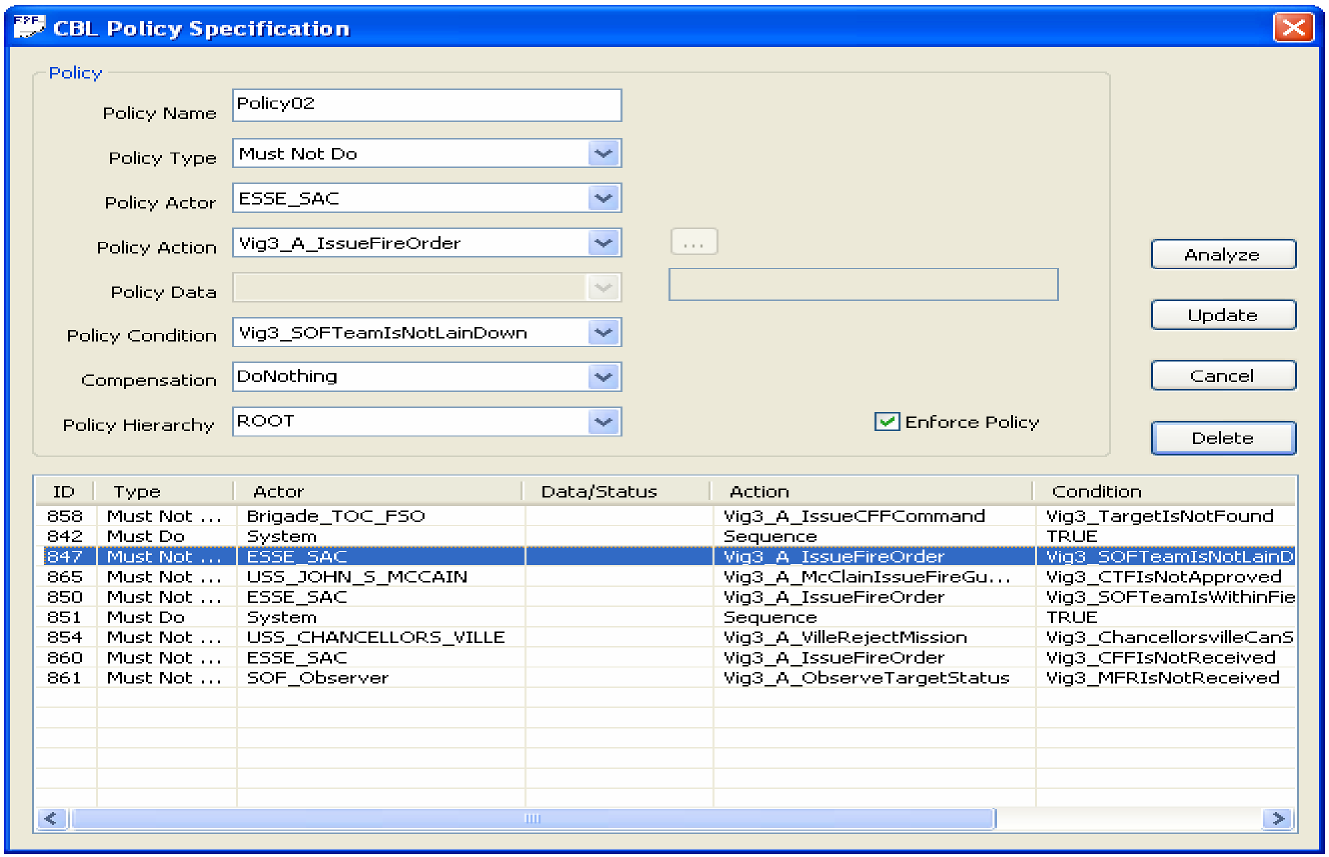
- Once simulation is performed, variety kinds of analyses can be carried out based on simulation, both runtime and off-line:
  – Policy specification and enforcement
  – Dynamic C&C analysis
  – Performance analysis
  – Safety analysis
  – Behavior analysis

Policy Specification and Enforcement

- One can specify kinds of policies in the system ACDATE/Scenario model using our automated scenario tool
- Once policies are specified, simulation can dynamically check and enforce the specified policies at runtime.
- Any policy violation will be reported and recorded in a log file.

Policy Specification

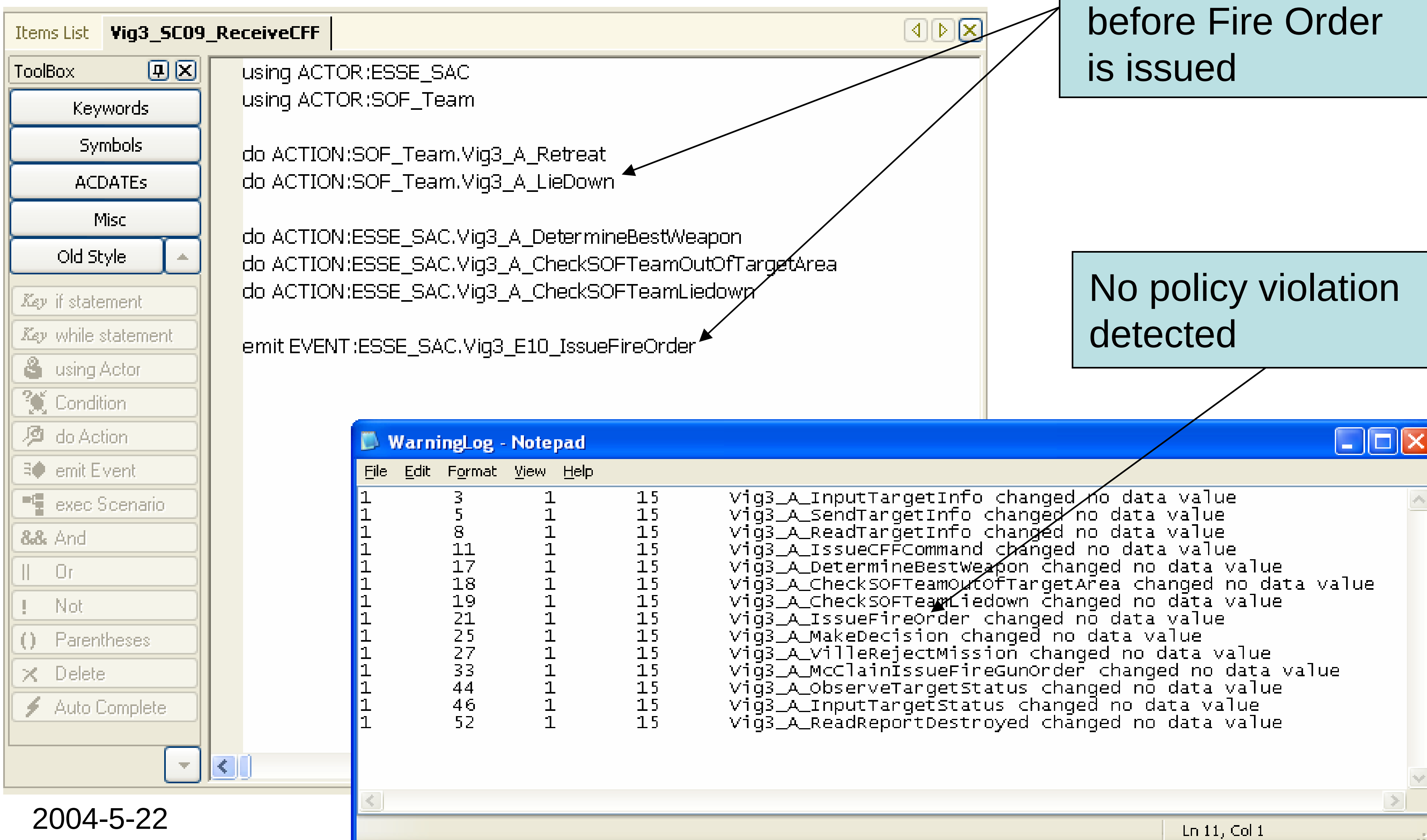
- Policy 2: Supporting Arms Coordinator (SAC) must NOT issue a Fire Order if SOF Team has not laid down
- Specification



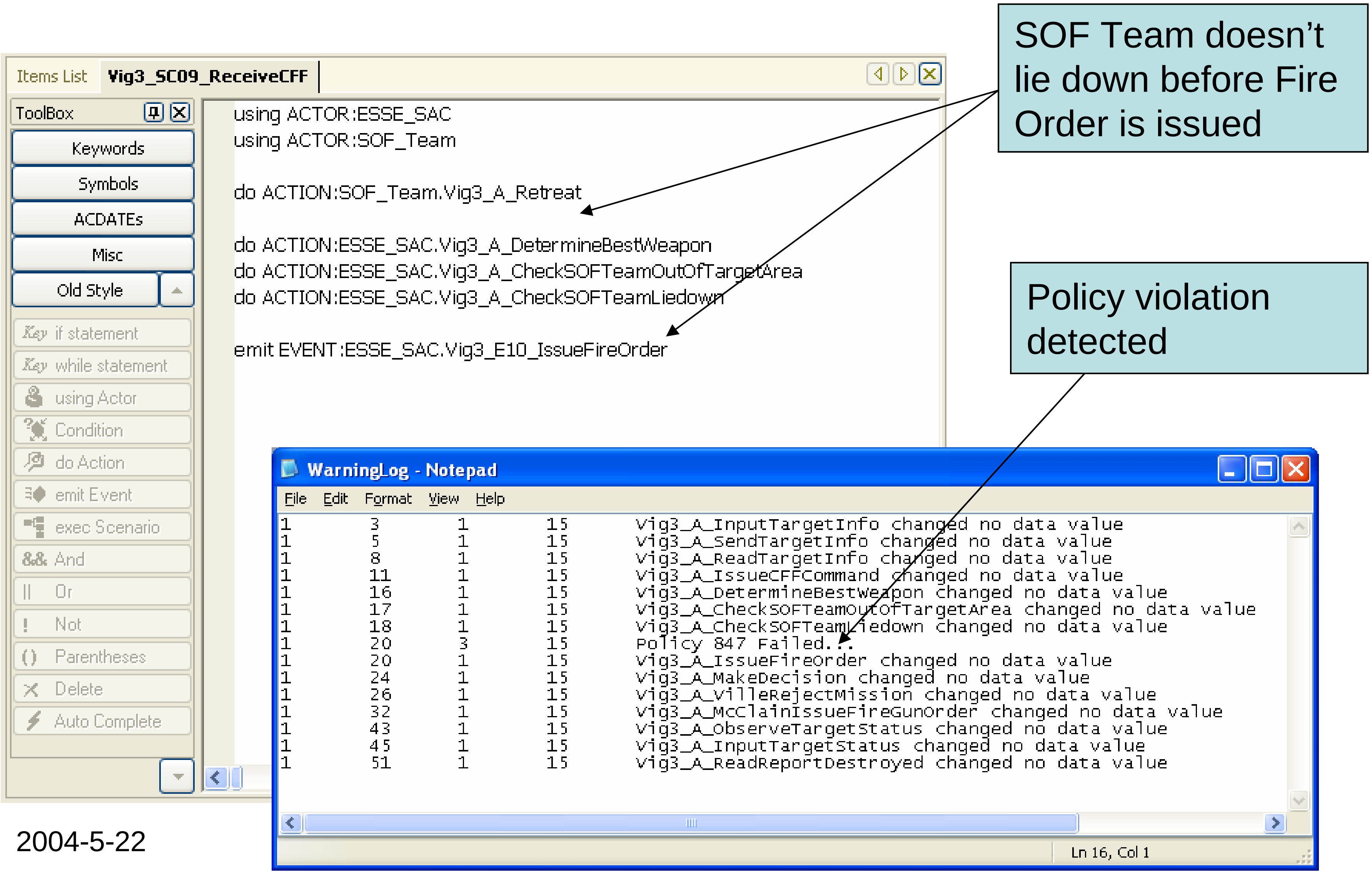
Acceptable Scenario

SOF Team lies down before Fire Order is issued

No policy violation detected



Scenario Violating Policy



Dynamic C&C analysis

- Some incompleteness or inconsistency can only be observed during runtime when concurrency comes into play
- In a recent experiment with a real time network-centric C2 system that has around 1000 entities and 120 scenarios, it takes 3 minutes to generate and execute the simulation and it detected around 200 bugs related to incompleteness or inconsistency.

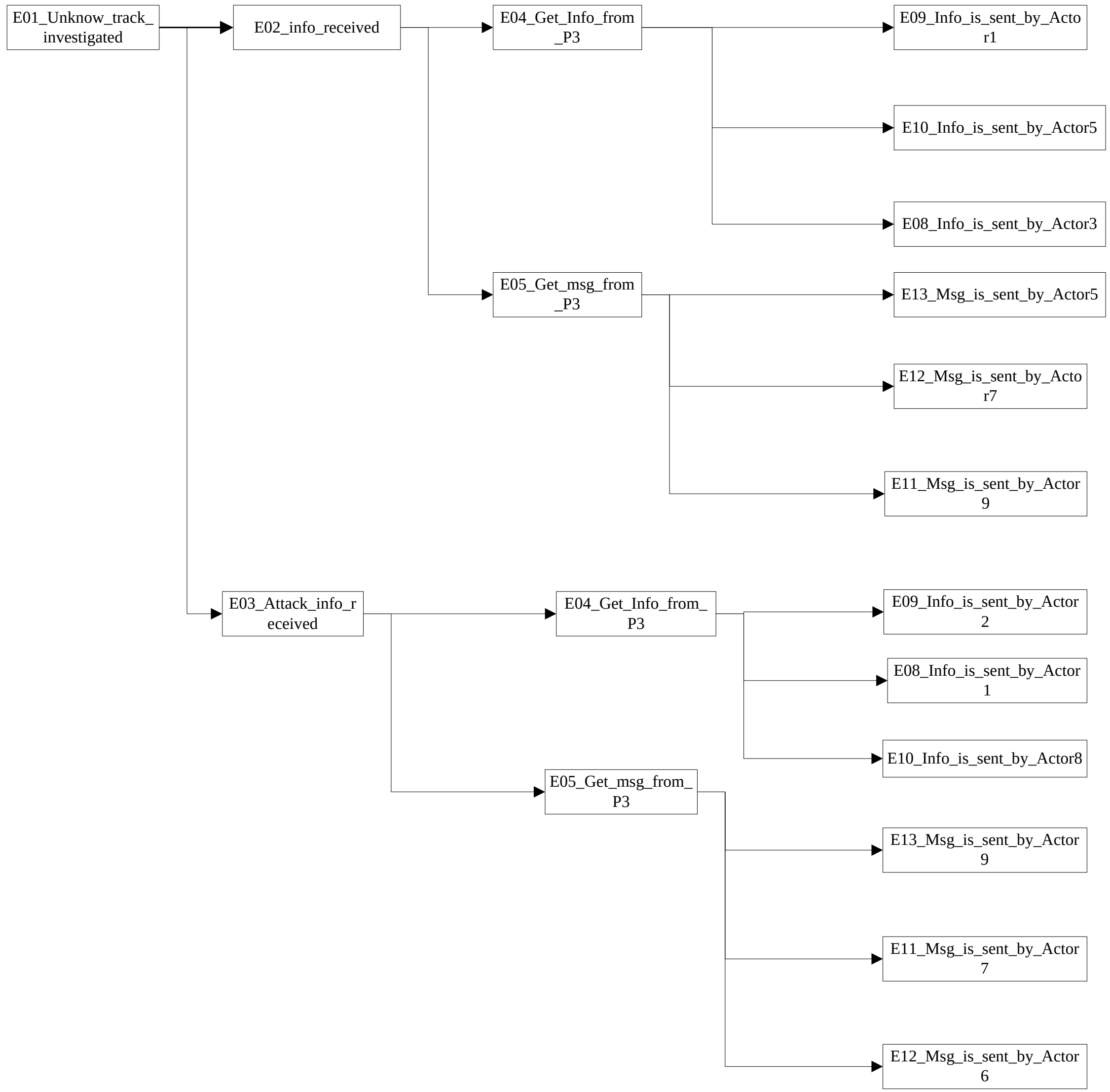
Performance Analysis

- During the simulation, system time will be recorded for each action when it starts or ends, event when it is emitted or handled, and data when the value changes.
- The recorded time information can be used for performance analysis such as the throughput and delay of the system processes.

Safety Analysis

- Event sequence can be useful for safety analysis.
- By using event sequence tree and traditional event tree, one can pinpoint which system components that failed during failure analysis.

Sample Event Sequence Tree



Behavior analysis

- Reachability analysis
- State model generation
- Linear Temporal Logic (LTL) analysis
- Model checking using SPIN
- Sequence diagram generation

State Model Generation

- We can generate the state model from the result of simulation
- Information contained in one entry of the simulation result
  - Time stamp
  - Starting and ending system state
  - Action performed
  - Event that triggers the action

State Model Generation

- From information provided in the result of simulation, we can get the global state transitions and single actor state transitions
- But there is something more for the single actor state transition generation – guard condition, i.e. some state transition can happen when some other actors are in certain conditions. For example, alarm can not be turned on when the driver's door is open.
- State transition for global system:
  - (starting global state) – external event/triggered actions → (ending global state)
- State transition for single actor
  - (starting actor state) – external event[guard condition]/triggered actions → (ending actor state)

Model Checking with SPIN

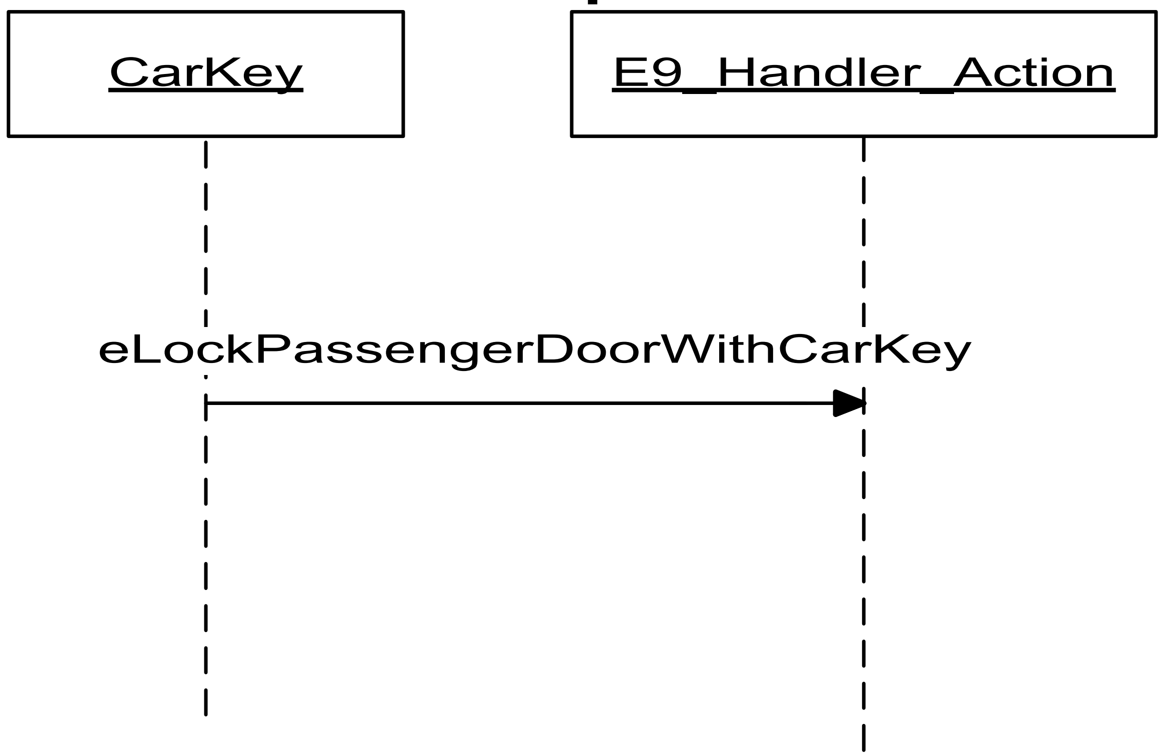
- One can generate the single actor state models automatically from simulation
- One can also generate the single actor state models from requirements or design manually
- Two sets of state models can put into SPIN to perform cross checking

Sample State Model

	A	B	C	D	E	F	G
1	StartState	Event	GuardCondition	Actions	EndState		
2	Door Closed Not Loc	eLock/ArmButtonPressedWRem	Door Closed Not Locked	(Lock driver door) (Lock pa	Door Closed And Locked		
3	Door Closed And Loc	eLockPassengerDoorWKey	No Guard	No Action	Door Closed And Locked		
4	Door Closed And Loc	eLock/ArmButtonPressedWRem	Door Closed And Locked	(Beep Three Times)	Door Closed And Locked		
5	Door Closed And Loc	eLock/ArmButtonPressedWRem	Door Closed And Locked	(Beep Three Times)	Door Closed And Locked		
6	Door Closed And Loc	eLock/ArmButtonPressedWRem	Door Closed And Locked	(Beep Three Times)	Door Closed And Locked		
7	Door Closed And Loc	eLock/ArmButtonPressedWRem	Door Closed And Locked	(Beep Three Times)	Door Closed And Locked		
8	Door Closed And Loc	eLock/ArmButtonPressedWRem	Door Closed And Locked	(Beep Three Times)	Door Closed And Locked		

This is a sample state model for the Driver's Door

Generated Sequence Diagram Sample



Conclusion

- A systematic process to perform variety kinds of static and dynamic analyses based on scenario specification
- Once system scenarios are specified, the simulation code can be automatically generated, and the system can be simulated without any additional programming
- The simulation can be used to perform various dynamic analyses including C&C checking, safety analysis, and performance analysis. The SDSE is being integrated into an automated tool E2E.