Georgia Tech Research Institute
Problem. Solved.

Conceptual Data Modeling for the Functional Decomposition of Mission Capabilities

February 27, 2018

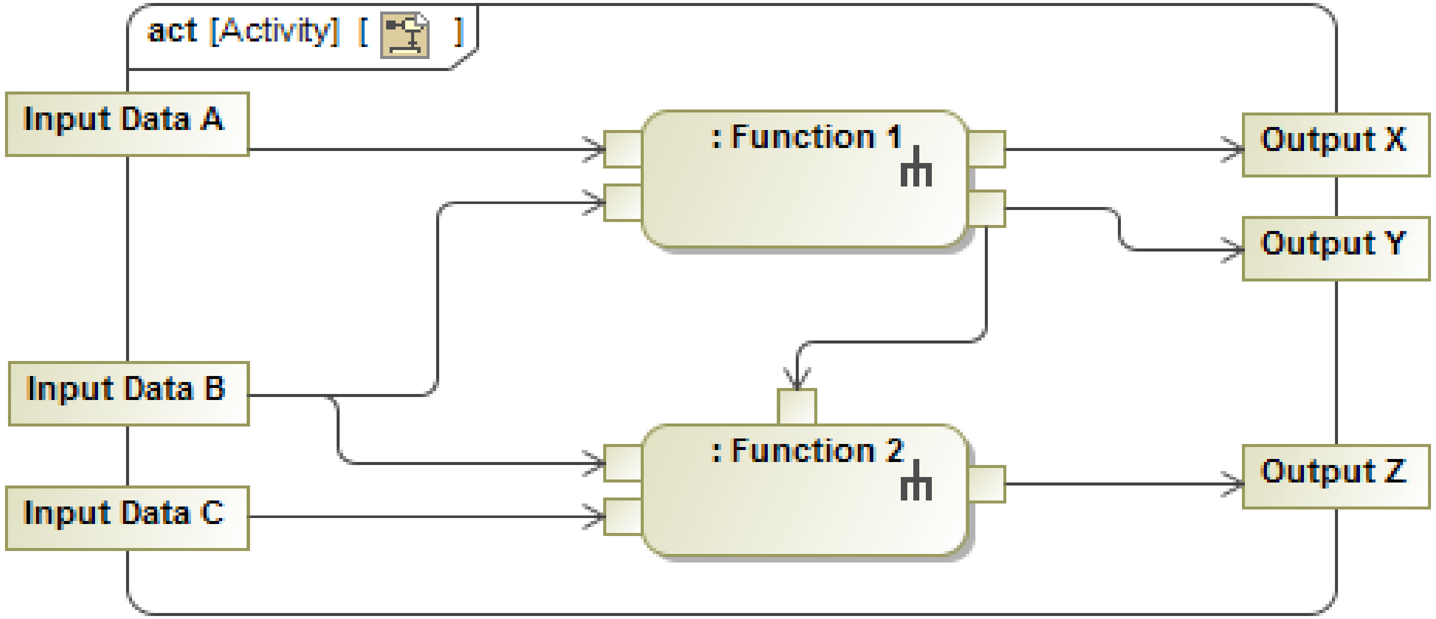
Andrew Battigaglia

Andrew.Battigaglia@gtri.gatech.edu

Motivation – Describing Data

- The purpose of a functional decomposition for mission capabilities is to break down complex activities (e.g., noncombatant evacuation operations, landing operations) into more manageable functions
- While these functions provide a greater understanding for a wider audience, the data elements are often simply labeled using words
- In some cases, a dictionary is employed to encourage consistent phrasing and identical data elements are reused where appropriate
- Interpretation left up to a human to discern subtle differences or make real connections between the elements

Activity Diagrams



Typical representation of the elements of functions after a functional decomposition

Conceptual Data Modeling

- Commonly used to document software engineering requirements through the identification of entities, relationships, and possible attributes of these items
- Can be applied in systems engineering by connecting data elements (inputs and outputs) of an implementation-agnostic functional decomposition to a set of common entities and relationships to assist in the consistent documentation of these functions
- Provides a clear way for a functional decomposition to define relationships between data entities and to identify the attributes of those relationships
- Both data and functional modeling are concerned with capturing and representing familiar and related aspects of the real world

Goals

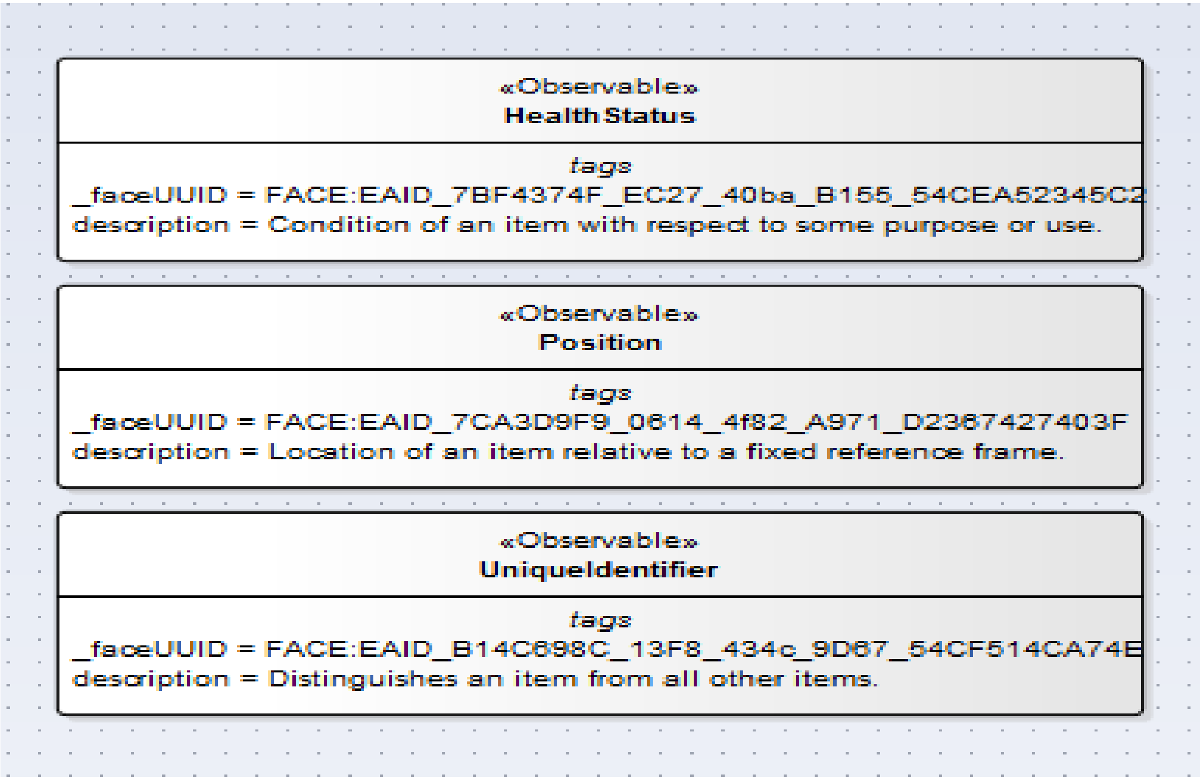
- **Consistent Interpretation.** Using a conceptual data model which contains entities, relationships, and attributes that are formally structured removes the question of whether multiple functions are referencing some of the same concepts
- **Function Reusability**. By employing the principles of conceptual data modeling, functions can be readily identified as requiring greater inspection and probable change if they contain details suggesting a specific implementation
- **Data Reusability.** When a conceptual data model is created to represent function data elements, the work done can be leveraged with one or more functional decompositions of mission capabilities

Approach

- A conceptual data modeling approach using the Future Airborne Capability Environment (FACE) Data Architecture was used
- A specific data architecture was selected in order to ensure consistent terminology, a standard metamodel, and a set of constraints to ensure interoperability between data elements and to allow future expansion for additional mission capabilities
- These benefits of a formal data architecture also assist in the validation effort to ensure that the final data model contents conform to the rules of the selected approach
- The FACE Data Architecture is built around four main concepts: observables, entities, associations, and views

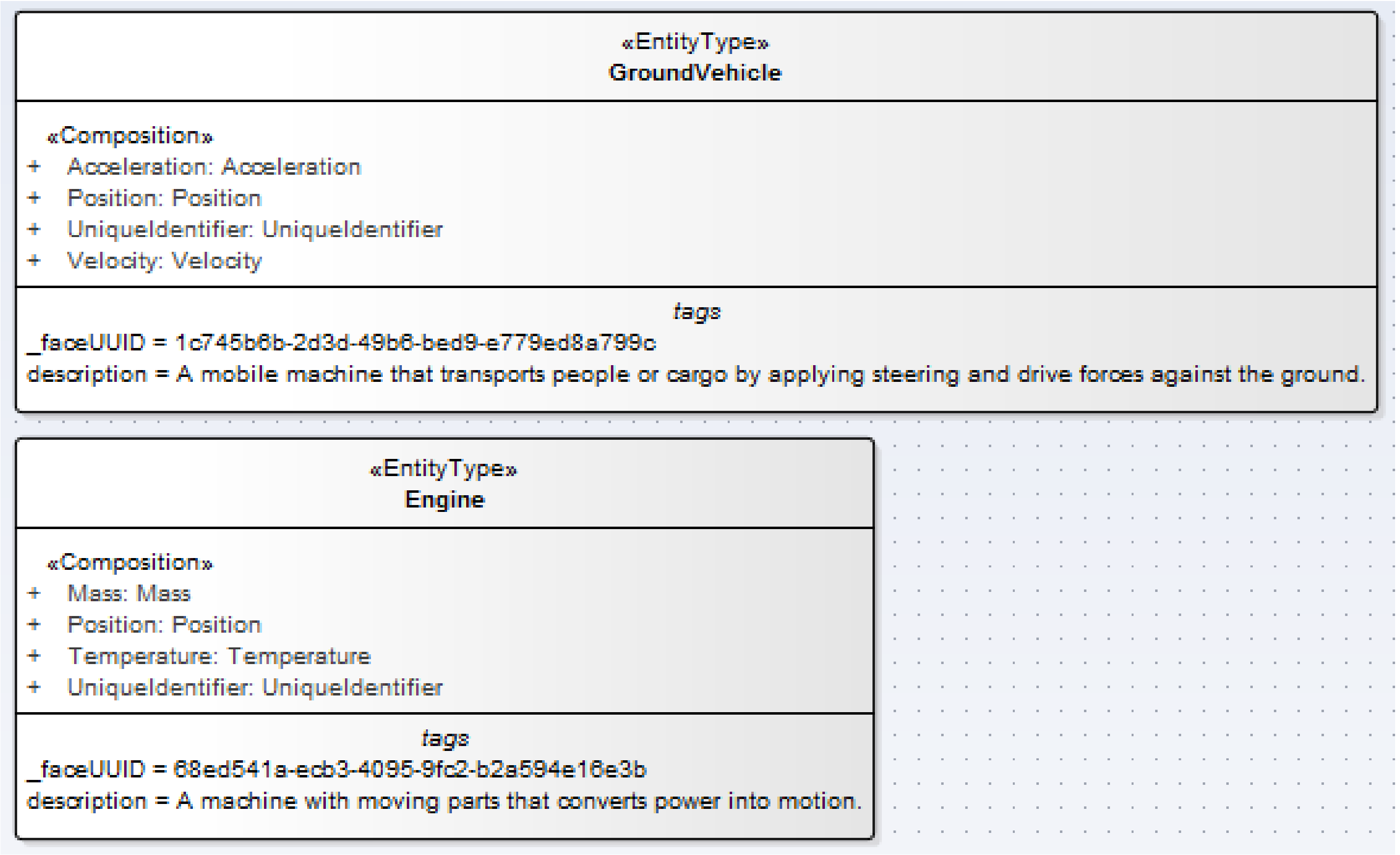
Observables

- Items that can be observed but not further characterized, and is typically obtained through measurements of the physical world



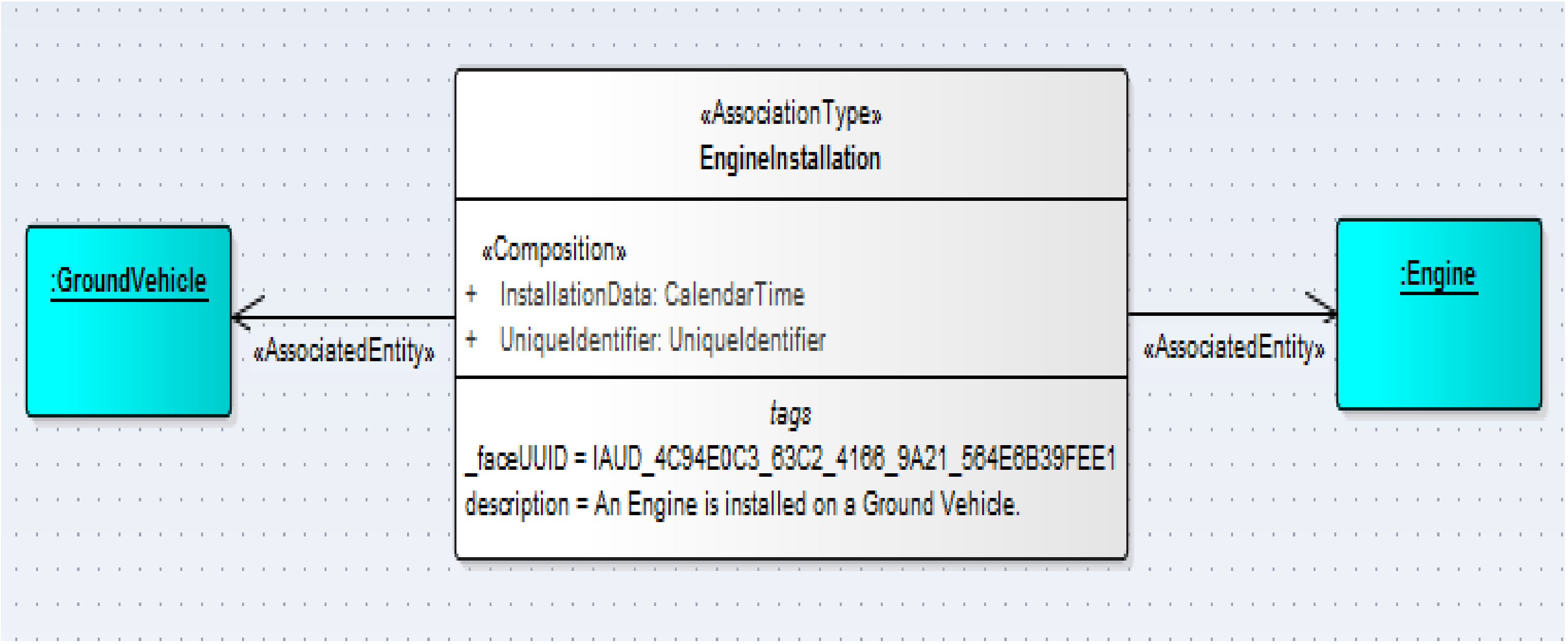
Entities

- Things that are capable of an independent existence that can be uniquely identified



Associations

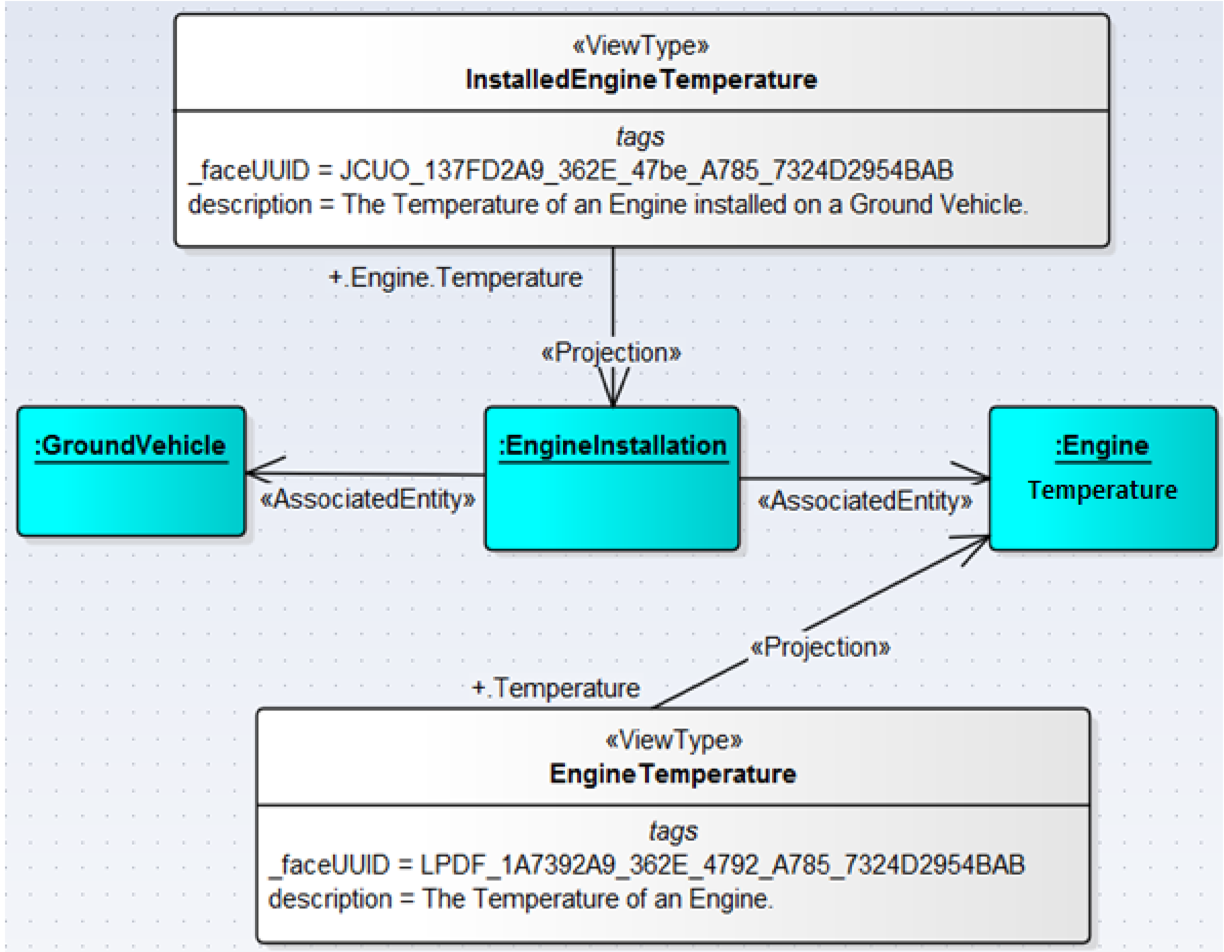
- Used to model the relationships between Entities



Views

- With the entity-relationship model in place, a mechanism is required to adequately select the relevant parts that are required by the different data elements of a functional decomposition
- Views work by projecting an entity, association, or observable in the respective concept’s context, typically accomplished through a projection path
- A functional decomposition is able to directly reference each view using the respective UUID and view name
- Using dot notation for the projection path, only the relevant elements necessary for this view are extracted from the model and become readily available to represent the data consumed or produced by a function of a mission capability

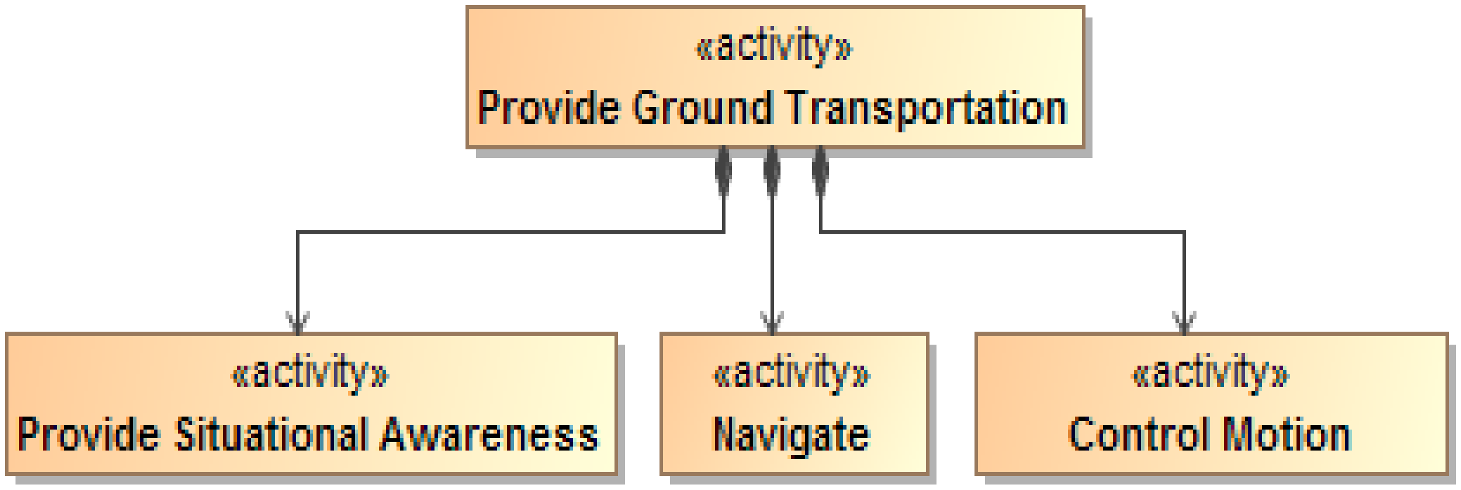
Views


«ViewType»
InstalledEngineTemperature
tags
_faceUUID = JCUO_137FD2A9_362E_47be_A785_7324D2954BAB
description = The Temperature of an Engine installed on a Ground Vehicle.
+.Engine.Temperature
«Projection»
:GroundVehicle
:EngineInstallation
:Engine
Temperature
«AssociatedEntity»
«AssociatedEntity»
«Projection»
+.Temperature
«ViewType»
EngineTemperature
tags
_faceUUID = LPDF_1A7392A9_362E_4792_A785_7324D2954BAB
description = The Temperature of an Engine.

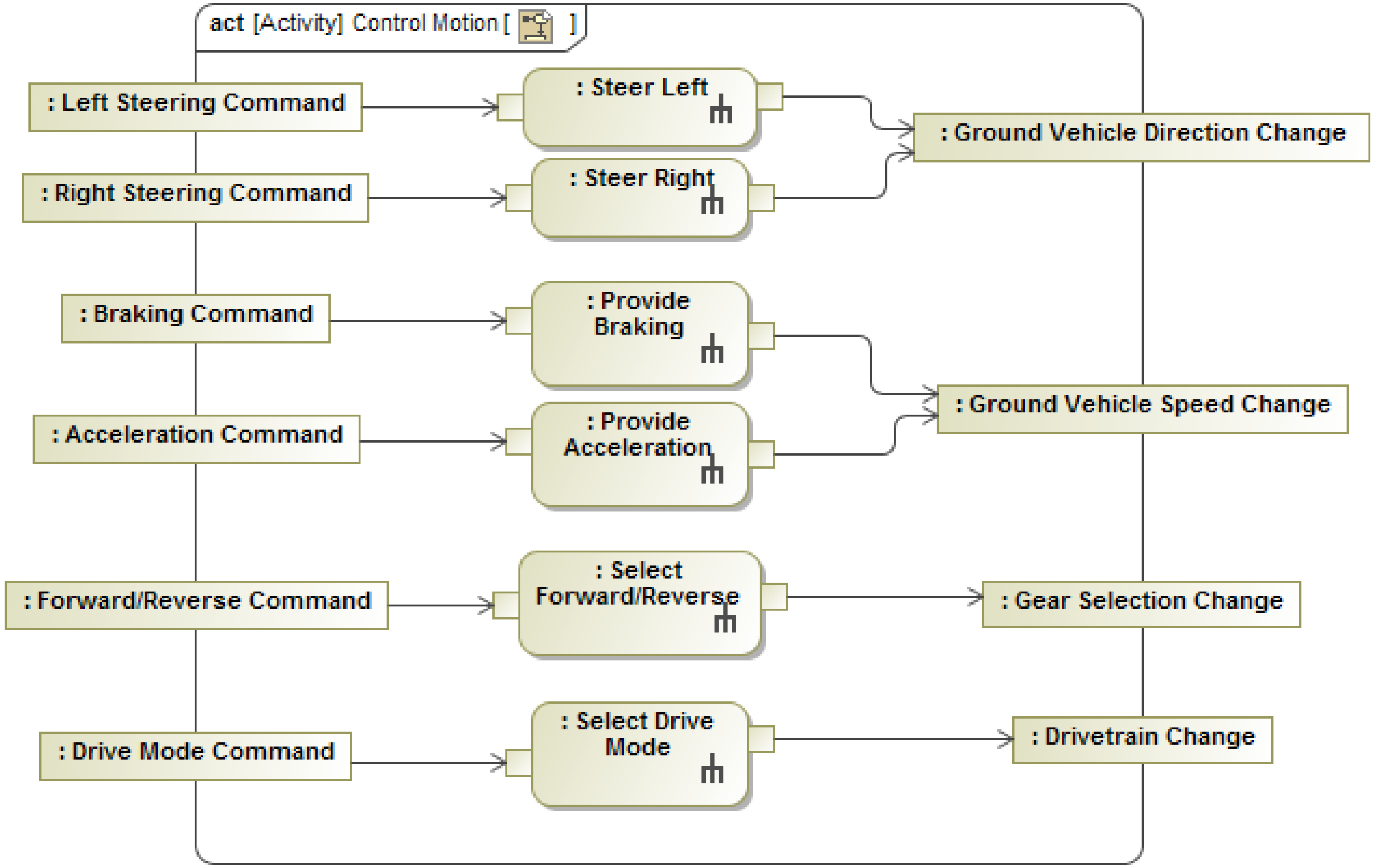
Illustrating the Approach

- Creating or leveraging a data model assists in building better functions by cross-checking the data model with function data elements and then refining the two concurrently
- “Provide Ground Transportation” is defined as the “ground support transportation function that includes moving and transferring units, personnel, equipment and supplies by vehicle to support the operations”
- Scope of this mission capability includes software functionality for determining the ground vehicle’s situation, deciding how to navigate the vehicle from where it is to where it needs to be, and controlling the motion of the vehicle
- Other functionality, such as loading the vehicles and performing maintenance operations, was excluded

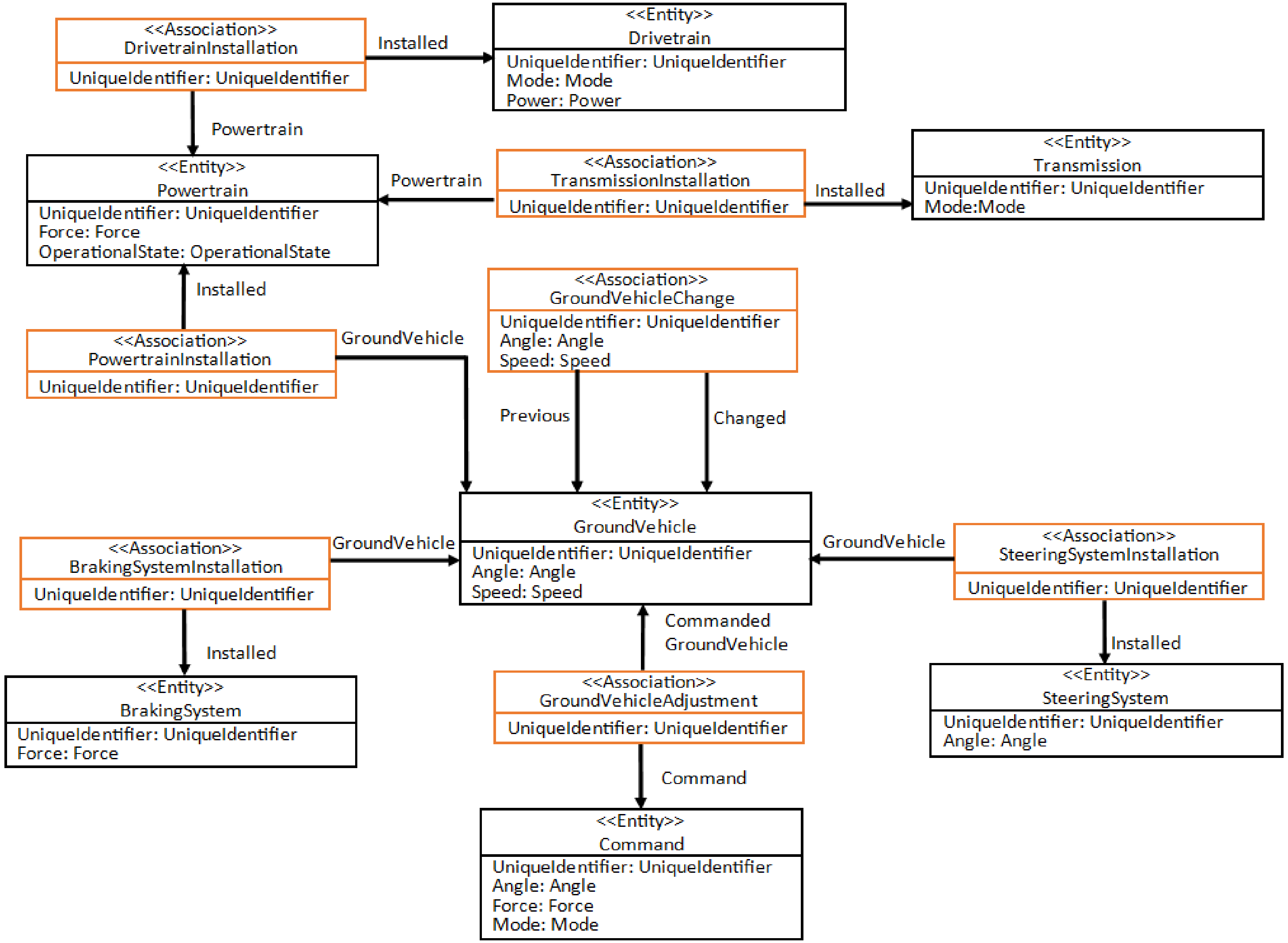
Top Level Functions



Original Control Motion Activity

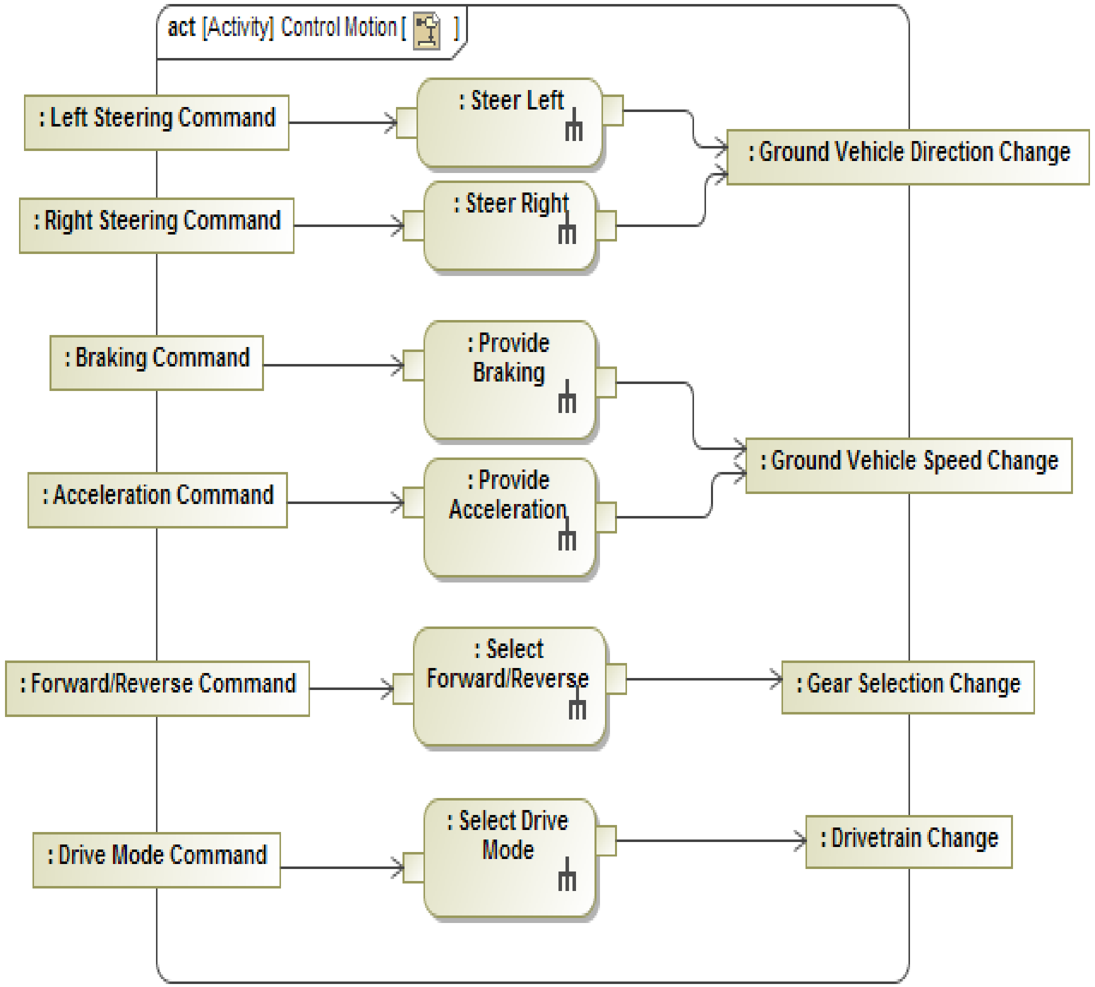


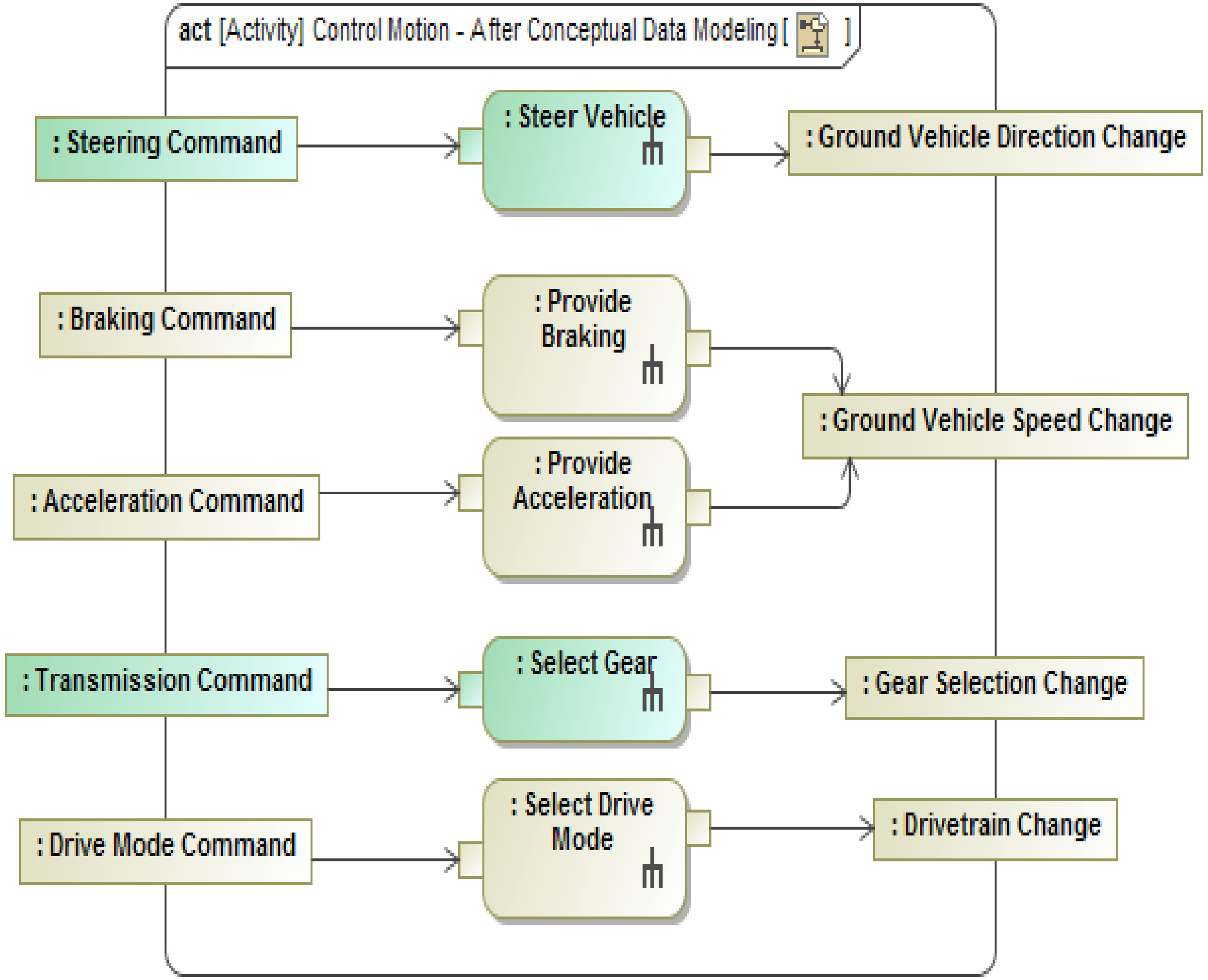
Constructing the Data Model



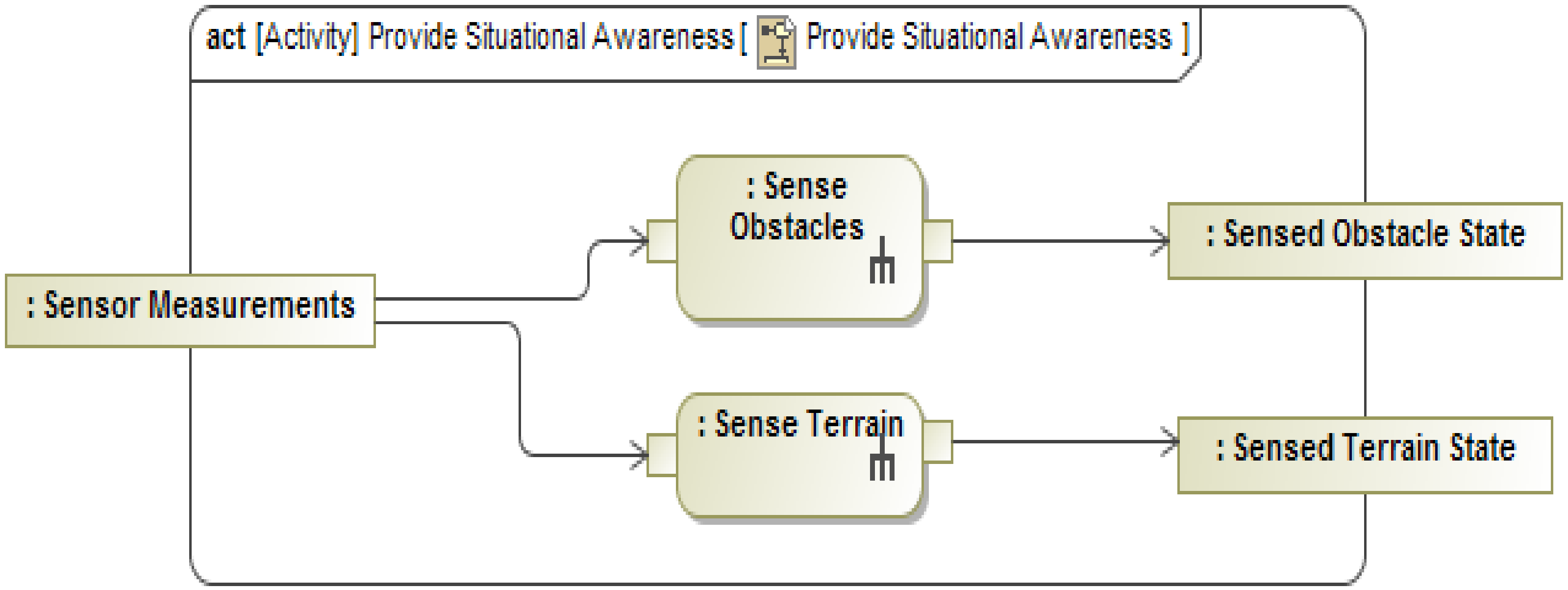
View (with View Characteristic)	Data Model Entry Point	Path
<<View>> **SteeringCommand** Angle	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[SteeringSystemInstallation].Installed.Angle
<<View>> **BrakingCommand** Force	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[BrakingSystemInstallation].Installed.Force
<<View>> **AccelerationCommand** Force	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[PowertrainInstallation].Installed.Force
<<View>> **TransmissionCommand** Mode	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[PowertrainInstallation].Installed ->Powertrain[TransmissionInstallation].Installed.Mode
<<View>> **DriveModeCommand** Mode	Command	->Command[GroundVehicleAdjustment].Commanded GroundVehicle->GroundVehicle[Powertraininstallation] .Installed->Powertrain[DrivetrainInstallation].Installed.Mode
<<View>> **GroundVehicle DirectionChange** Angle	GroundVehicle Change	.Changed.Angle

Control Motion Activity Post-Data Model

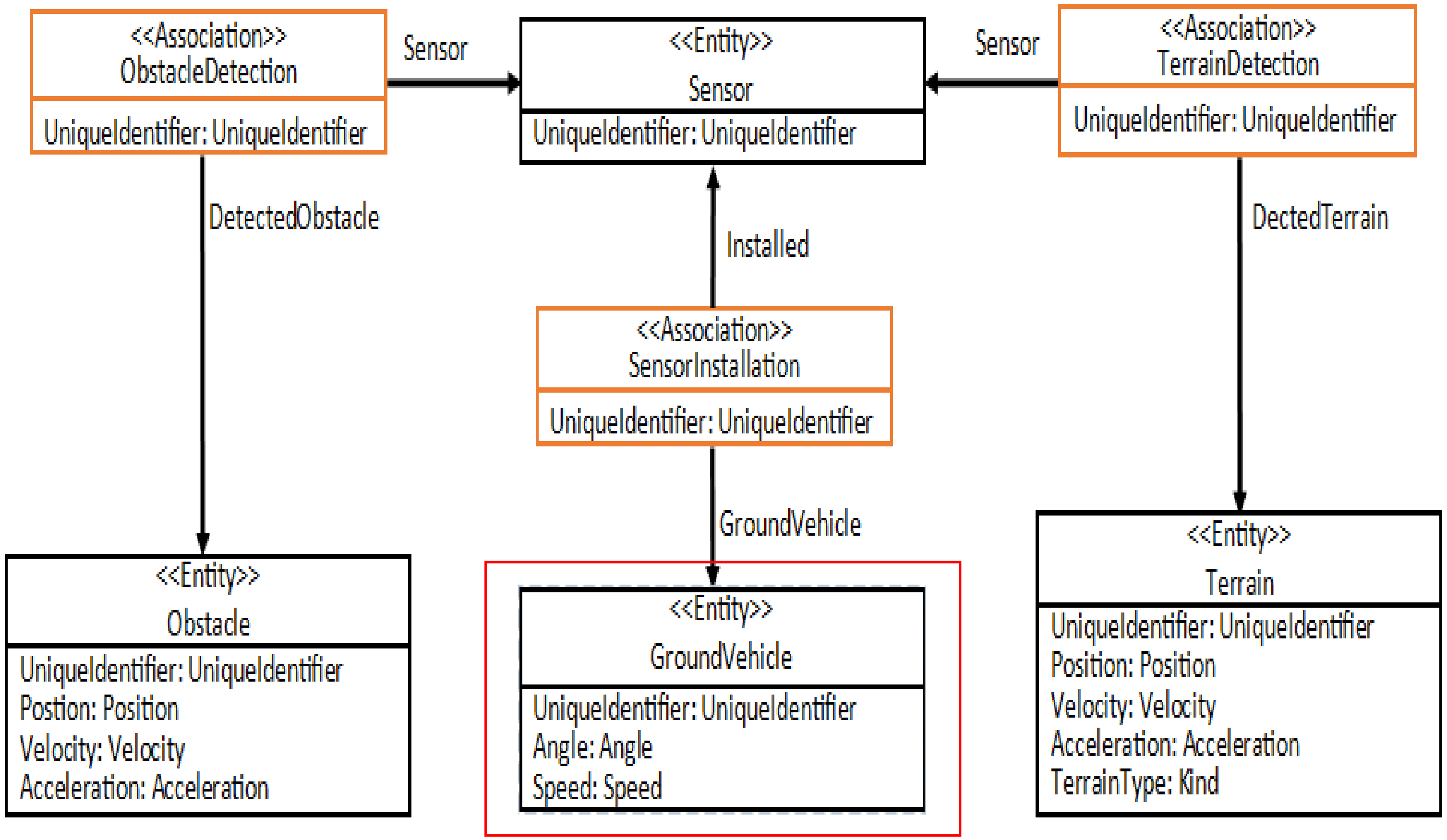




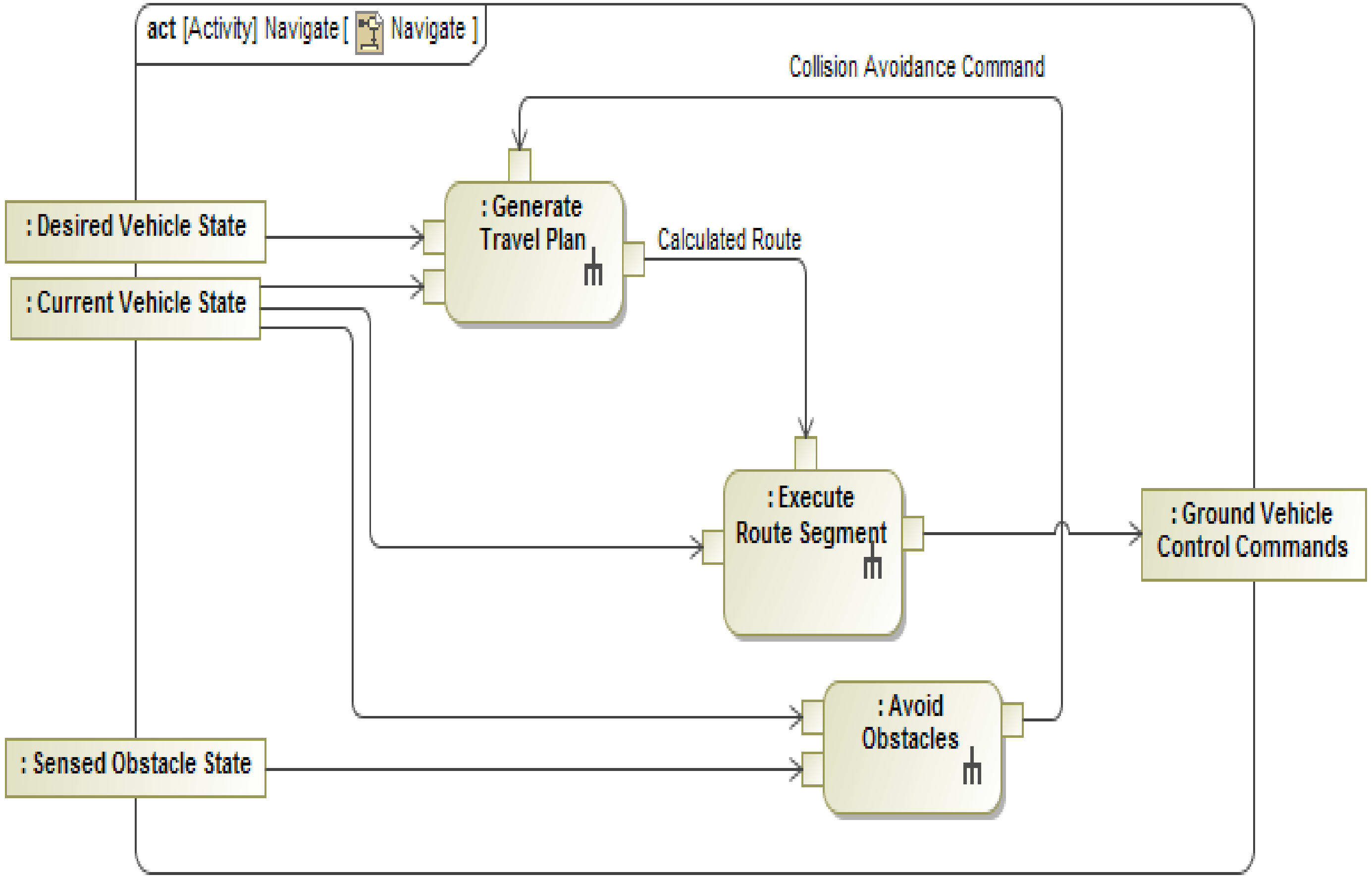
Provide Situational Awareness Activity



Expanding the Data Model



Navigate Activity



View	Characteristic	Entry Point	Path
<<View>> GroundVehicle ControlCommands SteeringCommand BrakingCommand AccelerationCommand TransmissionCommand DriveModeCommand	Steering Command	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[SteeringSystemInstallation].Installed .Angle
	Braking Command	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[BrakingSystemInstallation].Installed .Force
	Acceleration Command	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[PowertrainInstallation].Installed .Force
	Transmission Command	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[PowertrainInstallation].Installed ->Powertrain[TransmissionInstallation].Installed.Mode
	DriveMode Command	Command	->Command[GroundVehicleAdjustment] .CommandedGroundVehicle ->GroundVehicle[Powertraininstallation].Installed ->Powertrain[DrivetrainInstallation].Installed.Mode

Next Steps

- Next phase is to begin using existing functions to create new system implementations
- To continue using a conceptual data model for specific applications, it will have to be expanded to the logical and physical schemas
- Constructing a data model that contains conceptual, logical, and physical levels entails describing the attributes, or observables, in terms of units, frames of reference, and physical datatypes
- A benefit of taking this next step is that it will become apparent if an entity, association, or observable is missing from the conceptual data model because a specific implementation might require a physical datatype that does not appropriately link to the conceptual level

Questions?