SRR Objects

This hobby report tries to provide an overview about all SRR objects.

Version 4.3 is indicating the version at the first serious specification of the release "Arimathea" (step 0033.11), where the basic concepts of "Arimathea" seem to be stable now.

1 Introduction

SRR objects are nothing more than MIDAS objects that can be used together with the SRR Framework.

- SRR Framework = SMUOS Framework + Train Manager Extension

The SMUOS Framework is explained in the HR 007 "New Concept Paper" (Chapter 11), where the Train Manager extension is explained here in the present report.

Most SRR objects are dependent on the Train Manager Extension (TME) and they help

- to model tracks and switches,
- to model rail vehicles (TODO11, TODO12),
- and to link all of them together functionally (TODO11, TODO12).

SRR objects already existed at the first LAN party in March 2010 and with the re-design that has been ongoing since then, there are no big changes to them:

- It will not be possible to couple vehicles at the end of Step 0033 and collisions will not be considered.
- The "Interlocking of the 1900s" will not be available
- Real Multibrowser Capability will not be available
- Application of X3D Earth will not be used

Nevertheless, there will be some small additions, which were not available in March 2010.

1. The master avatar container will be abandoned (moved to Simple Scene Controller)
2. Dynamic Modules will be supported
3. "Roles" will be abandoned and completely replaced by "Keys"
4. A MIDAS Base (MIB) will exist to ease implementation of MIDAS Objects
5. A "Beam to meeting place" function will exist
6. The SRR Framework will output status messages occasionally
7. A function will exist to reset all keys
8. There will be changes in the documentation and in the blogs
9. It will be possible to use the SRR Framework from web spaces (monolithic layouts)
10. The base module of the SRR Framework will be replaced by the SMUOS Framework
11. Unbound Objects (UBOs), Handover and Moving Modules will be available (TODO11)

2 Content

Content of the Paper

3 Literature

[1] Dissertation; Hürlimann, D.; 2002;
Objektorientierte Modellierung von Infrastrukturelementen und Betriebsvorgängen im Eisenbahnwesen;
Institut für Verkehrsplanung und Transportsysteme, ETH Zürich, Zürich.

4 What is a colon graph?

If you want to simulate operations in the railway industry, then the dissertation of Dr. techn. D. Hürlimann at the ETH Zurich, [1], always a good clue.

Following these approaches, the SRR Framework models the track topology using a so-called colon graph.

Therefore we conducted a "Rollercoaster Project" in year 2008 and came to following conclusions:

Conclusions from the "Rollercoaster" Project

1. the track layout will be modeled by double-nodes and edges
2. each axle of the trains will be moved independently over the mesh of track nodes and track edges, based on "deltaEss" events
3. each axle will have a "transform" property that is updated every frame
4. the position and orientation of the vehicles will be derived from the "transform" properties of all/some of their axles
5. each track edge will have geometric properties that allow to calculate the position and orientation of each axle, based on the axle's properties "ess", "parentEdge", "isAtoB" and "inverse", where
    1. "ess" is the current position of the axle, relative to the parent edge
    2. "parentEdge" is the current parent edge of the axle
    3. "isAtoB": if "isAtoB"=true, then "deltaEss" will be added to "ess", otherwise it will be subtracted
    4. "inverse": indicates, whether the axle is inserted "in backward direction" into the vehicle. This is important, if the rotation of the axle is visible or if the axle is not symmetric.
6. switches are modeled by switch objects (exactly SrrSwitchB objects) that are the children of track nodes. The "actualState" of the switch will be taken as input for the routing of axles

So you can represent the most important track types:

- Track section
- Two-way switch
- Three-way switch – TODO11
- Crossing – TODO11
- Switch-crossing – TODO11
- Bumper – TODO11
- Turntable, transfer table – TODO11
- Rail tracks on rail vehicles (for example, to transport another rail vehicle) – TODO11

Figure 1 shows an example of a colon graph for two track sections, a two-way switch and a bumper.

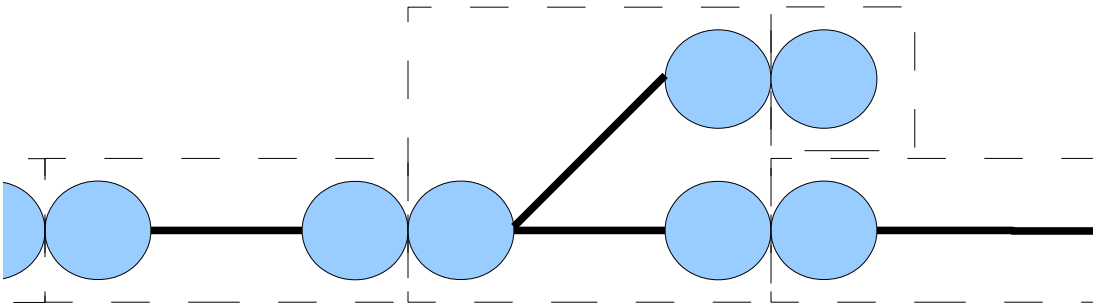

Figure 1: Track Elements in a Colon Graph

The node representing the bumper does not terminate an edge, the end nodes of a normal track segment each terminate one edge, and the node modeling a vertex terminates two edges.

In order to model such colon graphs, the SRR Framework provides the SRR objects for tracks and turnouts.

These are:

- SrrTrackEdge.....a track edge
- SrrTrackNode.....a track node
- SrrSwitchB.....a point machine
- SrrBasicTrackSection.....orchestrates 2 track nodes and 1 track edge
- SrrBasicTurnout2Way.....orchestrates 3 track nodes, 2 track edges and 1 point machine

You can learn more about them in chapter 6 .

5 Topology versus Geometry

As you can easily see, the colon graph is just the logical topology of a track layout, but not geometric or even geographic features are shown, even if only the geometric length of an edge in meters.

For this purpose, the SRR Framework defines an interface with which you can assign geometrical properties to each edge (the SRR object SrrTrackEdge), the so-called "track geometry".

The SRR Framework comes with an "exemplary track geometry" that defines a "track geometry node" (TGN) and some finished track and turnout models that build on this "track geometry node".

More about this "exemplary track geometry" can be found in chapter 7.

6 SRR Objects for Tracks and Turnouts

6.1 Overview

Figure 2 shows an overview of the relationships between the various SRR objects for tracks and switches:

- A "Basic Track Section" orchestrates exactly one "Track Edge" and two "Track Nodes"
- A "Basic Turnout 2-Way" orchestrates exactly two "Track Edges", three "Track Nodes" and one "Point Machine"
- Each track edge contains one and only one TGN

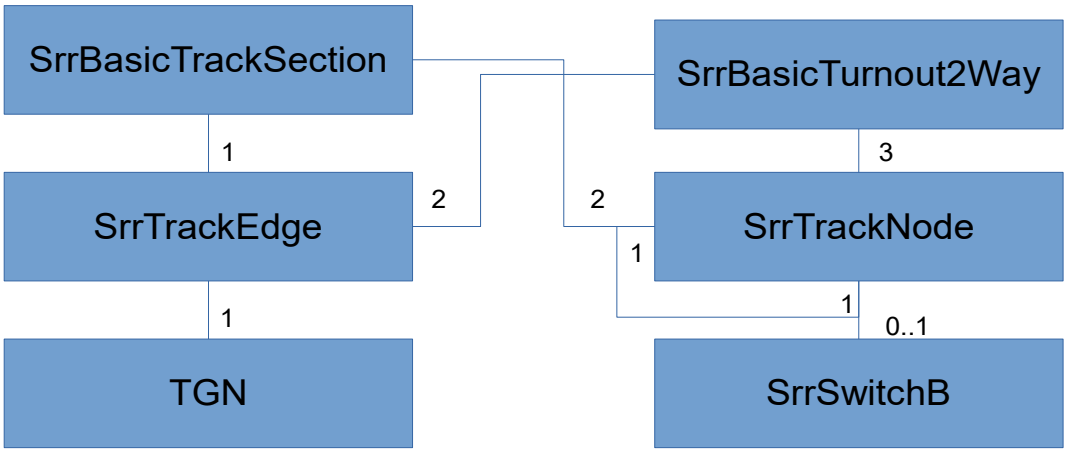

Figure 2: Overview about SRR Objects for Tracks and Turnouts

The track and turnout SRR objects only support the MOOs "LOADED", II and V, which means they can only be used as bound objects. Hence the models of tracks and turnouts are always Bound Models.

The fields of the uiObj interface do not offer the following fields:

- "universalObjectClass", "commParam" and "initialized",

while the following fields are actually offered:

- objId, modParam, disable, parentObj, attached, enabledOut, and dependentMobs.

6.2 SrrTrackNode

An "SrrTrackNode" SRR object models a track node in the colon graph.

It always has one of the two objects SrrBasicTrackSection or SrrBasicTurnout2Way as parent object and offers the following parameters:

Let N be the number of track edges terminated by this node.

The field name = 'neighborName' type = 'SFString' value = ''

contains the "objId" of the desired neighbor node with which the node is to be linked when "linking the tracks". This field is from accessType = "initializeOnly" and thus can only be set once.

The array <field name = 'trackEdges' type = 'MFNode' value = '' />

contains pointers to all N edges that are terminated by this node. This field is set to the correct values during the (re-) attachment of the parent object and can already be used when "linking the tracks".

The array name = 'isNodeA' type = 'MFBool' value = ''

For all N edges terminated by this node, it indicates whether it is the A-side or the B-side of the edge. This field is set to the correct values during the (re-) attachment of the parent object and can already be used when "linking the tracks".

The field name = 'neighbor' type = 'SFNode' value = 'NULL'

points as a pointer to the neighboring node of this node. This field is NULL if and only if the node is not linked to any neighbors. This field is set by "linking the tracks" and reset to "null" when the node is "leased". This also applies if the neighboring node is left behind by this node

The field name = 'turnoutSwitch' type = 'SFNode' value = 'NULL'

contains a pointer to the "turnout drive" when N is greater than 1. If N is less than or equal to 1, this pointer contains the value NULL.

The field name = 'gauge' type = 'SFFloat' value = '1.435'

contains the gauge of the track that is valid at this node. Neighboring nodes should always have the same gauge, but along one edge, a running gauge can be modeled by setting different gauges at the A-node and the B-node of the edge.

6.3 SrrTrackEdge

A "SrrTrackEdge" SRR object models a track edge in the colon graph.

It always has one of the two objects SrrBasicTrackSection or SrrBasicTurnout2Way as parent object and offers the following parameters:

The field name = 'trackNodeA' type = 'SFNode' value = 'NULL'

contains a pointer to the track node on the A side of the track edge. This field is set at the "Track linking" when the node is linked to its neighbor node. When the node is "unlinked", this field is reset to the value NULL. This also applies if the neighboring node is left behind by this node.

The field name = 'trackNodeB' type = 'SFNode' value = 'NULL'

contains a pointer to the track node on the B side of the track edge. This field is set at the "Track linking" when the node is linked to its neighbor node. When the node is "unlinked", this field is reset to the value NULL. This also applies if the neighboring node is left behind by this node.

The field name = 'trackGeometry' type = 'SFNode' value = 'NULL'

contains a pointer to a TGN. Each "normal" track edge must be assigned exactly one TGN, which deals with the geometric and geographic properties of the track edge.

The field name = 'exitViewpoint' type = 'SFNode' value = 'NULL'

contains a pointer to an X3D <Viewpoint> node.

If a user enters a rail vehicle, then one must "bind" a <Viewpoint>, which moves with the rail vehicle, so that the user is "taken" by the vehicle. Now, when the user leaves the vehicle, you have to bind a <Viewpoint> again, which rests opposite the landscape.

This field allows the scene author to associate each track edge with a <Viewpoint>, which is bound by the SRR Framework as soon as the user leaves the vehicle and when the vehicle is just entering this track edge.

6.4 Track Geometry Nodes (TGNs)

With the TGN, the SRR framework accesses the geometric properties of a track edge.

The TGN can also perform calculations to give the surrounding track model a basis for the graphical representation of the track.

This is best seen by looking at the file tg/SrrTrackSectionA.x3d.

This implements a model of a track section that corresponds to a track edge using the "ABI track geometry". The "ABI track geometry" is defined in the TGN "SrrTrackGeometryABI.x3d".

Here you can see first the SRR objects and then the graphic elements, which then really become visible:

```
<!--  ***** SRR objects to instrument the track section *****  -->
                <ProtoInstance DEF='TrackSection' name='SrrBasicTrackSection'>
                    <fieldValue name="dependentMobs">
                        <ProtoInstance DEF="TrackNodeA" name="SrrTrackNode">
                            <fieldValue name="gauge" value="0.545"/>
                            <IS>
                                <connect nodeField="neighbourName" protoField="neighbourA"/>
                            </IS>
                        </ProtoInstance>
                        <ProtoInstance DEF="TrackNodeB" name="SrrTrackNode">
                            <fieldValue name="gauge" value="0.545"/>
                            <IS>
                                <connect nodeField="neighbourName" protoField="neighbourB"/>
                            </IS>
                        </ProtoInstance>
                        <ProtoInstance DEF="TrackEdge" name="SrrTrackEdge">
                            <fieldValue name="trackGeometry">
                                <ProtoInstance DEF='TrackGeometry' name="SrrTrackGeometryABI">
                                    <fieldValue name="trackElementLength" value="0.5"/>
                                    <IS>
                                        <connect nodeField="vectorA" protoField="vectorA"/>
                                        <connect nodeField="vectorB" protoField="vectorB"/>
                                        <connect nodeField="vectorI" protoField="vectorI"/>
                                        <connect nodeField="normalA" protoField="normalA"/>
                                        <connect nodeField="normalB" protoField="normalB"/>
                                    </IS>
                                </ProtoInstance>
                            </fieldValue>
                            <IS>
                                <connect nodeField="exitViewpoint" protoField="exitViewpoint"/>
                            </IS>
                        </ProtoInstance>
                    </fieldValue>
                </ProtoInstance>
<!--  ***** graphical representation of the track section *****  -->
                <Switch DEF="OnOffSwitch">
                    <Shape>
                        <TriangleStripSet DEF='VisualTrackGeometry' stripCount="3">
                            <Coordinate DEF='VisualTrackCoordinates' point='0 0 0, 1 0 0, 0 1 0'/>
                            <TextureCoordinate DEF='VisualTrackTextureCoordinates' point='0 0, 1 0, 0 1'/>
                        </TriangleStripSet>
                        <Appearance>
                            <Material />
                            <ImageTexture url='"../tg/trackElementA.jpg"'/>
                        </Appearance>
                    </Shape>
                </Switch>
            </Group>
```

The TGN (marked in red) is inserted here as the child of the SrrTrackEdge node and receives the values "vectorA", "vectorB", "vectorI", "normalA" and "normalB" from the user of the model.

These values are then used to calculate a circle segment that is the base of the track section.

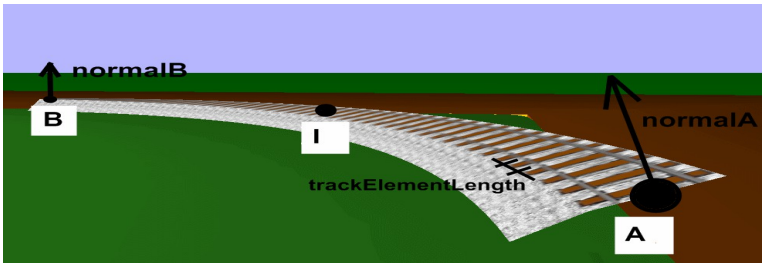

Figure 3: Parameter of the ABI track geometry

If you initialize the TGN, then it takes these parameters and the parameter "trackElementLength", first calculates the circle section internally and the external parameter "length", divides the length "length" into a number of track elements and outputs the parameters "trackCoordinates", "alongVectors" and "normalVectors".

These output parameters are arrays of vectors used by a script (not shown here) to set the geometric properties of the <TriangleStripSet /> before it is made visible using the OnOffSwitch switch.

The <ImageTexture /> node is scaled so that the image "tg/trackElementA.jpg" is displayed exactly "length" / "trackElementLength" times.

Each TGN (not just the TGN of the "ABI track geometry") MUST offer the following parameters:

The field name = 'objId' type = 'SFString' value = ''

serves to assign a <objId> to the TGN.

The field name = 'parentObj' type = 'SFNode' value = 'NULL'

serves to determine the <objId> s of the parent objects. Thus, a compound <objId> can be generated, e.g. for tracer issues.

The field name = 'modParam' type = 'SFNode' value = 'NULL'

is used to (re) initialize the TGN. Here, the term "(re) initialization" is used because a TGN is not really an SRR object (usually we use the term "attachment", when triggering with "modParam").

With the field name = 'attached' type = 'SFNode'

the TGN must tell if the (re-) initialization was successful. The value initialized = NULL means that the (re) initialization has gone wrong.

The field name = 'enabledOut' type = 'SFBool'

must be set to false if the TGN has been disabled.

The TGN must have the field name = 'length' type = 'SFFloat' value = '0.0'

set to the correct value at the latest during initialization.

With the field name = 'calculateAxleTransformation' type = 'SFNode'

one instructs a TGN to represent an axle relative to the parent edge of this TGN. If you pass the pointer "Value" to an axle on the TGN, then the TGN must evaluate the properties of the axle and then set the Value.transformation.transformation property of that axis to the correct value.

With the field name = 'disable' type = 'SFTime'

you can disable the TGN.

6.5 SrrBasicTrackSection and SrrBasicTurnout2Way

These SRR objects are the "parent objects" to "orchestrate" all SRR objects of each track type.

SrrBasicTrackSection orchestrates 2 track nodes and one track edge into a general track section. The track edge must contain a TGN.

SrrBasicTurnout2Way orchestrates 3 track nodes and 2 track edges into a 2-way turnout. The track edges must each contain a TGN and the track node at the point point must contain a point machine.

6.6 Orchestration of the tracks and points

The systematics of the MIDAS objects (in the form of the prototype MibCore) ensures that the dependent MOBs are first initialized and attached - ie

- TGN (Track Geometry Node) - one per track edge,
- SrrSwitchB (point machine) - one per switch,
- SrrTrackNode - 2 pieces at a track section and 3 pieces at a switch and
- SrrTrackEdge - one at a track section and two at a switch,

before the "parent objects" are initialized and attached, ie

- SrrBasicTrackSection
- SrrBasicTurnout2Way

Since the module coordinator (exactly the TMM extension of the module coordinator) keeps both a list of all track edges of the module and a list of all module track nodes, these two nodes will announce themselves during their (re-) attachments to the MC and wait for them Success message before the "parent object" can be further initialized and attached.

SrrTrackEdge additionally checks if it actually contains a TGN.

Afterwards - during the attachment of the "parent object" - this sets some pointers in the Track Edge and Track Node objects, so that then the "linking of the tracks" can take place properly.

The tracks are still not linked, a "wheels of the wheels" is not yet possible.

6.6.1 Linking the tracks – TODO11

"Linking the tracks" is a service of the SRR Module Coordinator (which is an extension of the SMS Module Coordinator Base).

This service will be implemented during step 0033.11.

6.6.2 Unlinking a node – TODO11

"Unlinking a node" is a service of the SRR Module Coordinator (which is an extension of the SMS Module Coordinator Base).

This service will be needed at a later stage of step 0033.11, e.g. for turntables and transfer tables.

7 The "Example Track Geometry"

The SRR Framework (which uses the SMUOS framework) has an intentional void.

From the beginning we have only implemented a very simple, one might say rudimentary, track geometry that reduces each track section to a circular arc.

Although this circular arc is at least spatially designed - so that you can model even hills and valleys in the track - to roller coasters - but the railroad specialist missing the beloved transition bows. These are currently not modelable.

To remedy this, we have "outsourced" the track geometry, so to speak.

This means that we have left a gap in the SRR framework, in which everyone can use his own track geometry, but have added an "example track geometry" to the SRR framework, including some models for track sections and switches, namely the "examplary track geometry".

This is in the directory tg / and is also licensed with an LGPL.

8 SRR Objects for Rail Vehicles – TODO11/TODO12

8.1 Basic Considerations about Trains and Vehicles – TODO11/TODO12

SrrTrains Vehicles and Trains are what-we-call "Unbound Objects" (UBOs).

This means they will be instances of what-we-call "Object Types" (OTs), where each Object Type is a member of a "Universal Object Class" (UOC) and defines a URL, where the UBO can be instantiated from.

UOCs are defined by the SrrTrains project (i.e. by the Train Manager Extension TME of the Simple Scene Controller) as follows:

- UOC SrrVehicleTrain.....a train that consists of exactly one vehicle – TODO11
- UOC SrrVehicle.....a vehicle that can be related to an SrrTrain – TODO
- UOC SrrTrain.....a train that consists of one or more SrrVehicles – TODO
- UOC SrrRelation.....a relation between SrrVehicle and SrrTrain – TODO

OTs are defined by model authors. OTs can be something like

- StephensonsRocket
- OebbRh1044

UBOs are created and deleted during runtime of the simulation, by mechanisms of the SMUOS Framework and with support of the Train Manager Extension (TME).

Each train will have a central SRR Object, which holds the so-called "train state". The train state is related to a "reference axle". Each train has got one and only one invisible reference axle, which defines the position and the orientation of the train, as follows:

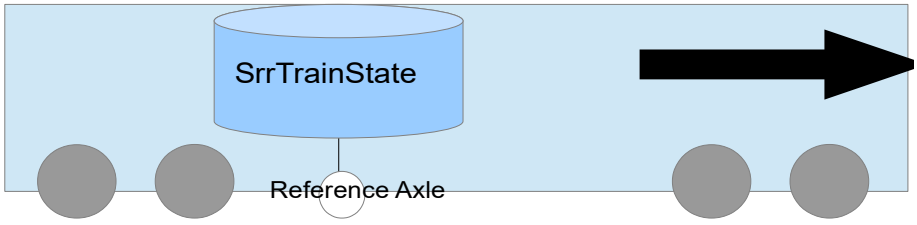

Figure 1: Train State is calculated relative to an invisible "reference axle"

The SrrTrainState is an "Animated MIDAS Object" that cares for the global state of the train

- parent module, parent edge, ess and isAtoB of the reference axle (i.e. the train's position and orientation) – TODO11
- velocity for the railway kinematics – TODO11
- acceleration for the railway dynamics – TODO12

The train state will calculate a "deltaEss" event at every frame for the movement. This "deltaEss" event will be distributed from the train state to all axles via the "train/vehicle bus" (TVB).

The Train State, the Being Positioned and the Being Visible of Vehicles

Each train is an unbound object (UBO) and each vehicle is a UBO. Each of these UBOs is assigned to some module at any time. If

- the module the train has been assigned to, is loaded and if
- that module is active,

then **the train has got a valid train state** (position and velocity, acceleration). If

- the train has got a valid train state and if
- the module the vehicle has been assigned to, is loaded and if
- that module is active

then **the vehicle gets into the state positioned** and hence becomes visible

8.2 Structure of a UBO of UOC SrrVehicleTrain – TODO11

A UBO of the UOC SrrVehicleTrain is only one compound object, but it already contains all the essential parts of a train and of a vehicle, only with the restrictions that only one vehicle can be used with the train and the train can neither be coupled with other trains nor decoupled into two independent trains:

- SrrTrainState with a "reference axle"
- an SRR Vehicle containing an SrrBasicCab, an arbitrary set of SRR Drives each containing an arbitrary set of "standard" axles and an arbitrary set of SRR Cabs
- The Train/Vehicle Bus (TVB)

SrrVehicleTrain is an Orchestrator SRR Object that orchestrates all of the above mentioned objects within the model of an "SRR Vehicle Train".

Note: SrrTrains v0.01 will not be able to handle "SRR Train" objects nor "SRR Vehicle" Objects. As long, as we do not need to perform coupling or uncoupling of trains, this should not harm, IMHO.

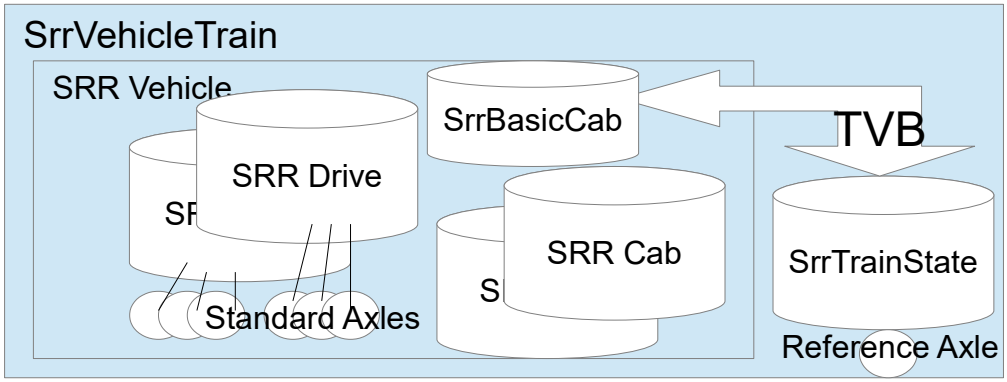

Figure 2: Structure of a UBO of UOC SrrVehicleTrain

8.3 Structure of the UBOs of UOCs SrrTrain and SrrVehicle – TODO

Tbd.

8.4 Railway Kinematics – TODO11

Each vehicle may provide a "basic user interface" (SrrBasicCab). If the model author provides this user interface, then each avatar gets the possibility to:

- toggle the "vConstMode" of the train (by default the vConstMode is "ON")
- set the "vConst" value of the train, which is applicable, if "vConstMode" = "ON"
- "pull" or "push" the train manually, which is applicable, if "vConstMode" = "OFF"

If the "vConstMode" is "ON", then "railway kinematics" is applicable to the train (this chapter).

If the "vConstMode" is "OFF", then "railway kinetics" is applicable to the train (next chapter).

Railway Kinematics can be explained as follows:

A) The "vConst" is set to a value by the "basic user interfaces" of all vehicles of the train

B) The SrrTrainState calculates a "deltaEss" event for the whole train from the value of "vConst"

C) The "deltaEss" event is distributed via the "train/vehicle bus" to all axles of the train. This comprises all "standard" axles, all "collision axles" and the "reference axle" of the train

D) Each axle calculates the own properties "parentEdge", "ess" and "isAtoB" freshly and then triggers the TGN of the (new) parent track edge to calculate the "transform" property of the axle

E) The properties of the <Transform> nodes of the bogies of the vehicle bodies and of all parts of the train are calculated from the "transform" properties of all or some "standard" axles of the train

F) The rotation of the axles derives directly from the "deltaEss" event

8.5 Railway Kinetics – TODO12

Each vehicle may provide a "basic user interface" (SrrBasicCab). If the model author provides this user interface, then each avatar gets the possibility to:

- toggle the "vConstMode" of the train (by default the vConstMode is "ON")
- set the "vConst" value of the train, which is applicable, if "vConstMode" = "ON"
- "pull" or "push" the train manually, which is applicable, if "vConstMode" = "OFF"

If the "vConstMode" is "ON", then "railway kinematics" is applicable (previous chapter).

If the "vConstMode" is "OFF", then "railway kinetics" is applicable to the train (this chapter).

Railway Kinetics can be explained as follows

In case of railway kinetics, then the basic user interface of each vehicle (SrrBasicCab) may be used to "pull" or "push" trains by some handle that is implemented by the model author.

SRR Drive objects may be used to induce forces into the SrrTrainState and SRR Cab objects may be used to control the SRR Drive objects via the "train/vehicle bus".

The rotation of the "standard" axles is rather controlled by their parent drives than directly by the "deltaEss" event. This enables the modeling of hurling or blocking SRR Drives.

A) The SrrTrainState collects all force components from the basic user interfaces of all vehicles and from all SRR Drives. At this stage, each axle of the train is assigned a fraction of the whole mass of the train in order to be able to immediately correct the influence of gravity

B) The SrrTrainState Objekt sums up all force components and calculates the overall acceleration of the train, knowing the whole mass of the train. With the acceleration it calculates a new velocity and a "deltaEss" event for the whole train.

C) The "deltaEss" event is distributed via the "train/vehicle bus" to all axles of the train. This comprises all "standard" axles, all "collision axles" and the "reference axle" of the train

D) Each axle calculates the own properties "parentEdge", "ess" and "isAtoB" freshly and then triggers the TGN of the (new) parent track edge to calculate the "transform" property of the axle

E) The properties of the <Transform> nodes of the bogies of the vehicle bodies and of all parts of the train are calculated from the "transform" properties of all or some "standard" axles of the train

F) The rotation of the "standard" axles either derives directly from the "deltaEss" event or from the parent SRR Drive objects (this is decided by the SRR Drive Objects)

9 SRR Objects for Handover and Moving Modules – TODO11

Tbd.