PSI

Anpassungen railML 2.5 für Fahrgastinformationen am Bahnhof

Autor	Milan Wölke
Version	0.7
Datum	02.04.2019
Status	In Arbeit

Inhalt

1 Allgemein ***1***

2 Anforderungen RhB ***2***
2.1 Ansagen *2*
2.2 Zugsymbole *2*
2.3 Produktlogos und -namen *2*
2.4 Mehrsprachige Zieltexte *3*
2.5 Sprachen bei der Ankündigung eines Zuges *3*
2.6 Wendeinformationen *3*

3 Umsetzung ***4***
3.1 Erweiterungen an `//annotations/annotation` *4*
3.2 Erweiterungen an `//trainPart` *7*
~~*3.3 Erweiterungen an `//blockPart`*~~ *11*

1 Allgemein

Im Rahmen des Updates der Systeme der Rhätischen Bahn RhB von railML 1.0 auf railML 2.X sind Anforderungen bezüglich der Abbildung von Fahrgastinformationenen in railML aufgekommen, die hier kurz erläutert werden sollen.

Zur Abbildung von Fahrgastinformationen wird in railML das Tag `<annotation>` verwendet. Damit werden die generell verwendeten Fahrgastinformationen angegeben. Hier kann ein Text in mehreren Sprachen hinterlegt werden. Aus `<trainPart>` und `<stopDescription>` kann dann auf diese verwiesen werden.

2 Anforderungen RhB

2.1 Ansagen

Für die RhB wäre es nun wichtig nicht nur Texte als Fahrgastinformationen hinterlegen zu können. Vielmehr sollen etwa auch Ansagen für die Zugfahrten und deren Halten definiert werden können. Nun ist es nicht möglich oder sinnvoll tatsächliche Audiodaten in einer XML-basierten Struktur zu transportieren, allerdings wäre es denkbar auf Schlüssel von Ansagen in den abnehmenden Systemen zu verweisen. Weiterhin sollte es möglich sein Ansagezeitpunkte zu definieren, um etwa „Ansagen bei Einfahrt“ von „Ansagen vor Ausfahrt“ unterscheiden zu können. Dabei sollte auch berücksichtigt werden, dass u. U. periodische Ansagen geplant werden, die mit einem bestimmten Intervall eine bestimmte Zeit vor einem Endzeitpunkt (Einfahrt des Zuges, Ausfahrt des Zuges) abgespielt werden.

2.2 Zugsymbole

Bei Fahrgastinformationssystemen in der Schweiz ist es üblich Zugsymbole für Züge auf den Stationsdisplays darzustellen. Dabei handelt es sich zum Teil um Symbole, die Informationen über Dienste und Möglichkeiten an Bord des Zuges kodieren, wie etwa Fahrradmitnahme oder Restaurant, welche sich aus den Informationen des Subschemas Rollingstock ableiten lassen. Zum Teil handelt es sich aber auch um Informationen, für die keine andere Entsprechung in railML existiert, bspw. Selbstkontrolle, also die Information, dass an Bord keine Fahrscheinkontrolle vorgenommen wird. Diese Symbole werden dann bei der Ankündigung des Zuges auf Überkopfanzeigen etc. am Bahnhof zur Anzeige gebracht.

2.3 Produktlogos und -namen

Eine weitere Klasse von Daten, die von der RhB gerne auf den Fahrgastinformationsdisplays dargestellt werden würden und zu diesem Zweck via railML zwischen den verschiedenen beteiligten Systemen ausgetauscht werden sollten, sind Produktinformationen. Hintergrund ist, dass die RhB touristische Marken wie den Panorama-Express zu etablieren versucht und daher entsprechend durchgängige Fahrgastinfos mit den passenden Logos und Bezeichnungen darstellen möchte. Im Unterschied zu den unter 2.2 erwähnten Symbolen, handelt es sich hier um Grafiken oder Videos, wogegen man sich unter Symbolen eher Zeichen aus einem Zeichensatz vorstellen kann.

2.4 Mehrsprachige Zieltexte

Bei einbrechenden und ausbrechenden Verkehren wäre es sinnvoll Ziel- bzw. Herkunftstexte via railML austauschen zu können. Dabei sollte berücksichtigt werden, dass diese auch mehrsprachig vorliegen können (Biel/Bienne, Wien/Vienna, Bautzen/Budyšin). Entsprechend sollte es möglich sein entsprechende Texte mehrsprachig in railML abbilden zu können. Um auch entsprechende Ansagen zu erlauben, sollte es auch möglich sein den Ziel- bzw. Herkunftsort und die entsprechenden Audioresourcen mithilfe eines enthaltenen Schlüssels bestimmen zu können.

2.5 Sprachen bei der Ankündigung eines Zuges

Die RhB betreibt neben Zügen mit touristischer Ausrichtung auch Züge, deren vorwiegender Zweck die Anbindung abgelegener Orte ist, und die entsprechend eher von den Einheimischen genutzt werden. Daraus leiten sich unterschiedliche Anforderungen an die Fahrgastinformation ab. Für Züge mit touristischer Ausrichtung müssen Ansagen in internationalen Sprachen erfolgen, wogegen bei lokal dominierten Angeboten eher regionale Sprachen bei der Ansage berücksichtigt werden müssen. Für die RhB wäre es wünschenswert diese Informationen ebenfalls in railML kodieren zu können.
Diese Anforderung speist sich aus der Annahme, dass Ansagen in FGI-Systemen oft so hinterlegt sind, dass inhaltlich gleiche Texte in verschiedenen Sprachen als eine Ansage gebündelt werden. Das heißt, dass bspw. hinter der Ansage „Der Zug fällt aus“, nicht nur die deutschsprachigen Audiofiles liegen, sondern auch Audiodaten in Englisch, Französisch, etc. Wenn nun für einen Zug auf die Ansage verwiesen wird, dann wird erwartet, dass die Information, die durch die Ansage kodiert wird, in den Sprachen am Bahnhof ausgegeben wird, die an dem entsprechenden Bahnhof verstanden werden. Darüber hinaus möchte die RhB pro Zug angeben können in welchen Sprachen den Zugbetreffende Ansagen erfolgen sollen. Um beim Beispiel von oben zu bleiben, würde die Ansage „Zug fällt aus“ an einem Bahnhof, für den die Sprachen Deutsch, Englisch, Französisch und Rätoromanisch als relevant konfiguriert sind, nur in Deutsch und Rätoromanisch ausgerufen werden, wenn für den entsprechenden Zug nur diese beiden Sprachen als Ansagesprache festgelegt wurden. Wenn hingegen für den Zug aufgrund seiner touristischen Auslegung die Ansagesprachen Deutsch, Englisch und Französisch festgelegt wurden, erfolgt die Ansage am selben Bahnhof nur in diesen Sprachen.

2.6 Wendeinformationen

Für die korrekte Anzeige von Fahrgastinformationen ist es nötig für einen Zug bestimmen zu können auf welche Art und Weise die Wende in die Folgezugfahrt erfolgen soll. Dabei müssen 3 verschiedene Wendearten unterschieden werden:

- Weiterfahrt in gleicher Richtung: Fahrgästen wird bereits vor Erreichen des Zielbahnhofs der Zieltext der Folgefahrt angezeigt. Für den Fahrgast handelt es sich um eine einzige Fahrt. Betrieblich hingegen kommen unterschiedliche Zugnummern zum Einsatz.

- Bahnsteigwende: Bei Einfahrt des Zuges in den Endbahnhof wird dem Fahrgast bereits die Folgefahrt angezeigt, da direkt nach Einfahrt eingestiegen werden kann.
- Kehrenwende: Bei Einfahrt des Zuges am Endbahnhof muss „Nicht einsteigen“ angezeigt werden, da der Zug vor der Weiterfahrt über die Kehre gewendet wird.

3 Umsetzung

Im Folgenden sollen für die oben angeführten Punkte Vorschläge für die Umsetzung gemacht werden, die dann in der Community diskutiert werden sollen.

3.1 Erweiterungen an `//annotations/annotation`

In Abbildung 1 ist die aktuelle Abbildung des Tag `<annotation>` in railML 2.4 dargestellt. Wie dargestellt geht die aktuelle Modellierung davon aus, dass es sich bei den verwalteten Fahrgastinformationen ausschließlich um Texte handelt. Daher muss `<text>` auch mindesten 1 Mal angegeben werden.

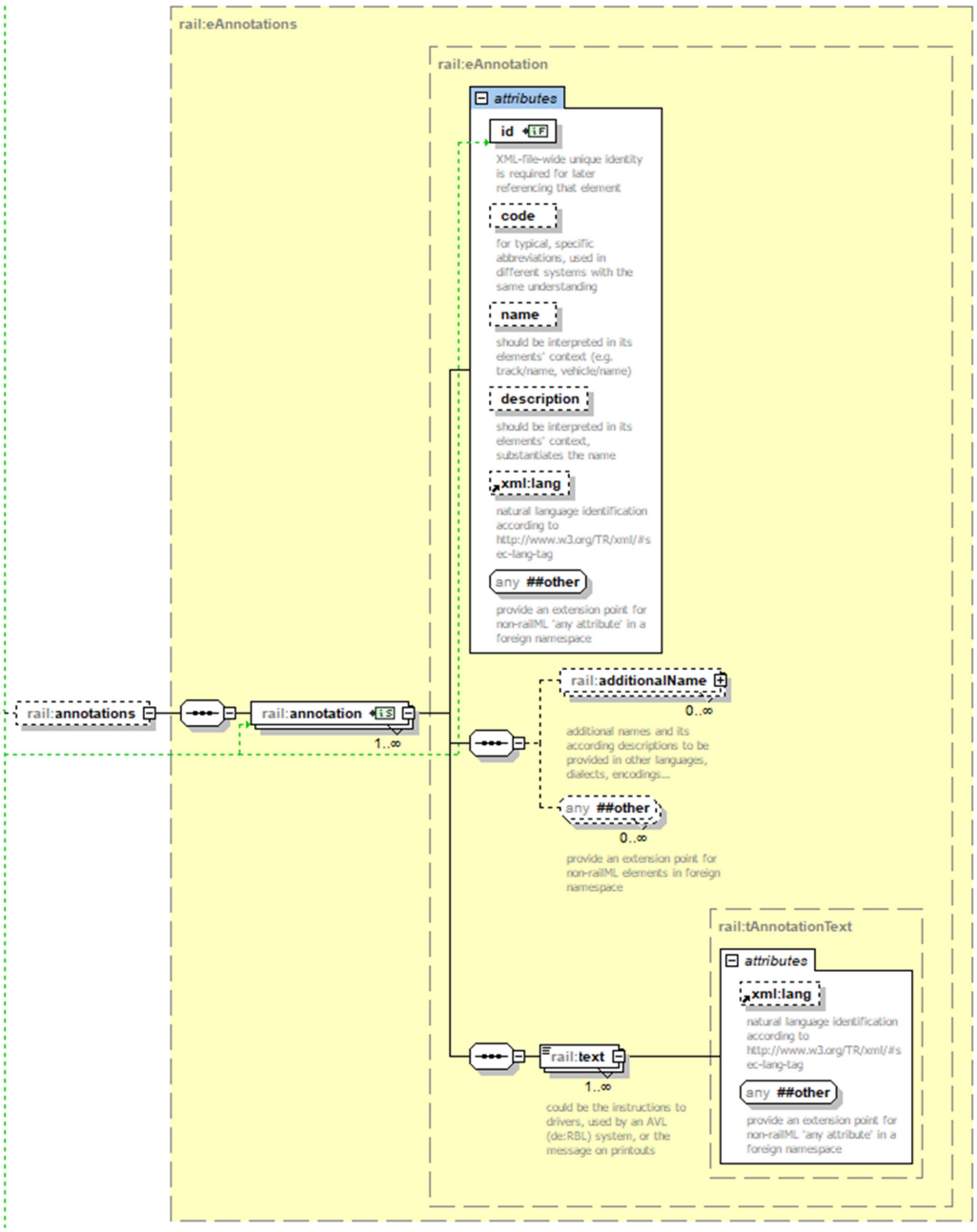

Abbildung 1: Ausschnitt aus dem railML Schema bzgl. `<annotation>`

Um auch Bilder und Symbole wie unter 2.2 und 2.3 beschrieben abbilden zu können, sollte das Tag `<text>` von 1..∞ zu 0..∞ geändert werden. Die Attributgruppe von `<annotation>` sollte um `@type` erweitert werden. Dieses Attribut sollte als Enumeration definiert sein und mindestens über die folgenden Werte verfügen:

- `Text`
- `Symbol`
- `Image`
- `Other (Erweiterbar)`

Default-Wert für dieses Attribut müsste aus Gründen der Rückwärtskompatiblität „`Text`“ sein.

Auf diese Weise wäre es möglich wie bisher Texte als Fahrgastinformationen zu modellieren und außerdem auf Bilder, etc. zu verweisen. Über das bereits vorhandene Attribut `@code` könnte bei Bildern auf einsprechende im Abnehmerystem vorhandene Resourcen verwiesen werden. Dies scheint zweckmäßig, da wie bereits oben erwähnt eine Übertragung der entsprechenden Binärdaten als Teil des railML Datenaustauschs nicht sinnvoll ist. Auch Symbole könnten via `@code` abgebildet werden.
Sollte es nötig sein eine über `@type` hinausgehende Klassifizierung vorzunehmen, um etwa Produktnamen oder besonders zu behandelnde Sondertexte von anderen Texten zu unterscheiden, so ließe sich dies über `<additionalName>` erreichen
Für Produktinformationen wäre auch denkbar ein komplett neues Element unterhalb von `<timetable>` vorsehen, auf das dann vom `<trainPart>` referenziert wird.
Für die Abbildung von Ansagen würde dieser Mechanismus allerdings nicht ausreichen, obwohl die unabhängig vom `<trainPart>` zu erfassenden Daten inhaltlich sehr ähnlich sein und im wesentlichen aus einem `@code` bestehen dürften, der im Abnehmersystem bekannt sein müsste. Hintergrund ist, dass beim Verweis auf eine Ansage vom `<trainPart>` bzw. von der `<stopDescription>` aus zusätzliche Informationen erforderlich sind. So muss etwa angegeben werden wann eine Einzelansage abgespielt werden soll, etwa bei Einfahrt des Zuges oder erst vor der Ausfahrt. Um syntaktisch sicherzustellen, dass ein solcher Verweis die nötigen Informationen enthält, muss dieser auf ein anderes Element als `<annotation>` verweisen. Ein Vorschlag wäre Ansagen über ein neues Element `<announcement>` parallel zu `<annotation>` zu definieren.

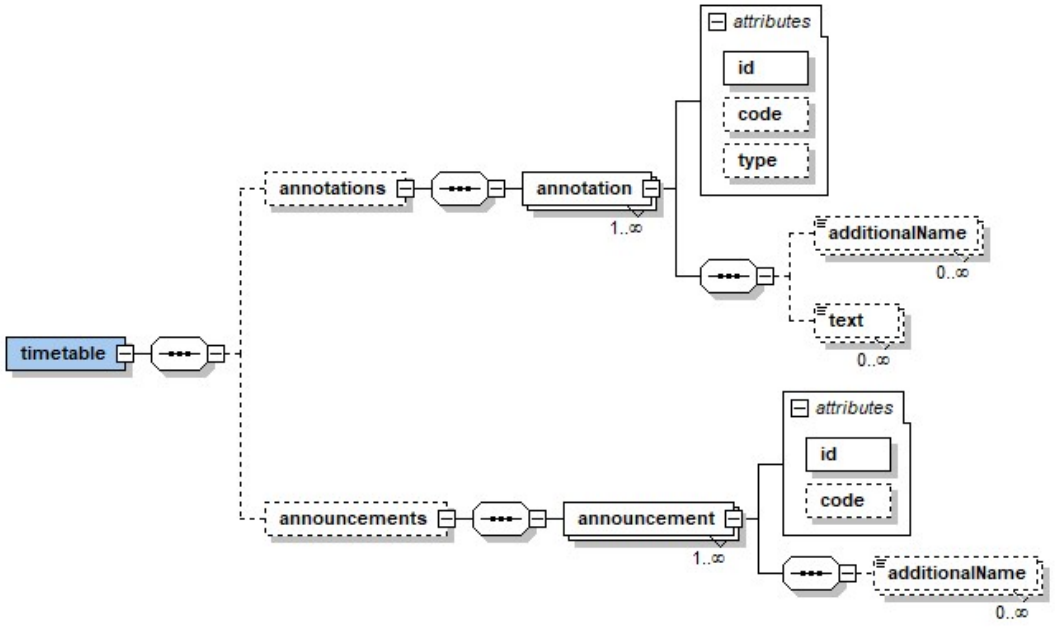

Abbildung 2: Mögliche Abbildung von Ansagen und Sonderinformationen

3.2 Erweiterungen an `//trainPart`

Um auf Fahrgastinformationen am `<trainPart>` zu verweisen, können seit railML 2.2 mit `<annotationRef>` Verweise auf zuvor definierte Inhalte definiert werden. Unter Berücksichtigung der oben beschriebenen Erweiterungen kann auf diesem Wege auch auf Symbole oder Bilder verwiesen werden, wie unter 2.2 und 2.3 gefordert. Wie oben aber bereits erwähnt, können Ansagen auf diesem Weg nur begrenzt beschrieben werden. Es sollte möglich sein, den Zeitpunkt des Abspielens der Ansage zu hinterlegen. Auch periodische Ansagen für einen Zug sollten beschrieben werden können.

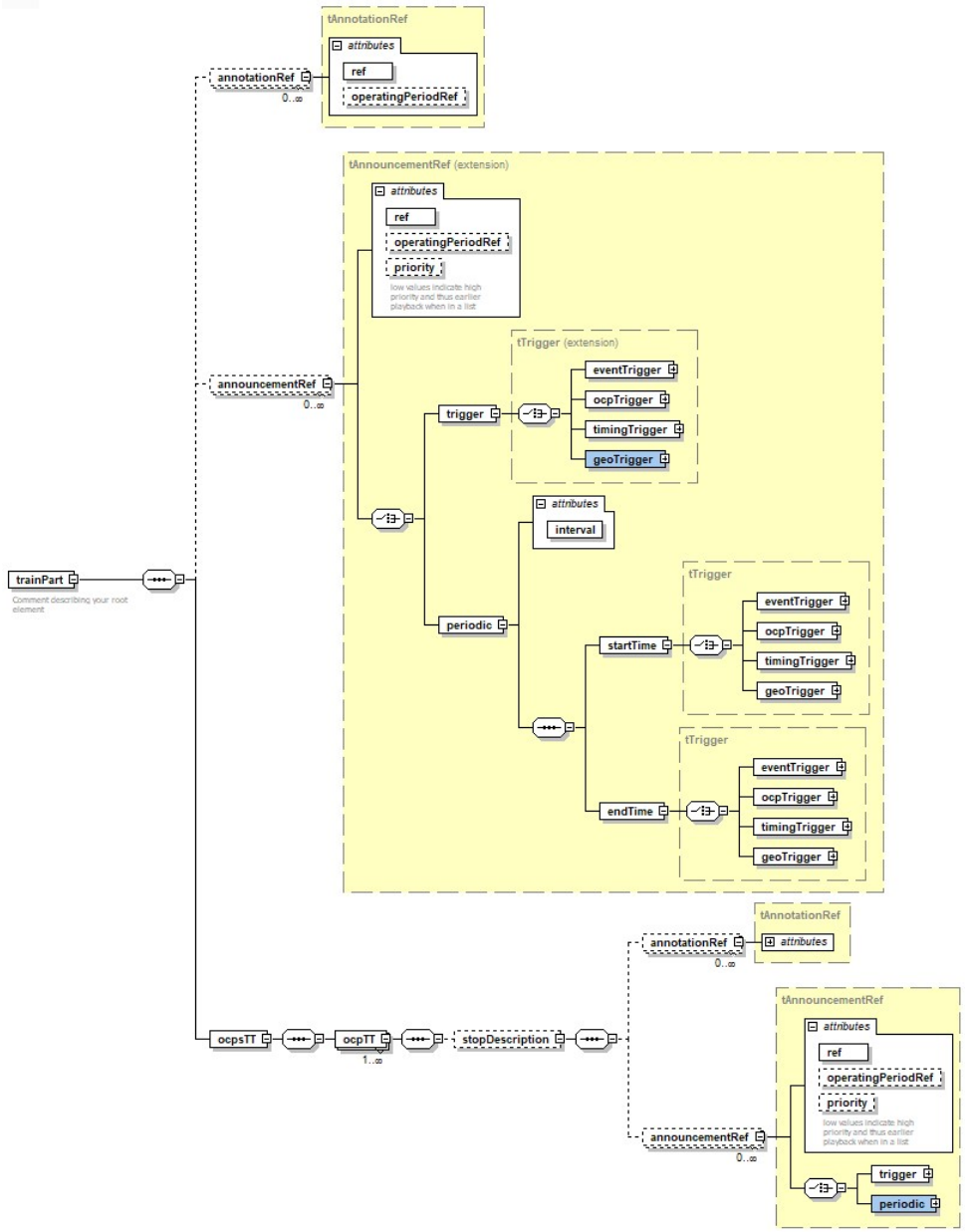

Abbildung 3: Möglicher Verweis auf Ansagen und Sonderinformationen

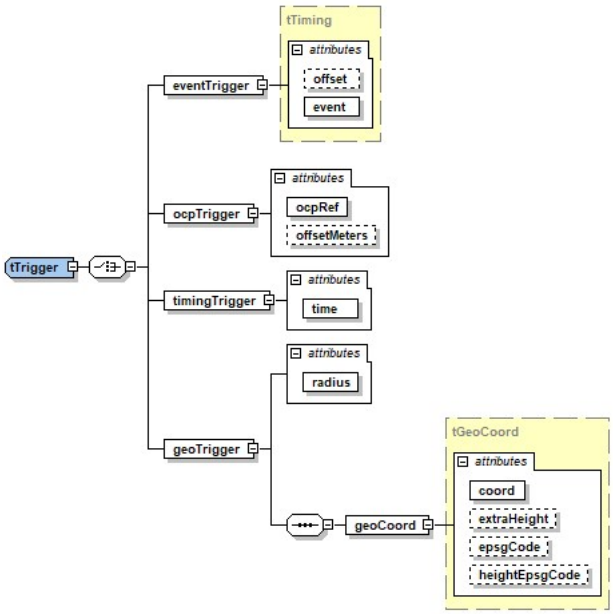

Abbildung 4: Detail-Darstellung von tTrigger

Eine entsprechende Abbildung der Ansagezeitpunkte könnte wie im Diagramm **Fehler! Verweisquelle konnte nicht gefunden werden.** modelliert werden. Unterhalb von `<trigger>` könnten dann die verschiedenen möglichen Ereignisse beschrieben werden, die zum Abspielen der Ansage führen.
Vorgesehene mögliche Ereignisse wären:

- `<eventTrigger>` - Ausgelöst durch ein Fahrtereignis, wie bspw. Ankunft an einem Halt. Das Attribut `@event` ist hier als Enumeration definiert, die über die folgenden Werte verfügt:
  - `Arrival` – Tatsächliche Ankunft (bzw. Prognose)
  - `ScheduledArrival` – Geplante Ankunft
  - `Departure` – Tatsächliche Abfahrt (bzw. Prognose)
  - `ScheduledDeparture` – Geplante Abfahrt
  - `Other` (Erweiterbar)

  Über das Attribut `@offset` kann ein zeitlicher Offset zum Ereignis angegeben werden.
- `<ocpTrigger>` - Ausgelöst durch das Erreichen einer logischen Position im Fahrtverlauf. Die Position wird dabei durch einen Verweis auf einen `<operationControlPoint>` und einen Offset in Metern angegeben.
- `<timingTrigger>` - Ausgelöst zu einer bestimmten Zeit
- `<geoTrigger>` - Ausgelöst durch das Erreichen einer bestimmten Geoposition unter Berücksichtigung eines angegebenen Fangbereichs.

In XML könnte ein Verweis auf eine Ansage dann wie folgt aussehen:

```
<trainPart id="4711">
   <announcementRef ref="announce-2019">
      <trigger>
         <eventTrigger  event="Arrival" offset="-PT30S"/>
      </trigger>
   </announcementRef>
   <announcementRef ref="announce-2020">
      <periodic>
         <startTime>
            <ocpTrigger ocpRef="ocp1234" offsetMeters="-1000"/>
         </startTime>
         <endTime>
            <eventTrigger event="Departure"/>
         </endTime>
      </periodic>
   </announcementRef>
</trainPart>
```

Um, wie unter 2.4 beschrieben, alternative Ziel- und Herkunftstexte bzw. solche, die sich nicht aus dem Laufweg des Zuges ergeben (etwa bei ein- und ausbrechenden Verkehren) abbilden zu können, sollte `<trainPart>` um entsprechende Möglichkeiten erweitert werden. So könnten etwa die optionalen Tags `<origin>` und `<destination>` eingeführt werden.
Eine mögliche Struktur könnte in XML wie folgt aussehen:

```
<trainPart id="tp-2" trainNumber="71" timetablePeriodRef="ttp-633" categoryRef="cat-45" opera-
tor="RhB">
            <formationTT formationRef="tm-185"/>
            <operatingPeriodRef ref="op-343"/>
            <origin ocpRef="ocp-1234"/>
            <destination code="CodeOfDestinationInConsumingSystem">
               <text xml:lang="de">Warschau</text>
               <text xml:lang="en">Warsaw</text>
               <text xml:lang="pl">Warszawa</text>
            </destination>
…
</trainPart>
```

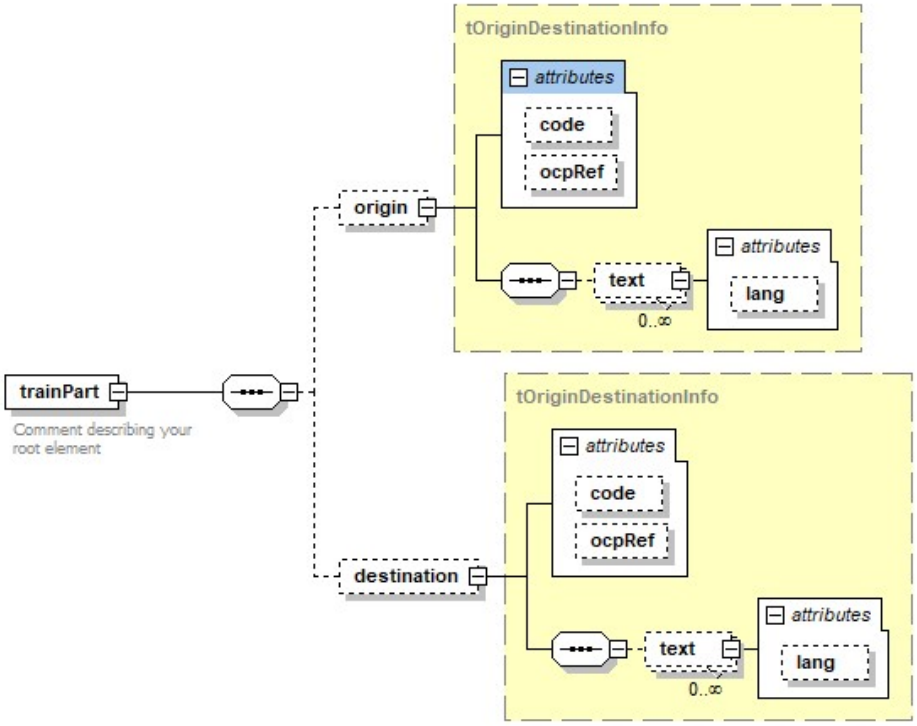

Abbildung 5: Abbildung von Ziel- und Herkunftstexten

Um, wie unter 2.5 beschrieben, für einen Zug festlegen zu können in welchen Sprachen die Fahrgastinformation erfolgen sollte, sollte der `<trainPart>` des Weiteren um ein entsprechendes Element erweitert werden. Prinzipiell sollte es möglich sein für einen gesamten `<trainPart>` die Menge der Sprachen festzulegen, sowie für einzelne Fahrthalte Ausnahmen zu definieren. Entsprechend sollte die gleiche Modellierung für `<trainPart>` und `<stopDescription>` vorgesehen werden.
Eine solche Struktur könnte wie folgt aussehen:

```
<trainPart id="tp-2" trainNumber="71" timetablePeriodRef="ttp-633" categoryRef="cat-45" operator="RhB">
   <formationTT formationRef="tm-185"/>
   <operatingPeriodRef ref="op-343"/>
   <passengerInfoLanguages>
      <language xml:lang="de"/>
      <language xml:lang="en"/>
      <language xml:lang="jp"/>
   </passengerInfoLanguages>
   ...
</trainPart>

<stopDescription commercial="true" purpose="Verkehrshalt">
   <stopTimes minimalTime="PT1M"/>
   <passengerInfoLanguages>
      <language xml:lang="de"/>
      <language xml:lang="en"/>
      <language xml:lang="jp"/>
   </passengerInfoLanguages>
</stopDescription>
```

Alternativ könnte man für die Abbildung der erforderlichen Sprachen auch die Elemente `<annotationRef>` und `<announcementRef>` erweitern. Dies hätte den Vorteil einerseits den Zusam-

menhang zwischen den angegebenen Sprachen und den Fahrgastinhalten klar zu machen und würde andererseits auch mehr Flexibilität bezüglich der genutzten Sprachen erlauben (erste Ansage in 2 Sprachen, zweite Ansage nur in einer Sprache). Eine Abbildung könnte so aussehen:

```
<trainPart id="4711">
   <annotationRef ref="anno-2019">
      <outputLanguages>
         <language xml:lang="de"/>
      </outputLanguages>
   </annotationRef>
   <annotationRef ref="anno-2020">
      <outputLanguages>
         <language xml:lang="de"/>
      </outputLanguages>
   </annotationRef>
   <announcementRef ref="announce-2019">
      <outputLanguages>
         <language xml:lang="de"/>
         <language xml:lang="en"/>
         <language xml:lang="jp"/>
      </outputLanguages>
      <trigger event="Arrival" offset="-PT30S"/>
   </announcementRef>
   <announcementRef ref="announce-2020">
      <outputLanguages>
         <language xml:lang="de"/>
      </outputLanguages>
      <periodic>
         <startTime event="Arrival"/>
         <endTime event="Departure"/>
      </periodic>
   </announcementRef>
</trainPart>
```

~~3.3 Erweiterungen an //blockPart~~

~~Um die unter 2.6 beschriebene Abbildung der Wendeinformationen zu ermöglichen wäre es nötig eine Erweiterung am `<blockPart>` vorzunehmen. Der `<blockPart>` stellt den Verweis auf den `<trainPart>` innerhalb des Umlaufs dar. An dieser Stelle könnte über das neue optionale Attribut `@transitionType` angegeben werden, auf welche Weise die Wende in die Folgefahrt erfolgt. Als Basiswertebereich für das Attribut sollten die oben Beschriebenen möglich sein:~~

- ~~`ShortTurn`~~
- ~~`LongTurn`~~
- ~~`ContinueInSameDirection`~~

Diese Erweiterung erscheint überflüssig, da dieselbe Information auch heute bereits kodiert werden kann. Die Unterscheidung zwischen `ShortTurn` und `LongTurn` kann mithilfe eines zusätzlichen `<blockPart>` erreicht werden. Dabei ist es gar nicht nötig eine entsprechende Rangierfahrt zu planen. Vielmehr reicht es aus einen `<blockPart>` mit `@mission="shunting"` zu erzeugen um eine Kehrenwende abzubilden. Weiterfahrt in gleicher Richtung könnte über einen entsprechenden `<train>` mit `@type="commercial"` modelliert werden.

Änderungsübersicht

Art der Änderung	Version	Datum	Bearbeiter
Initiale Version	0.1	02.04.2019	Milan Wölke
Wendeinformationen hinzugefügt	0.2	03.04.2019	Milan Wölke
Erste Umsetzungsideen hinzugefügt	0.3	04.04.2019	Milan Wölke
Umsetzungsideen verfeinert, Beispiele hinzugefügt	0.4	08.04.2019	Milan Wölke
Erster Review und Kommentierung	0.5	12.04.2019	Philip Wobst
Einarbeitung der beim Review angemerkten Verbesserungen	0.6	24.04.2019	Milan Wölke
Einarbeitung der ersten Rückmeldungen aus dem Forum	0.7	20.05.2019	Milan Wölke

PSI

Changes to railML 2.5 for passenger info at stations

Author	Milan Wölke
Version	0.7
Date	02.04.2019
State	In progress

Content

1 General ***1***

2 Requirements RhB ***2***
2.1 Announcements *2*
2.2 Trainsymbols *2*
2.3 Productlogos and -names *2*
2.4 Multilingual destination texts *2*
2.5 Languages when announcing a train *3*
2.6 Turning information *3*

3 Implementation ***4***
3.1 Extensions at `//annotations/annotation` *4*
3.2 Extensions at `//trainPart` *6*
~~*3.3 Extensions at `//blockPart`*~~ *10*

1 General

As part of the update of the RhB systems from railML 1.0 to railML 2.X, requirements regarding the mapping of passenger information in railML have arisen, which will be briefly explained here.
The tag <annotation> is used to represent passenger information in railML. This tag indicates the generally used passenger information. A text in several languages can be stored here. You can refer to them from `<trainPart>` and `<stopDescription>`.

2 Requirements RhB

2.1 Announcements

For the RhB it would be important not only to be able to store texts as passenger information. It should also be possible to define announcements for train journeys and stops. Of course, it is neither possible nor useful to transport actual audio data in an XML-based structure, but it would be possible to refer to keys of announcements in the receiving systems. Furthermore, it should be possible to define announcement times in order to be able to distinguish between "announcements when arriving" and "announcements before departing". It should also be taken into account, that periodic announcements may be scheduled which are played at a certain interval a certain time before an end time (train arrival, train departure).

2.2 Trainsymbols

With passenger information systems in Switzerland, it is common practice to display train symbols for trains on station displays. Some of these symbols encode information about services and possibilities on board the train, such as bicycle transport or restaurants, which can be derived from the information in the Rollingstock subschema. In some cases, however, it is also information for which no other equivalent exists in railML, e.g. self-monitoring, i.e. the information that no ticket control is carried out on board. These symbols are then shown when the train is displayed on overhead displays etc. at the station.

2.3 Productlogos and -names

Another class of data that the RhB would like to see displayed on the passenger information displays and that should be exchanged between the various systems involved via railML for this purpose is product information. The background to this is that the RhB is trying to establish tourist brands such as the Panorama Express and would therefore like to present consistent passenger information with the appropriate logos and designations. In contrast to the symbols mentioned under 2.2, these are graphics or videos, whereas symbols are more like characters from a character set.

2.4 Multilingual destination texts

It would be useful to be able to exchange destination and origin texts via railML for incoming and outgoing traffic. It should be taken into account that these can also be available in several languages (Biel/Bienne, Wien/Vienna). Accordingly, it should be possible to display corresponding texts in sev-

eral languages in railML. In order to also allow corresponding announcements, it should also be possible to determine the destination and/or origin and the corresponding audio resources with the help of a provided key.

2.5 Languages when announcing a train

In addition to trains with a touristic orientation, the RhB also operates trains whose main purpose is to connect remote villages, and which are therefore used more frequently by the locals. This results in different requirements for passenger information. For trains with a touristic orientation, announcements must be made in international languages, whereas in the case of locally dominated services, regional languages must rather be taken into account in the announcement. It would be desirable for the RhB to be able to encode this information in railML as well.
This requirement is based on the assumption that announcements in PIS systems are often stored in such a way that texts with the same content in different languages are bundled as one announcement. This means, for example, that behind the announcement "The train is cancelled", not only the German-language audio files are stored, but also audio data in English, French, etc. When a train is referring to the announcement, it is expected that the information encoded by the announcement is output in the languages understood at the station. In addition, the RhB would like to be able to indicate in which languages the train announcements should be made. To stay with the example from above, the announcement "train cancelled" at a station for which the languages German, English, French and Rhaeto-Romanic are configured as relevant would only be announced in German and Rhaeto-Romanic if only these two languages have been defined as the announcement language for the corresponding train. If, on the other hand, the announcement languages German, English and French have been defined for the train due to its touristic nature, the announcement at the same station will only be made in these languages.

2.6 Turning information

For the correct display of passenger information, it is necessary to be able to determine how a train is to be turned into the next train journey. There are 3 different types of turn:

- - Continue in the same direction: Passengers are shown the destination text of the next journey before reaching the destination station. For the passenger, this is a single journey. However, different train numbers are used for operational purposes.
- - Platform turn: When the train arrives at the terminal station, the passenger is already shown the next journey, as he or she can board the train immediately after entering the station.
- - Pool turn: When the train arrives at the terminal station, "Do not board" must be displayed, as the train is turned by exiting into the pool and returning from there before continuing its journey.

3 Implementation

In the following, suggestions for the implementation of the above points will be made, which will then be discussed in the community.

3.1 Extensions at `//annotations/annotation`

Figure 1 shows the current representation of the tag `<annotation>` in railML 2.4. As shown, the current modelling assumes that the managed passenger information is only text. Therefore `<text>` must also be specified at least once.

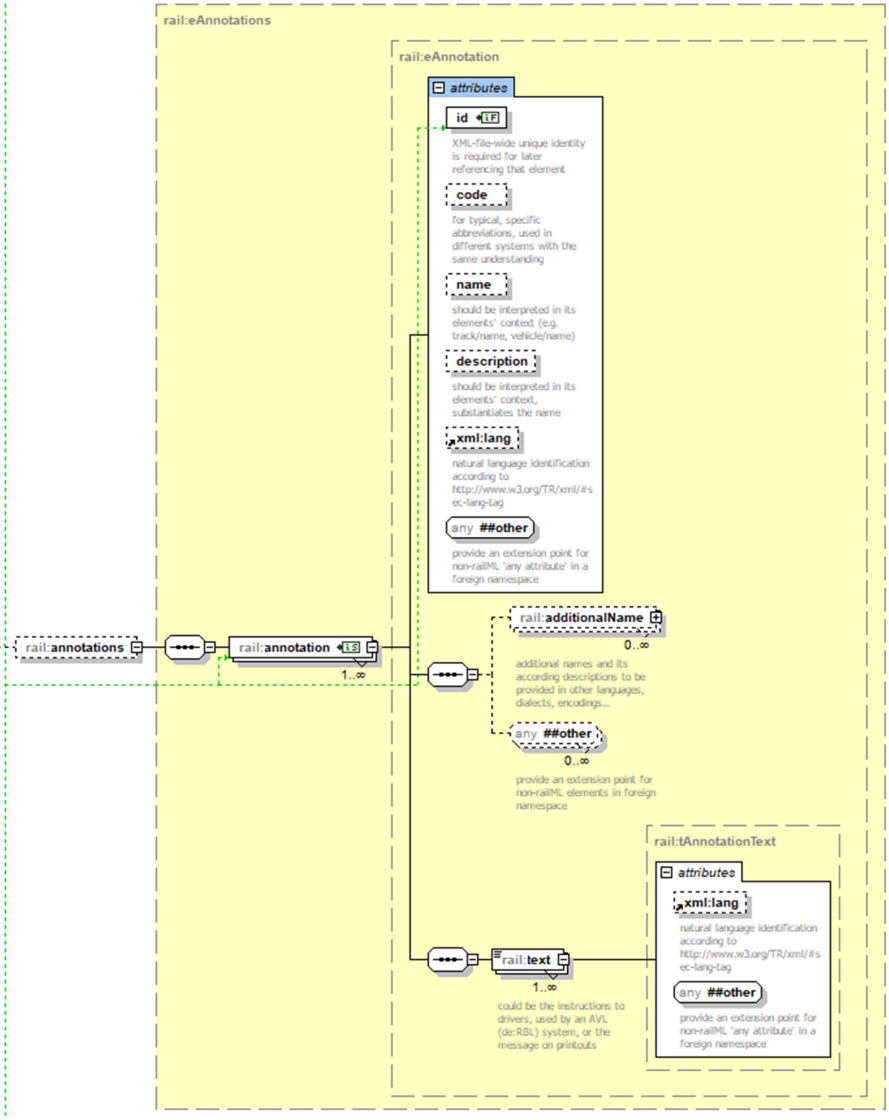

Figure 1: Excerpt of the current railML modelling

In order to display images and symbols as described under 2.2 and 2.3, the tag <text> should be changed from 1..∞ to 0..∞. The attribute group of <annotation> should be extended by @type. This attribute should be defined as enumeration and should have at least the following values:

- Text
- Symbol
- Image
- Other

The default value for this attribute should be "Text" for backward compatibility reasons.

In this way it would be possible to model texts as passenger information as before and also refer to pictures, etc. The already existing @code attribute could be used to refer to corresponding resources in the receiving system. This seems to be useful because, as mentioned above, a transfer of the corresponding binary data as part of the railML data exchange does not make sense. Symbols could also be represented via @code.
Should it be necessary to carry out a classification beyond @type in order to distinguish product names or special texts to be treated differently from other texts, this could be achieved by <additionalName>.
For product information (see 2.3) a completely new element below <timetable> could also be provided, which would then be referenced by the <trainPart>.
This mechanism, however, would not be sufficient for the mapping of announcements, although the data to be captured independently of the <trainPart> would be very similar in content and would essentially consist of a @code that would have to be known in the receiving system. This is because additional information is required when referring to an announcement from <trainPart> or from <stopDescription>. For example, it must be specified when a single announcement is to be played, for example when the train arrives or before it departs. In order to ensure syntactically that such a reference contains the necessary information it must refer to an element other than <annotation>. A suggestion would be to define announcements via a new element <announcement> parallel to <annotation>.

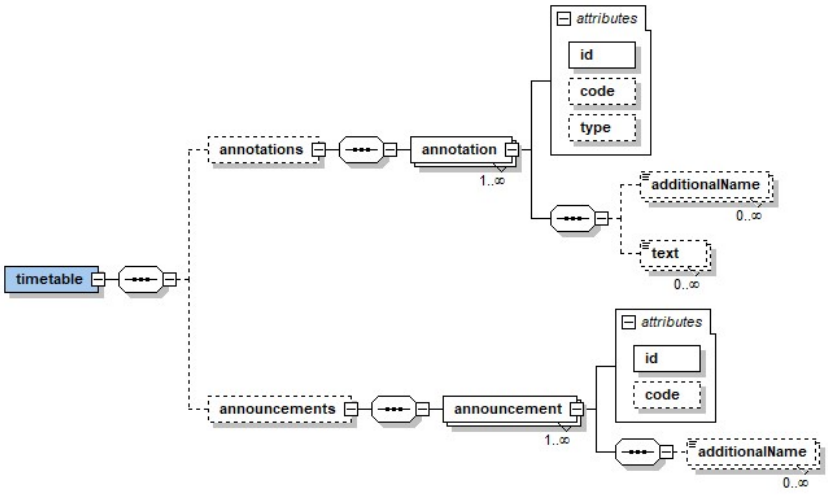

Figure 2: Possible modelling of announcements and special info

3.2 Extensions at `//trainPart`

In order to refer to passenger information at the `<trainPart>`, since railML 2.2 references to previously defined contents can be defined with `<annotationRef>`. Taking into account the extensions described above, symbols or images can also be referenced in this way, as required under 2.2 and 2.3. However, as mentioned above, announcements can only be described to a limited degree in this way. It should be possible to store the time when the announcement is played. It should also be possible to describe periodic announcements for a train.

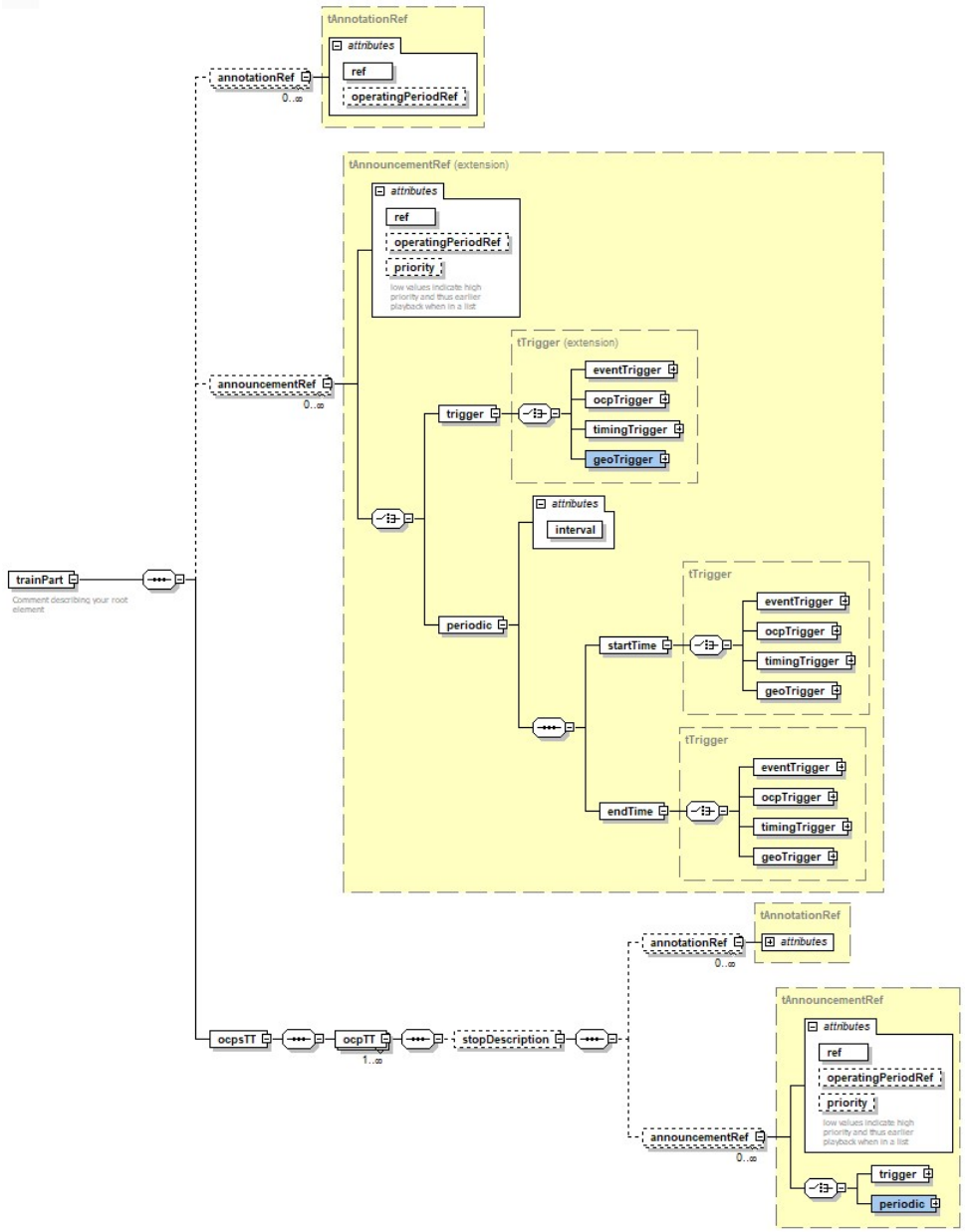

Figure 3: Possible modelling for referencing announcements and special info

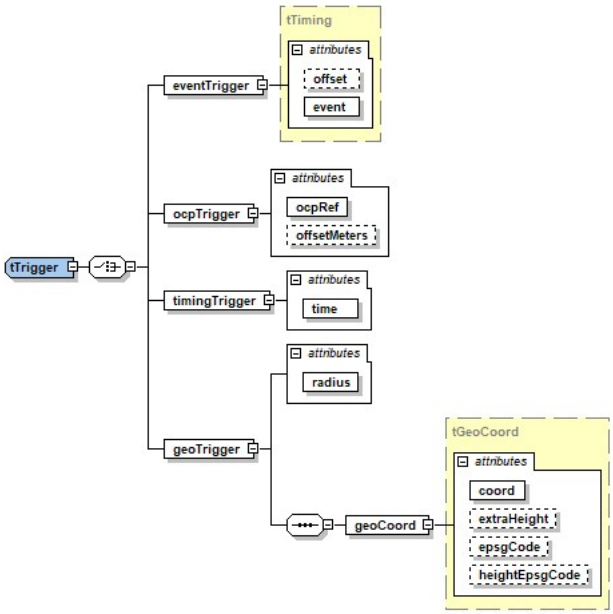

Figure 4: Details of modelling of tTrigger

A corresponding representation of the announcement times could be modelled as in the diagram above. Below `<trigger>`, the various possible events that lead to the announcement being played could be described.
Possible events would be:

- `<eventTrigger>` - Triggered by a driving event, such as arrival at a stop. The attribute `@event` is defined here as enumeration, which has the following values:
  - `Arrival` - Actual arrival (or forecast)
  - `ScheduledArrival` - Planned Arrival
  - `Departure` - Actual departure (or forecast)
  - `ScheduledDeparture` - Scheduled departure
  - `Other` (Extendable)

  The `@offset` attribute can be used to specify a time offset to the event.
- `<ocpTrigger>` - Triggered by reaching a logical position in the trainrun. The position is indicated by a reference to a `<operationControlPoint>` and an offset in meters.
- `<timingTrigger>` - Triggered at a specific time
- `<geoTrigger>` - Triggered by reaching a specific geoposition while respecting a specified catch range.

In XML, a reference to an announcement could look like this:

```
<trainPart id="4711">
   <announcementRef ref="announce-2019">
      <trigger>
         <eventTrigger  event="Arrival" offset="-PT30S"/>
      </trigger>
   </announcementRef>
   <announcementRef ref="announce-2020">
      <periodic>
         <startTime>
            <ocpTrigger ocpRef="ocp1234" offsetMeters="-1000"/>
         </startTime>
         <endTime>
            <eventTrigger event="Departure"/>
         </endTime>
      </periodic>
   </announcementRef>
</trainPart>
```

In order to be able to map alternative destination and origin texts, as described in 2.4, or texts that do not result from the train's route (for example, for incoming and outgoing services), `<trainPart>` should be extended to include corresponding possibilities. For example, the optional tags `<origin>` and `<destination>` could be introduced.
A possible structure in XML could look like this:

```
<trainPart id="tp-2" trainNumber="71" timetablePeriodRef="ttp-633" categoryRef="cat-45" opera-
tor="RhB">
            <formationTT formationRef="tm-185"/>
            <operatingPeriodRef ref="op-343"/>
            <origin ocpRef="ocp-1234"/>
            <destination code="CodeOfDestinationInConsumingSystem">
               <text xml:lang="de">Warschau</text>
               <text xml:lang="en">Warsaw</text>
               <text xml:lang="pl">Warszawa</text>
            </destination>
…
</trainPart>
```

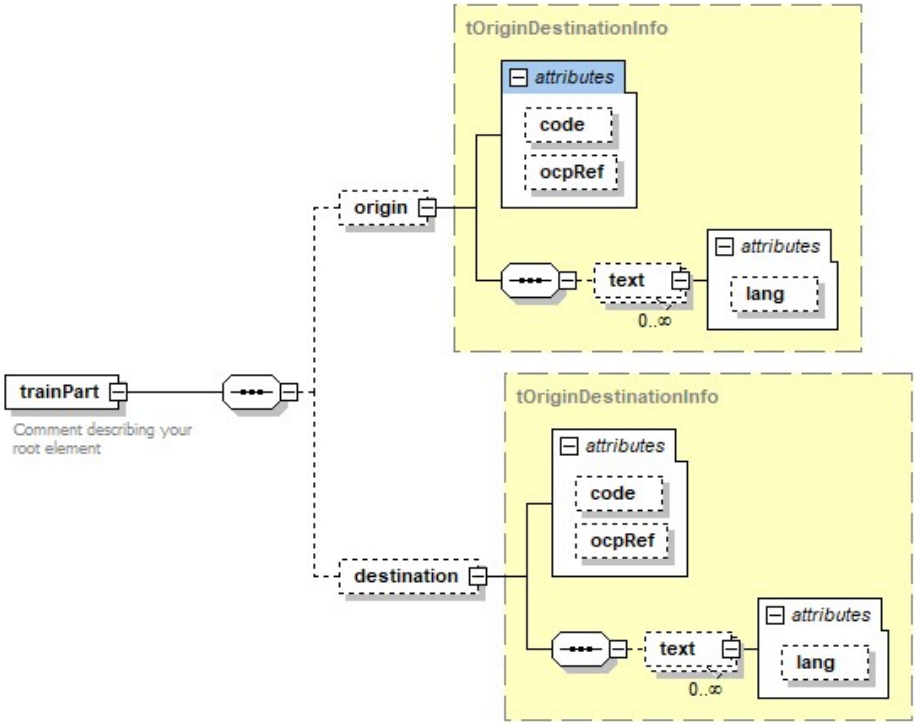

Figure 5: Modelling of origin and destination texts

In order to be able to determine the languages in which passenger information should be provided for a train, as described under 2.5, the `<trainPart>` should also be extended by a corresponding element. In principle, it should be possible to define the set of languages for an entire `<trainPart>` and to define exceptions for individual stops. Accordingly, the same modeling should be used for `<trainPart>` and `<stopDescription>`.
Such a structure could look as follows:

```
<trainPart id="tp-2" trainNumber="71" timetablePeriodRef="ttp-633" categoryRef="cat-45" operator="RhB">
   <formationTT formationRef="tm-185"/>
   <operatingPeriodRef ref="op-343"/>
   <passengerInfoLanguages>
      <language xml:lang="de"/>
      <language xml:lang="en"/>
      <language xml:lang="jp"/>
   </passengerInfoLanguages>
   ...
</trainPart>

<stopDescription commercial="true" purpose="Verkehrshalt">
   <stopTimes minimalTime="PT1M"/>
   <passengerInfoLanguages>
      <language xml:lang="de"/>
      <language xml:lang="en"/>
      <language xml:lang="jp"/>
   </passengerInfoLanguages>
</stopDescription>
```

Alternatively, the elements `<annotationRef>` and `<announcementRef>` could be extended to display the required languages. This would have the advantage of clarifying the connection between

the specified languages and the passenger content on the one hand and on the other hand would allow more flexibility with regard to the languages used (first announcement in 2 languages, second announcement only in one language). An example could look like this:

```
<trainPart id="4711">
   <annotationRef ref="anno-2019">
      <outputLanguages>
         <language xml:lang="de"/>
      </outputLanguages>
   </annotationRef>
   <annotationRef ref="anno-2020">
      <outputLanguages>
         <language xml:lang="de"/>
      </outputLanguages>
   </annotationRef>
   <announcementRef ref="announce-2019">
      <outputLanguages>
         <language xml:lang="de"/>
         <language xml:lang="en"/>
         <language xml:lang="jp"/>
      </outputLanguages>
      <trigger event="Arrival" offset="-PT30S"/>
   </announcementRef>
   <announcementRef ref="announce-2020">
      <outputLanguages>
         <language xml:lang="de"/>
      </outputLanguages>
      <periodic>
         <startTime event="Arrival"/>
         <endTime event="Departure"/>
      </periodic>
   </announcementRef>
</trainPart>
```

~~3.3 Extensions at //blockPart~~

~~In order to allow the representation of the turning information described under 2.6, it would be necessary to make an extension to the <blockPart>. The <blockPart> represents the reference to the <trainPart> within the circulation. Here the new optional attribute @transitionType could be used to specify the way in which the turn into the following <trainPart> takes place. The base value range for the attribute should be the range described above:~~

- ~~ShortTurn~~
- ~~LongTurn~~
- ~~ContinueInSameDirection~~

This extension seems unnecessary since the same information can already be coded today. The distinction between ShortTurn and LongTurn can be achieved with the help of an additional <blockPart>. It is not even necessary to plan a corresponding shunting journey. Rather, it is sufficient to create a <blockPart> with @mission="shunting" in order to map a turnaround via pool. Continued travel in the same direction could be modelled with a corresponding <train> with @type="commercial".

PSI

Overview of changes

Type of change	Version	Date	Author
Initial version	0.1	02.04.2019	Milan Wölke
Turning info added	0.2	03.04.2019	Milan Wölke
First ideas for implementation added	0.3	04.04.2019	Milan Wölke
Improved implementation ideas	0.4	08.04.2019	Milan Wölke
First review and comments	0.5	12.04.2019	Philip Wobst
Incorporation of suggested improvements	0.6	24.04.2019	Milan Wölke
Incorporation of first feedback from Forum	0.7	20.05.2019	Milan Wölke