Execution Flow (Part 2)
openPASS Release 0.6

04.12.2019 – René Paris, on behalf of BMW AG
Bird's eye view

RunInstantiator::Run

Get Configs
- Read into internal structures: slaveConfig, experimentConfig, scenario, scenery

Build Empty World
- Use static information from openDRIVE file

For each invocation
- Each invocation is newly seeded
- E.g. remove cars from World and reset to initial state, delete detected Events, ...
- E.g. TimeOfDay for World or TrafficConfig for SpawnPoints

Sample Parameters

Generate Result Storage
- Observable results for this invocation

Init Stochastics
- Make aware of results object or inject stochastic parameters

Clean Up
- All observers call their `SlavePreRunHook`

Sample Parameters
- Executes a single run which fills `result`; periodically calls observers' `SlaveUpdateHook`

Generate Result Storage
- All observers call their `SlavePostRunHook`

Init Modules
- All observers call their `SlavePostHook`

Observers: Init Run

Observers: Finalize Run

Observers: Finalize All

Run Scheduler
Spawn agents \( t=0 \) [create scenario] and roll next agent (future step) [pre-spawning]

Evaluate conditions given by openSCENARIO description

Execute actions given by openSCENARIO description

→ All instantiated agent components are executed in order of priority

→ The first component (here \( c_1 \)) does not receive signals

→ Inputs are updated for the next component (here \( c_2 \))

Write all updates on the AgentInterfaces into the ground truth

Record information about (the end) of the current time step

Set time for next time step
**PreSpawning Phase:**
When a car is placed, the spawn point calculates, when the next car should enter.

**Actual Spawning Phase:**
When next car should spawn, the world is checked once again for potential changes.

---

**For given/sampled**
- Lane, Traffic volume,
  Vehicle model,
  Target velocity, ...

**Calculate**
- Next spawn time

**Check**
- TTC for leading car in lane
  (this might have changed)

**Adjust**
- SpawnTime if necessary
- Velocity if nothing else works

---

**Runtime Spawning**
Work in progress...
Signal flow
The big picture
Simulation
Driver: Data Flow

SensorDriver → AgentFollowingDriver → AlgorithmLateral → DynamicsRegularDriving → AgentUpdater

Interface call

e.g. GetPosition() → AlgorithmLongitudinal → e.g. SetPosition(position)

Signals

Queued for Update

All components get the same values for the same timestep
SensorDriver \rightarrow AgentFollowingDriver

AlgorithmLateral

AlgorithmLongitudinal

DynamicsRegularDriving \rightarrow AgentUpdater

No signal input

Gather info for the driver

(Primitive) Driver Model

Calculate Pedalposition + Gear

Dynamics Model

Write back actual values

Calculate Steeringwheel Angle

LateralSignal

SteeringSignal

DynamicsSignal
Simulation Assistance System: Data Flow

- Sensor OSI
- SensorFusion OSI
- Assistance System
- Algorithm Lateral
- Algorithm Longitudinal
- Dynamics RegularDriving
- Agent Updater

Interface call

Queued for Update

e.g. GetWorldData()

Interface call

e.g. SetPosition(position)

WorldInterface

AgentInterface

All components get the same values for the same timestep
1. Forwards only active signal
2. ADAS always prioritized

- Driver
- Algorithm Longitudinal Driver
- Signal Prioritizer \(<\text{LongitudinalSignal}>\)
- Dynamics Regular Driving
- Assistance System
- Algorithm Longitudinal Vehicle Components

① Lateral skipped for brevity
The ComponentController

Driver

Vehicle Component (ADAS)

Dynamics_Trajectory Follower

Component Controller

Driver

Vehicle Component (ADAS)

Dynamics_Trajectory Follower

max reachable state, VC states, warnings

max reachable state, VC states

max reachable state, VC states

max reachable state, VC states

type (Driver), name, current state

t_i

t_{i+1}

name,
current
state,
warning (optional)

name,
current
state

name,
current
state

name,
current
state

max reachable state,
VC states

max reachable state,
VC states

max reachable state,
VC states

max reachable state,
VC states

The ComponentController
Exemplary activation at \( t = 1 \)

- **Assistance System inactive**
- **EventDetector** e.g. TimeTrigger
- **Manipulator ComponentState**
- **ComponentController**
  - `Trigger()` `UpdateOutput()`
- **AssistanceSystem**
  - `UpdateInput()` `Trigger()`
- **AgentUpdater**
  - `UpdateInput()` `Trigger()`
- **SyncGlobalData()**
  - \((x, y) \rightarrow (x, y)\)

**maxReachableState:** active

Update values for observations at \( t_3 \)
**Special Case: Trajectory Follower**

Exemplary activation at $t = 1$

- **Initial State**
  - Nobody moves

- **Event: Start TrajectoryFollower**

- **SyncGlobalData()**
  - $(x_2, y_2) = (x, y)$

- **UpdateInput()**
  - AgentUpdater
  - AgentInterface always deliver values from $t_1$

- **Trigger()**
  - TrajectoryFollower
  - Last instance writing to AgentInterface

- **UpdateOutput()**
  - Updated values passed as signals

- **Update values for observations at $t_2$**
Sample Profiles
- SampleDriverProfile
- SampleVehicleProfile
- SampleVehicleComponentProfiles

Dynamic Parameters
- Sample stochastic parameter (e.g. manufacturer variation)
  *e.g. Input: Frequency Variation iforner Parameter: Single Frequency*

AgentBuildInfo
- GatherBasicComponents
- GatherDriverComponents
- GatherVehicleComponents
- GatherSensors
• Select to which concrete profiles a agent should have (ProfileCatalog.xml)

• **SAMPLEDPROFILE** has already been sampled by respecting EGO, SCENARIO or COMMON agent selection

```java
SampledProfiles::make(...)  
- Which driver?  
  - Source XPath: //AgentProfile[@Name='SAMPLEDPROFILE'][@Type='Dynamic']/DriverProfiles .SampleDriverProfile()  
- Which car?  
  - Source XPath: //AgentProfile[@Name='SAMPLEDPROFILE'][@Type='Dynamic']/VehicleProfiles .SampleVehicleProfile()  
- Which components of the car?  
  - Source XPath: //VehicleProfile[@Name='SAMPLEDVEHICLE']/Components .SampleVehicleComponentProfiles();
```

• Currently, only the sensors have dynamic parametres

```java
- Source XPath: //SensorProfile/NormalDistribution[@Key='Latency']
DynamicParameters::make(...).SampleSensorLatencies();
```
GatherBasicComponents

Automatically*

- AgentUpdater
- ComponentController
- Dynamics_Collision
- Dynamics-RegularDriving
- Parameters_Vehicle
- PrioritizerDynamics
- PrioritizerLongitudinal
- PrioritizerSteering
- Sensor_RecordState

GatherDriverComponents

Configurable from ProfilesCatalog.xml
Xpath: //DriverProfile/String[@Key="..."] with @Key =

- Type (= Driver)
- ParametersModule
- SensorDriverModule
- AlgorithmLateralModule
- AlgorithmLongitudinalModule

Automatically*

- Action_LongitudinalDriver
- Action_SecondaryDriverTasks
- PrioritizerTurningIndicator

Must be set
Optional
Defaults to Sensor_Driver
Defaults to AlgorithmLateralDriver
Defaults to AlgorithmLongitudinalDriver

*) currently hardcoded and defined in dynamicAgentTypesGenerator.h
**GatherVehicleComponents**

Automatically*

- Algorithm_LateralVehicleComponents
- Algorithm_LongitudinalVehicleComponents
- PrioritizerAccelerationVehicleComponents
- PrioritizerLateralVehicleComponents
- LimiterAccelerationVehicleComponents

Additionally

- Resolve VehicleComponents for given **VehicleProfile**
- Create a parameter sets for every found **SensorLink**

**GatherSensors**

Automatically*

- SensorFusion and its dynamic inputs
- One ObservationModule per sensor

*) currently hardcoded and defined in dynamicAgentTypesGenerator.h

---

**AgentBlueprintProvider**

Compositional Part: AgentBuildInformation 1/2
Sensor OSI

Sensor Fusion

- Single Instance
- Only 1 Local LinkId Single Signal Type
- Same here

Some Vehicle Component

I need a camera sensor
I need a camera and ultrasonic sensor
I need a radar sensor
Sensor Fusion

Start with original LinkId and increment

Filter based on given parameter set

Sensor Link0: Camera

Parameter Set: SensorLink0: Camera

Sensor Link1: Ultrasonic

Parameter Set: Sensor Link0: Camera SensorLink1: Ultrasonic

Sensor Link2: Radar

Parameter Set: SensorLink2: Radar

Combine all sensors and sent (all) as output

Concept Sensor Fusion

Instantiation