Schweizerische Eidgenossenschaf
Cinfédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement für Umwelt, Verkehr, Energie und Kommunikation UVEK
Département fédéral de l'environement, des transports, de l'énergie et de la communication DETEC
Direzione federale dell'ambiente, dei trasporti, dell'energia e delle communicazione DATEC

Bundesamt für Strassen
Office fédéral des routes
Officio federle delle Strade

MDAinSVT

Einsatz modellbasierter Datentransfernormen (INTERLIS) in der Strassenverkehrstelematik am Beispiel der Verkehrsdaten

Utilisation des standards d'échange de données basés modélisation pour la télématique des transports routiers à l'exemple des données de trafic

Use of model driven data transfer standards in the road transport telematic exemplified by trafic data

Eidgenössische Technische Hochschule Zürich (ETHZ)
IGP Gruppe GIS und Fehlertheorie
Hans Rudolf Gnägi, dipl. Mathematiker
Stefan Henrich, dipl. Kulturing. ETHZ

AWK Group, Zürich Oerlikon
Martina Münster, dipl. Bauing. ETHZ
Roger Rüegg, dipl. Informatiking. ETHZ

Eisenhut Informatik AG, Burgdorf
Claude Eisenhut, dipl. Informatiking. FH

Forschungsauftrag VSS 2007/902 auf Antrag des
Verbandes Schweizer Strassen- und Verkehrsfachleute (VSS)

April 2009 **1267**

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement für Umwelt, Verkehr, Energie und Kommunikation UVEK
Département fédéral de l'environnement, des transports, de l'énergie et de la communication DETEC
Dipartimento federale dell'ambiente, dei trasporti, dell'energia e delle comunicazioni DATEC

Bundesamt für Strassen
Office fédéral des routes
Ufficio federale delle Strade

MDAinSVT

Einsatz modellbasierter Datentransfernormen (INTERLIS) in der Strassenverkehrstelematik am Beispiel der Verkehrsdaten

Utilisation des standards d'échange de données basés modélisation pour la télématique des transports routiers à l'exemple des données de trafic

Use of model driven data transfer standards in the road transport telematic exemplified by trafic data

Eidgenössische Technische Hochschule Zürich (ETHZ)
IGP Gruppe GIS und Fehlertheorie
Hans Rudolf Gnägi, dipl. Mathematiker
Stefan Henrich, dipl. Kulturing. ETHZ

AWK Group, Zürich Oerlikon
Martina Münster, dipl. Bauing. ETHZ
Roger Rüegg, dipl. Informatiking. ETHZ

Eisenhut Informatik AG, Burgdorf
Claude Eisenhut, dipl. Informatiking. FH

Forschungsauftrag VSS 2007/902 auf Antrag des Verbandes Schweizer Strassen- und Verkehrsfachleute (VSS)

April 2009 **1267**

AWK Group
Consulting and Engineering

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Departement Bau, Umwelt und Geodäsie
Institut für Geodäsie und Photogrammetrie
Gruppe GIS und Fehlertheorie

Eisenhut
Informatik AG

Projekt MDAinSVT

Einsatz modellbasierter Datentransfernormen (INTERLIS) in der Strassenverkehrstelematik am Beispiel der Verkehrsdaten

Hans Rudolf Gnägi, ETHZ IGP GF
Stefan Henrich, ETHZ IGP GF
Martina Münster, AWK Group
Roger Rüegg, AWK Group
Claude Eisenhut, Eisenhut Informatik AG

Version 2.6

Zürich – April 2009

Inhaltsverzeichnis

1. Zusammenfassung

In der Strassenverkehrstelematik der Schweiz sind auf den verschiedenen Ebenen der Leittechnik sehr unterschiedliche Systeme im Einsatz mit sehr verschiedenen Schnittstellen. Entsprechend gibt es eine Vielzahl von Transferformaten für den Datenaustausch zwischen diesen Systemen. Diese Arbeit zeigt, dass mit dem modellbasierten Vorgehen eine Methode zur Verfügung steht, mit der die vielen proprietären Transferformate auf einige wenige Standardformate reduziert werden können. Dabei ist das modellbasierte Vorgehen einerseits zweckmässig für die Herleitung der Standardformate, ausgehend von den gegebenen proprietären Formaten. Es kann andererseits auch verwendet werden für die automatische Qualitätsprüfung der Transferdaten und für die Ablösung der proprietären Formate durch Standardformate, ohne die beteiligten Systeme selbst umbauen zu müssen.

Zunächst wird die Ausgangslage am Beispiel der Verkehrdatenerfassung als Bestandteil der Strassenverkehrstelematik und die entsprechende Abgrenzung des Forschungsprojekts erläutert (Kapitel 2). Dann wird das modellbasierte Vorgehen (Model Driven Approach MDA) dargestellt, seine Wurzeln, die 5 wesentlichen Elemente des MDA und wie diese zusammenspielen beim system-neutralen Strukturumbau (Kapitel 3).

Die folgenden 5 Kapitel zeigen, dass und wie die 5 Elemente des modellbasierten Vorgehens auf die Verkehrsdatenerfassung der Strassenverkehrstelematik anzuwenden sind.

1) Beschreibung des Realitätsausschnitts in Umgangssprache (Kapitel 4)
2) Konzeptionelle Datenmodellierung (Kapitel 5)
3) Automatisch hergeleitete Beschreibungen der Standard-Transferformate (Kap. 6)
4) Formatumbau vom proprietären zum Standard mit dem 1:1 Prozessor (Kap. 7)
5) Strukturumbau mit Hilfe semantischer Transformation (Kapitel 8)

Schliesslich wird das Konzept des Demonstrator-Programms erläutert und welche Elemente im Rahmen dieses Projektes realisiert werden konnten (Kapitel 9).

Wie in den Schlussfolgerungen (Kapitel 10) festgehalten ist, bringt der Einsatz des modellbasierten Vorgehens und insbesondere der modellbasierten Transfermethode mit UML und INTERLIS für den Datentransfer in der SVT die erwarteten wesentlichen Vorteile. Man erhält einheitliche, präzise und vergleichbare Spezifikationen der Daten von Schnittstellen verschiedener Typen durch Datenmodelle. Auf dieser Basis ist trotz Vielfalt der beteiligten Systeme eine Harmonisierung, Vereinfachung und automatische Qualitätskontrolle des Datentransfers möglich. Dies ist realisierbar, wie das implementierte Demonstrator-Programm zeigt, ohne dass existierende Systeme sofort modifiziert oder ersetzt werden müssen.

Als nächster Schritt drängt sich die Implementierung des Demonstrators in der Realität des Testgebietes auf. Als weiteres Forschungsthema zeigt sich kurzfristig die Frage nach modellbasierter Behandlung der Koordinatentransformation (z.B. zwischen Kilometrierung bzw. RBBS und Landeskoordinaten) und längerfristig der Vergleich einer hierarchischen und einer „flachen“ Kommunikationsstruktur.

Ein Schwergewicht des Projektes lag bei Modellierung und Programmierung, was im Bericht nur auszugsweise wiedergegeben werden kann. Auch müssen gewisse Informatikkenntnisse beim Leser für das Verständnis vorausgesetzt werden.

1. Résumé

La télématique des transports routiers (TTR) en Suisse a recours à beaucoup de systèmes comportant des interfaces différentes et fonctionnant sur des niveaux spécifiques de technique de contrôle. Beaucoup de transformations sont donc nécessaires afin de pouvoir échanger des données entre ces systèmes. Cette recherche montre que l'approche basée modélisation (en anglais : model driven approach MDA) est une méthode qui permet de réduire le grand nombre de formats propriétaires à seulement quelques formats standards. D'une part, l'approche basée modélisation permet de dériver des standards indépendants des formats propriétaires. D'autre part elle s'utilise aussi pour tester automatiquement la qualité des données, pour les restructurer si nécessaire et pour remplacer les formats propriétaires par des standards sans devoir modifier ou remplacer immédiatement les systèmes concernés.

Dans un premier temps, la situation initiale est expliquée à l'aide de l'exemple du levé des données de trafic comme partie intégrante de la TTR, et puis le domaine et les limites du projet de recherche sont fixés (chapitre 2). Ensuite l'approche basée modélisation est introduite, en décrivant son origine, ses 5 éléments principaux et leur collaboration pour réaliser des restructurations de données d'une manière indépendante des systèmes (chapitre 3).

Les 5 chapitres suivants montrent comment les 5 éléments du MDA s'appliquent au levé des données trafic de la TTR.

1) Description de la sélection de réalité (du monde réel) dans un language naturel (ch. 4)
2) Modèlisation conceptuelle de données avec UML et INTERLIS 2 (chapitre 5)
3) Déduction automatique de la description du format standard correspondant au modèle conceptuel de données (chapitre 6)
4) Transformation des données du format propriétaire au format standard par un processeur 1:1 (chapitre 7)
5) Conversion de la structure des données par transformation sémantique (chapitre 8)

Finalement le concept du programme démonstrateur est expliqué et on présente les parties réalisées dans le cadre du projet (chapitre 9).

Les conclusions (chapitre 10) montrent, que l'application du MDA et en particulier celle du transfert des données basé modélisation à l'aide de UML et d'INTERLIS offre vraiment les avantages espérés : grâce aux modèles conceptuels des données, les spécifications d'interfaces très diverses deviennent uniformes, précises et comparables. Sur cette base, il devient possible – malgré l'hétérogénéité des systèmes concernés – d'harmoniser, de simplifier et de tester automatiquement le transfert des données. Et, comme le montre le programme démonstrateur implémenté, il n'est pas nécessaire de modifier ou de remplacer dans l'immédiat les systèmes actuels.

Un prochain pas serait de tester le programme démonstrateur dans la réalité sur une région test. Comme poursuite de la recherche, on propose de s'occuper de la question des transformations des systèmes de coordonnées curvilinéaires de la TTR intégrées dans le MDA (par exemple entre le système de repérage spatial de base (SRB) et le système des coordonnées nationales). À plus long terme, on pourrait envisager une recherche sur la comparaison basée modélisation des structures de communication TTR hiérarchiques et « plates » .

Modélisation conceptuelle et programmation étaient des points principaux de ce projet, mais ils ne peuvent être présentés que partiellement. En plus, quelques connaissances d'informatique sont nécessaires pour pouvoir suivre différents arguments.

1. Summary

Many different systems with very different interfaces are used in the road traffic telematic (RTT) of Switzerland on the specific layers of control technique. Thus,a lot of transformations are needed to enable data exchange between these systems. This report shows the model driven approach being a method which allows reducing the many proprietary to only several standard transfer formats. On the one hand the model driven approach is useful to derive standard formats starting from the given proprietary ones. On the other hand it can also be used for automatic quality checking for transferring data and for replacing the proprietary formats by standards without the necessity to replace or rebuild the systems concerned.

First, the start situation will be explained with the example of traffic data capture as part of the road traffic telematic, and then the extent and delimitations of the research project (chapter 2). In the following the model driven approach (MDA) is presented, its roots, its 5 principle elements and how they collaborate to provide system-neutral data restructuring (chapter 3).

The next 5 chapters show how the 5 elements of the MDA can be applied to the capture of road traffic data in the RTT:

1) Description of the reality selection (universe of discourse) in natural language (ch. 4)
2) Conceptual data modeling with UML and INTELIS 2 (chapter 5)
3) Automatic derivation of the standard transfer format description from the conceptual data model (chapter 6)
4) Reformatting from proprietary to standard by 1:1-processors (chapter 7)
5) Data structure conversion with semantic transformation (chapter 8)

Finally, the concept of the demonstrator program is explained and which elements have been realized during this project (chapter 9).

The conclusions (chapter 10) show, that the application of the MDA and especially of the model driven transfer method with UML and INTERLIS to exchange data in the RTT provides the expected important advantages. Uniform, precise, comparable and human readable specifications result for the data of very different interfaces by data models. On this basis it is possible – despite the heterogeneity of the systems – to harmonize, simplify and automatically control the data transfer. And, as shown by the implemented demonstrator program, this is possible without the need to (immediately) modify or replace existing systems.

In the next step, the demonstrator should be implemented in the real world of the test region. Follow-up research should focus on the question, how to integrate the curvilinear coordinate systems of RTT and their transformations into the MDA. A long-term research topic might deal with the MDA-based comparison of hierarchical and “flat” communication structures in the RTT.

The main goal of this project was conceptual modeling and programming, which can only partially be presented in this report. In addition some ICT knowledge is a prerequisite for the reader in order to understand some arguments.

2. Ausgangslage Verkehrdatenerfassung und Abgrenzung des Forschungsgebiets

2.1. Referenzarchitektur

Ein Leitsystem der SVT wird typischerweise nach den Grundätzen der Leittechnik [14] in mehrere Ebenen aufgeteilt. Jede Ebene ist hierbei grundsätzlich von der nächsthöheren Ebene unabhängig, kann also bei Ausfall der höheren Ebene unabhängig weiter funktionieren.

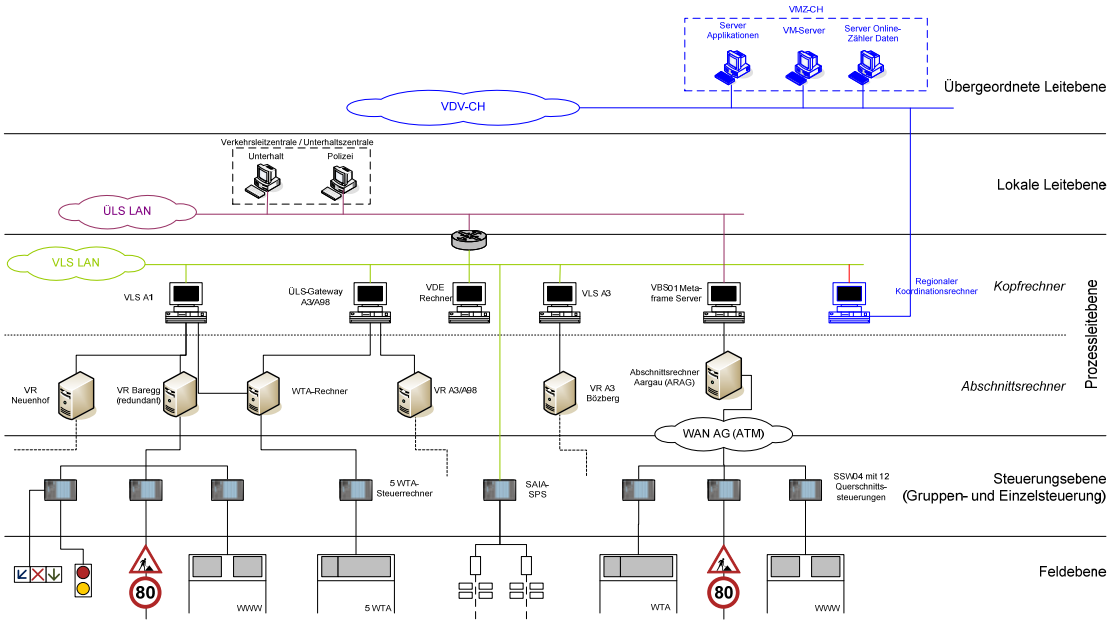


Abbildung 1 Beispielarchitektur SVT Kt. Aargau (Quelle: [5])

2.1.1. Feldebene

In der Feldebene befinden sich die Aktoren und Sensoren des Systems. Typischerweise kommunizieren die Elemente der Einzelsteuerebene mit den Aktoren und Sensoren über Relais-Kontakte oder analoge elektrische Signale. Die Geräte der Feldebene kommunizieren mit der Steuerungsebene mittels Feldbussystemen, die meist ein deterministisches zyklisches Zeitverhalten aufweisen und damit das Echtzeit-Verhalten des Systems sicherstellen können. Die Funktionalität der Feldebene beschränkt sich im Wesentlichen auf das Weiterleiten der Sensorzustände und Befehle, sowie auf das Weiterführen eines sicheren Betriebszustandes bei Feldbus-Ausfall. Die eigentliche Logik des Systems ist in den höheren Ebenen realisiert.

2.1.2. Steuerungsebene (Einzel- und Gruppensteuerung)

In der Steuerungsebene befinden sich die Streckenstationen, die zur Erfassung von Prozessdaten wie Verkehrs- und Umfelddaten in aggregierter Form und das Absetzen von Steuer- und Regelungsbefehle an die zugeordneten Anlagenteile eingesetzt werden. Meist werden die Streckenstationen basierend auf einer speicherprogrammierbaren Steuerung (SPS) aufgebaut. Die Kommunikation zwischen Feld- und Steuerungsebene erfolgt heute meist über Feldbusse.

Auf Stufe Gruppensteuerung steht normalerweise eine lokale Bedienmöglichkeit zur Verfügung, die jedoch nur in Ausnahmefällen genutzt wird (Rückfallebene). Die Steuerung durch die Bediener erfolgt im Normalfall von der Prozessleitebene aus.

2.1.3. Prozessleitebene

In der Prozessleitebene ist die eigentliche Verarbeitungslogik eines Prozesses realisiert und beinhaltet die Steuerung, Regelung und Überwachung der Anlage. Meist wird diese Ebene mit Industrie-PC oder SPS realisiert. Die Kommunikation von der untergeordneten Steuerungsebene zur Prozessleitebene erfolgt heute meist mit IP-basierten Prozessbussen.

Die Prozessleitebene wurde bisher abhängig von den kantonsspezifischen Architekturen in mehrere Sub-Ebenen unterteilt.

2.1.4. Lokale Leitebene

Auf der lokalen Leitebene steht die Überwachung und Steuerung des Geschehens durch die Bediener im Vordergrund. Eine wichtige Funktion ist die Alarmierung der Bediener bei ungewöhnlichen Betriebszuständen. Zudem werden auf dieser Ebene die Prozessdaten historisiert und archiviert; es stehen auch Funktionen zur Auswertung der Prozesse über längere Zeiträume zur Verfügung (Statistiken, Trends).

2.1.5. Übergeordnete Leitebene

Die Rolle der übergeordneten Leitebene liegt in der übergeordneten Verarbeitung aller Prozessdaten der Systeme zur Herleitung von Handlungsempfehlungen an die Bediener. Direkte Schaltbefehle an die Leitebene werden meistens nicht ausgelöst. Die übergeordnete Leitebene wird auch Strategieebene genannt.

2.2. Ziele des Forschungsprojektes gemäss Projektdefinition

Die Zielsetzung kann dem ARAMIS Formular 2.2, Kreditbegehren (Projektinhalt) entnommen werden. Gegenüber der dortigen Formulierung sind die 3 Punkte von Ziel Z1 in einem Satz zusammengefasst und die Ziele Z2 und Z3 kommen vor Z4 statt nach Z5.

Z1. Erarbeiten von Lösungsmöglichkeiten für Schnittstellenspezifikation und Datenflussorganisation mit Hilfe des modellbasierten Vorgehens entsprechend den

Bedürfnissen von Strassendaten allgemein und der SVT im Besonderen (z.B. Real Time Aspekte, spezielle Datentypen) .

Z2. Aufzeigen, wie die Forschungsergebnisse auf weitere SVT-Systeme anwendbar sind.

Z3. Realisierung und Implementierung eines Demonstrators mit den erarbeiteten Lösungen für ein Praxisbeispiel.

Z4. Aufzeigen der Möglichkeiten und Grenzen des modellbasierten Vorgehens zum Schaffen der datentechnischen Voraussetzungen für automatische Koordinatentransformationen zwischen verschiedenen in der SVT gebräuchlichen Referenzsystemen.

Z5. Die Resultate sollen als Grundlage verwendet werden können für die Realisierung des Zielzustandes VM-CH 2012 (hochautomatischer und multimodaler Datenaustausch auf und zwischen internationaler, nationaler und regionaler Ebene).

2.3. Abgrenzung des Forschungsgebietes

Als Forschungsgebiet (Untersuchungsgegenstand des Forschungsprojektes) für Anwendungen der SVT werden folgende Funktionen bzw. die dahinter stehenden Systeme betrachtet:

- lokales System zur Erfassung des Verkehrsaufkommens
- (über)regionales Verkehrsdaten-System (VD)
- Verkehrsmanagementsystem (VM)
- Verkehrslenkungssystem (VL)

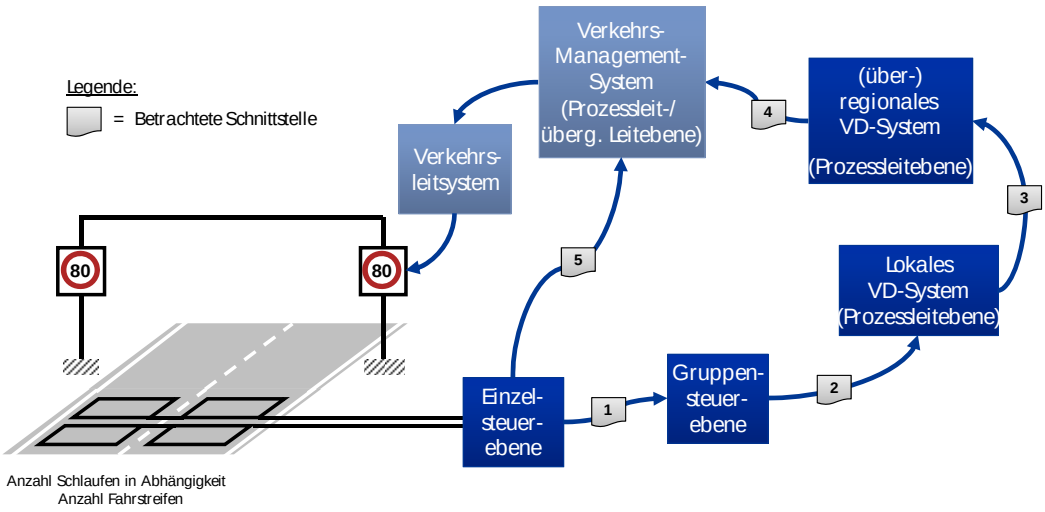


Abbildung 2: Komponenten und Schnittstellen des Forschungsgebietes

Das betrachtete Forschungsgebiet verfügt über die im Folgenden aufgeführten Aktoren und Sensoren.

- Sensoren
  - Induktionsschlaufen
  - Infrarot
  - Ultraschall
  - Laser
  - Videokameras
- Aktoren
  - Signalisation (Geschwindigkeits-, Gefahren-, Fahrstreifen(licht)signale, Überholverbot, Wechselwegweiser, Wechseltextanzeigen etc.)

Im Rahmen der Forschungsarbeit werden insbesondere die folgenden Schnittstellen betrachtet (s. Abbildung 2):

1) Einzelsteuerebene ↔ Gruppensteuerebene

2) Gruppensteuerebene ↔ Lokales VD-System (Prozessleitebene)

3) Lokales VD-System (Prozessleitebene) ↔ (über-)regionales VD-System (Prozessleitebene)

4) (über-)regionales VD-System (Prozessleitebene) ↔ Verkehrsmanagementsystem (Prozessleit-/übergeordnete Leitebene)

5) Einzelsteuerebene ↔ Verkehrsmanagementsystem (Prozessleit-/übergeordnete Leitebene)

Es ist nicht Gegenstand dieses Forschungsauftrages, die betrachteten Systeme und deren Konzepte zu hinterfragen bzw. zu bewerten.

Der hierarchischen Struktur von Ebenen der Leittechnik entspricht eine hierarchische Kommunikationsstruktur. Jede Ebene der Leittechnik gibt ihre Auswertungen an die nächst höhere weiter. Datentransfer in dieser hierarchischen Kommunikationsstruktur ist Gegenstand dieses Projektes. Hingegen wäre es allenfalls interessant, in einem anderen Projekt die Frage zu beantworten, ob statt der hierarchischen eine flache Kommunikationsstruktur zweckmässig wäre. Flach heisst im Gegensatz zu hierarchisch in unserem Fall eine Kommunikationsstruktur, bei der nur die Ereignismeldungen der Zählstellen übermittelt werden, aber an alle Ebenen der Leittechnik. Auf jeder Ebene der Leittechnik werden aus diesem Meldungsstrom die benötigten Meldungen ausgewählt, um damit die gewünschten Statistiken zu berechnen.

2.4. Beschreibung Verkehrsdatenerfassung

Gemäss [3] umfasst die Verkehrsdatenerfassung die folgenden Funktionsblöcke:

- Erfassung des Verkehrsaufkommens
- Ereignis- und Gefahrenerfassung
- Erfassung der Strassenverhältnisse
- Belegungs- und Abfertigungserfassung

- Zustandserfassung öffentlicher Verkehr

Die Forschungsarbeit befasst sich im Rahmen der Verkehrsdatenerfassung mit dem Funktionsblock „Erfassung des Verkehrsaufkommens“ und den Schnittstellen zu den übergeordneten Systemen. Die Funktion zur Erfassung des Verkehrsaufkommens misst Eigenschaften von Fahrzeugen auf dem Strassennetz, welche sich auf die Bewegungsparameter dieser Fahrzeuge sowie auf Eigenschaften für die Zuordnung zu Fahrzeugklassen und für die Wiedererkennung der Fahrzeuge an unterschiedlichen Stellen des Strassennetzes beziehen.

Mittels Sensorik im Strassennetz (z.B. Induktionsschleifen) werden die Fahrzeuge an ortsfesten Stellen (Zählstellen) erfasst und dem Verkehrsdatenerfassungssystem (VDE) zugeführt. In der VDE werden die Rohdaten zu einer Verkehrszustandsinformation pro Messsegment aufbereitet und die Einzelfahrzeugdaten für die Wiederverwendung in den weiteren Ausbaustufen des Systems abgelegt.

Die ermittelten Messgrössen der VDE werden einem (über-)regionalen VD-System und/oder einem Verkehrsmanagementsystem zugeführt. Unter Berücksichtigung des momentanen Signalisierungszustands und interner Plausibilitätsüberprüfungen (Zeitpunkt, Verträglichkeit mit Nachbarzonen, Längsabgleich) werden Szenarien für Geschwindigkeitsschaltungen bzw. Stauwarnungen an entsprechende Verkehrsbeeinflussungssystem (VBS) weitergegeben und dynamisch vollzogen.

2.5. Terminologie

Im Rahmen des Forschungsprojektes ist eine einheitliche Terminologie zu verwenden. Als Grundlage gelten die Begriffsdefinitionen des Normenentwurfs „SVT – Funktionale Systemarchitektur“ [3]. Zusätzliche Begriffe sind Kapitel 12 zu entnehmen.

3. Das modellbasierte Vorgehen

3.1. Grundsatz

Das modellbasierte Vorgehen soll eingesetzt werden, um Lösungsmöglichkeiten für Schnittstellenspezifikation und Datenflussorganisation mit Strassendaten allgemein und speziell mit Daten der SVT zu erarbeiten.

Das modellbasierte Vorgehen wird zunächst am Beispiel des Datentransfers eingeführt (3.2). Dann werden die 5 Elemente des modellbasierten Vorgehens vorgestellt (3.3) und es wird gezeigt, wie diese 5 Elemente zusammenspielen, um u.a. Datenstrukturumbau mit Hilfe semantischer (bzw. genauer Semantik erhaltender) Transformation zu realisieren (3.4). Ferner wird gezeigt, wie sich die Modellierungssprachen UML und INTERLIS 2 unterscheiden (3.5), wie INTERLIS 2 gegenüber ASN1 abzugrenzen ist (3.6) und wie sich das vorliegende Projekt in der Normenwelt positioniert (3.7).

3.2. Modellbasierter Datentransfer

Das Konzept des modellbasierten Vorgehens ist am Besten zu verstehen am Beispiel des Datentransfers. Angenommen, wir haben ein Startsystem *S*, aus dem (Geo-) Daten transferiert werden sollen in ein Zielsystem *Z* . Dabei soll es keine gemeinsame Schnittstelle geben und mindestens eines der beiden Systeme soll kein Standardformat irgendeiner Art produzieren bzw. lesen können.

Wie geht man normalerweise vor? Kenner der beiden Systeme setzen sich zusammen und definieren ein Transferformat. Sie können sich für ein XML-Format (extensible markup language) entscheiden, dann legen sie fest, welche Tags gewählt werden sollen für die zu transferierenden Werte. Entscheiden sie sich für ein csv-Format (comma separated values), dann müssen sie festlegen, wie die Zeilen in Datenfelder zu gliedern sind und welche Werte in die Datenfelder zu setzen sind.

Wesentlich anders ist das modellbasierte Vorgehen. Dabei steht nicht das Transferformat, sondern die Datenstruktur der auszutauschenden Daten im Vordergrund. Es geht zunächst nicht um die Frage der Tags oder Felder der Transferdatei, sondern darum, wie diese Daten strukturiert sind, aus welchen Objekten diese Daten bestehen, welche Eigenschaften (Attribute) diese Objekte haben, welche Beziehungen (Assoziationen) zwischen den Objekten bestehen, wie die Objekte zu Klassen zusammengefasst werden können etc. Diese Datenstruktur beschreibt man durch ein Datenmodell, unabhängig vom System, auf dem sich die Daten befinden, und unabhängig vom Format, mit dem sie transferiert werden sollen, eben konzeptionell. Das (konzeptionelle) Datenmodell (conceptual schema) ist nichts anderes als die exakte Beschreibung der Datenstruktur mit einer formalen Sprache, ähnlich einer Programmiersprache, einer sogenannten (konzeptionellen) Datenbeschreibungssprache (Data Description Language DDL oder Conceptual Schema Language CSL) wie INTERLIS, EXPRESS, UML, OWL. Und erst jetzt, wenn das Datenmodell vorliegt, wendet man sich dem Transferformat zu. Die Beschreibung des Transferformats kann nämlich aus dem Datenmodell hergeleitet werden entsprechend den Codierungsregeln, die für die verwendete Datenbeschreibungssprache normativ festgelegt sind (z.B. für INTERLIS im Kapitel 3 des

Referenzhandbuchs, für EXPRESS in Part 42). Die benötigte Beschreibung der Tags oder Felder der Transferdatei bekommt man also automatisch, sobald man das Datenmodell erstellt hat. Entsprechend dieser Formatbeschreibung wird die Transferdatei erstellt. Transferiert werden dann nicht nur die Daten, sondern auch das Datenmodell, wie in Abbildung 3 symbolisch dargestellt.

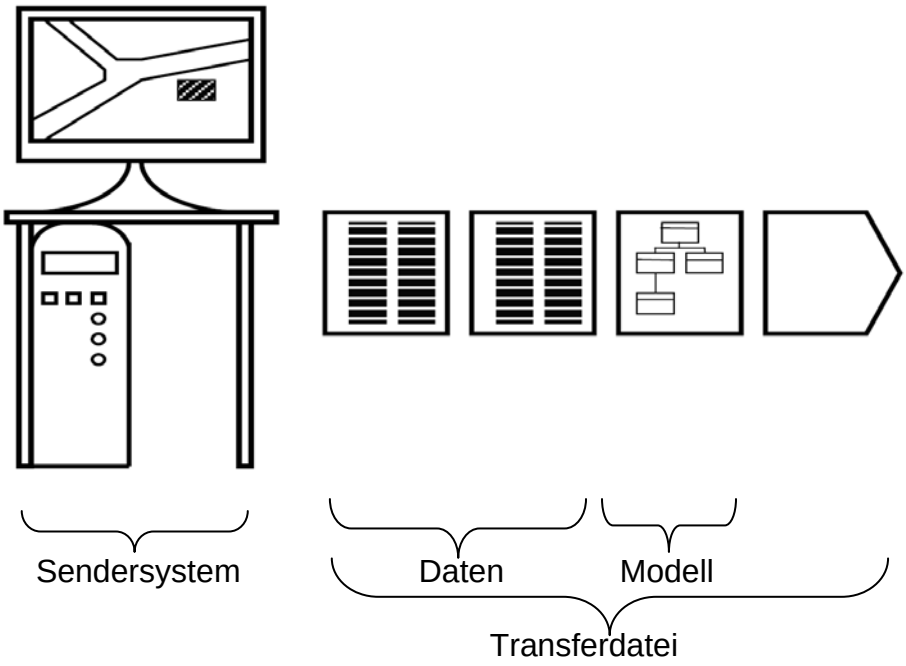


Abbildung 3: Prinzip des modellbasierten Datentransfers

Wie wir später sehen werden, kann man mit Datenmodell und Daten im entsprechenden Standardformat nicht nur Daten klar dokumentiert transferieren und bei Bedarf auch sichern, sondern auch automatisch prüfen (Kapitel 9.3) und umstrukturieren durch Definition des Modellumbaus auf konzeptionellem Niveau (Kapitel 8).

3.3. Die 5 Elemente des modellbasierten Vorgehens

Ein wesentlicher Vorteil des modellbasierten Vorgehens ist, dass man vom proprietären Format eines Startsystems mit bestimmter Datenstruktur ausgehen kann und nach Strukturumbau bei einem anders strukturierten Zielsystem wieder im entsprechenden proprietären Format ankommen kann. Dazu braucht es die nachfolgend aufgeführten fünf wesentlichen Elemente (A) bis (E) des modellbasierten Vorgehens. In Abschnitt 3.4 ist der Prozess beschrieben, wie man Daten zwischen verschieden strukturierten Systemen gleichen Inhalts transferiert. In diesem Prozess kommt jedes der fünf Elemente an mehreren Stellen zum Einsatz und es wird gezeigt, wie die Elemente zusammenarbeiten.

(A) Der **Realitätsausschnitt** (reality selection, universe of discourse) ist in Umgangssprache so präzis wie möglich zu beschreiben. In unserem Fall ist dies die Datenschnittstelle eines gegebenen Systems in Form des entsprechenden proprietären Transferformats. Unter „Umgangssprache“ sind auch tabellarische Übersichten, technische Dokumentationen, Objektartenkataloge etc. zu verstehen. Man vergleiche dazu Kapitel 4 mit den Realitätsausschnitten für die Systeme der verschiedenen SVT-Layer. Beim

modellbasierten Datentransfer ist der Realitätsausschnitt sowohl für das Start- als auch für das Zielsystem zu beschreiben.

(B) Das **konzeptionelle Datenmodell** (oder **konzeptionelle Schema**) der Datenstruktur hinter dem proprietären Transferformat kann sowohl grafisch als auch textuell formuliert werden. Um den Überblick zu gewinnen, verwenden wir die grafische konzeptionelle Modellierungssprache UML (Unified Modelling Language) [6]. Um die nötige Präzision für die Beschreibung von Klassen, Attributen und Beziehungen zu erhalten, verwenden wir die textuelle CSL INTERLIS 2 [7]. Diese Schweizer Norm (SN612031 [8]) wurde entsprechend den Grundsätzen der weltweiten GI-Normung durch das technische Komitee ISO/TC211 formuliert und ist ein praxisorientiertes und implementiertes Profil der Normenserie ISO19100 [9]. Mit dem Werkzeug UML/INTERLIS Editor [10] kann ein Datenmodell grafisch entworfen und im interaktiven Dialog präzisiert werden. Auf Knopfdruck kann der INTERLIS-Text produziert werden. Das Werkzeug INTERLIS-Compiler [11] erlaubt syntaktische und teils semantische Prüfung des INTERLIS-Textes. Auch das konzeptionelle Datenmodell ist beim modellbasierten Datentransfer sowohl für das Start- als auch für das Zielsystem zu formulieren.

C) Die **Beschreibung des (modellspezifischen) Standard-Transferformat**s folgt automatisch aus dem konzeptionellen Datenmodell gemäss vordefinierten Regeln, d.h. das Standard-Startformat aus dem Startmodell und das Standard-Zielformat aus dem Zielmodell. Wir verwenden das INTERLIS2-XML-Transferformat. Dessen Herleitungsregeln aus dem konzeptionellen Datenmodell sind in Kapitel 3 des INTERLIS2-Referenzhandbuchs [7] festgelegt und entsprechen der Norm ISO19118 Encoding [12]. Wie für alle XML-Formate erfolgt die Formatbeschreibung in XML-Schema. Der INTERLIS Compiler [11] liefert diese Formatbeschreibung (nebst anderen möglichen) automatisch für ein fehlerfreies konzeptionelles Datenmodell. In Kapitel 6 sind die Standardformate zu den Datenmodellen von Kapitel 5 besprochen und verglichen mit den entsprechenden proprietären Formaten von Kapitel 4. Zur Beschreibung der Standardformate in XML-Schema siehe Annex A4.

(D) Für den Transfer vom Startsystem S zum Zielsystem Z liegen die Startdaten vor im proprietären Format von S vor. Des Weiteren werden die Zieldaten im proprietären Format von Z benötigt. Für den Einsatz der Checker- und Transformations-Werkzeuge benötigen wir diese Daten allerdings im Standardformat. Zum Umformatieren von Daten, gegeben im proprietären Format der Start-Schnittstelle, in das entsprechende Standardformat bzw. zum Umformatieren der Daten im Standardformat für die Ziel-Schnittstelle in deren proprietäres Format braucht es einen sog. **1:1-Prozessor**. Dieser transformiert die Input-Textdatei im proprietären Format in die Output-Textdatei im Standardformat bzw. umgekehrt. Beide Dateien entsprechen derselben Datenstruktur. Daher ist ein 1:1 Prozessor nicht komplexe Software. Diese Art von Formatumbau ist oft sogar möglich mit einem Texteditor. Die Frage ist, wie viel Datenprüfung in den1:1-Prozessor einzubauen ist. Kapitel 7 bespricht eine Minimal- und ein Maximal-Variante, welche im Rahmen des Demonstratorprogramms (siehe Kapitel 9) realisiert worden sind.

(E) Die **semantische Transformation** ermöglicht den **Umbau der Datenstruktur** des Startsystems auf die Datenstruktur des Zielsystems format- und system- unabhängig. Auf konzeptioneller Ebene wird der Strukturumbau definiert: Mit Abbildungs- und Umrechnungsfunktionen wird das Start-Datenmodell in das Ziel-Datenmodell umgebaut. Ist der Modellumbau definiert und liegen die Startdaten im Standardformat vor, dessen Beschreibung aus dem Startdatenmodell mit dem Compiler vom Computer

hergeleitet wurde, dann können diese Daten automatisch transformiert werden in das Standardformat des Zielsystems. Dazu stehen heute verschiedene Werkzeuge zur Verfügung, leider noch ohne Programmschnittstellen (InfoGrips Tools [15], INTERLIS Studio [17], UMLT, ILIT [18, 19, 20]). Auch bei Implementierung durch direkte Programmierung wird mit Vorteil der Transformationsentwurf auf Modellebene vorgenommen.

3.4. Zusammenfügen der 5 Elemente zu modellbasierten Diensten

Voraussetzung für den Einsatz modellbasierter Dienste ist, dass von den beteiligten Datenschnittstellen konzeptionelle Datenmodelle und die Daten im entsprechenden Standard-Transferformat vorliegen. Die Elemente (A, B, C, D) aus 3.3 werden also immer benötigt.

- Damit kann bereits nachhaltige Datensicherung erfolgen. Denn Daten im Standardformat (C, D) zusammen mit dem entsprechenden konzeptionellen Datenmodell (A, B) erlauben immer eine zweifelsfreie Interpretation der gespeicherten Daten, solange der entsprechende Datenträger gelesen werden kann.
- Daten im Standardformat (C, D) können mit Hilfe des entsprechenden Datenmodells (A, B) automatisch überprüft werden. Es steht Checker Software zur Verfügung [13] für diese automatische Qualitätssicherung auch von geometrischen und topologischen Eigenschaften.
- Modellbasierter Datentransfer findet statt, wenn sich Sender und Empfänger auf ein Datenmodell für die auszutauschenden Daten (A, B) einigen und die beteiligten Systeme das entsprechende Standardformat (C) schreiben (D beim Sendersystem) bzw. lesen (D beim Empfängersystem).
- Modellbasierter Datentransfer mit Strukturumbau (semantische Transformation): Abbildung 4 zeigt am Beispiel von zwei Systemen der SVT, wie alle 5 Elemente (A) bis (E) des modellbasierten Vorgehens zusammenspielen, um den Umbau einer Startdatenstruktur *S* in eine Zieldatenstruktur *Z* zu erreichen.

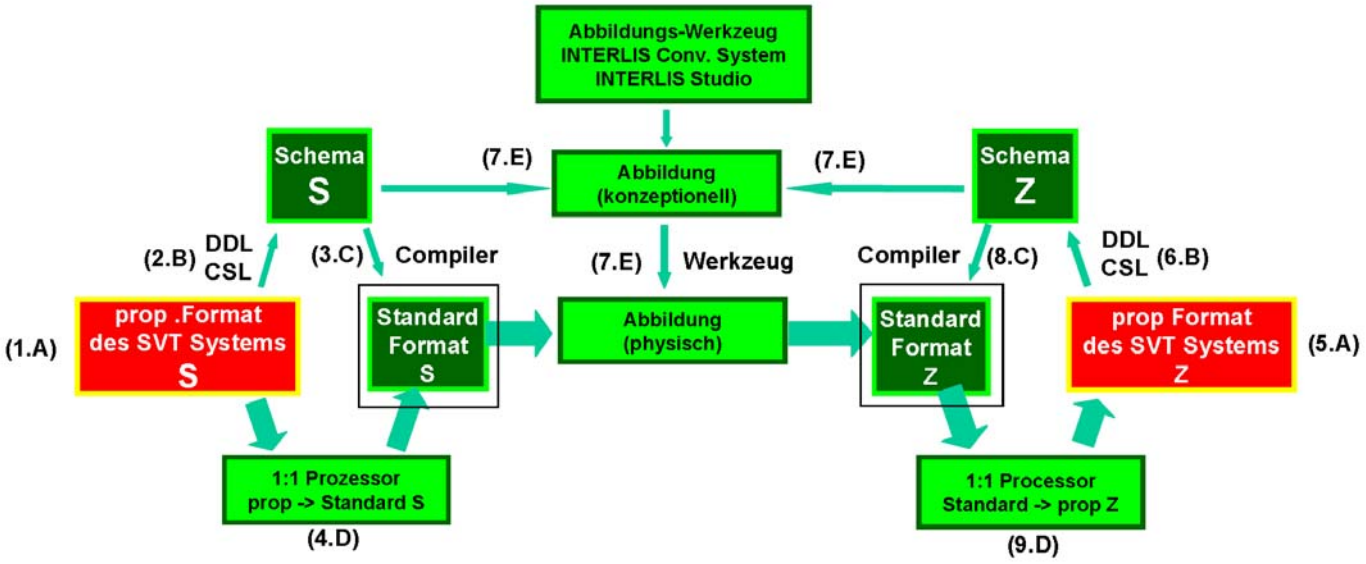


Abbildung 4: Semantische Transformation mit Elementen des modellbasierten Vorgehens. In ( ) Nummer für Reihenfolge, Buchstabe für Typ des Elementes gem. Kap. 3.3

Hier die Details zu den einzelnen Schritten:

1. Das Startsystem *S* liefert Daten im proprietären *S*-Transferformat, dessen ausführliche Beschreibung eine wesentliche, oft die einzige Dokumentation des Realitätsausschnitts ist. Wir setzen voraus, dass die Daten ASCII codiert sind. (Element (A) von 3.3)
2. Auf Grund von (A) ist nun die Datenstruktur von *S* exakt zu beschreiben mit einer DDL/CSL → Datenmodell / Schema *S* gemäss Element (B) von 3.3.
3. Der Compiler liefert die Beschreibung des Standardformats von *S* gemäss Schema *S* (Element (C) von 3.3).
4. Um die Startdaten von *S* im proprietären *S*-Format umzubauen auf den Standardformat verwenden wir einen 1:1-Prozessor (Element (D) von 3.3). Dabei können wir auf eine Umcodierung verzichten, da wir im Schritt 1 voraussetzen, dass die proprietäre Startdatei ASCII-codiert vorliegt.
5. Auch vom Zielsystem *Z* benötigt man einen Realitätsausschnitt in Umgangssprache, mindestens eine ausführliche Beschreibung des proprietären Formats von *Z* (Element (A) von 3.3 fürs Zielsystem). Auch hier setzen wir ASCII Codierung voraus.
6. Davon ausgehend ist auch die Datenstruktur von *Z* exakt zu beschreiben mit einer DDL/CSL. Wir erhalten das Datenmodell / Schema *Z* (wieder Element (B) von 3.3).
7. Nun können wir auf systemneutraler konzeptioneller Ebene die Umstrukturierung der Datenmodelle definieren: Schema *S* → Schema *Z.* Der Dateiumbau Standardformat *S* → Standardformat *Z* erfolgt automatisch ((E) von 3.3).
8. Die *Z*-Daten brauchen wir im proprietären Format des Zielsystems Z , das wir seit Schritt 5 kennen. Der Compiler liefert noch die Beschreibung des Standardformats von *Z* gemäss Schema *Z* (Element (C) von 3.3 für *Z*)
9. Schliesslich braucht es noch einen 1:1 Prozessor zum Umbau Standardformat *Z* → proprietäres Format *Z* (Element (D) von 3.3 für *Z*). Auch hier können wir wegen der Bemerkung in Schritt 5 auf Umcodierung verzichten.

Damit ist der Umbau von *S* nach *Z* beschrieben. Wie ist es mit dem Umbau zurück, von *Z* nach *S* ? Welche der Elemente können wieder verwendet werden, welche sind anzupassen? Die Realitätsausschnitte (A, 1, 5), die konzeptionellen Datenmodelle (B, 2, 6) und die Beschreibungen der Standardformate (C, 3, 8) bleiben gleich. Hingegen braucht es die inversen 1:1-Prozessoren (D, 9, 4) und den inversen Strukturumbau (E, 7).

Vereinfachungen des geschilderten Ablaufs sind denkbar. Etwa, wenn der 1:1-Prozessor ins System integriert wird, wie das bei verschiedenen Geoinformationssystemen heute schon der Fall ist. Dann könnten die Daten direkt im Standardformat ausgegeben oder eingelesen werden. Dazu müssten die Elemente (A, B, C) für die Datenschnittstelle einmal erarbeitet und dann normativ festgelegt werden. Das Datenmodell (C) müsste allerdings für die semantische Transformation zur Verfügung stehen.

- Das Beispiel von Abbildung 4 zeigt die semantische Transformation zwischen zwei Ebenen der Leittechnik. Wie die Referenzarchitektur in Kapitel 2.1 zeigt, gibt es bei den Verkehrsdaten eine Mehrstufigkeit in dem Sinne, dass auf verschiedenen Ebenen der Leittechnik aus Inputdaten über gewisse Funktionen Outputdaten generiert werden, welche an die übergeordnete Ebene der Leittechnik weitergeleitet

werden. D.h. für je zwei aufeinanderfolgende Stufen ist das beschriebene zweistufige Verfahren zu realisieren.

- Im Falle einer flachen statt hierarchischen Kommunikationsstruktur, wie sie am Schluss in 2.4 erwähnt wird und allenfalls Gegenstand eines anderen Forschungsprojektes sein könnte, würde die Anzahl der Schnittstellen reduziert auf die eine der Zählstellen. Alle höheren Ebenen der Leittechnik hätten dieselbe Schnittstelle. Allerdings wäre ein massiver Umbau der heutigen Berechnungsprogramme auf diesen höheren Ebenen der Leittechnik erforderlich.

3.5. Abgrenzung UML zu INTERLIS

Die Grundidee von UML ist, Software intensive Systeme zu visualisieren. Die Grundidee von INTERLIS ist, Daten fachlich (konzeptionell) und maschinell interpretierbar (z.B. zur Herleitung eines Transferformats) zu beschreiben. Aus der unterschiedlichen Zielsetzung ergeben sich Unterschiede, aber auch Gemeinsamkeiten.

Das wichtigste Gemeinsame ist: beides sind Sprachen. Beides sind also Ausdrucksmittel des Menschen. Die Interpretierbarkeit durch Computer steht somit nicht im Vordergrund. Beide bieten dieselben Sprachkonzepte: Klassen, Attribute, Assoziationen, Datentypen.

Der wichtigste Unterschied: UML ist eine grafische Sprache, d.h. man drückt sich in Symbolen aus, man diskutiert Bilder. Bilder haben den Vorteil, dass Strukturen gut dokumentierbar, gut sichtbar sind. Sie springen ins Auge. INTERLIS ist eine textuelle Sprache, d.h. man drückt sich in Form eines Textes aus. Strukturen sind zwar gut dokumentierbar (=exakter Text), aber schlecht sichtbar. Sie springen nicht ins Auge.

Ein weiterer wichtiger Unterschied: Datentypen. UML kennt zwar das Sprachkonzept Datentyp, definiert aber selber keine Datentypen. Modelliert man mit UML, muss man somit zuerst einen Satz Datentypen definieren (in der Regel indem man ein sogenanntes Profil einbindet). Um eine Interoperabilität des Modells zu erreichen, muss man somit auch das Profil mit den Datentypen vorgängig festlegen. INTERLIS hingegen beinhaltet einen vordefinierten Satz an Datentypen.

Sehr zweckmässig ist die Kombination der beiden Werkzeuge. So kann mit dem UML-INTERLIS Editor [10] ein Datenmodell als UML Klassendiagramm entworfen werden, während im Hintergrund automatisch eine präzise INTERLIS Modellstruktur aufgebaut wird. Dabei werden für Attributstypen, Beziehungsrollen, Kardinalitäten etc. Defaultwerte zugeordnet, die über pop-up Menus interaktiv modifiziert werden können. Aus dem UML Editor heraus kann eine Syntax (und teilweise Semantik) Prüfung durch den INTERLIS Compiler [11] gestartet werden sowie die Ausgabe einer INTERLIS Datei oder eines Feature Katalogs gemäss ISO19110.

3.6. Abgrenzung INTERLIS zu ASN1

ASN.1 ist in der Norm ISO 8824-1 beschrieben und ist eine Standard Notation für Datentypen und Werte. Es werden Basistypen und deren Werte definiert sowie Regeln zum Kombinieren von Basistypen und deren Werten zu komplexen Typen und deren Werten. Die Notation der Typen ist unabhängig von der Codierung. Zu ASN.1 passend

gibt es eine Normenserie mit Codierungsregeln von verschiedenem Typ, unter anderen auch „XML Encoding Rules (XER)“ ISO 8825-4.

ASN.1 ist von der ITU, der International Telecommunication Union, entwickelt worden zur Beschreibung von Meldungsstrukturen unabhängig von einem konkreten Transferformat. ASN.1 kann als Beschreibungssprache verglichen werden mit der Formatbeschreibungssprache XML-Schema, ist aber unabhängig von XML. Die in ASN.1 beschriebene Meldungsstruktur kann dann übersetzt werden in XML, aber auch in andere Transferformate. Meldungsbeschreibung mit ASN.1 ist eine exakte Datenbeschreibung und in dem Sinne auch eine Modellierung. Allerdings handelt es sich nicht um objektorientierte konzeptionelle Modellierung sondern um physische Modellierung der Transferstruktur. Datentypen braucht man auch bei der konzeptionellen Modellierung, um Wertebereiche von Attributen zu definieren. Vergleicht man etwa die in INTERLIS 2 vorhandenen Basisdatentypen und die Möglichkeiten, selber Erweiterungen daraus herzuleiten, dann entsprechen diese weitgehend den Basistypen und den Kombinationsmöglichkeiten von ASN.1. Parallelen und Unterschiede wären im Detail abzuklären. Sicher fehlen bei ASN.1 die geometrischen Typen.

Was aber an konzeptionelle Modellierungsmöglichkeiten bei ASN.1 zwar mit der Zusatznorm ISO 8824-2 „Information Object Classes“ nachträglich beigefügt wurde, aber nicht praktisch zum Einsatz kommt, ist das Objektkonzept. Bei konzeptioneller Modellierung werden Objekte beschrieben, welche eine Objektidentität haben, einem Konzept entsprechen, einen Zustand haben (beschrieben durch Attributswerte) und ein Verhalten (beschrieben durch Operationen). Entsprechend werden die Möglichkeiten, Beziehungen zwischen Objekten zu beschreiben und Vererbung nicht genutzt (abgesehen vielleicht vom Subtyp Konzept, das als eine Art Vererbung oder hierarchische Beziehung betrachtet werden kann). Einheiten und Koordinatensysteme werden in ASN.1 hingegen gar nicht benötigt. Es fehlt eine Anbildung von ASN.1 in eine grafische Visualisierung, etwa nach UML bzw. in eine Teilmenge (Profil) davon. Details wären auch hier durch Analyse von ISO 8824-1 und -2 abzuklären.

Aus der Normdokumentation geht nicht hervor, wie weit ASN.1 implementiert ist. Es gibt aber Software zur Prüfung der Syntax und zur automatischen Herleitung der Beschreibung der verschiedenen möglichen Transferformate. Es ist unklar, ob es möglich ist, Daten in einem der entsprechenden Format zu testen gegenüber einer ASN.1 Beschreibung. Sicher gibt es keine Werkzeuge für die Definition von Strukturumbau auf dem Niveau von ASN.1 Beschreibungen.

3.7. Modellbasiertes Vorgehen und (Geo-) Normung

3.7.1. ISO und CEN Vorgaben: Normung im Geobereich mit dem modellbasierten Vorgehen (MDA)

Ausgehend vom Bedürfnis, Geodaten über Landes-, Regions- und System-Grenzen zu transferieren, begann 1992 in Europa die Normung im Geobereich durch das technische Komitee TC287 von CEN. Die Verallgemeinerung existierender nationaler Transferformate (wie EdiGéo in Frankreich oder NTF in England) zu einem europäischen Transferformat erwies sich als unmöglich, da die nationalen Datenstrukturen hinter den Transferformaten zu sehr verschieden waren. Man entschloss sich daher für das modellbasierte Vorgehen, d.h. primär wird die Datenstruktur eines bestimmten Anwen-

dungsbereiches mit Hilfe einer konzeptionellen Datenbeschreibungssprache (Conceptual Schema Language CSL) exakt beschrieben.

Die Arbeiten von CEN/TC287 wurden 1999 abgeschlossen mit europäischen Vornormen (ENV) statt Normen (EN). Die erarbeiteten Normendokumente erwiesen sich als nicht implementierbar, da man mit EXPRESS eine ungeeignete CSL gewählt hatte. Die Mitglieder von CEN/TC287 entschlossen sich, bei der weltweiten Normung durch ISO/TC211 mitzuwirken und die dort erarbeiteten Normen später als EN zu übernehmen.

ISO/TC211 startete 1994 die Normungsarbeit und entschied sich ebenfalls für den MDA. Als CSL wurde die graphische Unified Modelling Language (UML) gewählt. Mit UML verträgliche textuelle CSL sind ebenfalls erlaubt. Transferformat ist XML. Die Normenserie ISO 19100 von ISO/TC211 ist in 3.7.2 kurz vorgestellt.

Ende 2003 ist CEN/TC287 wieder zum Leben erweckt worden und ist daran, die Normenserie ISO 19100 als Euronormen zu übernehmen.

Schweiz:

In der Schweiz wurde 1985 beim Versuch, eine Transferdatei für die Daten der amtlichen Vermessung zu definieren, die CSL INTERLIS entwickelt. INTERLIS (1) ist bereits seit 1991 operativ im Einsatz, machte die Entwicklung der Normenserie ISO 19100 mit und braucht heute ebenfalls UML als graphischen Startentwurf sowie XML als Transferformat. Mit INTERLIS wurden zunächst die Daten der amtlichen Vermessung modelliert (Fixpunkte, Bodenbedeckung, Liegenschaften, administrative Grenzen, etc), weiter Leitungs- und Werkkataster (Wasser, Abwasser, Gas, Elektrizität, Telekommunikation, Fernwärme), neu auch Strassen und Bahnen und Anwendungen mit Daten ohne Raumbezug.

Holland:

In Holland begann man parallel dazu mit der Beschreibung der Anwendung Transportsysteme (Intelligent Transport Systems ITS) aber mit eigener graphischer CSL (NIAM) und eigener textueller CSL (ESN). CEN/TC278 und später ISO/TC204 nehmen die internationale Normung dieses speziellen Transferformats GDF (Geographic Data File) vor. Siehe 3.7.3 für den Zusammenhang zwischen GDF und der Normenserie ISO 19100.

3.7.2. MDA Realisierung durch die Normenserie ISO 19100

ISO 19103 GI Conceptual schema language: Diese Norm legt fest, dass in der ganzen Normenfamilie der objektorientierte Formalismus zur Beschreibung der Datenstrukturen verwendet werden soll und definiert die “Unified Modelling Language” (UML) als konzeptionelle Beschreibungssprache der ISO 19100 Normenserie. UML ist eine graphische formale Sprache, die von der „Open Management Group“ (OMG) normiert wird. Alle textuellen konzeptionellen Beschreibungssprachen kompatibel mit UML sind ebenfalls ISO-konforme konzeptionelle Beschreibungssprachen, z.B. INTERLIS.

ISO 19107 GI – Spatial schema / ISO 19137 GI – Profiles of spatial schema: ISO 19107 gibt eine Übersicht aller zurzeit existierenden räumlichen Datentypen zur Beschreibung von Geodaten. Ein minimales Profil, d.h. eine unbedingt notwendige Teilmenge von ISO 19107 wird von ISO 19137 definiert.

ISO 19109 GI – Rules for application schema: Zeigt, wie die verschiedenen Normen der Serie ISO 19100 kombiniert werden müssen, um Datenbeschreibungen realer Anwendungen zu bekommen.

ISO 19111 GI – Spatial referencing by coordinates: Behandelt Koordinatenreferenzsysteme und Transformationen zwischen ihnen.

ISO 19115 GI – Metadata: Metadaten sind Daten über Daten und wesentlich für die Realisierung systematischer Datensuchdienste. Dies ist eine sehr umfassende Norm und eine der ersten, die in verschiedenen Ländern implementiert wurde.

ISO 19118 GI – Encoding / ISO 19136 GI – GML: In ISO 19118 wird definiert, wie die Struktur der zugehörigen Transferdatei automatisch bestimmt werden kann bei gegebenem konzeptionellem Schema. Die eXtensible Markup Language (XML) ist als Transferformat zu verwenden. Allerdings sind die Regeln nicht exakt genug formuliert, um eine eindeutige Codierung zu gewährleisten. Das Projektteam 36 entwickelt zusammen mit dem Open Geospatial Consortium (OGC) die Geography Markup Language (GML) als Transferformat für ISO 19100 und muss diese Codierungsregeln in ISO 19118 präzisieren.

3.7.3. GDF (Geographic Data File) und ISO 19100

Die Norm ISO 14825 GDF – Geographic Data Files (2004) definiert im Sinne der allgemeinen Normenserie ISO 19100 die Datenstruktur einer konkreten Anwendung, nämlich der Anwendung „Strassendaten“. Dabei kommt das modellbasierte Vorgehen zum Einsatz, allerdings nicht mit automatisiertem Übergang zwischen den einzelnen Phasen (siehe 3.3 und 3.4). Es wird der relationale Formalismus verwendet und nicht der objektorientierte wie in ISO 19100. Als konzeptionelle Beschreibungssprache kommt NIAM zum Einsatz, eine graphische formale Sprache, eine spezielle Form des Entitity Relationship Diagramms. In Vorbereitung ist die Ablösung von NIAM durch UML.

Das konzeptionelle Schema der Strassendaten findet man in den Kapiteln 5 bis 10 in Form von NIAM-Diagrammen und von beschreibendem Text.

- Zunächst wird in einem Übersichtsdiagramm (genannt CONCEPTUAL DATA MODEL) gezeigt, welche grundsätzlichen Möglichkeiten für ein Datenobjekt (genannt Feature) bestehen, Attribute zu haben, Beziehungen einzugehen, geometrische oder topologische Eigenschaften zu haben, Layern zuzugehören etc. (Kapitel 5).
- Der FEATURE CATALOGUE (Kapitel 6) beschreibt die einzelnen mit GDF transferierbaren Datenobjekte.
- Der ATTRIBUTE CATALOGUE (Kapitel 7) beschreibt die verschiedenen Eigenschaften der Datenobjekte, genannt Attribute, gruppiert nach Objektthemen.
- Der RELATIONSHIP CATALOGUE (Kapitel 8) beschreibt nicht-geometrische und nicht-topologische Beziehungen zwischen Objekten.
- Die FEATURE REPRESENTATION RULES (Kapitel 9) beschreiben die Darstellungsattribute für die verschiedenen Objekte.
- Der METADATA CATALOGUE (Kapitel 10) schliesslich enthält die Spezifikation von Metadaten, d.h. von Daten, welche die GDF-Dateien als Ganzes be-

schreiben (z.B. die Adresse des Datenproduzenten, eine Beschreibung des Dateiinhalts in Umgangssprache, die geografische Begrenzung des betroffenen Bereiches etc.).

Ebenfalls zur konzeptionellen, d.h. system- und format-unabhängigen Beschreibung der Strassendaten gehören die sog. LOGICAL DATA STRUCTURES (Kapitel 11). Hier wird eine textuelle Datenbeschreibungssprache verwendet, nämlich ESN.

Das physische Schema des Transferformats wird durch die MEDIA RECORD SPECIFICATIONS definiert. Die verschiedenen Felder der sequentiellen Recordstruktur werden beschrieben sowie spezifische Einschränkungen, die Transferdateien betreffend. Leider kann diese Formatbeschreibung nicht automatisch aus dem konzeptionellen Schema in NIAM oder ESN hergeleitet werden. Die Nachführung von GDF Dateien ist ebenfalls definiert, damit Änderungen in der Realität unmittelbar durch entsprechende Änderungen in den Daten beschrieben werden.

In den Anhängen werden Details zu Aufzählattributen beschrieben in Form von Codelisten

Im Bereich Verkehrstelematik bildet diese Vornorm ein wesentliches Regelwerk, sie befriedigt die Bedürfnisse des Strassentransports und stellt die nötigen Werkzeuge zur Verfügung für die koordinatenbasierte Beschreibung der Geodaten und deren Austausch. Die rasche Entwicklung und Verbreitung der Fahrzeug-Navigationssysteme, welche globale Positionierungssysteme und globale Referenzsysteme brauchen, wurden wesentlich erleichtert durch diese Vornorm.

Das modellbasierte Vorgehen ist auch hier zweckmässig, allerdings sollten Werkzeuge eingesetzt werden, die erlauben, aus dem einmal definierten konzeptionellen Schema logische und physische Schemata für Transferformate, Datenbankkonfiguration, Serviceprotokolle etc. automatisch herzuleiten, wie das bei UML / INTERLIS 2 der Fall ist.

3.7.4. INTERLIS und ISO 19100

INTERLIS ist Modellierungssprache und Datentransfermethode. INTERLIS ist system- und anwendungsneutral und wurde ursprünglich im Auftrag der Schweizerischen Vermessungsdirektion entwickelt, um den problemlosen Datenaustausch zwischen GIS zu ermöglichen. Ausgetauschte Daten sind jedoch nur dann brauchbar, wenn von ihnen eine genaue und einheitliche Beschreibung vorliegt.

Kernidee von INTERLIS ist daher die exakte Beschreibung der Datenstrukturen in Form von Datenmodellen, indem die entsprechenden Objekte, ihre Attribute mit den Wertebereichen sowie die Beziehungen zwischen den Objekten genau festgelegt werden. Die INTERLIS-Norm legt auch fest, wie aus dem Datenmodell die Beschreibung des entsprechenden Transferformats automatisch hergeleitet werden kann. Entsprechend den Forderungen von ISO/TC211 wurde INTERLIS objektorientiert weiterentwickelt, hat UML integriert und bietet auch XML-basierte Transferformate an. Neben dem ursprünglichen INTERLIS 1 Transfer Format (ITF) steht der von ISO bevorzugte XML-Dialekt GML zur Verfügung sowie das INTERLIS 2 spezifische XML Transfer Format (XTF), das im Rahmen dieses Projektes eingesetzt wird und das auch inkrementelle Nachlieferung, polymorphes Lesen und Grafiktransfer unterstützt. Damit entspricht INTERLIS der Normenserie ISO 19100 (siehe 3.7.2) und bietet darüber hinaus noch weitere Möglichkeiten.

Die entsprechenden Werkzeuge sind implementiert und in täglichem Gebrauch: „Compiler“ zur Prüfung von Datenmodellen und zur Herleitung des Transferformats, „Checker“ zur Prüfung der Daten gegenüber ihrem Datenmodell, „Conversion System“ zur semantischen Transformation, „GeoShop“ zur grafischen Präsentation im Internet und als Datendrehscheibe etc. INTERLIS ist damit eine funktionstüchtige Implementierung des modellbasierten Vorgehens.

3.7.5. Beurteilung der Lösungen

Entsprechend den Anforderungen der europäischen und der weltweiten Normung sollte auch zur Bearbeitung und zum Austausch der Daten in der Strassenverkehrstelematik und in der Strassentechnik allgemein das modellbasierte Vorgehen zum Einsatz kommen. Als Möglichkeiten stehen zur Verfügung:

(I) UML + GML gemäss ISO 19100

(II) NIAM + ESD + GDF gemäss ISO 14825 GDF

(III) UML + INTERLIS + ITF/XTF/GML gemäss ISO 19100 und SN 612031

Beurteilung: Im Falle (I) ist die nötige Präzision auf Attributsebene nicht gegeben und der direkte Übergang von UML nach GML ist nicht definitiv geregelt und auch nicht implementiert. Im Falle (II) ist kein automatischer Übergang möglich von NIAM (grafisches Datenmodell) weder nach ESN (textuelles Datenmodell) noch nach GDF (Transferformat). Im Falle (III) ist ein Normenwerk gegeben, das ISO 19100 entspricht, die zweckmässigen Automatisierungen aufweist und mit den nötigen Werkzeugen implementiert und getestet worden ist. Wir haben deshalb Variante (III) gewählt als Basis für die Bearbeitung von Systemen der Strassenverkehrstelematik und deren Datenaustausch. Der „Demonstrator“ (siehe Kapitel 9) zeigt praktisch die Eignung von Variante (III) für die systemübergreifende Verfügbarkeit von Daten der SVT in sehr heterogenem Systemumfeld, ohne dass Systemwechsel notwendig sind.

3.7.6. Vergleich mit den VSS Normen 640948, 640948-1/2 zu Strassenverkehrsdaten.

Die Norm 640948 mit dem Titel „Katalog für Verkehrsdaten – Grundlagen“ führt die in der Normenserie verwendeten Begriffe ein – Verkehrswerte, Klassifikation, Klassen, Ganglinien, Zeitreihen u.a. – und zeigt die gegenseitigen Beziehungen in einem vollständigen Datenmodell (Abbildung 5). Diese Grundnorm enthält ferner die Definition der Daten für den Messort. Die Norm 640948-1 mit dem Titel „Katalog für Verkehrsdaten – Stammdaten“ befasst sich mit den Möglichkeiten, welche Verkehrswerte wie in Klassen geordnet zu Ganglinien und Zeitreihen zusammengefasst werden können. Daten zur Charakterisierung von Ganglinienarten, Klassifikationen und Klassen werden definiert. Schliesslich umfasst die Norm 640948-2 mit dem Titel „Katalog für Verkehrsdaten – Verkehrswerte in Zeitreihen“ die Definition der Daten zur Charakterisierung von Verkehrswert, Ganglinie und Zeitreihe sowie zu den zusätzlich benötigten Objekten Episode, Periode und absolutes Zeitintervall. Die Definition der Daten erfolgt in allen 3 Normen durch sog. Datenkataloge in Tabellenform mit den Kolonnen Attributsname (+ Kennzeichnung von Schlüsseln und Fremdschlüsseln), Erklärung, Format, Wertebereich. Zu jedem Datenkatalog gibt es ein ausführliches Datenbeispiel ebenfalls in Tabellenform mit den Kolonnen Attribut(nummer), Beispielwert, Erläuterung.

In diesem Normenpaket steht die Festlegung der Verkehrsdaten und der zu ihrer Aggregierung nötigen Konstrukte wie z.B. Ganglinien oder Perioden im Vordergrund. Dabei ist der Gesichtspunkt klar derjenige der Redundanzfreiheit aus Sicht Datei bzw. Datenbank. Das Forschungsprojekt dagegen ist ausgegangen von gegebenen Verkehrsdaten in Form von Einzelfahrzeugmessungen und Intervallsummen. Während die Einzelfahrzeugmessungen nicht einer bestimmten Ganglinie zugeordnet sind, kann man von den Intervallsummen sagen, dass sie im Prinzip Klassenwerte sind einer Ganglinie mit Basis Stunde, Definitionsbereich Tag, Verkehrsgrösse Anzahl Fahrzeuge pro Stunde für die Klassifizierungen Fahrzeuglänge und Fahrzeugtyp. Schwergewicht des Forschungsprojektes ist die präzise und computerlesbare Beschreibung der vorgegebenen Datenstrukturen durch konzeptionelle Datenmodelle, aus denen sich das Austauschformat automatisch ergibt, sowie die Möglichkeit, existierende Prüf- und Umbau-Werkzeuge auf Modell und Daten anzuwenden. Dabei ergeben sich verschiedene interessante Beziehungen zum Normenpaket, z.B. zu folgenden Themen:

- Sicht Anwendungen:

  Übereinstimmung der im Projekt übernommenen Verkehrswerte mit den Vorgaben des Normenpakets.

- Methodische Sicht:

  Möglichkeiten und Bedeutung objektorientierte Beschreibung der in den Datenkatalogen des Normenpakets beschriebenen Datenstrukturen durch konzeptionelle Datenmodelle mit der Möglichkeit, Standardformate zu generieren.

- Sicht Transfer und Auswertungen:

  Minimalsatz von Zusatzdaten zu Einzelfahrzeugmessungen, so dass meldungsbasierter Transfer möglich ist und darauf basierend individuelle statistische Auswertungen bei jedem Empfänger gemäss seinen lokalen oder übergeordneten Bedürfnissen (Ganglinien, Zeitreihen etc.)

3.7.7. Vergleich mit der VSS Norme 671941 zu Koordinatentransformationen

Diese Norm behandelt die Grundsätze der räumlichen und der topologischen Referenzierung von Daten des Verkehrs- und Transportmanagements und beschreibt die Transformationen zwischen den entsprechenden Referenzsystemen. Das Forschungsprojekt hat als Ziel Z4 formuliert, die Möglichkeiten und Grenzen des modellbasierten Vorgehens aufzuzeigen zum Schaffen der datentechnischen Voraussetzungen für automatische Koordinatentransformationen zwischen verschiedenen in der SVT gebräuchlichen Referenzsystemen. Das in Abschnitt 9.2 der Norm 671941 erwähnte Beispiel der Verkehrszähler entspricht genau der im Projekt gegebenen Situation. Auch die Verkehrszähler des Projektes sind im räumlichen Basisbezugssystem referenziert und deren Verkehrszählungen sollten auf nationalen Übersichten dargestellt werden. Für diese kartografische Darstellung ist die Transformation „linear zu planar“ notwendig. Leider musste die Realisierung von Ziel Z4 im Rahmen des Projektes aus Kapazitätsgründen zurückgestellt werden.

Das Vorgehen ist aber soweit klar. Die datentechnischen Voraussetzungen für die konzeptionelle Modellierung von linearen, planaren und räumlichen Koordinatensystemen mit geraden und elliptischen Achsen und deren Umsetzung in Standard Transferformate sind gegeben. Nicht so für die kurvilinearen Koordinatensysteme der Strassen. Bei diesen entspricht die erste Achse dem Verlauf der Strassenachse, die zweite verläuft in jedem Achspunkt in der Strassenebene senkrecht zur Achse und die dritte steht im Achspunkt senkrecht zur Strassenebene. Zunächst ist ein entsprechendes Sprachele-

ment der Modellierungssprache zu definieren. Dann ist die Koordinatentransformation vom RBBS über ein kurvilineares ins planare (oder räumliche) Landeskoordinatensystem in der semantischen Transformation zu implementieren und mit den gegebenen Testdaten zu prüfen. Dabei ergäbe sich die Gelegenheit, festzustellen, ob die im Abschnitt 9.2 der Norm 671941 erwähnten Transformationsparameter genügen.

3.7.8. Technische Lieferbedingungen für Streckenstationen (TLS)

Die Technischen Lieferbedingungen für Streckenstationen (TLS) sind ein Standard für den Aufbau von Verkehrsbeeinflussungsanlagen an Bundesfernstraßen. Zusammen mit dem Merkblatt für die Ausstattung von Verkehrsrechnerzentralen und Unterzentralen (MARZ) beschreiben sie die Architektur des Verkehrsbeeinflussungssystems für das Bundesfernstraßennetz. Die TLS werden im Auftrag des Deutschen Bundesministeriums für Verkehr, Bau und Stadtentwicklung in Zusammenarbeit mit den Straßenbauverwaltungen der Bundesländer und der Industrie fortgeschrieben.

Mit den TLS wird das Ziel verfolgt, Funktionen und Schnittstellen einheitlich festzulegen (Standardisierung), so dass Geräte unterschiedlicher Hersteller, die vom Leistungsumfang her weitgehend identisch sind, auch im Wettbewerb miteinander vergleichbar sind. Dagegen wird im vorliegenden Forschungsprojekt der Datenaustausch in heterogenen Systemumgebungen beschrieben.

Auch im Rahmen der TLS Standardisierung von Funktionen und Schnittstellen liesse sich das modellbasierte Vorgehen nutzbringend anwenden. Einerseits könnte durch die objektorientierte Modellierung Präzision und Computer – Bearbeitbarkeit auf konzeptioneller Ebene erreicht werden. Andererseits wäre automatisch ein Standard Transferformat vom browsertauglichen Typ XML gegeben und u.a. die Möglichkeit automatischer Qualitätskontrolle der transferierten Daten.

3.7.9. DATEX II

Der DATEX Standard wurde für den Informationsaustausch zwischen Verkehrs-Managementanlagen entwickelt und wurde später in DATEX II auch um die Integration von Aktoren der Verkehrs- und Reiseinformation erweitert. Die Mechanismen sind auf der Datenformatebene angesiedelt.

Auch in diesem Zusammenhang wäre es sehr interessant, abzuklären, ob durch das modellbasierte Vorgehen, mit dem Primärfokus auf der Struktur der Daten hinter den Formaten, für die DATEX II Festlegungen auf der Datenformatebene sich nebst XML Standard Format und automatischer Qualitätskontrolle nicht auch eine gewisse Flexibilisierung und bessere Anpassungsfähigkeit an Weiterentwicklungen ergeben würde.

4. Realitätsausschnitt

Das modellbasierte Vorgehen (Model Driven Approach MDA) beginnt immer mit der Beschreibung des Realitätsausschnitts bzw. der Realitätsausschnitte der beteiligten Systeme. Wie die Übersichten von Kapitel 3.3 und 3.4 zeigen, ist der Realitätsausschnitt Elements (A) des MDA.

4.1. Geographische Ausdehnung

Als Realitätsausschnitt wird das lokale System zur Erfassung des Verkehrsaufkommens (VDE[1]) auf dem Streckenabschnitt Anschluss Urdorf Süd bis Anschluss Birmensdorf der A4 betrachtet.

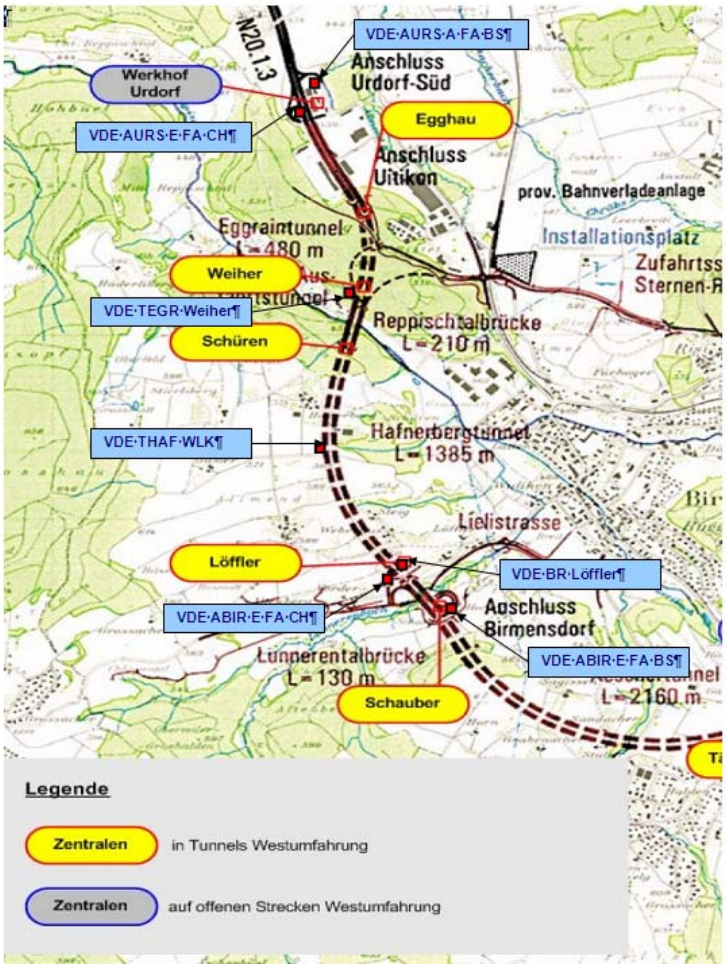


Abbildung 5: Übersicht geographische Ausdehnung Realitätsausschnitt (Quelle: ABB Schweiz AG)

[1] Im Beispielgebiet wird das System als Verkehrsdatenerfassungssystem (VDE) bezeichnet, es wird jedoch nur das Verkehrsaufkommen erfasst.

National-Strasse	Objekt	Richtung	Spuren	Betriebs-Kilometer	Streckenstation
A4	Anschluss Urdorf Süd	Einfahrt nach Basel	Spur 1		SST_URDORF-SUED_A
		Ausfahrt von Chur	Spur 1	0.008	
		Einfahrt nach Chur	Spur 1	0.150	SST_URDORF-SUED_B
		Ausfahrt von Basel	Spur 1	0.010	
	Anschluss Uitikon (Ristettunnel)	Einfahrtstunnel	Spur 1	0.738	SST_EGGRAIN
		Ausfahrtstunnel	Spur 1	0.123	
	Eggraintunnel	Basel-Chur	Spur 1-3	95.484	
		Chur-Basel	Spur 1-3	95.526	
	Hafnerbergtunnel	Basel-Chur	Spur 1-3	96.547	SST_HAFNERBERG_A
		Chur-Basel	Spur 1-3	96.552	
	Anschluss Birmens-dorf	Ausfahrt von Basel	Spur 1	0.134	SST_BIRMENSDORF_A
		Einfahrt nach Basel	Spur 1	0.190	SST_BIRMENSDORF_B

Tabelle 1: Standorte Messstellen im gewählten Realitätsausschnitt

4.2. Systemarchitektur

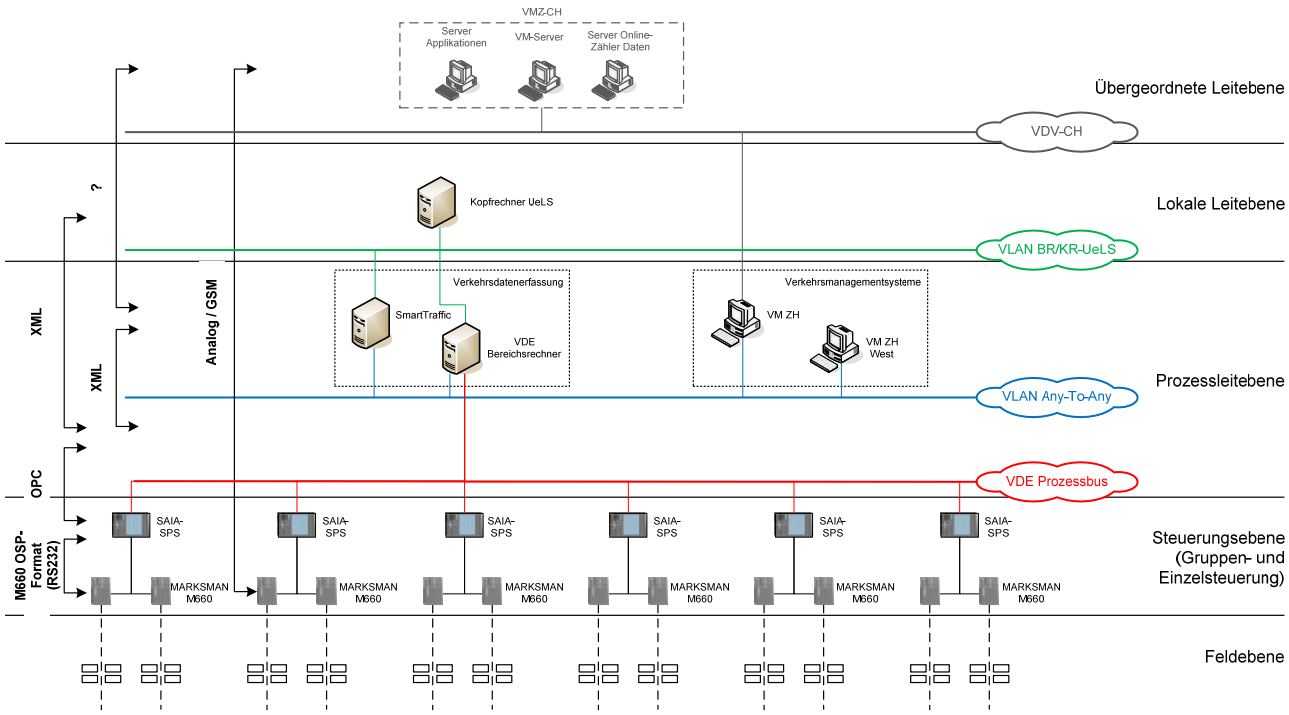


Abbildung 6: Systemarchitektur VDE ZH West

Abbildung 6 zeigt die Systemarchitektur eines Verkehrserfassungssystems am Beispiel des VDE ZH West auf. Die Fahrzeuge werden über Doppelschlaufen detektiert (Feldebene, s. 4.2.1) und im Verkehrszähler von Golden River MARKSMAN 660 protokolliert (Einzelsteuerebene, 4.2.2). Mehrere Verkehrszähler werden über eine SAIA SPS zusammengefasst und die entsprechenden Daten aggregiert (Gruppensteuerebene, s. 4.2.3). Eine Ebene darüber werden alle Verkehrsdaten für das VDE ZH West im VDE Bereichsrechner zusammengetragen, um daraus die notwendigen Operationen abzuleiten (Prozessleitebene, s. 4.2.4). Die Beschreibung der einzelnen Schnittstellen erfolgt in Kapitel 4.4.

4.2.1. Feldebene

Pro Messquerschnitt sind bis zu acht Doppelschlaufen verlegt. Diese übergeben die Messdaten an Verkehrszähler (Golden River MARKSMAN 660 Loop Data Loggers).

4.2.2. Einzelsteuerebene

Auf der Einzelsteuerebene sind 12 Verkehrszähler Golden River MARKSMAN 660 Loop Data Loggers an sieben Messquerschnitten/Zählstellen angeordnet.

4.2.3. Gruppensteuerebene

Auf der Gruppensteuerebene sind Streckenstationen vom Typ SAIA-SPS PCD3 der Fa. ABB angeordnet.

4.2.4. Prozessleitebene

Auf der Prozessleitebene ist der VDE Bereichsrechner (INOVIS IPC) der Fa. ABB im Hafnerbergtunnel/Tunnelzentrale Löffler untergebracht.

4.3. Koordinatensysteme

Im Realitätsausschnitt des Kantons ZH ist der AKS gemäss Konzept AKS-ZH [1] spezifiziert. Als Element für ein mögliches Koordinatensystem werden im Kanton ZH die Abschnittsbezeichnung und sowie die Kilometrierung der Nationalstrassen verwendet. Diese Kilometrierung wird auch im Rahmen des räumlichen Basisbezugssystem der Nationalstrassen RBBS des ASTRA verwendet.

Anmerkung der Redaktion: Inwieweit der AKS-ZH als Koordinatensystem im vorliegenden Projekt genutzt werden kann, ist im Rahmen der auf das Forschungsprojekt folgenden Arbeiten zu bestimmen.

4.4. Datenschnittstellen

4.4.1. Verkehrszähler mit Intervallsummen (Einzelsteuerebene) → Streckenstation (Gruppensteuerebene)

4.4.1.1. Prinzip

Pro Fahrzeug können von den Verkehrszählern die folgenden Parameter erfasst werden:

- Geschwindigkeit [km/h]

- Fahrzeuglänge [cm]
- Fahrzeugabstand Front - Front [s]
- Fahrzeugabstand Front - Heck (Lücke) [s]

4.4.1.2. Transferformat

Das Transferformat kann 5.1.1 entnommen werden.

Im praktischen Einsatz werden je nach Einsatzzweck teils aggregierte Daten (z.B. über 30 Sekunden, s. Anhang 5.1.1) teils Einzelfahrzeugdaten (s. Abschnitte 4.4.4 und 5.1.4) von den Verkehrszählern an übergeordnete Systemkomponenten oder Drittsysteme versendet.

4.4.1.3. Transferprozess

Der Datenaustausch zwischen dem Verkehrszähler Marksman 660 und der Streckenstation geschieht über eine serielle RS232-Schnittstelle.

4.4.1.4. Varianten

Weitere Anbieter Verkehrszähler sind:

- Siemens Schleifendetektor LD4
- Weiss-Electronic MC2014 T2 Schleifendetektoren
- ECTN Laserscanner

4.4.2. Streckenstation (Gruppensteuerebene) → VDE Bereichsrechner (Prozessleitebene)

4.4.2.1. Prinzip

Die von den Verkehrszählern erfassten Verkehrsdaten werden in der Streckenstation auf der Gruppensteuerebene zu gewünschten Output-Werten für den VDE Bereichsrechner aggregiert.

4.4.2.2. Transferformat

Das Transferformat kann Anhang A.1.2 entnommen werden.

4.4.2.3. Transferprozess

Die Kommunikation zwischen Streckenstation und VDE Bereichsrechner erfolgt mittels OPC DA über Ethernet TCP/IP SAIA S-BUS. Die Übertragung der Daten erfolgt blockweise, indem alle Werte einer Spur in einer OPC-Gruppe zusammengefasst und somit gleichzeitig übertragen werden. Die Steuerung der Kommunikation erfolgt durch Handshake-Bits, die mittels einer weiteren OPC-Gruppe bidirektional übertragen werden.

4.4.2.4. Varianten

Im Rahmen des Forschungsprojekts werden hierzu keine weiteren Varianten betrachtet.

4.4.3. VDE Bereichsrechner (Prozessleitebene) → Verkehrsmanagement Subsystem ZH West (Prozessleitebene)

4.4.3.1. Prinzip

Auf dem Bereichsrechner werden die erhaltenen Werte aller Streckenstationen bzw. der entsprechenden Strassenabschnitte zusammengefasst, einer Glättung unterzogen (parametrisierbar) und in gewünschter Form dem Verkehrsmanagement Subsystem ZH West zyklisch zur Verfügung gestellt.

4.4.3.2. Transferformat

Die Kommunikation zwischen dem VDE Bereichsrechner und dem übergeordneten Verkehrsmanagement-Subsystem ZH West erfolgt in Form von XML Files. Die Beschreibung des XML-Transferformats mit XML-Schema ist Auszugsweise in Anhang A.1.3 zu finden

Pro Verkehrszähler-Standort stehen die folgenden Messwerte zur Verfügung:

- Fahrzeugfrequenz [Fz/h]
- Fahrzeuggeschwindigkeit [km/h]
- Fahrzeugdichte [Fz/km]
- Fahrtrichtung [1,2,9][2]

Die Berechnung der Messwerte bezogen auf den Querschnitt erfolgt wie folgt:

- Fahrzeuggeschwindigkeit = arithmetisches Mittel aller mit der Frequenz pro Spur gewichteten Geschwindigkeiten pro Spur
- Frequenz = Summer aller Frequenzen pro Spur
- Fahrzeugdichte = Summe aller Dichten pro Spur

Die Messdaten werden vom VDE Bereichsrechner dem Subsystem in einem Takt von 30 Sekunden abgegeben. Mit diesen Informationen und definierten Staukriterien (z.B. Stau = Geschwindigkeit unter x km/h) kann im jeweiligen Messquerschnitt Stau ermittelt werden. Eine Plausibilisierung durch Berücksichtigung mehrerer Messquerschnitte ist nur über mehrere 30 Sekunden-Intervalle möglich (60 oder 90 Sekunden).

[2] 0 = kein Verkehr / 1 = Verkehr verläuft in Normalrichtung / 2 = Verkehr verläuft in Gegenfahrtrichtung / 9 = der Verkehr verläuft bezogen auf den Messquerschnitt in beiden Richtungen, in uneinheitlicher Richtung oder in unbekannter Richtung.

Datenpunktausgänge an VDE	Datenpunkteingänge von VDE
Keine	Messdaten pro Fahrstreifen: Frequenz Geschwindigkeit Ausfälle Prozessausfall Systemzustand Testdatenpunkte

Tabelle 2: Datenpunktausgänge an VDE

4.4.3.3. Transferprozess

Die Kommunikation erfolgt über eine Datenpunktkommunikation mittels Telegrammen auf Basis XML. Die Telegrammzykluszeit ist ein Vielfaches von 30 Sekunden.

4.4.3.4. Varianten

Im Rahmen des Forschungsprojekts werden hierzu keine weiteren Varianten betrachtet.

4.4.4. Verkehrszähler Einzelfahrzeuge MM660 (Feldebene, ASTRA-Zähler) → VM-Rechner (Verkehrmanagement, CH-System)

4.4.4.1. Prinzip

Ausgewählte Verkehrszähler werden durch das ASTRA zu Statistikzwecken über einen analogen Kanal erschlossen unter der Verwendung der zweiten seriellen Schnittstelle am Verkehrszähler.

Zu Statistikzwecken werden die Anzahl Fahrzeuge stundenweise kumuliert für die folgenden Kategorien erhoben:

- Längenklassen
  - LN1 = Länge < 270 cm
  - LN2 = Länge < 700 cm
  - LN3 = Länge < 1300 cm
  - LN4 = Länge < 2500 cm
- Swiss10
  - CS1 = Bus
  - CS2 = Motorräder
  - CS3 = Personenwagen
  - CS4 = Personenwagen + Anhänger
  - CS5 = Lieferwagen
  - CS6 = Lieferwagen + Anhänger
  - CS7 = Lieferwagen + Auflieger
  - CS8 = Lastwagen
  - CS9 = Lastenzüge
  - CS10 = Sattelzüge

4.4.4.2. Transferformat

Das Transferformat ist in 5.1.4 beschrieben.

4.4.4.3. Transferprozess

Die Datenübertragung geschieht in Form eines Tagesfiles, welches via Fixnet oder GSM von den entsprechenden Zählstellen heruntergeladen wird.

Zu VM-CH siehe di Bemerkung in 4.4.5.3

4.4.4.4. Varianten siehe Abschnitt 4.4.5

4.4.5. Verkehrzähler Einzelfahrzeuge ECTN (Feldebene, ASTRA Zähler) → VM-Rechner (Verkehrmanagement, CH-System)

4.4.5.1. Prinzip

Mittels der speziellen Überkopf Anordnung der Laserscanner des TIC-Systems (Traffic Information Collector) von ECTN werden 3D-Informationen generiert. An exponierten Strassenquerschnitten können Daten über Fahrzeuge für verkehrsleitende Massnahmen und Sicherheitsüberwachung eingesetzt werden.

Folgende Informationen können mittels eines Systems mit Laserscannern erfasst werden:

- Höhen-, Breiten- und Längenerfassung
- Geschwindigkeits- und Abstandsmessung
- Mehrere Spuren mit einem Scanner
- Hohe Systemflexibilität
- Mobiler Einsatz möglich
- Klassifizierung nach TLS 5+1, TLS 8+1 und Swiss10

4.4.5.2. Transferformat

Die erfassten Daten werden in eine SQL-Datenbank eingetragen und als CSV-Datei exportiert. Siehe 5.1.5

4.4.5.3. Transferprozess

Die Daten werden in einer Recheneinheit vor Ort gesammelt und per Ethernet Schnittstelle, Telefonleitung oder per Memory Card zum Basisrechner transferiert. Die Datenbank kann vor Ort (Export, Transfer der CSV-Dateien) oder in der Leitstelle (Transfer der Messdaten vom Scanner) sein.

Es werden momentan ausschliesslich Einzelfahrzeugdaten durch diese Art Verkehrszähler erfasst. Ein Algorithmus für kumulierte Daten wird erst bei Bedarf integriert werden.

Bemerkung: Aufgrund der beiden verschiedenen vorhandenen Messsysteme für Einzelfahrzeugdaten (hier in 4.4.5 der Laserscanner von ECTN und in 4.4.4 der Schleifenzähler von Marksmann 660) wird ein Standardvorschlag für die Schnittstelle VM-CH vorgeschlagen und damit gearbeitet.

4.4.5.4. *Varianten* Siehe Kapitel 4.4.4

4.4.6. Verkehrsmanagement Subsystem ZH West (Prozessleitebene) → Verkehrsmanagementzentrale Schweiz (übergeordnete Leitebene)

Da zum Zeitpunkt der Forschungsarbeit noch keine abschliessende Schnittstellendefinition zwischen den VM-Systemen Schweiz und Kantone vorliegt, muss die Betrachtung davon zurückgestellt werden.

4.4.7. Verkehrsmanagement Subsystem ZH West (Prozessleitebene) → Verkehrsleitsystem (Prozessleitebene)

4.4.7.1. Prinzip

Aufgrund der vom Verkehrsmanagement Subsystem ZH West ermittelten Stauräume werden an die entsprechenden Verkehrsleitsysteme Szenarien versendet. Diese Szenarien werden durch das Verkehrsleitsystem interpretiert und in zu schaltende Betriebszustände umgewandelt. Hierbei werden durch das Verkehrsleitsystem reine Stau-Betriebszustände geschaltet.

4.4.7.2. Transferformat

Die Szenarien werden in XML beschrieben und über binäre Datenpunkte vom VM an die entsprechenden Systeme abgegeben (Szenarien kommend und gehend). Im vorliegenden Realitätsausschnitt sendet das VM ZH West der Verkehrslenkung Honeret die zu schaltenden Szenarien zu. Bei den durch die VDE-Daten abgeleiteten Szenarien bzw. Betriebszuständen handelt es sich um reine Stau-Betriebszustände.

4.4.7.3. Transferprozess

Die Kommunikation erfolgt über eine Datenpunktkommunikation mittels Telegrammen als XML-Dateien.

4.4.7.4. Varianten

Im Rahmen des Forschungsprojekts werden hierzu keine weiteren Varianten betrachtet.

5. Konzeptionelle Modellierung

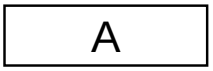
Damit kommen wir zum Element (B) des modellbasierten Vorgehens (Model Driven Approach MDA). Basierend auf der Beschreibung des Realitätsausschnitts geht es jetzt um die präzise Beschreibung der Datenstruktur mit einem konzeptionellen Datenmodell und zwar einerseits grafisch mittels eines UML-Klassendiagramms (5.1) und andererseits textuell mit INTERLIS 2 (5.2). Zur Übersicht über den MDA vergleiche Kapitel 0 und 3.4. Zur Frage, warum es neben einem Datenmodell in UML auch noch eines in INTERLIS braucht, vergleiche Kapitel 3.5.

5.1. Datenstrukturen: Grafische Übersicht mit dem UML-Klassendiagramm

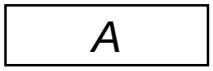
Damit man mit den UML Diagrammen besser zurechtkommt, sind die Elemente von UML in Abbildung 7 kurz zusammengestellt.

Die genaue Beschreibung der Datenstruktur einer Schnittstelle durch ein Datenmodell kann sich unter anderem nach der Antwort auf die Frage richten, ob der ins Auge gefasste Datentransfer filebasiert oder meldungsbasiert erfolgen soll. Das ist besonders dann zweckmässig, wenn modellbasierter Datentransfer im Vordergrund steht und das Datenmodell primär verwendet wird, um daraus einen Standard-Transferformat herzuleiten. Das Datenbeispiel in der folgenden Abbildung 8 deutet auf einen Filetransfer hin. Hingegen wurde bei der Implementierung des Demonstrators festgestellt, dass es zweckmässiger ist, Meldungen dann zu übermitteln, wenn bei einem der beteiligten System ein Ereignis auftritt, z.B. die Registrierung eines Einzelfahrzeuges oder das Ende eines Messintervalls. Wenn nötig, könnte man die Zusammenstellung von Files aus Meldungen beim Empfängersystem vornehmen, evtl. durch dessen Input 1:1 Prozessor. In Kapitel 5.1.1 wird gezeigt, wie sich die Modellierung der Datenstruktur einer Schnittstelle danach richten kann, ob file- oder meldungsbasiert transferiert werden soll.

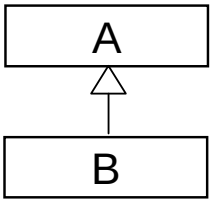
Eine zweite Frage stellt sich, wenn man sich für den Transfer von Ereignismeldungen entscheidet. Nämlich, was an Headerdaten einem Meldungsobjekt beigefügt werden muss, damit dieses vollständig ist. Beim Beispiel für die Ausarbeitung einer gemeinsamen Basisstruktur gemäss Kapitel 5.1.6 aus den beiden verschiedenen Einzelfahrzeugmessungen (siehe Kapitel 5.1.4 und 5.1.5) wird diese Frage wieder aufgegriffen.



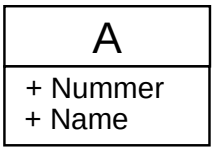
Konkrete Klasse A



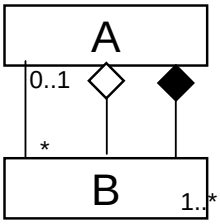
Abstrakte Klasse A
(Name *kursiv geschrieben)*



Klasse B erbt Klasse A, d.h. Objekte von Klasse B haben alle Eigenschaften der Objekte von Klasse A und zusätzlich diejenigen für die Subklasse (oder Erweiterung oder Spezialisierung) B speziell definierten.



Klasse A hat die sichtbaren (daher +) Attribute Name und Nummer.

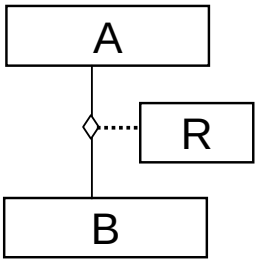


Objekte a der Klasse A und Objekte b der Klasse B stehen zueinander in **Beziehung** (von links nach rechts):

- **Assoziation**: schwache Beziehung, d.h. beteiligte Objekte sind eigenständig. [A: Halter - B: Auto]
- **Agregation**: stärkere Beziehung, d.h. die Objekte b in Klasse B sind zwar selbstständig möglich, zu einem Objekt a aus Klasse A gehören aber normalerweise gewisse Objekte b aus Klasse B. [A: Auto – B: Räder]
- **Komposition**: starke Beziehung, d.h. zu einem Objekt a aus Klasse A gehören gewisse Objekte b aus Klasse B zwingend [A: Auto – B: Chassis]

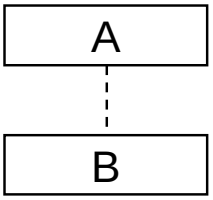
Mit Zahlen und/oder Symbolen wird die **Kardinalität**, d.h. die Anzahl der zugeordneten Objekte beschrieben. Steht eine solche Kardinalitäts-beschreibung bei der Beziehungslinie zwischen den Klassen A und B auf der Seite von B, so bedeutet sie: Einem a aus A sind zugeordnet

- beliebig viele oder kein b aus B (∗ oder 0..∗)
- beliebig viele aber mindestens ein b aus B (1..∗)
- beliebig viele b aus B aus dem gegebenen Bereich (z.B. 2..10)
- eine bestimmte Anzahl b aus B (z.B. 4)
- kein oder ein b aus B (0..1)

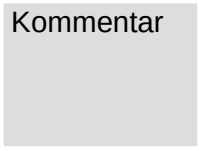


Name und Attribute einer Beziehung R (Assoziation oder Agregation oder Komposition) zwischen den Klassen A und B.

Als Zusatz zu UML



Objekte a der Klasse A und Objekte b der Klasse B stehen zueinander über geometrische Attribute (Koordinaten, Linien, Flächen) in Beziehung.

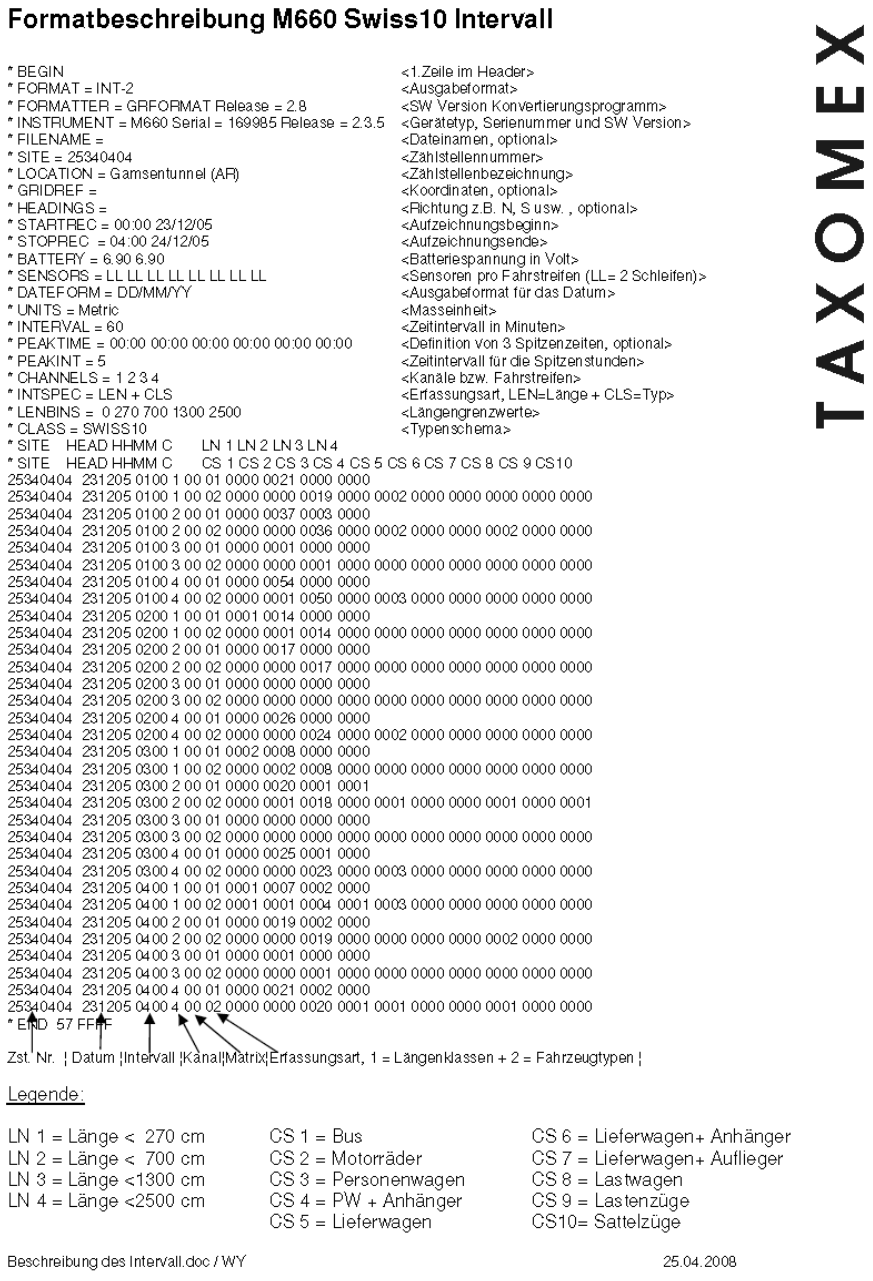


Für alle Elemente im markierten Bereich gilt der angegebene Kommentar.

Abbildung 7: Elemente von UML

5.1.1. Verkehrszähler Intervallsummen (Typ MM660) – Output

Hier geht es um das Datenmodell der Output-Schnittstelle des Verkehrszählers für Intervallsummen. Ausgangspunkt ist die Beschreibung des Realitätsausschnitts und vor allem das Beispiel mit den Daten und erläuternden Beschreibungen in Abbildung 8. Daraus gewinnt man zunächst die grafische Übersicht mit Hilfe des UML Klassendiagramms und dann das entsprechende INTERLIS 2 Datenmodell mit Präzisierung der Attributstypen.



Formatbeschreibung M660 Swiss10 Intervall

TAXOMEX

```
* BEGIN                                                  <1.Zeile im Header>
* FORMAT = INT-2                                         <Ausgabeformat>
* FORMATTER = GRFORMAT Release = 2.8                     <SW Version Konvertierungsprogramm>
* INSTRUMENT = M660 Serial = 169985 Release = 2.3.5      <Gerätetyp, Seriennummer und SW Version>
* FILENAME =                                             <Dateinamen, optional>
* SITE = 25340404                                        <Zählstellennummer>
* LOCATION = Gamsentunnel (AR)                           <Zählstellenbezeichnung>
* GRIDREF =                                              <Koordinaten, optional>
* HEADINGS =                                             <Richtung z.B. N, S usw. , optional>
* STARTREC = 00:00 23/12/05                              <Aufzeichnungsbeginn>
* STOPREC  = 04:00 24/12/05                              <Aufzeichnungsende>
* BATTERY = 6.90 6.90                                    <Batteriespannung in Volt>
* SENSORS = LL LL LL LL LL LL LL LL                      <Sensoren pro Fahrstreifen (LL= 2 Schleifen)>
* DATEFORM = DD/MM/YY                                    <Ausgabeformat für das Datum>
* UNITS = Metric                                         <Masseinheit>
* INTERVAL = 60                                          <Zeitintervall in Minuten>
* PEAKTIME = 00:00 00:00 00:00 00:00 00:00 00:00         <Definition von 3 Spitzenzeiten, optional>
* PEAKINT = 5                                            <Zeitintervall für die Spitzenstunden>
* CHANNELS = 1 2 3 4                                     <Kanäle bzw. Fahrstreifen>
* INTSPEC = LEN + CLS                                    <Erfassungsart, LEN=Länge + CLS=Typ>
* LENBINS =  0 270 700 1300 2500                         <Längengrenzwerte>
* CLASS = SWISS10                                        <Typenschema>
* SITE    HEAD HHMM C      LN 1 LN 2 LN 3 LN 4
* SITE    HEAD HHMM C      CS 1 CS 2 CS 3 CS 4 CS 5 CS 6 CS 7 CS 8 CS 9 CS10
25340404  231205 0100 1 00 01 0000 0021 0000 0000
25340404  231205 0100 1 00 02 0000 0000 0019 0000 0002 0000 0000 0000 0000 0000
25340404  231205 0100 2 00 01 0000 0037 0003 0000
25340404  231205 0100 2 00 02 0000 0000 0036 0000 0002 0000 0000 0002 0000 0000
25340404  231205 0100 3 00 01 0000 0001 0000 0000
25340404  231205 0100 3 00 02 0000 0000 0001 0000 0000 0000 0000 0000 0000 0000
25340404  231205 0100 4 00 01 0000 0054 0000 0000
25340404  231205 0100 4 00 02 0000 0001 0050 0000 0003 0000 0000 0000 0000 0000
25340404  231205 0200 1 00 01 0001 0014 0000 0000
25340404  231205 0200 1 00 02 0000 0001 0014 0000 0000 0000 0000 0000 0000 0000
25340404  231205 0200 2 00 01 0000 0017 0000 0000
25340404  231205 0200 2 00 02 0000 0000 0017 0000 0000 0000 0000 0000 0000 0000
25340404  231205 0200 3 00 01 0000 0000 0000 0000
25340404  231205 0200 3 00 02 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
25340404  231205 0200 4 00 01 0000 0026 0000 0000
25340404  231205 0200 4 00 02 0000 0000 0024 0000 0002 0000 0000 0000 0000 0000
25340404  231205 0300 1 00 01 0002 0008 0000 0000
25340404  231205 0300 1 00 02 0000 0002 0008 0000 0000 0000 0000 0000 0000 0000
25340404  231205 0300 2 00 01 0000 0020 0001 0001
25340404  231205 0300 2 00 02 0000 0001 0018 0000 0001 0000 0000 0001 0000 0001
25340404  231205 0300 3 00 01 0000 0000 0000 0000
25340404  231205 0300 3 00 02 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
25340404  231205 0300 4 00 01 0000 0025 0001 0000
25340404  231205 0300 4 00 02 0000 0000 0023 0000 0003 0000 0000 0000 0000 0000
25340404  231205 0400 1 00 01 0001 0007 0002 0000
25340404  231205 0400 1 00 02 0001 0001 0004 0001 0003 0000 0000 0000 0000 0000
25340404  231205 0400 2 00 01 0000 0019 0002 0000
25340404  231205 0400 2 00 02 0000 0000 0019 0000 0000 0000 0000 0002 0000 0000
25340404  231205 0400 3 00 01 0000 0001 0000 0000
25340404  231205 0400 3 00 02 0000 0000 0001 0000 0000 0000 0000 0000 0000 0000
25340404  231205 0400 4 00 01 0000 0021 0002 0000
25340404  231205 0400 4 00 02 0000 0000 0020 0001 0001 0000 0000 0001 0000 0000
* END  57 FFFF
```

Zst. Nr. ¦ Datum ¦Intervall ¦Kanal¦Matrix¦Erfassungsart, 1 = Längenklassen + 2 = Fahrzeugtypen ¦

Legende:

LN 1 = Länge < 270 cm	CS 1 = Bus	CS 6 = Lieferwagen+ Anhänger
LN 2 = Länge < 700 cm	CS 2 = Motorräder	CS 7 = Lieferwagen+ Auflieger
LN 3 = Länge <1300 cm	CS 3 = Personenwagen	CS 8 = Lastwagen
LN 4 = Länge <2500 cm	CS 4 = PW + Anhänger	CS 9 = Lastenzüge
	CS 5 = Lieferwagen	CS10= Sattelzüge

Beschreibung des Intervall.doc / WY 25.04.2008

Abbildung 8: Transferformat Verkehrszähler (Intervallsummen) – Streckenstation

Das Datenbeispiel in Abbildung 8 zeigt die Gliederung der Verkehrszählerdaten mit Intervallsummen in einen Header und in Messungszeilen. Je nachdem, ob der Transfer dieser Daten filebasiert oder meldungsbasiert erfolgen soll, sind verschiedene Modelle

der Datenstruktur der Schnittstelle zweckmässig. Es werden nachfolgend vier Varianten betrachtet.

Variante 1 zeigt die redundanzfreie Modellierung, d.h. jedes Datenelement kommt genau einmal vor. Die Messungszeilen sind in Messungselemente aufgeteilt. Es gibt den Gesamtheader, den Messungsheader und schliesslich das Messelement. Die Messelemente sind durch Komposition mit dem Messheader verknüpft und jeder Messheader wie der durch Komposition mit dem Gesamtheader. Für jedes Objekt der Klasse `LaengenS` (`FzKatS`) gilt `LSum > 0` (`FzSum > 0`).Wird zu einem bestimmten Zeitpunkt, der im Messheader definiert ist, zu einem bestimmten `LTyp` (`FzTyp`) kein Fahrzeug gezählt, d.h. `LSum = 0` (`FzSum = 0`), dann entsteht kein Objekt der Klasse `LaengenS` (`FzKatS`).

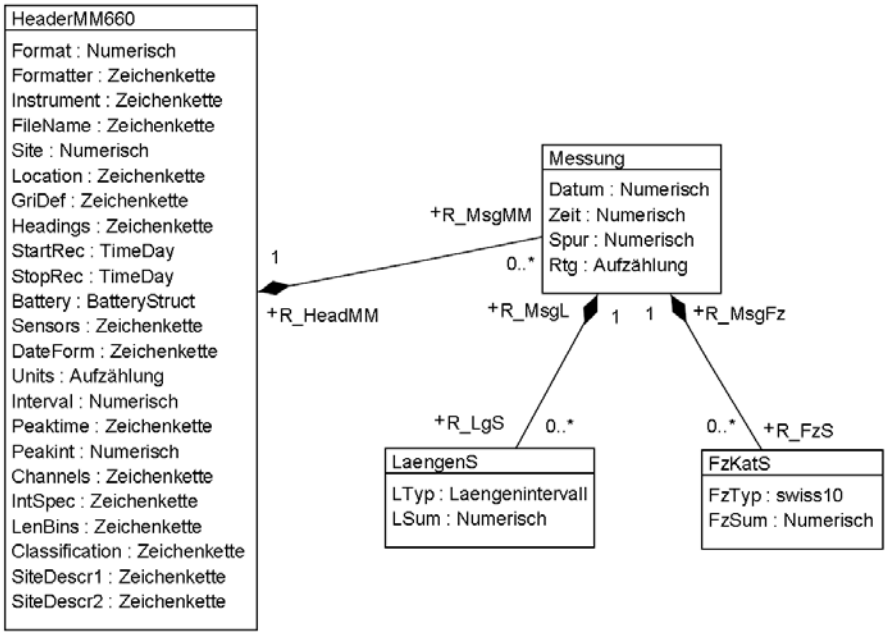


Abbildung 9: Variante 1 Datenstruktur Verkehrszähler Intervallsummen – Output

Variante 2 ist gegliedert in Gesamtheader (`HeaderMM660`, hier ohne Attribute, vergleiche Abbildung 10), Messheader (`Messung`) und Messungen (`MessungLgS`, `MessungFzKatS`). Ein Objekt der Messungsklassen besteht jetzt aus dem Array aller Messelemente einer Messung, die Nullwerte sind jetzt wieder dabei. Ferner sind diese Messungsklassen jetzt Spezialisierungen der Messheaderklasse. Für den Datentransfer heisst das, jedes Messungsobjekt der Klassen `MessungLgS` und `MessungFzKS` enthält nicht nur die Intervallsummen für jede Längen- bzw Fahrzeugtyp-Kategorie sondern auch die Werte aller Messheaderattribute. Damit sind die Messzeilen der Datei in Abbildung 8 elegant modelliert. Ein Objekt der beiden Messungsklassen entspricht genau einer kurzen bzw. langen Zeile der Datei in Abbildung 8.

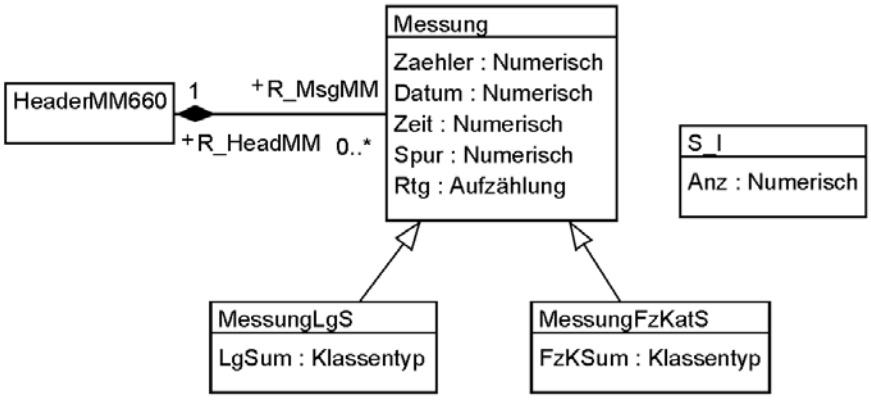


Abbildung 10: Variante 2 Datenstruktur Verkehrszähler Intervallsummen – Output (Details zur Klasse HeaderMM660 siehe Abbildung 9)

Diese Modellvariante 2 ist eher geeignet für meldungsbasierten als für filebasierten Transfer. Eine Meldung könnte aus einem Messungsobjekt bestehen mit integriertem Messheader. Der Gesamtheader müsste vorab als eigene Meldung übermittelt werden.

Im UML Diagramm ist der Attributstyp von `LgSum` in Klasse `MessungLgS` (bzw. von `FzKatS` in Klasse `MessungFzKatS`) nicht klar ersichtlich. Es handelt sich dabei um einen Array der Länge 4 (bzw. 10). Ein Array wird in INTERLIS 2 mit Schlüsselwort `LIST OF` als Liste der Länge 4 (bzw. 10) modelliert. Da als Listenelemente (noch) nicht Datentypen möglich sind, sondern nur Strukturelemente (d.h. Objekte ohne Identifikator), muss noch eine Struktur (verallgemeinerte Klasse) `S_Num` definiert werden mit einem einzigen numerischen Attribut .

Variante 3 ist wieder ein meldungsorientiertes Modell. Diesmal ohne Vererbung. Die Klasse Messung enthält die beiden Messungszeilen einer Spur zu einem bestimmten Zeitpunkt. Zeitstempel und Spurnummer sind für beide Messreihen nur einmal vorhanden. Hingegen bilden jetzt die zwei verschieden langen Zeilen von eigentlich sehr verschiedenem Inhalt ein Datenobjekt, was aus der Sicht des meldungsorientierten Transfers nicht so zweckmässig ist. Um zu den verschiedenen Daten zu gelangen, muss der Empfänger das erhaltene Datenobjekt auseinandernehmen. (Das Attribut RecTyp ist überflüssig.)

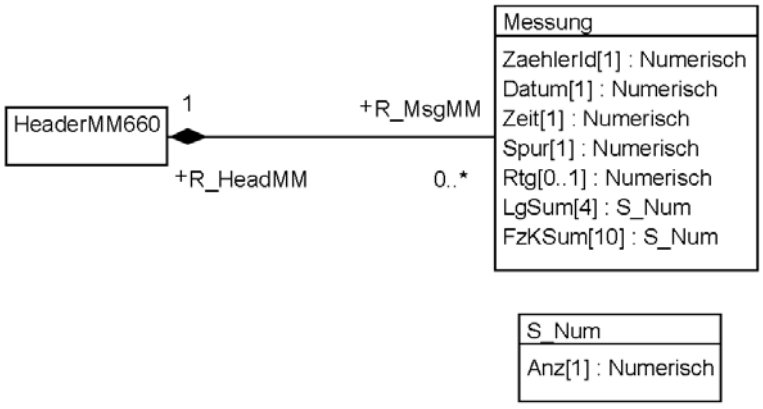


Abbildung 11: Variante 3 Datenstruktur Verkehrszähler Intervallsummen – Output (Details zur Klasse HeaderMM660 siehe Abbildung 9)

In den bisher besprochenen drei Varianten ist die Gesamtheaderklasse durch Komposition verknüpft mit den Messungsklassen.

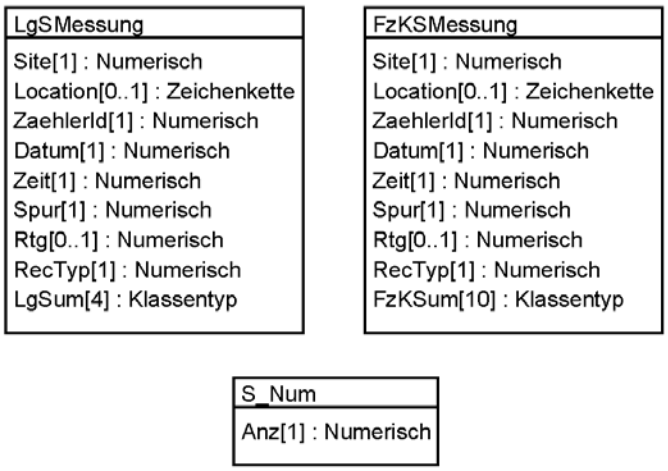


Abbildung 12: Variante 4 Datenstruktur Verkehrszähler Intervallsummen – Output

Variante 4 greift zurück auf Variante 2. Die beiden Messungszeilen pro Zeitpunkt und Spur sind ja eigentlich zwei verschiedene Objekte, die kürzere Zeile entspricht 4 Kategorien von Fahrzeuglängen und die grössere Zeile entspricht 10 Kategorien von Fahrzeugtypen. Entsprechend gibt es jetzt 2 Klassen, `LgSMessung` und `FzKatSMessung`. Damit sind zwar Zeitstempel und Spurnummer wieder redundant modelliert, aber dafür ist das Modell sehr nahe am Schnittstellenformat und liefert klare Meldungsobjekte. Variante 4 ist die “flachgewalzte“ Variante 2, d.h. die von den beiden Messungsklassen geerbten Attribute des Messungsheaders sind jetzt in den Messungsklassen notiert. Vom Gesamtheader werden vorerst nur die Attribute `site` und `location` übernommen. Die übrigen Attribute des Gesamtheaders scheinen vor allem für die Initialisierung der beteiligten Systeme und Verbindungen wesentlich zu sein, nicht aber für jede Meldung einer Messung.

5.1.2. Streckenstation – Output

Der Output der Streckenstation entspricht dem Datenbeispiel von Anhang A.1.2. Das entsprechende INTERLIS 2 Datenmodell mit Präzisierung der Attributstypen findet sich in Anhang A.3.2.

Dieses Datenmodell ist wieder auf Datei-Transfer ausgerichtet und möglichst redundanzfrei. Die Datei enthält Summen und Mittelwerte von fahrzeugweise erhobenen Anzahlen und Geschwindigkeiten über ein gewisses Zeitintervall. Der Gesamtheader (Spurbezeichnung) enthält den Zeitstempel für den Abschluss der Summenbildung, Spurnummer und Gesamttotal der Messungen über alle Fahrzeugkategorien. Summen und Mittelwerte pro Fahrzeugkategorie sind eigene Objekte entsprechend eigenen Klassen und werden nur in die Datei übernommen, wenn Summen und/oder Mittel verschieden sind von Null. Jede dieser Klassen ist durch Kompositionen mit der Ge-

samtheaderklasse verbunden. Fehlermeldungen sowohl für Streckenstation als auch für Verkehrszähler sind eigene Objekte entsprechend eigenen Klassen.

Für diesen Typ von Transfer fehlt im Gesamtheader die Streckenstationsnummer. Da in der aktuellen Situation der Datenaustausch zwischen Streckenstation und VDE Bereichsrechner mit OPC offenbar über Punkt-zu-Punkt-Verbindung erfolgt, wird davon ausgegangen, dass die Streckenstationsnummer nicht nötig ist (Identifikation über entsprechenden OPC-Parameter).

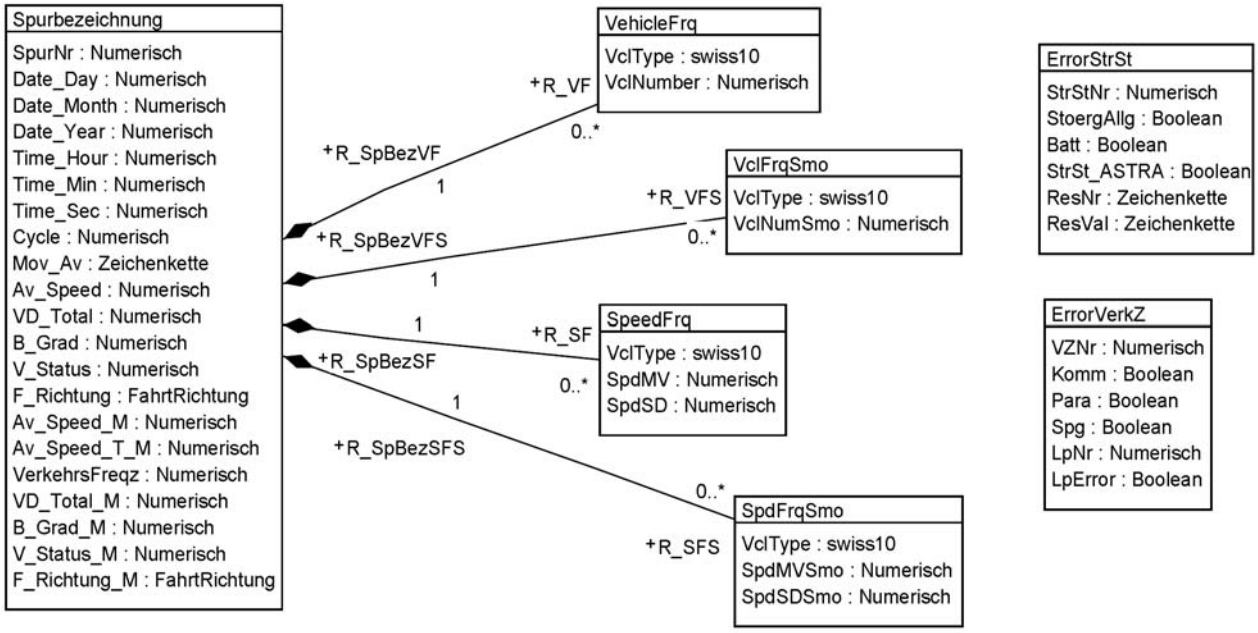


Abbildung 13: Datenstruktur Streckenstation – Output

5.1.3. Bereichsrechner VDE – Output

Entspricht XML-Schema von Anhang A.1.3. Das entsprechende INTERLIS 2 Datenmodell mit Präzisierung der Attributstypen findet sich in Anhang A.3.3.

Bemerkung: Die von den Klassen `Befehl, Antwort, Meldung` und `DatenpunktInfo` ausgehenden Beziehungen schliessen sich gegenseitig aus (XOR), es ist immer nur eine dieser Beziehungen möglich. Das kommt im UML-Diagramm nicht zum Ausdruck, da die entsprechende grafische Darstellung Teile des Diagramms unleserlich machen. Im INTERLIS 2 Datenmodell sind diese Assoziationen richtig definiert.

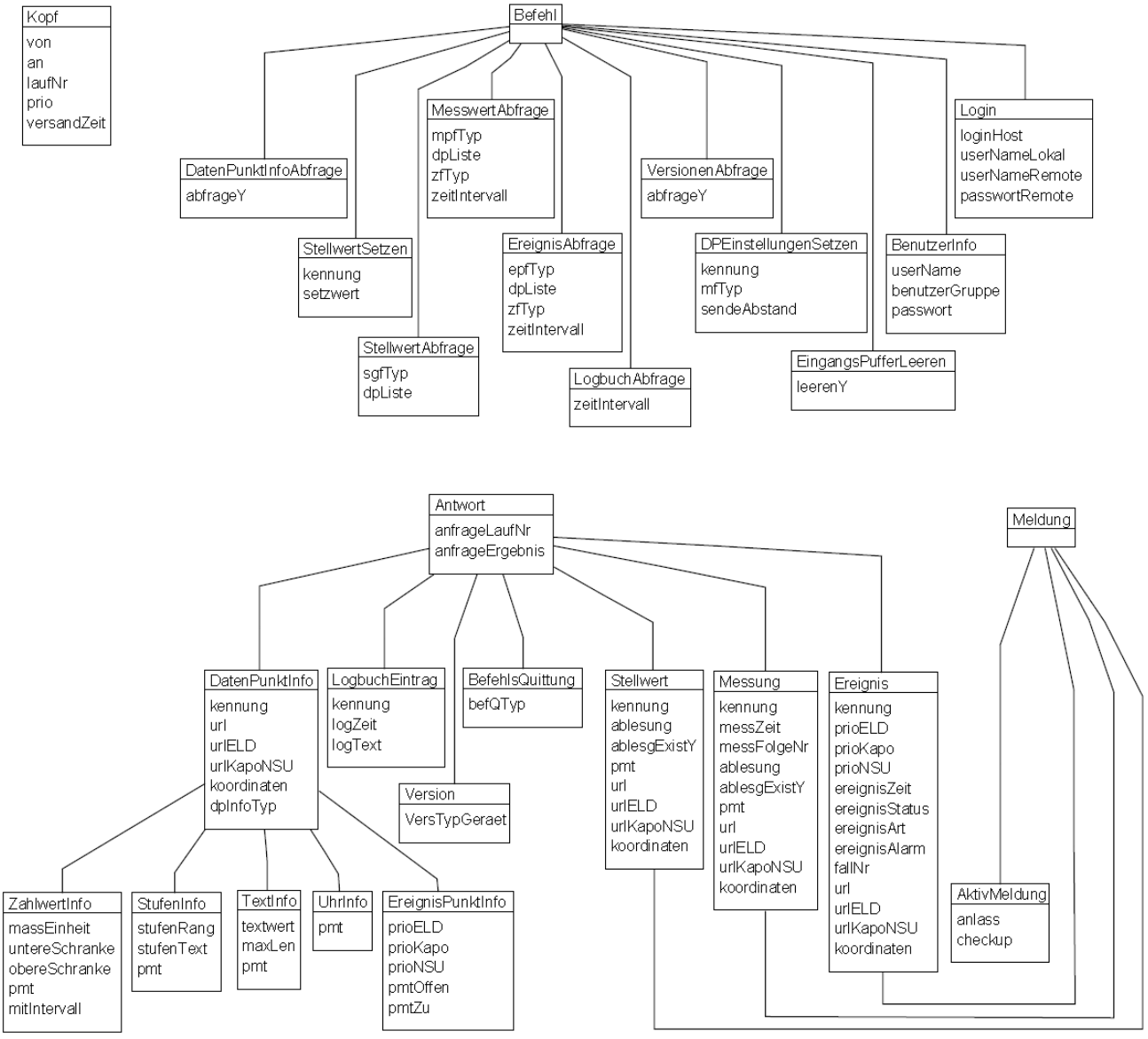


Abbildung 14: Datenstruktur Bereichsrechner VDE – Output

Aufgrund dieser sich gegenseitig ausschliessenden Beziehungen liegt das Splitting der Datenstruktur in kompakte Meldungsklassen auf der Hand. Jede der Klassen `Befehl` und `Meldung` bildet zusammen mit jeder einzelnen damit assoziierten Klasse eine solche kompakte Meldungsklasse. Für die Klasse `Antwort` ist es ebenso, ausser wenn `Datenpunktinfo` mit `Antwort` assoziiert ist. In diesem Fall ist auch noch eine der mit `Datenpunktinfo` XOR-assoziierten Klassen dazu zu nehmen. Das ergibt insgesamt 26 kompakte Meldungsklassen. Abbildung 15 zeigt einige Beispiele:

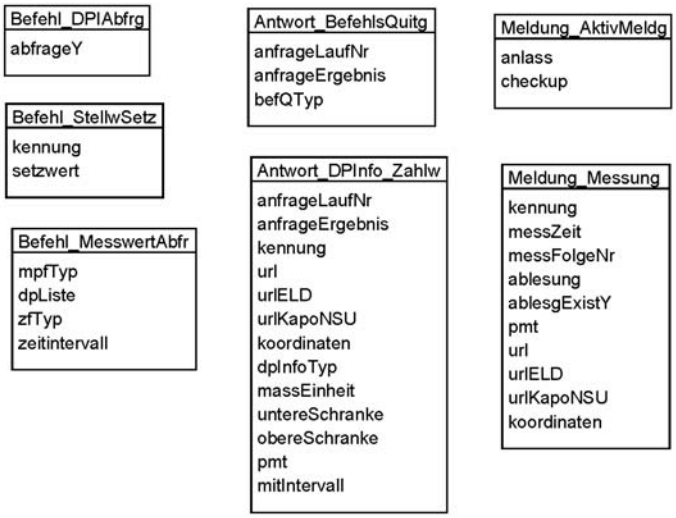


Abbildung 15: Beispiele für kompakte Meldungsklassen VDE – Output

An ihren Namen kann abgelesen werden, aus welchen assoziierten Originalklassen die kompakten Meldungsklassen entstehen. So ist etwa die kompakte Meldungsklasse `Befehl_DPIAbfrg` zusammengesetzt aus den Originalklassen `Befehl` und `DatenPunktInfoAbfrage` oder `Antwort_DPI_Zahlw` aus den drei Originalklassen `Antwort, DatenPunktInfo` und `ZahlwertInfo.`

Es stellt sich die Frage, welche dieser 26 kompakten Meldungsklassen effektiv zum Einsatz kommen. Aus der Sicht VDE werden vor allem die Themen Messwert und Ereignis eine wesentliche Rolle spielen, allenfalls auch noch Logbuch, und zwar bei Befehlen, Antworten und Meldungen. Aus Abbildung 14 wären damit die kompakten Meldungsklassen `Befehl_MesswertAbfr` und `Meldung_Messung` beteiligt.

Die Datenstruktur legt zwei Einsatzmöglichkeiten nahe:

1) Auf Befehl oder Anfrage der übergeordneten Verkehrmanagementzentrale wird vom unterstellten Bereichsrechner die entsprechende Antwort übermittelt, dann vermutlich meist als Datei. Die Klasse `Befehl` und die mit ihr assoziierten Klassen beschreiben in diesem Fall die Struktur der Input-Datenschnittstelle des Bereichsrechners gegenüber der übergeordneten Verkehrsmanagement-Zentrale.
2) Auf Grund von Input der unterstellten Streckenstationen übermittelt der Bereichsrechner selbständig Meldungen an die übergeordnete VM-Zentrale, dann eher meldungsbasiert.

Für die VDE trifft am ehesten Variante 2 zu.

5.1.4. Verkehrszähler Einzelfahrzeuge (Typ MM660) – Output

Definition Verkehrszähler M660 OSP - Format

TAXOMEX

```
* SITE = 04240404                          <Zählstellennummer>
* SENSORS = LL LL LL LL LL LL LL LL        < Sensoren pro Fahrstreifen (LL= 2 Schleifen)>
* PRUNITS = KPH-CM-10KG                    <Augabeeinheiten>
* CLASS = SWISS10                          <Typenschema>
* OSPFILTER = ALL                          <Datenfilter, ALL = Alle Fahrzeuge>
* OSPTIME = 0                              <Zeitlimite der Datenausgabe>
* OSPVEH = 0                               <Mengenlimite der Datenausgabe>
* HEAD DDMMYY HHMM SS HH RESCOD  L D HEAD  GAP SPD LENTH CS CH
036120 180603 1427 55 61 000000  4 1  1.2  1.0  78   446  3 VL
036121 180603 1427 55 45 000000  1 1  3.7  3.5  87  1127  8  H
036122 180603 1427 56 55 000000  4 1  0.9  0.7  79   382  3  L
036123 180603 1427 57 21 000000  1 1  1.7  1.2  84   440  3  L
036124 180603 1427 58 13 000000  4 1  1.5  1.4  79   487  3  M
036125 180603 1427 58 68 000000  4 1  0.5  0.3  77   491  3  H
036126 180603 1427 59 19 000000  1 1  1.9  1.7  78   427  3  L
036127 180603 1427 59 92 000000  1 1  0.7  0.5  75   410  3  L
036128 180603 1428 00 15 000000  4 1  1.4  1.2  71   424  3  L
036129 180603 1428 01 14 000000  3 1  7.5  7.3  92   539  5  M
036130 180603 1428 01 67 000000  4 1  1.5  1.3  74   453  3 VL
036131 180603 1428 01 85 000000  3 1  0.7  0.4  89   411  3  M
036132 180603 1428 02 33 000000  4 1  0.6  0.4  74   458  3  L
036133 180603 1428 04 44 000000  3 1  2.5  2.4  98   464  3 VL
036134 180603 1428 06 37 000000  4 1  4.0  3.8  80   412  3  L
036135 180603 1428 07 02 000000  3 1  2.5  2.4  93   439  3 VL
036136 180603 1428 08 16 000000  3 1  1.1  0.9  84   452  3  L
036137 180603 1428 08 75 000000  1 1  8.8  8.6  83   418  3  L
```

Legende:

Bezeichner	*Erklärung*	*String - Position*
HEAD	Fortlaufende Fahrzeugnummer	1 – 6
DDMMYY	Tag /Monat/Jahr	8 – 13
HHMM	Stunden / Minuten	15 – 18
SS	Sekunden	20 – 21
HH	Hundertstelssekunden	23 – 24
RESCOD	Technischer Ergebniscode	26 – 31
L	Spur (1 – 8)	34
D	Überfahrrichtung der Schlaufen (1 oder 2)	36
HEAD	Fahrzeugabstand Front --> Front in Sekunden	38 – 41 (max.99.9)
GAP	Fahrzeugabstand Front --> Heck (Lücke) in Sekunden	43 – 46 (max.99.9)
SPD	Geschwindigkeit (km/h)	48 – 50 (999)
LENTH	Fahrzeuglänge (cm)	52 – 56 (9999)
CS	Fahrzeugklasse (gemäss eingestelltem Typenschema)	58 – 59
CH	Chassishöhe (VL, L, M, H)	61 – 62

Format OSP_ohne WIM.doc 25.04.2008 / WY

Abbildung 16: Transferformat Verkehrszähler Einzelfahrzeuge (Typ MM660) – VM-CH-System

Betrachtet wird hier der Verkehrszähler für Einzelfahrzeugdaten Modell Marksman 660. Die Beschreibung des entsprechenden Realitätsausschnitts findet sich in Kapitel 4.4.4 und ein Beispiel mit Daten in Abbildung 8. Hier gewinnt man die graphische

Übersicht über die Datenstruktur mit Hilfe des UML-Klassendiagramms. Das entsprechende INTERLIS 2 Datenmodell mit Präzisierung der Attributstypen findet sich im Anhang A.3.4.

Dem Datenbeispiel in *Abbildung 16* ist zu entnehmen, dass diese Einzelfahrzeugdaten in einen Header und in Messungszeilen gegliedert sind. Auch hier werden zwei Modellvarianten betrachtet, die eine für filebasierten und die andere für meldungs- bzw. messungsbasierten Transfer.

Zu Variante 1 für den Datei-Transfer gibt es eine Headerklasse `EfzHeaderMM660` und eine Messungsdatenklasse `EfzData.` Jedem Headerobjekt sind die entsprechenden Messungsobjekte durch Komposition zugeordnet. Transferiert wird eine Datei bestehend aus einem oder mehreren Headern und den je zugehörigen Messzeilen.

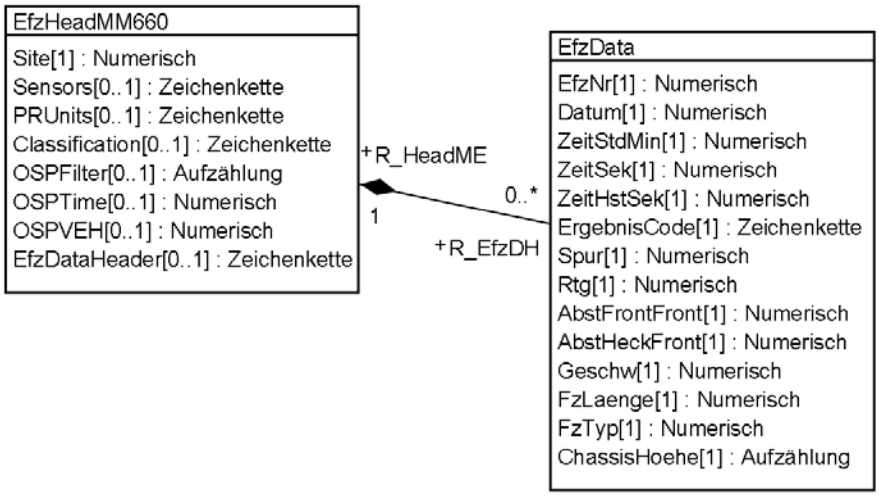


Abbildung 17: Datenstruktur Verkehrszähler Einzelfahrzeuge (Typ MM660) Output (Variante 1)

Variante 2 andererseits ist wieder mehr meldungsorientiert, besteht nur aus einer Klasse und übernimmt vom Header nur gerade das Attribut `Site.` Transferiert wird jede Meldungszeile einzeln.

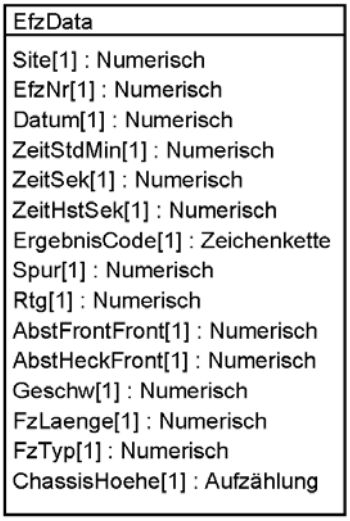


Abbildung 18: Datenstruktur Verkehrszähler Einzelfahrzeuge (Typ MM660) Output (Variante 2)

5.1.5. Verkehrszähler Einzelfahrzeuge (Typ ECTN) – Output

ECTN

Datenformat der Datensätze in den exportierten CSV-Dateien

Verfügbare Spalten

Spalte	Beschreibung
Datum, Zeit, Zeit (ms)	Datum, Zeit und Sekundenbruchteil der Fahrzeugfront unter der Messstelle
Kategorie	Kategorie TLS 8+1 (oder Swiss10, je nach Bedarf)
Spur	Spurnummer gemäss separater Definition
Fahrtrichtung	True = Normalrichtung, False = Gegenrichtung
Geschwindigkeit	Durchschnittliche Geschwindigkeit des Fahrzeuges bei der Durchfahrt durch den Messbereich
Länge	Länge des Fahrzeuges
Höhe	Maximale Höhe des Fahrzeuges
Breite	Maximale Breite des Fahrzeuges zur Längsachse
Anzahl Gefahrgutschilder	Anzahl Erfassungen von Gefahrgutschildern auf diesem Fahrzeug (ein oder mehrere verschiedene Schilder)
Gefahrgutnummer	Gefahrenzahl gemäss ADR 2007 Zusätzlich: 0 bedeutet „leeres Gefahrgutschild“ -1 bedeutet „nicht als Gefahrgutschild identifiziert“
Substanznummer	UN-Nummer gemäss ADR 2007 Zusätzlich: 0 bedeutet „leeres Gefahrgutschild“ -1 bedeutet „nicht als Gefahrgutschild identifiziert“

Datum	Zeit	Zeit (ms)	Kategorie	Spur	Fahrtrichtung	Geschw. [m/s]	Länge [mm]	Höhe [mm]	Breite [mm]	Anzahl Gefahrg.S	GefahrgtNr (Schild 1)	SubstzNr (Schild 1)
08.03.2008	23:59:54	773	7	1	TRUE	33.47284	4920	1420	1643	0		
08.03.2008	23:59:53	573	7	1	TRUE	29.12615	3495	1350	1561	0		
08.03.2008	23:59:46	610	7	1	TRUE	33.77201	4052	1644	1731	0		
08.03.2008	23:59:34	983	7	1	TRUE	45.73935	4253	1421	1544	0		
08.03.2008	23:59:21	540	7	1	TRUE	26.36339	4560	1939	1755	0		
08.03.2008	23:59:11	57	7	1	TRUE	30.82983	4100	1394	1666	0		
08.03.2008	23:58:36	290	7	1	TRUE	33.26793	4890	1445	1882	0		
08.03.2008	23:58:31	343	7	1	TRUE	32.96377	4384	1468	1732	0		
08.03.2008	23:58:27	980	7	1	TRUE	28.35821	3800	1392	1462	0		
08.03.2008	23:58:26	273	7	1	TRUE	30.32176	4851	1493	1742	0		

Abbildung 19: Transferformat Verkehrszähler Einzelfahrzeuge (Typ ECTN) – VM-CH-System

Der Verkehrszähler Typ ECTN ist eine weitere Möglichkeit, um Einzelfahrzeugdaten zu erfassen. Wie der Detailbeschreibung und dem Datenbeispiel in Abbildung 19 (Transferformat Verkehrszähler Einzelfahrzeuge (Typ ECTN) – VM-CH) zu entnehmen ist, gibt es hier keine Headerdaten und sogar die Identifikation des Zählerstandor-

tes fehlt. Hier hat man von vornherein nur die Klasse der Messobjekte, entsprechend der Variante 2 des Marksmanzählers von Kapitel 5.1.4.

Abbildung 20: Datenstruktur Verkehrszähler Einzelfahrzeuge (Typ ECTN) – Output

5.1.6. VM-CH-System, harmonisierte Struktur von Einzelfarzeugdaten – Input (Vorschlag)

Wie in den Kapitel 5.1.4 und 5.1.5 aufgezeigt, gibt es mindestens zwei verschiedene Verkehrszähler, die Einzelfahrzeugdaten an das System VM-CH liefern.

Es stellt sich damit eine dritte Frage im Zusammenhang mit der Datenmodellierung: Welche Datenstruktur kann ein Zielsystem erwarten, das von verschiedenen Startsystemen bedient wird? Zur Beantwortung dieser Frage müssen zunächst die Datenstrukturen der verschiedenen Lieferantensysteme analysiert und verglichen werden. Für die Definition der Zielstruktur – wenn sie nicht fest vorgegeben ist – gibt es zwei Extrem-möglichkeiten:

- Die „dicke“ Variante: Alle Attribute werden übernommen ins Zielsystem und diejenigen Attribute, die nicht in allen Startsystemen vorkommen, sind optional.
- Die „dünne“ Variante: Es werden nur diejenigen Attribute in die Zielstruktur übernommen, welche in jeder Startstruktur vorkommen.

Das folgende UML-Diagramm und das entsprechende INTERLIS 2 Modell aus Anhang A.3.6 ist eine „dünne“ Kombination der Kapitel 5.1.4 und 5.1.5, bestehend aus allen obligatorischen Attributen; das sind diejenigen mit Kardinalität [1] im UML-Diagramm Abb. 21. Diese „dünne“ Kombination wurde ergänzt um einige Attribute, die nur im einen oder anderen Modell vorkommen und daher fakultativ sind, d.h. mit Kardinalität [0..1] im UML-Diagramm Abb. 21. Beispiele: `AbstFrontFront, AbstHeckFront, Chassishoehe` aus Kapitel 5.1.4. Andere Attribute sind weggelassen, zum Beispiel `EfzNr, Ergebniscode` aus Kapitel 5.1.4.

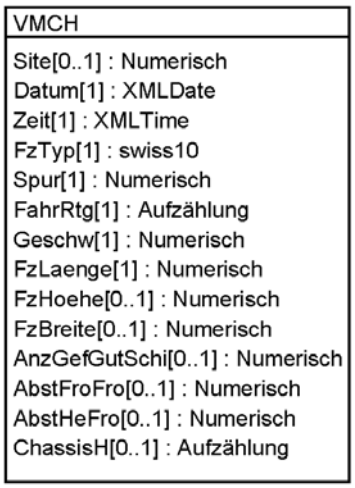


Abbildung 21: Datenstruktur Verkehrszähler Einzelfahrzeuge harmonisiert - Input

5.2. Datenstrukturen: Präzise Beschreibung mit INTERLIS 2

Es geht immer noch um das Element (B) des modellbasierten Vorgehens (3.3 und 3.4), nämlich um das konzeptionelle Datenmodell. Allerdings soll das jetzt nicht mehr grafisch beschrieben werden, sondern als Text mit Hilfe einer formalen Sprache, einer sogenannten konzeptionellen Beschreibungssprache (Conceptual Schema Language CSL)

Während das UML-Klassendiagramm mit seiner Grafik ideal den Überblick über eine Datenstruktur vermittelt, bietet die textuelle konzeptionelle Beschreibungssprache INTERLIS 2 die nötige Präzision. Vergleichen wir zunächst UML und INTERLIS für die Klasse VMCH. .Abb. 21 von Kapitel 5.1.6 zeigt, dass in UML pro Attribut bestenfalls Name, Kardinalität und grobe Umschreibung des Attributstyps möglich sind. Dagegen kann mit INTERLIS der Attributstyp genau präzisiert werden, wie der folgende Ausschnitt des entsprechenden INTERLIS-Schemas zeigt (aus Anhang A.3.6, Details siehe INTERLIS Referenzhandbuch [7] Kapitel 2.8).

```
MODEL VMCH_Mod (de) AT "http://www.gis.ethz.ch" VERSION "20080829"  =
  IMPORTS Units,INTERLIS;

  DOMAIN

    swiss10 = (Bus, Motorrad, Personenwagen, PWmitAnhaenger, Lieferwagen,
      LieferwagenMitAnhaenger, LieferwagenMitAufleger, Lastwagen,
      Lastenzug, Sattelzug
    );

  TOPIC VMCH_Top =

    CLASS VMCH =
      Site : 0 .. 99999999;
      Datum : MANDATORY INTERLIS.XMLDate;
      Zeit : MANDATORY INTERLIS.XMLTime;
      FzTyp : MANDATORY VMCH_Mod.swiss10;
      Spur : MANDATORY 1 .. 8;
      FahrRtg : MANDATORY (keine_0, normal_1, gegen_2, beide_9);
      Geschw : MANDATORY 0.0000 .. 250.0000 [Units.kmh];
      FzLaenge : MANDATORY 1000 .. 99999 [Units.mm];
      ......etc.
```

Abbildung 22: INTERLIS 2 Modell Verkehrszähler Einzelfahrzeuge VM-CH - Input

Für die Attribute `Site` und `Spur` sind die Intervalle natürlicher Zahlen definiert durch Angabe der unteren und der oberen Grenze getrennt durch 2 Punkte. Auch die `FzLaenge` ist ein solcher Bereich natürlicher Zahlen. Zusätzlich ist in eckigen Klammern die Einheit `mm` angegeben, welche im Standardmodell `Units` definiert ist ([7], Anhang F). Für die `Geschw` ist ein Intervall von Dezimalzahlen definiert mit 4 Nachkommastellen und der Einheit `kmh` ebenfalls aus dem Modell `Units.` Interessant ist der Aufzähltyp von `FahrRtg,` mit dem die einzelnen Werte des Attributs und ihre Nummer angegeben werden können. Auch zum `FzTyp` gehört ein Aufzähltyp. Er wird in der Attributszeile mit seinem Namen `swiss10` referenziert, zu dem in diesem Datenmodell `VMCH_mod` im Bereich `DOMAIN` die zugehörigen Werte definiert sind. Schliesslich brauchen die Attribute `Datum` und `Zeit` je die vordefinierten Attributstypen `XMLDate` und `XMLTime,` beide aus dem Basismodell `INTERLIS` ([7] Anhang A)

Im UML – INTERLIS Editor [10] wird im Hintergrund das der Grafik entsprechende INTERLIS 2 Modell aufgebaut. Die Attributstypen können über Menus per Mausklick ausgewählt werden und der korrekte INTERLIS Text des ganzen Modells kann am Schluss automatisch erstellt werden.

Beziehungen werden aus dem UML-Klassendiagramm exakt übersetzt in INTERLIS 2, wie das folgende Beispiel zeigt. Abbildung 23 ist eine vereinfachte Sicht des UML-Diagramms für den Verkehrszähler von Einzelfahrzeugen Typ MM660 in Kapitel 5.1.4 (Abbildung 17). Die Beschreibung der Attribute ist in beiden Klassen weggelassen.

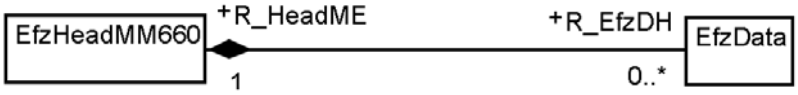


Abbildung 23: UML-Diagramm (ohne Attribute) für Verkehrszähler von Einzelfahrzeugen (Typ MM660) – Output (Variante 1)

Die entsprechenden INTERLIS 2 Beschreibung der Beziehung (Komposition) zwischen den Klassen `EfzHeadMM660` und `EfzData` zeigt Abbildung 24. Die Assoziationsklasse `A_HeadMM_EfzD` übernimmt exakt die Rollenbeschreibung aus dem UML-Diagramm und beinhaltet für die Rolle jeder der beteiligten Klassen Rollennamen, Stärke, Kardinalität und Klassennamen. Man beachte insbesondere das eng an die Grafik angelehnte Symbol `-<#>` für die Stärke Komposition (Details in [7] Kapitel 2.7.).

```
ASSOCIATION A_HeadMM_EfzD =
  R_HeadME -<#> {1} EfzHeadMM660;
  R_EfzDH -- {0..*} EfzData;
END A_HeadMM_EfzD;
```

Abbildung 24: INTERLIS 2 Schema für die Beziehung in Abbildung 23 (aus dem Datenmodell in Anhang A.1.4)

Weitere wesentliche Möglichkeiten der Datenbeschreibung, die INTERLIS 2 bietet aber die bei unserer SVT-Modellierung nur beschränkt zum Zuge kamen, sind:

- Geometrische Attributstypen für Punkte, Polylinien, unabhängige Flächenstücke (die sich überlagern können) und Gebietseinteilungsflächen (die höchstens Randpunkte gemeinsam haben dürfen) ([7] 2.8.12 und 2.8.13..
- Präzise Beschreibungsmöglichkeit für Koordinatensysteme.([7] Anhang I)
- Konsistenzbedingungen: Wir verwendeten das Schlüsselwort `MANDATORY` für die Kennzeichnung obligatorischer Attribute. Mit einer `UNIQUE` Klausel können eindeutige Attribute (Schlüssel) bezeichnet werden. Weitere Details zu Konsistenzbedingungen mit logischen Ausdrücken und Funktionen in [7] Kapiteln 2.12, 2.13, 2.14. Wesentlich dabei ist, dass die so formulierten Konsistenzbedingungen durch den INTERLIS 2 Checker ([13] freeware) automatisch überprüft werden können.
- Darstellungsbeschreibung ([7] Kapitel 2.16)
- etc.

Die vollständigen INTERLIS Datenmodelle zu den UML Klassendiagrammen von 5.1 befinden sich in Anhang A.3

6. Standard Transferformate

6.1. Automatische Herleitung aus dem Datenmodell

Wenn das Datenmodell im Schritt (C) des modellbasierten Vorgehens mit der konzeptionellen Beschreibungssprache INTERLIS 2 als Text formuliert ist, wie in Kapitel 5.2 besprochen, dann kann im nun folgenden Schritt (D) daraus für den Transfer der entsprechenden Daten automatisch die Beschreibung des Standard Transferformats hergeleitet werden. Die nötigen Regeln sind normativ festgelegt in Kapitel 3 der Schweizer Norm SN612031 [8] und des INTERLIS Referenzhandbuchs [7].

Diese Codierungsregeln sollen nur kurz anhand des Vergleichs von Modell und Transferdaten in den beiden folgenden Abbildungen erläutert werden. Abbildung 25 enthält das Datenmodell für die Messungen des Einzelfahrzeugzählers Marksman 660 in INTERLIS 2 und Abbildung 26 den Start der Transferdatei und die beiden ersten Messungen vollständig im XML Transferformat.

```
INTERLIS 2.3;

MODEL ZaehlEfz (de) AT "http://www.gis.ethz.ch" VERSION "20080829"  =
  IMPORTS INTERLIS,Units;

  TOPIC ZE_MM660 =

    CLASS EfzData =
      Site : MANDATORY 1 .. 99999999;
      EfzNr : MANDATORY 1 .. 999999;
      Datum : MANDATORY 10100 .. 311299;
      ZeitStdMin : MANDATORY 0 .. 2359;
      ZeitSek : MANDATORY 0 .. 59 [INTERLIS.s];
      ZeitHstSek : MANDATORY 0 .. 99;
      ErgebnisCode : MANDATORY TEXT*6;
      Spur : MANDATORY 1 .. 8;
      Rtg : MANDATORY 0 .. 9;    !! (keinVerk_0, normal_1, gegen_2, undef_9);
      AbstFrontFront : MANDATORY 0.1 .. 99.9 [INTERLIS.s];
      AbstHeckFront : MANDATORY 0.1 .. 99.9 [INTERLIS.s];
      Geschw : MANDATORY 0 .. 299 [Units.kmh];
      FzLaenge : MANDATORY 100 .. 9999 [Units.cm];
      FzTyp : MANDATORY 1 .. 10; !! Swiss10;
      ChassisHoehe : MANDATORY (VL, L , M, H);
    END EfzData;

  END ZE_MM660;

END ZaehlEfz.
```

Abbildung 25: Konzeptionelles Datenmodell für Einzelfahrzeugzähler Typ Marksman 660

Zuerst zur XML Datei in Abbildung 26. Wie jede XML Datei besteht auch diese vor allem aus XML Elementen. Ein solches besteht aus einem Wert, der eingerahmt ist durch eine Anfangsmarke und eine Endmarke, auch Anfangstag und Endtag genannt („tag“ hier Englisch für „Marke“), z.B. `<Datum>180603</Datum>` .

XML Elemente können geschachtelt werden:

```
<Datum>
  <Tag>18</Tag>
  <Monat>06</Monat>
  <Jahr>03</Jahr>
</Datum>
```

Nach diesen Startbemerkungen zu XML hier der Beginn der Marksmann-Zählerdaten im Standardsformat INTERLIS 2 XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.3">
  <HEADERSECTION VERSION="2.3" SENDER="Gn&#228;gi">
    <MODELS>
      <MODEL NAME="ZaehlEfz" URI="http://www.gis.ethz.ch/" VERSION="2008-04-14" />
    </MODELS>
    <ALIAS>
      .......Unter XML Elemente weggelassen, Details dazu in [7] Kapitel 3
    </ALIAS>
  </HEADERSECTION>
  <DATASECTION>
    <ZaehlEfz.ZE_MM660 BID="8000">
      <ZaehlEfz.ZE_MM660.EfzData TID="8001">
        <Site>4240404</Site>
        <EfzNr>36120</EfzNr>
        <Datum>180603</Datum>
        <ZeitStdMin>1427</ZeitStdMin>
        <ZeitSek>55</ZeitSek>
        <ZeitHstSek>61</ZeitHstSek>
        <ErgebnisCode>000000</ErgebnisCode>
        <Spur>4</Spur>
        <Rtg>1</Rtg>
        <AbstFrontFront>1.2</AbstFrontFront>
        <AbstHeckFront>1.0</AbstHeckFront>
        <Geschw>78</Geschw>
        <FzLaenge>446</FzLaenge>
        <FzTyp>3</FzTyp>
        <ChassisHoehe>VL</ChassisHoehe>
      </ZaehlEfz.ZE_MM660.EfzData>
      <ZaehlEfz.ZE_MM660.EfzData TID="8002">
        <Site>4240404</Site>
        <EfzNr>36121</EfzNr>
        <Datum>180603</Datum>
        <ZeitStdMin>1427</ZeitStdMin>
        <ZeitSek>55</ZeitSek>
        <ZeitHstSek>45</ZeitHstSek>
        <ErgebnisCode>000000</ErgebnisCode>
        <Spur>1</Spur>
        <Rtg>1</Rtg>
        <AbstFrontFront>3.7</AbstFrontFront>
        <AbstHeckFront>3.5</AbstHeckFront>
        <Geschw>87</Geschw>
        <FzLaenge>1127</FzLaenge>
        <FzTyp>8</FzTyp>
        <ChassisHoehe>H</ChassisHoehe>
      </ZaehlEfz.ZE_MM660.EfzData>
      <ZaehlEfz.ZE_MM660.EfzData TID="8003">
        <Site>4240404</Site>
        <EfzNr>36122</EfzNr>
        ......... (etc)
        .........
      </ZaehlEfz.ZE_MM660.EfzData>
    </ZaehlEfz.ZE_MM660>
  </DATASECTION>
</TRANSFER>
```

Abbildung 26: Daten im XML Standardformat für Einzelfahrzeugzähler Typ Marksman 660 (2 ganze Messungen und Rahmen XML Elemente)

Ab Zeile 13 in Abbildung 26 finden wir die Messwerte einer ersten Messung des Verkehrszählers, pro Zeile einen Messwert. Der erste Wert lautet **4240404**, er ist eingerahmt von einem Anfangstag **<Site>** und von einem Endtag **</Site>.** Der zweite Wert lautet **36120** mit Anfangstag **<EfzNr>** und Endtag **</EfzNr>** . Dabei handelt es sich offensichtlich um die Werte der beiden ersten Attribute im Datenmodell von Abbildung 23 und als Namen in den tags werden gerade die Attributsnamen verwendet. Das ist eine erste Codierungsregel für das Standardformat.

Eine weitere Codierungsregel ist auch leicht ersichtlich: Die XML Elemente der Attribute sind Unterelemente des XML Elementes für ein Datenobjekt. Vor dem ersten Attributswert kommt der Anfangstag des Objektes und nach dem letzten Attribut der Endtag des Objektes. Vergleichen wir den Namen in den Objekttags mit dem Datenmodell von Abbildung 25, stellen wir fest, dass sich dieser Name zusammensetzt aus Modellname (hier **ZaehlEfz**), Topicname (hier **ZE_MM660**) und Klassenname (hier **EfzData**), je getrennt durch einen Punkt. Dieser Name ist für alle Objekte einer Klasse gleich, in Abbildung 26 **ZaehlEfz.ZE_MM660.EfzData** .

Im Anfangstag des Objektes finden wir noch eine andere Art, wie man im XML Format Werte codieren kann, nämlich nicht nur als XML Elemente, sondern auch als XML Attribute; **TID="8001"** ist ein solches XML Attribut. Es besteht aus einem Namen **TID** , dem durch das Gleichheitszeichen **=** ein Wert in Gänsefüsschen **"8001"** zugeordnet ist. Diese Codierung als XML Attribut **TID="8001"** ist grundsätzlich gleichwertig wie die entsprechende als XML Element **<TID>8001</TID>** . XML Attribute müssen immer im Anfangstag eines XML Elementes stehen. Die Transferidentifikation **TID** ist ein Hintergrundsattribut, um das sich der objektorientierte Modellierer nicht zu kümmern braucht: Jedes Datenobjekt hat eine Objektidentifikation (OID), die ihm vom System, das dieses Objekt erzeugt, automatisch zugeordnet wird. Diese Objektidentifikation kann auch als Transferidentifikation (TID) verwendet werden. Daher muss die TID nicht als Attribut modelliert werden, und weil sie nicht als Attribut modelliert wird, codiert man sie auch nicht als Unter XML Element sondern als XML Attribut des Objekt XML Elementes.

Weitere XML Attribute, um deren Werte man sich konzeptionell nicht zu kümmern braucht, finden sich in den XML Elementen am Anfang der Transferdatei. Für weitere Details, insbesondere betreffend die Rahmen XML Elementen zu Topic (hier mit Namen **ZaehlEfz.ZE_MM660**) zur **DATASECTION** und zur **HEADERSECTION** sei auf das Kapitel 3 des INTERLIS Referenzhandbuchs [7] verwiesen.

Die Formatbeschreibung, die wir hier in Umgangssprache dargelegt haben, erfolgt präzis durch eine Formatbeschreibungssprache. Für XML Transferformate heisst diese Formatbeschreibungssprache „XML-Schema“. Die XML-Schema Beschreibung des Transferformats für ein Datenmodell ist selbst eine XML-Datei und wird vom Compiler auf Wunsch automatisch hergestellt. Diese ist sehr umfangreich, auch für das hier besprochene kleine Datenmodell. Sie ist vorhanden, aber in den Anhang A4 verbannt worden, kann dort besichtigt und mit Datenmodell und Daten in diesem Kapitel hier verglichen werden.

6.2. Vergleich mit den proprietären Formaten des Realitätsausschnitts

Um das proprietäre und das Standard Transferformats vergleichen zu können, betrachten wir zu den Daten im Standardformat der Abbildung 26 die entsprechenden Originaldaten des Einzelfahrzeugzählers im proprietären Format in Abbildung 27.

```
* SITE = 04240404 <Zählstellennummer>
..........weitere Headerzeilen
036120 180603 1427 55 61 000000 4 1 1.2 1.0 78 446 3 VL
036121 180603 1427 55 45 000000 1 1 3.7 3.5 87 1127 8 H
036122 180603 1427 56 55 000000 4 1 0.9 0.7 79 382 3 L
..........etc.
```

Abbildung 27: Daten im proprietären Originalformat (csv) für Einzelfahrzeugzähler Typ Marksman 660 (3 erste Messungen und 1 Headerzeile, aus Abbildung 16)

Im Datenmodell (Abbildung 25) und im Standardformat (Abbildung 26) ist je aus dem Header des proprietären Formats (Abbildung 27) nur das Attribut **SITE** übernommen worden als zusätzliches erstes Attribut jeder Messung. Die übrigen Attribute des ersten XML Messungsobjektes stimmen in Reihenfolge und Wert überein mit den Attributswerten der ersten proprietären Messungszeile in einem csv Format (comma separated values; die Separator sind allerdings nicht Kommas sondern Leerstellen!). Desgleichen für zweites XML Objekt und zweite proprietäre Messungszeile.

Der Vergleich der beiden Formate ergibt folgende Beurteilung des INTERLIS 2 XML Formats (mit: +/-, für die Beurteilung des csv-Formats ist +/- durch -/+ zu ersetzen):

+ enger Zusammenhang Format mit Modell über Tags
+ Browser-kompatibel
+ Standard öffnet Weg zu Tools
- wesentlich umfangreicher als csv

Als Ausgangspunkt für die Beschreibung der Datenstruktur des VDE Bereichsrechners war eine XML-Schema Beschreibung des entsprechenden proprietären XML-Transferformats gegeben (Anhang A.1.3). Wesentlich wäre der Vergleich des automatisch aus dem Datenmodell hergeleiteten INTERLIS 2-XML-Schemas mit Original XML-Schema des proprietären Formats (siehe Empfehlung weiteres Vorgehen in Kapitel 10).

7. Vom proprietären zum Standard-Format und umgekehrt mit 1:1 Prozessoren

Der 1:1 Prozessor ist Element (D) des modellbasierten Vorgehens. Er dient zum Umbau von Daten im proprietären Format (hier csv) auf Standardformat (hier INTERLIS 2 XML) oder umgekehrt. Wie in Kapitel 3.3 erläutert, ist ein 1:1 Prozessor grundsätzlich nicht ein kompliziertes Stück Software. Denn hinter beiden Formaten steckt dasselbe Datenmodell. Auf Grund des proprietären Formats wurde das Datenmodell entwickelt (MDA Element (B) Kapitel 3.3, 3.4 und 5) und aus diesem Datenmodell wurde die Beschreibung des Standardformats automatisch hergeleitet (MDA Element (C) Kap. 3.3, 3.4 und 6).

Es kann unterschieden werden zwischen starren und generischen 1:1 Prozessoren. Während ein starrer 1:1-Prozessor auf ein bestimmtes Datenmodell zugeschnitten und entsprechend einfach zu programmieren aber mühsam an andere Datenmodelle anzupassen ist, kann ein generischer 1:1-Prozessor einfacher an andere Datenmodelle angepasst werden. Er braucht dazu aber vorgefertigte Programmelemente, die mit dem Datenmodell in digitaler Form konfiguriert werden können. Das Demonstrator Programm umfasst zwei starre und zwei generische 1:1 Prozessoren.

Es gibt verschiedene Software-Werkzeuge, die als 1:1 Prozessoren verwendet werden können. FME von der kanadischen Firma Safe stellt für eine grosse Zahl von Transferformaten Umformatierungsfunktionen zur Verfügung und kann auch INTERLIS 2 XML (und INTERLIS 1 ITF) lesen und schreiben [16]. Ebenso die IG-Tools der Zürcher Firma InfoGrips [15] und das INTERLIS Studio der Firma GEOCOM aus Burgdorf. Nun sind aber die Formate, die wir bei den Systemen der SVT antreffen, nicht gängige Formate, die in den erwähnten Werkzeugen bereits integriert sind. Dieses Problem wäre lösbar, da diese Werkzeuge auch für die Bearbeitung von nicht integrierten Formaten konfiguriert werden können. Hingegen müssten wir diese Werkzeuge aus dem Rahmenprogramm unseres Demonstrators aufrufen können, über Programmschnittstellen. Das ist leider noch nicht möglich. Daher haben wir unsere 1:1 Prozessoren selbst programmiert. Wie am Anfang des Kapitels erwähnt, ist der entsprechende Aufwand vergleichsweise klein.

Als Werkzeuge für die Programmierung von 1:1 Prozessoren kommen „ausgewachsene“ Programmiersprachen (wie C++, Java etc.) oder Skriptsprachen in Frage (wie awk, von den Informatikern A.V. Aho, G.M. Weinberg und B.W. Kerningham entwickelt, C-ähnlich, mit sehr viel Zeichenketten Manipulationsfunktionen ausgestattet, als exe-Datei von 50 KB verfügbar, siehe [22]). Im Demonstrator wird Java verwendet.

Das Hauptanliegen eines 1:1 Prozessors ist die Änderung der Formatierung. Im Allgemeinen funktioniert ein 1:1 Prozessor nach folgendem Muster:

- Das Eingabeformat wird Stück für Stück (z.B. eine Zeile) gelesen und für jedes Stück ein Objekt in einer interne Datenstruktur erzeugt
- Für jedes Objekt in der internen Datenstruktur wird wieder ein Stück im Ausgabeformat geschrieben.

Die Struktur der internen Objekte ergibt sich aus dem fachlichen Datenmodell (das ja bei einem 1:1 Prozessor für das Ausgangs- und Zielformat identisch ist) und ist entwe-

der fix programmiert (starrer 1:1 Prozessor) oder wird vom 1:1 Prozessor selbst aus dem Datenmodell erzeugt (generischer 1:1 Prozessor).

Ein starrer 1:1 Prozessor ist im Prinzip aus 3 Abschnitten aufgebaut

- Einlesen der Datenfelder des Eingangsformats (im Falle des Output 1:1 Prozessors eines Verkehrszählers ein csv-Format) in einen String-Array.
- Allenfalls Decodieren gemäss Datenmodell und Produzieren von Fehlermeldungen. Darauf sollte allerdings eher verzichtet werden, denn die Fehlerprüfung erfolgt viel effizienter durch ein Checkerprogramm mit allen Raffinessen nach dem Codieren des Ausgabeformats.
- Codieren und schreiben des Ausgabeformats. Handelt es sich dabei um ein XML Encoding, dann sind die Tags fest programmiert gemäss Datenmodell.

Bei einem generischen 1:1 Prozessor ist die Struktur der internen Objekte nicht fix programmiert, sondern ergibt sich aus dem fachlichen Modell. Es ist also zwingend ein maschinenlesbares fachliches Modell erforderlich. Ein generischer 1:1 Prozessor funktioniert also nach folgendem Muster:

- Das fachliche Modell wird gelesen.
- Die Struktur der internen Objekte wird aus dem fachlichen Modell hergeleitet.
- Die Abbildung des Eingabeformats auf die internen Objekte wird hergeleitet.
- Die Abbildung der internen Objekte auf das Format wird hergeleitet.
- Das Eingabeformat wird Stück für Stück (z.B. eine Zeile) gelesen und für jedes Stück wird ein Objekt in der internen Datenstruktur erzeugt.
- Für jedes Objekt in der internen Datenstruktur wird wieder ein Stück im Ausgabeformat geschrieben.

Ein generischer 1:1 Prozessor ist nur möglich, wenn sowohl das Ausgangs- wie auch das Zielformat aus dem fachlichen Modell automatisch hergeleitet werden können.

8. Strukturumbau mit semantischer Transformation

8.1. Allgemeines und Beispiele

Die semantische Transformation ist das Element (E) des modellbasierten Vorgehens (siehe Kapitel 3.3). Beim modellbasierten Datentransfer mit Strukturumbau (siehe dazu 3.4) kommt dieses Element (E) nur an einer, der zentralen Stelle vor. Im Gegensatz dazu braucht es jedes der anderen Elemente (A) bis (D) jeweils an mindestens zwei Stellen, beim Startsystem und beim Zielsystem.

Mit semantischer (genauer: Semantik erhaltender) Transformation wird der Umbau von einer Start-Datenstruktur (beschrieben durch ein konzeptionelles Start Datenmodell) in eine Ziel-Datenstruktur (beschrieben durch ein konzeptionelles Ziel Datenmodell) beschrieben. Die Hauptidee dabei ist, dass der Umbau auf Modellebene definiert und der Umbau der Daten entsprechend dem definierten Modellumbau automatisch durch ein Werkzeug ausgeführt wird. Bedingung ist, dass die Startdaten im Standardformat vorliegen, das dem Start-Datenmodell entspricht.

Es gibt verschiedene solche Werkzeuge auf dem Markt, so die IG-Tools der Zürcher Firma InfoGrips [15], die Feature Manipulation Engine FME der kanadischen Firma Safe [16] und das INTERLIS Studio der Firma Geocom [17] aus Burgdorf. Die IG-Tools haben ein ausserordentlich reiches Sortiment von Umbaufunktionen zur Verfügung. Leider ist das Werkzeug nur mit einer recht umständlichen Skriptsprache konfigurierbar. INTERLIS Studio und FME haben sehr schöne grafische Benützeroberflächen zur Definition des Strukturumbaus. Dabei stellt FME wesentlich mehr Umbaufunktionen zur Verfügung als INTERLIS Studio. Die Grenzen von FME liegen beim Umbau von Beziehungen zwischen Klassen. Für alle Werkzeuge gilt – abgesehen von kostenwirksamer Lizenzpflicht – dieselbe Einschränkung, die schon beim 1:1 Prozessor auftrat: Sie haben keine Programmschnittstellen, um aus dem Demonstrator Pogramm aufgerufen zu werden. Es mussten daher im Rahmen des vorliegenden Forschungsprojekts eigene Programme für die benötigten semantischen Transformationen entwickelt werden.

Das Prinzip der semantischen Transformation soll am Beispiel des Umbaus der beiden verschiedenen Einzelfahrzeugzähler-Strukturen auf die vorgeschlagene Struktur von VM-CH erläutert werden. In beiden Fällen geht es um den Umbau einer einzelnen Klasse in eine einzige andere Klasse. In Abbildung 28 und Abbildung 29 sieht man, welche Umbaufunktionen dabei benötigt werden:

- Zuordnung eines Startattributes zu einem Zielattribut
- x2XMLDate: Datumsumbau auf XML Version
- x2XMLTime: Zeitattribute zusammenfassen und umbauen auf XML Version
- cm2mm: Länge in cm umwandeln in mm
- ms2kmh: Geschwindigkeit in m/s umwandeln in km/h
- etc.

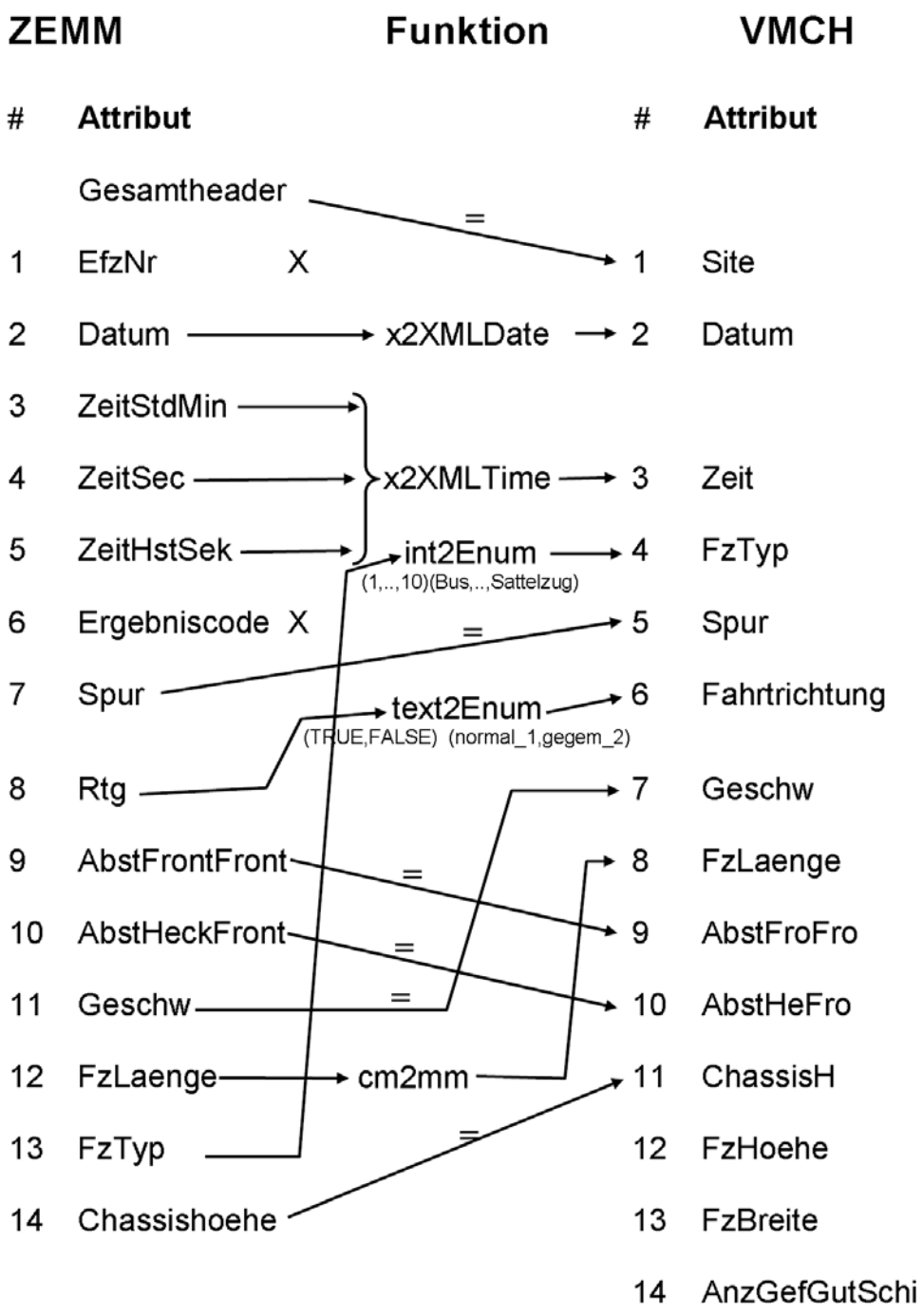


Abbildung 28: Semantische Transformation ZEMM nach VM-CH

Viel versprechend und auch textuell gut verständlich ist die generische Definitionsmöglichkeit für semantische Transformationen mit Hilfe von UMLT und ILIT, welche im Rahmen eines Forschungsprojektes TU München – ETH Zürich entwickelt wurde (siehe Kapitel 8.3 und [18], [19], [20]).

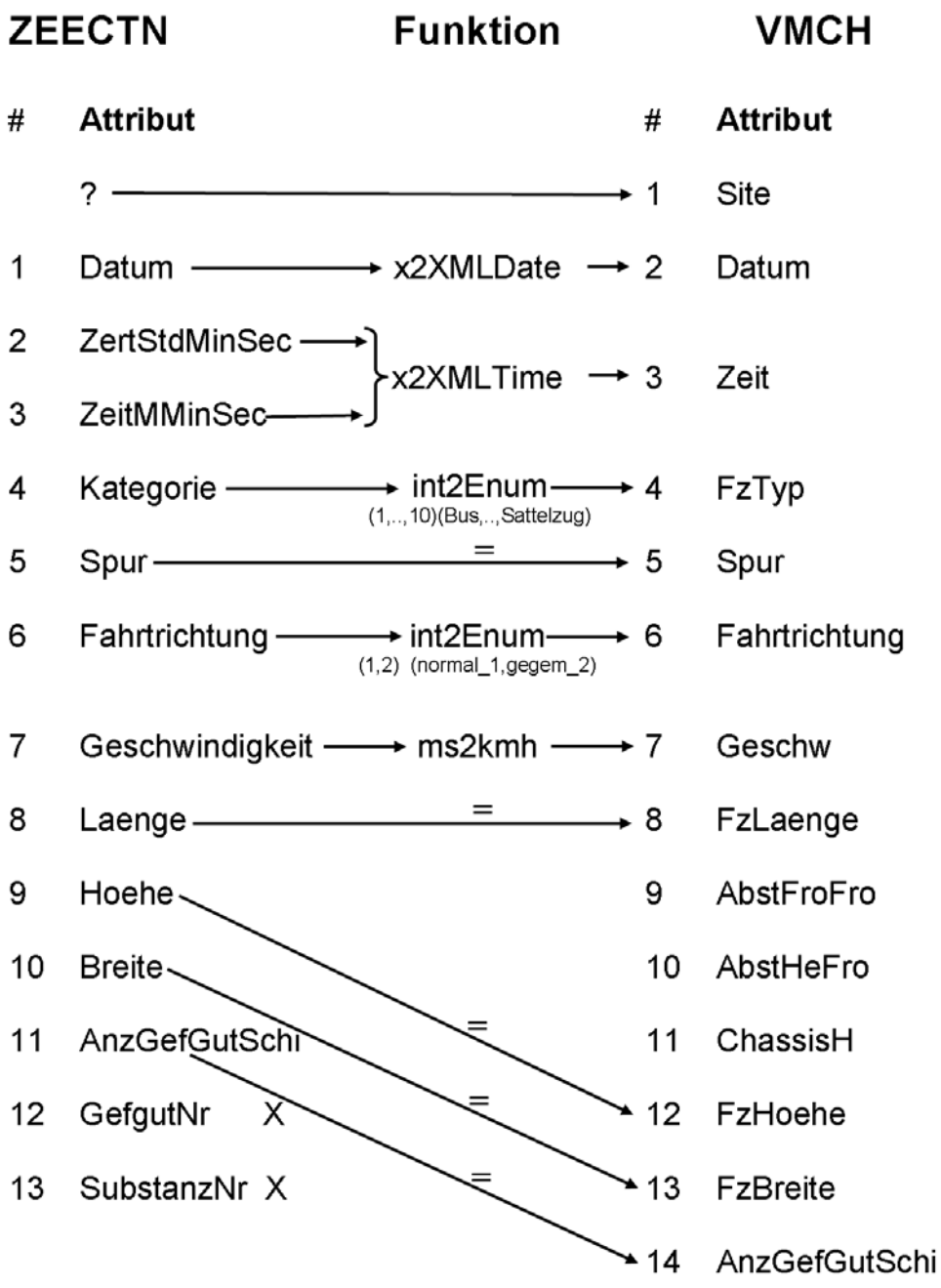


Abbildung 29: Semantische Transformation ZEECTN nach VM-CH

8.2. Starr programmierte semantische Transformation

Im Rahmen dieser Arbeit können zu Demozwecken nur diese beiden semantischen Transformationen realisiert werden, wie bereits erwähnt durch direkte Programmierung. Auch dabei konnte festgestellt werden, dass es sich lohnt, die Transformationsfunktionen zu isolieren und systematisch zu analysieren. Damit kommt man bei direkter Programmierung zu wieder verwendbarem Code.

Das Transformationsprogramm besteht aus den 3 Teilen

- Einlesen der Startdaten ab XML-Datei in interne Datenstruktur gemäss Start-Datenmodell fest programmiert.
- Umbau aus der internen Start-Datenstruktur in die interne Ziel-Datenstruktur gemäss einer der in 8.1 festgelegten semantischen Transformation. Für die Umrechnungen von einem oder mehreren Start-Attributswerten in einen oder mehrere Ziel-Attributswerte werden die vordefinierten Transformationsfunktionen eingesetzt. Der Umbau ist fest programmiert.
- Ausgabe der Zieldaten ab interner Datenstruktur in XML-Datei gemäss Ziel-Datenmodell fest programmiert.

8.3. Generische semantische Transformation mit UMLT

Bei generischer Programmierung von Werkzeugen für die semantische Transformation werden Transformationssprachen bzw. grafische Benutzeroberflächen zur Definition einer semantischen Transformation verwendet.

Die beiden benötigten semantischen Modelltransformationen (`ZaehlEfz → VMCH` sowie `ZaehlEfz_ECTN → VMCH`) sollen mit einem konzeptionellen Formalismus beschrieben werden. Dazu kann eine Transformationssprache angewendet werden, die in einem aktuellen Forschungsprojekt an der ETH Zürich entwickelt worden ist (siehe [18], [19], [20]). Die zweite der beiden Transformationen wird im Detail betrachtet und zwar zuerst deren grafische Form mit UMLT und dann deren Text in ILIT.

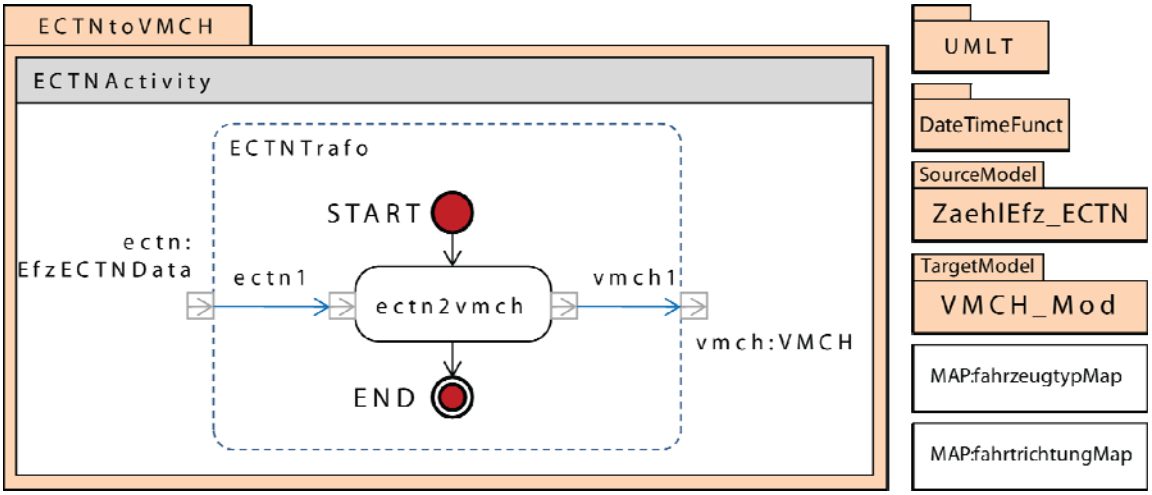


Abbildung 30: Modelltransformation `ZaehlEfz_ECTN → VMCH` in UMLT

```
INTERLIS 2.3;
MAPPING MODEL ECTNtoVMCH (de) AT "www.gis.ethz.ch" VERSION "0.1" =

IMPORTS UNQUALIFIED UMLT;            !! Basisfunktionen
IMPORTS UNQUALIFIED DateTimeFunct;  !! Formatierungsfunktionen
IMPORTS ZaehlEfz_ECTN;              !! Quellmodell
IMPORTS VMCH_Mod;                   !! Zielmodell

  TOPIC ECTNtoVMCH =

    ACTIVITY ECTNActivity =

      TRANSFORMATION ECTNTrafo =
        IN ectn: ZaehlEfz_ECTN.ZE_ECTN.EfzECTNData;
        OUT vmch: VMCH_Mod.VMCH_Top.VMCH;

        TRAFO_ACTION ectn2vmch =
          IN ectn1;
          OUT vmch1;

          MAP fahrzeugtypMap =
            1 -> "Bus";
            2 -> "Motorrad";
            3 -> "Personenwagen";
            4 -> "PWmitAnhaenger";
            5 -> "Lieferwagen";
            6 -> "LieferwagenMitAnhaenger";
            7 -> "LieferwagenMitAufleger";
            8 -> "Lastwagen";
            9 -> "Lastenzug";
            10 -> "Sattelzug";
          END fahrzeugtypMap;

          MAP fahrtrichtungMap =
            "TRUE" -> "normal_1";
            "FALSE" -> "gegen_2";
          END fahrtrichtungMap;

          MAPPING
            vmch1->Datum := Transformer.ECTNtoXmlDate(ectn1->Datum);
            vmch1->Zeit := Transformer.ECTNtoXmlTime(ectn1->ZeitStdMinSek,
                                                     ectn1->ZeitMilliSek);
            vmch1->FzTyp := ValueMapper.map(ectn1->Kategorie, fahrzeugtypMap);
            vmch1->Spur := ectn1 -> Spur;
            vmch1->FahrRtg := ValueMapper.map(ectn1->Fahrtrichtg, fahrtrichtungMap);
            vmch1->Geschw := ectn1->Geschwindkt*3.6;
            vmch1->FzLaenge := ectn1->Laenge;
            vmch1->FzHoehe := ectn1->Hoehe;
            vmch1->FzBreite := ectn1->Breite;
            vmch1->AnzGefGutSchi := ectn1->AnzGefGutSchi;

        END ectn2vmch;

        CONTROLFLOW
          START_FLOW --- ectn2vmch;
          ectn2vmch --- END_FLOW;

        DATAFLOW
          ectn --- ectn2vmch.ectn1;
          ectn2vmch.vmch1 --- vmch;

      END ECTNTrafo;  !! Transformation
    END ECTNActivity; !! of Activity
  END ECTNtoVMCH;       !! of Topic

END ECTNtoVMCH.
```

Abbildung 31: Modelltransformation `ZaehlEfz_ECTN` → `VMCH` als Text in ILIT

Die Transformationssprache „UMLT“ baut auf UML 2-Aktivitäten auf und erlaubt es, Modelltransformationen als Prozesse zu modellieren. Die Datenmodelle werden in INTERLIS 2.3 kodiert und die Transformationsmodelle in „ILIT“; der textuellen Repräsentation von UMLT [19]. Eine semantische Transformation wird als strukturierte Transformation, welche aus Transformations-Actions zusammengesetzt ist, gebildet. Die vorhergehende Seite enthält die Anwendung von UMLT auf den Umbau
`ZaehlEfz_ECTN` → `VMCH`

Zunächst werden im Transformationsmodell die beiden an der Transformation beteiligten Datenmodelle (Quellmodell gemäss Kapitel 5.1.5 bzw. Anhang A.3.5 und Zielmodell gemäss Kapitel 5.1.6 bzw. Anhang A.3.6) importiert. Daneben werden Funktionalitäten zur Verfügung gestellt, welche in der Transformation benötigt werden, beispielsweise eine Datumsformat-Konversion oder Wertezuweisungen. Diese Funktionen werden in einem gesonderten Modell beschrieben [7] wie folgt, damit die Funktionssignaturen angesprochen werden können:

```
INTERLIS 2.3;
MODEL DateTimeFunct (en) AT "http://www.gis.ethz.ch" VERSION "0.1" =
  TOPIC TDM =
    CLASS Transformer =
      FUNCTION MMtoXmlDate ( datum: NUMERIC ) : XMLDate;
      FUNCTION MMtoXmlTime ( hm: NUMERIC; sec: NUMERIC; csec: NUMERIC ) : XMLTime;
      FUNCTION ECTNtoXmlDate ( datum: TEXT ) : XMLDate;
      FUNCTION ECTNtoXmlTime ( hms: XMLTime; msec: NUMERIC) : XMLTime;
    END Transformer;
  END TDM;
END DateTimeFunct.
```

Abbildung 32: Zusatzmodell mit Umbaufunktionen

Danach wird die Transformations-Aktivität eingerichtet, wobei aus den Quell- und Zielobjekten Input- beziehungsweise Output-Pins definiert werden müssen. Damit wird der Prozess- und Objektfluss während der Transformation gesteuert.

Als nächstes folgen Wertezuweisungstabellen, so genannte „Maps“. Damit können beispielsweise Wertebereiche in Aufzähltypen abgebildet werden.

Schliesslich wird die eigentliche Objekttransformation im „Mapping“ definiert. Ein Mapping besteht immer aus einer Anzahl von Attributzuweisungen, wobei immer ein Zielattribut aus einer Funktion eines (oder mehrerer) Quellattribute gebildet wird. Allgemein:

$$\text{Zielattribut-Pin -> Zielattribut := f(Quellattribut-Pin}_i \text{ -> Quellattribut}_i\text{)}$$

Abbildung 33: Syntax der allgemeinen Umbauzeile

Ein UMLT-Prozessor prüft zunächst die syntaktische Richtigkeit der Modelle und führt die Transformation der Datenobjekte auf einer Datenbank aus. Die angesprochenen Funktionen müssen dafür gemäss Deklaration im Funktionenmodell im Prozessor implementiert sein.

9. Implementierung des Demonstratorprogramms

9.1. Übersicht der Programmklassen und Ablaufprinzip

Ausgehend von der aktuellen Systemstruktur, d.h. von den Systemen und deren Verbindungen, die wir im Beispielgebiet vorfanden, wurden als Konzept für den Demonstrator die Pogrammklassen und Datentransfer-Verbindungen gemäss Abbildung 26 zusammengestellt:

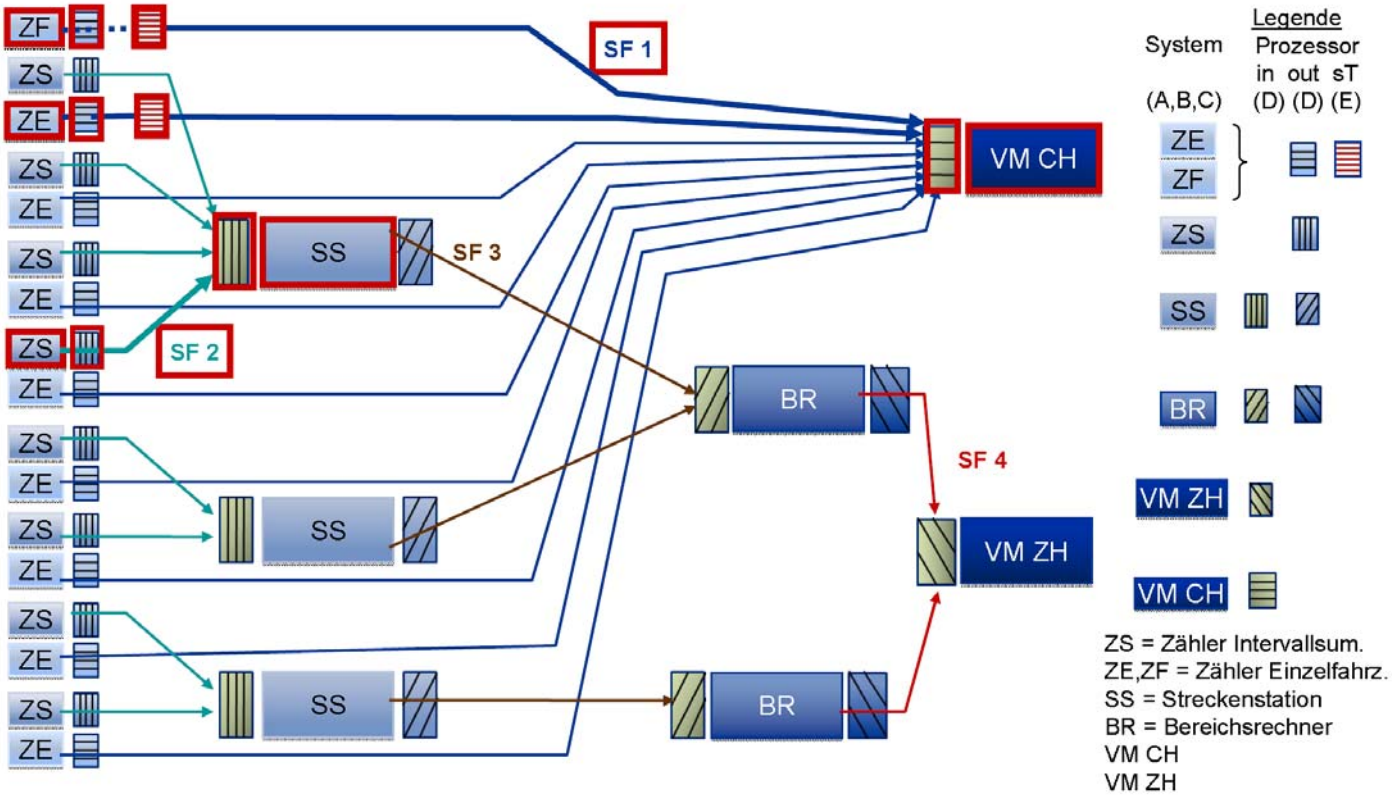


Abbildung 34: Programmklassen mit Datentransfer-Verbindungen
roter Rahmen: Im Demonstrator realisierte Systeme, 1:1 Prozessoren, Transformatoren
fette Verbindungslinien: Im Demonstrator realisierter Datentransfer

Ablaufprinzip: Jeder Zählerklasse am linken Rand von Abbildung 34 und auch jeder anderen Systemklasse ist eine Datei zugeordnet, welche die vom System gemessenen oder berechneten Werte enthält, und zwar in Form von abzugebenden Meldungen, mit Abgabezeitpunkt und im proprietären Format des Systems. Vom Hauptprogramm bekommen die Zähler einen Zeitimpuls und schicken ab ihrer Messdatei diejenigen Meldungen weg mit Meldungszeit kleiner als Impulszeit. Diese Meldungen werden – wenn nötig über einen Output 1:1 Prozessor und semantische Transformation – ans nächste System geschickt, bzw. an dessen Input 1:1 Prozessor und dann im proprietären Inputformat ans System. Von diesem System werden auf Grund der vom Vorgängersystem mit dessen Meldung übermittelten Meldungszeit alle Berechnungsmeldungen aus der eigenen Datei mit Berechnungszeit kleiner als eingetroffene Meldungszeit verschickt –

wieder via Output 1:1 Prozessor, ev. semantische Transformation und Input 1:1 Prozessor – ans nächste System, etc.

Um mit minimalem Aufwand alle wesentlichen Konzepte des modellbasierten Vorgehens (MDA) im lauffähigen Programm überprüfen zu können, wurden die in Abbildung 26 rot eingerahmten Programmklassen und die fett ausgezogenen Datentransfers realisiert.

- Zu den MDA Elementen (A) Realitätsausschnitt, (B) konzeptionelles Datenmodell und (C) entsprechendes Standard Transferformat wurden implementiert die Systemklassen
  - Einzelfahrzeugzähler ECTN (ZF, siehe 4.4.5, 5.1.5),
  - Einzelfahrzeugzähler Marksman 660 (ZE, 4.4.4, 5.1.4 Variante 2),
  - Intervallsummenzähler Marksman 660 (ZS, 4.4.1, 5.1.1 Variante 4),
  - Streckenstation SAIA_SPS PCD3 (SS, nur Meldungsempfänger, gleiches Input-Format wie Output-Format von ZS),
  - VM-CH Rechner (VM-CH, nur Meldungsempfänger, harmonisierter Input-Format 5.1.6)
- Zum MDA Elementen (D) 1:1 Prozessor die Prozessorklassen
  - Output 1:1 Prozessor (fix) zum Einzelfahrzeugzähler ECTN (ZF)
  - Output 1:1 Prozessor (fix) zum Einzelfahrzeugzähler Marksman 660 (ZE),
  - Output 1:1 Prozessor (generisch) zum Intervallsummenzähler Marksman 660 (ZS),
  - Input 1:1 Prozessor (generisch) zur Streckenstation SAIA_SPS PCD3 (SS),
  - Input 1:1 Prozessor (fix) zum VM-CH Rechner (VM-CH)
- Zum MDA Elementen (E) semantische (bzw. Semantik erhaltende) Transformation
  - Strukturumbau Einzelfahrzeugzähler ECTN nach VM-CH (ZF → VM-CH)
  - Strukturumbau Einzelfahrzeugzähler Marksman 660 nach VM-CH (ZE → VM-CH)

Im Folgenden einige Details zu den realisierten Programmklassen.

9.1.1. Implementierung von Systemen Stufe Einzelsteuerung

Es wurden drei Zählertypen implementiert als Programmklassen: Zwei Zählertypen für Einzelfahrzeuge (Marksman 660 und ECTN) sowie einer für Intervallsummen (ebenfalls Marksman 660). Die entsprechenden Programme sind so realisiert, dass sie jeweils auf eine Datei mit Messdaten zugreifen und diese zeilenweise einlesen. Jede Zeile beinhaltet in diesem Stadium Daten im proprietären Format und wird deshalb an den zugehörigen Output 1:1-Prozessor des Zählers weitergeleitet. Allfällige Header-Informationen werden im Hinblick auf das zu erzeugende Standardformat ebenfalls an den Ausgabe 1:1 Prozessor übergeben.

9.1.2. Implementierung 1:1-Prozessoren

Es gibt 1:1-Prozessoren auf der Output Seite der Systeme und solche auf der Input Seite.

Auf der Output Seite verarbeiten die 1:1-Prozessoren die von der Systemklasse des Zählers (bzw. des allgemeinen Systems) gelieferte, im proprietären Format vorliegende Zeile und wandeln sie in ein XML-Objekt um, welches passend zur zugehörigen INTERLIS-(Datenmodell-)Klasse strukturiert sein muss. Effektiv wird an dieser Stelle also ein Objekt erzeugt, welches dieselbe INTERLIS Klasse zur Grundlage hat, wie die proprietären Startdaten. Dieses Objekt wir nun an die Klasse weitergeleitet, welche den Strukturumbau (semantische Transformation) vornimmt.

Es wurden drei Output 1:1 Prozessoren programmiert, zwei starre für die beiden Einzelfahrzeugzähler (ZE und ZF in Abbildung 34, sie produzieren aus den Proprietär formatierten Daten von ZE und ZF zwei verschiedene Standardformate entsprechend den verschiedenen Datenmodellen für ZE und ZF) sowie ein generischer für den Intervallsummen-Zähler (ZS in Abbildung 34, er produziert den Standardformat SF2).

Alle 1:1 Prozessoren übermitteln an ihr Zielsystem meldungsbasiert. Die beiden starren 1:1 Prozessoren produzieren auch noch eine vollständige Transferdatei mit allen Daten, die sie von ihren Zählersystemen erhalten. Diese Transferdateien können gegenüber dem entsprechenden Datenmodell durch einen Checker automatisch geprüft werden.

Der 1:1-Prozessor auf der Input Seite eines Zielsystems übernimmt im XML Standardformat die Daten, die vom Vorgängersystem kommen und allenfalls semantisch transformiert worden sind auf das Datenmodell des Zielsystems. Diese Daten im Standardformat werden umgebaut auf das proprietäre Format des Zielsystems und dann an die Programm-Klasse des Zielsystems weitergeleitet.

Es wurde ein generischer Input 1:1 Prozessor programmiert für die Streckenstation (SS in Abbildung 34). Er nimmt den Standardformat SF2 entgegen und produziert daraus das proprietären Format der Streckenstation.

9.1.3. Implementierung Semantische Transformation

Es wurden die zwei in Kapitel 9.1. vorgestellten semantischen Transformationen von den beiden Typen der Einzelfahrzeugzähler auf die vorgeschlagene harmonisierte VM-CH Struktur programmiert. Das erfolgt einerseits starr, nach dem in Kapitel 8.2 erläuterten Vorgehen, andererseits wird der konzeptionell mit UMLT beschriebene Umbau von Kapitel 8.3 auf dem Prototyp des Forschungsprojektes TU München – ETH Zürich [20] implementiert.

Hier wäre auch die Koordinatentransformation einzubauen, sobald klar ist, wie die kurvilinearen Koordinatensysteme der SVT konzeptionell exakt beschrieben werden können.

9.1.4. Implementierung eines Systems Stufe Gruppensteuerungsebene

Die Programmklassen für die Streckenstationen wurden implementiert, allerdings im Moment nur als „Protokollführer“ für die vom Input 1:1-Prozessor erhaltenen Daten im proprietären Streckenstationsformat, d.h. diese werden zur Kontrolle ausgedruckt.

9.1.5. Implementierung eines Systems Stufe Übergeordnete Leitebene

Auf Stufe der übergeordneten Leitebene wurde die Klasse für das System VM-CH implementiert, allerdings auch nur zum Ausdruck der Daten im proprietären VM-CH Format, die vom Input 1:1 Prozessor des VM_CH Systems aus den Standarddaten der semantischen Transformation berechnet werden.

9.2. Ergebnisse der Tests

Die Implementierung des Demonstratorprogramms zeigte:

- Hauptergebnis: In der gegebenen SVT-Systemstruktur ist mit dem modellbasierten Vorgehen effizient Datentransfer mit Strukturumbau möglich unter Einbezug der bestehenden Systeme und ohne Eingriff in dieselben.
- Mit den Programmklassen für die verschiedenen Systeme und der Realisierung der Kommunikation zwischen diesen mit Hilfe der objektorientierten Meldungen ist ein modifizierbarer Testrahmen entstanden, mit dem bei Bedarf auch neue Kommunikationsnetzstrukturen simuliert werden können.
- 1:1 Prozessoren sind einfach zu programmieren. Zum Schreiben und lesen von XML Transferdaten stehen umfangreiche Java Pakete zur Verfügung und zur Bearbeitung von csv Dateien gibt es sehr viele Textbearbeitungsmethoden.
- XML allgemein als Transferformat ist zwar sehr „wortreich“ aber hat wesentliche Vorteile. Vor allem seine Internet-Tauglichkeit aber auch die eben erwähnte Vielfalt an existierenden Software Werkzeugen.
- INTERLIS 2 XML ist zusätzlich sehr wertvoll, denn es eröffnet u.a. den Zugang zum Checker und zur semantischen Transformation, d.h. zu automatischer Qualitätsprüfung und zu Strukturumbau auf konzeptioneller, systemneutraler Ebene.
- Das CASE Werkzeug Eclipse erlaubt unglaublich effizientes Programmieren.
- Aus verschiedenen Gründen (siehe 8.1) wurde die semantische Transformation explizit programmiert. Für umfangreichere Strukturumbauten ist die benutzerfreundliche Formulierung mit ILIT und vor allem auch ein übersichtlicher grafischer Zugang mit UMLT unabdingbar. Entsprechende Werkzeuge sollten rasch zu Produktionsreife gebracht werden.
- Ein Problem bot die Schwierigkeit, zu Testdaten zu kommen. Nicht zuletzt deshalb wurde auf eine Implementierung der Systeme Streckenstation, Bereichsrechner VDE und VM-ZH verzichtet. Denn dafür waren keine vom entsprechenden System produzierte Daten vorhanden. Nur zum Einzelfahrzeugzähler ECTN gab es neben dem kleinen Satz von Daten zur Erklärung des proprietären Transferformats noch einige grössere Testdateien.

9.3. Checker Einsatz nach 1:1 Prozessor und semantischer Transformation

Der Checker Einsatz ist zweckmässig nach dem Formatumbau durch einen 1:1 Prozessor und nach dem Strukturumbau durch semantische Transformation. An beiden Stellen hat man ein konzeptionelles Datenmodell und Daten im entsprechenden Standardformat. Da auch der Checker zurzeit noch nicht mit Programmschnittstelle verfügbar ist, konnte er nicht direkt an der entsprechenden Stelle des Demonstrator Programms aufgerufen werden. Um trotzdem die automatische Qualitätsprüfung mindestens für die Daten der Zählersysteme durchzuführen, wird durch die Output 1:1 Prozessoren aus den Meldungen ans nächste System im Standardformat auch eine Datei erstellt. Diese kann offline, d.h. ausserhalb des Demonstrators, durch den Checker mit dem Datenmodell verglichen werden.

Die beiden Testdateien (siehe 5.1.1 und 5.1.6) erwiesen sich als fast fehlerfrei, ausser dass bei der Messung Nr. 14 (von 28) des ECTN Zählers als swiss10-Code der unzulässige Wert 11 gefunden wurde.

9.4. Platzierung der Software (1:1 Prozessor und semantische Transformation) in der Realwelt

Für die Anordnung der Konversions-Software (1:1 Prozessor + ev. semantische Transformation) gibt es zwei Möglichkeiten:

1) Man übernimmt das aktuelle Netzwerk der Systeme (Verkehrszähler, Streckenstationen etc) und implementiert die Konversionsklassen als „add-on“ zu den bestehenden Systemen. Die Kommunikation (Datenaustausch) erfolgt über die heutige Netzwerkstruktur.
2) Die Konversionsklassen werden zwischen die Systemklassen geschoben und organisieren den Datenaustausch selbst. Die Konversionsklassen können dann auch unmittelbar vor den Empfängersystemen angeordnet sein, jedenfalls dort, wo mehr Rechenkapazität zur Verfügung steht.

In der Realwelt ist die Platzierung der Komponenten abhängig von den einzelnen Systemen. Auch ist zu beachten, welche modellbasierten Dienste (s. Kapitel 3.4) wie z.B. nachhaltige Datensicherung, Qualitätssicherung etc. für einzelne Systemfunktionalitäten genutzt werden können. In der Regel wird aber die zweite Möglichkeit, die auch beim Demonstrator verwendet wird (2), in den meisten Systemen zum Einsatz kommen können.

9.5. Umfangreichere Modelle und Daten der Realwelt als diejenigen der Testimplementierung

Im Rahmen dieses Forschungsprojektes wurde als Realitätsausschnitt das Beispiel eines Verkehrsdatenerfassungssystems mit mässiger Komplexität gewählt. In der Realität gibt es weitaus umfangreichere Modelle und Daten, wie sie zum Beispiel im Rahmen von Verkehrsmanagement und Betriebs- und Sicherheitsausrüstung notwendig sind. Das modellbasierte Vorgehen kann mit den beschriebenen Methoden aber auch darauf angewendet werden.

10. Schlussfolgerungen

Der Vergleich der Ergebnisse dieser Arbeit mit den gesetzten Zielen erlaubt folgende Feststellungen:

Z1: „Erarbeiten von Lösungsmöglichkeiten für Schnittstellenspezifikation und Datenflussorganisation mit Hilfe des modellbasierten Vorgehens entsprechend den Bedürfnissen von Strassendaten allgemein und der SVT im Besonderen (z.B. Real Time Aspekte, spezielle Datentypen)“.

Die in Kapitel 3 entwickelte Methode des modellbasierten Vorgehens ermöglichte die erfolgreiche Analyse sowie einheitliche, präzise und vergleichbare Spezifikationen der Daten von Schnittstellen verschiedensten Typs durch Datenmodelle (Kapitel 4 und 5). Darauf aufbauend konnte am Beispiel von zwei verschiedenen Einzelfahrzeugzählern gezeigt werden, wie eine harmonisierte übergeordnete Datenstruktur erreicht werden kann mit den beiden Extremen einer „fetten“ bzw. „mageren“ Ausprägung (in Kapitel 5.1.6). Durch das Datenmodell und den Standard-Transferformat, der eindeutig und automatisch aus dem Datenmodell herleitbar ist, wird der Zugang zu system-neutralen Werkzeugen eröffnet, z.B. für automatische Datenprüfung (Checker) und für semantikerhaltenden Strukturumbau (semantische Transformation, Kapitel 8). Es konnte auch gezeigt werden, dass der notwendige Übergang vom aktuellen proprietären Transferformat auf dieses Standardformat mit einfachen Mitteln möglich ist (1:1 Prozessoren, Kapitel 7). Als Defizit mussten wir feststellen, dass zurzeit einfach bedienbare und von Programmen aufrufbare Werkzeuge für die semantische Transformation erst im Aufbau begriffen sind (siehe Kapitel 8.1, 8.3 und 9.1.3).

Z2: „Aufzeigen, wie die Forschungsergebnisse auf weitere SVT-Systeme anwendbar sind.“

Das modellbasierte Vorgehen kann überall dort erfolgreich zum Einsatz kommen, wo es darum geht, Daten zwischen heterogen strukturierten Systemen / Schnittstellen auszutauschen. Hier kann nur darauf hingewiesen werden, dass im Einsatzgebiet des Verkehrmanagements sicher die Aktorensteuerung ein Anwendungsgebiet wäre, und dass im weiteren Sensorumfeld auch die Schnittstellen im Zusammenhang mit Ereignisdetektion ins Anwendungsblickfeld rücken.

Z3: „Realisierung und Implementierung eines Demonstrators mit den erarbeiteten Lösungen für ein Praxisbeispiel.“

Siehe Kapitel 9.

Z4: „Aufzeigen der Möglichkeiten und Grenzen des modellbasierten Vorgehens zum Schaffen der datentechnischen Voraussetzungen für automatische Koordinatentransformationen zwischen verschiedenen in der SVT gebräuchlichen Referenzsystemen.“

Die Betrachtung der datentechnischen Voraussetzungen für die automatische Koordinatentransformation wurde aufgrund Empfehlung der EK 9.03 zugunsten des Demonstrators weggelassen. Da gerade die kurvilinearen Koordinatensysteme aus dem Strassenbereich einer Präzisierung bedürfen und im Bereich Koordinatensysteme und Transformationen im INTERLIS Umfeld wesentliche Vorarbeiten existieren (Refe-

renzhandbuch [7] Anhang J), sollte dieser Punkt in den folgenden Schritten näher betrachtet werden.

Z5: „Die Resultate sollen als Grundlage verwendet werden können für die Realisierung des Zielzustandes VM-CH 2012 (hochautomatischer und multimodaler Datenaustausch auf und zwischen internationaler, nationaler und regionaler Ebene).“

Das ist nach Beurteilung der Forschungsstelle gegeben, allerdings mit der Einschränkung, dass für die semantische Transformation brauchbarere Werkzeuge zur Verfügung stehen müssen. Voraussetzung ist, dass die Grundlagen des modellbasierten Vorgehens von allen Beteiligten anerkannt werden, nämlich die normative Festlegung einer praxistauglichen konzeptionellen Beschreibungssprache (wie z.B. UML – INTERLIS 2) und eindeutiger Codierungsregeln für die automatische Herleitung des Standard-Transferformats aus dem Datenmodell.

Es stellte sich zu Beginn dieser Arbeit unter anderem die Frage, ob mit dem Einsatz des modellbasierten Vorgehens in der Strassenverkehrstelematik nicht mit „Kanonen auf Spatzen geschossen werde“. Schon beim jetzigen Stand an Auswertungen und Dokumentation kann die Frage verneint werden. Zunächst sind einige unmittelbare Vorteile des modellbasierten Vorgehens ersichtlich:

- Die präzise Schnittstellenbeschreibung erweist sich bei verschiedenartigen Systemen, die denselben Empfänger bedienen, als wesentliche Voraussetzung für die Harmonisierung eines übergeordneten Grundmodells.
- Die klare Abgrenzung von 1:1 Prozessoren und semantischer Transformation vereinfacht in beiden Bereichen die Ausarbeitung und Mehrfachnutzung von Basisfunktionskomponenten,
- Der Ersatz proprietärer Formate durch modellbasierte Standardformate ohne Informationsverlust ermöglicht den Einsatz system-, d.h. schnittstellen-neutraler Werkzeuge u.a. für Datenprüfung und semantische Transformation.
- Die Arbeit mit Datenmodellen vereinfacht die Übersicht (grafisches UML-Diagramm), ohne auf Präzision verzichten zu müssen (textuellen INTERLIS 2 im Hintergrund) und legt Kapazitäten frei zur Analyse und Lösung von Problemen, die unter Datenhaufen verschüttetet sind.

Empfehlungen für das weitere Vorgehen:

- Die modellbasierte Analyse verschiedener Schnittstellen zeigte, dass an mehreren Stellen sowohl dateibasierter als auch meldungsbasierter Transfer möglich ist, was die beteiligten Datenstrukturen betrifft. Es stellt sich die Frage, ob dadurch allenfalls neue, intensivere Nutzungen bestehender Datennetze möglich wären oder ob es andere kritische Randbedingungen gibt, welche eine der beiden Möglichkeiten ausschliessen, und ob allenfalls einfachere Datennetze zweckmässiger wären. Diese Punkte sollten im Rahmen von weiterführenden Untersuchungen analysiert werden. Die entwickelt Rahmensoftware ist flexibel und geeignet zur Prüfung von Varianten.
- Ausbau des Demonstrators: Der hierarchische Transfer vom Summenzähler über Streckenstation und Bereichsrechner VDE bis zum VM-ZH ist nicht nur für die erste Stufe und für einen Zähler zu realisieren, sondern mit realistischen Messungs- und Berechnungsdateien über alle Hierarchiestufen. In einem weiteren Schritt können die Dateien mit den Berechnungsmeldungen ersetzt werden durch

Algorithmen, welche die Berechnungsmeldungen produzieren. Als letzte Phase könnten auch nur noch Einzelfahrzeugmessungen herumgeschickt werden, aber an alle Systeme, und jedes System berechnet sich unabhängig von allen anderen Systemen, was es braucht.

- Die zurückgestellte Koordinaten-Transformation sollte unbedingt auch aus der Sicht der konzeptionellen Modellierung bearbeitet werden. Dabei sind zwei Schritte nötig für Realisierung und Einsatz, z.B. im geplanten Forschungsprojekt „validation expérimentale du modèle de repérage spatial pour les services télématiques“
  - Klare konzeptionelle Definition der kurvilinearen Koordinatensysteme
  - Implementierung weniger wesentlicher Koordinatentransformationen in der Software für semantische Transformation. Am interessantesten dürften sein die Transformation vom RBBS nach Landeskoordinaten und zurück.
- Die Ergebnisse der starren semantischen Transformation (Kapitel 8.2) sind im Detail zu vergleichen mit den Ergebnissen der generischen Definition (Kapitel 8.3) und es ist abzuklären, wie und wo die jeweils nötige ausführbare Software in den aktuellen Netzen der SVT implementiert werden kann.
- Ein Ausbau der Varianten an Messgeräten könnte zu anderer Harmonisierung übergeordneter Zieldatenmodelle führen und als Folge davon zu anderen Austauschformaten.
- Ferner sollte auch der Vergleich des automatisch aus dem Datenmodell hergeleiteten INTERLI 2 XML-Schemas mit dem Original XML-Schema des proprietären Formats für den Bereichsrechner VDE vorgenommen werden. Dabei wären folgende Fragen zu beantworten:
  - Wo sind Unterschiede?
  - Sind sie wesentlich?
  - Was spricht gegen die Anwendung des modellbasierten Vorgehens schon zu Beginn, d.h. beim Entschluss, Daten in einem XML-Format zu transferieren? Dagegen nämlich, zuerst die Datenstruktur exakt zu beschreiben mit einem Datenmodell und daraus die Formatbeschreibung in XML-Schema automatisch herzuleiten?
- Anwendung der Erkenntnisse dieser Forschungsarbeit auf weitere Gebiete der SVT (z.B. Ereignisdetektion, Aktorsteuerung, etc.).
- Erarbeitung einer Norm für Zieldatenmodelle auf den verschiedenen Ebenen der Leittechnik basierend auf der Harmonisierung der Ausgangsdatenmodelle. Damit stehen auch automatisch harmonisierte Austauschformate zur Verfügung.

11. Referenzen

11.1. Experten, die beim Projekt mitwirkten

Dieses Projekt wurde realisiert durch die Gruppe GIS und Fehlertheorie des Instituts für Geodäsie und Photogrammetrie der ETH Zürich, durch die Ingenieurfirma AWK Group in Zürich Oerlikon und durch die Eisenhut Informatik AG in Burgdorf. Die Forschungsstelle bestand aus folgenden Personen:

- Die Herren H. R. Gnägi und S. Henrich von ETHZ IGP GF
- Frau M. Münster und Herr R. Rüegg von AWK Group
- Herr C. Eisenhut von Eisenhut Informatik AG

Das Projekt wurde ferner von folgenden Personen unterstützt

- Herr P. Y. Gilliéron vom Laboratoire de Topométrie der EPF Lausanne
- Herr E. Bernard von Infolite AG, Bern
- Herr C. Marschal vom Ingenieurbüro Rosenthaler und Partner, Muttenz

Die Kommission EK 9.03 Strasseninformationstelematik hat diese Forschung begleitet und ihre Mitglieder haben mit ihren Kompetenzen während der ganzen Projektdauer wesentliche Beiträge geleistet.

11.2. Bibliographie

Titel	Autor / Herausgeber	Datum	Version
[1] Konzept AKS-ZH	Electrowatt-Ekono AG	Februar 2005	2.1
[2] Zählstellen-Anschlusslösungen	ASTRA	Juni 2007	1.1
[3] SN 671951 – Funktionale Systemarchitektur	VSS	2008	Entwurf
[4] AGRAM Forschungsbericht	VSS	2003	Bericht
[5] Verifikation Grobkonzept Zielzustand Systemarchitektur VM-CH	AWK	September 2007	V1.3
[6] Unified Modelling Language (UML)	OMG		
[7] INTERLIS 2 Referenzhandbuch	KOGIS, www.interlis.ch		2.3
[8] SN 612031 INTERLIS 2 Modellierung und Datentransfermethode	SNV, www.snv.ch		
[9] ISO 19100 Geographic information – Geomatics, Normenserie	ISO/TC 211 www.isotc211.org		
[10] UML-INTERLIS Editor (freeware und open source)	KOGIS, www.interlis.ch		
[11] INTERLIS Compiler (freeware und opensource)	KOGIS, www.interlis.ch		
[12] ISO 19118 GI - Encoding	ISO/TC 211 www.isotc211.org		
[13] INTERLIS 2 Checker (freeware in CH und Liechtenstein)	KOGIS, www.interlis.ch		
[14] DIN V 19222 Leittechnik – Begriffe	DIN – Deutsches Institut für Normung	September 2001	4.1.2
[15] InfoGrips Tools (IG-Tools)	InfoGrips GmbH, www.infogrips.ch		
[16] FME (Feature Manipulation Engine)	Safe Software, Kanada, www.safe.ca		
[17] INTERLIS Studio	Geocom AG Burgdorf, www.geocom.ch		
[18] A Model-Driven Web Feature Service for Enhanced Semantic Interoperability	Peter Staub: OSGeo Journal 1(3):38—43	Dezember 2007	
[19] Semantic Interoperability through the Definition of Conceptual Model Transformations.	Peter Staub, Hans Rudolf Gnägi und Andreas Morf: Transactions in GIS 12(2):193—207	April 2008	

[20] Modellbasierter Ansatz für den Web-Zugriff auf verteilte Geodaten am Beispiel grenzübergreifender GIS-Anwendungen.	Andreas Donaubauer, Astrid Fichtinger, Matthäus Schilcher, Florian Straub, Alessandro Carosio, Hans Rudolf Gnägi, Andreas Morf und Peter Staub: Projektbericht, ETH Zürich/TU München	2006/2007	
[21] SN671941 Strassenverkehrstelematik – Referenzierung für Verkehrsdaten und Verkehrsinformationen	VSS	Oktober 2008	Entwurf
[22] The AWK Programming Language	A.V. Aho, B.W. Kerningham, G.H. Weinberger,	2008	Buch, Addison Wesley

12. Abkürzungen und Terminologie

12.1. Abkürzungen de Umgangssprache

(fachtechnische Abkürzungen siehe Kapitel A.5.2)

- Abk. Abkürzung.
- Art. Artikel (in Gesetzestexten).
- Abs. Absatz (in Gesetzestexten).
- Def. Definition.
- de deutsch.
- en english.
- fr français.
- Syn. Synonym.
- → *A* *A* ist ein Begriff, der in diesem Glossar 12.3 definiert ist.

12.2. Technische Abkürzungen

Abkürzung	Begriff
AKS	Anlagen-Kennzeichnungs-System
CAD	Computer Aided Design, → *Layer*
CSL	Conceptual Schema Language, → *Datenbeschreibungssprache*
csv	Comma separated values, Formattyp
DCOM	Distributed Component Object Model
DDL	Data Description Language, → *Datenbeschreibungssprache*
ECTN	Enterprise Corporated Technology Netword, Firmenname
GIS	Goeinformationssystem oder geografisches Informationssystem
IDDL	INTERLIS Data Description Language, → *INTERLIS Datenbeschreibungssprache*
INTERLIS	INTER Land Informations System, d.h. zwischen den GIS → *INTERLIS*
IXML	→ *INTERLIS 2* → *XML* → *Transferformat*
MDA	Model Driven Approach, modellbasiertes Vorgehen (siehe Kapitel 3)
OID	→ *Objektidentifikator*
OLE	Object Linking and Embedding
OO	Objektorientiert, Objektorientierung
OPC	→ *OLE for Process Control*
OPC AE	→ *OPC for Alarms and Events*
OPC DA	→ *OPC for Data Access*
RBBS	Räumliches → *Basisbezugssystem*
SN	Schweizer Norm
SPS	Speicherprorammierbare Steuerung

Abkürzung	Begriff
SRB	Système de Repérage de Base (Syn. von → *Basisbezugssystem*)
SVT	Strassen Verkehrs Telematik
TID	→ *Transferidentifikation*
TLS	Technische Lieferbedingungen für → *Streckenstationen*
UML	→ *Unified Modelling Language*
XML	→ *Extensible Markup Language*

12.3. Terminologie

Begriff	Definition
Abbildung	(aus einer Menge A in eine Menge Z:) Vorschrift, die einem Element a aus A genau ein Element z aus Z zuordnet. Bemerkung: Mit einer besonderen A. hat man zu tun, wenn die Menge A und die Menge Z je ein → *Raum* ist definiert durch ein → *Koordinatensystem*; → *Koordinatentransformation*
Aggregation	Gerichtete → eigentliche Beziehung zwischen einer übergeordneten → Klasse und einer untergeordneten → Klasse. Einem Ganzen (Ober-Objekt der übergeordneten → Klasse) sind mehrere Teile (Unter-Objekte) der untergeordneten → Klasse zugeordnet. Einem Teil können auch mehrere Ganze zugeordnet sein. Beim Kopieren eines Ganzen werden alle zugeordneten Teile mitkopiert. Beim Löschen eines Ganzen können alle zugeordneten Teile weiter existieren. Bemerkung 1: Mit Hilfe der A. wird die → Beziehung zwischen einem Ganzen und seinen Teilen beschrieben (z.B. Auto/Motor). Die → Rolle der → Unterklasse kann bezeichnet werden mit "ist-Teil-von", "is-part-of" (en). Bemerkung 2: In → INTERLIS 2 wird die A. in Analogie zur → UML-Klassendiagramm-Notation mit einem (leeren) Rhombus (-<>) angegeben. Bemerkung 3: Siehe auch → Komposition.
Assoziation	→ *eigentliche Beziehung*, welche die Unabhängigkeit der beteiligten → *Klassen* nicht einschränkt. Die zugeordneten → *Objekte* können unabhängig voneinander kopiert und gelöscht werden. Eine A. heisst bidirektional, wenn sie beidseitig direkt navigierbar ist. Syn. association (en, fr). Bemerkung 1: In → *INTERLIS 2* steht für die Beschreibung der A. die → *Assoziationsklasse* zur Verfügung. Bemerkung 2: Siehe auch → *Referenzattribut*, → *Aggregation* und → *Komposition.*
Attribut	→ *Daten(elemente)* entsprechend einer spezifischen Eigenschaft von → *Objekten* einer → *Klasse*. Ein A. hat einen Attributsnamen und einen → *Wertebereich.* Syn. Merkmal (de), attribute (en). Bemerkung: Jedes → *Objekt* einer → *Klasse* enthält für jedes A. einen individuellen → *Wert, d.h.* ein → *Datenelement* aus

Begriff	Definition
	dem → *Wertebereich* des A.. Anschaulich entspricht ein A. der Kolonne einer → *Tabelle*.
Basisbezugssystem	→ *Bezugssystem* mit 3-dimensionalem → *Referenzbereich* und → *kurvilinearem* → *Koordinatensystem.* Syn.: Système de repérage de base (fr CH), räumliches Basisbezugssystem (de CH) Abk.: SRB (fr CH), RBBS (de CH) Bemerkung: Vergleiche dazu AGRAM Bericht S. 1 ff und SN 640910.
Behälter	Menge von → *Objekten*, die zu einem → *Thema* oder zu dessen → *Erweiterungen* gehören. Syn. basket (en).
Benutzerschnittstelle	Bedienungsoberfläche eines Computerprogramms. Syn. (mehrdeutig!) Schnittstelle (de),graphic user interface (en) Bemerkung: Siehe auch → *Klassenschnittstelle* und → *Datenschnittstelle*.
Beziehung	Menge von Objektpaaren (bzw. im allgemeinen Fall von Objekt-n-Tupeln, die auch Beziehungsobjekte heissen). Das erste → *Objekt* jedes Paares gehört zu einer ersten → *Klasse* A, das zweite zu einer zweiten → *Klasse* B. Dabei soll die Zuordnung von → *Objekten* zu den Paaren vorgegeben sein, sie muss also nur beschrieben, d.h. modelliert werden. Man unterscheidet → *eigentliche B.* (nämlich → *Assoziation,* → *Aggregation,* → *Komposition*), → *Vererbungsbeziehung* und → *Referenzattribut*. Syn. relationship (en); relation (fr). Bemerkung 1: Wie das Sichten-Konzept zeigt, ist es im Gegensatz dazu auch möglich, Zuordnungen algorithmisch z.B. aufgrund von Attributwerten zu berechnen. Bemerkung 2: Siehe auch → *Objektbeziehung*. Bemerkung 3: In einer eigentlichen B. ist fürjede beteiligte → *Klasse* ihre → *Rolle* definiert.
Beziehungsobjekt	Paar (bzw. im allgemeinen Fall n-Tupel) von → *Objekten*, die einander durch eine → *Beziehung* zugeordnet sind. Gleichwertige Def.: Zwei (bzw. im allgemeinen Fall n) → *Objekte,* die einander zugeordnet sind durch eine → *Beziehung* zwischen den → *Klassen,* denen sie angehören. Syn. Objektbeziehung (de), link (en)
Bezugssystem	→ *Referenzbereich* mit → *Koordinatensystem.* Syn.: Raumbezugssystem Bemerkung: Siehe → *lineares B.* → *flächiges B.,* → *Raumbezugssystem*
Datei	Def. siehe Informatik. Syn. file (en), fichier (fr).
Daten	Def. siehe Informatik. Syn. data (en), données (fr).
Datenaustausch	→ *Datentransfer* von → *System* A nach → *System* Z und umgekehrt.
Datenbeschreibung	Mehrdeutiges Syn. für → *Datenschema* und → *Datenmodell.*

Begriff	Definition
Datenbeschreibungssprache	Formale Sprache zur exakten Beschreibung von Daten. Syn. Data description language (DDL), conceptual schema language (CSL) Abk.: DDL, CSL. Bemerkung: Beispiele für (konzeptionelle) D. sind das → *UML*-Klassendiagramm (grafisch) und → *INTERLIS* (textuell).
Datenelement	Def. siehe Informatik. Vergleiche dazu → *Wertebereich*.
Datenformat	Syn. von → *Transferformat*
Datenmodell	Exakte Beschreibung von Daten (so genanntes → *konzeptionelles* → *Datenschema*), die vollständig und in sich geschlossen ist. Das D. ist das hierarchisch höchste → *Modellierungselement*. Syn. Modell, Datenbeschreibung. Bemerkung 1: Vorsicht! In der Datenbanktheorie ist D. gebräuchlich als Syn. für konzeptionellen Formalismus (d.h. ein D. wird als → *Methode* für die Herstellung eines → *konzeptionellen Schemas* betrachtet). Bemerkung 2: Ein D. besteht aus mindestens einem → *Thema*. Bemerkung 3: In → *INTERLIS* durch das Schlüsselwort MODEL bezeichnet. Das → *Package*, das dem D. entspricht, ist oberhalb aller → Packages, die den → *Themen* eines D. entsprechen.
Datenpunkt	→ *Attribut* oder → *Klasse* für ein → *System* der → *Leittechnik* Bemerkung 1: Definition aus der Praxis: Prozessvariable eines Leitsystems. Bemerkung 2: Prozesszustände und Schalthandlungen werden i.d.R. durch mehrere Datenpunkte beschrieben.
Datenschema	Beschreibung von Inhalt und Gliederung von → *Daten*, die einen anwendungsspezifischen Ausschnitt der Realität charakterisieren, sowie von Regeln, die dafür gelten und von → *Operationen*, welche mit den → *Daten* ausgeführt werden können. Syn. Datenbeschreibung, Schema, konzeptionelles Schema. Bemerkung 1: Mehrzahl: Datenschemata oder Datenschemas. Bemerkung 2: Entsprechend dem Abstraktionsniveau, auf dem man die → *Daten* beschreibt, unterscheidet man das → *konzeptionelle Schema*, das logische Schema und das physische Schema. Zur Formulierung eines D. gibt es geeignete → *Datenbeschreibungssprachen*. Bemerkung 3: Bei → *Datenbanken* wird das dem → *konzeptionellen Schema* entsprechende und gemäss den systemspezifischen Gliederungsmöglichkeiten formulierte logische Schema auch internes Schema genannt. Logische oder auch physische Schemata von peripheren Geräten oder Austauschdateien heissen oft auch externe Schemata oder Formatschemata.

Begriff	Definition
Datenschnittstelle	Zugriff auf → *Daten* eines → *Systems* definiert durch das → *konzeptionelle Schema* der verfügbaren Daten, durch das → *Transferformat* derselben und durch den Transferprozess. Syn. (mehrdeutig!) Schnittstelle (de). Bemerkung 1: D. heisst etwa auch ein Programm zum Umformatieren von → *Transferdateien.* Bemerkung 2: Im objektorientierten Systemaufbau ist die D. gegeben durch die → *Schnittstellensignaturen* der → *Operationen,* mit welchen Daten eingelesen / ausgegebem werden können. Eine objektorientierte Kurzdefinition für D. lautet daher: → *Protokoll* für den → *Datentransfer.* Bemerkung 3: Siehe auch → *Klassenschnittstelle* und → *Benutzerschnittstelle.*
Datentransfer	Verschiebung von → *Daten* von einem → *System S* zu einer anderen → *System Z. S* wird bezeichnet als Startsystem, Ausgangssystem, Quelle, Sender, Sendersystem, Source, *Z* als Zielsystem, Empfänger, Target. Die Lieferung der zu transferierenden → *Daten* durch → *System S* wird auch als Export bezeichnet, die Übernahme durch → *System Z* als Import. Syn. Transfer, Datenübertragung.
Datentransfer-Mechanismus	(Konzeptionelle) → *Datenbeschreibungssprache* und (physisches) → *Transferformat* sowie Regeln zur Herleitung eines solchen → *Transferformats* für Daten, die mit der → *Datenbeschreibungssprache* beschrieben sind.
Datentyp	Syn. für → *Wertebereich.*
Dienst	Angebot zur Lösung einer bestimmten Aufgabe mittels → *Systemen,* deren → *Klassenschnittstellen,* → *Protokolle* und Nutzungsbedingungen eindeutig definiert sind. Syn: Service (englisch)
Ebene	Mehrdeutiges Syn. für 2-dimensionalen Unterraum des →*Raumes* (Mathematik) und Teilmenge von → *Systemen* der → *Leittechnik* (SVT).
Eigentliche Beziehung	Def. siehe bei → *Beziehung*.
Einheit	Basiselement einer Mess-Skala (Beispiele: Meter, Sekunde). Syn. unit (en).
Einzelsteuerebene	Teilmenge von → *Systemen* der → *Leittechnik.* Def. siehe dort
Element	Grundbegriff der Mengenlehre. Eine Menge besteht aus E. Syn. Instanz.
Empfänger	Def. siehe → *Datentransfer*. Syn. Zielsystem.
Entität	Syn. für → *Objekt.*
Entitätsmenge	Syn. für → *Klasse*.
Erweiterung	Syn. für → *Spezialisierung.*

Begriff	Definition
Export	Def. siehe bei → *Datentransfer*.
Extensible Markup Languge	→ *Transferformat*, bei dem jedes Datenelement mit „Tags“ (Kennzeichnungselementen) eingerahmt wird. Abk.: XML
Feature	Syn. für → *Objekt* bzw. oft auch für → *Klasse*.
Feature type	Syn. für → *Klasse*.
Feldebene	Teilmenge von → *Systemen* der → *Leittechnik*. Def. siehe dort
File	Syn. für → *Datei*
Flächiges Bezugssystem	→ *Bezugssystem* mit 2-dimensionalem → *Referenzbereich*. Syn.: Système de repèrage planaire (fr CH) Bemerkung: Vergleiche dazu AGRAM Forschungsbericht [4] S. 8
Format	Syn. für → *Transferformat*
Generalisierung	→ *Rolle* der → *Oberklasse* in einer → *Vererbungsbeziehung*. Syn. generalization (en); généralisation (fr). Bemerkung: G. wird gelegentlich als Syn. für → *Vererbung* verwendet (obschon damit eigentlich die Gegenrichtung gemeint ist).
Generelle Identifikation	→ *Identifikation*, die für alle (modellierten) → *Objekte* einer → *Transfergemeinschaft* eindeutig ist. Bemerkung: Siehe auch → *Objektidentifikation*.
Geo-Informationssystem	→ *System* zur Bearbeitung von Geodaten. Abk: GIS
Gerichtete Beziehung	→ *Aggregation* oder → *Komposition* oder → *Referenzattribut* oder → *Vererbungsbeziehung*.
Gruppensteuerebene	Teilmenge von → *Systemen* der → *Leittechnik*. Def. siehe dort
Identifikation	→ *Attribut* oder Attributskombination, deren → *Wert* ein → *Objekt* in seiner → *Klasse* eindeutig kennzeichnet. Abk. ID. Syn. Identifikator, Identität. Bemerkung: Innerhalb einer → *INTERLIS 2*-Transferdatei erhält jedes → *Objekt* zusätzlich zu den im → *Datenschema* beschriebenen Attributswerten eine I., die es innerhalb der → *Transferdatei* eindeutig kennzeichnet, die so genannte → *Transferidentifikation (→ TID)*. Ist eine solche → *TID* eine → *generelle* und → *stabile I.*, dann nennt man sie eine → *Objektidentifikation (→ OID*).
Identifikator	Syn. für → *Identifikation*.
Identität	Syn. für → *Identifikation*.

Begriff	Definition
Import	Def. siehe bei → *Datentransfer*.
Instanz	Syn. für → *Element* (konkretes Exemplar) einer Menge (Abstraktion). Syn. instance (en; fr). Bemerkung: Beispiele für I.: Ein → *Wert* ist eine I. eines → *Datentyps*. Ein → *Objekt* ist eine I. einer → *Klasse*. Ein → *Behälter* ist eine I. eines → *Themas*. Ein Objektpaar ist eine I. einer → *Assoziationsklasse*.
INTERLIS.	→ *Datentransfer-Mechanismus* für Geodaten bestehend aus der → *INTERLIS-Datenbeschreibungssprache* (IDDL) und dem INTERLIS-XML-Transferformat (IXML) sowie Regeln für die Herleitung des IXML für eine mit IDDL beschriebene Datenstruktur. IDDL, IXML und Umsetzungsregeln sind definiert in der Schweizer → *Norm* SN 612031.
Interface	Syn. (en) für → *Schnittstelle*.
INTERLIS-Compiler	Programm, das aus einem → *Datenschema* in → *IDDL* die Beschreibung des zugehörigen IXML herleitet. Dabei wird die syntaktische Richtigkeit des → *Datenschemas* überprüft (so genanntes Parsing).
INTERLIS-Datenbeschreibungssprache	(Konzeptionelle) → *Datenbeschreibungssprache* des → *Datentransfer-Mechanismus* → *INTERLIS*. Syn. INTERLIS Data Description Language (kurz IDDL). Bemerkung: Ein in IDDL beschriebenes → *Datenschema* kann als (Text-) Datei gespeichert werden. Für solche Schema-Dateien ist das Kürzel "ILI" als Dateinamenzusatz üblich. Beispiel: Die Schema-Datei des Grunddatensatzes der amtlichen Vermessung heisst DM01AV.ILI.
Kardinalität	Bestandteil der → *Rolle* jeder an einer → *Beziehung* beteiligten → *Klasse*. Die K. der → *Klasse A* in der → *Beziehung* zwischen den → *Klassen A und B* ist die Anzahl → *Objekte* der → *Klasse B*, die einem → *Objekt* der → *Klasse A* zugeordnet werden können. Syn. Multiplizität (de), cardinality, multiplicity (en). Bemerkung: In → *UML* wird dafür auch der Begriff der → *Multiplizität* verwendet; mit "Kardinalität" will man dort die konkrete Anzahl der → *Objektbeziehungen* zwischen → *Objektinstanzen* bezeichnen
Klasse	Menge von → *Objekten* mit gleichem Konzept, gleichen Eigenschaften und gleichen → *Operationen*. Jede Eigenschaft wird durch ein → *Attribut* beschrieben, jede → *Operation* durch ihre → *Schnittstellensignatur*. Syn. Objektklasse, Entitätsmenge, Objekttyp (de), feature type, feature, class (en), classe (fr). Bemerkung 1: Eine mit → *INTERLIS 2* beschriebene K. entspricht einer → *UML*-K. mit lauter öffentlichen ("public", d.h. sichtbaren) → *Attributen*. Bemerkung 2: Siehe auch → *Oberklasse,* → *Unterklasse,* → *Tabelle* sowie → *Klassenelement*.

Begriff	Definition
	Bemerkung 3: Eine K. muss nicht → *Objekte* enthalten. Wenn sie → *Objekte* enthalten kann, spricht man von einer konkreten K., wenn nicht, von einer abstrakten K.
Klassendiagramm	Grafische Darstellung von → *Klassen* und ihren → *Beziehungen*. Syn. class diagram (en); diagramme des classes (fr).
Klassenschnittstelle	Zugriff auf einen Teil oder die Gesamtheit der → *Operationen* einer → *Klasse*. Syn. (mehrdeutig!) Schnittstelle (de), interface (en, fr). Bemerkung 1: Eine → *Klasse* kann mehrere K. haben. Für jede derselben kann eine separate → *Schnittstellenklasse* definiert werden. Das → *konzeptionelle* → *Schema* einer → *Schnittstellenklasse* enthält nur → *Schnittstellensignaturen*. Bemerkung 2: Siehe auch → *Benutzerschnittstelle* und → *Datenschnittstelle*.
Komposition	Gerichtete → *eigentliche Beziehung* zwischen einer übergeordneten → *Klasse* und einer untergeordneten → *Klasse*. Einem Ganzen (Ober-Objekt der übergeordneten → *Klasse*) sind mehrere Teile (Unter-Objekte der untergeordneten → *Klasse*) zugeordnet, während einem Teil höchstens ein Ganzes zugeordnet sein kann. Beim Kopieren eines Ganzen werden alle zugeordneten Teile mitkopiert. Beim Löschen eines Ganzen werden alle zugeordneten Teile ebenfalls gelöscht. Syn. composition (en, fr). Bemerkung 1: Die Teile haben dabei keine Selbständigkeit, sondern gehören fest zum Ganzen. Die beteiligten → *Klassen* führen also keine gleichwertige → *Beziehung*, sondern stellen eine Ganzes-Teile-Hierarchie (en: consists-of), dar. Bemerkung 2: In → *INTERLIS* wird die K. als → *Assoziationsklasse* definiert.
Konzeptionelles Schema	Syn. für konzeptionelles → *Datenschema*. Def. siehe bei → *Datenschema* Bemerkung 2. Syn. conceptual schema (en), schéma conceptionelle (fr).
Koordinatensystem	Basis eines Euklidischen Vektorraumes bzw. Urbild der Basis des zugeordneten Euklidischen Vektorraumes beim Kartenhomöomorphismus einer Mannigfaltigkeit (Details siehe Vektoranalysis). Syn. coordinate system (en). Bemerkung 1: Aus Sicht der → *Daten* ist ein K. definiert durch seine Achsen, die entweder Geraden sind (in → *INTERLIS* so genannte LengthAXIS) oder Kreis- bzw. Ellipsen-Bogen (so genannte AngleAXIS) oder einfache Linienzüge (so gennnte PolylineAXIS) entsprechend der Art des → *Referenzbereiches* den sie auszumessen erlauben. Bemerkung 2: Es gibt Cartesische, sphärische, ellipsoidische und → *kurvilineare* K.
Koordinatentransformation	→ *Abbildung* von einem → *Koordinatensystem* (bzw. von seinem → *Referenzbereich*) auf ein anderes → *Koordinatensystem* (bzw. auf dessen → *Reerenzbereich*), wenn die Abbildungsvorschrift (Formel) auf Grund von Annahmen (Hypothesen) fest-

Begriff	Definition
	gelegt wird und die → *Parameter* durch meist statistische Analyse von Messungen in beiden → *Koordinatensystemen* ermittelt werden. Syn. coordinate transformation (en), transformation de coordonnées (fr).
Kurvilineares Koordinatensystem	→ *Koordinatensystem,* dessen erste Achse (u) ein einfacher Linienzug ist und dessen zweite (v) und dritte Achse (w) Geraden sind und in jedem Punkt der ersten Achse auf dem Mittelwert von deren beidseitigen Tangentialvektor-Grenzwerten und aufeinander senkrecht stehen. Bemerkung 1: Musterbeispiel: → *RBBS* mit → *Referenzbereich,* der durch eine Strassenfläche definiert ist, die erste Koordinatenachse durch den Verlauf der Strassenachse (u), die zweite enkrecht dazu in der Strassenfläche (v) und die dritte (w) senkrecht auf den erstenbeiden. Bemerkung 2: Der zu einem k.K. gehörende → *Referenzbereich* eines → *Bezugssystems* enthält im allgemeinen eine 2-dimensionale Untermannigfaltigkeit. Die erste Achse und für jeden ihrer Punkte die entsprechende zweite Achse sind Teilmengen derselben.
Layer	Im CAD-Bereich übliche Bezeichnung für die Zusammenfassung grafischer → *Daten* eines bestimmten → *Typs*. Gelegentlich auch in → *GIS* verwendet für → *Thema*.
Leittechnik	Gesamtheit von hierarchisch geordneten → *Systemen* und Datenströmen zwischen ihnen, um einen bestimmten Prozess zu steuern und zu überwachen. In der L. der SVT werden die beteiligten → *Syseme* in die folgenden Teilmengen (auch Ebenen genannt) gegliedert bzw. wie folgt hierarchisch aufgebaut: - Feldebene: enthält Sensoren und Aktoren - Einzelsteuerebene: enthält → *Streckenstationen* und dient der Erfassung von Prozessdaten sowie zum Absetzen von Steuerbefehlen an die → *Systeme* der Feldebene. - Gruppensteuerebene: dient der Kontrolle von mehreren → *Streckenstationen*. - Prozessleitebene: beinhaltet die eigentliche Prozessleitung (Abbildung der Prozesse) und umfasst auch die lokalen Bedieneinheiten. - Übergeordnete Leitebene: dient der konsolidierten Visualisierung der untergeordneten Systeme der Prozessleitebene.
Lineares Bezugssystem	→ *Bezugssystem* mit 1-diemnsionalem → *Referenzbereich* Syn.: Système de repérage linéaire Bemerkung: Vergleiche dazu AGRAM Bericht [4] S. 8
Merkmal	Syn. für → *Attribut.* Syn. property (en).
Metadaten	→ *Daten* über → *Daten.* Syn. metadata (en), métadonnées (fr). Bemerkung: Speziell in der Geoinformatik sind M. → Daten, die unter anderem Objektbeschreibung in Umgangssprache, Erfassung der → Objekte, Gliederung, Raumbezug, Qualität, Verfügbarkeit und Herkunft, usw. bezeichnen.

Begriff	Definition
Methode	Implementierung einer → *Operation* durch eine Folge von Anweisungen (d.h. durch ein Programm). Syn. method (en); méthode (fr). Bemerkung: Mehrdeutiger Begriff. Oft als Syn. für → *Operation* verwendet.
Modell	Syn. für → *Datenmodell*. Bemerkung: Die objektorientierte Modellierung unterscheidet Objekt-M. (als Syn. für den Teil eines → *Datenschemas*, der Inhalt und Gliederung der → *Daten* beschreibt) und Verhaltens-M. (als Syn. für den Teil eines → *Datenschemas*, der die → *Operationen* beschreibt, die mit den → *Daten* ausgeführt werden können).
Modellbasierter Dienst	→ *Dienst* , der → *Daten* in einem → *Transferformat* bearbeitet, das einem → *Datenmodell* entspricht, zusammen mit diesem → *Datenmodell*.
Modellbasiertes Vorgehen	Vorgehensweise, um von einem → *Realitätsausschnitt* über ein → *konzeptionelles Schema* zu Daten und Programmen für deren Bearbeitung zu gelangen. Syn. model driven approach, model driven architecture (en). Abk. MBV (de); MDA (en). Bemerkung 1: Das modellbasierte Vorgehen hat vier Phasen mit folgenden Resultaten: (1) Beschreibung des → *Realweltausschnitts* in Umgangssprache, (2) konzeptionelles, (3) logisches, (4) physisches → *Datenschema*. Die Phasen (1) und (2) und ihre Resultate sind systemunabhängig. Bemerkung 2: Für die Erstellung des → *konzeptionellen Schemas* kommen Werkzeuge, wie → *UML* und → *INTERLIS 2* zum Einsatz. → *INTERLIS 2* stellt auch Codierungsregeln zur Verfügung, um aus einem → *konzeptionellen Schema* (in → *INTERLIS 2* → *CSL*) das physische → *Datenschema* einer → *Transferdatei* (das →*Transferformat*) herzuleiten. Bemerkung 3 Ein grosser Vorteil des modellbasierten Vorgehens ist, dass durch exakte Formulierung, insbesondere des → *konzeptionellen Schemas*, die Verständigung zwischen Fachleuten über Datenstrukturen ermöglicht wird.
Multiplizität	Syn. für → *Kardinalität*.
Norm	Eine 'de jure' N. (oder kurz N.) ist eine technische Vorschrift, die von nationalen oder internationalen Normenverbänden festgelegt wird. Eine 'de facto' N. ist eine allgemein anerkannte und mehrheitlich genutzte technische Vorschrift; weniger verbindlich als eine 'de jure' N. Syn. standard (en), norme (fr). Bemerkung 1: Ein Gesetz ist eine Vorschrift, die über 'de jure' und 'de facto' N. steht. Bemerkung 2: Deutsches Syn. von 'de facto' N. ist 'Standard'. Die englische Übersetzung von N. ist (gleich geschrieben und fast gleich lautend) 'standard', womit in Deutsch nicht immer klar wird, ob jetzt 'de facto' oder von 'de jure' N. gemeint ist.

Begriff	Definition
Oberklasse	Def. siehe bei → *Vererbungsbeziehung*. Syn. Superklasse (de), super class (en), classe supérieure (fr).
Objekt	Daten eines Gegenstandes der realen Welt zusammen mit den → *Operationen*, die mit diesen Daten ausgeführt werden können, und mit einer → *Objektidentifikation*. Syn. Entität, Tupel, Objektinstanz (de), object instance, feature, feature instance (en), instance d'un object (fr). Bemerkung 1: Siehe auch → *Instanz*, → *Klasse*. Bemerkung 2: Ein O. hat im Gegensatz zu einem → Wert eine → *Identität*, existiert in Raum und Zeit, ist veränderbar bei Wahrung der → *Identität* und kann über Verweise gemeinsam benutzt werden. Ein O. ist konkret. Es ist an die Existenz realer Dinge gebunden. Bemerkung 3: In der objekt-orientierten Literatur findet man folgende blumige Umschreibung des Begriffs O.: Eine konkret vorhandene → *Einheit* mit eigener (unveränderbarer) → *Identität* und definierten Grenzen (im übertragenen Sinne), die Zustand und Verhalten kapselt. Der Zustand wird repräsentiert durch → *Attribute* und → *Beziehungen*, das Verhalten durch → *Operationen*. Jedes O. gehört zu genau einer → *Klasse*. Die definierte → *Struktur* ihrer → Attribute gilt für alle O. einer → *Klasse* gleichermassen, ebenso das Verhalten. Die → Werte der → *Attribute* sind jedoch individuell für jedes O.
Objektbeziehung	Syn. für → *Beziehungsobjekt.*
Objektidentifikation	→ *generelle* und → *stabile Identifikation*. Abk. OID. Syn. Objektidentifikator, Objektidentität (de), object identifier, object identity (en). Bemerkung 1: Die O. wird normalerweise nur von einem → *System* und nicht vom Anwender verändert. Die O. ist eine Eigenschaft, die ein → *Objekt* von allen anderen unterscheidet, auch wenn es möglicherweise die gleichen Attributwerte besitzt. Bemerkung 2: Siehe auch → *Transferidentifikation.* Bemerkung 3: Anhang D zum → *INTERLIS 2*-Referenzhandbuch enthält einen Vorschlag für eine O.
Objektidentifikator	Syn. für → *Objektidentifikation.*
Objektidentität	Syn. für → *Objektidentifikation.*
Objektinstanz	Syn. für → *Objekt*.
Objektklasse	Syn. für → *Klasse*.
Objekttyp	Syn. für → *Klasse*.

Begriff	Definition
OLE for Process Control	Transferprozess zwischen Automatisierungskomponenten mit Steuerungshardware von Feldgeräten. Abk.: OPC Bemerkung: O., wie es in der heute installierten Basis von Produkten implementiert ist, basiert auf Microsoft Distributed Component Objet Model (DCOM).
OPC Alarms and Events	→ *Datenschema* für die Übertragung von Daten zur Überwachung von Alarmen (A) und von Ereignissen (E) über → *OPC*.. Abk.: OPC AE.
OPC Data Access	→ *Datenschema* für Übertragung von Echtzeitwerten über → *OPC* Abk.: OPC DA
Operation	→ *Abbildung* aus den Attributswertebereichen einer → *Klasse* und/oder aus → *Wertebereichen* von Eingabe → *Parametern* in den → *Wertebereich* eines Ausgabe → *Parameters.* Bemerkung 1: Die Implementierung einer O. durch eine Folge von Anweisungen (d.h. durch ein Programm) heisst → *Methode*. Bemerkung 2: Die Beschreibung einer O. heisst → *Schnittstellensignatur* und besteht aus Operationsnamen und Beschreibung der → *Parameter*.
Optional	Muss nicht zwingend vorhanden oder anwendbar sein, ist fakultativ. Gegenteil: Nicht-optional, d.h. obligatorisch. Bemerkung 1: → *Attribute* sind o., wenn nicht gefordert ist, dass sie obligatorisch (mandatory) sind. Für obligatorische → *Attribute* steht in → *IDDL* das Schlüsselwort MANDATORY zur Verfügung. Bemerkung 2: In → *IDDL* bezieht sich "nicht zwingend vorhanden" auf die → *Transferdatei.*
Package	UML-Sprachelement zur Beschreibung von → *Modellen*, → *Themen* und Teilen von Themen. Syn. Paket (de). Bemerkung 1: Ein P. definiert einen Namensraum, d.h. innerhalb eines P. müssen die Namen der enthaltenen benannten Schemaelemente eindeutig sein. Jedes benannte Schemaelement kann in anderen P. referenziert werden, gehört aber zu genau einem (Heimat-) P. Bemerkung 2: Bei → *UML* können die P. wiederum P. enthalten. Das oberste P. beinhaltet das Gesamtsystem entsprechend dem → *Datenmodell* von → *INTERLIS 2*.
Parameter	Daten(elemente), deren → Wert einer → *Funktion*, einer → *Operation* oder einem → *Metaobjekt* übergeben und/oder von → *Funktionen* oder → *Operationen* zurückgegeben werden. Zu jedem P. gehört ein Name, ein → *Wertebereich* und - bei → *Funktionen* oder → *Operationen* - eine Übergaberichtung (in, out, inout). Der konkrete → *Wert* eines P. heisst → Argument Bemerkung 1: Siehe auch → *Laufzeitparameter*. Bemerkung 2: Mittels P. werden diejenigen Eigenschaften von → *Metaobjekten* bezeichnet, die nicht das → *Metaobjekt* selber, sondern dessen Gebrauch in der Anwendung betreffen.

Begriff	Definition
Protokoll	Menge der → *Operationen*, die zu einer Menge von → *Klassen* gehören.
Prozesssleitebene	Teilmenge von → *Systemen* der → *Leittechnik*. Def. siehe dort
Querschnittsteuerung	Syn. von → *Streckenstation*
Raum	3-dimensionaler Euklidischer Raum. (Def. siehe Mathematik.)
Raumbezugssystem	Syn. von → *Bezugssystem*.
Räumliches Basisbezugssystem	Syn. von → *Basisbezugssystem*. Abk.: RBBS
Realitätsausschnitt	Teil der Realwelt, der für die Bearbeitung eines Themas oder Themenbereiches wesentlich ist. Bmerkung: Ein Gegenstand der Realwelt kann zu verschiedenen Realitätsausschnitten gehören und entsprechend können sehr verschiedene Eigenschaften wesentlich sein. Beispiel: Strasse für Strassenbau und für Adressenverwaltung.
Referenzattribut	→ *Beziehung*, die nur dem ersten → *Objekt* jedes Objektpaars der → *Beziehung* bekannt ist. Syn.. einseitige → *Beziehung*.
Referenzbereich	Teilmenge des → *Raumes*, die eine Mannigfaltigkeit der Dimension 1, 2 oder 3 ist. Syn.: Systéme de référence (fr CH), Referenzfläche (de CH). Bemerkung 1: Details zu Mannigfaltigkeiten siehe Vektoranalysis. Bemerkung 2: Zum système de référence vergleiche AGRAM Bericht [4].
Referenzfläche	Syn. für → *Referenzbereich*.
Relation	Syn. für → *Tabelle*.
Rolle	Bedeutung der → *Objekte* einer → *Klasse* in einer → *Beziehung*. Bemerkung: In einer → *eigentlichen Beziehung* wird die R. jeder beteiligten → *Klasse* beschrieben durch ihren Namen, ihre → *Stärke* und ihre → *Kardinalität*. Ein → *Referenzattribut* beschreibt die R. der → *Klasse* mit diesem → *Attribut*. In der → *Vererbungsbeziehung* sind die R. implizit definiert.
Schema	Syn. für → *Datenschema* (Mehrzahl: Schemata oder neu auch Schemas).
Schnittstelle	Mehrdeutiges Syn. für → *Klassenschnittstelle*, → *Benutzerschnittstelle* und → *Datenschnittstelle*. Syn. interface (en, fr).
Schnittstellenklasse	Syn. für Klassenschnittstellen-Klasse. Def. siehe bei → *Klassenschnittstelle*.

Begriff	Definition
Schnittstellensignatur	Beschreibung des Aufrufs einer → *Operation*, setzt sich zusammen aus dem Namen der → *Operation*, den → *Datentypen* und allenfalls Namen ihrer → *Parameter* und evtl. der Angabe eines Rückgabe-Datentyps. Syn. Signatur (de).
Sender	Def. siehe → *Datentransfer*.
Semantische Transformation	Umbau eines → *Datenmodells* in ein inhaltlich (semantisch) aequivalentes → *Datenmodell*. Bemerkung: S.T wird auch semantikerhaltende Transfornation genannt
Service	Syn. von → *Dienst.*
Spezialisierung	→ *Rolle* der → *Unterklasse* in einer → *Vererbungsbeziehung*, oft auch Syn. für → *Vererbung.* Syn. Erweiterung (de), extension (en), specialisation (fr).
Stabile Identifikation	→ *Identifikation*, die zeitunabhängig ist, d.h. während dem Lebenszyklus eines → *Objektes* nicht verändert werden kann. Die s. I. eines gelöschten → *Objektes* darf nicht mehr verwendet werden. Bemerkung: Siehe auch → *Objektidentifikation*.
Standard	Syn. (de) für 'de facto' → *Norm* und (en) für 'de facto' oder 'de jure' → *Norm*.
Stärke	Bindung der Teile (Unter-Objekte der untergeordneten → *Klasse*) an das Ganze (Ober-Objekt der übergeordneten → *Klasse*) bei einer → *eigentlichen Beziehung*.
Steuerungsebene	Teilmenge von → *Systemen* der → *Leittechnik*. Def. siehe dort
Streckenstation	→ *System* der → *Gruppensteuerebene* der → *SVT* → *Leittechnik* zur Datenerfassung, Datenaggregation, Datenübermittlung und zur Weitergabe von Steuerbefehlen Syn.: Querschnittsteuerung. Bemerkung: Ein Beispiel ist SAIA SPS Typ 3, siehe 4.2.3 .
System	Gesamtheit aller zu einer EDV-Anlage gehörenden Komponenten (Hardware und Software), die für einen bestimmten Zweck genutzt werden.
Tabelle	→ *Klasse* für deren → *Objekte* kein → *Objektidntifikator* (implizit) existiert und keine → *Operationen* explizit definierbar sind. Syn. Relation
Thema	Menge von → *Klassen*, deren → *Daten* in gewissem Sinne zusammengehören, die z.B. eine Beziehung haben, oder zu derselben Datenverwaltungsstelle gehören, oder einen ähnlichen Nachführungs-Rhythmus besitzen. Die → *Instanzen* von T. sind → *Behälter*. Syn. topic (en), thème (fr). Bemerkung 1: Ein T. wird mit → *INTERLIS 2* als → *TOPIC* beschrieben. Ein → *TOPIC* wird in → *UML* durch ein → *Package* unterhalb eines → *Datenmodells* beschrieben, mit der zusätzlichen Bedeutung, dass dieses → *Package* (a) einen eigenen

Begriff	Definition
	Namensraum hat und (b) von anderen → *Packages* abhängig (z.B. erweitert) sein kann. Ein UML-Package, das einem T. zugeordnet ist, kann weitere (geschachtelte) → *Packages* enthalten. Bemerkung 2: Beachte: Mit → *Layer*, dem im CAD-Bereich gebräuchlichen Ausdruck für "Ebene", wird eine Zusammenfassung grafischer → *Daten* bezeichnet. Ein T. kann mehrere (grafische) → *Layer* umfassen und zusätzlich strukturierte Sachdaten.
Topic	Syn. für → *Thema*.
Transfer	Syn. für → *Datentransfer*.
Transferdatei	Zum → *Datentransfer* vorbereitete → *Datei* in geeignetem → *Transferformat*.
Transferformat	Gliederung einer → *Transferdatei* in Datenfelder. Syn. Format.
Transfergemeinschaft	Gemeinschaft von → *Sendern* und → *Empfängern*, die sich an einem → *Datentransfer* beteiligen.
Transferidentifikation	Def. siehe bei → *Identifikation*. Abk. TID.
Transformation	Mehrdeutiges Syn. für → *Koordinatentransformation* und → *semantische Transformation*
Tupel	Syn. für Zeile einer→ *Tabelle*.
Typ	Mehrdeutiges Syn. für → *Datentyp* (d.h. → *Wertebereich*), → *Klassenschnittstelle* und → *Schnittstellensignatur*.
Übergeordnete Leitebene	Teilmenge von → *Systemen* der → *Leittechnik*. Def. siehe dort.
Unified Modeling Language.	Def. siehe www.omg.org/. Abk.: UML Bemerkung: Das UML-Klassendiagrmm wird als grafische konzeptionelle → *Datenbeschreibungssprahe* verwendet.
Unit	Syn. für → *Einheit*.
Unterklasse	Def. siehe → *Vererbungsbeziehung*. Syn. Unterobjektklasse, Subobjektklasse (de), subclass (en), classe inférieure (fr).
Vererbung	Möglichkeit zur Def. von → *Vererbungsbeziehungen* zwischen → *Oberklassen* und → *Unterklassen*. Möglich sind → *Klassenspezialisierung* und → *Attributsspezialisierung*. Syn. inheritance (en), héritage (fr). Bemerkung 1: Anschaulich entsprechen → *Unterklassen* demselben Konzept, sie haben dieselben Eigenschaften wie ihre → *Oberklassen* und spezialisieren diese. Bemerkung 2: Man unterscheidet → *Einfachvererbung* und → *Mehrfachvererbung*. Bei einer → *Einfachvererbung* (en: single

Begriff	Definition
	inheritance; fr: héritage singulaire) erbt eine → *Unterklasse* nur von einer direkten → *Oberklasse*. Bei der → *Mehrfachvererbung* erbt eine → *Klasse* von mehreren → *Oberklassen*. Bemerkung 3: → *INTERLIS 2* lässt nur einfache V. zu (wie z.B. Java).
Vererbungsbeziehung	→ *Gerichtete Beziehung* zwischen einer übergeordneten → *Klasse*, genannt → *Oberklasse*, und einer untergeordneten → *Klasse*, genannt → *Unterklasse*, definiert durch → *Vererbung*. Die → *Rolle* der → *Oberklasse* heisst → *Generalisierung*, die → *Rolle* der → *Unterklasse* heisst → *Spezialisierung*. Bemerkung 1: Die → *Objekte* der → *Oberklasse* sind Verallgemeinerungen (→ *Generalisierungen*) der → *Objekte* der → *Unterklasse*. Die → *Objekte* der → *Unterklasse* sind Einschränkungen (→ *Spezialisierungen,* → *Erweiterungen*) der → *Objekte* der → *Oberklasse*. Bemerkung 2: Die V. ist die Teilmengenbeziehung, die → *Objekte* der → *Unterklasse* bilden eine Teilmenge der → *Objekte* der → *Oberklasse*. Es handelt sich also bei den → *Objekten* der → *Unterklasse* nicht um neue → *Objekte*, sondern um einen Teil oder eine Unterteilung der → *Objekte* der → *Oberklasse*. Die beiden → *Objekte* eines Objektpaars der V. haben dieselbe → *OID*
Wert	→ *Datenelement* eines → *Wertebereichs.*
Wertebereich	Menge gleichartiger → *Datenelemente*. Ein → *Datenelement* eines W. heisst → *Wert.* Syn. Datentyp.

13. Liste der Anhänge

A1: Transferformate

A2: Einführung in OPC

A3: Datenmodelle in INTERLIS 2

A4: Beschreibung der Standardformate in XML-Schema

A. Anhang

A.1. Transferformate

A.1.1. Transferformat Verkehrszähler (Intervallsummen) – Streckenstation

Siehe Abbildung 8 in Kapitel 5.1.1

A.1.2. Transferformat Streckenstation – VDE Bereichsrechner

Pro Spur (für 16 Spuren)				
Bezeichnung	**Einheit**	**Werte-bereich**	**OPC-Datentyp**	**OPC-Tag**
Kopfdaten (pro Zykluszeit T)				
Zeitstempel (Triggerzeit) Tag		0 - 31	Integer	DATE_DAY
Zeitstempel (Triggerzeit) Monat		0 - 12	Integer	DATE_MONTH
Zeitstempel (Triggerzeit) Jahr		0 - 65535	Integer	DATE_YEAR
Zeitstempel (Triggerzeit) Stunde		0 - 23	Integer	TIME_HOUR
Zeitstempel (Triggerzeit) Minute		0 - 59	Integer	TIME_MIN
Zeitstempel (Triggerzeit) Sekunde		0 - 59	Integer	TIME_SEC
Zykluszeit T	Sekunden	30 - 3600	Integer	CYCLE
Glättungstiefe M (Moving Average)		0 - 10	Integer	MOV_AV
Spurdaten (pro Zykluszeit T)			Integer	
Mittlere Geschwindigkeit	Km/h	0 - 255	Integer	AV_SPEED
Standardabweichung	δ / T	0 - 255	Integer	AV_SPEED_T
Verkehrsmenge / -frequenz	Fz / T	0 - 4000	Integer	VF_TOTAL
Verkehrsmenge / -frequenz Fahrzeugklasse 1	Fz. Kl. 1 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL1
Verkehrsmenge / -frequenz Fahrzeugklasse 2	Fz. Kl. 2 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL2
Verkehrsmenge / -frequenz Fahrzeugklasse 3	Fz. Kl. 3 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL3
Verkehrsmenge / -frequenz Fahrzeugklasse 4	Fz. Kl. 4 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL4
Verkehrsmenge / -frequenz Fahrzeugklasse 5	Fz. Kl. 5 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL5
Verkehrsmenge / -frequenz Fahrzeugklasse 6	Fz. Kl. 6 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL6
Verkehrsmenge / -frequenz Fahrzeugklasse 7	Fz. Kl. 7 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL7
Verkehrsmenge / -frequenz Fahrzeugklasse 8	Fz. Kl. 8 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL8
Verkehrsmenge / -frequenz Fahrzeugklasse 9	Fz. Kl. 9 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL9
Verkehrsmenge / -frequenz Fahrzeugklasse 10	Fz. Kl. 10 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL10
Verkehrsdichte	Fz / km	0 - 200	Integer	VD_TOTAL
Belegungsgrad	Belegungszeit%	0 - 100	Integer	B_GRAD
Verkehrszustand		0 - 6	Integer	V_STATUS
Fahrtrichtung	0 = kein Fzg. / T 1 = in Fahrtrichtung 2 = gegen	0,1,2,9	Integer	F_RICHTUNG

	Fahrtrichtung 9 = beide Richtungen in einem Zyklus T			
Reserve 2		0 - 65535	Integer	RES_2
Reserve 3		0 - 65535	Integer	RES_3
Reserve 4		0 - 65535	Integer	RES_4
Reserve 5		0 - 65535	Integer	RES_5
Spurdaten pro Fahrzeugklasse (pro Zykluszeit T)			Integer	
Mittlere Geschwindigkeit Fahrzeugklasse 1	Km/h	0 - 255	Integer	AV_SPEED_KL1
Mittlere Geschwindigkeit Fahrzeugklasse 2	Km/h	0 - 255	Integer	AV_SPEED_KL2
Mittlere Geschwindigkeit Fahrzeugklasse 3	Km/h	0 - 255	Integer	AV_SPEED_KL3
Mittlere Geschwindigkeit Fahrzeugklasse 4	Km/h	0 - 255	Integer	AV_SPEED_KL4
Mittlere Geschwindigkeit Fahrzeugklasse 5	Km/h	0 - 255	Integer	AV_SPEED_KL5
Mittlere Geschwindigkeit Fahrzeugklasse 6	Km/h	0 - 255	Integer	AV_SPEED_KL6
Mittlere Geschwindigkeit Fahrzeugklasse 7	Km/h	0 - 255	Integer	AV_SPEED_KL7
Mittlere Geschwindigkeit Fahrzeugklasse 8	Km/h	0 - 255	Integer	AV_SPEED_KL8
Mittlere Geschwindigkeit Fahrzeugklasse 9	Km/h	0 - 255	Integer	AV_SPEED_KL9
Mittlere Geschwindigkeit Fahrzeugklasse 10	Km/h	0 - 255	Integer	AV_SPEED_KL10
Standardabweichung Fahrzeugklasse 1	δ / T	0 - 255	Integer	AV_SPEED_T_KL1
Standardabweichung Fahrzeugklasse 2	δ / T	0 - 255	Integer	AV_SPEED_T_KL2
Standardabweichung Fahrzeugklasse 3	δ / T	0 - 255	Integer	AV_SPEED_T_KL3
Standardabweichung Fahrzeugklasse 4	δ / T	0 - 255	Integer	AV_SPEED_T_KL4
Standardabweichung Fahrzeugklasse 5	δ / T	0 - 255	Integer	AV_SPEED_T_KL5
Standardabweichung Fahrzeugklasse 6	δ / T	0 - 255	Integer	AV_SPEED_T_KL6
Standardabweichung Fahrzeugklasse 7	δ / T	0 - 255	Integer	AV_SPEED_T_KL7
Standardabweichung Fahrzeugklasse 8	δ / T	0 - 255	Integer	AV_SPEED_T_KL8
Standardabweichung Fahrzeugklasse 9	δ / T	0 - 255	Integer	AV_SPEED_T_KL9
Standardabweichung Fahrzeugklasse 10	δ / T	0 - 255	Integer	AV_SPEED_T_KL10
Aufbereitete Daten (über Glättungstiefe M)				
Mittlere Geschwindigkeit	Km/h	0 - 255	Integer	AV_SPEED_M
Standardabweichung	δ / T	0 - 255	Integer	AV_SPEED_T_M
Verkehrsmenge / -frequenz	Fz / T	0 - 4000	Integer	VF_TOTAL_M
Verkehrsmenge / -frequenz Fahrzeugklasse 1	Fz. Kl. 1 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL1_M
Verkehrsmenge / -frequenz Fahrzeugklasse 2	Fz. Kl. 2 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL2_M
Verkehrsmenge / -frequenz Fahrzeugklasse 3	Fz. Kl. 3 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL3_M

Verkehrsmenge / -frequenz Fahrzeugklasse 4	Fz. Kl. 4 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL4_M
Verkehrsmenge / -frequenz Fahrzeugklasse 5	Fz. Kl. 5 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL5_M
Verkehrsmenge / -frequenz Fahrzeugklasse 6	Fz. Kl. 6 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL6_M
Verkehrsmenge / -frequenz Fahrzeugklasse 7	Fz. Kl. 7 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL7_M
Verkehrsmenge / -frequenz Fahrzeugklasse 8	Fz. Kl. 8 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL8_M
Verkehrsmenge / -frequenz Fahrzeugklasse 9	Fz. Kl. 9 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL9_M
Verkehrsmenge / -frequenz Fahrzeugklasse 10	Fz. Kl. 10 / T (gem Swiss10)	0 - 4000 (1)	Integer	VF_KL10_M
Verkehrsdichte	Fz / km	0 - 200	Integer	VD_TOTAL_M
Belegungsgrad	Belegungszeit%	0 - 100	Integer	B_GRAD_M
Verkehrszustand		0 - 6	Integer	V_STATUS_M
Fahrtrichtung	0 = kein Fzg. / T	0,1,2,9	Integer	F_RICHTUNG_M
Reserve 2		0 - 65535	Integer	RES_2_M
Reserve 3		0 - 65535	Integer	RES_3_M
Reserve 4		0 - 65535	Integer	RES_4_M
Reserve 5		0 - 65535	Integer	RES_5_M
Aufbereitete Daten pro Fahrzeugklasse (über Glättungstiefe M)				
Mittlere Geschwindigkeit Fahrzeugklasse 1	Km/h	0 - 255	Integer	AV_SPEED_KL1_M
Mittlere Geschwindigkeit Fahrzeugklasse 2	Km/h	0 - 255	Integer	AV_SPEED_KL2_M
Mittlere Geschwindigkeit Fahrzeugklasse 3	Km/h	0 - 255	Integer	AV_SPEED_KL3_M
Mittlere Geschwindigkeit Fahrzeugklasse 4	Km/h	0 - 255	Integer	AV_SPEED_KL4_M
Mittlere Geschwindigkeit Fahrzeugklasse 5	Km/h	0 - 255	Integer	AV_SPEED_KL5_M
Mittlere Geschwindigkeit Fahrzeugklasse 6	Km/h	0 - 255	Integer	AV_SPEED_KL6_M
Mittlere Geschwindigkeit Fahrzeugklasse 7	Km/h	0 - 255	Integer	AV_SPEED_KL7_M
Mittlere Geschwindigkeit Fahrzeugklasse 8	Km/h	0 - 255	Integer	AV_SPEED_KL8_M
Mittlere Geschwindigkeit Fahrzeugklasse 9	Km/h	0 - 255	Integer	AV_SPEED_KL9_M
Mittlere Geschwindigkeit Fahrzeugklasse 10	Km/h	0 - 255	Integer	AV_SPEED_KL10_M
Standardabweichung Fahrzeugklasse 1	δ / T	0 - 255	Integer	AV_SPEED_T_KL1_M
Standardabweichung Fahrzeugklasse 2	δ / T	0 - 255	Integer	AV_SPEED_T_KL2_M
Standardabweichung Fahrzeugklasse 3	δ / T	0 - 255	Integer	AV_SPEED_T_KL3_M
Standardabweichung Fahrzeugklasse 4	δ / T	0 - 255	Integer	AV_SPEED_T_KL4_M
Standardabweichung Fahrzeugklasse 5	δ / T	0 - 255	Integer	AV_SPEED_T_KL5_M
Standardabweichung Fahrzeugklasse 6	δ / T	0 - 255	Integer	AV_SPEED_T_KL6_M
Standardabweichung Fahrzeugklasse 7	δ / T	0 - 255	Integer	AV_SPEED_T_KL7_M
Standardabweichung Fahrzeugklasse 8	δ / T	0 - 255	Integer	AV_SPEED_T_KL8_M

Standardabweichung Fahrzeugklasse 9	δ / T	0 - 255	Integer	AV_SPEED_T_KL9_M
Standardabweichung Fahrzeugklasse 10	δ / T	0 - 255	Integer	AV_SPEED_T_KL10_M
Sonstige Daten				
Alarme und Statusmeldungen				
Bezeichnung	**Einheit**	**Werte-bereich**	**OPC-Datentyp**	**OPC-Tag**
Streckenstation SAIA PCD3				
Allgemeine Störmeldung (muss noch definiert werden)	High = Alarm	0 - 1	Boolean	PCD_SA
Batteriestörung SPS	High = Alarm	0 - 1	Boolean	PCD_BATT
Status Kommunikation Marks-man SPS / ASTRA (Modem) (4)	High = SPS Low = Astra	0 - 1	Boolean	PCD_MM_SPS
Störung Reserve 2 (3)	High = Alarm	0 - 1	Boolean	PCD_S_2
Störung Reserve 3 (3)	High = Alarm	0 - 1	Boolean	PCD_S_3
Störung Reserve 4 (3)	High = Alarm	0 - 1	Boolean	PCD_S_4
Störung Reserve 5 (3)	High = Alarm	0 - 1	Boolean	PCD_S_5
Störung Reserve 6 (3)	High = Alarm	0 - 1	Boolean	PCD_S_6
Störung Reserve 7 (3)	High = Alarm	0 - 1	Boolean	PCD_S_7
Störung Reserve 8 (3)	High = Alarm	0 - 1	Boolean	PCD_S_8
Pro Marksman 660				
Kommunikationsüberwachung	High = Alarm	0 - 1	Boolean	MM_X_KOMM
Parametrierfehler (5)	High = Alarm	0 - 1	Boolean	MM_X_PARA
Spannungsversorgung Interne Akku High < 6.7V	High = Alarm	0 - 1	Boolean	MM_X_SPG
Schlaufenüberwach. Loop 1_1	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_1_1
Schlaufenüberwach. Loop 1_2	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_1_2
Schlaufenüberwach. Loop 2_1	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_2_1
Schlaufenüberwach. Loop 2_2	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_2_2
Schlaufenüberwach. Loop 3_1	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_3_1
Schlaufenüberwach. Loop 3_2	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_3_2
Schlaufenüberwach. Loop 4_1	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_4_1
Schlaufenüberwach. Loop 4_2	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_4_2
Schlaufenüberwach. Loop 5_1	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_5_1
Schlaufenüberwach. Loop 5_2	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_5_2
Schlaufenüberwach. Loop 6_1	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_6_1
Schlaufenüberwach. Loop 6_2	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_6_2
Schlaufenüberwach. Loop 7_1	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_7_1
Schlaufenüberwach. Loop 7_2	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_7_2
Schlaufenüberwach. Loop 8_1	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_8_1
Schlaufenüberwach. Loop 8_2	High = Alarm	0 - 1	Boolean	MM_X_LPSTATUS_8_2

A.1.3. Transferformat ab Bereichsrechner VDE

Das Transferformat ab Bereichrechner VDE ist ein XML-Format und liegt in einer Beschreibung mit Hilfe von XML-Schema vor. Hier in den Berichtstext übernommen werden nur Ausschnitte aus der umfangreichen XML-Schema Datei. Die vollständige Formatbeschreibung findet sich als xsd-Datei namens TelegrammAnnahme_BerRechVDE.xsd in der zum Bericht gehörigen zip-Datei MDAinSVT_20080917.zip

(a) Start der XML-Schema Datei und Definition einiger Aufzähltypen

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- WSDL Definition: UeLS ZH -->
<!-- Version : $Revision: 1.8 $ -->
<!-- Datum : $Date: 2005/05/11 09:00:37 $-->
<!-- $Author: dsch $-->
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:t0="http://www.hlszh.ch/Telegramm"
targetNamespace="http://www.hlszh.ch/Telegramm" elementFormDefault="qualified" attributeForm-
Default="unqualified" version="2005-05-11"
xmlns:xdb="http://www.borland.com/schemas/delphi/6.0/XMLDataBinding">
     <element name="Telegramm" type="t0:TelegrammStrukturType"/>
     <element name="TelegrammAnnahme">
          <complexType>
               <annotation>
                    <appinfo/>
                </annotation>
                <sequence>
                    <element name="tlg" type="t0:TelegrammStrukturType" minOccurs="0"/>
               </sequence>
          </complexType>
     </element>
     <element name="TelegrammAnnahmeResponse">
          <complexType>
               <annotation>
                    <appinfo xdb:docElement="TelegrammAnnahmeResponse"/>
                </annotation>
               <sequence>
                    <element name="TelegrammAnnahmeResult" type="t0:AnnahmeQuittung"/>
               </sequence>
          </complexType>
     </element>

     <simpleType name="AnlassType">
          <restriction base="string">
               <enumeration value="Start"/>
               <enumeration value="Fortgang"/>
          </restriction>
     </simpleType>
     <simpleType name="BefehlsQuittungType">
          <restriction base="string">
               <enumeration value="OK"/>
               <enumeration value="NOK"/>
               <enumeration value="Busy"/>
          </restriction>
     </simpleType>
     <simpleType name="AnnahmeQuittung">
          <restriction base="string">
               <enumeration value="ACK"/>
               <enumeration value="NACK"/>
          </restriction>
     </simpleType>
.............
```

(b) Beginn des XML-Schema Bereichs, der als Beschreibung der Formate von Klassen und deren Attribute interpretiert werden kann. Für die Übersetzung der XML-Schema `choice` Elemente nach UML / INTERLIS kommen zwei Varianten zum Einsatz. (1) Bei Attributauswahl: Diese Attribute plus eines vom Aufzähltyp und Konsistenzbedingung (`DEFINED`), vgl. den folgenden `AblesungType` mit der Lösung für `ablesung` in Klasse `stellwert` von A.3.3. (2) Bei Klassenauswahl: XOR Assoziation, vgl. `AnfrageErgebnisType` mit `AssociationDef93` in A.3.3.

```
<complexType name="AblesungType">
     <choice>
          <element name="WertAngabe" type="t0:WertAngabeType" minOccurs="0"/>
          <element name="NichtVerfuegbar" type="t0:LeerType" minOccurs="0"/>
     </choice>
</complexType>

<complexType name="AktivMeldungType">
     <sequence>
          <element name="Anlass" type="t0:AnlassType">
              <annotation>
                     <documentation>Start/Fortgang</documentation>
              </annotation>
          </element>
          <element name="CheckUp" type="t0:CheckUpType">
              <annotation>
                     <documentation>Alles ok / Probleme / Konfiguration</documentation>
              </annotation>
          </element>
     </sequence>
</complexType>

<complexType name="AnfrageErgebnisType">
     <annotation>
          <documentation>
              Struktur fÃ¼r Antwort-Inhalte,
              differenziert nach Art der Anfrage</documentation>
          <appinfo/>
     </annotation>
     <choice>
          <element name="DatenPunktInfoListe" type="t0:DatenPunktInfoListeType"
           minOccurs="0"/>
          <element name="StellWerte" type="t0:StellWertListeType" minOccurs="0"/>
          <element name="BefehlsQuittung" type="t0:BefehlsQuittungType" minOccurs="0"/>
          <element name="Messungen" type="t0:MessungListeType" minOccurs="0"/>
          <element name="Ereignisse" type="t0:EreignisListeType" minOccurs="0"/>
          <element name="LogbuchAuszug" type="t0:LogbuchAuszugType" minOccurs="0"/>
          <element name="Versionen" type="t0:VersionsListeType" minOccurs="0"/>
     </choice>
</complexType>

<complexType name="AntwortType">
     <annotation>
          <documentation>Anfragereferenz und Inhalt</documentation>
          <appinfo/>
     </annotation>
     <sequence>
          <element name="AnfrageLaufNr" type="positiveInteger">
              <annotation>
                     <documentation>
                            LaufNr der Anfrage des Antwort-EmpfÃ¤ngers<
                     /documentation>
              </annotation>
          </element>
          <element name="AnfrageErgebnis" type="t0:AnfrageErgebnisType">
              <annotation>
                     <documentation>Angeforderte Info</documentation>
              </annotation>
          </element>
     </sequence>
</complexType>
```

(c) Übersicht der Gesamtstruktur des Transferformats für den Output des Bereichsrechners VDE mit Hilfe der complexTypes `TelegrammStrukturType`, `TelegrammKopfType` und `TelegrammRumpfType`. Das Element `Kopf` von `TelegrammStrukturType` sowie die Elemente `Befehl`, `Antwort` und `Meldung` von `TelegrammRumpfType` bilden die Hauptklassen im UML_Diagramm 5.1.3 und im INTERLIS Text A.3.3.

```
<complexType name="TelegrammKopfType">
     <annotation>
          <documentation>KopfElemente</documentation>
          <appinfo/>
     </annotation>
     <sequence>
          <element name="An" type="t0:DatenPunktKennungType"/>
          <element name="Von" type="t0:DatenPunktKennungType"/>
          <element name="LaufNr" type="positiveInteger"/>
          <element name="Prio" type="t0:PrioStufeType"/>
          <element name="VersandZeit" type="dateTime">
              <annotation>
                     <documentation>Datum und Zeit im ISO 8601 Standard-Format
                     </documentation>
              </annotation>
          </element>
     </sequence>
</complexType>

<complexType name="TelegrammRumpfType">
     <choice>
          <element name="Befehl" type="t0:BefehlType" minOccurs="0"/>
          <element name="Antwort" type="t0:AntwortType" minOccurs="0"/>
          <element name="Meldung" type="t0:MeldungType" minOccurs="0"/>
     </choice>
</complexType>

<complexType name="TelegrammStrukturType">
     <annotation>
          <documentation>TelegrammElemente</documentation>
          <appinfo/>
     </annotation>
     <sequence>
          <element name="Kopf" type="t0:TelegrammKopfType"/>
          <element name="Rumpf" type="t0:TelegrammRumpfType"/>
          <element name="Anhang" type="string" minOccurs="0"/>
     </sequence>
</complexType>

<complexType name="TextInfoType">
     <sequence>
          <element name="Textwert" type="t0:Text100Type">
              <annotation>
                     <documentation>Initialtext</documentation>
              </annotation>
          </element>
          <element name="MaxLen" type="integer"/>
          <element name="PMT" type="t0:Text160Type"/>
     </sequence>
</complexType>

..............

  <complexType name="WertAngabeType">
        <choice>
              <element name="MengenAngabe" type="t0:MengenAngabeType" minOccurs="0"/>
              <element name="Stufe" type="t0:Text50Type" minOccurs="0"/>
              <element name="Text" type="t0:Text100Type" minOccurs="0"/>
              <element name="UhrStellung" type="dateTime" minOccurs="0"/>
        </choice>
  </complexType>
```

(d) Schluss der XML-Schema Datei

```
    <complexType name="ZeitSpanneType">
        <sequence>
            <element name="StartZeit" type="dateTime">
                <annotation>
                    <documentation>Datum und Zeit im ISO 8601 Standard-Format
                    </documentation>
                </annotation>
            </element>
            <element name="EndZeit" type="dateTime">
                <annotation>
                    <documentation>Datum und Zeit im ISO 8601 Standard-Format
                    </documentation>
                </annotation>
            </element>
        </sequence>
    </complexType>

    <complexType name="BenutzerInfoType">
        <annotation>
            <documentation>Informatione bezÃ¼glich eines einzelnen Benutzers des UeLS
            </documentation>
            <appinfo/>
        </annotation>
        <sequence>
            <element name="Username" type="t0:Text100Type"/>
            <element name="Benutzergruppe" type="t0:Text100Type"/>
            <element name="Passwort" type="t0:Text100Type"/>
        </sequence>
    </complexType>

    <complexType name="BenutzerInfoListeType">
        <annotation>
            <documentation>
                Liste mit allen Informationen bezÃ¼glich der Benutzers des Systems
            </documentation>
            <appinfo/>
        </annotation>
        <sequence>
            <element name="BenutzerInfo" type="t0:BenutzerInfoType" minOccurs="0"
              maxOccurs="unbounded"/>
        </sequence>
    </complexType>
</schema>
```

A.1.4. Verkehrszähler Einzelfahrzeuge (Typ MM660) – VM-CH-System

Siehe Abbildung 16 in Kapitel 5.1.4

A.1.5. Verkehrszähler Einzelfahrzeuge (Typ ECTN) – VM-CH-System

Siehe Abbildung 19 in Kapitel 5.1.5

A.1.6. ASTRA-Zählstellen

Die nachfolgende Tabelle zeigt einen beispielhaften Ausschnitt der Messstellen der Automatischen Strassenverkehrszählung (AVZ) in der Umgebung des Beispielgebietes (Kt. Zürich).

Nr	Beschreibung	Richtung 1	Richtung 2	Str	S10	Koo E	Koo N
13	KILCHBERG N, HORN	Sargans	Zürich	K3		683407	243205
20	Umf Zürich N Affoltern (AB)	Glattbrugg	Verz A10/A1	A10	Ja	681000	253875
29	Zürich L'talstr, Stadtgrenze	Zürich	Baden	K296		678415	251026
54	Adliswil (AB)	Sargans	Zürich	A3	Ja	682858	241587
55	Richterswil (AB)	Sargans	Zürich	A3	Ja	693420	229725
65	Schlieren (AB)	Zürich	Bern	A1	Ja	676950	251050

Tabelle 3 Ausschnitt Messstellenverzeichnis ASTRA, Stand 02.2008

Da laufend weitere Anpassungen der Liste durch das ASTRA stattfinden, ist es zweckmässig, sich über das Internet in der aktuellen Version der Liste zu informieren. Link: http://www.astra.admin.ch/verkehrsdaten/00299/00301/00379/

A.2. Einführung in OPC

Die nachfolgende Einführung von OPC ist der freien Web-Enzyklopädie Wikipedia (www.wikipedia.org) entnommen.

OLE for Process Control (OPC) war der ursprüngliche Name für standardisierte Software-Schnittstellen, die den Datenaustausch zwischen Anwendungen unterschiedlichster Hersteller in der Automatisierungstechnik ermöglichen. Durch die fortschreitende Weiterentwicklung dieser Schnittstellen und der damit einhergehenden Abnahme der Relevanz des OLE-Objektsystems wird heute lediglich die Bezeichnung OPC, ohne auf eine Abkürzung hinzuweisen, genutzt.

A.2.1. Entstehung

OPC ist der Versuch, industriellen Bussystemen und Protokollen eine universelle Möglichkeit zur Verständigung zu geben. Geschaffen wurde der Standard von der OPC Task Force, einem Zusammenschluss verschiedener großer Firmen der Automatisierungsindustrie wie Fisher-Rosemount, Intellution und Siemens, nachdem man erkannt hatte, welchen Aufwand die Anpassung der zahlreichen Herstellerstandards auf individuelle Steuerungs- und Überwachungs-Infrastrukturen verursacht hatte.

Kurz nach der Veröffentlichung der OPC Specification Version 1.0 im August 1996 wurde die OPC Foundation gegründet, die bis heute zuständig ist für Pflege und Verbreitung des Standards. Ihr gehören mittlerweile 448 (Stand: 08.01.2008) Unternehmen an.

Heute ist OPC der Standard zur herstellerunabhängigen Kommunikation in der Automatisierungstechnik. Die Zertifizierungssoftware OPC Compliance Test welche den OPC-Mitgliedern kostenlos zur Verfügung gestellt wird, stellt die Kompatibilität sicher. Die Hersteller von OPC-Servern können damit ihre Server schon während der Entwicklung testen. Diese Software testet die vollständige OPC-Funktionalität, simuliert Fehlverhalten eines Clients und überprüft alle Fehlercodes. Zusätzlich werden noch logische Tests, Stress- und Performance-Tests durchgeführt. Diese Testreihe deckt mehr Tests ab, als man mit einem normalen Client erreicht. Nach bestandenem Test können die Hersteller die Ergebnisse an die OPC Foundation senden und erhalten das Zertifikat Compliance Tested. Es ist nicht zu empfehlen, Server zu kaufen, die dieses Zertifikat nicht besitzen.

A.2.2. Einsatzgebiet

OPC wird dort eingesetzt, wo Sensoren, Regler und Steuerungen verschiedener Hersteller ein gemeinsames, flexibles Netzwerk bilden. Ohne OPC benötigten zwei Geräte zum Datenaustausch genaue Kenntnis über die Kommunikationsmöglichkeiten des Gegenübers. Erweiterungen und Austausch gestalten sich entsprechend schwierig. Mit OPC genügt es, für jedes Gerät genau einmal einen OPC-konformen Treiber zu schreiben. Idealerweise wird dieser bereits vom Hersteller zur Verfügung gestellt. Ein OPC-Treiber lässt sich ohne großen Anpassungsaufwand in beliebig große Steuer- und Überwachungssysteme integrieren.

OPC unterteilt sich in verschiedene Unterstandards, die für den jeweiligen Anwendungsfall unabhängig voneinander implementiert werden können. OPC lässt sich damit verwenden für Echtzeitdaten (Überwachung), Datenarchivierung, Alarm-Meldungen und neuerdings auch direkt zur Steuerung (Befehlsübermittlung).

A.2.3. Spezifikationen

OPC gliedert sich in die folgenden Spezifikationen:

- OPC DA (Data Access): Spezifikation zur Übertragung von Echtzeitwerten über OPC (DCOM basierend). Aktueller Stand der Spezifikation ist 3.0. OPC DA war die erste OPC Spezifikation.
- OPC A/E (Alarms and Events): Spezifikation zur Übertragung von Alarmen und Ereignissen (Alarms & Events).
- OPC HDA (Historical Data Access): Spezifikation zur Übertragung historischer Werte.
- OPC DX (Data eXchange): Spezifikation zur direkten Kommunikation zwischen OPC Servern.
- OPC Command: Spezifikation zur Ausführung von Befehlen (= Kommandos).
- OPC XML DA: Spezifikation zur XML- basierten Übertragung von Echtzeitwerten. Diese Spezifikation ist der Vorläufer von OPC UA. Da früh bekannt wurde, dass eine DCOM-unabhängige Spezifikationen in Form von OPC UA geplant ist, verbreiteten sich OPC XML DA Server nur gering.
- OPC UA (Unified Architecture): Spezifikation die alle bisherigen Spezifikationen (OPC DA, OPC HDA, OPC A/E) plattform- und DCOM- unabhängig umsetzen soll. Bisher (Stand Juni 2007) ist diese Spezifikation noch nicht abgeschlossen.

A.2.4. Funktionsweise

Für die Kommunikation zwischen den Anwendungen benutzt OPC derzeit hauptsächlich Microsofts DCOM Technologie (Distributed Component Object Model). Dank DCOM ist es für OPC-Anwendungen transparent, ob die über OPC ausgetauschten Daten von einer Anwendung im eigenen Adressraum, von einem fremden, lokalen Prozess oder auch von einem entfernt über TCP/IP angebundenen Rechner kommen. Die Übertragungs- und Zugriffsgeschwindigkeiten werden dabei DCOM-üblich kaum von unnötigem Verwaltungsaufwand ausgebremst. Die Kommunikationswege sind Abbildung 35 dargestellt.

DCOM macht anderen Anwendungen, (kompilierte) Funktionen und Objekte zugänglich. Der OPC-Standard definiert nun bestimmte DCOM-Objekte, d.h. die Funktionen/Schnittstellen, die ein OPC-Teilnehmer (über DCOM) zur Verfügung stellen muss, um mit anderen OPC-Anwendungen Daten austauschen zu können. Die für eine Implementierung notwendigen genauen Spezifikationen lassen sich frei auf der Seite der OPC Foundation herunterladen.

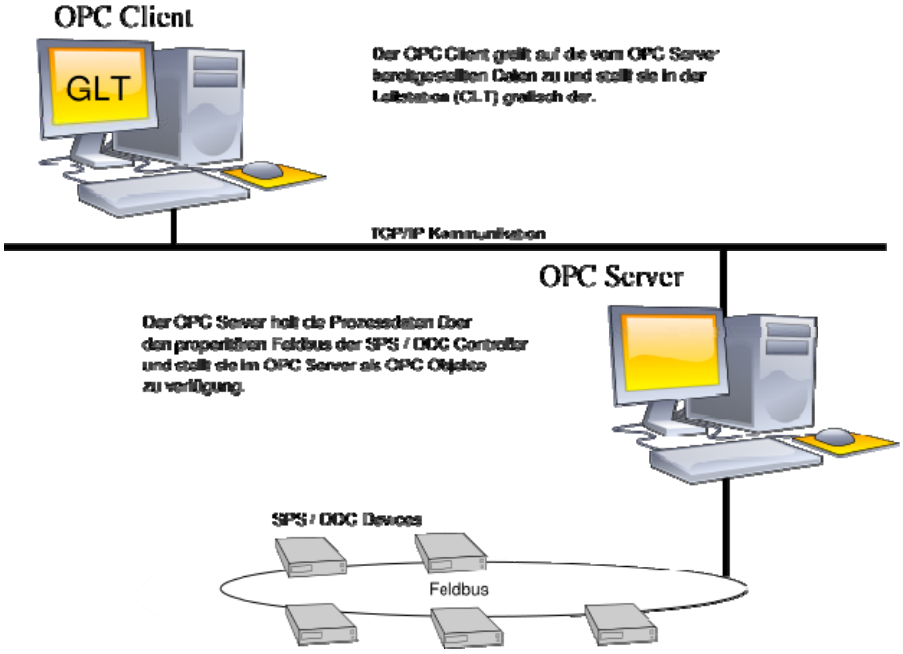


Abbildung 35 Darstellung des Datenaustausches (Quelle: Wikipedia)

Die wenigsten am Markt vorhandenen OPC-Server und OPC-Clients sind durch die OPC Foundation zertifiziert, da dieser Prozess Geld kostet. Der größte Kostenblock ist der jährliche Mitgliedsbeitrag an die OPC Foundation. Das Werkzeug zur eigenen Zertifizierung eines OPC-Server ist im Rahmen der Mitgliedschaft kostenlos erhältlich. Die Liste der zertifizierten Server/Clients ist auf der Seite der OPC Foundation zu finden. Zum Debuggen der Kommunikation zwischen Client und Server gibt es kostenlose Software, die sich als Sniffer zwischen den Kommunikationspartnern einklinkt.

Mit OPC XML-DA wurde die erste Webservice-basierte Schnittstelle geschaffen. Die Funktionalität ist ähnlich der normalen Data-Access-Schnittstelle, welche die erste und immer noch wichtigste Schnittstelle bei OPC ist. Mit dem Webservice steht OPC auch auf anderen Plattformen wie z. B. Linux zur Verfügung. Mit Webservice-Toolkits wie gSOAP, EasySoap++, Qt, etc. für C/C++ oder Java kann man so sehr schnell OPC XML-DA Client und Server entwickeln. Viele Hersteller von OPC-Servern haben als ersten Schritt Adapter entwickelt, welche OPC XML-DA Aufrufe einfach auf die vorhandenen COM OPC DA Server abbilden. Im Gegensatz zu DCOM verwenden Webservices Port 80 (HTTP) was es ebenfalls einfacher macht, durch Firewalls zu kommunizieren oder den Datenverkehr zu tunneln (SSH).

OPC Unified Architecture beschreibt eine neue Generation von OPC-Servern. Diese Spezifikation befindet sich noch in Entwicklung und soll die bisherigen Spezifikationen Data Access, Alarm & Events, Historical Data Access, Data eXchange, Batch und Security vereinheitlichen. Es wird nur noch einen Adressraum mit Objekten geben, die Werte beinhalten, Alarme senden, eine Historie besitzen und wie bei DX verschaltet werden können. Die bisher recht unterschiedlichen Browse-Interfaces werden so durch ein einheitliches Browsing ersetzt. Diese neue Spezifikation beschreibt kein COM-Interface mehr, sondern eine WSDL (Web Services Description Language), die nach COM und in verschiedene Webservice-Protokolle umgesetzt werden kann, womit die Portabilität sichergestellt wird. Ebenso wird verstärkt auf Skalierbarkeit und Sicherheit Wert gelegt.

A.2.5. Kritikpunkte

OPC basiert (bis auf wenige Spezifikationen) auf Microsofts DCOM Spezifikation. Somit ist eine Kommunikation über die Grenzen von Firewalls oder Domänen nur unter Einsatz sogenannter OPC Tunnel möglich. Diese Softwareprodukte wandeln die OPC Kommunikation in "normale" TCP/IP Kommunikation um, transportieren sie übers Netz und wandeln im Zielrechner die TCP/IP Kommunikation wieder in OPC Kommunikation um.

A.3. Datenmodelle in INTERLIS 2

A.3.1. Verkehrszähler Intervallsummen

(a) Variante 1

```
INTERLIS 2.3;

MODEL ZaehlSum (de) AT "http://www.gis.ethz.ch" VERSION "20080406"  =
  !! Variante 1
  IMPORTS UNQUALIFIED INTERLIS, UNQUALIFIED Units;     !!Variante 1

  TOPIC ZS_MM660 =

    DOMAIN
      BatteryStruct = TEXT*10;
      TimeDay = TEXT*14;
      Laengenintervall = (GE0_LT270_cm, GE270_LT700_cm, GE700_LT1300
      , GE1300_LT2500_cm);
      swiss10 = (Bus, Motorrad, Personenwagen, PWmitAnhaenger, Lieferwagen
      , LieferwagenmitAnhaenger, LieferwagenmitAuflieger, Lastwagen
      , Lastenzug, Sattelzug);

    CLASS HeaderMM660 =
      Format : 0 .. 255;
      Formatter : TEXT*30;
      Instrument : TEXT*50;
      FileName : TEXT*30;
      Site : 10000000 .. 99999999;
      Location : TEXT*40;
      GriDef : TEXT*20;
      Headings : TEXT*20;
      StartRec : TimeDay;
      StopRec : TimeDay;
      Battery : BatteryStruct;
      Sensors : TEXT*50;
      DateForm : TEXT*8;
      Units : (metrisch, amerikanisch, andere);
      Interval : 1 .. 240 [min];
      Peaktime : TEXT*20;
      Peakint : 1 .. 20 [min];
      Channels : TEXT*20;
      IntSpec : TEXT*40;
      LenBins : TEXT*50;
      Classification : TEXT*50;
      SiteDescr1 : TEXT*50;
      SiteDescr2 : TEXT*100;
    END HeaderMM660;

    CLASS Messung =
      Datum : 10100 .. 311299;
      Zeit : 0 .. 2359;
      Spur : 1 .. 8;
      Rtg : (keinVerk_0, normal_1, gegen_2, undef_9);
    END Messung;

    CLASS FzKatS =
      FzTyp : swiss10;
      FzSum : 1 .. 9999;
    END FzKatS;
```

```
    CLASS LaengenS =
      LTyp : Laengenintervall;
      LSum : 1 .. 9999;
    END LaengenS;

    ASSOCIATION R_HeadMMR_MsgMM =
      R_HeadMM -<#> {1} HeaderMM660;
      R_MsgMM -- {0..*} Messung;
    END R_HeadMMR_MsgMM;

    ASSOCIATION R_LgSR_MsgL =
      R_LgS -- {0..*} LaengenS;
      R_MsgL -<#> {1} Messung;
    END R_LgSR_MsgL;

    ASSOCIATION R_MsgFzR_FzS =
      R_MsgFz -<#> {1} Messung;
      R_FzS -- {0..*} FzKatS;
    END R_MsgFzR_FzS;

  END ZS_MM660;

END ZaehlSum.
```

(b) Variante 4

```
INTERLIS 2.3;

MODEL ZaehlSum (de) AT "http://www.gis.ethz.ch" VERSION "20080217"  =
  IMPORTS INTERLIS,Units;

  TOPIC ZS_MM660 =

    DOMAIN
      BatteryStruct = TEXT*10;
      Laengenintervall = (
        GE0_LT270_cm, GE270_LT700_cm, GE700_LT1300, GE1300_LT2500_cm);
      swiss10 = (Bus, Motorrad, Personenwagen, PWmitAnhaenger,
        Lieferwagen, LieferwagenmitAnhaenger, LieferwagenmitAuflieger,
        Lastwagen, Lastenzug, Sattelzug);
      TimeDay = TEXT*14;

    STRUCTURE S_Num =
      Anz : MANDATORY 0 .. 9999;
    END S_Num;

    CLASS FzKSMessung =
      Site : MANDATORY 10000000 .. 99999999;
      Location : TEXT*40;
      ZaehlerId : MANDATORY 1 .. 99999999;
      Datum : MANDATORY 10100 .. 311299;
      Zeit : MANDATORY 0 .. 2359;
      Spur : MANDATORY 1 .. 8;
      Rtg : 0 .. 99;
      RecTyp : MANDATORY 1 .. 2;
      FzKSum : LIST {10} OF S_Num;
    END FzKSMessung;

    CLASS LgSMessung =
      Site : MANDATORY 10000000 .. 99999999;
      Location : TEXT*40;
      ZaehlerId : MANDATORY 1 .. 99999999;
      Datum : MANDATORY 10100 .. 311299;
      Zeit : MANDATORY 0 .. 2359;
      Spur : MANDATORY 1 .. 8;
      Rtg : 0 .. 99;
      RecTyp : MANDATORY 1 .. 2;
      LgSum : LIST {4} OF S_Num;
    END LgSMessung;

  END ZS_MM660;

END ZaehlSum.
```

A.3.2. Streckenstation

```
INTERLIS 2.3;

MODEL StrStat (de) AT "http://www.gis.ethz.ch" VERSION "20080407"  =
  IMPORTS UNQUALIFIED Units,UNQUALIFIED INTERLIS;

  TOPIC StSt_ABBSAJA =

    UNIT

      FahrGeschwindigkeit [kmh] = (km / h);

    DOMAIN

      FahrtRichtung = (keinVerk_0, normal_1, gegen_2, undef_9);

      swiss10 = (Bus, Motorrad, Personenagen, PWmitAnhaenger, Lieferwagen,
        LieferwagenmitAnhaenger, LieferwagenmitAuflieger, Lastwagen,
        Lastenzug, Sattelzug);

    CLASS ErrorStrSt =
      StrStNr : 1 .. 10;
      StoergAllg : BOOLEAN;
      Batt : BOOLEAN;
      StrSt_ASTRA : BOOLEAN;
      ResNr : TEXT*20;
      ResVal : TEXT*20;
    END ErrorStrSt;

    CLASS ErrorVerkZ =
      VZNr : 1 .. 8;
      Komm : BOOLEAN;
      Para : BOOLEAN;
      Spg : BOOLEAN;
      LpNr : 1 .. 2;
      LpError : BOOLEAN;
    END ErrorVerkZ;

    CLASS Spurbezeichnung =
      SpurNr : 0 .. 16;
      Date_Day : 1 .. 31;
      Date_Month : 1 .. 12;
      Date_Year : 1990 .. 2020;
      Time_Hour : 0 .. 23 [h];
      Time_Min : 0 .. 59 [min];
      Time_Sec : 0 .. 59 [s];
      Cycle : 30 .. 3600 [s];
      Mov_Av : TEXT*20;
      Av_Speed : 0 .. 255 [kmh];
      VD_Total : 0 .. 200;
      B_Grad : 0 .. 100;
      V_Status : 0 .. 6;
      F_Richtung : FahrtRichtung;
      Av_Speed_M : 0 .. 255 [kmh];
      Av_Speed_T_M : 0 .. 255 [kmh];
      VerkehrsFreqz : 0 .. 4000;
      VD_Total_M : 0 .. 200;
      B_Grad_M : 0 .. 100;
      V_Status_M : 0 .. 6;
      F_Richtung_M : FahrtRichtung;
```

```
    END Spurbezeichnung;

    CLASS VehicleFrq =
      VclType : swiss10;
      VclNumber : 0 .. 4000;
    END VehicleFrq;

    CLASS VclFrqSmo =
      VclType : swiss10;
      VclNumSmo : 0 .. 4000;
    END VclFrqSmo;

    CLASS SpeedFrq =
      VclType : swiss10;
      SpdMV : 0 .. 255 [kmh];
      SpdSD : 0 .. 255 [kmh];
    END SpeedFrq;

    CLASS SpdFrqSmo =
      VclType : swiss10;
      SpdMVSmo : 0 .. 255 [kmh];
      SpdSDSmo : 0 .. 255 [kmh];
    END SpdFrqSmo;

    ASSOCIATION R_SpBezVFR_VF =
      R_SpBezVF -<#> {1} Spurbezeichnung;
      R_VF -- {0..*} VehicleFrq;
    END R_SpBezVFR_VF;

    ASSOCIATION R_SpBezVFSR_VFS =
      R_SpBezVFS -<#> {1} Spurbezeichnung;
      R_VFS -- {0..*} VclFrqSmo;
    END R_SpBezVFSR_VFS;

    ASSOCIATION R_SpBezSFR_SF =
      R_SpBezSF -<#> {1} Spurbezeichnung;
      R_SF -- {0..*} SpeedFrq;
    END R_SpBezSFR_SF;

    ASSOCIATION R_SpBezSFSR_SFS =
      R_SpBezSFS -<#> {1} Spurbezeichnung;
      R_SFS -- {0..*} SpdFrqSmo;
    END R_SpBezSFSR_SFS;

  END StSt_ABBSAJA;

END StrStat.
```

A.3.3. Bereichsrechner VDE

```
INTERLIS 2.3;

MODEL BerRechVDE (de) AT "mailto:hgnaegi@localhost" VERSION "2008-05-04" =
  IMPORTS UNQUALIFIED INTERLIS, UNQUALIFIED Units;

  DOMAIN
    ereignisPrio = 0 .. 8;
    LKoord = COORD 400000 .. 999999 [m],
                   0 .. 300000 [m],
            ROTATION 2 -> 1;
    prioStufeTyp = (normal, dringlich);

  STRUCTURE datenPunktKennung =
    anlageId : MANDATORY TEXT*100;
    datenPunktId : MANDATORY TEXT*100;
    aks : TEXT*100;
    bezeichnung : TEXT*100;
  END datenPunktKennung;

  STRUCTURE zeitSpanne =
    startZeit : MANDATORY XMLDateTime;
    endZeit : MANDATORY XMLDateTime;
  END zeitSpanne;

  STRUCTURE mengenAngabe =
    zahl : MANDATORY -1000000000 .. 1000000000;
    massEinheit : MANDATORY TEXT*50;
    intervall :  zeitSpanne;
  END mengenAngabe;

  STRUCTURE wertAngabe =
    waTyp : MANDATORY (ma, stu, txt, uhrStel);
    menge :  mengenAngabe;
    stufe : TEXT*50;
    beschreibung : TEXT*100;
    uhrStellung : XMLDateTime;
    MANDATORY CONSTRAINT
      (waTyp == #ma AND DEFINED(menge)) OR
      (waTyp == #stu AND DEFINED(stufe)) OR
      (waTyp == #txt AND DEFINED(beschreibung)) OR
      (waTyp == #uhrStel AND DEFINED(uhrStellung));
  END wertAngabe;

  TOPIC Telegramm =

    CLASS AktivMeldung =
      anlass : MANDATORY (Start, Fortgang);
      checkup : MANDATORY (alles_ok, Probleme, Konfiguration);
    END AktivMeldung;

    CLASS Antwort =
      anfrageLaufNr : MANDATORY 1 .. 999999999;
      anfrageErgebnis : MANDATORY (DatenPunktInfoListe, Stellwerte
      , BefehlsQuittung,Messungen, Ereignisse, LogbuchAuszug, Versionen);
    END Antwort;


    CLASS Befehl =
    END Befehl;
```

```
CLASS BefehlsQuittung =
  befQTyp : MANDATORY (OK, nOK, busy);
END BefehlsQuittung;

CLASS BenutzerInfo =
  userName : MANDATORY TEXT*100;
  benutzerGruppe : MANDATORY TEXT*100;
  passwort : MANDATORY TEXT*100;
END BenutzerInfo;

CLASS DatenPunktInfo =
  kennung : MANDATORY  datenPunktKennung;
  url : URI;
  urlELD : URI;
  urlKapoNSU : URI;
  koordinaten : LKoord;
  dpInfoTyp : (StellGroessenInfo, MessPunktInfo, EreignisPunktInfo);
END DatenPunktInfo;

CLASS DatenPunktInfoAbfrage =
  abfrageY : MANDATORY BOOLEAN;
END DatenPunktInfoAbfrage;

CLASS DPEinstellungenSetzen =
  kennung : MANDATORY  datenPunktKennung;
  mfTyp : MANDATORY (sendeAbst, aus);
  sendeAbstand : 1 .. 3600 [s];
  MANDATORY CONSTRAINT
    mfTyp == #sendeAbst AND DEFINED(sendeAbstand);
END DPEinstellungenSetzen;

CLASS EingangsPufferLeeren =
  leerenY : MANDATORY BOOLEAN;
END EingangsPufferLeeren;

CLASS Ereignis =
  kennung : MANDATORY  datenPunktKennung;
  prioELD : MANDATORY ereignisPrio;
  prioKapo : MANDATORY ereignisPrio;
  prioNSU : MANDATORY ereignisPrio;
  ereignisZeit : MANDATORY XMLDateTime;
  ereignisStatus : MANDATORY (offen, zu);
  ereignisArt : MANDATORY (Betrieb, Technik);
  ereignisAlarm : MANDATORY (Flanke, Status);
  fallNr : MANDATORY 1 .. 999999999;
  url : MANDATORY URI;
  urlELD : MANDATORY URI;
  urlKapoNSU : MANDATORY URI;
  koordinaten : LKoord;
END Ereignis;

CLASS EreignisAbfrage =
  epfTyp : MANDATORY (alle, dpLst);
  dpListe :  datenPunktKennung;
  zfTyp : MANDATORY (aktuell, zeitInt);
  zeitIntervall :  zeitSpanne;
  MANDATORY CONSTRAINT
    (epfTyp == #dpLst AND DEFINED(dpListe)) AND
    (zfTyp == #zeitInt AND DEFINED(zeitIntervall));
END EreignisAbfrage;
```

```
CLASS EreignisPunktInfo =
  prioELD : MANDATORY ereignisPrio;
  prioKapo : MANDATORY ereignisPrio;
  prioNSU : MANDATORY ereignisPrio;
  pmtOffen : MANDATORY MTEXT*160;
  pmtZu : MANDATORY TEXT*160;
END EreignisPunktInfo;

CLASS Kopf =
  von : MANDATORY  datenPunktKennung;
  an : MANDATORY  datenPunktKennung;
  laufNr : MANDATORY 1 .. 999999999;
  prio : MANDATORY prioStufeTyp;
  versandZeit : MANDATORY XMLDateTime;
END Kopf;

CLASS LogbuchAbfrage =
  zeitIntervall : MANDATORY  zeitSpanne;
END LogbuchAbfrage;

CLASS LogbuchEintrag =
  kennung : MANDATORY datenPunktKennung;
  logZeit : MANDATORY XMLDateTime;
  logText : MANDATORY TEXT*200;
END LogbuchEintrag;

CLASS Login =
  loginHost : MANDATORY TEXT*100;
  userNameLokal : MANDATORY TEXT*100;
  userNameRemote : MANDATORY TEXT*100;
  passwortRemote : MANDATORY TEXT*100;
END Login;

CLASS Meldung =
END Meldung;

CLASS Messung =
  kennung : MANDATORY  datenPunktKennung;
  messZeit : MANDATORY XMLDateTime;
  messFolgeNr : MANDATORY 1 .. 999999999;
  ablesung :  wertAngabe;
  ablesgExistY : MANDATORY BOOLEAN;
  pmt : TEXT*160;
  url : URI;
  urlELD : URI;
  urlKapoNSU : URI;
  koordinaten : LKoord;
  MANDATORY CONSTRAINT
    ablesgExistY AND DEFINED(ablesung);
END Messung;

CLASS MesswertAbfrage =
  mpfTyp : MANDATORY (alle, unterdrueckungen, dpLst);
  dpListe :  datenPunktKennung;
  zfTyp : MANDATORY (EnumElement81);
  zeitIntervall :  zeitSpanne;
  MANDATORY CONSTRAINT
    (mpfTyp == #dpLst AND DEFINED(dpListe)) AND
    (zfTyp == #zeitInt AND DEFINED(zeitIntervall));
END MesswertAbfrage;
```

```
CLASS Stellwert =
  kennung : MANDATORY  datenPunktKennung;
  ablesung :  wertAngabe;
  ablesgExistY : MANDATORY BOOLEAN;
  pmt : TEXT*160;
  url : URI;
  urlELD : URI;
  urlKapoNSU : URI;
  koordinaten : LKoord;
  MANDATORY CONSTRAINT
    ablesgExistY AND DEFINED(ablesung);
END Stellwert;

CLASS StellwertAbfrage =
  sgfTyp : MANDATORY (alle, alleUhren, dpListe);
  dpListe :  datenPunktKennung;
  MANDATORY CONSTRAINT
    sgfTyp == #dpLst AND DEFINED(dpListe);
END StellwertAbfrage;

CLASS StellwertSetzen =
  kennung : MANDATORY  datenPunktKennung;
  setzwert : MANDATORY  wertAngabe;
END StellwertSetzen;

CLASS StufenInfo =
  stufenRang : MANDATORY 0 .. 255;
  stufenText : MANDATORY TEXT*50;
  pmt : MANDATORY TEXT*160;
END StufenInfo;

CLASS TextInfo =
  textwert : MANDATORY TEXT*100;
  maxLen : MANDATORY 0 .. 255;
  pmt : MANDATORY TEXT*160;
END TextInfo;

CLASS UhrInfo =
  pmt : MANDATORY TEXT*160;
END UhrInfo;

CLASS Version =
  VersTypGeraet : MANDATORY TEXT*100;
END Version;

CLASS VersionenAbfrage =
  abfrageY : MANDATORY BOOLEAN;
END VersionenAbfrage;

CLASS ZahlwertInfo =
  massEinheit : MANDATORY TEXT*50;
  untereSchranke : -1000000000 .. 1000000000;
  obereSchranke : -1000000000 .. 1000000000;
  pmt : MANDATORY TEXT*160;
  mitIntervall : BOOLEAN;
END ZahlwertInfo;

ASSOCIATION AssociationDef102 =
  RoleDef103 -- {0..*} Ereignis
     OR Messung
     OR Stellwert
     OR AktivMeldung;
  RoleDef104 -- {0..*} Meldung;
END AssociationDef102;
```

```
    ASSOCIATION AssociationDef122 =
      RoleDef123 -- {0..*} ZahlwertInfo
         OR StufenInfo
         OR EreignisPunktInfo
         OR TextInfo
         OR UhrInfo;
      RoleDef124 -- {0..*} DatenPunktInfo;
    END AssociationDef122;

    ASSOCIATION AssociationDef84 =
      RoleDef85 -- {0..*} Login
         OR LogbuchAbfrage
         OR EingangsPufferLeeren
         OR StellwertSetzen
         OR DPEinstellungenSetzen
         OR DatenPunktInfoAbfrage
         OR StellwertAbfrage
         OR MesswertAbfrage
         OR BenutzerInfo
         OR VersionenAbfrage
         OR EreignisAbfrage;
      RoleDef86 -- {0..*} Befehl;
    END AssociationDef84;

    ASSOCIATION AssociationDef93 =
      r_ergebnis -- {0..*} DatenPunktInfo
         OR BefehlsQuittung
         OR Messung
         OR Ereignis
         OR Messung
         OR Stellwert
         OR Version
         OR LogbuchEintrag;
      r_Antwort -- {1} Antwort;
    END AssociationDef93;

  END Telegramm;

END BerRechVDE.
```

A.3.4. Verkehrszähler Einzelfahrzeuge (Typ MM660)

(a) Variante 1

```
INTERLIS 2.3;

MODEL ZaehlEfz (de) AT "http://www.gis.ethz.ch" VERSION "20080903"  =
  IMPORTS INTERLIS,Units;

  DOMAIN
   swiss10 = (Bus, Motorrad, Personenwagen,PWmitAnhaenger,Lieferwagen
   LieferwagenMitAnhaenger, LieferwagenMitAufleger, Lastwagen,
   Lastenzug, Sattelzug);

  TOPIC ZE_MM660 =

    CLASS EfzData =
      EfzNr : MANDATORY 1 .. 999999;
      Datum : MANDATORY 10100 .. 311299;
      ZeitStdMin : MANDATORY 0 .. 2359;
      ZeitSek : MANDATORY 0 .. 59 [INTERLIS.s];
      ZeitHstSek : MANDATORY 0 .. 99;
      ErgebnisCode : MANDATORY TEXT*6;
      Spur : MANDATORY 1 .. 8;
      Rtg : MANDATORY 0 .. 9; !! (keinVerk_0, normal_1, gegen_2, undef_9);
      AbstFrontFront : MANDATORY 0.1 .. 99.9 [INTERLIS.s];
      AbstHeckFront : MANDATORY 0.1 .. 99.9 [INTERLIS.s];
      Geschw : MANDATORY 0 .. 299 [Units.kmh];
      FzLaenge : MANDATORY 100 .. 9999 [Units.cm];
      FzTyp : MANDATORY 1 .. 10;   !! ZaehlEfz.swiss10;
      ChassisHoehe : MANDATORY (VL, L , M, H);
    END EfzData;

    CLASS EfzHeadMM660 =
      Site : MANDATORY 10000000 .. 99999999;
      Sensors : TEXT*50;
      PRUnits : TEXT*30;
      Classification : TEXT*30;   !! swiss10
      OSPFilter : (all, selection, others);
      OSPTime : 0 .. 60 [INTERLIS.min];
      OSPVEH : 0 .. 60 [INTERLIS.min];
      EfzDataHeader : TEXT*70;
    END EfzHeadMM660;

    ASSOCIATION A_HeadMM_EfzD =
      R_HeadME -<#> {1} EfzHeadMM660;
      R_EfzDH -- {0..*} EfzData;
    END A_HeadMM_EfzD;

  END ZE_MM660;

END ZaehlEfz.
```

(b) Variante 2

```
INTERLIS 2.3;

MODEL ZaehlEfz (de) AT "http://www.gis.ethz.ch" VERSION "20080829"  =
  IMPORTS INTERLIS,Units;

  DOMAIN
    swiss10 = (Bus, Motorrad, Personenwagen, PWmitAnhaenger, Lieferwagen,
      LieferwagenMitAnhaenger, LieferwagenMitAufleger, Lastwagen,
      Lastenzug, Sattelzug);

  TOPIC ZE_MM660 =

    CLASS EfzData =
      Site : MANDATORY 10000000 .. 99999999;
      EfzNr : MANDATORY 1 .. 999999;
      Datum : MANDATORY 10100 .. 311299;
      ZeitStdMin : MANDATORY 0 .. 2359;
      ZeitSek : MANDATORY 0 .. 59 [INTERLIS.s];
      ZeitHstSek : MANDATORY 0 .. 99;
      ErgebnisCode : MANDATORY TEXT*6;
      Spur : MANDATORY 1 .. 8;
      Rtg : MANDATORY 0 .. 9; !! (keinVerk_0, normal_1, gegen_2, undef_9);
      AbstFrontFront : MANDATORY 0.1 .. 99.9 [INTERLIS.s];
      AbstHeckFront : MANDATORY 0.1 .. 99.9 [INTERLIS.s];
      Geschw : MANDATORY 0 .. 299 [Units.kmh];
      FzLaenge : MANDATORY 100 .. 9999 [Units.cm];
      FzTyp : MANDATORY 1 .. 10; !! ZaehlEfz.swiss10;
      ChassisHoehe : MANDATORY (VL, L , M, H);
    END EfzData;

  END ZE_MM660;

END ZaehlEfz.
```

A.3.5. Verkehrszähler Einzelfahrzeuge (Typ ECTN)

```
INTERLIS 2.3;

MODEL ZaehlEfz_ECTN (de) AT "http://www.gis.ethz.ch" VERSION "20080903"  =
  IMPORTS INTERLIS, Units;

  DOMAIN

      swiss10 = (Bus, Motorrad, Personenwagen, PWmitAnhaenger , Lieferwagen
      Lieferwagenmit Anhaenger, LieferwagenmitAufleger, Lastwagen,
      Lastenzug, Sattelzug
    );

  TOPIC ZE_ECTN =

    CLASS EfzECTNData =
      Datum : MANDATORY TEXT*10;
      ZeitStdMinSek : MANDATORY FORMAT INTERLIS.XMLTime "0:0:0" .. "23:59:59";
      ZeitMiliSek : MANDATORY 0 .. 999;
      Kategorie : MANDATORY 1 .. 10;   !! swiss10
      Spur : MANDATORY 1 .. 8;
      Fahrtrichtg : MANDATORY TEXT*5;
      Geschwindkt : MANDATORY 0.0000 .. 80.0000 [Units.ms];
      Laenge : MANDATORY 1000 .. 99999 [Units.mm];
      Hoehe : MANDATORY 1000 .. 9999 [Units.mm];
      Breite : MANDATORY 1000 .. 2999 [Units.mm];
      AnzGefGutSchi : MANDATORY 0 .. 5;
      GefahrgutNr : -1 .. 5;
      SubstanzNr : -1 .. 20;
    END EfzECTNData;

  END ZE_ECTN;

END ZaehlEfz_ECTN.
```

A.3.6. Input VM-CH-System, harmonisierte Struktur von Einzelfarzeugdaten (Vorschlag)

```
INTERLIS 2.3;

MODEL VMCH_Mod (de) AT "http://www.gis.ethz.ch" VERSION "20080829"  =
  IMPORTS Units,INTERLIS;

  DOMAIN

    swiss10 = (Bus, Motorrad, Personenwagen, PWmitAnhaenger, Lieferwagen,
      LieferwagenMitAnhaenger, LieferwagenMitAufleger, Lastwagen,
      Lastenzug, Sattelzug
    );

  TOPIC VMCH_Top =

    CLASS VMCH =
      Site : 0 .. 99999999;
      Datum : MANDATORY INTERLIS.XMLDate;
      Zeit : MANDATORY INTERLIS.XMLTime;
      FzTyp : MANDATORY VMCH_Mod.swiss10;
      Spur : MANDATORY 1 .. 8;
      FahrRtg : MANDATORY (keine_0, normal_1, gegen_2, beide_9);
      Geschw : MANDATORY 0.0000 .. 250.0000 [Units.kmh];
      FzLaenge : MANDATORY 1000 .. 99999 [Units.mm];
      FzHoehe : 1000 .. 9999 [Units.mm];
      FzBreite : 1000 .. 2999 [Units.mm];
      AnzGefGutSchi : 0 .. 5;
      AbstFroFro : 1.0 .. 99.9 [INTERLIS.s];
      AbstHeFro : 1.0 .. 99.9 [INTERLIS.s];
      ChassisH : (VL, L, M, H);
    END VMCH;

  END VMCH_Top;

END VMCH_Mod.
```

A.4. Beschreibung der Standardformate in XML-Schema

In Kapitel 6 wird anhand eines Beispiels beschrieben, nach welchen standardisierten Regeln man vom konzeptionellen Datenmodell zum Standard Transferformat gelangt. Es wird dort erwähnt, dass der INTERLIS Compiler eine exakte und computer-bearbeitbare Beschreibung des Standardformats auf Wunsch automatisch liefert. Zur Beschreibung dieses Standardformats (INTERLIS 2 XML) wird die allgemein zur Beschreibung von XML-Formaten eingesetzte Sprache „XML-Schema“ verwendet. Diese wird selber im XML Format geschrieben, wobei gewisse Einschränkungen zu beachten sind. So gibt es verschiedene reservierte Attributsnamen in tags (wie **`schema, annotation, appinfo, element, sequence, complexType`** etc.) und weitere reservierte Attributsnamen (wie **`name, type, use, minOccurs`** etc.). Jeder tag-Name beginnt mit **`xsd:`** . Die Beschreibungssprache XML-Schema wird durch das World Wide Web Consortium W3C entwickelt und normiert. Die Dokumentation ist frei im Internet verfügbar und zu finden unter

`http://www.w3.org/TR/`

Die Formatbeschreibungen in XML-Schema sind sehr umfangreich. Wir beschränken uns daher auf ein einziges Beispiel. Im Folgenden ist die vollständige Formatbeschreibung aufgeführt für die Daten des Einzelfahrzeugzählers Typ Marksman 660, das in Kapitel 6 ausführlich besprochen ist (Abbildungen 23, 24, 25). Zu Beginn der Formatbeschreibung sind die Codierungen der INTERLIS Sprachelemente beschrieben. Diese werden benötigt um – ab der drittletzten Seite, nach zwei Leerzeilen – die Codierung der Attribute der einzigen Klasse **`EfzData`** unseres Datenmodells zu definieren.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.interlis.ch/INTERLIS2.3" xmlns:ili2c="http://www.interlis.ch/ili2c" target-
Namespace="http://www.interlis.ch/INTERLIS2.3" elementFormDefault="qualified" attributeFormDe-
fault="unqualified">
  <xsd:annotation>
    <xsd:appinfo source="http://www.interlis.ch/ili2c/ili2cversion">3.6.8-20080825</xsd:appinfo>
    <xsd:appinfo source="http://www.interlis.ch/ili2c">
      <ili2c:model>Units</ili2c:model>
      <ili2c:modelVersion>2005-06-06</ili2c:modelVersion>
      <ili2c:modelAt>http://www.interlis.ch/models</ili2c:modelAt>
    </xsd:appinfo>
    <xsd:appinfo source="http://www.interlis.ch/ili2c">
      <ili2c:model>ZaehlEfz</ili2c:model>
      <ili2c:modelVersion>20080829</ili2c:modelVersion>
      <ili2c:modelAt>http://www.gis.ethz.ch</ili2c:modelAt>
    </xsd:appinfo>
  </xsd:annotation>
    <xsd:element name="TRANSFER" type="Transfer"/>
    <xsd:simpleType name="IliID">
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="([a-zA-Z_][0-9a-zA-Z_\-\.]*:)?[0-9a-zA-Z_][0-9a-zA-Z_\-\.]*"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="Transfer">
      <xsd:sequence>
        <xsd:element name="HEADERSECTION" type="HeaderSection"/>
        <xsd:element name="DATASECTION" type="DataSection"/>
      </xsd:sequence>
    </xsd:complexType>
    <!-- ALIAS TABLE
      <ENTRIES FOR="ZaehlEfz">
        <TAGENTRY FROM="ZaehlEfz.ZE_MM660" TO="ZaehlEfz.ZE_MM660"/>
        <TAGENTRY FROM="ZaehlEfz.ZE_MM660.EfzData" TO="ZaehlEfz.ZE_MM660.EfzData"/>
      </ENTRIES>
        ALIAS TABLE -->
    <xsd:complexType name="HeaderSection">
```

```
  <xsd:sequence>
    <xsd:element name="MODELS" type="Models"/>
    <xsd:element name="ALIAS" type="Alias" minOccurs="0"/>
    <xsd:element name="OIDSPACES" type="OidSpaces" minOccurs="0"/>
    <xsd:element name="COMMENT" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="VERSION" type="xsd:decimal" use="required" fixed="2.3"/>
  <xsd:attribute name="SENDER" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="Models">
  <xsd:sequence>
    <xsd:element name="MODEL" type="Model" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Model">
  <xsd:attribute name="NAME" type="INTERLIS.NAME" use="required"/>
  <xsd:attribute name="VERSION" type="xsd:string" use="required"/>
  <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
</xsd:complexType>
<xsd:complexType name="Alias">
  <xsd:sequence>
    <xsd:element name="ENTRIES" type="Entries" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Entries">
  <xsd:sequence>
    <xsd:choice maxOccurs="unbounded">
      <xsd:element name="TAGENTRY" type="Tagentry"/>
      <xsd:element name="VALENTRY" type="Valentry"/>
      <xsd:element name="DELENTRY" type="Delentry"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="FOR" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="Tagentry">
  <xsd:attribute name="FROM" type="xsd:string" use="required"/>
  <xsd:attribute name="TO" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="Valentry">
  <xsd:attribute name="TAG" type="xsd:string" use="required"/>
  <xsd:attribute name="ATTR" type="xsd:string" use="required"/>
  <xsd:attribute name="FROM" type="xsd:string" use="required"/>
  <xsd:attribute name="TO" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="Delentry">
  <xsd:attribute name="TAG" type="xsd:string" use="required"/>
  <xsd:attribute name="ATTR" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="OidSpaces">
  <xsd:sequence>
    <xsd:element name="OIDSPACE" type="OidSpace" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="OidSpace">
  <xsd:attribute name="NAME" type="xsd:string" use="required"/>
  <xsd:attribute name="OIDDOMAIN" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="CoordValue">
  <xsd:sequence>
    <xsd:element name="C1" type="xsd:double"/>
    <xsd:element name="C2" type="xsd:double" minOccurs="0"/>
    <xsd:element name="C3" type="xsd:double" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ArcPoint">
  <xsd:sequence>
    <xsd:element name="C1" type="xsd:double"/>
    <xsd:element name="C2" type="xsd:double"/>
    <xsd:element name="C3" type="xsd:double" minOccurs="0"/>
    <xsd:element name="A1" type="xsd:double"/>
    <xsd:element name="A2" type="xsd:double"/>
    <xsd:element name="R" type="xsd:double" minOccurs="0"/>
```

```
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="RoleType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="REF" type="IliID" use="required"/>
      <xsd:attribute name="BID" type="IliID"/>
      <xsd:attribute name="ORDER_POS" type="xsd:positiveInteger"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="DataSection">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="ZaehlEfz.ZE_MM660" type="ZaehlEfz.ZE_MM660"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="INTERLIS.HALIGNMENT">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Left"/>
    <xsd:enumeration value="Center"/>
    <xsd:enumeration value="Right"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="INTERLIS.VALIGNMENT">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Top"/>
    <xsd:enumeration value="Cap"/>
    <xsd:enumeration value="Half"/>
    <xsd:enumeration value="Base"/>
    <xsd:enumeration value="Bottom"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="INTERLIS.URI">
  <xsd:restriction base="xsd:normalizedString">
    <xsd:maxLength value="1023"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="INTERLIS.NAME">
  <xsd:restriction base="xsd:normalizedString">
    <xsd:maxLength value="255"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="INTERLIS.INTERLIS_1_DATE">
  <xsd:restriction base="xsd:normalizedString">
    <xsd:maxLength value="8"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="INTERLIS.STANDARDOID">
  <xsd:attribute name="OID" type="IliID" use="required"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.METAOBJECT">
  <xsd:sequence>
    <xsd:element name="Name" type="INTERLIS.NAME"/>
  </xsd:sequence>
  <xsd:attribute name="TID" type="IliID" use="required"/>
  <xsd:attribute name="BID" type="IliID"/>
  <xsd:attribute name="OPERATION" type="xsd:string"/>
  <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.METAOBJECT_TRANSLATION">
  <xsd:sequence>
    <xsd:element name="Name" type="INTERLIS.NAME"/>
    <xsd:element name="NameInBaseLanguage" type="INTERLIS.NAME"/>
  </xsd:sequence>
  <xsd:attribute name="TID" type="IliID" use="required"/>
  <xsd:attribute name="BID" type="IliID"/>
  <xsd:attribute name="OPERATION" type="xsd:string"/>
  <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.AXIS">
```

```
  <xsd:sequence>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.REFSYSTEM">
  <xsd:sequence>
    <xsd:element name="Name" type="INTERLIS.NAME"/>
  </xsd:sequence>
  <xsd:attribute name="TID" type="IliID" use="required"/>
  <xsd:attribute name="BID" type="IliID"/>
  <xsd:attribute name="OPERATION" type="xsd:string"/>
  <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.COORDSYSTEM">
  <xsd:sequence>
    <xsd:element name="Name" type="INTERLIS.NAME"/>
    <xsd:element name="Axis">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="INTERLIS.AXIS" type="INTERLIS.AXIS" maxOccurs="3"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="TID" type="IliID" use="required"/>
  <xsd:attribute name="BID" type="IliID"/>
  <xsd:attribute name="OPERATION" type="xsd:string"/>
  <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.SCALSYSTEM">
  <xsd:sequence>
    <xsd:element name="Name" type="INTERLIS.NAME"/>
  </xsd:sequence>
  <xsd:attribute name="TID" type="IliID" use="required"/>
  <xsd:attribute name="BID" type="IliID"/>
  <xsd:attribute name="OPERATION" type="xsd:string"/>
  <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.SIGN">
  <xsd:sequence>
    <xsd:element name="Name" type="INTERLIS.NAME"/>
  </xsd:sequence>
  <xsd:attribute name="TID" type="IliID" use="required"/>
  <xsd:attribute name="BID" type="IliID"/>
  <xsd:attribute name="OPERATION" type="xsd:string"/>
  <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.TIMESYSTEMS.CALENDAR">
  <xsd:sequence>
    <xsd:element name="Name" type="INTERLIS.NAME"/>
  </xsd:sequence>
  <xsd:attribute name="TID" type="IliID" use="required"/>
  <xsd:attribute name="BID" type="IliID"/>
  <xsd:attribute name="OPERATION" type="xsd:string"/>
  <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.TIMESYSTEMS.TIMEOFDAYSYS">
  <xsd:sequence>
    <xsd:element name="Name" type="INTERLIS.NAME"/>
  </xsd:sequence>
  <xsd:attribute name="TID" type="IliID" use="required"/>
  <xsd:attribute name="BID" type="IliID"/>
  <xsd:attribute name="OPERATION" type="xsd:string"/>
  <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.TimeOfDay">
  <xsd:sequence>
    <xsd:element name="Hours" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="23"/>
        </xsd:restriction>
```

```
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Minutes" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="59"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Seconds" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:double">
          <xsd:minInclusive value="0.0"/>
          <xsd:maxInclusive value="59.999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.UTC">
  <xsd:sequence>
    <xsd:element name="Hours" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="23"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Minutes" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="59"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Seconds" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:double">
          <xsd:minInclusive value="0.0"/>
          <xsd:maxInclusive value="59.999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="INTERLIS.GregorianYear">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1582"/>
    <xsd:maxInclusive value="2999"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType  name="INTERLIS.GregorianDate">
  <xsd:sequence>
    <xsd:element name="Year" type="INTERLIS.GregorianYear" minOccurs="0"/>
    <xsd:element name="Month" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1"/>
          <xsd:maxInclusive value="12"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Day" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1"/>
          <xsd:maxInclusive value="31"/>
        </xsd:restriction>
      </xsd:simpleType>
```

```
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType  name="INTERLIS.GregorianDateTime">
  <xsd:sequence>
    <xsd:element name="Year" type="INTERLIS.GregorianYear" minOccurs="0"/>
    <xsd:element name="Month" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1"/>
          <xsd:maxInclusive value="12"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Day" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1"/>
          <xsd:maxInclusive value="31"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Hours" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="23"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Minutes" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="59"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Seconds" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:double">
          <xsd:minInclusive value="0.0"/>
          <xsd:maxInclusive value="59.999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>


<xsd:complexType  name="ZaehlEfz.ZE_MM660.EfzData">
  <xsd:sequence>
    <xsd:element name="Site">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1"/>
          <xsd:maxInclusive value="99999999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="EfzNr">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1"/>
          <xsd:maxInclusive value="999999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Datum">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="10100"/>
```

```
      <xsd:maxInclusive value="311299"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="ZeitStdMin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="2359"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="ZeitSek">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="59"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="ZeitHstSek">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="99"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="ErgebnisCode">
  <xsd:simpleType>
    <xsd:restriction base="xsd:normalizedString">
      <xsd:maxLength value="6"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="Spur">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="1"/>
      <xsd:maxInclusive value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="Rtg">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="9"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="AbstFrontFront">
  <xsd:simpleType>
    <xsd:restriction base="xsd:double">
      <xsd:minInclusive value="0.1"/>
      <xsd:maxInclusive value="99.9"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="AbstHeckFront">
  <xsd:simpleType>
    <xsd:restriction base="xsd:double">
      <xsd:minInclusive value="0.1"/>
      <xsd:maxInclusive value="99.9"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="Geschw">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="299"/>
```

```
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="FzLaenge">
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="100"/>
            <xsd:maxInclusive value="9999"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="FzTyp">
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="1"/>
            <xsd:maxInclusive value="10"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="ChassisHoehe">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="VL"/>
            <xsd:enumeration value="L"/>
            <xsd:enumeration value="M"/>
            <xsd:enumeration value="H"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="TID" type="IliID" use="required"/>
    <xsd:attribute name="BID" type="IliID"/>
    <xsd:attribute name="OPERATION" type="xsd:string"/>
    <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="ZaehlEfz.ZE_MM660">
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="ZaehlEfz.ZE_MM660.EfzData" type="ZaehlEfz.ZE_MM660.EfzData"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="BID" type="IliID" use="required"/>
    <xsd:attribute name="TOPICS" type="xsd:string"/>
    <xsd:attribute name="KIND" type="xsd:string"/>
    <xsd:attribute name="STARTSTATE" type="xsd:string"/>
    <xsd:attribute name="ENDSTATE" type="xsd:string"/>
    <xsd:attribute name="CONSISTENCY" type="xsd:string"/>
  </xsd:complexType>
</xsd:schema>
```