Beschreibung von Eisenbahninfrastrukturen mit railML und ihre Verifikation

Michael Lodemann / Carsten Gerke / Norbert Luttenberger

Der Artikel beschreibt die Entwicklung eines regelbasierten Verifikationssystems zur Optimierung des Planungsprozesses von Eisenbahninfrastruktur- und Sicherungselementen. Das System verwendet Technologien der Wissensrepräsentation und ermöglicht die Abbildung von Planungs- und Projektierungsrichtlinien in Form von erweiterbaren Regelsätzen. Bei einer automatischen Verifikation wird ein Prüfbericht generiert, der den manuellen Prüfaufwand stark vermindert. Als Transferformat zwischen Planungsanwendung und dem Verifikationssystem wird railML®, ein eisenbahnspezifisches XML-Schema, verwendet.

1 Planung von Eisenbahninfrastruktur

Die sicherungstechnische Planung von neuer oder geänderter Eisenbahninfrastruktur, insbesondere eines ESTW, ist ein sehr komplexer Prozess, an dem viele Institutionen und Personen beteiligt sind und der sich über mehrere Jahre erstreckt. Er beginnt beispielsweise bei der DB Netz AG mit der Definition der betrieblichen Aufgabenstellung, die zunächst in einer Vorplanung (VP) umgesetzt wird, wobei ggf. mehrere Varianten der Planung untersucht werden. Die Planung wird dann über die Entwurfsplanung (EP) und die Ausführungsplanung Planteil 1 (PT 1) konkretisiert und verfeinert. In der anschließenden Ausführungsplanung Planteil 2 (PT 2) werden dann die konkreten baulichen Umsetzungen und die herstellerspezifischen Eigenschaften der sicherungstechnischen Elemente geplant bzw. projektiert. Bei komplexeren Planungen erfolgt die Planung nicht immer durchgehend nach diesem Ablauf, es werden Bauphasen und Bauzustände eingeführt, die aufeinander aufbauen, teilweise aber auch parallel geplant werden müssen.

Bei allem ist zu berücksichtigen, dass sicherheitsrelevante Eisenbahninfrastruktur geplant und projektiert wird. Die Planung erfolgt daher nach einem festgelegten Regelwerk. Bei Planungen für die DB Netz AG ist das zu Grunde liegende Regelwerk im Wesentlichen die Richtline „819 – LST-Anlagen planen“ [1]. Auch die Projektierung des ESTW durch die Stellwerkshersteller im Planteil 2 erfolgt nach konkreten Vorgaben in Form von Projektierungsrichtlinien.

Eine wesentliche Aufgabe bei der Planung besteht darin, in den jeweiligen Planungsphasen, insbesondere zum Abschluss des Planteils 1 und des Planteils 2, sicherzustellen, dass die Planung den Anforderungen der zugrundeliegenden Richtlinien entspricht. Dieser Prozess der Verifikation erfolgt sowohl durch den Planer selbst als auch durch unabhängige Institutionen und ggf. durch eine zuständige Aufsichtsbehörde wie das Eisenbahn-Bundesamt (EBA). Bei dem Prozess ist außerdem dafür zu sorgen, dass die Konsistenz der Planungen zwischen den einzelnen Phasen und Bauzuständen eines Projektes gewahrt bleibt.

Derzeit ist dieser Prozess geprägt durch diverse manuelle Prüfungen der Daten. Obwohl zur Erstellung der Planungen und Projektierungen im Allgemeinen elektronische Werkzeuge eingesetzt werden (z. B. ProSig [2], BEST [3] sowie Werkzeuge der Stellwerksherstellen), werden die Daten meist in Papierform als Zeichnungen und Tabellen weitergegeben und auch auf Papier geprüft. Dies bedingt einen zeitaufwändigen Prozess, bei dem es auch leicht zu Fehlern kommen kann.

Vor diesem Hintergrund wurde 2007 von Funkwerk Information Technologies gemeinsam mit dem Institut für Informatik an der Christian-Albrechts-Universität Kiel unter Förderung des Landes Schleswig-Holstein ein Forschungsvorhaben gestartet mit dem Ziel, die Verifikation der Planung und Projektierung von Eisenbahninfrastruktur zu automatisieren. Maßgeblich hierfür war, dass bei der Entwicklung des neuen ESTW-R von Funkwerk IT [4] auch beim Hersteller intern die Problematik der manuellen Verifikation der ESTW-Projektierung unter Einhaltung der Planungs- und Projektierungsrichtlinien auftritt.

Um eine automatische Verifikation der ESTW-Planung zu ermöglichen, ist es notwendig, dass diese in einer maschinenlesbaren Form vorliegt. In dem Forschungsprojekt fiel die Entscheidung, railML als Basis zu verwenden. Dieses wird im weiteren Verlauf des Artikels beschrieben. Das Projektierungwerkzeug des Betriebs- und Stellwerkssimulators BEST wurde dazu um eine Exportschnittstelle erweitert, über die die Planungsdaten ins railML-Format (Version 1.0) überführt werden.

2 railML

2.1 Was ist railML?

Die railML-Initiative [5] wurde 2001 mit dem Ziel gegründet, ein XML-basiertes open-source Datenaustauschformat für eisenbahnspezifische Anwendungen zu entwickeln. Sie besteht aus einem Zusammenschluss von Eisenbahnunternehmen, Software- und Consultingfirmen sowie akademischen Institutionen und versteht sich als eine Plattform, über die Experten unterschiedlicher Fachrichtungen aus dem Bereich schienengebundener Verkehr über die Entwicklung eines gemeinsamen Datenaustauschstandards diskutieren.

Die Initiative verabschiedete railML als digitales Austauschformat für Eisenbahndaten. railML ist ein sogenanntes XML-Schema, welches die logische Struktur von Dokumenten vorgibt. Somit muss in einem konformen railML-Dokument z. B. ein Track-Element (Gleisabschnitt) ein Unterelement „trackTopology“ aufweisen, worin u. a. der Start und Endpunkt des Gleisabschnitts beschrieben wird. Ein railML-Dokument kann automatisch gegen das railML-Schema validiert werden. Fehlt nun z. B. beim Track-Element das trackTopology-Unterelement, wird dieser Fehler bei der Validierung beanstandet. So wird gewährleistet, dass ein railML-Dokument alle nötigen Inhalte in der vorgegebenen Struktur enthält.

railML liegt zum Zeitpunkt der Verfassung dieses Artikels in der Version 2.0

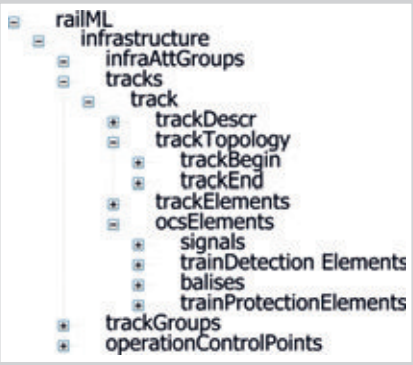

Bild 1: Struktur des Subschemas „infrastructure“ von railML

vor. Das Schema unterliegt jedoch einem kontinuierlichen Weiterentwicklungsprozess, um dem Ziel der railML-Initiative zu entsprechen: Das Schema ständig zu verbessern, zu verfeinern, zu erweitern und dessen Anwendbarkeit für weitere Bereiche der Eisenbahnindustrie zu erschließen. Es soll ein digitaler Datenaustausch von Anwendungen in sämtlichen Bereichen des schienengebundenen Verkehrs mithilfe von railML ermöglicht werden. Diese Interoperabilität von Anwendungen ist in der heutigen vernetzten Welt ein Zugewinn an Performance und Automatisierung und kann als ein wesentlicher Bestandteil zur Optimierung von Abläufen und Prozessen der Eisenbahnindustrie verstanden werden.

In einer Vielzahl von Projekten wird railML bereits eingesetzt. Das Spektrum der Anwendungen umfasst dabei Simulations- (E-Traktion, GPS-InfraDat), Dispositions- und Fahrplanprogramme (SBB-Netzfahrplan) sowie Infrastruktur- und Umlaufplanungsanwendungen und vieles mehr. RailML ist nicht auf Deutschland beschränkt, sondern wird als internationales Format wahrgenommen. Einige Mitglieder der railML-Initiative stammen aus Ländern wie der Schweiz, Österreich, Slowakei bis hin zu Japan.

2.2 Die Bestandteile von railML

Das railML-Schema besteht aus mehreren Subschemata, die sich auf unterschiedliche Bereiche des Eisenbahnsektors beziehen und im Folgenden erläutert werden. Das railML-Schema referenziert das Dublin Core MetaData Schema [6], um railML-Dokumente mit Zusatzdaten, wie beispielsweise Verfasser, Titel, Ursprung, etc., anzureichern.

Das Subschema „infrastructure“ kann im Allgemeinen als eine Gliederung von Elementen angesehen werden, die die stationären, physikalischen Ausprägungen von Eisenbahnstrecken beschreiben. Auch die Modellierung und Zuordnungen von Zugsicherungs- und Zugdetektionssystemen, wie Gleismagneten, Balisen, Achszähler etc., befinden sich neben anderen Informationen im „infrastructure“-Subschema.

Das Subschema „infrastructureVisualizations“ ergänzt das beschriebene Subschema „infrastructure“ um zweidimensionale Positionsangaben der einzelnen Infrastrukturelemente. Dies ermöglicht beispielsweise die Modellierung einer Bedienoberfläche in der Bedienzentrale eines Stellwerks.

Im Subschema „rollingstock“ werden einzelne Bestandteile eines Zuges beschrieben. So existieren beispielsweise Elemente für unterschiedliche Typen von Triebfahrzeugen, Personen- und Güterwaggons. Ebenso können Konstellationen und Anordnungen der einzelnen Zugelemente mit dem „rollingstock“-Subschema modelliert und mit Eigenschaften wie Länge, Bremssysteme etc. angereichert werden.

Im Subschema „timetable“ sind fahrplanspezifische Elemente angesiedelt. Es können sowohl der Regelbetrieb als auch komplexe Fahrplansituationen, wie beispielsweise außerplanmäßige Fahrten oder der Ausnahmebetrieb an Feiertagen, modelliert werden.

Das railML-Schema und seine Subschemata verfügen in weiten Teilen über sogenannte „any“-Attribute und -Elemente. Dieser Modellierungsaspekt ermöglicht Anwendern, das Schema mit eigenen, evtl. firmenspezifischen Elementen und Attributen zu erweitern.

2.3 Das Subschema „infrastructure“

Im Subschema „infrastructure“ werden, wie bereits erwähnt, alle Elemente der Eisenbahntopologie, wie z. B. Gleisabschnitte, Signale, Bahnübergänge, etc., beschrieben. Die Forschungsarbeit, die diesem Artikel zu Grunde liegt, hat als Schwerpunkt die rechnergestützte Verifikation von Planungs- und Projektierungsdaten neuer oder erweiterter Eisenbahnstrecken und bezieht sich somit zum Großteil auf das Subschema „infrastructure“. Aus diesem Grund werden die Elemente dieses Subschemas im Folgenden detaillierter beschrieben.

Bild 1 zeigt die Struktur des Subschemas „infrastructure“ von railML. Das Wurzelelement „railML“ stammt aus der übergeordneten railML-Schemadefinition. Das Element „infrastructure“ ist das erste Element im Subschema „infrastructure“ und beinhaltet alle anderen Elemente des Subschemas. Im Element „infraAttGroups“ befinden sich Unterelemente, die die Strecke(n) im Allgemeinen beschreiben. Dort können beispielsweise der Streckenbetreiber, die Elektrifizierung, die Richtgeschwindigkeit etc. angegeben werden.

Der Hauptelementzweig im Subschema „infrastructure“ befindet sich im „tracks“-Element. Dort werden sämtliche Gleiselemente deklariert. Das Unterelement „trackTopology“ beschreibt die räumliche Ausdehnung eines Gleiselements und beinhaltet die Referenzen auf Nachbarelemente. Im „track“ untergeordneten Element „trackElements“ befinden sich Beschreibungen zu Tunneln, Brücken, Bahnübergängen und vieles mehr. Unter „ocsElements“ werden dem Gleisabschnitt zugeordnete Signale, Balisen, Zugdetektions- und Zugbeeinflussungselemente etc. beschrieben.

2.4 Beispiel für die Anwendung von railML

Die Anwendung von railML, insbesondere des „infrastructure“-Subschemas, verdeutlicht folgendes Beispiel. Bild 2 ist ein Ausschnitt aus einer möglichen Lupenansicht der Bedienoberfläche eines Stellwerks. Sie zeigt zwei Gleisabschnitte „t82B1“ und „t82B2“, ein Vorsignal „s1“ und ein Hauptsignal „s2“. Der gelbe Pfeil am linken Rand der Abbildung beschreibt die Richtung, in der die Strecke befahren werden kann.

Bild 2: Ausschnitt der Bedienoberfläche eines ESTW

Die Modellierung dieses Streckenabschnitts in railML wird im Bild 3 dargestellt. Im Folgenden werden Auszüge aus der railML-Datei erläutert. Die Zeilen 1 – 5 beschreiben den Header der Datei. Dort werden alle notwendigen Namespace-Imports deklariert, also jene Informationen, die besagen, welche Elementnamen verwendet werden dürfen und wo sich das railML-Schema befindet, gegen das diese Datei validiert werden soll. Die Definition des ersten Gleisabschnitts (Track) „t82B1“ beginnt ab Zeile 8. Dort erhält der Track seine eindeutige ID. Das Attribut „mainDir=up“ beschreibt, dass der Gleisabschnitt nur in aufsteigender Kilometrierungsrichtung befahren werden darf. Unter dem Element „trackTopology“ werden Anfang und Ende des Gleisabschnitts deklariert. „trackBegin“ mit dem Unterelement „openEnd“ besagt, dass der Gleisabschnitt eine Anfangsverbindung zu einem „offenen Ende“, also außerhalb der modellierten Infrastruktur, aufweist. Das Unterelement „connection“ von „trackEnd“ hat ein Attribut „ref“, dessen Wert der ID der „connection“ des Nachbargleisabschnitts (t82B2) entspricht. Auf diese Weise werden Nachbarschaftsbeziehungen von Gleisabschnitten mit railML modelliert. In einer tieferen Hierarchiestufe des Gleisabschnitts „t82B1“ befindet sich das „signal“ mit der ID „s1“. Dieses Signal ist somit dem entsprechenden Gleisabschnitt zugeordnet. Es weist als „type“ den Wert „distant“ auf, der ein Vorsignal spezifiziert. Analog zum Gleisabschnitt „t82B1“ wird der Gleisabschnitt „t82B2“ modelliert.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <railml xmlns="http://www.railml.org/schemas/2009"
3      xmlns:evu="http://www.example.com/EVU"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://www.railml.org/schemas/2009 ../schema/railML.xsd">
6      <infrastructure id="RTVE" version="2.0">
7          <tracks>
8              <track id="t82B1" mainDir="up">
9                  <trackTopology>
10                 <trackBegin pos="1.000" id="left01">
11                     <openEnd id="x81"/>
12                 </trackBegin>
13                 <trackEnd pos="5.000" id="left02">
14                     <connection ref="cRight01" id="cLeft02"/>
15                 </trackEnd>
16             </trackTopology>
17             <ocsElements>
18                 <signals>
19                     <signal pos=".500" id="s1" type="distant" dir="up"/>
20                 </signals>
21             </ocsElements>
22         </track>
23         <track id="t82B2" mainDir="up">
24             <trackTopology>
25                 <trackBegin pos="5.000" id="right01">
26                     <connection ref="cLeft02" id="cRight01"/>
27                 </trackBegin>
28                 <trackEnd pos="10.000" id="right02">
29                     <openEnd id="x83"/>
30                 </trackEnd>
31             </trackTopology>
32             <ocsElements>
33                 <signals>
34                     <signal pos="5.000" id="s2" type="main" dir="down"/>
35                 </signals>
36             </ocsElements>
37         </track>
38     </tracks>
39 </infrastructure>
40 </railml>
```

Bild 3: railML code

Wie vorher erwähnt, kann die Struktur einer railML-Datei gegen das railML-Schema validiert werden. Die vorliegende Datei ist dem Schema entsprechend valide, da die Syntax-Konventionen eingehalten werden. Einem aufmerksamen Leser mag allerdings aufgefallen sein, dass die Datei semantische Fehler enthält. So beginnt der Gleisabschnitt „t82B1“ bei Kilometer 1000 (Zeile 10 – Element „pos“) und endet bei Kilometer 5000 (Zeile 13 – Element „pos“). Das ihm zugeordnete Signal „s1“ befindet sich allerdings bei Kilometer .500 (Zeile 19 – Element „pos“), also außerhalb des Intervalls des Gleisabschnitts. Das ist syntaktisch korrekt, entspricht jedoch einem Fehler semantischer Art. Ein weiterer semantischer Fehler zeigt sich bei der Ausrichtung „dir“ des Signals „s2“ (Zeile 34). Das Signal hat die Ausrichtung „down“, während der ihm zugeordnete Gleisabschnitt „t82B2“ die Fahrtrichtung „up“ (Zeile 23 – Element „mainDir“) aufweist.

Solche semantischen Fehler können nicht durch eine Schemavalidierung erkannt werden. Jedoch ist gerade die zuverlässige Identifikation semantischer Inkonsistenzen besonders wichtig, um eine Planung bzw. Projektierung einer Eisenbahninfrastruktur zu verifizieren. Wie im Abschnitt 1 beschrieben, ist die derzeitige manuelle Verifikation zeit- und kostenintensiv. Eine automatische Verifikation ist eine Möglichkeit zur Optimierung und Beschleunigung des Planungs- und Projektierungsprozesses.

Folgende Anforderungen sind an ein automatisches Verifikationswerkzeug zu stellen:

- Der Verifikationsprozess muss nachvollziehbar sein.
- Die Regeln müssen unterschiedliche Projektierungsrichtlinien unterschiedlicher Versionen abbilden können, ihnen eindeutig entsprechen und klar nachvollziehbar sein.
- Die Regelbasis der Verifikationsanwendung muss in sich konsistent, modular gestaltet und erweiterbar sein.

Um den Anforderungen zu entsprechen und eine konsistente und nachvollziehbare Modellierung der Konzepte und Zusammenhänge von Eisenbahninfrastrukturelementen in Bezug auf Planungs- und Projektierungsregeln zu ermöglichen, müssen technologische Konzepte erfüllt werden, die über ein XML-Schema-basiertes Datenaustauschformat wie railML hinausgehen. Im informationstechnischen Bereich der Wissensrepräsentation wird schon seit Längerem das Konzept von sogenannten „Ontologien“ für ähnliche Zwecke eingesetzt. Auch die diesem Artikel zu Grunde liegende Forschungsarbeit nutzt dieses Konzept, dass im Folgenden erläutert werden soll.

3 Verifikationssystem

3.1 Wissensbasis – Ontologien

Der Begriff „Ontologie“ stammt aus dem Griechischen, bedeutet frei übersetzt „Die Lehre des Seienden“ und bezeichnet ursprünglich eine Disziplin der Philosophie. In der Informatik wurde die Bezeichnung aufgegriffen und im Bereich der Wissensrepräsentation eingesetzt.

Ontologien erlangten im Zusammenhang mit dem Semantic Web einen gewissen Bekanntheitsgrad. Das Semantic Web beschreibt eine Weiterentwicklung des heutigen Internets. Die Neuerung besteht darin, dass die semantische Be-

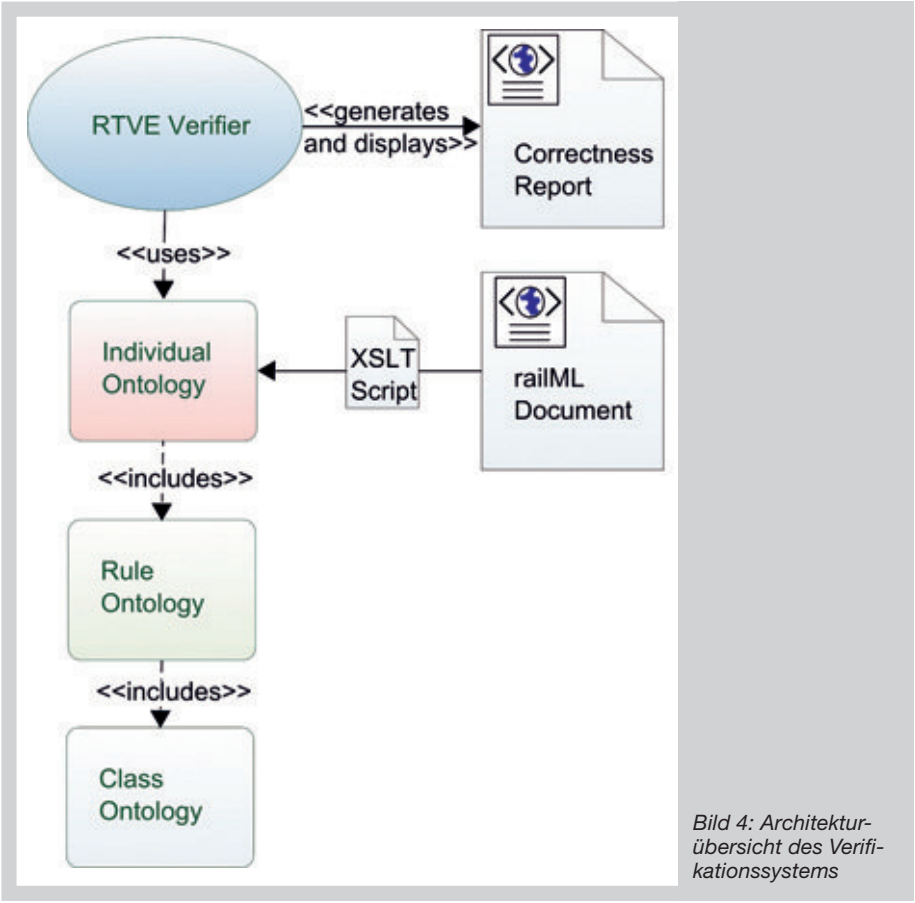

Bild 4: Architekturübersicht des Verifikationssystems

deutung von Webseiten und deren Komponenten formalisiert bereitgestellt wird, so dass sie auch von Computern interpretiert werden kann. Ein Programm kann nun z. B. „verstehen“, dass ein Benutzer sich die Webseite eines Arztes anschaut und über andere Webseiten die Bushaltestellen oder Apotheken in der Nähe heraussuchen. Dies wird ermöglicht durch eine Anreicherung der Webseite mit Meta-Informationen, die u. a. auch Verknüpfungen zu anderen Webseiten enthalten können. Die Menge von verknüpften Meta-Informationen einer Wissensdomäne wird als Ontologie angesehen.

Eine Ontologie ist somit im informationstechnischen Sinn definiert als die „explizit formale Spezifikation einer Konzeptualisierung“ [7]. Ontologische Modelle sind netzartig aufgebaut, im Gegensatz zu beispielsweise XML-Schemata wie railML, die einer Taxonomie gleichen, also hierarchisch gegliedert sind. Der Vorteil der netzartigen Struktur liegt in ihrer Ausdrucksfähigkeit. So werden ontologische Expertensysteme z. B. in der Medizintechnik eingesetzt, um Ärzte bei der Interpretation von Krankheitssymptomen und bei der Diagnoseentwicklung zu unterstützen.

In der Informatik findet die Ontologie-Beschreibungssprache OWL [8] weite Verbreitung. Mit ihr ist es möglich, ausdrucksstarke Ontologien formalisiert zu modellieren. OWL folgt einer XML-Syntax, geht aber weit über die Beschreibungsmöglichkeiten von regulärem XML hinaus.

3.2 Systemaufbau

Das Verifikationssystem (Bild 4) besteht aus unterschiedlichen Komponenten. Es gibt die ontologische Wissensbasis und die eigentliche Verifikationssoftware, welche die Wissensbasis nutzt, um eine Verifikation durchzuführen.

Die Wissensbasis des Verifikationssystems ist in drei Ebenen unterteilt: Die „Class Ontology“ beinhaltet alle Konzepte aus railML. Sie besteht im Wesentlichen (wie alle ontologischen Konzepte) aus Klassen (z. B. „Signal“), Eigenschaften (z. B. „Signalposition“) und Relationen zwischen Klassen (z. B. „Ein Signal ist einem Gleisabschnitt zugeordnet“ oder „Ein Gleisabschnitt ist benachbart mit einem anderen Gleisabschnitt“). Relationen erlauben mächtige Ausdrucksformen und sind somit nicht auf Typen beschränkt, die „ist-ein“- oder „hat-ein“-Beziehungen beschreiben. Sie ermöglichen unterschiedlichste Datentypenbezüge und Verknüpfungen. Bild 5 stellt grafisch einen Auszug der „Class Ontology“ dar. Man erkennt die netzartige Struktur einer Ontologie. Die orangefarbenen Ellipsen bilden die Basisklassen, von denen untergeordnete Klassen die Attribute erben. So ist z. B. ein „Track“ (Gleisabschnitt) ein „Relational Object“ und besitzt somit Nachbarschaftsrelationen. Ein „Relational Object“ wiederum ist eine Vererbung der Klasse „BaseID Name“ und weist somit eine eindeutige ID und einen bezeichnenden Namen auf. Ein „Signal“ wiederum erbt von „Directed Point Object“ und besitzt somit eine eindimensionale Positionsangabe. Die grünen Ellipsen beschreiben verifizierte Elemente. So kann ein Signal beispielsweise der Klasse „Correct Placement“ angehören, wenn es vom Verifikator als plausibel platziert klassifiziert wird.

Die Klassifikationen der einzelnen Elemente werden anhand von Regeln ermöglicht. Die „Rule Ontology“ beinhaltet diese Regelbasis, die die Konzepte der „Class Ontology“ nutzt. Aufbau und Struktur der einzelnen Regeln werden im Kapitel 3.3 näher erläutert. Der Vorteil einer regelbasierten Anwendung gegenüber einer konventionell programmierten Anwendung liegt in der Flexibilität und Erweiterbarkeit. Eine Steigerung des Funktionsumfangs des regelbasierten Verifikators und dessen Abdeckungsgrad – bezogen auf die Planungsrichtlinien – kann einfach durch das Hinzufügen von Konzepten und Regeln in der ontologischen Wissensbasis ermöglicht werden. Eine komplizierte Programmierung, um Funktionserweiterungen einzurichten, ist in einem solchen regelbasierten System nicht nötig.

Eine ontologische Wissensbasis kann grundlegend in eine konzeptionelle und eine individuelle Ebene unterteilt werden. Die „Class und Rule Ontologies“ bilden die konzeptionelle Ebene, während die „Individual Ontology“ die individuelle Ebene beschreibt. Hier werden, ausgehend von den Konzepten, konkrete Objekte instanziiert, wie z. B. „Signal (P1)“ oder „Gleis (G55) ist Nachbar von Gleis (G56)“ etc. Diese konkreten Daten stammen aus der Projektierung und Planung einer tatsächlichen Eisenbahnstrecke. Sie sind idealerweise in einem Planungstool, wie z. B. dem Projektierungstool des BEST-Simulators, entstanden und liegen im railML-Format vor. Mithilfe einer XSL-Transformation wird das railML-Dokument in eine Ontologie umgewandelt und in die Wissensbasis integriert. Bei dieser Technik werden in einer Skriptdatei sogenannte Templates verwendet, die beschreiben, wie Elemente eines Dokuments (im railML-Format) transformiert werden müssen, um Elementen eines an-

deren Dokuments (im OWL-Format) zu entsprechen. Diese Transformation wird anhand der XSLT-Skriptdatei automatisch durchgeführt.

Ist die Transformation vollzogen, wird die Verifikation durchgeführt. Als Ergebnis wird ein Bericht generiert, der zur Anpassung der Planungsdaten verwendet werden soll. Eine manuelle Verifikation ist nur noch für die nicht durch Regeln abgedeckten Fälle nötig.

3.3 Semantische Regeln

Ein wesentliches Charakteristikum von Ontologien besteht in der Möglichkeit der Erweiterung durch semantische Regeln. Der Regelaufbau folgt einem Voraussetzungs-Konsequenz-Prinzip und besteht aus einem Bedingungsteil (Körper) und einem Aktionsteil (Kopf). Körper wie auch Kopf bestehen aus Verknüpfungen einzelner Axiome. Die Axiome werden aus den definierten Klassen, Eigenschaften und Relationen gebildet. Zur Beschreibung der Implikations-Regeln wird die „Semantic Web Rule Language“ SWRL [9] verwendet. Ein großer Vorteil von SWRL besteht drin, dass sich, bedingt durch die OWL-Syntax der Regelsprache, SWRL-Regeln in OWL-Ontologien einbetten lassen und diese um verfeinerte Ausdrucksmöglichkeiten bereichern.

So können durch Regeln z. B. semantische Fehler detektiert werden. Bezogen auf die Fehler aus dem vorigen Beispiel (Bilder 3 und 4) können folgende Regeln zu Plausibilitätsprüfungen formuliert werden: „Das Signal eines Gleisabschnitts muss immer zwischen dessen Beginn und Ende positioniert sein.“ Diese konkrete Regel würde in der abstrakten SWRL Syntax folgendermaßen aussehen:

Signal(?s) ∧ Track(?t)
∧signalIsOnTrack(?s, ?t)
∧to(?t, ?to)
∧from(?t, ?from)
∧SignalPosition(?s, ?pos)
∧swrlb:greaterThanOrEqual(?pos, ?from)
∧swrlb:lessThanOrEqual(?pos, ?to)
-> CorrectPlacement(?s)

Im Erfolgsfall wird das Signal „s“ der Klasse „CorrectPlacement“ zugewiesen. Eine weitere Regel könnte folgendermaßen formuliert sein: „Ein Signal muss immer in Fahrtrichtung des zugehörigen Gleisabschnitts aufgestellt sein.“ Die SWRL Darstellung sieht aus wie folgt:

Signal(?s) ∧ Track(?t)
∧signalIsOnTrack(?s, ?t)
∧signalHasOrientation(?s, ?o)
∧trackHasDirection(?t, ?o)
-> CorrectOrientation(?s)

Hier wird, wenn die Signalausrichtung „o” gleich der Fahrtrichtung „o” des Gleisabschnitts ist, das Signal „s“ der Klasse „CorrectOrientation“ zugewiesen.

Das sind relativ banale Beispiele, die jedoch Grundbausteine für wesentlich umfangreichere Regelsätze bilden können, denn ein weiterer Vorteil von Ontologien besteht darin, dass die Regeln hierarchisch gegliedert werden können. Das ermöglicht die Einführung unterschiedlicher Abstraktionsniveaus und erlaubt eine gute Übersichtlichkeit bei einem hohen Komplexitätsgrad. Ein Beispiel hierzu wäre die Regel zur Formulierung der Richtlinie, dass ein Vorsignal immer vor einem Hauptsignal stehen muss. Dazu müssen die Signale korrekt platziert und korrekt ausgerichtet sein (vgl. Elemente/Relationen in Bild 5):

Signal(?s1) ∧ CorrectPlacement(?s1)
∧CorrectOrientation(?s1)
∧hasSignalType(?s1, ?type1)
∧hasDirection(?s1, ?dir)
∧isOnRO(?s1, ?t1)
∧sameAs(distant, ?type1)
∧Signal(?s2)

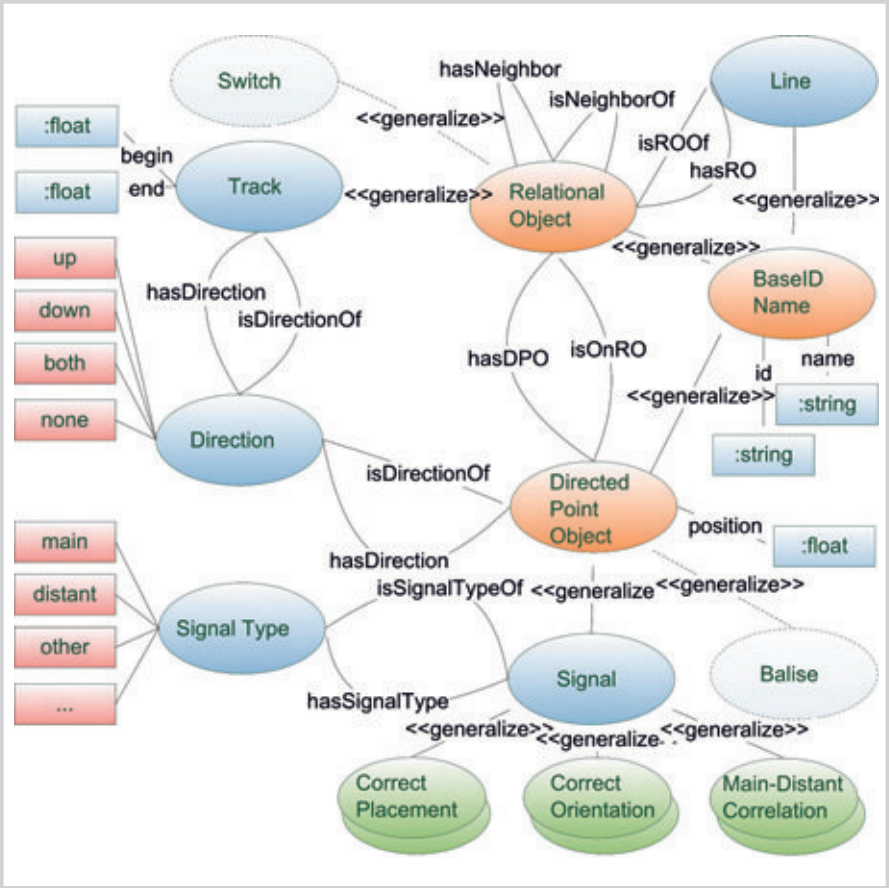

Bild 5: Auszug aus der „Class Ontology“ des Verifikationssystems

∧CorrectPlacement(?s2)
∧CorrectOrientation (?s2)
∧hasSignalType(?s2, ?type2)
∧hasDirection(?s2, ?dir)
∧dPOIsOnRO(?s2, ?t2)
∧isMajorNeighbourOf(?t2, ?t1)
∧sameAs(main, ?type2)
-> CorrectMainDistantCorrelation(?s1)
∧CorrectMainDistantCorrelation(?s2)

4 Fazit

Am Beispiel von railML wurde beschrieben, wie ein digitales Datenaustauschformat im Planungsprozess von Anlagen der Leit- und Sicherungstechnik eingesetzt werden kann, um Infrastrukturdaten anwendungsübergreifend verfügbar zu machen. Im Rahmen einer Forschungsarbeit wurde mit der Entwicklung eines Verifikationssystems eine Grundlage geschaffen, Planungsdaten von Eisenbahninfrastrukturen automatisiert auf die Einhaltung von Planungs- und Projektierungsrichtlinien zu überprüfen. Dies ermöglicht, den kosten- und zeitintensiven Aufwand der manuellen Verifikation der Daten zu minimieren. Das System ist regelbasiert und ermöglicht somit einen hohen Flexibilitätsgrad in Bezug auf Erweiterbarkeit, Änderbarkeit und Wartbarkeit. Desweiteren besitzt eine solche Systemarchitektur den Vorteil, die unterschiedlichen Versionen von Planungsrichtlinien mit Hilfe entsprechender versionierter Regelsätze abbilden zu können. Auch in den verschiedenen Phasen des LST-Planungsprozesses, die sich vor allem im Detailierungsgrad unterscheiden, kann mit entsprechend angepassten Regelsätzen eine Verifikation erfolgen. So kann bereits in frühen Stadien des Planungsprozesses eine automatische Verifikation erfolgen und damit der gesamte Planungsprozess optimiert werden.

LITERATUR

[1] Deutsche Bahn AG: LST-Anlagen planen – Richtlinie 819
[2] Planungstool Prosig: http://www.prosig.de/
[3] Seemann/Demitz: Betriebs- und Stellwerkssimulation in der ESTW-Planung – Pilotprojekt Dresden-Neustadt, S+D 4/2008
[4] Bahr/Tschorn/Sänger: Das SPS-basierte ESTW Alister 2.0 im Pilotprojekt ESTW-R Lindaunis, S+D 5/2009
[5] railML-Initiative: Das railML-Schema, http://railml.org – Stand: November 2009
[6] Dublin Core Metadata Initiative: Expressing Dublin Core metadata using CML, http://dublincore.org/documents/dc-xml/ – Stand November 2009
[7] Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2): 199–220, 1993
[8] W3C-Rcommendation: OWL Web Ontology Language, http://www.w3.org/TR/owl-guide/ – Stand: November 2009
[9] W3C-Submission: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/SWRL/ – Stand: November 2009

Die Autoren

Prof. Dr.-Ing. Norbert Luttenberger
Leiter des Instituts für Informatik,
Christian-Albrechts-Universität zu Kiel
Anschrift: Chr.-Albrechts-Platz 4,
D-24118 Kiel
E-Mail: nl@informatik.uni-kiel.de

Michael Lodemann
Wissenschaftlicher Mitarbeiter der AG Commsys, Institut für Informatik der Christian-Albrechts-Universität zu Kiel
Anschrift: Christian-Albrechts-Platz 4,
D-24118 Kiel
E-Mail: milo@informatik.uni-kiel.de

Dr. Carsten Gerke
Projektleiter, Funkwerk Information Technologies GmbH
Anschrift: Edisonstraße 3, D-24145 Kiel
E-Mail: carsten.gerke@funkwerk-it.com

SUMMARY

Description and verification of railway infrastructure with railML

The article describes the development of a rule-based verification system for optimising the planning process of railway infrastructure and safety elements. The system uses technologies of knowledge representation and enables mapping of planning and engineering guidelines as extendable sets of rules. During automatic verification a test report is generated which greatly reduces the time and work involved in manual testing. railML, a railway-specific XML schema, is used as transfer format between planning application and the verification system.

MARKET LEADING QUALITY

axle counting & wheel detection

Reliable all over the world!

FRAUSCHER

SENSOR TECHNOLOGY

RELY ON THE BEST –
Frauscher systems are well-established in 5 continents, over 45 countries and a wide range of application areas:

- **Main lines**
- **Regional lines**
- **Marshalling yards**
- **Metros, mass transit**
- **Level crossings**
- **Various applications**

  (hot box detection, switching/trigger tasks, warning systems)

Visit us at INFRARAIL 2010
April, 13th–15th | booth 873

www.frauscher.com