CONCEPT AGENT INTERFACE
OPENPASS

Status Quo: Bloated Interface

Set of unstructured, non-cohesive methods

⇒ Class AgentInterface (AI) inherits from class WorldObjectInterface (WOI)
⇒ 197 public methods concerning different fields
⇒ Some methods overridden in AI and WOI (e.g. Locate)
⇒ Or split up: SetLength in AI and GetLength in WOI
⇒ WOI contains only Getter and Locate/Unlocate
⇒ VehicleModelParameters is globally defined struct that contains vehicle specific parameters that also have individual Getters and Setters
Refactor AgentInterface

• Group related methods / Increase cohesion
• Whenever possible, declare methods with const keyword
• Specify methods unambiguously in AgentInterface code and as far as possible in names of methods, regarding:
  • Units
  • Coordinate systems
Refactor AgentInterface: Interface Segregation

- Interface segregation principle:
  
  CLIENTS SHOULD NOT BE FORCED TO DEPEND UPON INTERFACES THAT THEY DO NOT USE.

- Specific solutions:
  - Divide interface into smaller ones and let AgentInterface inherit them by multiple inheritance
  - Reduce AgentInterface to cohesive set and create new interfaces for respective clients, where the new interfaces can inherit AgentInterface
=> Construct each client with a reference to its particular interface
**Cohesive Sets: Dynamic State**

```c
struct DynamicState
{
    double xPos;
    double yPos;
    double yaw;
    double xVel;
    double yVel;
    double yawVel;
    double xAcc;
    double yAcc;
    double yawAcc;
};

enum class ReferenceFrame
{
    World = 0;
    AgentStaticFrame; // origin in the vehicle position (ReferencePoint), static relative to the world frame
};

enum class ReferencePoint
{
    GeometricalCenter = 0; // Center of length and width
    CenterOfMass;
    CenterOfRearAxle; // Only for vehicles
};
```
Cohesive Sets: Internal States and AgentSignals

```cpp
struct InternalStaticState {
    double spawnTime; // or obsolete
    std::string DriverProfileName;
    std::string AgentTypeName;
    std::string ScenarioName;
    AgentCategory agentCategory;
};

struct InternalDynamicState {
    int gear;
    double steeringWheelAngle;
    double distanceTraveled;
    double engineSpeed;
    double speedGoalMin;
    double effAccelPedal;
    double effBrakePedal;
};

struct AgentSignals {
    IndicatorState indicatorstate;
    LightState lightstate;
    bool brakeLight;
    bool horn;
    bool headlight;
    bool highBeamLight;
    bool flasher;
};
```
Process of extending interfaces

Extend without modification (Open-Closed Principle)

• Adapter Pattern, Visitor Pattern, ...