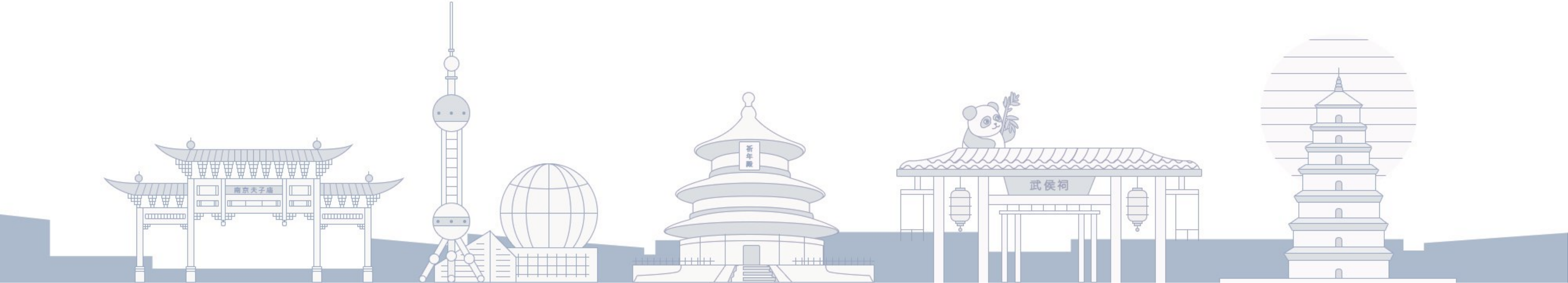
Glaway soft®

南京国睿信维软件有限公司

智慧企业解决方案及自主工业软件提供商

Model Execution And System Simulation In Capella

Renfei Xu & Wenhua Fang



Contents

01 Background and Motivation

02 Execution and Simulation Rules

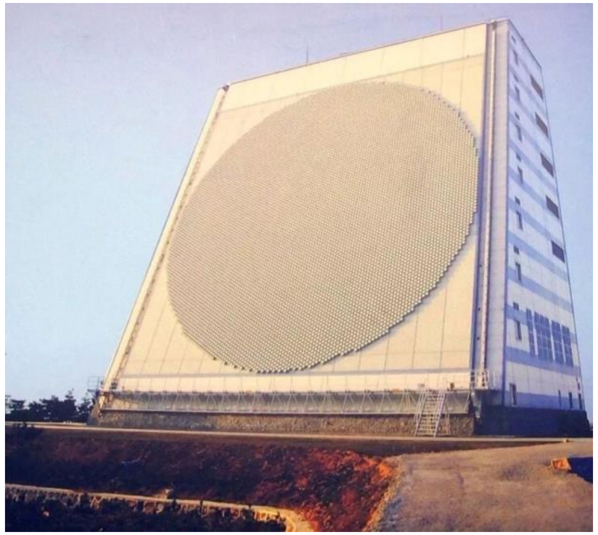
03 Example

01

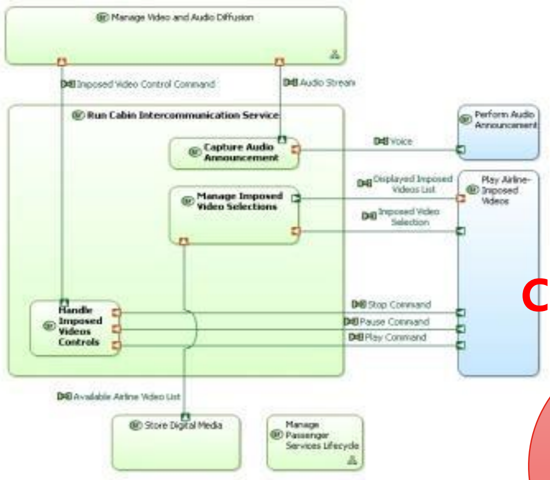
Background and Motivation

Background

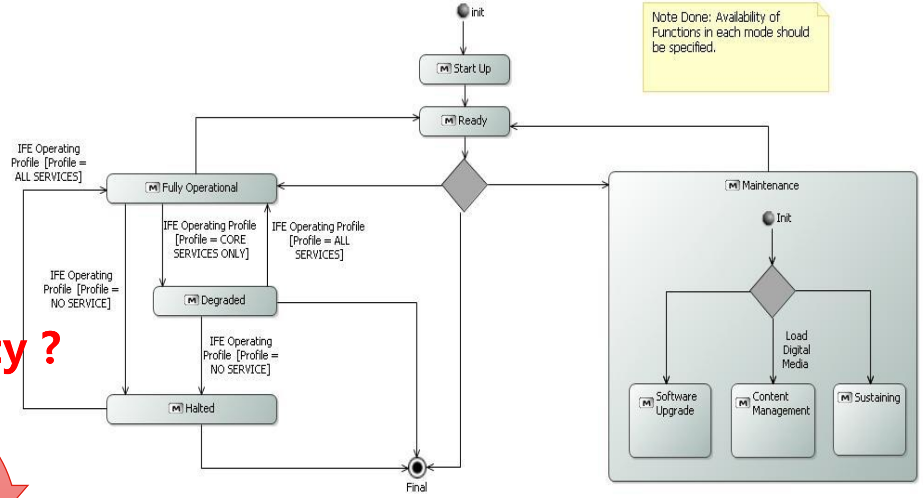
- Glaway has been deploying Capella-based MBSE solution in a Radar institute since late 2016
- The Radar products of this institute vary from seeker, spaceborne, airborne, shipborne to ground-based
- By now, each area in this institute has at least one product using Capella in its architecture design, including the most complex product



Motivation

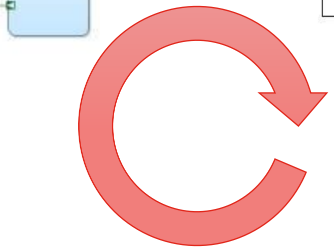


Manage Video and Audio Diffusion
Imposed Video Control Command
Audio Stream
Run Cabin Intercommunication Service
Capture Audio Announcement
Voice
Perform Audio Announcement
Displayed Imposed Videos List
Play Airline-Imposed Videos
Manage Imposed Video Selections
Imposed Video Selection
Handle Imposed Videos Controls
Stop Command
Pause Command
Play Command
Available Airline Video List
Store Digital Media
Manage Passenger Services Lifecycle

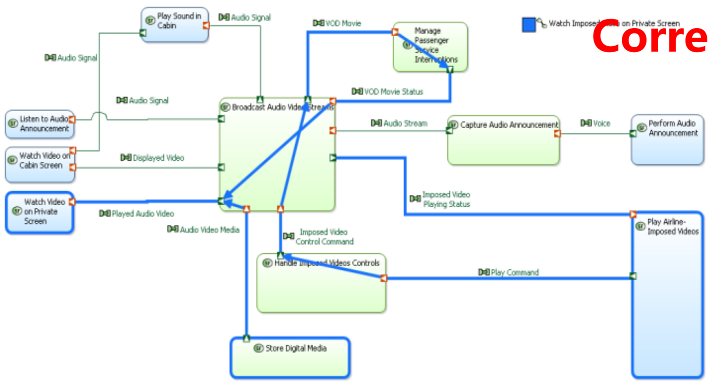


init
Note Done: Availability of Functions in each mode should be specified.
Start Up
Ready
IFE Operating Profile [Profile = ALL SERVICES]
Fully Operational
IFE Operating Profile [Profile = CORE SERVICES ONLY]
IFE Operating Profile [Profile = ALL SERVICES]
IFE Operating Profile [Profile = NO SERVICE]
Degraded
IFE Operating Profile [Profile = NO SERVICE]
Halted
Final
Maintenance
Init
Load Digital Media
Software Upgrade
Content Management
Sustaining

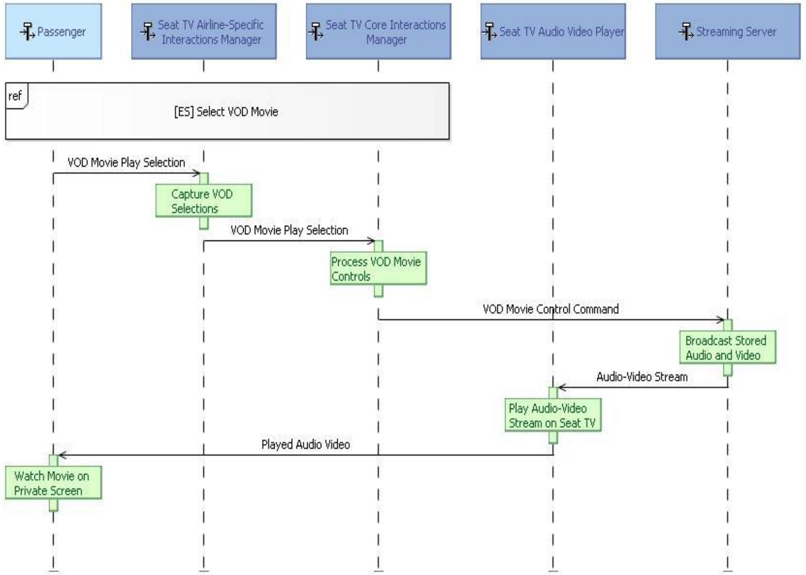
Consistency ?



Correctness ?



Play Sound in Cabin
Audio Signal
VOD Movie
Manage Passenger Service Interactions
Watch Imposed ... on Private Screen
Audio Signal
Audio Signal
VOD Movie Status
Listen to Audio Announcement
Broadcast Audio Video Stream
Audio Stream
Capture Audio Announcement
Voice
Perform Audio Announcement
Watch Video on Cabin Screen
Displayed Video
Imposed Video Playing Status
Watch Video on Private Screen
Played Audio Video
Play Airline-Imposed Videos
Audio Video Media
Imposed Video Control Command
Handle Imposed Videos Controls
Play Command
Store Digital Media

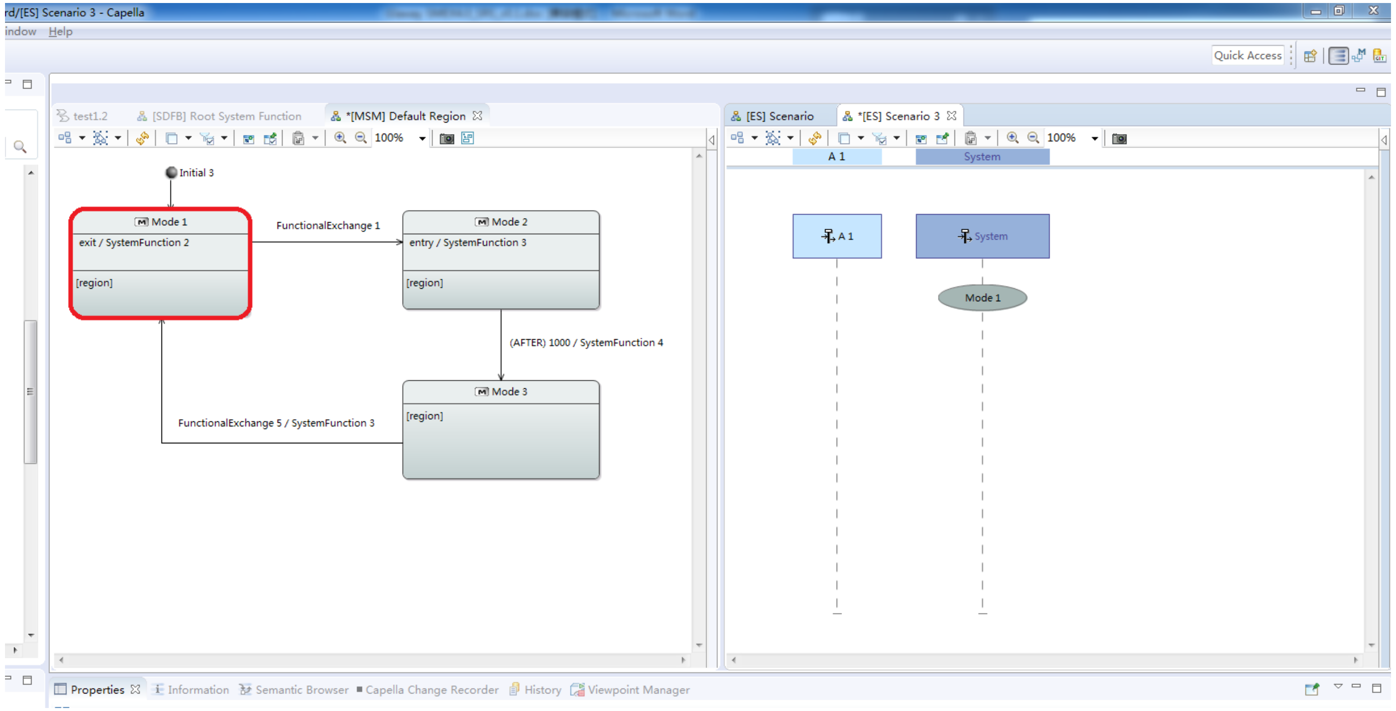


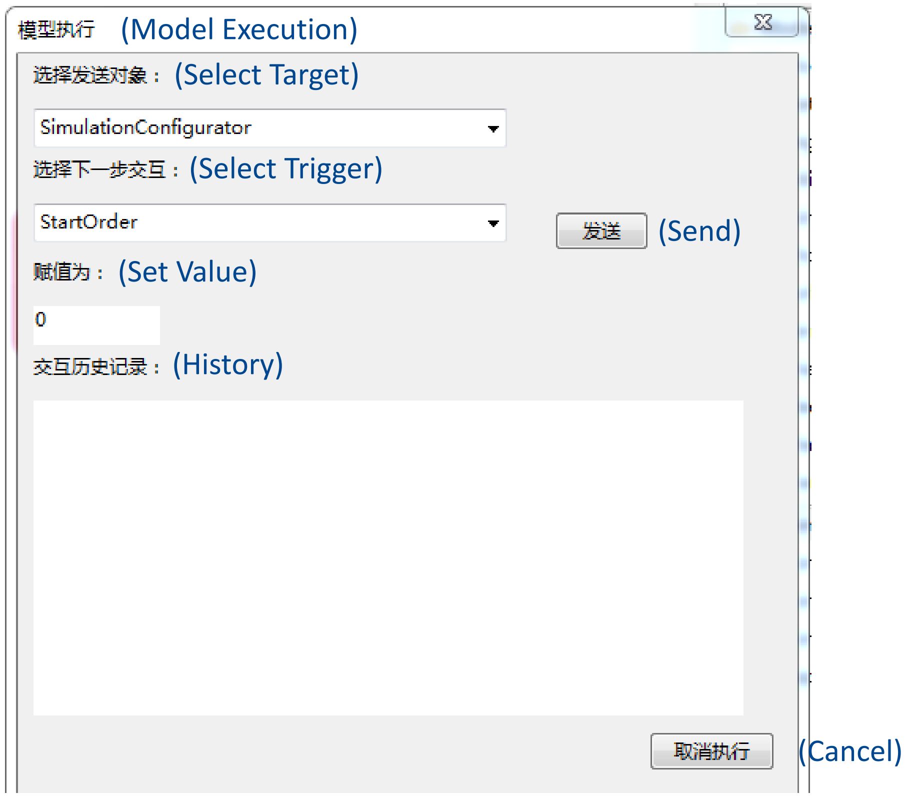
Passenger
Seat TV Airline-Specific Interactions Manager
Seat TV Core Interactions Manager
Seat TV Audio Video Player
Streaming Server
ref
[ES] Select VOD Movie
VOD Movie Play Selection
Capture VOD Selections
VOD Movie Play Selection
Process VOD Movie Controls
VOD Movie Control Command
Broadcast Stored Audio and Video
Audio-Video Stream
Play Audio-Video Stream on Seat TV
Played Audio Video
Watch Movie on Private Screen

Challenge: **how to verify the architecture design is correct ?**

- **Is the mode and state machine consistent with scenarios ?**
- **Is the function decomposition appropriate ? Can the coordination of various functions achieve the desired results ?**
- **…**

Solution





模型执行 (Model Execution)

选择发送对象： (Select Target)

SimulationConfigurator

选择下一步交互： (Select Trigger)

StartOrder

赋值为： (Set Value)

0

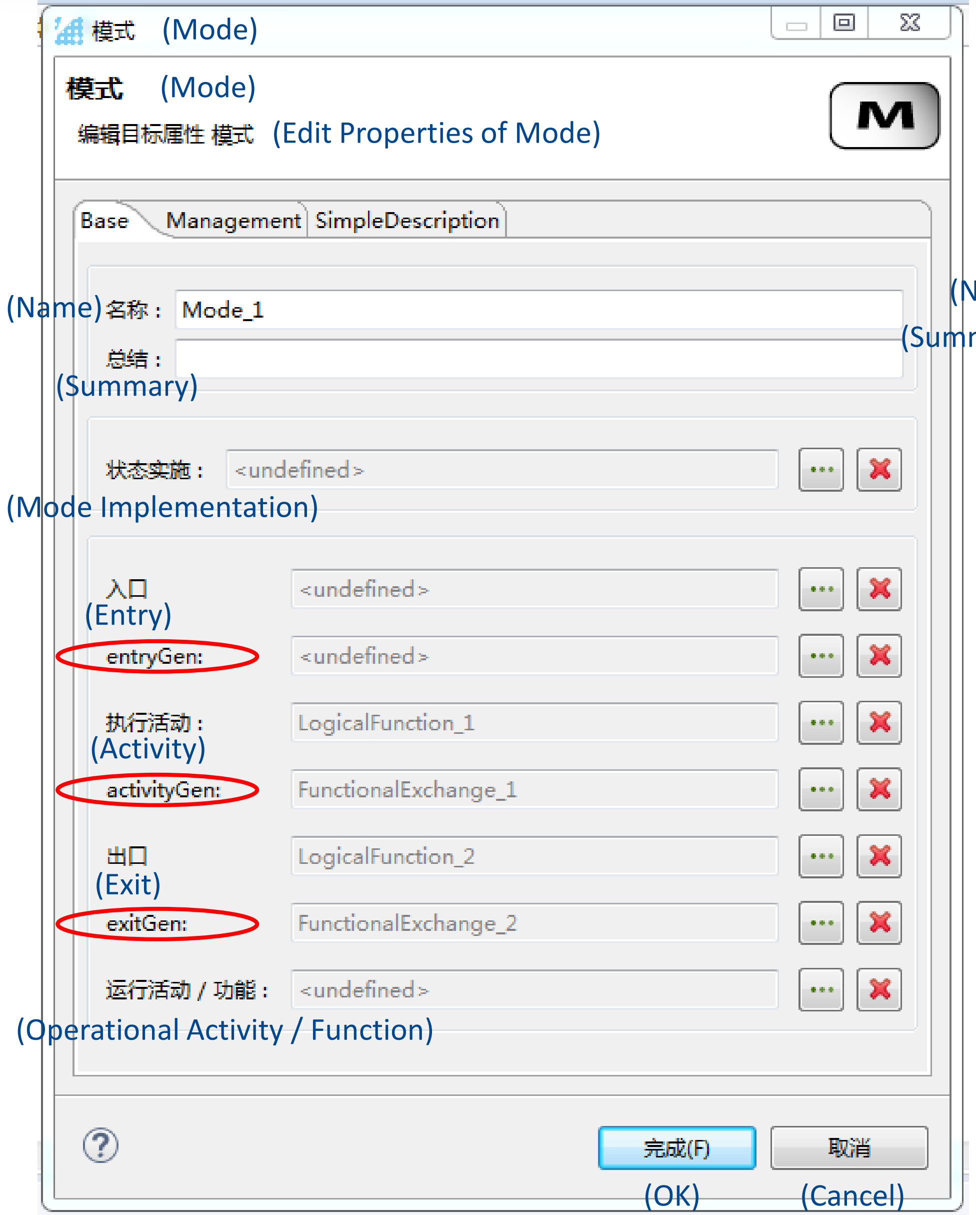
交互历史记录： (History)

- Develop an add-on to make mode and state machines executable. Use a control panel to interact with the state machine, and automatically record execution process as a scenario. Judge if the sequence of functions and interfaces is desirable by compare manual scenario and auto-record scenario.
- During state machine execution, M code (and C code) embedded in functions can be directly invoked to simulate the operation effect of the architecture.

02

Execution and Simulation Rules

Execution Rules of State Machine (1/3)



模式 (Mode)

模式 (Mode)

编辑目标属性 模式 (Edit Properties of Mode)

Base | Management | SimpleDescription

(Name) 名称： Mode_1

总结： (Summary)

状态实施： <undefined> (Mode Implementation)

入口 (Entry) <undefined>

entryGen: <undefined>

执行活动： (Activity) LogicalFunction_1

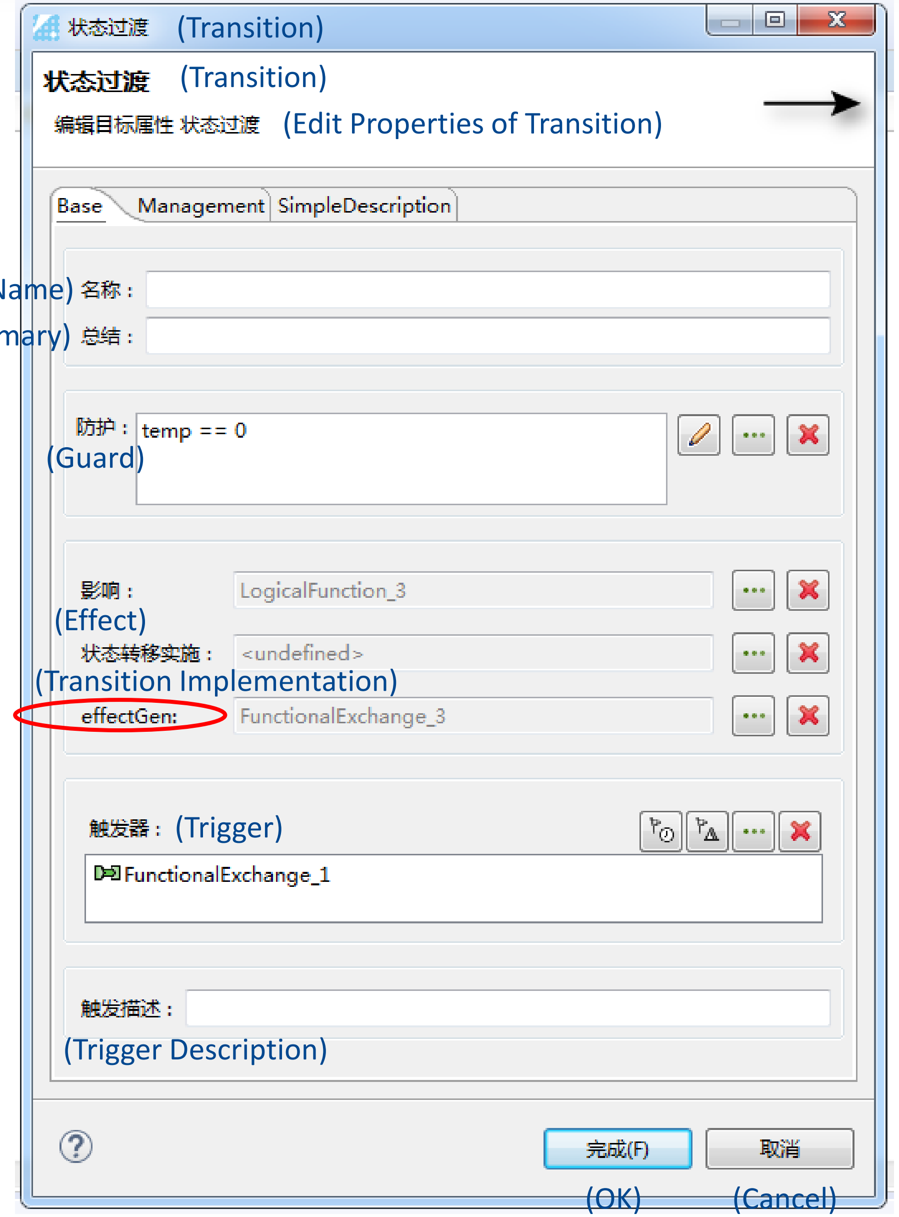
activityGen: FunctionalExchange_1

出口 (Exit) LogicalFunction_2

exitGen: FunctionalExchange_2

运行活动 / 功能： <undefined> (Operational Activity / Function)

完成(F) (OK) 取消 (Cancel)



状态过渡 (Transition)

状态过渡 (Transition)

编辑目标属性 状态过渡 (Edit Properties of Transition)

Base | Management | SimpleDescription

(Name) 名称：

(Summary) 总结：

防护： (Guard) temp == 0

影响： (Effect) LogicalFunction_3

状态转移实施： <undefined> (Transition Implementation)

effectGen: FunctionalExchange_3

触发器： (Trigger)

FunctionalExchange_1

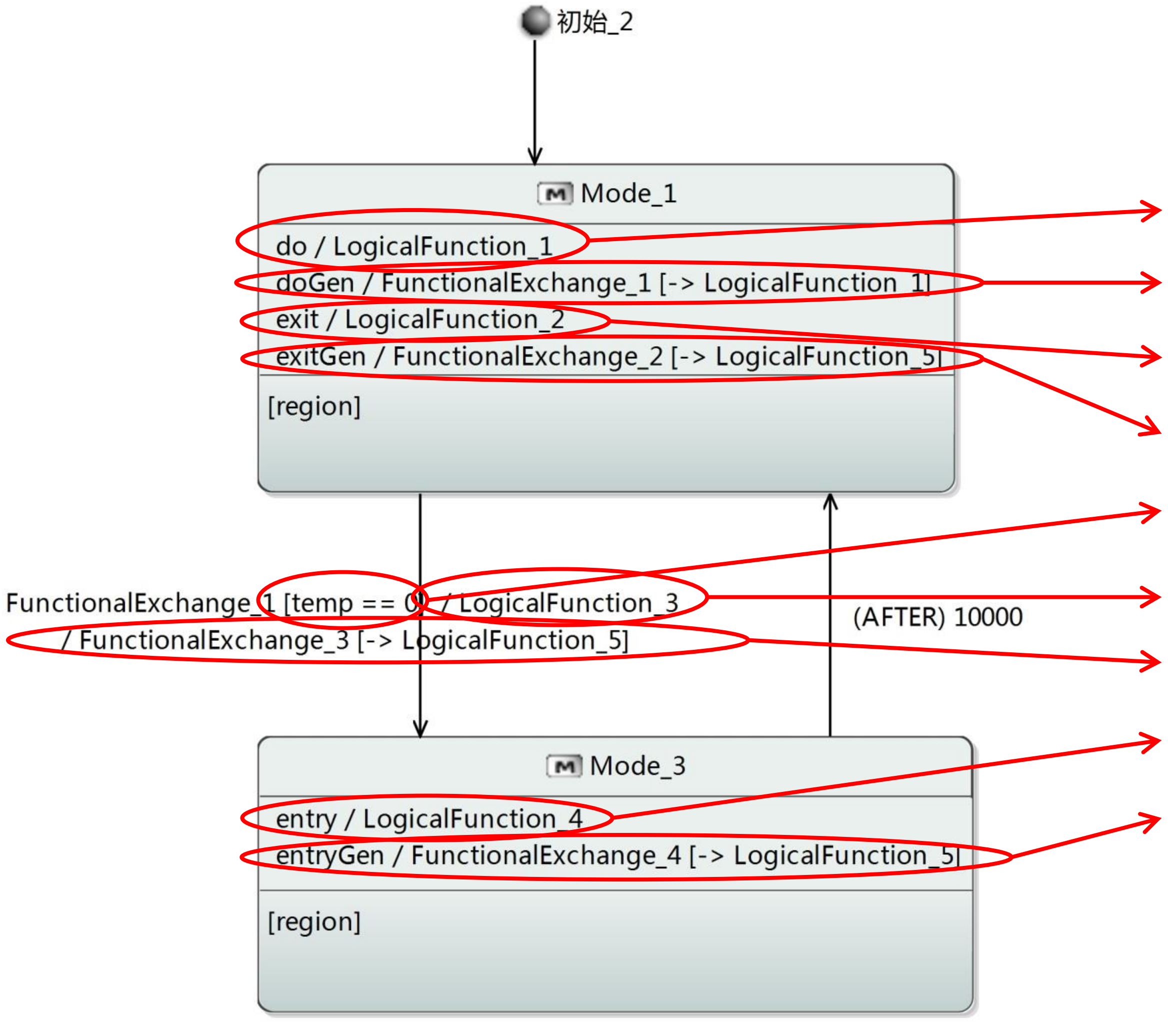
触发描述： (Trigger Description)

完成(F) (OK) 取消 (Cancel)

Based on Mealey machine, as follows:

- The trigger of transition can only be functional exchange or time event
- The guard of transition can directly reference property values from relative component
- Different state machine can use “Gen” mechanism to trigger each other

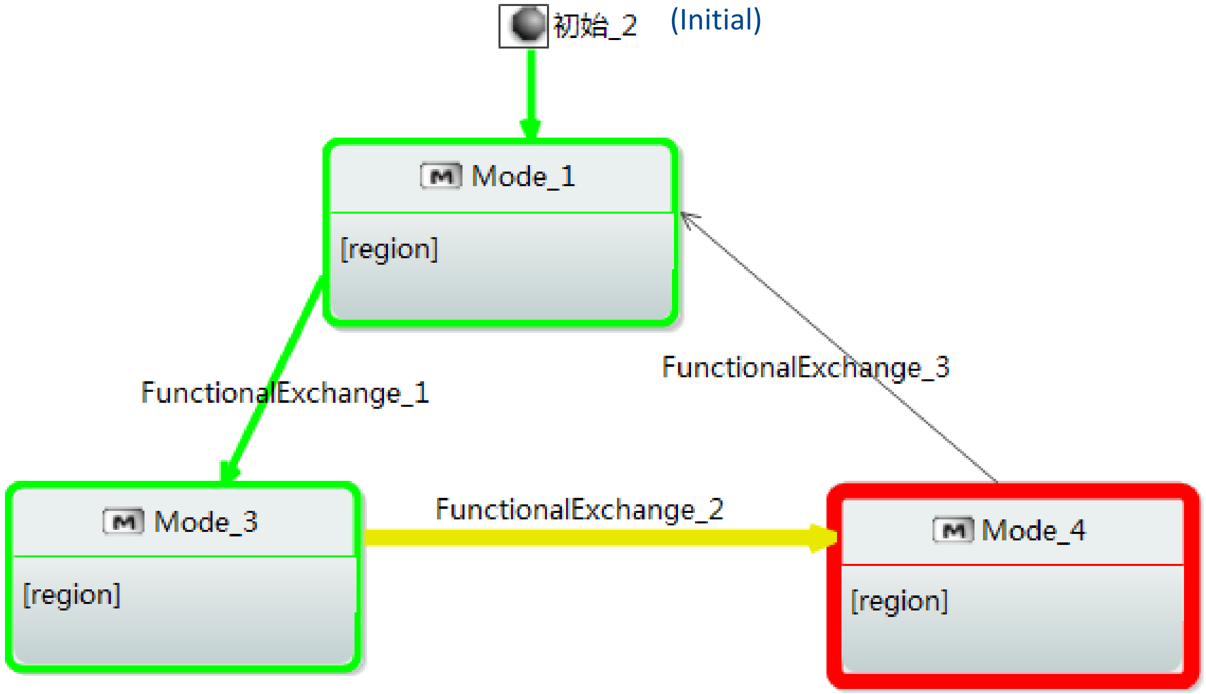
Execution Rules of State Machine (2/3)


初始_2
Mode_1
do / LogicalFunction_1
doGen / FunctionalExchange_1 [-> LogicalFunction_1]
exit / LogicalFunction_2
exitGen / FunctionalExchange_2 [-> LogicalFunction_5]
[region]
FunctionalExchange_1 [temp == 0] / LogicalFunction_3
/ FunctionalExchange_3 [-> LogicalFunction_5]
(AFTER) 10000
Mode_3
entry / LogicalFunction_4
entryGen / FunctionalExchange_4 [-> LogicalFunction_5]
[region]

When transition occurs, the sequence of function and Gen is as follows:

1. Execute do function of pre-state
2. Generate doGen Trigger of pre-state
3. Execute exit function of pre-state
4. Generate exitGen Trigger of pre-state
5. Guard of transition can reference property value
6. Execute effect function of transition
7. Generate effectGen Trigger of transition
8. Execute entry function of post-state
9. Generate entryGen Trigger of post-state

Execution Rules of State Machine (3/3)

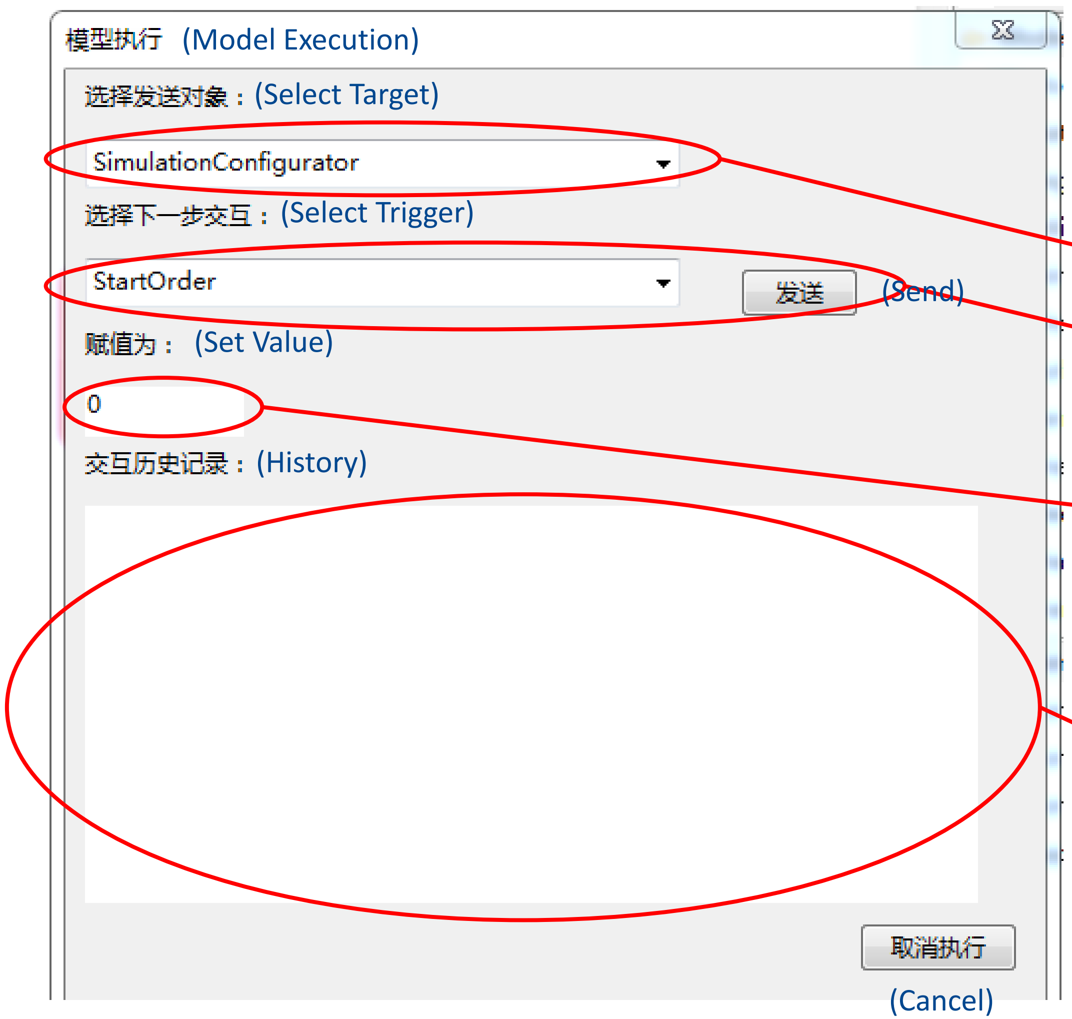


Diplay rules during execution are as follows:

- Current state/mode is highlighted by red
- Transition to current state is highlighted by yellow
- States/modes and transitions that have been through are highlighted by green
- States/modes and transitions that have not been through display as default

The process of execution will be record as scenarios automatically

Control Panel



模型执行 (Model Execution)

选择发送对象：(Select Target)

SimulationConfigurator

选择下一步交互：(Select Trigger)

StartOrder

赋值为： (Set Value)

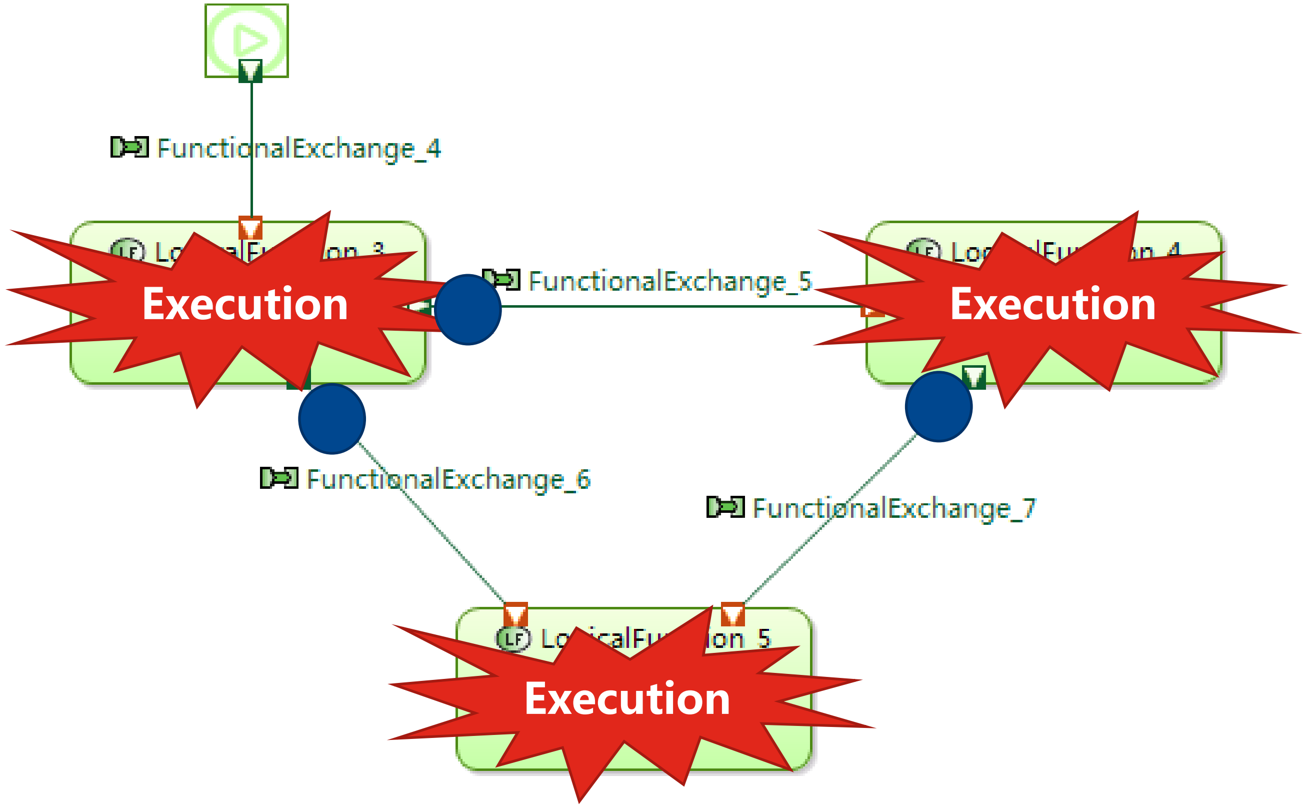
0

交互历史记录：(History)

Users can use control panel to interact with mode/state machines during execution:

- Select the target to send triggers
- Select the trigger to be sent, and send it
- Define the value of the trigger to be sent (the value will be used when relative function has embedded M code)
- History of triggers that have been sent during execution

Execution Rules of Function Data Flow



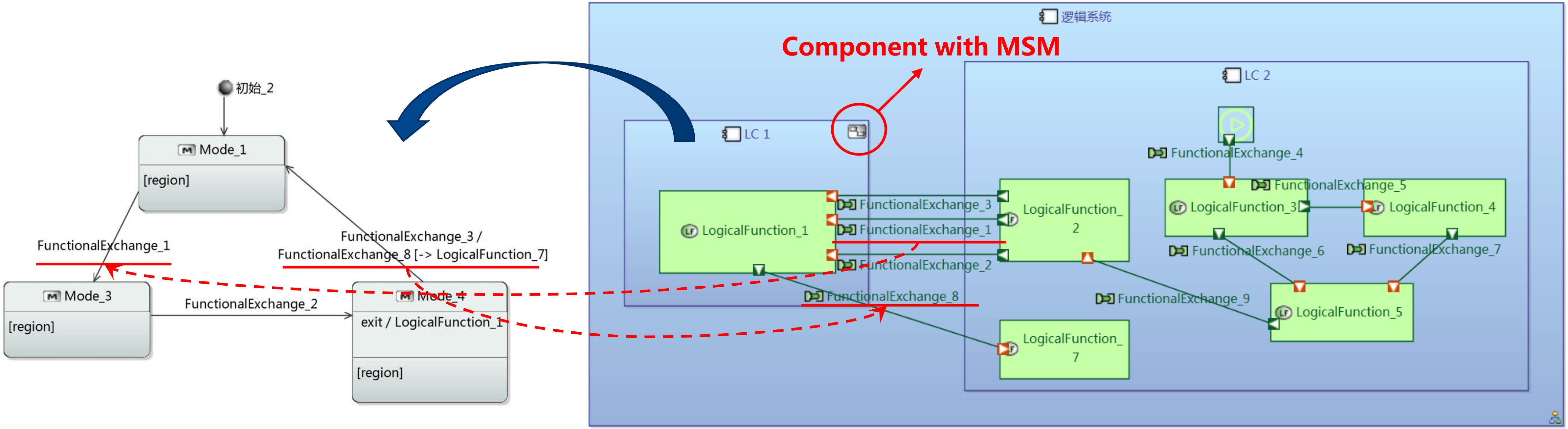
Our execution add-on can support execution of function data flow other than mode/state machine. The rule is based on Petri-net, as follows:

- Add initial node, the function connected to initial node will be executed at first
- The beginning condition of a function's execution is that all input ports has at least one token
- The execution of function will consume one token on each input port
- After execution, a function will produce one token on each output port
- A token will move from the output port to the input port immediately after it is produced
- The process of execution will be record as scenarios automatically

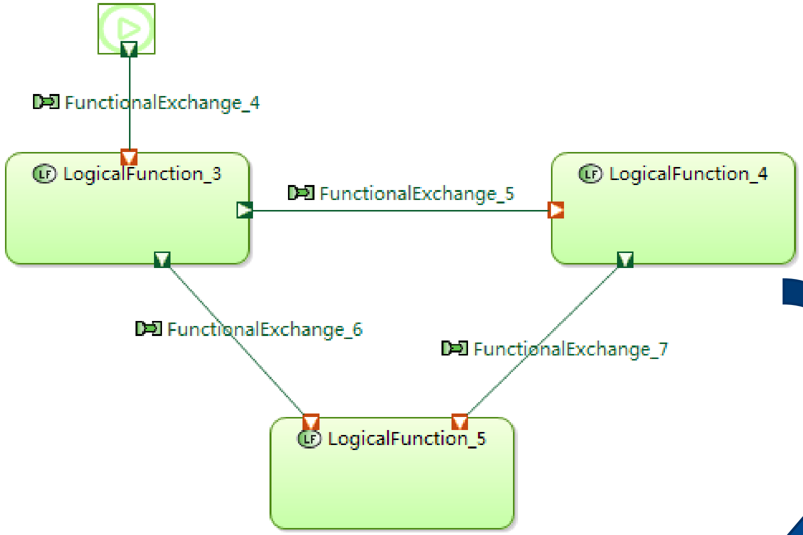
Execution Rules of the Whole Model

If some components have MSM while others don’t, our execution add-on can make a hybrid-execution of state machine rule and function data flow rule:

- **Components with MSM will execute under state machine rule, and others under function data flow rule**
- **At the border between state machine rule and function data flow rule, triggers in state machine are equal with tokens in function data flow**

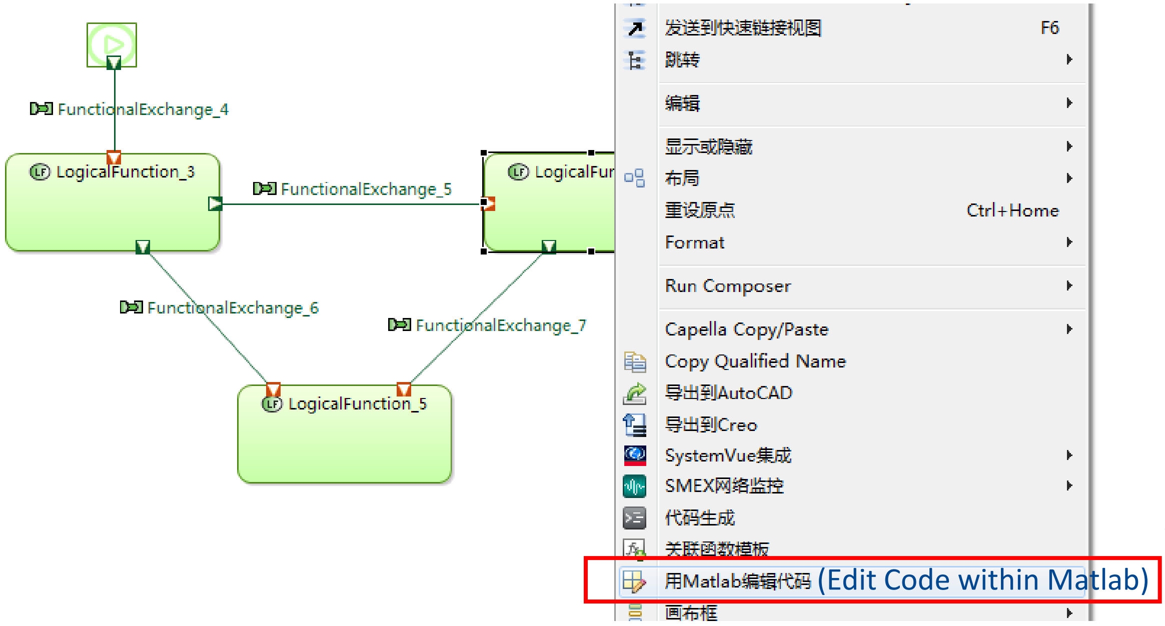


Matlab Code Embedded in Function

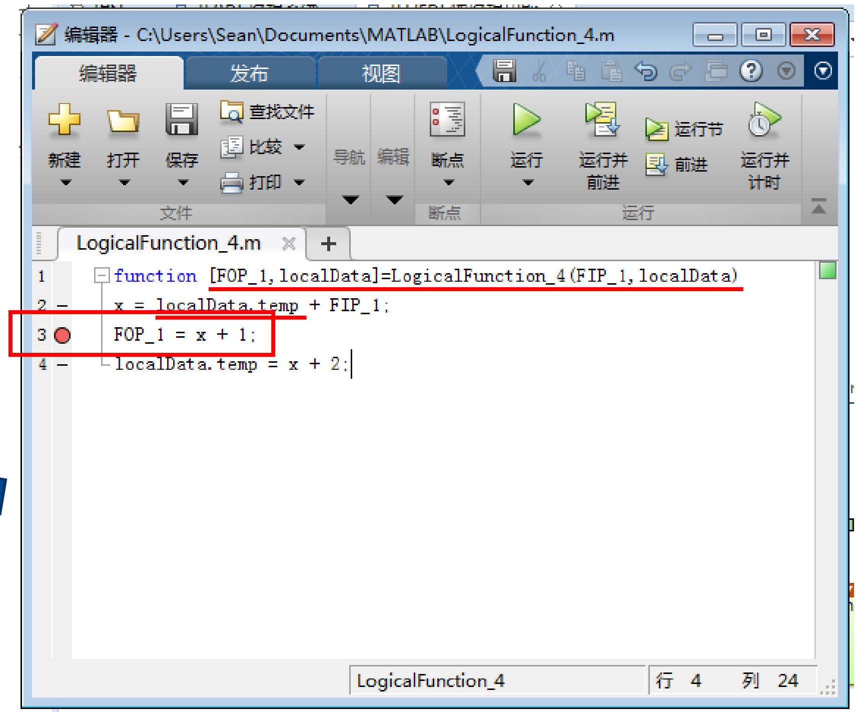


Users can invoke Matlab editor to edit embedded M code by function's right-click menu:

- The first line of Matlab function is auto-generated, re-using function name and functional input/output
- Property value of relative component can be referenced by "localData"
- Breakpoints can be added in Matlab Editor to debug while execution

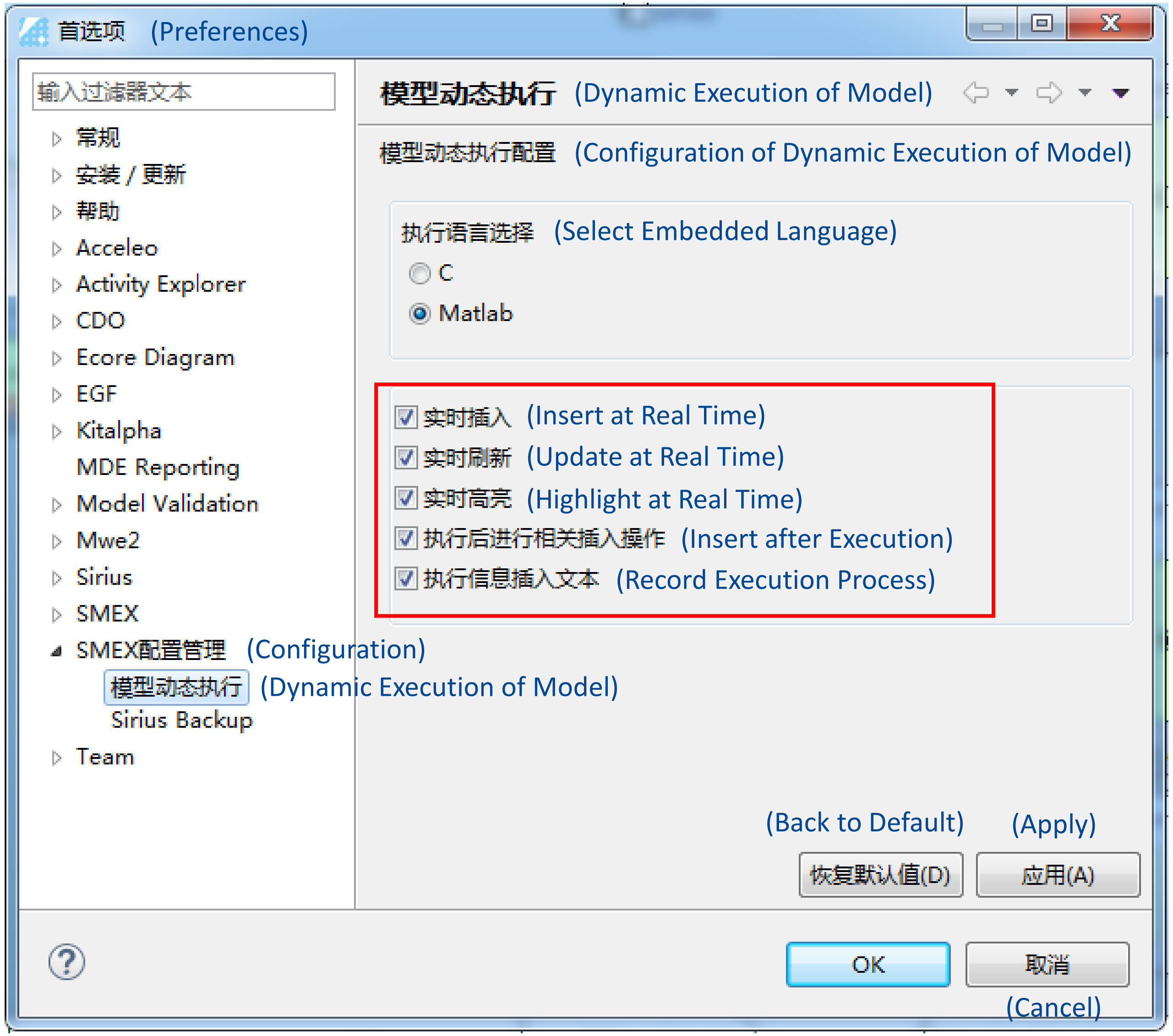


(Edit Code within Matlab)



```
function [FOP_1,localData]=LogicalFunction_4(FIP_1,localData)
x = localData.temp + FIP_1;
FOP_1 = x + 1;
localData.temp = x + 2;
```

Display Options for Execution



模型动态执行 (Dynamic Execution of Model)

模型动态执行配置 (Configuration of Dynamic Execution of Model)

执行语言选择 (Select Embedded Language)

○ C

● Matlab

☑ 实时插入 (Insert at Real Time)

☑ 实时刷新 (Update at Real Time)

☑ 实时高亮 (Highlight at Real Time)

☑ 执行后进行相关插入操作 (Insert after Execution)

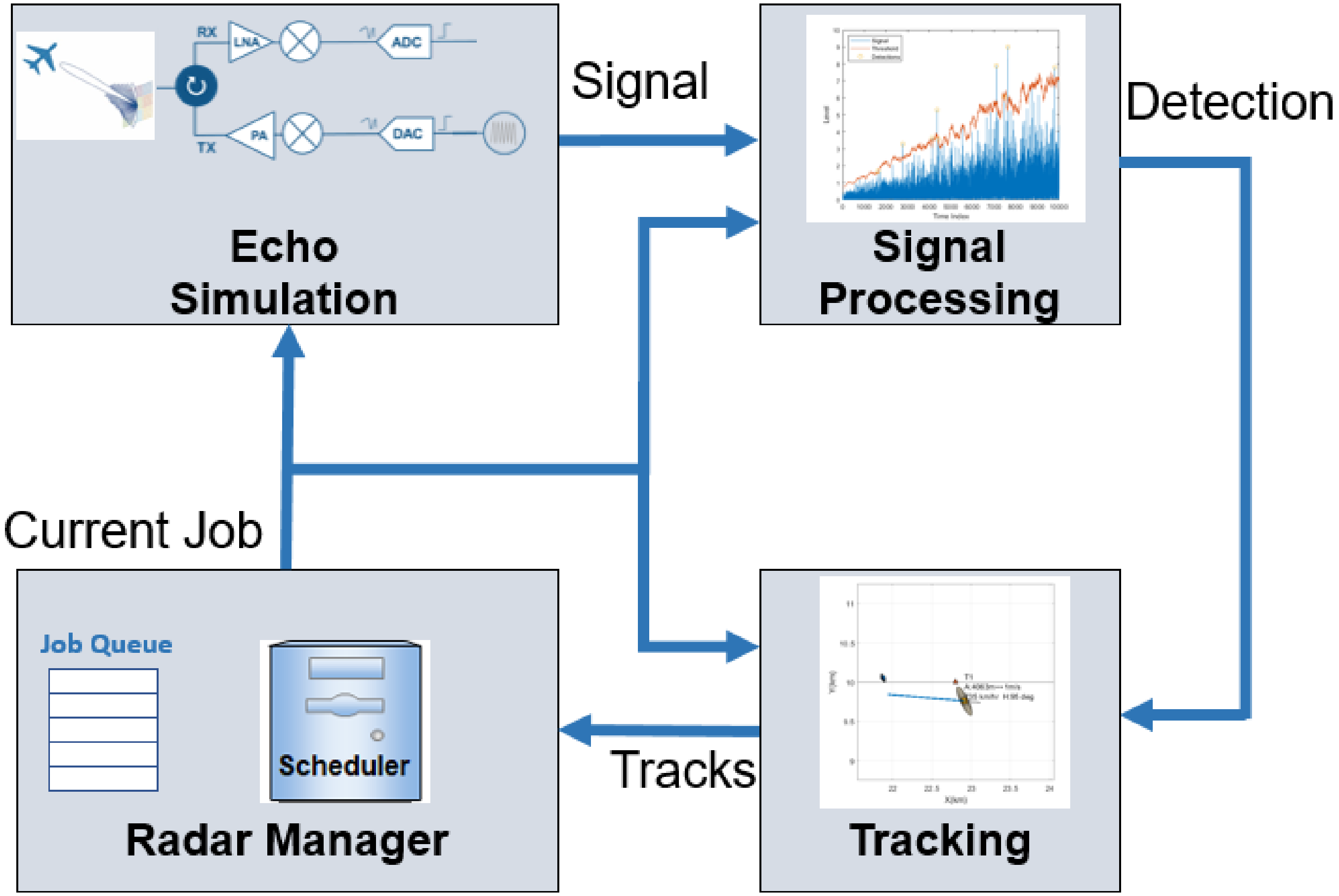
☑ 执行信息插入文本 (Record Execution Process)

- The number of functions and interfaces invoked in execution for a real (RADAR) system simulation is very large (from thousands to millions). Sometimes it’s impossible to highlight MSM and update scenarios in real time. So we need display options to improve execution efficiency in some cases.
- If the real-time update option is closed, the execution process will be record as xml file, which can be partially inserted into scenario after execution.

03

Example

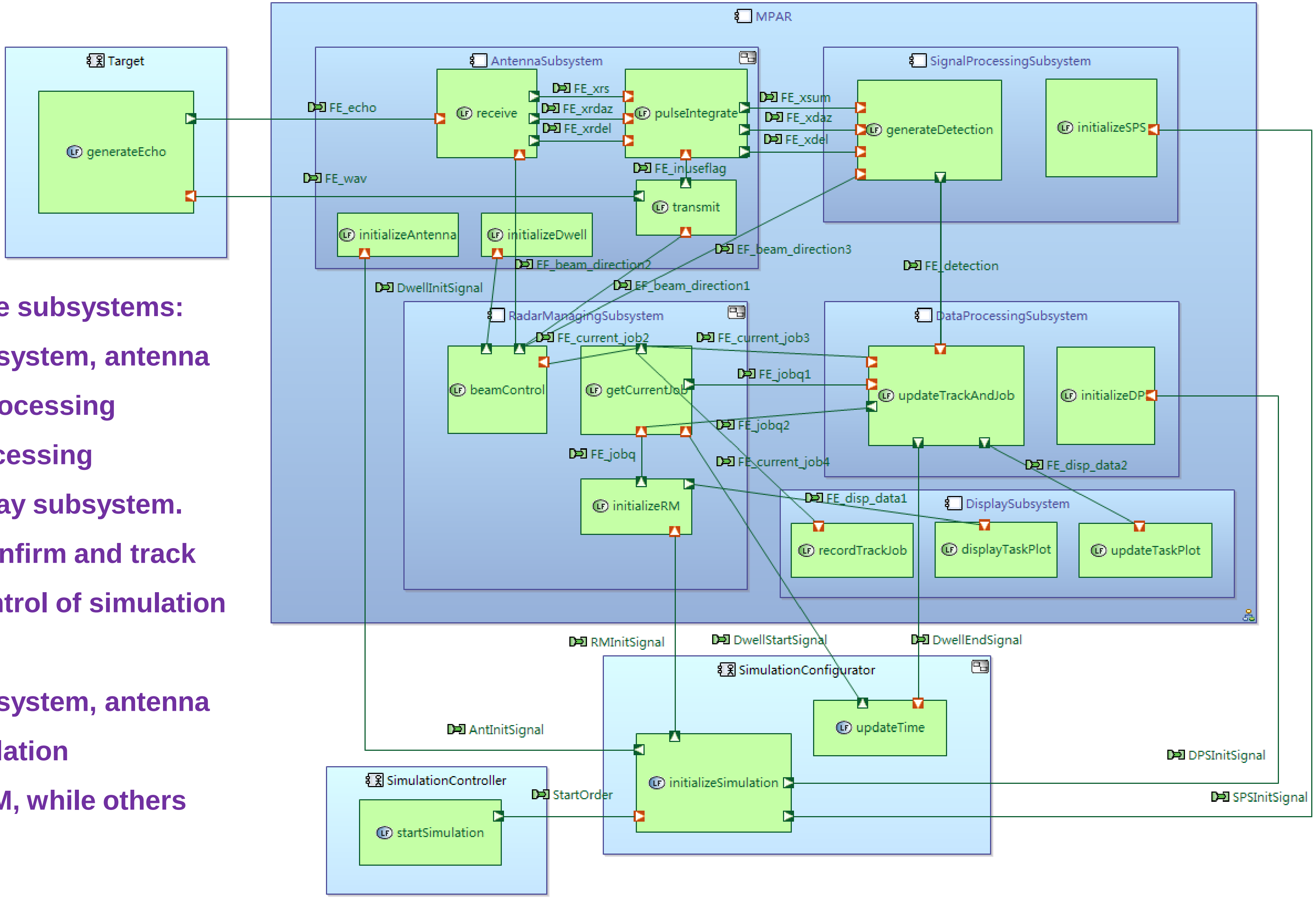
Introduction to MPAR Example



-- This picture comes from Matlab 2019a

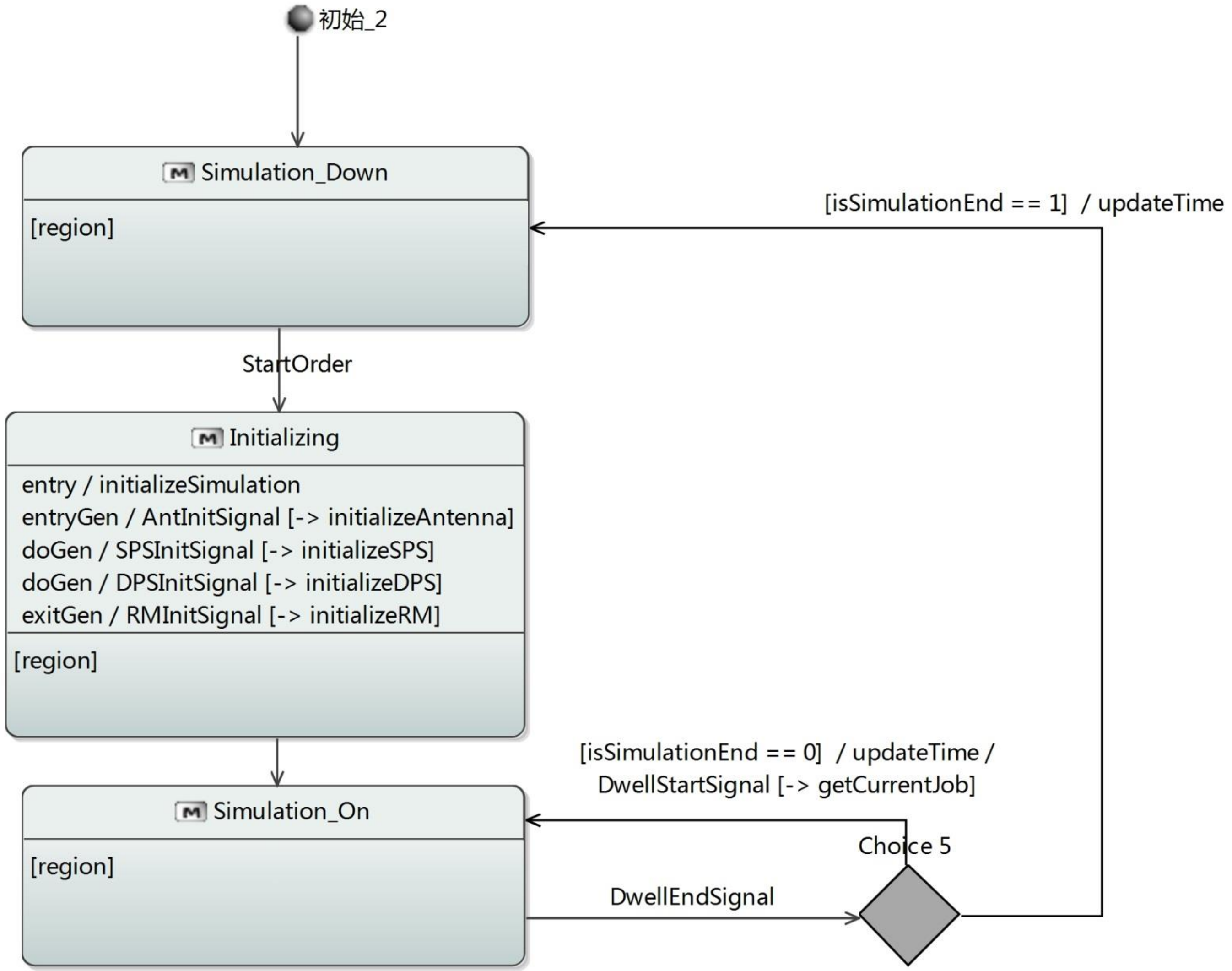
- MPAR (Multi-Functional Phased Array Radar) performs both scanning (searching) and tracking tasks. Most of its simulation code comes from MPARSearchTrackExample in Matlab 2019a.
- This example intends to show how to use execution add-on to do system simulation. It's not about how to design a real phased array Radar. So it only uses baseband signal, and simplified signal processing.

Logical Architecture of MPAR



- MPAR consists of five subsystems: Radar managing subsystem, antenna subsystem, signal processing subsystem, data processing subsystem and display subsystem. MPAR will search, confirm and track targets under the control of simulation configurator.
- Radar managing subsystem, antenna subsystem and simulation configurator has MSM, while others don’t.

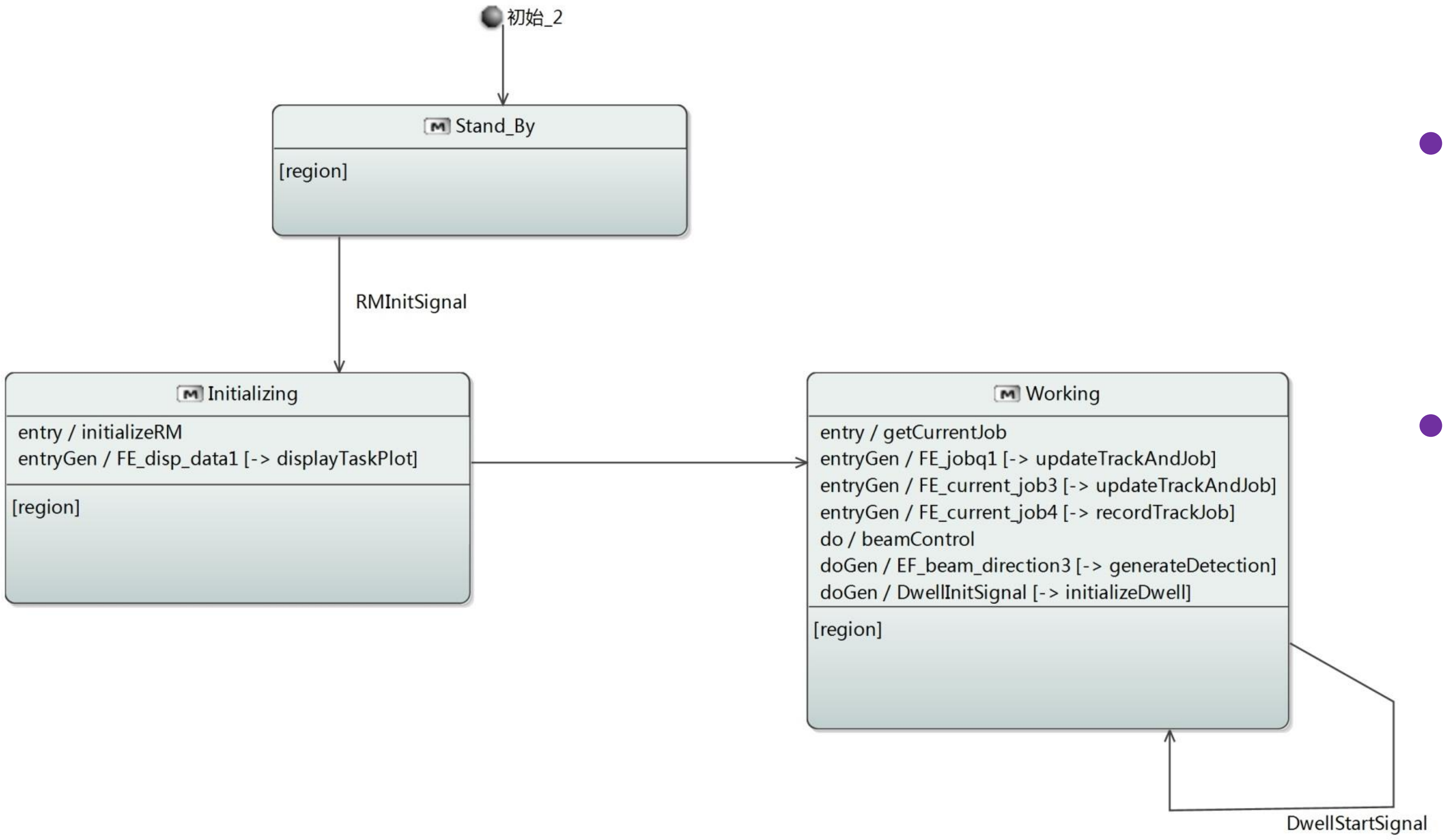
MPAR MSM: Simulation Configurator



初始_2
Simulation_Down
[region]
[isSimulationEnd == 1] / updateTime
StartOrder
Initializing
entry / initializeSimulation
entryGen / AntInitSignal [-> initializeAntenna]
doGen / SPSInitSignal [-> initializeSPS]
doGen / DPSInitSignal [-> initializeDPS]
exitGen / RMInitSignal [-> initializeRM]
[region]
[isSimulationEnd == 0] / updateTime / DwellStartSignal [-> getCurrentJob]
Simulation_On
[region]
Choice 5
DwellEndSignal

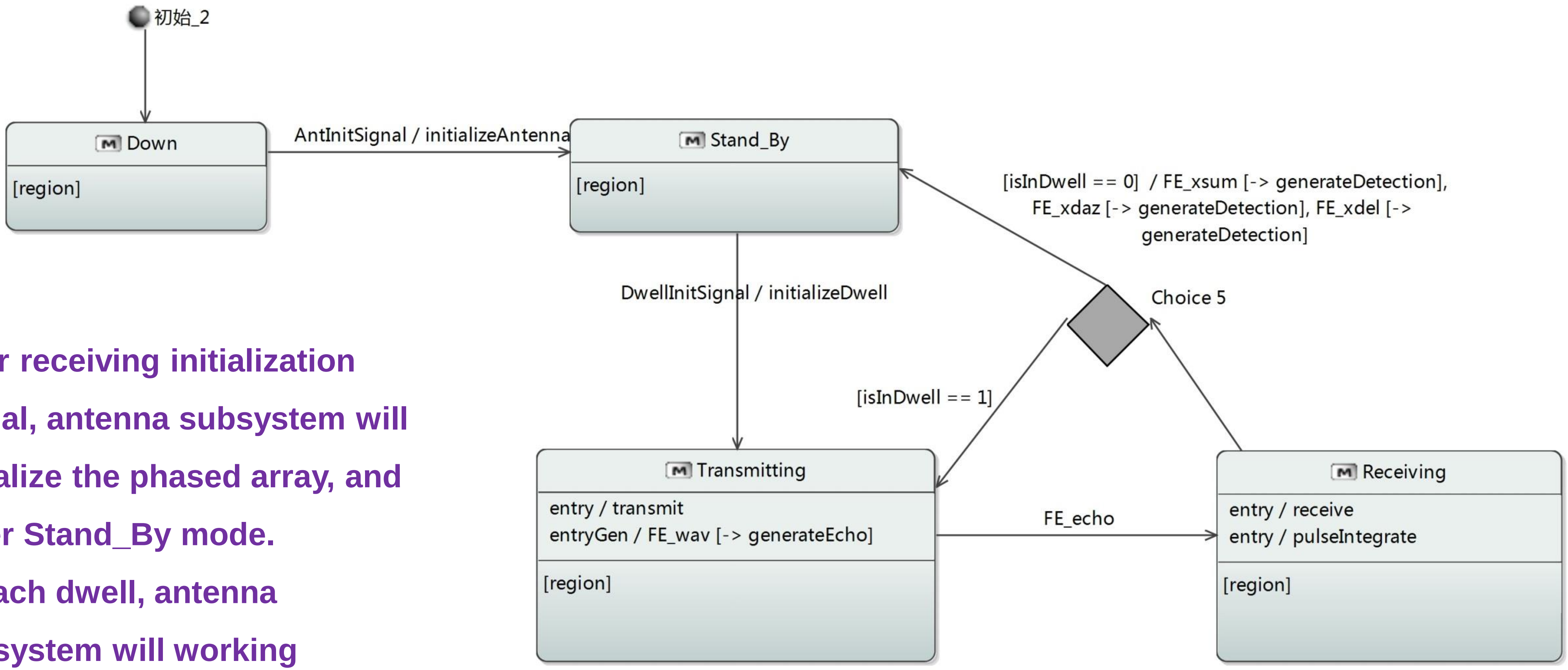
- **After receiving StartOrder from simulation controller, simulation configurator will initialize MPAR simulation environment, and send initialization signal to different subsystems.**
- **When simulation is on, the configurator will update simulation time at the end of each dwell.**

MPAR MSM: Radar Managing Subsystem


初始_2
Stand_By
[region]
RMInitSignal
Initializing
entry / initializeRM
entryGen / FE_disp_data1 [-> displayTaskPlot]
[region]
Working
entry / getCurrentJob
entryGen / FE_jobq1 [-> updateTrackAndJob]
entryGen / FE_current_job3 [-> updateTrackAndJob]
entryGen / FE_current_job4 [-> recordTrackJob]
do / beamControl
doGen / EF_beam_direction3 [-> generateDetection]
doGen / DwellInitSignal [-> initializeDwell]
[region]
DwellStartSignal

- After receiving initialization signal, Radar managing subsystem will initialize job queue, and enter working mode.
- In working mode, Radar managing subsystem will get current job at the beginning of each dwell, and provide job information and beam direction to other subsystems.

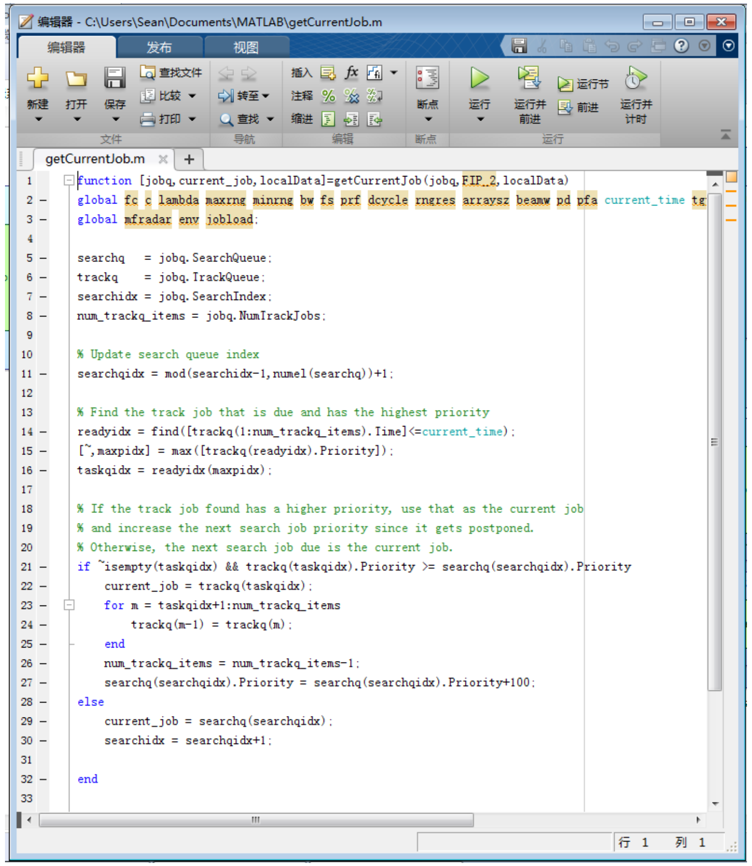
MPAR MSM: Antenna Subsystem



初始_2
Down
[region]
AntInitSignal / initializeAntenna
Stand_By
[region]
[isInDwell == 0] / FE_xsum [-> generateDetection], FE_xdaz [-> generateDetection], FE_xdel [-> generateDetection]
DwellInitSignal / initializeDwell
Choice 5
[isInDwell == 1]
Transmitting
entry / transmit
entryGen / FE_wav [-> generateEcho]
[region]
FE_echo
Receiving
entry / receive
entry / pulseIntegrate
[region]

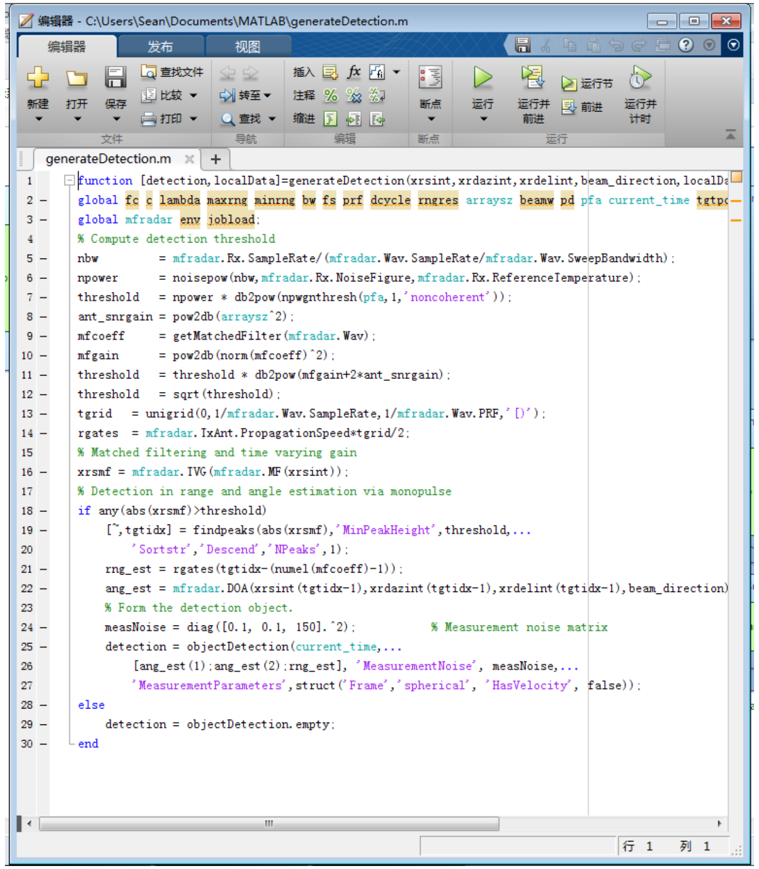
- After receiving initialization signal, antenna subsystem will initialize the phased array, and enter Stand_By mode.
- In each dwell, antenna subsystem will working through Transmitting mode and Receiving mode for every pulse, until all pulses are finished.

Embedded Simulation Code for MPAR Functions (Partial)



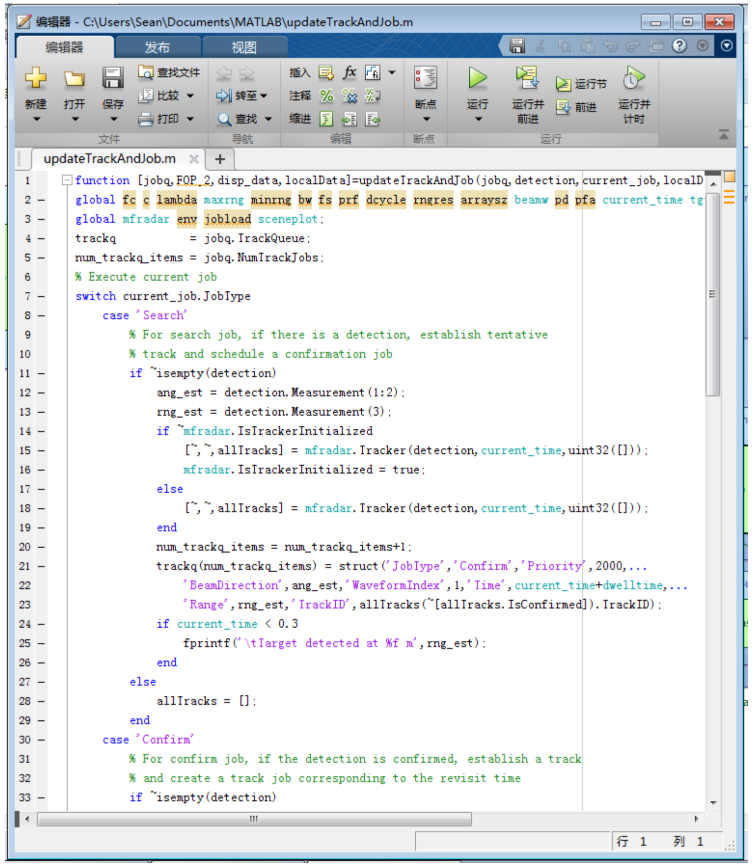
```matlab
function [jobq,current_job,localData]=getCurrentJob(jobq,FIP_2,localData)
global fc c lambda maxrng minrng bw fs prf dcycle rngres arraysz beamw pd pfa current_time tg
global mfradar env jobload;

searchq   = jobq.SearchQueue;
trackq    = jobq.TrackQueue;
searchidx = jobq.SearchIndex;
num_trackq_items = jobq.NumTrackJobs;

% Update search queue index
searchqidx = mod(searchidx-1,numel(searchq))+1;

% Find the track job that is due and has the highest priority
readyidx = find([trackq(1:num_trackq_items).Time]<=current_time);
[~,maxpidx] = max([trackq(readyidx).Priority]);
taskqidx = readyidx(maxpidx);

% If the track job found has a higher priority, use that as the current job
% and increase the next search job priority since it gets postponed.
% Otherwise, the next search job due is the current job.
if ~isempty(taskqidx) && trackq(taskqidx).Priority >= searchq(searchqidx).Priority
    current_job = trackq(taskqidx);
    for m = taskqidx+1:num_trackq_items
        trackq(m-1) = trackq(m);
    end
    num_trackq_items = num_trackq_items-1;
    searchq(searchqidx).Priority = searchq(searchqidx).Priority+100;
else
    current_job = searchq(searchqidx);
    searchidx = searchqidx+1;

end
```

getCurrentJob



```matlab
function [detection,localData]=generateDetection(xrsint,xrdazint,xrdelint,beam_direction,localDa
global fc c lambda maxrng minrng bw fs prf dcycle rngres arraysz beamw pd pfa current_time tgtpc
global mfradar env jobload;
% Compute detection threshold
nbw         = mfradar.Rx.SampleRate/(mfradar.Wav.SampleRate/mfradar.Wav.SweepBandwidth);
npower      = noisepow(nbw,mfradar.Rx.NoiseFigure,mfradar.Rx.ReferenceTemperature);
threshold   = npower * db2pow(npwgnthresh(pfa,1,'noncoherent'));
ant_snrgain = pow2db(arraysz^2);
mfcoeff     = getMatchedFilter(mfradar.Wav);
mfgain      = pow2db(norm(mfcoeff)^2);
threshold   = threshold * db2pow(mfgain+2*ant_snrgain);
threshold   = sqrt(threshold);
tgrid   = unigrid(0,1/mfradar.Wav.SampleRate,1/mfradar.Wav.PRF,'[)');
rgates  = mfradar.TxAnt.PropagationSpeed*tgrid/2;
% Matched filtering and time varying gain
xrsmf = mfradar.TVG(mfradar.MF(xrsint));
% Detection in range and angle estimation via monopulse
if any(abs(xrsmf)>threshold)
    [~,tgtidx] = findpeaks(abs(xrsmf),'MinPeakHeight',threshold,...
        'Sortstr','Descend','NPeaks',1);
    rng_est = rgates(tgtidx-(numel(mfcoeff)-1));
    ang_est = mfradar.DOA(xrsint(tgtidx-1),xrdazint(tgtidx-1),xrdelint(tgtidx-1),beam_direction)
    % Form the detection object.
    measNoise = diag([0.1, 0.1, 150].^2);           % Measurement noise matrix
    detection = objectDetection(current_time,...
        [ang_est(1);ang_est(2);rng_est], 'MeasurementNoise', measNoise,...
        'MeasurementParameters',struct('Frame','spherical', 'HasVelocity', false));
else
    detection = objectDetection.empty;
end
```

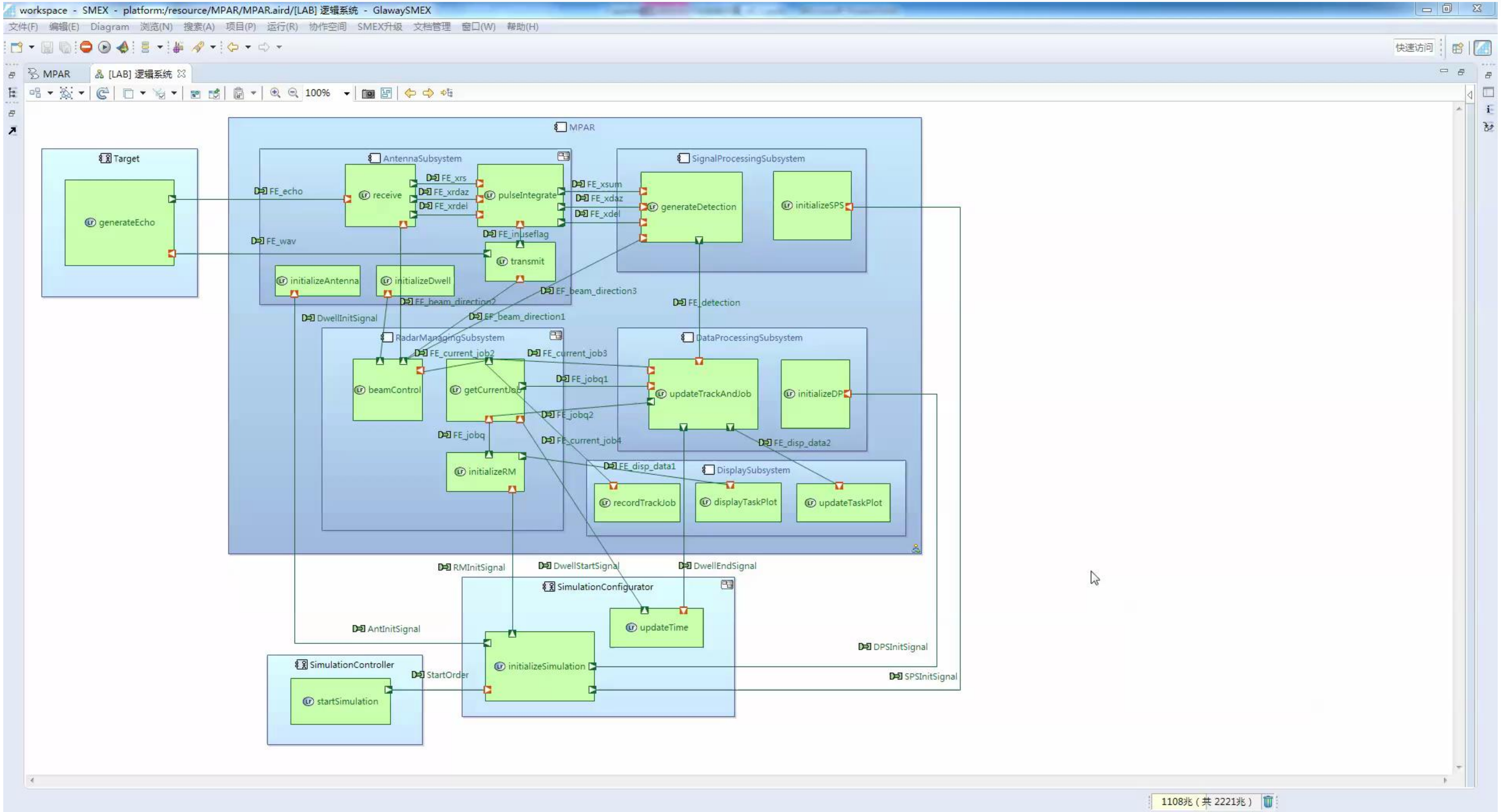
generateDetection



```matlab
function [jobq,FOP_2,disp_data,localData]=updateTrackAndJob(jobq,detection,current_job,localD
global fc c lambda maxrng minrng bw fs prf dcycle rngres arraysz beamw pd pfa current_time tg
global mfradar env jobload sceneplot;
trackq          = jobq.TrackQueue;
num_trackq_items = jobq.NumTrackJobs;
% Execute current job
switch current_job.JobType
    case 'Search'
        % For search job, if there is a detection, establish tentative
        % track and schedule a confirmation job
        if ~isempty(detection)
            ang_est = detection.Measurement(1:2);
            rng_est = detection.Measurement(3);
            if ~mfradar.IsTrackerInitialized
                [~,~,allTracks] = mfradar.Tracker(detection,current_time,uint32([]));
                mfradar.IsTrackerInitialized = true;
            else
                [~,~,allTracks] = mfradar.Tracker(detection,current_time,uint32([]));
            end
            num_trackq_items = num_trackq_items+1;
            trackq(num_trackq_items) = struct('JobType','Confirm','Priority',2000,...
                'BeamDirection',ang_est,'WaveformIndex',1,'Time',current_time+dwelltime,...
                'Range',rng_est,'TrackID',allTracks(~[allTracks.IsConfirmed]).TrackID);
            if current_time < 0.3
                fprintf('\tTarget detected at %f m',rng_est);
            end
        else
            allTracks = [];
        end
    case 'Confirm'
        % For confirm job, if the detection is confirmed, establish a track
        % and create a track job corresponding to the revisit time
        if ~isempty(detection)
```

updateTrackAndJob

Execution Effect of MPAR Example



Other Add-on Provided by Glaway

No.	Add-on Name	Introduction
1	DGS（Document Generation Server）	Transform Capella model into documents. Users can edit templates in Word in a “what you see is what you get” manner, without using Aql.
2	TPM（Technical Performance Manager）	Model and manage all MOE, MOP and TPM in Capella model. Users can make quantitative analysis among different measurements, and graphically analyze change impact of a given measurement.
3	ICM（Interface Control Manager）	Detailed design of interfaces based on functional exchange, component exchange, and physical link. Include message design, data word design and pin design based on different bus types, e.g. 1553B, RS422, RS485, CAN, AFDX and so on.

GW
Glaway soft®

南京国睿信维软件有限公司
智慧企业解决方案及自主工业软件提供商

Thanks！

客户至上　价值创新　以人为本　团队协作