Changes of the Configuration Files
openPASS Release 0.6 PR

04.07.2018 – René Paris, on behalf of BMW AG
Levels of Configurations

**Experiment**
- `frameworkConfig.xml` ➔ `masterConfiguration.xml`

**Features**
- `runConfiguration.xml` ➔ `combinationConfig.xml`
- `scenario.xosc`
- `scenery.xodr`
- ...

**Components**
- **Static Agents:** `systemConfig.xml`
- **Dynamic Agents:** `appConfig.xml`
Analysis PCM-UseCase

**Framework Configuration**

**Task**
- Configuration of Master
- Configuration of Slave Execution and Experiments

**Name Framework**
- Generic placeholder for the **Controlling Components** within Master and Slave, respectively

**Issues**
- Historically grown structure
- Very close to a specific use case
- Every change make modification necessary
  - E.g.: New configuration file necessary
- Increasing load for keeping up compatibility
  - E.g.: Some configuration files not necessary anymore

**Wish**
- Separation of Concerns:
  - Framework Configuration vs. Experiment Configuration
- Open for Extension:
  - Very high level of abstraction
- Closed for Modification:
  - No need for code modifications on changes

**Structure**
```xml
<?xml version="1.0" encoding="UTF-8"?>
<frameworkConfigurations>
  <frameworkConfiguration>
    <SlavePath>...</SlavePath>
    <LogFileMaster>...</LogFileMaster>
    <LogLevel>...</LogLevel>
    <frameworkConfiguration>
      <LibraryPath>...</LibraryPath>
      <AgentConfigFile>...</AgentConfigFile>
      <LogFileSlave>...</LogFileSlave>
      <ResultPath>...</ResultPath>
      <RunConfigFile>...</RunConfigFile>
      <ScenarioConfigFile>...</ScenarioConfigFile>
      <SceneryConfigFile>...</SceneryConfigFile>
    </frameworkConfiguration>
  </frameworkConfiguration>
  infos for second slave
</frameworkConfigurations>
```

**Note**
If root tag is `FrameworkConfiguration`, only a single slave configuration is loaded directly from beneath the root tag.
Task
- Configuration of Master
- Configuration of Slaves Execution

Changes
- Separation of concerns: Execution / Experiment
- Separation of common/individual Slave Configurations
- Removal of experiment information for slaves:
  E.g. Where are the libraries, but not what libraries are needed for the experiment
- Results: Experiment related changes do not change config of the master

Slave related control information
- Each entry is a string passed to the Slave via command line
- Slave decides what to do with that information (see next slide)

Structure
```xml
<?xml version="1.0" encoding="UTF-8"?>
<masterConfiguration>
  <logLevel>...</logLevel>
  <logFileMaster>...</logFileMaster>
  <slave>...</slave>
  <libraries>...</libraries>
  <slaveConfigurations>
    <slaveConfiguration>
      <logFileSlave>...</logFileSlave>
      <configurations>...</configurations>
      <results>...</results>
    <slaveConfiguration>
      <infos for second slave>
    </slaveConfigurations>
  </masterConfiguration>

Note
The log level is used by the master but also the slaves
Configuration files are not specified anymore
• The slave now load files from a relative path (current state)
• Or could do something completely different, e.g.
  <Configurations>192.168.0.5:2256?id=5</Configurations>

Results files are not specified anymore
• The slave now write results to a relative path (current state)
• Or could do something completely different, e.g.
  <Results>192.168.0.5:2257?id=5</Results>

Other Stuff
• Omitted tags are automatically defaulted, e.g. logFileSlave in Example on the right (see next slide)
• At least a single SlaveConfigurations/SlaveConfiguration needs to be defined

Example
<?xml version="1.0" encoding="UTF-8"?>
<masterConfiguration>
  <logLevel>2</logLevel>
  <slave>openPassSlave</slave>
  <libraries>lib</libraries>
  <slaveConfigurations>
    <slaveConfiguration>
      <configurations>experiment1</configurations>
      <results>results1</results>
    </slaveConfiguration>
    <slaveConfiguration>
      <configurations>experiment2</configurations>
      <results>results2</results>
    </slaveConfiguration>
  </slaveConfigurations>
</masterConfiguration>

Calls
> openPassSlave.exe --logLevel 2
--logFile OpenPassSlave.log --lib lib
--configs experiment1 --results results1
> openPassSlave.exe --logLevel 2
--logFile OpenPassSlave.log --lib lib
--configs experiment2 --results results2
Generally, parameters specified within the masterConfiguration are forwarded to the slave as command line parameters.

**Master**
- **--config** (masterConfiguration.xml) Path to config

*Note: Omitted parameters are defaulted to values in braces*

**Slave**
- **--logLevel** (0)
- **--logFile** (OpenPassSlave.log)
- **--lib** (lib) Path to the libraries
- **--configs** (configs) Path to the configuration files
- **--results** (results) Path where to put the results

*Note: Omitted parameters are defaulted to values in braces*

**Minimum masterConfiguration.xml**
```xml
<?xml version="1.0" encoding="UTF-8"?>
<masterConfiguration>
  <slaveConfigurations>
    </slaveConfiguration>
  </slaveConfiguration>
</masterConfiguration>
```

**Calls**
```bash
> OpenPassSlave.exe --logLevel 0
   --logFile OpenPassSlave.log --lib lib
   --configs configs --results results
```

*Note*
Due to the matched default values, this call is equivalent to calling openPassSlave.exe directly **without** parameters.
CombinationConfig.xml
(Random seed, number of invocations, Environment, Logging, Traffic, ...)

Scenario.xosc

ProfilesCatalog.xml
(Agents, Drivers, Vehicles, Sensors)

VehicleCatalog.xosc
(and PedestrianCatalog)

Scenery.xodr

AppConfig.xml

SystemConfig.xml

Note
- The name CombinationConfig.xml and AppConfig.xml is fixed, the other files are referenced from within the according files
- AppConfig and SystemConfig are necessary for dynamic and static agent allocation, respectively (see next slide)
- Static Agents: Linking against a system configuration (see next slide)
- Dynamic Agent: Randomly composed of the following profiles:

  - **AgentProfile**
  - **DriverProfile**
  - **VehicleProfile**
  - **Components** (profile ref. and linked to sensor)
  - **Sensors** (profile ref. and position)
  - **ComponentProfile**
  - **SensorProfile**

**Note**
Currently, AppConfig.xml is kind of a "SystemConfig" Tempalte, defining a superset of all possible components and their connections.
<AgentProfiles>
  <AgentProfile Name="EgoAgent" Type="Static">
    <System>
      <File>SystemConfig.xml</File>
      <Id>0</Id>
    </System>
    <VehicleModel>VehicleModelX</VehicleModel>
  </AgentProfile>
  <AgentProfile Name="MiddleClassCarAgent" Type="Dynamic">
    <DriverProfiles>
      <DriverProfile Name="Regular" Probability="1.0"/>
    </DriverProfiles>
    <VehicleProfiles>
      <VehicleProfile Name="VehicleModelA" Probability="0.4"/>
      <VehicleProfile Name="VehicleModelB" Probability="0.3"/>
      <VehicleProfile Name="VehicleModelC" Probability="0.3"/>
    </VehicleProfiles>
  </AgentProfile>
</AgentProfiles>
Current State

OpenSCENARIO Importer

Catalogs
VehicleCatalog, PedestrianCatalog can be imported (pedestrians are currently handled as vehicles)

RoadNetwork
Reference to scenery file is imported from RoadNetwork/Logics

Entities
• Can be imported
• Special entity object Ego
• Objects specify catalog reference and catalog entry name
  ① Deviation from standard: Reference of custom catalog „ProfilesCatalog.xml“
• Selections (groups) of entities can be defined, but currently the special selection „ScenarioAgents“ is used for spawning

Storyboard Parsing

Init
• Import of initial dynamics of agents (position, velocity, acceleration)

Story
• Actor Entities can be referenced
• Maneuvers can have a UserDefined action named ComponentStateChangeManipulator
  Command: SetComponentState
  <ComponentName> <Max. ComponentState>
• StartConditions parsed partially

Condition/ConditionGroup
• SimulationTime condition („ConditionalEventDetector“)  
  can be used in Maneuver StartConditions and Storyboard EndConditions
• Only condition type currently supported