Modeling of Space-Related Data

An introduction with regard to UML and INTERLIS

(version complemented and adjusted to INTERLIS 2.3)

Editor

Coordination, Geo-Information and Services (COGIS)
c/o swisstopo, Federal Office of Topography
Seftigenstrasse 264, CH–3084 Wabern

www.kogis.ch

Authors

Joseph Dorfschmid · do@adasys.ch
Sascha Brawer · sb@adasys.ch

Adasys AG, Dörflistrasse 67, CH – 8050 Zürich
www.adasys.ch

Acknowledgements

The authors wish to express their gratitude to all those who have contributed to this paper with their suggestions, discerning reviews and precise lecture.

Translation

This user manual was initially drafted in German, the authors of its English version have attempted to respect as much as possible the original text.

Index

1. Reading between fiction and truth

Who would not know it, this impression: This is beyond me! And all these technical terms! Must it really be that complicated?

That is why «Modeling of Space-Related Data» approaches the topic from an amateur's point of view and tries to embed it in a story: The successful story of Ilis Valley. Ilis Valley is nowhere because it is invented. At the same time Ilis Valley is everywhere because the story depicts typical situations and question that we all sometimes know only too well. In this document Ilis Valley lies in Ahland, also fictional. This coming and going between the fictional world of Ilis Valley and the tough facts of theory and reality repeatedly takes place, for different levels of interest and knowledge.

♦ **Overview** – The overview in chapter 2 introduces the most important elements of data modeling and model-based data transfer.

♦ **Modeling methods** – Chapter 3 illustrates several known methods of modeling, presents the positions of both UML and INTERLIS and will show you where additional information is to be found.

♦ **The example Ilis Valley** – In chapter 4 the example of Ilis Valley is represented in more detail, thereby modeling possible concerns within the range of tourism. However we do not claim to define an appropriate solution. Our main focus is the representation of modeling principle by means of examples. References to additional explanations in chapters 5 to 8 ought to familiarize you with theory. For all these that already have a certain knowledge of data modeling and model-based data transfer this chapter is likely to prove an ideal starting point.

♦ **Discussion of the example** – Since inheritance is a central principle of the modeling method thus introduced, it is more closely treated in chapter 5. The following chapters deal with data modeling in general terms (chapter 6), with its effects on software packages in use (chapter 7) and with the data transfer (chapter 8). This may seem to you to be going too much into detail, in that case simply skip chapters 5 – 8. These chapters can also be read punctually. That is why the titles of the sub-chapters always composed of a part according to the story of Ilis Valley and of a technical term.

♦ **General Considerations** – Chapter 9 takes up some basic aspects that are to be considered when modeling. Thus its perusal may also be of interest to those that are less concerned with details.

♦ **Final report for authorities and management** – The final chapter 10 compiles once more the most important points (to the attention of the authorities of Ilis Valley and the management of public transports and tourism organizations. For the busy manager of everyday-life it may be best to start with this chapter and thus obtain an entry into the problems and possibilities of the matter.

2. An overview with Ilis Valley taken as an example

2.1 Awakening in Ilis Valley

2.1.1 The starting shot

The tourist region of Ilis Valley has decided to update their WebPages. The various possibilities of public transportation should be interactively read back by means of a graphic dialog. Its presentation in the town hall was impressive: Beautiful pictures, trendy abbreviations such as HTML, XML, GIS, SVG! Of course other requests arose almost immediately. Not very long ago it had been decided in the department of construction to list addresses according to a new norm. This ought to be put to advantage! The managing director of the Mount Ilis Alpine Transports remembered that the national association of alpine transports intended to establish a service, which would render available the alpine transport on offer in the entire country. It should also state fares and above all the different season tickets available for the area. His colleague, responsible for technical matters draws their attention to the fact that he supervises the entire infrastructure on his computer. No trouble to procure information concerning railway tracks and ski runs. Although there are some minor problems. Recently he had asked for data regarding the new development «Upper Crescent». In the end they had succeeded somehow, even though some pieces of information had been lost. However they were not that important.

That made the president of the township prick up her ears. A colleague working in a town, closer to the capital has told her that in one department they have already changed software for the third time. Additional demands that she would have deemed normal had caused a complete overhaul at every time. Then they had been supported by an information technologist who had not so much talked about different techniques but together they had thought about what data actually caused these problems. Since this new procedure was in use they gradually achieved better results.

Thus the president of the township, a friendly yet resolute woman, charges the building secretary to attend to this matter together with the technical manager of the Mount Ilis Alpine Transport. As her counselor she calls in the information technologist of her colleague.

2.1.2 First appraisal

During the first meeting of the work group different arguments and key words tumbled to and for. What software package was used by the national association? Do we only speak of the railway company or each individual railway, each ski lift? The individual railways as well as their buildings already appear in national surveying. How can these data be put to use? What happens if data change or if new data are added? But my software package only understands DXF! And me nothing at all!

At this point it was agreed to call a halt. Remembering the old Roman saying «Divide et impera» they tried to master the situation by first establishing some order. To start with, the following questions were in the foreground:

- ♦ Who needs what data?
- ♦ Who is in charge of their recording and update?

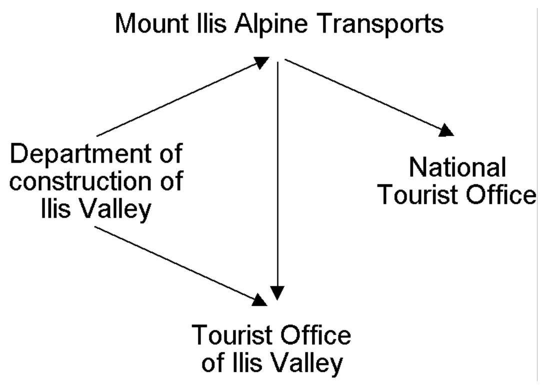

Mount Ilis Alpine Transports
Department of construction of Ilis Valley
National Tourist Office
Tourist Office of Ilis Valley

Figure 1: Parties concerned and flow of information.

But how are these data transferred from processor to user? E-mail, FTP, DXF, ASCII – and discussion started again. The information technologist recommended to put this question aside for the time being and to consider the modeling of these data instead. Modeling? What on earth had modeling to do with an IT-solution for Ilis Valley? The question what a data model actually is definitely was beyond the scope of their first meeting. Only that much be said: A data model describes the structure of data. What are the properties of the individual objects? Are there any relationships between objects? And this not as a literary work but in a clear, precise figurative or formal language!

2.2 First steps

2.2.1 The National Tourist Office has paved the way

The National Tourist Office has a software package NatTourSys that provides a survey of the available tickets of the different alpine transports. Tickets are issued by the individual railway companies. The tourist however is mainly concerned with their validity on different routes before starting their own project; the Ilis Valley crew wants to gain a general view.

One thing seems clear: In connection with this software package, alpine transports, railway companies as well as tickets have to be taken into consideration.

What is really meant by the term ‚ticket'? Is it the actual ticket that is sold? Surely not, it is the different types of tickets that ought to be described. Hence we had better speak of ticket types. It is obvious that the individual items have further properties: e.g. with the ticket types we can speak of price and validity.

- ♦ **Alpine transport** – An alpine transport conveys passengers between its bottom station and top station. One example of a means of alpine transport is the funicular Ilis Ville -

Mount Ilis. But there are also cog rails, aerial cable cars and gondolas, as well as ski and chair lifts. Even the new snow bus could be considered as an alpine transport. Each means of alpine transport has its own name.

- **Ticket type** – A ticket type is a certain sort of ticket. Examples of ticket types are the sport pass at 195 sovereigns, with a validity of seven days on all means of transport in the entire area of Ilis Valley, or the «Dino-Pass» at 10 sovereigns valid on the day of issue and for the pony lift only.
- **Railway company** – A railway company operates means of alpine transport. It has a name and sometimes also an abbreviation, such as the Mount Ilis Alpine Transports short MIT. Each company receives a certain percentage of the sale of those tickets that are valid on its lines. One railway company may be subsidiary or parent company of another.

The **object catalog** of an application lists all items that are of interest and describes them as accurately as possible in words.

If all characteristics of all items are to be described in words it will soon be very difficult to not to lose control. «A picture may mean more than a thousand words» will cross ones mind. An object diagram – that would be it! But all things considered it is not the individual object we wish to describe. Above all we want to show the similarity of items and their characteristics.

Such a diagram will depict quite well the essential:

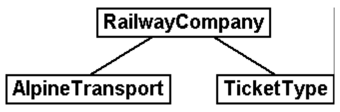


Figure 2: A first attempt of a data model.

AlpineTransport, TicketType and RailwayCompany are **object classes** (small boxes). **Relationships** exist between them (connecting lines). Together all definitions for the classes and their relationships form a **data model**. Its graphic representation ensues with **class diagrams**.

Terms related to object classes are: entity, table, type, ...

Terms related to relationships are: Association, reference, aggregation, (mutual) pointer, ...

Terms related to data model are: (conceptual) schema, data description, ...

Object classes are designated with nouns. We use the singular form to express that each individual object (e.g. each means of alpine transport) possesses the characteristics described with the class.

Each individual alpine transport, each railway company, each ticket type is described by a concrete **object**. These objects are the **data** whose structure and connections are described by the model.

Terms related to object are: example, instance, property, data set, line, tuple, entry, ...

Each means of alpine transport is run by a company. This company issues certain types of tickets. Without further specific information they must be valid on all railway companies. This however cannot always be satisfactory, because it is quite conceivable that bigger companies will issue types of tickets that are only valid on some of their lines. Thus it suggests itself that also a relationship between means of transport and ticket type be introduced. Hence we list for each ticket type the means of alpine transport it is valid for.

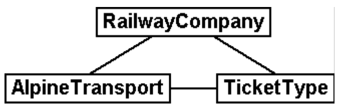


Figure 3: The data model has been expanded by the relationship means of alpine transport – TicketType.

However often several ticket types (e.g. day ticket, weekly season ticket, etc.) will be valid in the same area. With the model defined so far all assignations for each ticket type would have to be established individually. This is rather awkward and prone to errors. Most likely this is why the National Tourist Office has chosen a subtler model:

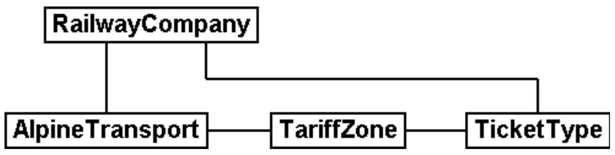


Figure 4: Revised data model. The bend in the connecting line between RailwayCompany and TicketType has no signification.

It will be worthwhile to consider first what object classes are necessary for your problem and what their mutual relationships may be. At this stage the characteristics of these objects are still relatively irrelevant. It is of more importance to find the accurate terms.

2.2.2 How many lines are run by one company?

Several means of alpine transport may be attributed to one company and vice versa one company may be assigned several lines. Several? How many exactly?

Cardinality records how many objects of one type can be assigned to the object of another type.

In the graphic both the minimum and maximum number of admissible other objects at the end of the connecting line is noted with the class of the other objects. If its number is unlimited we either add a (*) or omit an indication.

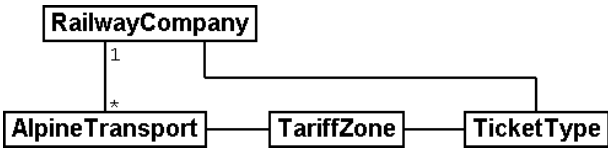


Figure 5: One AlpineTransport is rune by exactly one (1) company. On the other hand a company may run any number of (*) alpine transports.

2.2.3 Means of alpine transport, companies and subscription tickets have characteristics

Of course it is necessary for the application as it is planned to describe more in detail a means of alpine transport, company etc. A company will have a name and (typical with railway companies) an abbreviation (e.g. MountIlisAlpineTransport, MIT).

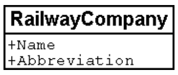


Figure 6: The object class Company with name and abbreviation.

- Name and abbreviation designate **attributes** of the object class company.
  Terms related with attribute are: column, field, property, ...

With these two attributes it is quite obvious what type they are: texts. With the price of a ticket type further indications will be of importance: franc, Euro, dollar, Ahland sovereign? Terming the period of validity will be even more demanding if it cannot simply be described with a number of days. If we indicate the length of a railway company it is naturally enough also of importance whether it is described in meters or kilometers. For the programs employed it is important to know how long the projected text attributes may be or within what range the projected numbers may lie.

- The **type** of attribute describes what values an attribute may take on and what is their significance.

A term related with type is value domain.

Object class RailwayCompany

Name:	Text
	Length: maximum 100 symbols
Abbreviation:	Text
	Length: maximum 10 symbols

Figure 7: The object class «RailwayCompany» possesses both a name and an abbreviation. The type of the property «Name» is a text with a maximum of one hundred symbols. For the property «Abbreviation» however only a maximum of ten symbols is admissible.

Nevertheless also other attribute types are easily conceivable:

Object class TicketType	
Name:	Text with a maximum of one hundred symbols
Price:	Number
	Precision: two decimals
	Admissible range: Between 0 and 5000
	Unity: Ahland Sovereigns

Figure 8: The object class TicketType with its properties and their types.

Unlike a ticket type or a railway company the bottom station of any line is an object that really exists at a certain place. It makes sense to describe localities by means of coordinates within a certain coordinate system such as e.g. the national system.

Object class AlpineTransport	
Name:	Text with a maximum of one hundred symbols
Position of bottom station:	Point
	Coordinate system: Ahland Projection Coordinates
Position of top station:	Point
	Coordinate system: Ahland Projection Coordinates

Figure 9: The object class AlpineTransport with its properties and their types.

Thus for each property we determine a suitable attribute type. In the case of a ski run its degree of difficulty is an enumeration, whereas the course of the run is a directed line in Ahland national coordinates. Details concerning the various types will be dealt with in chapter 6.

Object class SkiRun	
Course:	Directed line
	Coordinate system: Ahland projection coordinates
Degree of difficulty:	Enumeration
	Possible values: blue, red, black

Figure 10: The object class Ski Run with its properties and their types.

2.2.4 Models? It is data Ilis Valley is asking for!

After all these rather theoretical matters the people in charge in Ilis Valley insist upon deeds. An inquiry at the National Tourist Office resulted in the information that they would provide a simple program for the recording of data in accordance with their requirements. This would allow the export of data in INTERLIS-Format, which then could be sent to the National Tourist Office. The information technologist however argued that in this way at the most a first test would be possible and that the data should be stored either in the program package of the Mount Ilis Alpine Transports or in that of the department of construction. Nevertheless it was agreed to execute this test. After all it should not incur all that much work since neither are the Mount Ilis Alpine Transports that big nor is the number of ticket types that extensive.

⚠ Rushed actions only make sense if they really do not involve a lot of work.

The following means of alpine transports form the Mount Ilis Alpine Transports:

- Funicular Ilis Ville – Mount Ilis;
- Gondola Ilis Bath – Ilis Rock;
- Ski lift Ilis Rock – Mount Ilis;
- Chair lift Ilis Dale – Ilis Rock;
- Pony lifts in Ilis Ville and Ilis Bath.

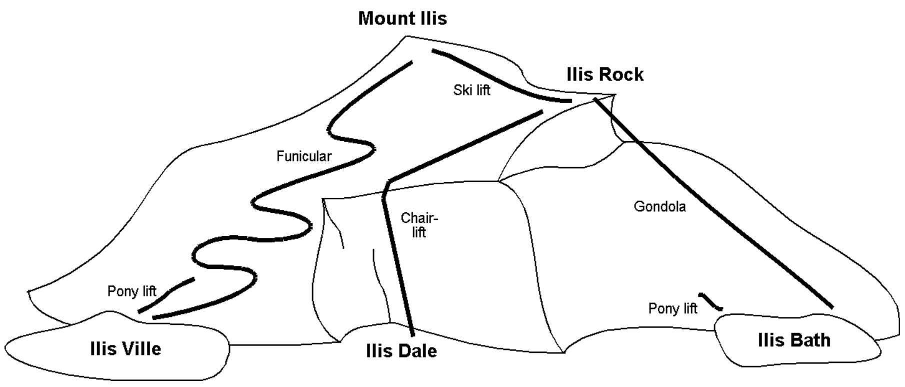


Figure 11: The Mount Ilis Alpine Transports operate several lines.

The Mount Ilis Alpine Transports issue the following ticket types:

- Individual tickets for the funicular (one way: 10 sovereigns; return-fare: 18 sovereigns);
- Individual tickets for the gondola (one way: 8 sovereigns; return-fare: 14 sovereigns);
- Hiker's Pass for the funicular and the gondola (price for one day 15 sovereigns; for seven days 55 sovereigns);
- The Sport Pass for all lines (price for one day: 40 sovereigns, for two days: 70 sovereigns, for seven days: 195 sovereigns; for the entire year: 635 sovereigns);
- The «Dino-Day Ticket» (10 sovereigns) and the «Weekly Ticket Ilosaurus Maximus» (45 sovereigns) for the pony lifts.

2.2.5 Ilis Valley transmits

Using the program for their test a file was generated containing all data.

- The simplest type of transfer is the **full transfer** completely transferring all data.

A quick look at the file revealed a lot that hardly seemed comprehensible but at least the texts «Mount Ilis Alpine Transports» could be read, followed by «MIT» and fares could also be easily found.

Just another test: The price for the area season ticket is lowered from 635 to 600 sovereigns and by means of the function update a new file is generated. The beginning may still look the same but the texts «Mount Ilis Alpine Transports» and «MIT» both are missing. However almost at the very end – this might be the new price!

- Thanks to **incremental update** it is not necessary to transfer all data after a modification but only the objects actually modified.

As agreed, both files were sent to the Tourist Office of Ilis Valley. And apparently could be read without any problems. Objection of the information technologist: This is not really astonishing. As long as we record the data exactly as required and furthermore with a program provided by the Tourist Office of Ilis Valley this was to be hoped for. But we people from Ilis Valley want more than that! Whenever possible we want to use our present program packages.

2.3 Ilis Valley wants more

2.3.1 Target

Thus Ilis Valley does not want to offer the same service as the National Tourist Office. Uppermost are the following additional performances:

- Indication of current operating and waiting times at the different railway companies and whether they can be used by hikers and with toboggans;
- Indication of all runs including degree of difficulty and condition;
- Graphic representation (including indication of forests and roads);
- Indication of all inns in the area;
- Indication where, depending on their postal address, buildings can be found.

2.3.2 Ilis Valley puts to use what is already existing

The data necessary for graphic representation of forests and roads should not have to be captured all over again, since the department of construction is in possession of all data of cadastral surveying, which also includes forests and roads. Moreover the department of construction has started to record addresses of buildings in accordance with the new norm. Hence it would make little sense to repeat all these definitions in the data model of Ilis Valley. It would be more convenient to simply use the existing models of cadastral surveying and of the addresses of buildings.

▶ A data model is not an isolated description; on the contrary it may be built upon already existing data models.

Terms related to data model in the sense of building-up are : Modules, packages, ...

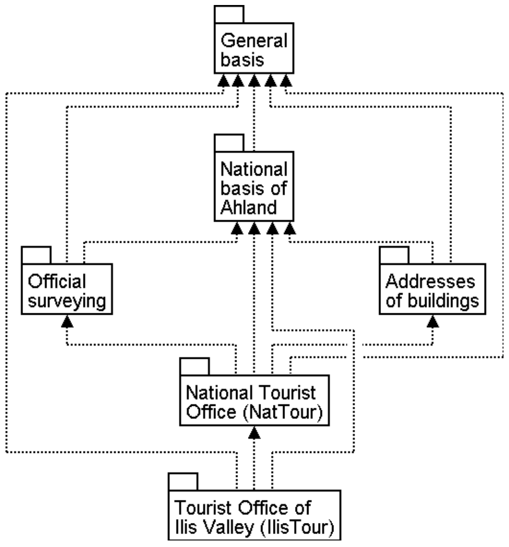


Figure 12: The tourism data model of Ilis Valley (IlisTour) need not make its own definitions. Instead it builds upon other models: It uses parts of the national tourism-model (NatTour), of the national basis of Ahland, of the cadastral surveying, of the addresses of buildings as well as common fundamentals. The dotted line with filled-in arrow means dependency. As in our example very often the common base is placed at the top, the special case at the bottom. However the opposite is also found.

2.3.3 Ilis Valley exceeds the National Tourist Office

Somehow the authorities of Ilis Valley did not want to use the model of the National Tourist Office in the given manner. In order to permit a graphic representation the track course also has to be described for every means of alpine transport. Furthermore they would like to display whether the conveyance can be used by hikers or with toboggans, its operating hours and the current waiting time. It seems obvious to define an individual class for the means of alpine transport of Ilis Valley. Should the attributes of the class AlpineTransport of the National Tourist Office be repeated therein? And there is also the relationship between AlpineTransport and TarifZone. What does a proper class mean for this relationship?

Luckily there is a thing called inheritance for such cases.

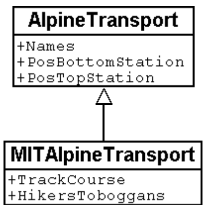


Figure 13: A MITAlpineTransport is a special means of AlpineTransport with additional attributes: TrackCourse as well as usability for HikersToboggans. The traced line with a white arrow means specialization.

- The class of the Ilis Valley MITAlpineTransport is an **extension** of the class of AlpineTransport. Thus it **inherits** all properties of alpine transports and adds others. [Details of inheritance will be described in chapter 5].

  Terms related to extension are: Specialization, sub class, …

Now would it be correct to also add the attributes OperatingHours and current WaitingTime to the class MITAlpineTransport? If the OperatingHours were a direct attribute of MITAlpine-Transport, then for each line one, typically the current operating hour, could be noted. However the managing director fixes the operating hours at the beginning of the season: In early season some lifts do not run yet, others take a lunch break; around the Christmas-holidays they run non-stop from 9a.m. until 3.30p.m.; starting in mid February – when days begin to lengthen – then operating hours gradually are extended until 4.30 p.m. Then again depending on snow and weather conditions some lines will shut down temporarily.

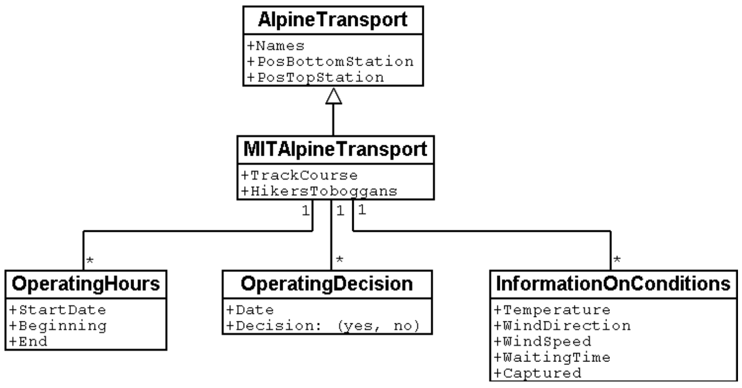


Figure 14: OperatingHours are not defined as independent objects.

If we define moreover that certain operating hours apply to several railway companies, then the costs of data collection can be reduced even further. Of course this does not make sense with waiting times. A waiting time noticed at a certain moment must be assigned to the line where passengers have to wait accordingly. Then why not recording the waiting time directly on the MITAlpineTransport? The following arguments explain why:

- When recording the waiting times as independent objects they can be evaluated at a later time (e.g. for statistics).

- ♦ The rhythm of modification and the responsibility for these values are quite different from the attributes of the MITAlpineTransport-class.

⚠ With properties that at first sight can be assigned to one class it always has to be considered whether this is really correct or whether they had not better be transferred to independent classes assigned via relationships.

With these considerations there are the real facts and not the usage e.g. for representations that is of relevance. However organizational conditions are of importance. Who is responsible for the update of data? How regularly will they be updated?

In the model of the National Tourist Office there are the individual railway companies that are responsible for the update of their data. The Ilis Valley model would like – as far as alpine transports are concerned – to use the model of the National Tourist Office but to extend it for the Mount Ilis Alpine Transports.

▶ Data models are divided into **Topics** to satisfy organizational conditions (e.g. different responsibilities and intervals of update).

That is why the Ilis Valley model extends the topic predefined by the National Tourist Office from Alpine Transports to MITAlpine Transport. In this local extension we define that the class MITAlpineTransport specializes the class AlpineTransport, which thus is extended by additional attributes.

Since operating hours, operating decisions and information on conditions are not only registered by different agencies but also at completely different intervals, they each are defined in individual topics (MITPlanning, MITOperation, MITCurrentEvents).

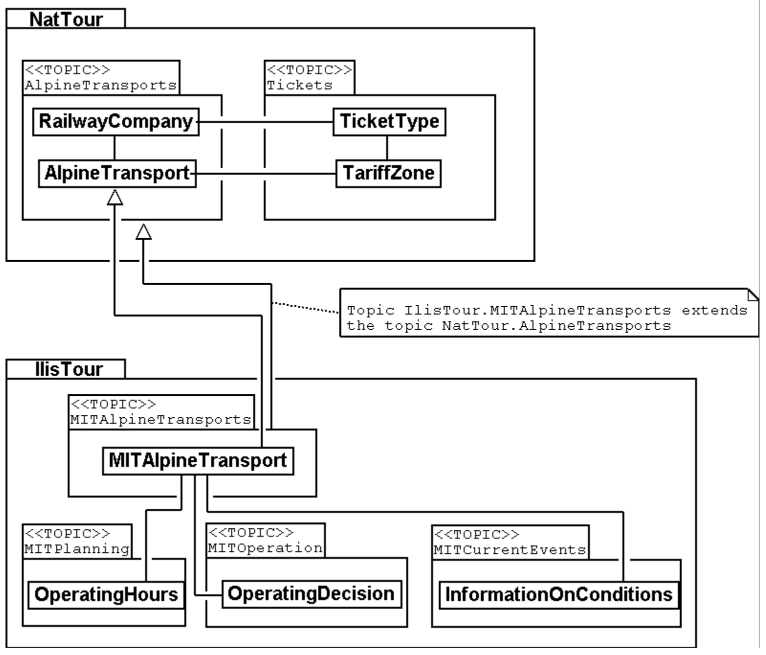


Figure 15: The Ilis Valley model (IlisTour) extends the model of the National Tourist Office (NatTour). IlisTour inherits the topic AlpineTransports from NatTour, extends the class AlpineTransport to MITAlpineTransport and adds further topics for planning, operation and current events.

- Inheritance does not only exist on a smaller scale (object classes), but also on a larger scale (entire topics).

2.3.4 Ilis Valley specialties

The authorities of Ilis Valley would also like to describe ski runs and restaurants. Hence they add further topics to the Ilis Valley model.

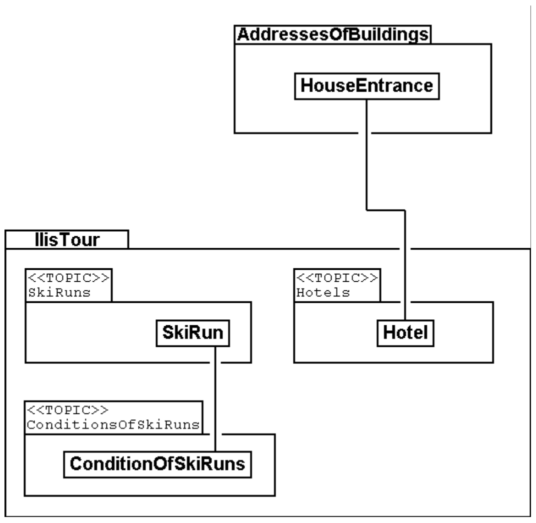


Figure 16: Further topics extend the Ilis Valley model of tourism.

Mainly with regard to the restaurants further questions arise. For example how is the fast-food INTERLUNCH to be represented graphically when it is well known that it is situated on 27, Village Road? But this does not permit the placing of a symbol on the map! The solution lies in the utilization of the building addresses. There the class HouseEntrance can be found that also features a position attribute (in national coordinates). That is why in the class Hotel no address is introduced; instead a relationship with the HouseEntrance is defined. To be concrete, the object that corresponds to the hotel Sun is linked to the house entrance object which describes 27, Village Road.

2.3.5 How can Ilis Valley implement their specialties?

A concept settles requirements but not their implementation. Where implementation is concerned we are basically free. The Mount Ilis Alpine Transports have decided on a standardized program package (LiftSys), this however can only process data in accordance with the extended model. Then again it is quite admissible to do without the class AlpineTransport and to insert its attributes directly in the class MITAlpineTransport.

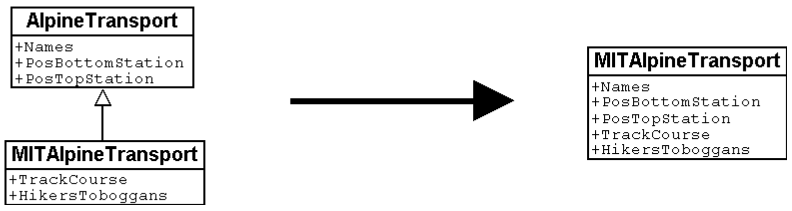


Figure 17: The program package for Ilis Valley tourism only has to roughly observe the conceptual model. For example it can combine two object classes into one, as long as the package is capable of supplying data in the format that corresponds to the conceptual model.

Analogous to the processing of classes according to the concept other questions arise as to how a certain computer system realizes ideas connected with the conceptual model.

2.3.6 How will Ilis Valley send their data to the National Tourist Office?

Once the LiftSys program package has been installed and all data has been captured, the question again arises as to how these data can be transmitted to the National Tourist Office. Because naturally enough it is not interested in all but only in certain data. The National Tourist Office for instance is neither interested in ski runs nor in their suitability for hikers and toboggans.

▶ An INTERLIS-data transfer always comprises data of one or several topics.

Hence in Ilis Valley they want to pass on to the National Tourist Office the data of the topics Alpine Transports and Tickets. But how can a program package generate a correct transfer file – when the manufacturing firm has had no knowledge whatsoever of the specifications of the National Tourist Office? The solution lies within the *model based transfer*.

▶ With a **model based transfer** there is no specific **transfer format**. On the contrary the format is governed by the data model.

Each modeling method (e.g. INTERLIS or the definitions which make up a certain program package) puts certain means of expression (object classes, attributes, types, relationships, tables, columns, etc.) at your disposal. For each of these means of expression its effect on the transfer is defined independently of the concrete data model. We only speak of a concrete transfer format (i.e. the exact order of the symbols which represent the respective data) when the corresponding data model is known. What is more, the transfer format is a direct result of the data model.

If LiftSys were capable of establishing the internal data model directly and in accordance with the conceptual data model, and if furthermore it supported the conversion of data within transfer files according to the specifications of INTERLIS, then there would be no problem at all. The transfer files could be generated in much the same simple way as with the test program of the union.

The program package of the department of construction (ConstSys) for example supports the generating of INTERLIS 2-conforming files. However it only knows some few tables which each consists of maybe several tables. Since the formatting rules of INTERLIS are organized in such a way that inheritance structure is not directly reflected in the transfer file so, it would be possible to directly generate correct files by using ConstSys. The conversion of internal to external data may be imagined in the following way:

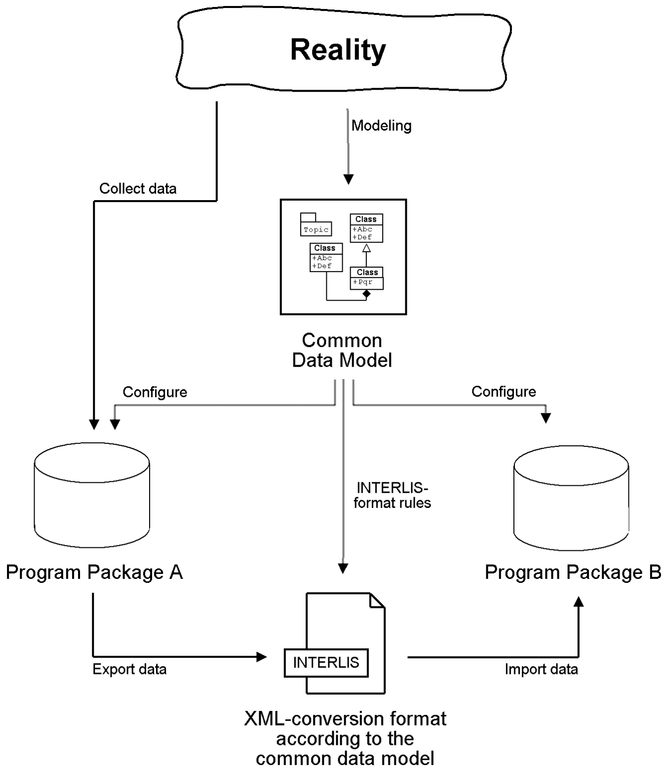


Figure 18: The internal data of program package A will be converted into a transfer file whose structure depends on the data model according to the INTERLIS format rules.
These data then will be imported into program package B, provided the program packages concerned have been configured in accordance with the data model.

However LiftSys does not support INTERLIS. What is to be done? Must MITAlpine Transport contemplate buying a new program package? There is an obvious alternative: LiftSys exports the data in another format, and then they will be with a conversion program according to INTERLIS. This conversion program can either be realized specifically for this concrete data model or more neutrally as a model-based tool.

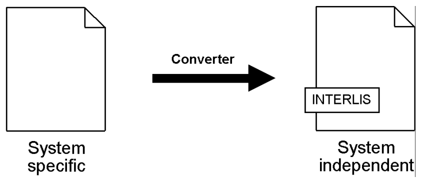


Figure 19: A converter generates INTERLIS-files from a format, which is specific for a certain computer system.

Once all seemed to go smoothly the file was sent to the National Tourist Office. The response: «Almost perfect – however there is a problem with the ski lift up to Ilis Rock!» Uff – somehow this is familiar and occurs repeatedly in e-mails: «Ilistäli» (Ilis Dale in German); these umlauts again.

Two things should be distinguished clearly:

- The **character codes utilized** determine which symbols can be used in text attributes.

- The **character coding** determines the bit pattern that represents the symbol within the computer.

Umlauts are part of the permitted character codes of INTERLIS. But with the conversion it was omitted to correctly indicate the character coding of the data provided by LiftSys. This correction having been made Ilis Valley receives a positive response from the National Tourist Office.

2.3.7 What does the national association of tourism do with the Ilis Valley data?

There is one small matter that slightly surprises the people in Ilis Valley: What might the computer system of the National Tourist Office (NatTourSys) have done with the supplementary attributes – such as the suitability for hikers and toboggans or the track course of the railway? The solution may sound simple: NatTourSys hat simply ignored them.

- Thanks to **polymorph reading** data can be read according to a «reduced» model, i.e. a model that does not yet recognize additional extensions.

Ilis Valley has transmitted all data in such a way that they contain all extensions according to the Ilis Valley model. The transfer rules of INTERLIS make sure that all the same these data can be read according to the model of the National Tourist Office without upsetting the reading program because of the additional data. Sole condition: the model according to which these data have been generated must be an extension of the model used at the receiving end. Thus the Ilis Valley model must extend the model of the National Tourist Office.

Chapter 5 further explains the usefulness of such extensions. Chapter 8 deals with the details of data transfers.

In the reading side it is possible to either read data directly with the program package of the receiver or to introduce a conversion program. And we also have to keep in mind that the concrete symbols of the text attributes must be converted correctly. The «ä» of Ilistäli may possibly be coded differently in LiftSys, on the transfer file and in NatTourSys. For each of the programs it is always obvious that it is an «ä».

2.4 Ilis Valley has made it

2.4.1 System overview

As far as the internet is concerned a relatively simple solution was opted for: The program package LiftSys generates the site plan as a static picture which then is put at the disposal of a web-presentation system (WebSys). In order to be able to require after the current condition of the railways, specific sections in the picture are accentuated. A mouse click on one of these spots will make the current data of the corresponding railway. Furthermore hotels with free rooms should be marked accordingly.

2.4.2 For the web site only the present condition is of interest

Ilis Valley has made an effort and their model now is clearly structured above all also for operational data of railways and ski runs. Unfortunately the program, which ought to continually update the internet site, is unable to derive the current condition from the multitude of operating hours, operating decisions and information on conditions. On one hand the management would like to receive data according to the topic MITTickets whenever a change has occurred. On the other hand information regarding operating conditions should be transmitted in 20-minute-intervals.

A **view** defines data that corresponds to the concept of a specific user and thus ought to be derivable from original data.

Related terms: View, derived data, ...

The view demanded links operating hours, operating decisions and waiting times with the one railway they are assigned to according to relationship and filter them in such a way that only the current conditions are described.

From the standpoint of the application view objects can be considered in much the same way as data-objects. That is why views can also be described by means of view classes.

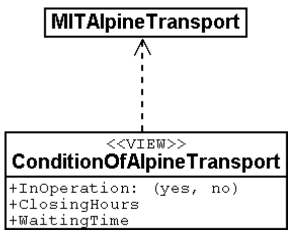


Figure 20: The condition of railways is not an independent object class, but is derived via a view from MITAlpineTransport. The view comprises such data as is necessary for the representation on an internet site.

2.4.3 How to represent hotels with available accommodation on the web site

For WebSys to be able to display as well which hotels have available rooms, of course the necessary information is required. Similarly to the conditions of railways, a view for hotels has to be defined. On one hand it contains the necessary data of hotels, on the other hand the horizontal coordinates according to the assigned house entrance.

Thanks to INTERLIS the necessary **symbols** can also be **defined system-neutrally** and the conversion of original- or view data can be described **graphically**.

Unfortunately WebSys is unable to process such conversion descriptions. However it is capable to read the symbol definitions. Furthermore it can receive data that state which symbol is represented at which spot and then execute this representation. Thus it is possible to put to advantage another facility, which is available on LiftSys.

▶ By means of INTERLIS it is also possible to transfer already converted graphic data.

That is why LiftSys does not provide WebSys with the view-data of the hotels, but undertakes their conversion into graphic data. Again the exact structure of the graphic data can be defined with classes. Typical attributes of such graphic data are position, symbol name and color.

3. Possibilities for the description of data

3.1 Description as a graphic: The Unified Modeling Language UML

A diagram is well suited to supply oneself or others with an overview to a model. That is why in information technology already since the mid 1970's graphic notation has been used to visualize models. Though for a long time it seemed impossible to agree on a uniform representation so that dozens of notations were in use at the same time. Finally at the beginning of the 1990's three methods (Booch, OMT and OOSE) prevailed. In 1995/96 their creators, the so-called «three amigos», agreed on one single modeling language the *Unified Modeling Language (UML)*. Since 1997 UML as a standard is being administered and maintained by an industrial syndicate, the *Object Management Group*.

UML is flexible and versatile. Whoever is familiar with its notation may grasp at first sight models from the most varied application ranges. By the way this language is not only suitable for the modeling of data (by means of the class diagrams presented in this paper). It also comprises elements for other aspects of a system for example the division in components or scenarios for its utilization.

An important factor for the success of UML is its adaptability: For many applications this language provides a sufficient infrastructure, nevertheless at any given moment it is possible to adapt this language to your proper needs by means of clearly defined equally standardized mechanisms of extension.

A visual language such as UML has its many advantages, however it is not the ideal tool for all applications. As soon as it is a matter of determining details of a large and complex model a drawing may become badly arranged and confusing. Besides, a graphic representation is rather unpractical e.g. where the arranging of computer systems is concerned. Hence in some cases a text form would be more suited: The models are formulated as a text in a standard language and thus become comprehensible for humans and computers likewise. Naturally enough the picture may still serve as a rough overview. At present suggestions for a purely textual notation of UML is being discussed, but until now no corresponding standard has been decided on.

3.2 Description as text: INTERLIS

In Switzerland it has been known for a long time that a model-based concept has its considerable advantages. Within the scope of cadastral surveying the textual modeling language INTERLIS was developed in 1991. Towards the end of the 1990's INTERLIS 2 was drawn up, which expanded and modernized the previous version. Have a dozen of specialists out of research, administration and industry ensured that this language would meet all demands of practical operation. Furthermore great importance was attached to the support to some particularities that are a result of the application:

- INTERLIS permits a precise and in the highest degree detailed description of data models. Hence an INTERLIS-model may directly become a component of an advertisement text, a contract or a decree and it is also possible to configure a program package based on an INTERLIS-model.
- INTERLIS does not only comprise a language for the modeling of data but also a procedure which permits the derivation of a transfer format for the exchange of the modeled data from a model. This is of major importance because the collection of geographical data is extremely costly and hence very valuable. Furthermore their life span comprises decades, in other words it is distinctly longer than that of information technology systems, which generally become antiquated within a few years. Such expensively collected data can only be protected if they are maintained in a form independent from a concrete program.
- Even though it is also possible to use INTERLIS in applications without any geographical reference whatsoever it is extremely well suited to space-related data. Already at the very core of the language geometry is supported. Moreover measures have been taken that permit the automatic examination of data supplied with regard to quality and plausibility.
- INTERLIS-data models have a clear structure: A model is divided into topics, which then comprise object classes. The description of an object class features object-properties as well as consistency and plausibility restraints. On both levels, i.e. models as well as topics, it is possible to declare commonly employed constructs (e.g. units, value domains, auxiliary structures, etc.).
- The federalist structure of Switzerland with its 26 cantons and semi-cantons, which are further divided into administrative units, makes high demands on a modeling language. INTERLIS 2 allows all parties concerned to use one, in its essence common data model, which then is expanded and refined according to regional needs. A smooth exchange of data is guaranteed even if locally different models are being used. Moreover INTERLIS permits that models and data can be drawn up in different languages. Thus each part of the country may use its own official language.

INTERLIS and UML complement one another: UML-diagrams are easy to draw and ideal for providing a rough overview. By using INTERLIS details may be captured in a precise and nevertheless easy to comprehend form, furthermore special support is offered where requirements of the federalist geodata-practice are concerned. There are tools that permit automatic conversion of UML-class diagrams into INTERLIS-descriptions. The reverse is also possible: the representation of an INTERLIS-model in UML-notation may take place automatically.

3.3 Standardized data models

There is little sense in establishing each data model completely from scratch. On the contrary predefined models can be used and refined. At different levels models have been and continue to be developed that may serve as a basis for applications:

- **International** – Within the scope of international endeavors for standardization a very comprehensive model for base geometry types (ISO 19107) has been developed. Current activities aim at getting to the very core of things, which would meet the most important requirements of practical operation (ISO 19137). For this spatial concepts of several common modeling languages have been analyzed, amongst others also those of INTERLIS. For instance whoever is concerned with the modeling of surfaces of a conic section or triangular networks is well advised to read the standard documents respectively the corresponding drafts. The series ISO-19100 also comprises data models for time intervals (ISO 19108), meta data (ISO 19115), services (ISO 19119), and more. However many of these standards still remain at planning stage and little can be said about their suitability for practical operation.
- **National** – In Switzerland standardized models exist for several fields of application. To some extent their use is prescribed in laws and regulations. Nation-wide valid models have been worked out for cadastral surveying, underground cadastre and recently also for geo-coded addresses.
- **Trade associations** – Trade associations are a further source of data models that serve as a basis for your own applications. For instance the Swiss Association of Engineers and Architects has developed an INTERLIS-model for the recording and registering of utility services (waste water, gas, water, electricity, telecom, cable communication).
- **INTERLIS** – INTERLIS predefines standard models for a few basic domains. For instance along with the INTERLIS-tools a model with units (units.ili) is supplied which defines meters, Newton, decibel etc.

As far as possible Swiss models are in keeping with ISO-standards. For example a national model for meta data concerning geographical data (e.g. origin, quality etc.) has been developed in accordance with ISO 19115. This model exists in the form of an INTERLIS-description.

Thanks to INTERLIS and its refinement–mechanisms a concrete application may fall back on an already existing data model without having to accept it *as it is*.

3.4 Formats for data exchange

At some point all data have to be memorized in a specific format so that they can be stored and exchanged between software packages. More recent transfer formats are all based upon the *Extensible Markup Language* (XML), a widespread format because of its extremely simple structure, its affinity with HTML (the format of web sites) and its derivation from SGML (a formerly frequently used format).

For instance the *Geographic Markup Language* (GML) is a strictly defined XML dialect propagated by OpenGIS, a manufacturer consortium of GIS-Software. INTERLIS, on the contrary, does not prescribe any specific XML dialect but supplies rules that allow the automatic derivation of an exchange format from the respective data model.

All things considered the question of the data format is secondary: As long as data to be transmitted are based upon similar models their conversion from one format into another requires little effort.

3.5 Important sources of information

Amongst others the following sources may provide further information:

- **www.interlis.ch** – The Internet portal to INTERLIS refers to documents, courses etc. in connection with INTERLIS. From the same address both the specification of this standard and the present text can be obtained electronically. Last but not least tools are also available on this web site. At present this is a computer program that permits the editing of INTERLIS-models in the graphic UML-notation («UML-Editor»), and a software library for the input, testing, altering and output of INTERLIS-models («INTERLIS-Compiler»).
- **www.omg.org** – The web site of the Object Management Group supplies the definition of the Unified Modeling Language. You may also find general documents (such as training materials or press texts) concerned with the model-based approach.
- **www.w3.org** – Amongst others the *World Wide Web Consortium* also supervises the specification of XML.

4. For example Ilis Valley in particular

4.1 Overview of the data model

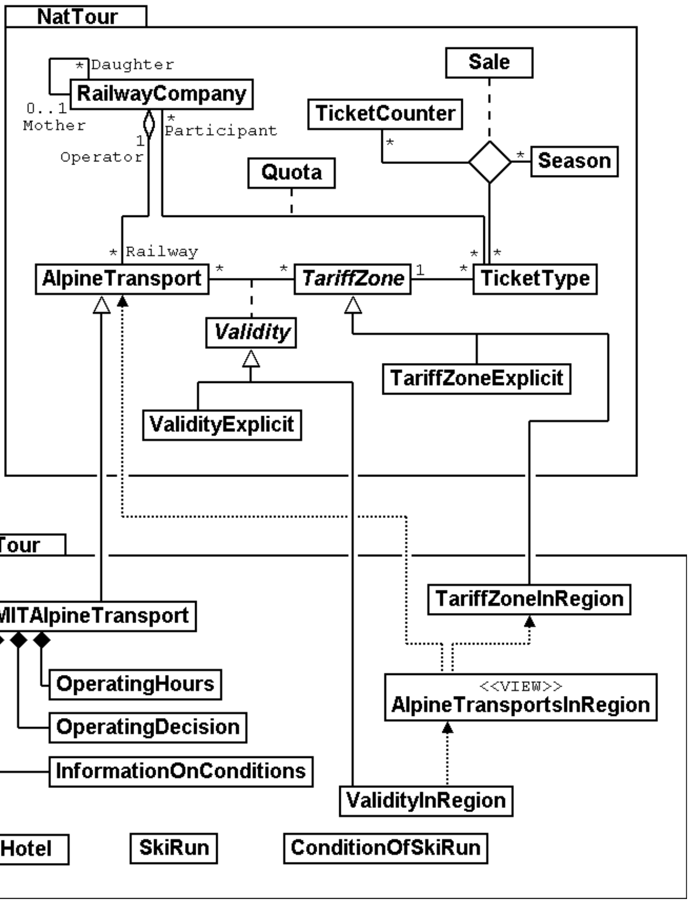


Figure 21: UML-representation of the data model. For the sake of clarity topics, structures, value domains and attributes do not appear. We refer to chapter 6, which deals more extensively with details.

4.2 The data model in INTERLIS

Subsequently you will find a complete rendering of the national basic model, the model of the National Tourist Office as well as the Illis Valley tourism model. There are extensive comments on each of these models. Furthermore boxes refer to paragraphs within the text that deal with a certain topic.

We recommend to only have a rough look at this example of a data model first, then to read chapters 5-8 where aspects of this data model will be treated in detail.

4.2.1 Ahland.ili – The national basic model

```
!! The following model is described in version 2.3 of INTERLIS.
INTERLIS 2.3;

!! The national basic model does not comprise any data but only defines
!! some types and units, in order that other models may refer to them.
TYPE MODEL Ahland AT "http://www.interlis.ch/models/ahland"
  VERSION "2008-01" =

  IMPORTS UNQUALIFIED INTERLIS;
  !! The national basic model is built upon the following basic models of INTERLIS.
  !! Units: units such as degrees Celsius
  !! CoordSys: basics for coordinate systems
  IMPORTS Units;
  IMPORTS CoordSys;

  !! Ilis Valley is situated in Ahland where the currency is sovereigns.
  !! The sovereign is a unit (UNIT) for money (MONEY). Even though the
  !! units-model (Units) already defines some currencies - such as the Swiss franc,
  !! the Euro and the US-dollar -, it does not name the Ahland sovereign.
  UNIT
    Sovereign EXTENDS MONEY;

  !! Time in Ahland is recorded in accordance with the local time zone AhlandTime.
  REFSYSTEM BASKET TimeSystems ~ INTERLIS.TIMESYSTEMS
    OBJECTS OF TIMEOFDAYSYS: AhT; !! AhlandTime

    !! The projection coordinate system of Ahland.
  REFSYSTEM BASKET CoordSystems ~ CoordSys.CoordsysTopic
    OBJECTS OF GeoCartesian2D: AhlandSys
    OBJECTS OF GeoHeight: AhlandHeightSys;

  DOMAIN
    NationalCoord = COORD 500.00 .. 91000.00 [m]
                            {CoordSystems.AhlandSys[1]},
                          700.00 .. 23000.00 [m]
                            {CoordSystems.AhlandSys[2]},
                          ROTATION 2 -> 1;
    NationalCoord3 = COORD 500.00 .. 91000.00 [m]
                             {CoordSystems.AhlandSys[1]},
                           700.00 .. 23000.00 [m]
                             {CoordSystems.AhlandSys[2]},
                          -200.00 .. 14000.00 [m]
                             {CoordSystems.AhlandHeightSys[1]},
                           ROTATION 2 -> 1;

  STRUCTURE LengthOfTime (ABSTRACT) =
  END LengthOfTime;
```

Units → 6.1.2

Coordinate Systems → 6.7.3

Value Domains → 6.6

Coordinate types → 6.7

3D-coordinates → 6.7.5

Length of time → 6.12

```
  STRUCTURE LengthOfTimeImplicit EXTENDS LengthOfTime =
    Duration: MANDATORY (Day, Week, Month, Year);
  END LengthOfTimeImplicit;

  STRUCTURE LengthOfTimeExplicit (ABSTRACT) EXTENDS LengthOfTime =
    Duration (ABSTRACT): MANDATORY NUMERIC [TIME];
  END LengthOfTimeExplicit;

  STRUCTURE LengthOfTimeInMinutes EXTENDS LengthOfTimeExplicit =
    Duration (EXTENDED): MANDATORY 0 .. 200 [Units.min];
  END LengthOfTimeInMinutes;

  STRUCTURE LengthOfTimeInDays EXTENDS LengthOfTimeExplicit =
    Duration (EXTENDED): MANDATORY 0 .. 1000 [Units.d];
  END LengthOfTimeInDays;

  STRUCTURE MomentInTime (ABSTRACT) =
  END MomentInTime;

  STRUCTURE AhlandTimeOfDay EXTENDS INTERLIS.TimeOfDay =
    Hours (EXTENDED): 0..23 CIRCULAR [INTERLIS.h];
  END AhlandTimeOfDay;

  DOMAIN
    AhlandXMLTimeOfDay = FORMAT BASED ON AhlandTimeOfDay
      ( Hours/2 ":" Minutes ":" Seconds );

END Ahland.
```

Inheritance → 5.

Moments in time → 6.12.5

4.2.2 Addresses.ili – The model for building addresses

We have not printed the model for building addresses as it would cover several pages and is not really of interest in connection with Ilis Valley. However the Ilis Valley tourism model establishes a relationship between hotels and the house entrances defined in the address model.

```
INTERLIS 2.3;

MODEL Addresses AT "http://www.interlis.ch/models/ahland"
  VERSION "2008-01" =


  TOPIC Buildings =

    CLASS HouseEntrance =
      !! ...
    END HouseEntrance;

  END Buildings;

END Addresses.
```

Reference to building addresses → 2.3.4

4.2.3 NatTour.ili – The model of the National Tourist Office

```
INTERLIS 2.3;

CONTRACTED MODEL NatTour AT "http://www.interlis.ch/models/ahland"
  VERSION "2008-01" =

  !! On its part the model of the National Tourist Office
  !! is built upon the national basic model of Ahland.
  IMPORTS Units, CoordSys, Ahland;
```

Using existing models → 6.16

```
  FUNCTION Multiply(factor1 : NUMERIC; factor2 : NUMERIC) : NUMERIC;

  !! A designation comprises a name plus the language
  !! in which this name
  !! has been written.
  STRUCTURE Designation =
```

Multilingual object characteristics → 6.11

```
    !! The name may be composed of any number of symbols.
    Name: TEXT;
```

Strings → 6.4

```
    !! Two letter-language code according to ISO 639.
    !! Examples: de = German, fr = French,
    !! it = Italian, rm = Romantsch, en = English.
    Language: TEXT*2;
  END Designation;


  TOPIC AlpineTransports =

    !! A railway designation is a common designation
    !! (but may not exceed 100 symbols), and comprises
    !! an abbreviation of the name such as "MIT"
    !! standing for MountIlisAlpineTransports.
```

Language dependent designation as structure elements → 6.11.2

```
    STRUCTURE RailwayDesignation EXTENDS Designation =
      Name (EXTENDED): TEXT*100;
      Abbreviation: TEXT*10;
    END RailwayDesignation;

    !! A railway company operates transport systems.
    CLASS RailwayCompany =
      !! The names of this company, if necessary in different
      !! languages. A minimum of one (1) name must be known,
      !! however no upper limit (*) restricts the number of names.
      Names: BAG {1..*} OF RailwayDesignation;
      !! Per language no more than one single railway
      !! designation: Thus the MountIlisAlpineTransports may
      !! only have one single Italian designation.
```

Uniqueness constraints → 6.14.3

```
      !! However this restriction only applies locally, in other words
      !! per railway company. After all the BlueMountainAlpineTransports
      !! should also be permitted to carry an Italian name.
    UNIQUE
      (LOCAL) Names : Language;
    END RailwayCompany;

    CLASS AlpineTransport =
      !! The names of this form of alpine transport, if necessary in different
      !! languages. A minimum of one (1) name must be known,
      !! however no upper limit (*) restricts the number of names.
      Names: BAG {1..*} OF Designation;
      PosBottomStation: Ahland.NationalCoord;
      PosTopStation: Ahland.NationalCoord;
```

Coordinate types → 6.7

```
    DurationOfTrip: Ahland.LengthOfTimeInMinutes;
    !! Exact kind of alpine transport.
    Kind: (CogRail,
           Funicular,
           AerialCableCar,
           SkiLift,
           ChairLift,
           Gondola);
  END AlpineTransport;
```

Types → 6.2

```
  ASSOCIATION =
    !! Indicates which means of transport are operated by one specific
    !! company. Example: The "MountIlisAlpineTransports" operate the
    !! funicular "Ilis Ville-Mount Ilis", the gondola
    !! "Ilis Bath-Ilis Rock" and the ski lift "Ilis Rock-Mount Ilis".
    !! A railway company may run an unlimited number {*} of alpine transports
    !! and there is always exactly one {1} operator per means of transport.
    !! The symbol -- stands for an ordinary
    !! relationship, -<> means that the strength of
    !! relationship is above the ordinary, a so-called
    !! aggregation.
    Operator -<> {1} RailwayCompany;
    Railway -- {*} AlpineTransport;
  END;
```

Force of a relationship → 6.13.2

```
  ASSOCIATION =
    Daughter -- {*} RailwayCompany;
    Mother -- {0..1} RailwayCompany;
  END;

END AlpineTransports;
```

Roles → 6.13.1

```
TOPIC Tickets =
  DEPENDS ON AlpineTransports;
  !! The nationally defined implicit durations of time are
  !! Day, Week, Month and Year. With tickets there is one
  !! more implicit duration, the season
  !! (for season tickets).

  STRUCTURE LengthOfTimeImplicit EXTENDS Ahland.LengthOfTimeImplicit =
    Duration (EXTENDED): (Season);
  END LengthOfTimeImplicit;
```

Length of time → 6.12

```
  !! An area within which a certain type of ticket is
  !! valid.
  CLASS TariffZone (ABSTRACT) =
  END TariffZone;

  CLASS TariffZoneExplicit EXTENDS TariffZone =
  END TariffZoneExplicit;

  !! One type of tickets, e.g. the "Ilosaurus-weekly ticket".
  CLASS TicketType =
    !! The names of this TicketType, if necessary in different languages.
    !! A minimum of one (1) names must be known, however there is no upper
    !! limit (*) of the number of names.
    Names: BAG {1..*} OF Designation;
```

Abstract classes → 5.3

```
      !! The price of a ticket in sovereigns. The currency
      !! is defined in the national basic model of Ahland.
      Price: MANDATORY 0.00 .. 9999.99 [Ahland.Sovereign];
      !! Validity of a ticket. Can be explicit,
      !! e.g. for tickets that are valid for 120 minutes, or
      !! implicit, e.g. for eweekly or season tickets.
      Validity: MANDATORY Ahland.LengthOfTime;
    END TicketType;

    ASSOCIATION =
      TariffZone -- {1} TariffZone;
      TicketType -- {*} TicketType;
    END;

    ASSOCIATION Validity (ABSTRACT) =
      AlpineTransport (EXTERNAL) -- {*} NatTour.AlpineTransports.AlpineTransport;
      TariffZone -- {*} TariffZone;
    END Validity;

    !! A relationship between alpine transport and tariff zone
    !! that has not been derived but entered manually.
    ASSOCIATION ValidityExplicit EXTENDS Validity =
      TariffZone (EXTENDED) -- TariffZoneExplicit;
    END ValidityExplicit;

    ASSOCIATION Quota =
      Participant (EXTERNAL) -- {*} NatTour.AlpineTransports.RailwayCompany;
      TicketType -- {*} TicketType;
    ATTRIBUTE
      Quota: 0.0 .. 100.0 [Units.Percent];
    END Quota;

    CLASS TicketCounter =
      Names: BAG {1..*} OF Designation;
    END TicketCounter;

    CLASS Season =
      Start: FORMAT INTERLIS.XMLDate "1900-1-1" .. "2299-12-31";
      End: FORMAT INTERLIS.XMLDate "1900-1-1" .. "2299-12-31";
    END Season;

    ASSOCIATION Sale =
      TicketCounter -- {*} TicketCounter;
      Season -- {*} Season;
      TicketType -- {*} TicketType;
    ATTRIBUTE
      Number: 1 .. 999999 [Units.CountedObjects];
      Amount: 0.00 .. 9999999.99 [Ahland.Sovereign]
        := Multiply(Number, TicketType -> Price);
    END Sale;

  END Tickets;

END NatTour.
```

Numeric data types
→ 6.1

Association classes
→ 6.13.3

Multiple relationships
→ 6.13.4

4.2.4 IlisTour.ili – The Ilis Valley tourism model

Contracts → 7.2

```
INTERLIS 2.3;

CONTRACTED MODEL IlisTour AT "http://www.interlis.ch/models/beotie"
  VERSION "2008-01" =

!! In order to implement this model, a program package
!! must support the function AhlandToWGS84. This cannot be
!! taken for granted but is subject to a contract with
!! the manufacturer. The necessity of such a contract
!! is stated by CONTRACTED.

  IMPORTS UNQUALIFIED INTERLIS;
  IMPORTS Units, CoordSys, Ahland, Addresses, NatTour;

  !! Tourists with a simple GPS-receiver should benefit
  !! from a special service. Their receivers display coordinates in
  !! the coordinate system WGS84. It uses angles in degrees, minutes
  !! and seconds; the corresponding angle unit is predefined in the
  !! INTERLIS-units model.
  REFSYSTEM BASKET CoordSystems ~ CoordSys.CoordsysTopic
    OBJECTS OF GeoEllipsoidal: WGS84
    OBJECTS OF GeoHeight: WGS84H;

  DOMAIN
    WGS84Coord = COORD -90.00000 ..  90.00000 [Units.Angle_Degree] {WGS84[1]},
                         0.00000 .. 359.99999 CIRCULAR [Units.Angle_Degree]
                                                       {WGS84[2]},
                        -2000.00 ..   9000.00 [m] {WGS84H[1]};
```

Geometry types → 6.9

```
    AhlandLine (ABSTRACT) = POLYLINE VERTEX Ahland.NationalCoord;
    AhlandLineNormal EXTENDS AhlandLine = POLYLINE WITH (STRAIGHTS, ARCS);
    AhlandLineDirected EXTENDS AhlandLineNormal = DIRECTED POLYLINE;
    AhlandSurface = SURFACE WITH (STRAIGHTS, ARCS) VERTEX Ahland.NationalCoord
                    WITHOUT OVERLAPS > 0.02;
    AhlandTessellation EXTENDS AhlandSurface = AREA;
```

Functions → 7.2

```
  !! Conversion of Ahland national coordinates to WGS84.
  FUNCTION AhlandToWGS84 (Ah: Ahland.NationalCoord): WGS84Coord;
  FUNCTION InSurface (Position: Ahland.NationalCoord;
                      Region: AhlandSurface): BOOLEAN;


  TOPIC MITAlpineTransports EXTENDS NatTour.AlpineTransports =

    CLASS MITAlpineTransport EXTENDS NatTour.AlpineTransports.AlpineTransport =
      !! In Ilis Valley there are not only the common national
      !! types of alpine transport but also a snow bus.
      Kind (EXTENDED): (SnowBus);
      !! The National Tourist Office is not interested in
      !! altitudes. However in a winter sport resort such as Ilis Valley
      !! they are of major importance. Hence positions in Ilis Valley
      !! are collected as three-dimensional coordinates (incl. altitudes),
      !! i.e. in comparison with the national model they are extended.
      PosBottomStation (EXTENDED): Ahland.NationalCoord3;
      PosTopStation (EXTENDED): Ahland.NationalCoord3;
      PosBottomStationWGS: WGS84Coord := AhlandToWGS84(PosBottomStation);
```

Extension of enumerations → 6.3.2

```
    PosTopStationWGS: WGS84Coord := AhlandToWGS84(PosTopStation);
    !! Some lines have installed a web-camera that continually displays
    !! the surroundings of the top station, thus tourists may judge whether
    !! a trip is worthwhile. The entry next to the line indicates
    !! via a Uniform Resource Identifier (URI, an address
    !! on the internet), where the most recent picture is
    !! available.
    PictureTopStation: URI;
    TrackCourse: AhlandLineNormal;
    HikersToboggans: (unsuitable, suitable);
  END MITAlpineTransport;

  VIEW CheckTrackStartAndEndPoint
    INSPECTION OF Tracks ~ MITAlpineTransport -> TrackCourse;
  =
  MANDATORY CONSTRAINT
    !! The first point of the tracks must be the bottom,
    !! the last point the top station.
    Tracks -> Segments[FIRST] -> SegmentEndPoint == PARENT -> PosBottomStation
      AND
    Tracks -> Segments[LAST] -> SegmentEndPoint == PARENT -> PosTopStation;
  END CheckTrackStartAndEndPoint;

  !! A tariff zone where the set of all railways participate in a
  !! clearly defined region.
  CLASS TariffZoneInRegion EXTENDS NatTour.Tickets.TariffZone =
    Region: AhlandSurface;
  END TariffZoneInRegion;

  !! A view that comprises the set of all railways whose bottom and top station
  !! are situated within the region of a tariff zone. Obviously
  !! only those tariff zones can be included that have been described as
  !! region (TariffZoneInRegion); an explicit tariff zone would not
  !! make sense here.
  VIEW AlpineTransportsInRegion
  JOIN OF At ~ NatTour.AlpineTransports.AlpineTransport,
          Z ~ TariffZoneInRegion;
  WHERE InSurface(At -> PosBottomStation, Z -> Region) AND
        InSurface(At -> PosTopStation, Z -> Region);
  =
  END AlpineTransportsInRegion;

  !! A relationship between ticket type and tariff zone,
  !! that was not entered manually but derived
  !! automatically based upon the position of
  !! bottom and top station.
  ASSOCIATION ValidityInRegion EXTENDS NatTour.Tickets.Validity
  DERIVED FROM AiR ~ AlpineTransportsInRegion
  =
    AlpineTransport (EXTENDED) -- AlpineTransport := AiR -> At;
    TariffZone (EXTENDED) -- TariffZoneInRegion := AiR -> Z;
  END ValidityInRegion;

END MITAlpineTransports;


TOPIC Hotels =
  DEPENDS ON Addresses.Buildings;
```

Internet addresses → 6.4

Inheritance due to organizational reasons → 5.6

Consistency constraints → 6.14

Views → 6.17

Derivable relationships → 6.13.7

```
    CLASS Hotel =
      !! The names of this hotel, if necessary in different
      !! languages. A minimum of one (1) name must be known, however there
      !! is no upper limit (*) for the number of names.
      Names: BAG {1..*} OF NatTour.Designation;
      !! The internet-address (Uniform Resource Identifier,
      !! URI for short) of a picture of a hotel.
      Picture: URI;
    END Hotel;

    !! Authorities in Ilis Valley do not define themselves what an address is.
    !! Instead they establish a relationship between a hotel and its
    !! house entrance. Hence they can accept the coordinates of the hotels
    !! from the data of cadastral surveying and do not be concerned with their
    !! collection.
    ASSOCIATION =
      Hotel -- Hotel;
      Entrance (EXTERNAL) -- Addresses.Buildings.HouseEntrance;
    END;

  END Hotels;


  TOPIC MITPlanning =
    DEPENDS ON IlisTour.MITAlpineTransports;

    CLASS OperatingHours =
      StartDate: INTERLIS.XMLDate;
      Beginning: Ahland.AhlandXMLTimeOfDay;
      End: Ahland.AhlandXMLTimeOfDay;
    END OperatingHours;

    ASSOCIATION =
      Line (EXTERNAL) -<#> {1} IlisTour.MITAlpineTransports.MITAlpineTransport;
      OperatingHours -- {*} OperatingHours;
    END;

  END MITPlanning;


  TOPIC MITOperation =
    DEPENDS ON IlisTour.MITAlpineTransports;

    CLASS OperatingDecision =
      MomentInTime: INTERLIS.XMLDateTime;
      Decision: (yes, no);
    END OperatingDecision;

    ASSOCIATION =
      Line (EXTERNAL) -<#> {1} IlisTour.MITAlpineTransports.MITAlpineTransport;
      OperatingDecision -- {*} OperatingDecision;
    END;

  END MITOperation;


  TOPIC MITCurrentEvents =
```

Internet addresses → 6.4

Composition → 6.13.2

```
    DEPENDS ON IlisTour.MITAlpineTransports;

    STRUCTURE IndicationOfWind =
      WindDirection: MANDATORY (N, NE, E, SE, S, SW, W, NW) CIRCULAR;
      WindSpeed: MANDATORY 0 .. 200 [Units.kmh];
    END IndicationOfWind;

    CLASS InformationOnConditions =
      !! Temperatures are indicated in degrees Celsius. This
      !! unit is defined by the INTERLIS-units model (Units).
      !! MANDATORY means that the temperature
      !! must be known.
      Temperature: MANDATORY -50 .. 50 [Units.oC];
      !! The attribute refers to the above-mentioned structure
      !! IndicationOfWind.
      Wind: IndicationOfWind;
      WaitingTime: Ahland.LengthOfTimeInMinutes;
      Captured: MANDATORY INTERLIS.XMLDateTime;
    END InformationOnConditions;

    ASSOCIATION =
      Transport (EXTERNAL) -<#> {1}IlisTour.MITAlpineTransports.MITAlpineTransport;
      InformationOnConditions -- {*} InformationOnConditions;
    END;

  END MITCurrentEvents;


  TOPIC SkiRuns =

    CLASS SkiRun =
      Difficulty: (blue, red, black: FINAL) ORDERED;
      Course: AhlandLineDirected;
    END SkiRun;

  END SkiRuns;


  TOPIC ConditionsOfSkiRuns =

    CLASS ConditionOfSkiRuns =
      PreparedSurface: AhlandTessellation;
    END ConditionOfSkiRuns;

  END ConditionsOfSkiRuns;

END IlisTour.
```

Structures → 6.10

Cylic enumeration → 6.3.1

Optional and mandatory attributes → 6.5

Enumerations → 6.3.1

Lines → 6.9

Tessellation → 6.9.4

4.3 Transfer data

If Ilis Valley wants to send their entire data to the National Tourist Office, they generate a transfer file (with their software package). Usually in this form it will be read by another computer system and not by human beings. Nevertheless below a small part of this transfer file appears in print in order to provide you with an idea of its structure.

Three dots (...) mark omissions; the boxes on the right are merely notes that do not belong to the transfer file.

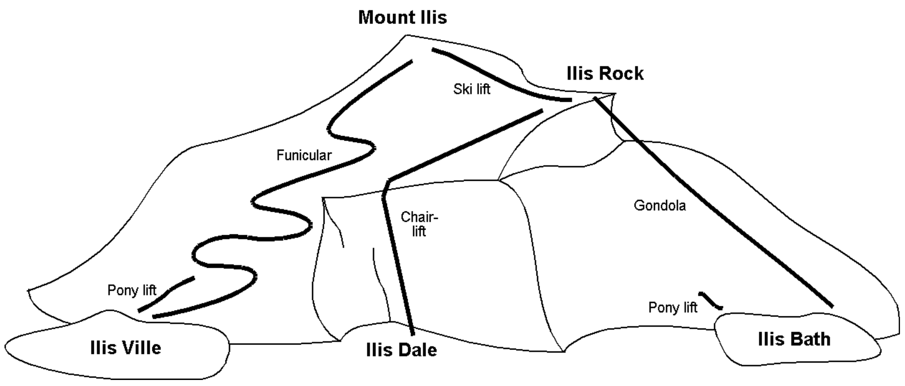


Figure 22: All means of alpine transport up to Mount Ilis are part of the data contained in a transfer file (repetition of figure 11). The following file contains some data concerning the pony lift Ilis Ville.

Header

```
<?xml version="1.0" encoding="utf-8"?>
<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.3">

<HEADERSECTION VERSION="2.3" SENDER="AHTOUMIT0">
  <ALIAS>...</ALIAS>
</HEADERSECTION>

<DATASECTION>
<BASKET BID="xAHTOUMIT01234567" TOPICS="IlisTour.MITAlpineTransports">
  <IlisTour.MITAlpineTransports.MITAlpineTransport
   TID="xAHTOUMIT04231336">
```

Names

```
    <Names>
      <NatTour.Designation>
        <Name>Pony lift Ilis Ville</Name>
        <Language>en</Language>
      </NatTour.Designation>
    </Names>
    <PosBottomStation>
```

Position bottom station

```
      <P>
        <C1>7931.11</C1>
        <C2>13171.23</C2>
        <C3>1771.34</C3>
      </P>
    </PosBottomStation>
    <PosTopStation>
```

Position top station

```
      <P>
        <C1>8020.60</C1>
        <C2>13188.62</C2>
        <C3>1789.04</C3>
      </P>
    </PosTopStation>
    <TravelTime>
```

Travel time

```
      <Ahland.LengthOfTimeInMinutes>
        <Duration>3</Duration>
      </Ahland.LengthOfTimeInMinutes>
```

```
    </TravelTime>
    <Type>SkiLift</Type>
    <PosBottomStationWGS>
      <P>
        <C1>23.68611</C1>
        <C2>44.20278</C2>
        <C3>1771.34</C3>
      </P>
    </PosBottomStationWGS>
    <PosTopStationWGS>
      <P>...</P>
    </PosTopStationWGS>
    <PictureTopStation>
      http://www.ilishornbahnen.com/webcam?bahn=pony4
    </PictureTopStation>
    <CourseOfTracks>...</CourseOfTracks>
    <HikersToboggans>unsuitable</HikersToboggans>
    <OperatingHours>...</OperatingHours>
    <OperatingDecision>...</OperatingDecision>
    <InformationOnConditions>
      <Ilistour.MITCurrentEvents.InformationOnConditions>
        <Temperature>13</Temperature>
        <Wind>
          <Ilistour.MITCurrentEvents.IndicationOfWind>
            <WindDirection>NE</WindDirection>
            <WindSpeed>13</WindSpeed>
          </Ilistour.MITCurrentEvents.IndicationOfWind>
        </Wind>
        <WaitingTime>
          <Ahland.LengthOfTimeInMinutes>
            <Duration>8</Duration>
          </Ahland.LengthOfTimeInMinutes>
        </WaitingTime>
        <Captured>2002-11-25T15:11:00</Captured>
      </Ilistour.MITCurrentEvents.InformationOnConditions>
    </InformationOnConditions>
  </IlisTour.MITAlpineTransports.MITAlpineTransport>
</BASKET>
</DATASECTION>
</TRANSFER>
```

Type

Position bottom station (WGS84)

Picture top station

Information on conditions

5. Inheriting becomes the fashion

5.1 Heirlooms and succession – Principles of inheritance

Actually a set of alpine transport is nothing extraordinary, because it has many characteristics in common with all railways. For instance like all other railways it has a name. Even the fact that a means of alpine transport is run by a company is not that special – after all it is the same for a tramway.

On the other hand it is obvious that alpine transports and tramways have some things in common but nevertheless are not quite the same.

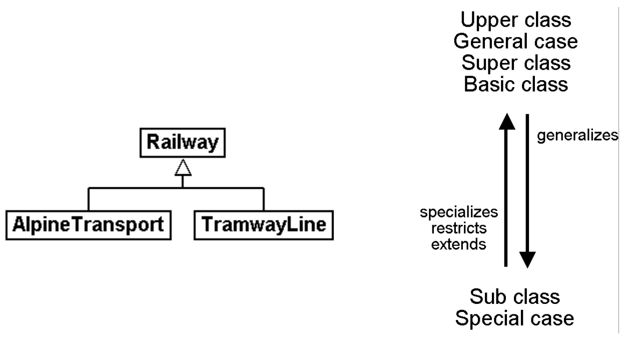


Figure 23: Similar in many aspects but not quite the same: AlpineTransport and TramwayLine both are special railways – they are sub classes of the more general super class railway.

By means of **inheritance** similarities and differences of object classes can be formulated. **Sub classes specialize** the more general **super classes**.

In diagrams it is customary to place the more general super class above the more special sub class. However more complicated diagrams tend to turn out badly arranged if one were to abide strictly by this principle. In any case it is the direction of the arrow and not the order on paper that is decisive.

Each alpine transport is a railway, but not every railway rides up a mountain: The set of alpine transport is a sub set of the set of all railways. We also speak of the fact that the sub class AlpineTransport is a **restriction** of the super class Railway.

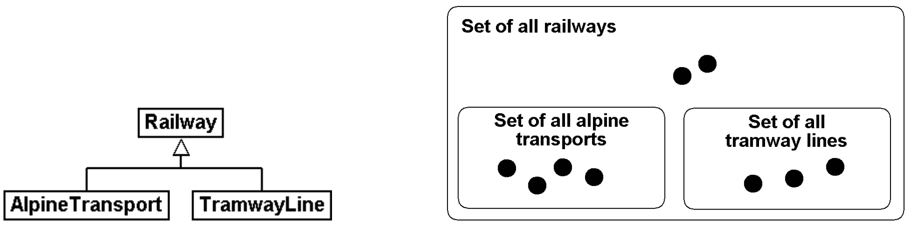


Figure 24: The specialization of classes corresponds to a sub-set relationship of object sets: The set of alpine transport (in the picture on the right with four elements) must be contained entirely in the set of all railways (nine elements), because in the model (picture on the left) the class AlpineTransport specializes the more general class Railway.

Occasionally we use the term **extension** – with the same meaning as «restriction» – for specialization.

It is confusing that with modeling the terms «restriction» and «extension» are often used with the same meaning. Here is the reason why: A class may also be understood as a number of conditions upon which it can be decided whether an object belongs to a class (e.g. criteria what exactly is a railway). A sub class heightens all these requirements: In order for something to qualify as an alpine transport it not only has to meet all the requirements of a railway but it must moreover comply with further demands. Thus by extending requirements, a sub class restricts at the same time the set of the specimens belonging to it.

Inheritance is a fabulous means of creating order in a complex world. On the other hand we may be tempted to formulate a more detailed model – i.e. to distinguish between classes without the least necessity.

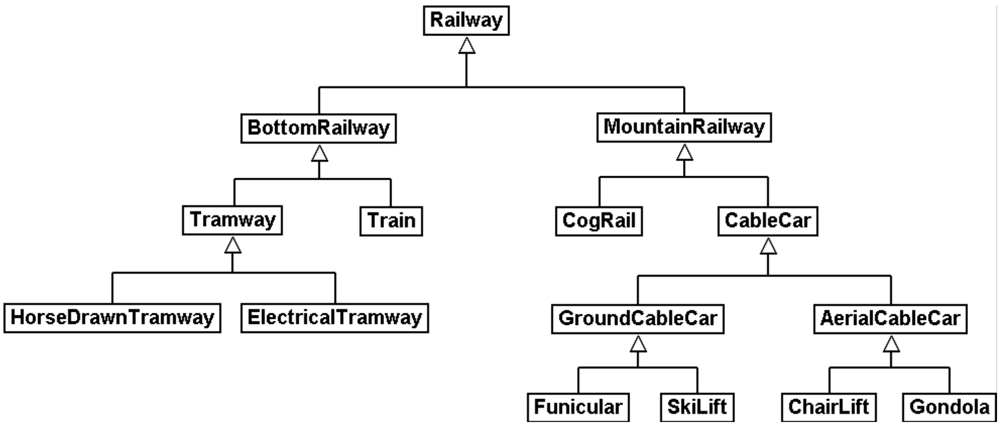
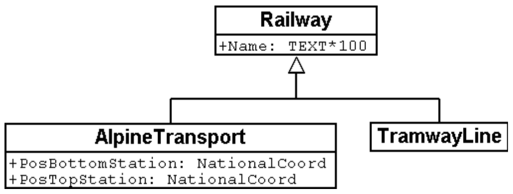


Figure 25: The powerful tool of inheritance may lead us into distinguishing between special cases even if an application would not render this necessary. It is true that a horse-drawn tramway is not the same as an electrical one but: Does it make any sense to state these differences in a data model or does it only unnecessarily blow up the model?

The corresponding characteristics help decide whether a subdivision in to special classes is worthwhile. Thus each railway has its name, but only alpine transports have a bottom and a top station.


Railway
+Name: TEXT*100
AlpineTransport
+PosBottomStation: NationalCoord
+PosTopStation: NationalCoord
TramwayLine

```
CLASS Railway =
  Name: TEXT*100;
END Railway;

CLASS AlpineTransport EXTENDS Railway =
  PosBottomStation: NationalCoord;
  PosTopStation: NationalCoord;
END AlpineTransport;

CLASS TramwayLine EXTENDS Railway =
END TramwayLine;
```

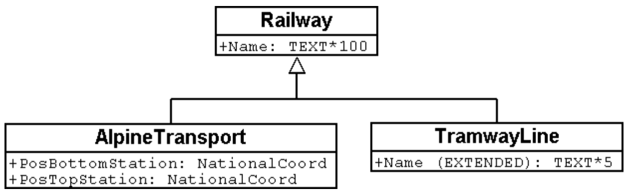
Figure 26: AlpineTransport and TramwayLine accept («inherit») the property name of their super class Railway, without this having to be stated once more. In addition to inherited properties an alpine transport possesses further characteristics, i.e. the position of bottom and top station (in national coordinates). On the right hand side the same is stated in the notation of INTERLIS.

- Sub classes accept or **inherit** always all the properties of their more general super classes. However they can define additional characteristics.

5.2 Refine what has been inherited

In general one hundred signs may be appropriate for the name of one particular alpine transport. After all at a time it had been considered to run a «Children and Family tow-rope Mount Ilis area». However at the very last minute it was decided to use «pony lift» instead, to the great relief of the local tourist organization.

But a tramway line with one hundred signs? Five signs should definitely be enough.



```
CLASS Railway =
  Name: TEXT*100;
END Railway;

CLASS AlpineTransport EXTENDS Railway =
  PosBottomStation: NationalCoord;
  PosTopStation: NationalCoord;
END AlpineTransport;

CLASS TramwayLine EXTENDS Railway =
  Name (EXTENDED): TEXT*5;
END TramwayLine;
```

Figure 27: Five signs are sufficient for the name of a Tramway Line: The type of the property «Name» is refined by the sub class (specified). On the right hand side the same is stated in the notation of INTERLIS.

The specified, refined information must be compatible with the one inherited. For instance the maximum length for a tramway line may not exceed the length of the common railway.

- Sub classes can **refine** inherited properties. However the specified information may not contradict the one inherited: they must be **compatible** with the definition of the super class.

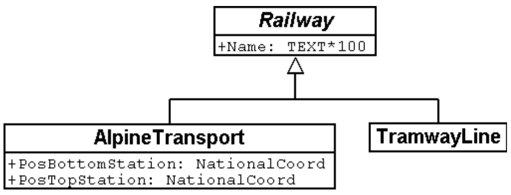
Otherwise a sub class might feature objects that are not part of the set of all objects of the super class.

5.3 Is there really such a thing? – Abstract classes

Some classes are purely mental tools: They do not represent actually existing items. For instance there is not one single living being in this world that would only be living being without being something more specific at the same time. Similarly a data model might determine that there is no railway as such but that every railway would have to be either a tramway line, an alpine transport etc.

▶ If a class should not feature any concrete objects, it is declared **abstract**.

Very often in a data model all its super classes will be abstract and only its very last, most specific classes will be concrete.


Railway
+Name: TEXT*100
AlpineTransport
+PosBottomStation: NationalCoord
+PosTopStation: NationalCoord
TramwayLine

```
CLASS Railway (ABSTRACT) =
  Name: TEXT*100;
END Railway;

CLASS AlpineTransport EXTENDS Railway =
  PosBottomStation: NationalCoord;
  PosTopStation: NationalCoord;
END AlpineTransport;

CLASS TramwayLine EXTENDS Railway =
END TramwayLine;
```

Figure 28: Railway as an abstract class: If it is required that there be no objects that are only railway without also being alpine transport or tramway line this is shown in the diagram in italics. On the right hand side the same is stated in the notation of INTERLIS.

5.4 We do not want to give such precise orders – Abstract properties

Let's assume an international association wishes to ensure that tickets are captured with their prices. Then again it does not want to dictate a certain currency and consequently it is not clear where a sensible upper limit for the price could be set. Nevertheless it is not contested that «price» should be a number and that we deal with money. After all prices are not measured in kilometers per hour!

```
CLASS TicketTypeWorldwide (ABSTRACT) =
  Price (ABSTRACT): NUMERIC [MONEY];
END TicketTypeWorldwide;

CLASS TicketTypeAhland EXTENDS TicketTypeWorldwide =
  Price (EXTENDED): 0.00 .. 9999.99 [Ahland.Sovereign];
END TicketTypeAhland;
```

▶ Not all properties have to be defined down to the last detail: with abstract classes **abstract properties** are admissible. It is then up to the concrete sub classes, to specify these properties. For instance this is handy when regulating something on national or international level without prescribing every single detail right from the beginning.

5.5 Details are of no interest – A closer look at the specific

In general whoever demands information on the transport system of a country does not want to know whether one particular means of transport is a cable car, a tramway or some other sub type of railway. Nor would he want to find out what system of cogs is used by a line, if it were to be a cog rail. Nothing but its name (that according to the data model is captured for each means of transport) is sufficient as an answer.

- ▶ Entities of a sub class can always be considered to be generalizing in terms of a super class.

  The Greek expression for this principle is polymorphism.

However this applies on one condition:

- ▶ Each extension must be **compatible** with its basic definition. Compatible means that each value possible with the extended definition can be mapped onto the basic definition in accordance with the rules of the basic type (text, enumeration, number coordinate etc.).

5.6 Inheritance on a larger scale

Not always the distinction between sub- and super class is justified on a mere factual basis. Organizational considerations may be decisive.

For instance in Ilis Valley they basically agree with the idea of an alpine transport as conceived by the National Tourist Office. Nevertheless they are not quite satisfied:

- ♦ For those lines that run up to Mount Ilis it would be interesting to know the course of the tracks. If it were to be captured, then the course could be added to the maps that are available free for tourists at the information centre.
- ♦ Furthermore Ilis Valley would like to record, which lines are suitable for hikers and toboggans.

Both are properties that basically apply to every alpine transport could feature – they simply do not feature in the national model. Of course Ilis Valley has asked the National Tourist Office to adjust their model accordingly. But the only reply they got as that they neither had the time nor the money to alter all their computer systems in the country just to gratify the wishes of some mountain valley. So what now?

Some reckoned that the National Tourist Office had best be ignored since it consisted mainly of bureaucrats without the least understanding for practical matters! (Other words were mentioned that had nothing whatsoever to do with the matter.)

Others could relate to the opinion of the National Tourist Office – just imagine if every valley were to have their own way. And besides they still profited from the National Tourist Office: with the data they are sent material for and about the Ilis Valley is produced.

So should the people in charge renounce their extra wishes? Or double gather all their data – once for themselves, once for the National Tourist Office?

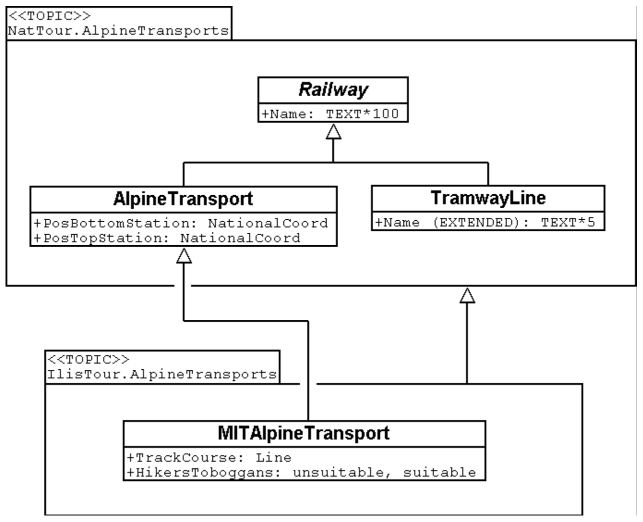


Figure 29: The National Tourist Office is not willing to adjust their model to the extra wishes of Ilis Valley. Thanks to inheritance Ilis Valley can still collect their data: Their topic AlpineTransports inherits everything from the national topic AlpineTransports, but adds as an extension the object class MITAlpineTransport with additional properties.

Thanks to inheritance this conflict could be solved. In Ilis Valley all railways are captured as MITAlpineTransport with all extensions. Since MITAlpineTransport is a sub class of AlpineTransport (in accordance with the National Tourist Office), each MITAlpineTransport can be read as an ordinary AlpineTransport. Hence Ilis Valley can send their data just as they are to the National Tourist Office.

▶ Inheritance can also be used to support federal characteristics.

To be exact it is due to polymorphism that is rendered possible by inheritance: Each entity of a sub class can be regarded as part of the super class (cf. paragraph 5.5). Thus the National Tourist Office can process data from each alpine transport in the country even if it is an example of a local sub class of «alpine transport» unknown to the National Tourist Office.

Inheritance does not go very far with INTERLIS: Not only classes and topics, but also domains (types), views, graphic definitions, in a certain sense even units can be inherited and specified.

5.7 Simple and multiple inheritance

Some modeling languages permit the simultaneous inheritance of several basic elements. Thus a class may refine several super classes at the same time.

In information technology it is debated as to how useful this really is. Models using multiple inheritance often become more confusing. Hence INTERLIS only applies simple inheritance.

6. The data model under closer scrutiny

6.1 Sovereigns and pennies – Numeric data types

6.1.1 Value domain

A simple ticket for a single ride with the funicular up to Mount Ilis costs 10 Ahland sovereigns, a yearly sport pass is worth 635 sovereigns. So how should the price of a ticket be modeled?

Whoever is used to dealing with programming language would think immediately of whole number («Integer») or floating point number («Real»). With many programming languages and data base it is possible to determine the memory space a number will take up, hence ensue size and precision of the numbers that can be stored («short integer», «long integer», «double precision», etc.). Thus one would consider the domain within which the value would approximately be situated and choose storage accordingly.

When modeling at conceptual level most likely one would not want to be bothered with storage. However it is possible to indicate the highest and lowest value admissible and to determine at the same time the number of digits and exponents.

Since prices lie between 10 and 635 sovereigns we could then write:

```
Price: 10 .. 635;
```

However this would also decide that the price can never be less than 10 sovereigns or more than 635 sovereigns, which of course is very undesirable. As a lower limit 0 seems plausible. But the upper limit? It is unlikely to be more than 10'000 sovereigns. So 10'000? Wherever because of a particular application a precise limit cannot be determined, in any case it would not make sense to select the value just so that shorter storage (e.g. *short integer* with value between –32768 and 32767) would not be sufficient.

Of course it is also important to consider the number of decimals necessary. If prices may feature pennies then the limit would have to be set with to decimals. With an attribute which for instance represents an investment budget you are likely to deal in millions, when dealing with dimensions two digits will be sufficient. However these may refer to a thousand, a million or even a billion.

```
Price: 0.00 .. 9999.99;
Budget: 0.00E6 .. 999.99E6;
GrossNationalProduct: 0.0E0 .. 9.9E20;
```

The number of decimals describes the precision: «4.3 Million» ($4.3E6 = 4.3 \cdot 10^6$) is not as precise an indication as «4300000».

6.1.2 Units

Now do we deal with sovereigns, francs, Euro or dollars? Units are – not only where money is concerned – of paramount importance. That is why we recommend indicating the unit not only as a comment or part of the attribute name but actually as constituent of the type:

```
UNIT
  Sovereign EXTENDS MONEY;

CLASS TicketType =
  Price: 0 .. 9999 [Sovereign];
END TicketType;
```

Sovereigns (or Swiss francs, Euro, dollars, etc.) are defined as units and can then be used when defining a numeric type. However most common units (amongst others also CHF, EUR, USD) actually need not be defined but are available via the specific units model (units.ili, cf. paragraph 3.3).

⚠ In the interest of clarity we recommend indicating units at any time. The INTERLIS-units-model comprises, besides numerous physical units, such units for common currencies and for number, percent and per mille.

6.1.3 Inheriting numeric types

Since the Mount Ilis Alpine Transports will never issue tickets that cost over a thousand sovereigns, one might come up with the idea of defining this in its own class:

```
CLASS TicketType (EXTENDED) =
  Price (EXTENDED): 0 .. 1000 [sovereigns];
END TicketType;
```

In this application this concrete example does not make much sense really. But in real life there certainly are cases where limiting the inherited value domain is a sensible thing to do.

Nevertheless those in charge in Ilis Valley cannot suit themselves where the value domains of the national models are concerned. The principle applies that each Illis Valley value also has to be admissible in the basic model of the National Tourist Office. Otherwise we could not ensure a smooth transfer of data from Ilis Valley to the National Tourist Office.

Thus an indication of price 0 .. 1000 is an admissible restriction of the value domain provided nationally of 0 .. 9999. If the value domain in Ilis Valley were to be extended – for instance with an indication of 0 .. 15000 – then all ticket types ranging between 10000 and 15000 were incorrect from the National Tourist Office's point of view. Hence such a definition is inadmissible.

The people in charge in Ilis Valley also insist on the possibility of issuing tickets with prices that not necessarily have to be an entire sovereign. One would like to write the following definition:

```
CLASS TicketType (EXTENDED) =
  Price (EXTENDED): 0.00 .. 1000.00 [Sovereign];
END TicketType;
```

Then again, would this be compatible with the basic model? Yes, because according to the Illis Valley model (e.g. 7.35) with a mathematical rounding each value can be mapped onto a correct value of the national model (e.g. 7).

The definition would be inadmissible if it were to comprise values that once rounded violate the basic model. For instance the maximum value of 9999.99 once rounded would become 10000 – which would be more than the 9999 predefined by the National Tourist Office. However Ilis Valley could define a value ranging from 0.00 .. 9999.49 without contradicting the basic model, because after rounding 9999.49 would become again 9999.

It is also inadmissible to renounce precision in the specified model. For instance if the national tourism model were to provide a domain of 0.00 .. 1000.00 Illis Valley could not indicate 0 .. 1000 in their specification.

And furthermore: Units of the extension must always be in accordance with those of the basic model!

6.1.4 Limits as yet unknown

Can the National Tourist Office determine what price is admissible for a ticket? If the limits are still entirely unknown, then – within the scope of abstract attributes – their notation can be omitted. Nevertheless it is possible to already determine a unit.

```
Price (ABSTRACT): NUMERIC [Ahland.Sovereign];
```

Maybe even the unit is still uncertain. It would only be known that e deal with e.g. a currency.

```
Price (ABSTRACT): NUMERIC [MONEY];
```

Thus at least the character of the unit would be settled. Therefore in an extension only such units could be defined as would figure as concretes of the abstract unit MONEY (for further information concerning abstract properties also see paragraph 5.4).

```
  Price (EXTENDED): 0 .. 10000 [CHF]; !! admissible
  Price (EXTENDED): 0 ..  2000 [USD]; !! admissible
  Price (EXTENDED): 0 ..  1000 [m];   !! inadmissible, because meter
                                      !! specifies LENGTH and not MONEY.
```

6.2 Types of alpine transport – Modeling of types of objects

For a brief overview it is sufficient to roughly divide alpine transports: Cog rail, funiculars, aerial cable cars, ski lift, chair lifts, gondolas. In the simplest of cases the type would be noted as a text attribute.

```
CLASS AlpineTransport =
  Name: TEXT*100;
  Kind: TEXT*50;
END AlpineTransport;
```

Consequently people in charge of data input would be very free in their description. Aerial cable car, gondola, ski lift, ski-lift – it is to be feared that a virtual jungle of terms would result. This could be avoided by using an enumeration.

```
CLASS AlpineTransport =
  Name: TEXT*100;
  Kind: (CogRail,
         Funicular,
         AerialCableCar
         SkiLift,
         ChairLift,
         Gondola);
END AlpineTransport;
```

Since all admissible possibilities have been enumerated, order would have been established. Often it will appear desirable to add further attributes such as the number of available seats on some means of transport. With funicular and aerial cable car this would be the capacity of the entire cabin, with ski and chair lifts it would be the number of persons per single ride. However with a cog rail where several wagons can be hooked up this information would not make sense. Maybe the cog system would be of greater interest. Now should the class AlpineTransport simply feature all attributes needed for the describing of the different kinds?

If the different kinds feature their respective properties (attributes or relationships) it makes sense to define individual classes that inherit from the basic class (cf. chapter 5).

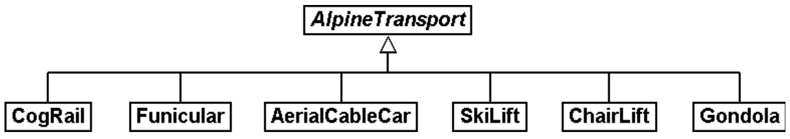


Figure 30: CogRail, Funicular etc. are specific types of AlpineTransport. However there is no alpine transport as such: All «concrete» types of alpine transport always belong to one of the sub classes. Hence *AlpineTransport* is an abstract class, which in the diagram would be shown by means of italics.

But there are no means of alpine transport that are exclusively alpine transport without at the same time being part of a sub class. Therefore the class AlpineTransport will be declared «abstract». Consequently a concrete means of alpine transport will always have to be a cog rail, an aerial cable car etc.

In the textual notation of INTERLIS 2 abstract classes are pointed out with the indication (ABSTRACT) in brackets. As a side remark: The INTERLIS-units model «Units» features a unit «CountedObjects» for counted objects such as the number of people in a aerial cable car cabin.

```
CLASS AlpineTransport (ABSTRACT) =
  Name: Text * 100;
END AlpineTransport;
CLASS CogRail EXTENDS AlpineTransport =
  CogSystem: (Riggenbach, Abt, vonRoll);
END CogRail;
CLASS Funicular EXTENDS AlpineTransport =
  Capacity: 0 .. 999 [Units.CountedObjects];
END Funicular;
CLASS AerialCableCar EXTENDS AlpineTransport =
  Capacity: 0 .. 999 [Units.CountedObjects];
END AerialCableCar;
CLASS SkiLift EXTENDS AlpineTransport =
  PersonsPerRide: 0 .. 10 [Units.CountedObjects];
END SkiLift;
CLASS ChairLift EXTENDS AlpineTransport =
  PersonsPerRide: 0 .. 24 [Units.CountedObjects];
END ChairLift;
CLASS Gondola EXTENDS AlpineTransport =
  Capacity: 0 .. 99 [Units.CountedObjects];
END Gondola;
```

For the meeting a railway person had been invited especially who then did a lengthy speech on cog rails. All present learnt a lot about what cog systems are in use world-wide and about their respective advantages and disadvantages. However at the end of the day the people in Ilis Valley asked themselves what, after all, did these cog systems have to do with their project. Nobody could imagine of what possible interest these or other pieces of information would be in a future extension. Thus this model was rejected because it went too much into detail and finally would incur unnecessary costs for the input and maintenance of data.

See also paragraph 5.1 which deals with the temptation to enter into too many details when modeling.

6.3 Is there such a thing as light blue ski runs? – Structured enumeration

6.3.1 Ordinary enumeration and its laws of inheritance

In order to achieve a rough distinction between the difficulty degrees of ski runs, three colors had been chosen: blue, red, black. These and only these difficulty degrees should prevail. Furthermore they all relate to an order. Blue depicts a simple ski run, a red run is more difficult than a blue one, and a black one is the most demanding. The following definition would describe this circumstance:

```
CLASS SkiRun =
  DifficultyDegree: (blue, red, black: FINAL) ORDERED;
END SkiRun;
```

If the FINAL-indication were to be missing, then this enumeration could still be added onto in an extension. For instance in the case of different kinds of alpine transport this might make sense:

```
!! Model of the National Tourist Office
CLASS AlpineTransport =
  Kind: (CogRail, Funicular, AerialCableCar,
        SkiLift, ChairLift, Gondola);
END AlpineTransport;

!! Model Ilis Valley
CLASS MITAlpineTransport EXTENDS AlpineTransport =
  Kind (EXTENDED): (SnowBus);
END MITAlpineTransport;
```

In the extended class the element snow bus is added to the enumeration – the latest of all novelties – at the end of the existing enumeration. But what will the National Tourist Office make of that? For them «Snow bus» is still unheard of.

▶ Each (horizontal) extension can be complemented with further value as long as it is not expressively excluded by **FINAL**. If somebody is only interested in these values in general according to the basic class, then all these values will be translated into the value **OTHER**.

For the basic class the value snow bus - and other possible values - are only recognizable as OTHER. However if FINAL is indicated, then the value OTHER no longer can occur. If an enumeration is defined as circular (**CIRCULAR**) then such extensions are impossible because circular means that the highest value is followed by the lowest and hence it would be impossible to know which of them is the highest.

```
WindDirection: (N, NE, E, SE, S, SW, W, NW) CIRCULAR;
```

6.3.2 Sub-enumeration

Thu s it had been decided not to model the different kinds of alpine transport with an entire landscape of classes. But all the railway enthusiasts were not quite satisfied: who knows if at some stage the cog system of the cog rail would not be of interest after all...

For each value of an enumeration a sub-enumeration can be defined. This may happen directly within the basic definition or only in an extension.

```
CLASS MITAlpineTransport EXTENDS AlpineTransport
  Kind (EXTENDED): (CogRail (Riggenbach, Abt, vonRoll));
END MITAlpineTransport;


WeekDay: (WorkingDay (Monday, Tuesday, Wednesday,
                      Thursday, Friday, Saturday),
          Sunday);
```

If such an enumeration is defined within an extension, then it is simply of no importance from the base’s point of view. So as far as the National Tourist Office is concerned a Riggenbach-cog rail would still be a cog rail.

Even sub-enumeration can be complemented with further values, as long as value has not been declared FINAL. The individual values of a sub-enumeration again can be specified by sub enumeration, which would result in entire enumeration trees.

6.4 Ilis Valley is concise – Strings and their rules of inheritance

As a general rule designations may consist of names of any chosen length. Nevertheless the national association has determined that the name of any alpine transport may not exceed a total of 100 signs. In general of course the names tend to be much shorter, but one would like to be on the safe side.

```
STRUCTURE RailwayDesignation EXTENDS Designation =
  Name (EXTENDED): TEXT*100;
END RailwayDesignation;
```

Whenever the length of a text attribute is discretionary or as yet completely unknown, the indication of its length can be omitted. If obviously this length will be determined within the scope of an extension of class, the attribute will be qualified as abstract:

```
Description (ABSTRACT): TEXT;
```

Some means of transport in Ilis Valley have installed a web camera, which continually registers the environment of the top station. Tourists may judge themselves whether the trip is worthwhile. The Internet address of the current picture also represents a (somewhat particular) form of text.

```
CLASS MITAlpineTransport =
  ...
  PictureTopStation: URI;
  ...
END MITAlpineTransport;
```

Internet addresses have got nothing in common with the Swiss canton of Uri – if anything then with Geneva where the first Web-Browser was developed at CERN. URI simply is the abbreviation of *Uniform Resource Identifier. Uniform Resource Locators (URLs)*, mainly used for web sites, are special URIs.

6.5 Calm – Optional and mandatory attributes

Current operational data also include weather data such as temperature, direction and force of wind. When it is calm it makes no sense to indicate the direction of the wind. All other information should always be displayed.

▶ The fact that an attribute can be **undefined**, respectively that it always has to be defined, forms part of the model.

⚠ Undefined is not simply 0 or some other slightly exceptional value. It is an independent value, which clearly reflects the fact of being undefined.

For instance in INTERLIS 2 we would write:

```
CLASS Weather =
  Temperature: MANDATORY -50 .. 50 [oC];
  WindDirection: (N, NE, E, SE, S, SW, W, NW) CIRCULAR;
  WindSpeed: MANDATORY 0 .. 200 [kmh];
END Weather
```

Hence both temperature and wind speed are compulsory (MANDATORY). Since the wind direction is not compulsory, it is optional. Thus the concrete value may be undefined. In extensions it is admissible to turn optional attributes into mandatory attributes. However mandatory attributes may never be turned into optional attributes, because in accordance with the basic class the value «undefined» is not permitted.

6.6 Waiting times and duration of ride – Value domains

Both waiting times at stations and duration of rides are noted in minutes.

```
CLASS AlpineTransport =
  Duration: 0 .. 200 [min];
END AlpineTransport;

CLASS StatusAlpineTransport =
  WaitingTime: 0 .. 200 [min];
END StatusAlpineTransport;
```

Both properties can adopt values of the same range. With a specifically defined value range (DOMAIN) this common factor can be highlighted further:

```
DOMAIN
  DurationInMinutes = 0 .. 200 [min];

CLASS AlpineTransport =
  DurationOfRide: DurationInMinutes;
END AlpineTransport;

CLASS StatusAlpineTransport =
  WaitingTime: DurationInMinutes;
END StatusAlpineTransport;
```

6.7 Where is Ilis Valley? – Coordinate types

6.7.1 General remarks concerning coordinate types

We would link the idea of a place in real world, shaped like a point, with the question «Where?». Such a place can be described by means of a coordinate. Typically such a coordinate is a pair of numbers that describe the position of a place, or a triple that describes position and altitude of a place.

For each dimension of a coordinate type we have to determine, in the same way as with numeric types, in which admissible values may range and what unit they refer to.

```
Position: COORD 500.00 .. 91000.00 [m],
                700.00 .. 23000.00 [m];

XPosition: 500.00 .. 91000.00 [m];
YPosition: 700.00 .. 23000.00 [m];
```

At first sight the difference between a position attribute with a coordinate type and one numeric attribute for both X- and Y-direction is minor. Thanks to its definition as a coordinate type it is obvious that these pieces of information belong together. A program package may also exploit this property. For instance many programs are capable of graphic representation of Cartesian coordinate values.

Cartesian coordinate values? Cartesian values are coordinates whose dimensions are perpendicular. Hence the definition of the position coordinates above depicts a rectangular window of approx. 90 times 22 kilometers. Does that mean we return to medieval times? Has the earth in Ilis Valley once more become a disc?

6.7.2 The wrapped up plum – What is a coordinate system?

Already Ptolemaeus considered the earth to be a sphere. Surveyors have had to take leave of this view a long time ago, because it simplifies to too great an extent.

A useful approximation of the earth surface is the Ellipsoid, in other words the surface that results when an ellipse revolves around its central axis.

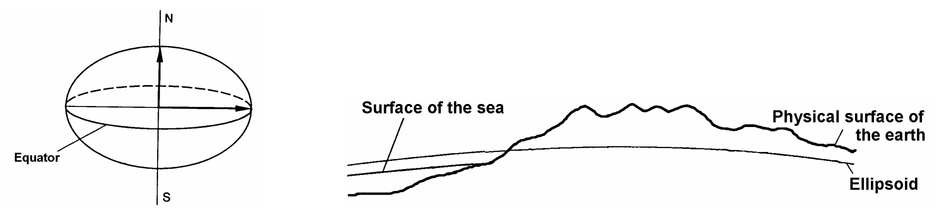


Figure 31: When an ellipse revolves around its own axis, a flattened sphere results in space. Thanks to such an ellipsoid an approximation of the shape of the earth surface can be attempted.
(All figures in this paragraph and in paragraph 6.7.5 from: K. Christoph Graf, Verwendung geodätischer Abbildungen bei der Geocodierung von Satelliten-Bildern. Zürich, 1988. Illustrations have been partially simplified. Original sources as stated above).

Depending on the part of the world, different ellipsoids are used, otherwise the approximation would become too imprecise. For instance Switzerland uses the same ellipsoid as Germany, but one slightly different from Sweden or France.

As a spatial formation ellipsoids are somewhat awkward to handle. For this reason surveyors will map the ellipsoid onto a surface. To this intent they will drape a cylinder or cone over the ellipsoid and light it from the interior, thus projecting the picture of the landscape onto the cylinder or cone.

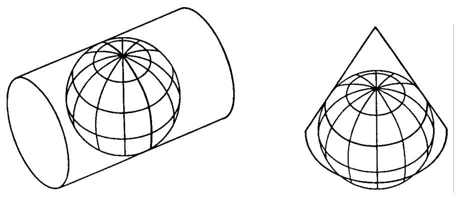


Figure 32: The ellipsoid is wrapped in a cylinder (left) or cone (right). Then it is lit from the interior.

In the next step the cylinder or cone is cut open with a pair of scissors, rolled out flat on a table – and there goes your map!

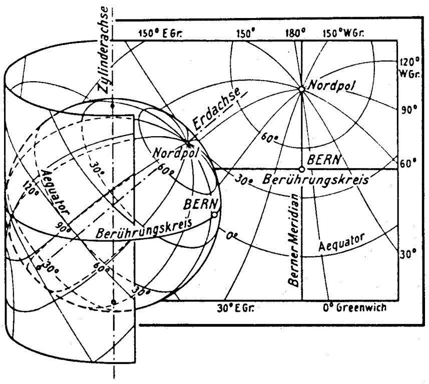


Figure 33: Once the projection has been completed, the cylinder (or cone) is cut open and rolled out. A convex body such as an ellipsoid or a sphere could be cut open but not rolled out flat.

At the end the map is superposed with fine perpendicular lines: the **coordinate system** of the map. That is why with each coordinate type it has to be determined what coordinate system it is based upon.

```
Position: COORD 480000 .. 850000.00 [m] {AhlandSys[1]},
                 60000 .. 320000.00 [m] {AhlandSys[2]};
```

The first dimension of the coordinate corresponds to the first axis of the coordinate system by the name of «AhlandSys», the second dimension to the second axis of the same system.

6.7.3 Information concerning the coordinate system – Meta data

Is «AhlandSys» a Cartesian, an ellipsoid system? What are the names of the axes? Are there any common features (e.g. map projections) with other coordinate systems? All this information can be described by means of data. In order to make it clear ho such data is structured, a corresponding data model is formulated for it. Such a model is called meta model, the appertaining data meta data because they serve to describe the actual data.

Data belonging to a meta model are «meta» in another more formal sense an information concerning origin or price (cf. paragraph 3.3). Unfortunately the same term is commonly used for both of them.

In simple cases here the application range of a data model will make it quite clear what coordinate system the coordinates actually belong to, you may omit the explicit indication of the coordinate system. Nevertheless it does make sense to make at least some sort of mention of the coordinate system in the name of the coordinate type.

```
NationalCoord =  COORD 500.00 .. 91000.00 [m],
                       700.00 .. 23000.00 [m];
```

```
Position: NationalCoord;
```

To avoid confusions, the responsible persons of Ilis Valley have given preference to a precise definition:

```
REFSYSTEM BASKET CoordSystems ~ CoordSys.CoordsysTopic
  OBJECTS OF GeoCartesian2D: AhlandSys;
```

Based upon the general model for coordinate systems (CoordSys), they have presented a precise definition of their national system. For its position they have registered an object of the class GeoCartesian2D with the name of AhlandSys. Within the model the existence of this data entry is referred to by means of OBJECTS OF. Thus the coordinate system AhlandSys becomes available in the model. When applying the system, there is no need to actually mention the name of our stock of meta data (CoordSystems), unless several such stocks of meta data were defined within the current modeling part.

```
LandesKoord =  COORD 500.00 .. 91000.00 [m] {CoordSystems.AhlandSys[1]},
                     700.00 .. 23000.00 [m] {CoordSystems.AhlandSys[2]};
```

6.7.4 Different coordinate systems

In order to offer a special service to those tourists that have a simple GPS-receiver at their disposal, Ilis Valley would like to make their coordinates also available as geographical coordinates in the global WGS84-System.

```
WGS84Coord = COORD -90.00000 ..  90.00000 [Units.Angle_Degree] {WGS84[1]},
                     0.00000 .. 359.99999 CIRCULAR [Units.Angle_Degree]
                                                   {WGS84[2]},

CLASS AlpineTransport =
  PosBottomstation: NationalCoord;
  PosBottomstationWGS: WGS84Coord;
  ....
END AlpineTransport;
```

It seems obvious that both attributes are directly related. Projection coordinates can be converted into WGS84-coordinates. However it is not up to the conceptual description of data to render a detailed definition of such a conversion. Nevertheless it would be desirable to indicate that it is possible to calculate one set of coordinates from the others.

```
!! Conversion of coordinates from the Ahland projection system to WGS84.
!! Functions will be discussed in paragraph 7.2
FUNCTION AhlandToWGS84 (Ah: Ahland.NationalCoord): WGS84Coord;

CLASS AlpineTransport =
  PosBottomStation: Ahland.NationalCoord;
  PosBottomStationWGS: WGS84Coord := AhlandToWGS84 (PosBottomStation);
  ....
END AlpineTransport;
```

6.7.5 Three dimensional coordinates

Of course projection coordinates are not enough for the skiers and hikers around the Ilis. Big differences in altitude delight the skiers, while hikers will either fear perspiration or shaking knees. Altitudes definitely are sought after! That is why coordinate types may also display three dimensions.

```
NationalCoord3 =  COORD 500.00 .. 91000.00 [m] {AhlandSys[1]},
                        700.00 .. 23000.00 [m] {AhlandSys[2]}
                          0.00 ..  9000.00 [m] {AhlandHeightSys[1]};

WGS84Coord = COORD -90.00000 ..  90.00000 [Angle_Degree] {WGS84[1]},
                     0.00000 .. 359.99999 CIRCULAR [Angle_Degree]
                                                   {WGS84[2]},
                  -2000.00 ..   9000.00 [m] {WGS84H[1]};
```

With altitudes an exceptional problem occurs. Where in fact would altitude 0 be situated? How can the altitude of any point be determined in relation to this altitude 0? Surveyors differentiate mainly between the altitudes according to the gravity field of the earth (gravity or geoid height; 0 being the altitude of the imaginary continuation of the sea below the continents) and altitudes according to the geometrical approximation of the earth (ellipsoid height; 0 being the surface of the ellipsoid).

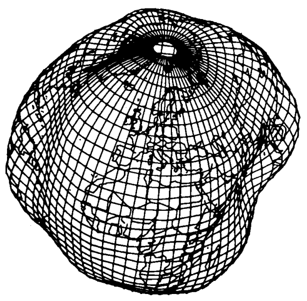


Figure 34: The gravity field of the earth: With the geoid the surface of the oceans is mentally continued under the continents. Mountain ranges, trenches etc influence the gravity field and thus alter the imaginary surface of the water. This drawing is quite exaggerated.

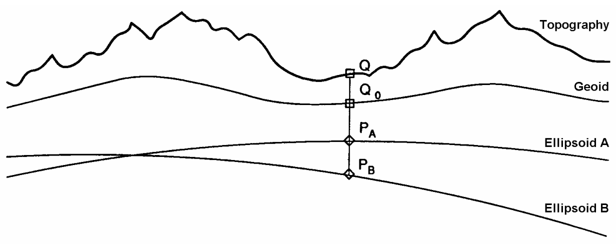


Figure 35: Depending on the reference system selected point Q will have a different altitude.

Typically projection coordinate systems will use geoid heights. That is why the third dimension of projection coordinates does not simply refer to the third axis of the projection system but to the first axis of a special height system.

In contrast coordinates in GPS-measurements are determined purely geometrically from satellite positions without taking into account the gravity field of the earth. Hence WGS84-altitudes are ellipsoid heights.

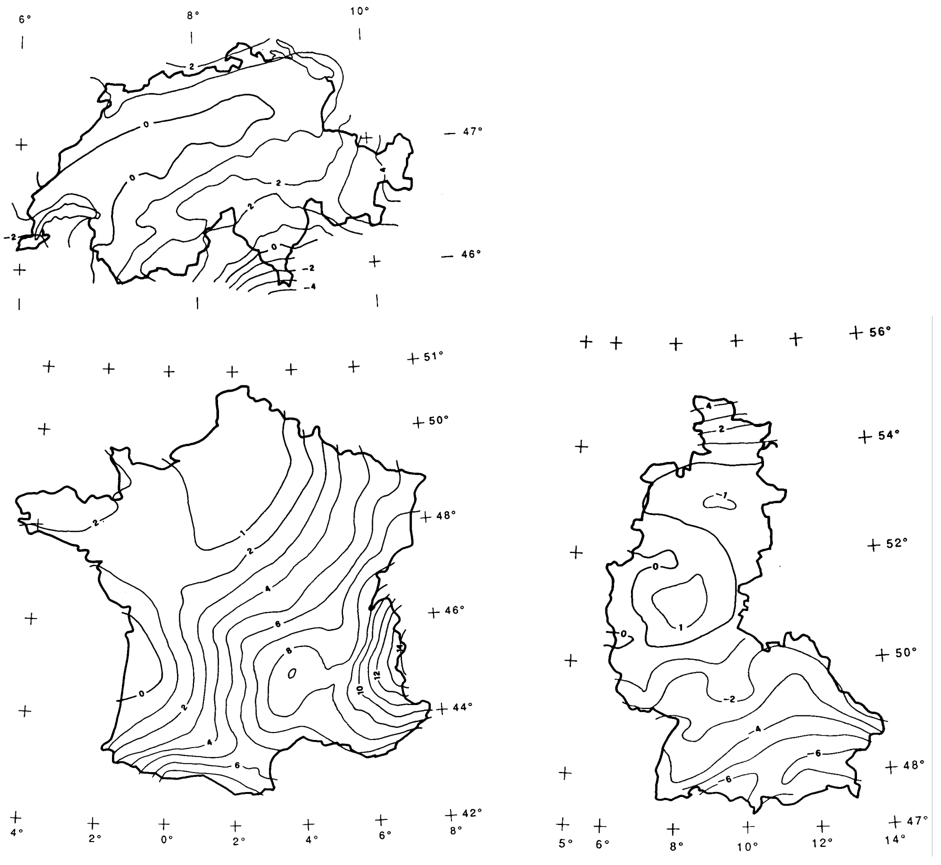


Figure 36: There may be a difference of several meters between the gravity height and the geoid height. Above you see the difference between the commonly used ellipsoid of Switzerland and the one of France and former West Germany.

The conversion between gravity heights and ellipsoid heights may pose a problem wherever the range of admissible coordinates extends over an area whose gravity field no longer is homogenous. Luckily these questions are of minor importance when modeling. Nevertheless they are worth a quick reflection.

6.8 Is zero up north? – Definitions for angles and directions

How big is a right angle? 90 Degrees or Pi / 2? This is a question of the unit in use. But when is the angle considered positive, when negative? Consequently the sense of rotation belongs to every type of angle: clockwise or counter clockwise.

```
DOMAIN
   AngleClockwise = -179 .. 180 CIRCULAR CLOCKWISE;
   AngleCounterClockwise = -179 .. 180 CIRCULAR COUNTERCLOCKWISE;
```

Standing on Mount Ilis we might want to know in which direction we have to look in order to see Twisted Peak. 50 degrees? 40 degrees? 310 degrees?

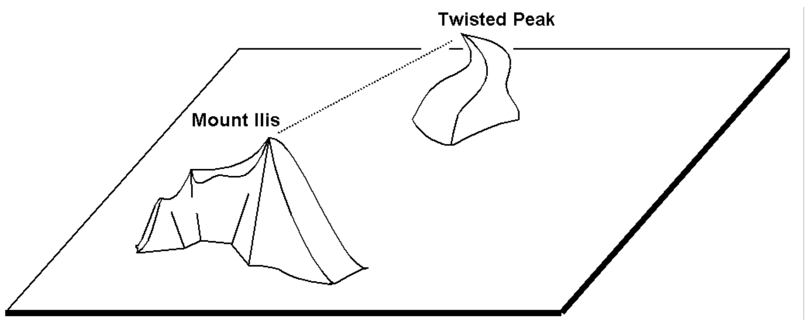


Figure 37: Whoever climbs Mount Ilis will be rewarded with a spectacular view of the surrounding mountains.
But in what angle will the Twisted Peak come into view? Unless we know what coordinate system the question refers to, it is impossible to offer an answer.

It all depends on the zero direction and how directions rotate. Whenever we speak of directions we also have to mention a reference system. That is why directions are closely linked with coordinate types. After all it also makes sense to determine the distance and direction of two points defined by coordinates.

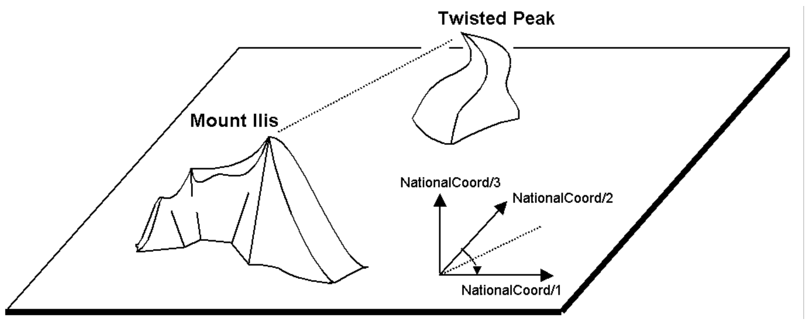


Figure 38: The indication of axes and rotation is part of the definition of a coordinate system.

```
NationalCoord = COORD 500.00 .. 91000.00 [m] {AhlandSys[1]},
                      700.00 .. 23000.00 [m] {AhlandSys[2]},
                     -200.00 .. 14000.00 [m] {AhlandHeightSys[1]},
                      ROTATION 2 -> 1;


Direction = 0.0 .. 359.9 CIRCULAR [Angle_Degree] {AhlandSys};
```

6.9 Is a ski run a line or a surface? – Geometry types

6.9.1 Simple conceptual view of a line

From the skiers' view point everything is clear: They want to know where a run starts, where it ends and roughly where it goes through. Is there a pub somewhere along the way? Does the run go over open hills or through a forest? For such information it is enough to describe the ski run as a line.

To begin with you may imagine a line type to be exactly what the word says: A more or less complicated connection between two points.

In this sense a line type is nothing but a numeric type or even better a coordinate type. Since all points concerned with this line have to be described by means of coordinates a line type will of necessity always have to be linked to a coordinate type.

In INTERLIS we could write:

```
AhlandLine = POLYLINE VERTEX Ahland.NationalCoord;

CLASS SkiRun =
  Course: AhlandLine;
END SkiRun;
```

The ski run is described by means of lines that are based upon the Ahland projection coordinate system. Thus the vertices of the lines in the Ahland reference system rely on the coordinate type of the reference system.

6.9.2 Segments

It is obvious: The ski run from Mount Ilis to Ilis Rock is a complicated line. In comparison the ski runs by the pony lifts are relatively simple. Could they all be described with the same type? The solution lies in breaking up the line as a whole into individual segments. Each segment itself is a simple geometry (e.g. a straight, part of an arc) and connects to its predecessor.

This state of matters could also be represented in the conceptual model. Then again this would be an unnecessary burden. Once it has been stated that lines are always structured in this way, this need no longer be displayed.

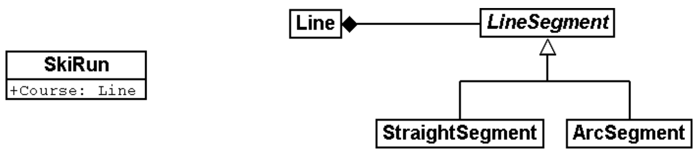


Figure 39: The course of a ski run is a line. It consists of individual segments that can be of various types: straight segments, arc segments, etc.

It definitely makes sense to indicate which types of segments may occur with one specific line type.

```
AhlandLine = POLYLINE WITH (STRAIGHTS, ARCS) VERTEX Ahland.NationalCoord;
```

This INTERLIS 2-definition indicates that lines of this type may feature straights and arcs.

In many cases – and it is the same with ski runs – it does not make sense if a line intersects itself. Such restrictions also belong to the conceptual model. However because of minor imprecision in the course of surveying (and partially also when computing) it is possible that a form that is actually without overlaps does end up with slight overlaps. Hence a maximum of an admissible overlap is part of the model.

Since the Ahland projection coordinate system uses meters, this definition permits overlaps up to 2 cm:

```
AhlandLine = POLYLINE WITH (STRAIGHTS, ARCS)
                      VERTEX Ahland.NationalCoord
                      WITHOUT OVERLAPS > 0.02;
```

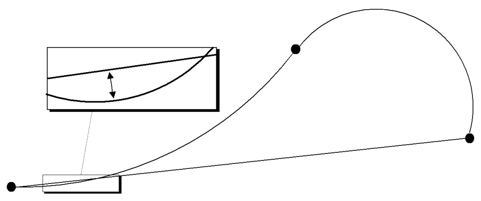


Figure 40: Sometimes small overlaps cannot be omitted. It is part of the model to define the maximum overlap (in this figure the height of arrow).

6.9.3 Directed lines

Of course any skier would expect the segments of the ski run from Mount Ilis to Ilis Rock to start at Mount Ilis and to end at Ilis Rock. After all the idea is to go downhill and not to climb up a slope! Then again for the description of other objects (e.g. hiking paths) direction is of no importance. Whenever the direction of lines is important this should be shown in the conceptual model.

```
AhlandLineDirected = DIRECTED POLYLINE VERTEX Ahland.NationalCoord;

CLASS SkiRun =
  Course: AhlandLineDirected;
END SkiRun;
```

6.9.4 Surfaces

For the maintenance service of the Mount Ilis Alpine Transports the question arises whether the description of the ski runs fulfils their demands. In order to clarify which of the areas have to be prepared a presentation in the form of a surface seems preferable.

```
DOMAIN
  AhlandLineDirected = DIRECTED POLYLINE WITH (STRAIGHTS, ARCS)
                       VERTEX Ahland.NationalCoord;

  AhlandSurface = SURFACE WITH (STRAIGHTS, ARCS)
                  VERTEX Ahland.NationalCoord;


CLASS SkiRun =
  Course: AhlandLineDirected;
  Prepared: AhlandSurface;
END SkiRun;
```

Just before Ilis Rock a big tree stands in the middle of the ski run – in other words the ski run passes on either side of the tree.

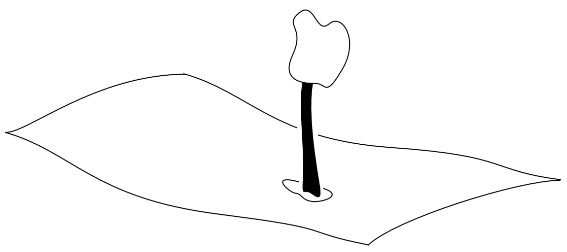


Figure 41: There is a big tree in the middle of the ski run. This might prove to be quite tricky for skiers, but we need not worry about the data model: Despite this the ski run will remain one whole surface.

Is the surface that has to be prepared still one whole surface? Surface – at least in the sense of INTERLIS – always means coherent areas. Even if they have interior blank spaces (holes, enclaves), they remain coherent areas and thus can be described as surfaces.

▶ A surface has exactly one **outer boundary**. It may possess none, one or several **interior boundaries** (enclaves).

At the very top of the Mount Ilis several ski runs are that close that as a result there is one common prepared surface. Now which part of this surface should be assigned to which ski run? In Ilis Dale two ski runs cross. Hence this surface is gathered twice which of course interferes when trying to evaluate the precise amount of work involved in preparing these ski runs.

That is why the maintenance service has decided to use a different form of modeling: instead of assigning the surfaces that are to be prepared directly to the ski runs, they are considered as independent segments of a ski run. Each segment is a surface. However these segments should never overlap, since one particular area will only have to be prepared once.

```
DOMAIN
  AhlandTessellation = AREA WITH (STRAIGHTS, ARCS)
                           VERTEX Ahland.NationalCoord;

CLASS ConditionOfSkiRun =
  PreparedSurface: AhlandTessellation;
END ConditionOfSkiRun;
```

Because such surfaces without overlaps are quite common, INTERLIS has introduced its proper type (AREA). Instead of surfaces we speak of (planar) tessellation.

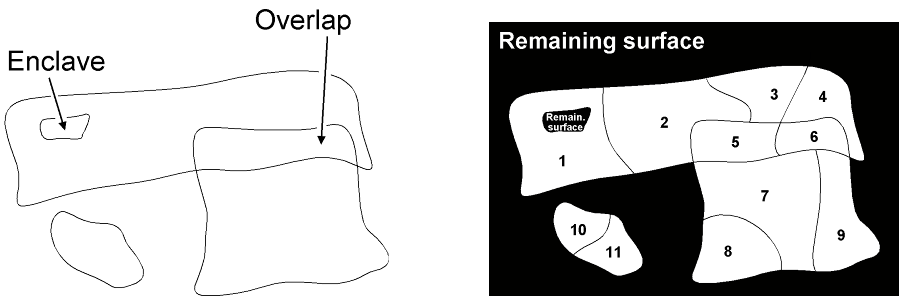


Figure 42: With the ordinary type of surface (SURFACE, left) surfaces of different objects may overlap. For instance there is nothing to stop one piece of land to belong to two ski runs. In the case of a tessellation (AREA, right) it is required that each point within the land be assigned unequivocally to one object, unless it were to belong to the remaining surface (shown in black); one example being the segments prepared by the maintenance service.

6.9.5 Three-dimensional line types

If a coordinate type belonging to a line definition is a three-dimensional type, then the line type also is three-dimensional. INTERLIS 2 does without stating the third dimension as equal besides the other two, because in geographical applications all three dimensions can always be subdivided into the position and information on height.

▶ INTERLIS 2 supports lines with 2.5 dimensions.

Thereby we proceed on the assumption that each vertex point between two segments is defined by its position and height and that the height on the segment will be subject to a linear interpolation according to the length of the segment.

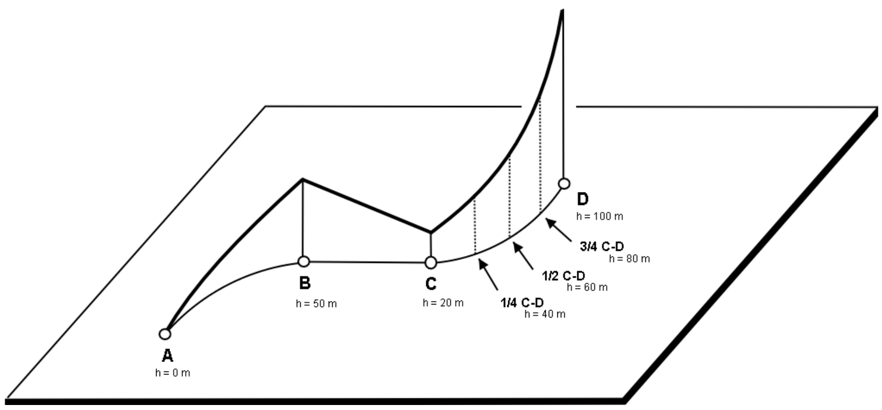


Figure 43: INTERLIS supports 2.5-dimensional lines: The Height between two vertexes is always subject to a linear interpolation. If at a given point on the ground a quarter of the distance

between C and D has been covered, we assume that at the same time a quarter of the difference in altitude has been conquered.

Now shouldn't we model the course of a ski run with a three-dimensional line type? From a purely technical point of view this would pose no problem, and after all elevation plays an important part in skiing. On the other hand the altitude of the course is no independent figure: Where the position is known, the height is a logical result of the terrain features. Thus we can calculate the elevation of the course of the ski run from its position and a topographical model. Hence from a conceptual point of view we prefer to do without the information on height when dealing with the course of a ski run.

The case may be different for roads and railways because with bridges and tunnels height and terrain height may not be the same. In some cases a degree of precision will be demanded for the height that renders a derivation from a topographical model impossible. In certain cases it may make sense to model artificial constructions (with height) independently of the course of a track. In such a case the actual height of the track within the range of artificial constructions would be computed from the model; at other places we would rely on the topographical model.

With this issue a decisive criteria would be the expenditure for collection and update.

6.10 What way does the wind blow? - Structures

6.10.1 Multiple properties

Just before Ilis Rock travelers on the chair lift from Ilis Dale pull their hats tightly over their ears: At this spot the wind tends to rip and whistle. When it comes to wind, it is not only speed that is decisive, but also direction. If, in a class description, both these properties simply appear along with other attributes, not enough emphasis will be put on this relevant fact.

```
CLASS Weather =
  Temperature: MANDATORY -50 .. 50 [oC];
  WindDirection: MANDATORY (N, NE, E, SE, S, SW, W, NW) CIRCULAR;
  WindSpeed: MANDATORY 0 .. 200 [kmh];
END Weather;
```

In situations where a specific fact cannot be described by only one value, it makes sense to define a structure combining both characteristics (wind direction, wind speed).

```
STRUCTURE IndicationOfWind =
  WindDirection: MANDATORY (N, NE, E, SE, S, SW, W, NW) CIRCULAR;
  WindSpeed: MANDATORY 0 .. 200 [kmh];
END IndicationOfWind;
```

Other concepts related to structure are: data type, structured data type, ….

This particular structure can be used whenever we deal with comments on wind.

```
CLASS Weather =
  Temperature: MANDATORY -50 .. 50 [oC];
  Wind: IndicationOfWind;
END Weather;

CLASS WindMeter =
  Position: MANDATORY NationalCoord;
  Wind: IndicationOfWind;
END WindMeter;
```

6.10.2 Several elements of structure

The wind-meter stationed at Ilis Rock is more special yet: It does not only display the current value, but also six values recorded previously. While this of course does not warm your ears, it is still quite astonishing to note how swiftly conditions can change.

```
CLASS WindMeter =
  Position: MANDATORY NationalCoord;
  Wind: LIST {6} OF IndicationOfWind;
END WindMeter;
```

Therefore the attribute wind comprises six elements (six values each with speed and direction). LIST Of states that their order is relevant (e.g. the most recent value comes first). If the order were not relevant, this would be indicated by BAG OF. As with relationships there is a possibility to indicate the minimum and maximum number of elements.

6.10.3 Structures and classes

From a formal point of view structures and (object) classes are quite similar, however with regard to their content there are considerable differences. A class (railway company, wind-meter) describes how objects are formed, organized. A structure describes more complex characteristics such as indication of wind. Hence a structure serves the same goal as a domain, it describes the organization of an attribute. Sometimes a structure only becomes necessary, if a characteristic must be described more in detail; for simple descriptions indicating a domain is sufficient (cf. paragraph 6.12).

Instances of classes are objects in their own right (Mount Ilis Alpine Transports, wind-meter at Ilis Rock). The instances of structures are structure elements (wind with a speed of 180km/h, blowing from a north-easterly direction). The value of a structure attribute can comprise exactly one structure element or a set of structure elements (BAG OF, LIST OF).

▶ While formally a **structure** closely resembles an object class, with regard to its contents it resembles a domain. However its corresponding items, its **structure elements,** do not have an identity of their own, they merely are values of attributes of an object.

Whereas relationships may exist between objects (cf. paragraph 6.13), the same is not possible between values (of domains or structures). However it is possible to compare similar values of distinct objects (and distinct classes) and thus establish a kind of relationship (cf. paragraph 6.17). For instance one might compare the price of a hiker’s pass with the price of a steak one intends to order at the restaurant on top of Mount Ilis. Nevertheless there is no relationship between the hiker's pass and the steak.

In some cases it is necessary to refer to another object in order to describe a particular characteristic (cf. paragraph 6.11.3). Yet it will never be possible to refer to a value or a structure element, as they have no identity.

6.10.4 Lines are special structures

The attribute course of a ski run (cf. paragraph 6.9.1) is defined as AhlandLine, which in turn is defined as POLYLINE. A POLYLINE may be understood as a set of segments of a line (cf. paragraph 6.9.2). Thus the definition as POLYLINE is nothing but a shortened notation for an ordered set of structures, the structures elements corresponding to a certain structure definition:

```
STRUCTURE AhlandSegment (ABSTRACT) =
  SegmentEndPoint: MANDATORY Ahland.NationalCoord;
END AhlandSegment;

STRUCTURE AhlandStartSegment EXTENDS AhlandSegment (FINAL) =
END AhlandStartSegment;

STRUCTURE AhlandStraightSegment EXTENDS AhlandSegment (FINAL) =
END AhlandStraightSegment;

STRUCTURE AhlandArcSegment EXTENDS AhlandSegment (FINAL) =
  ArcPoint: MANDATORY Ahland.NationalCoord;
  Radius: Length;
END AhlandArcSegment;

CLASS SkiRun =
  Course: LIST {2..*} OF AhlandSegment;
END SkiRun;
```

6.11 What languages are spoken in Ilis Valley? – Multilingualism

6.11.1 One attribute per language

In the existing model a railway company has got one name and an abbreviation. How can we then collect the information that the Mount Ilis Alpine Transports (MIT) in German are called *Ilishornbahnen (IhB)*?

It seems natural to supplement the object class RailwayCompany with the German name and its abbreviation:

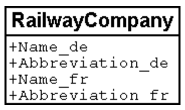


Figure 44: The object class RailwayCompany with names and abbreviations, in German and French respectively.

So much for that. But what, if at a later time somebody came up with the idea to gather the name in a third, fourth or even fifth language? Basically no problem – it would only mean a minor alteration in the data model!

As a matter of fact it is no big deal to enlarge a little box on a piece of paper and to add a few lines. However once the computer system has been realized, even such a small alteration may turn out to be quite costly: application forms have to be changed, programs need to be adjusted, data have to be gathered again, etc.

6.11.2 Language dependent terms as structure elements

Hence it would be preferable not to state a concrete language in the data model. In the following new version one railway company displays a set of designations. As it is a common requirement to deal with several languages, the structure RailwayDesignation will inherit the basic structure designation, itself comprising the language and a text.

```
STRUCTURE Designation =
  Name: TEXT;
  Language: TEXT*2;
END Designation;

STRUCTURE RailwayDesignation EXTENDS Designation =
  Name (EXTENDED): TEXT*100;
  Abbreviation: TEXT*10;
END RailwayDesignation;

CLASS RailwayCompany =
  Names: BAG {1..*} OF RailwayDesignation;
END RailwayCompany;
```

Or in a graphic representation:

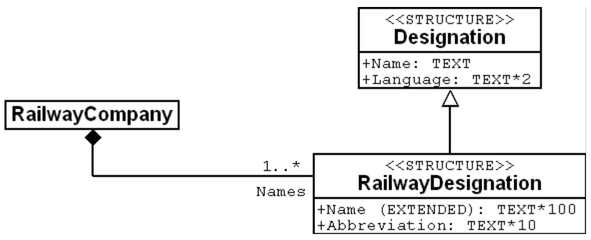


Figure 45: Railway designations are assigned to one railway company. Since one company may possess several names it is possible without further expenditure to add new names in different languages. Details of this assignation (indications such as 1..* or a filled-in square) will be explained below in connection with relationships.

Remember that every text attribute is not necessarily multilingual. For instance family names of persons are not translated.

When adding designations in another language, it is sufficient to gather new data. There is no need to adjust the data model.

6.11.3 Structure elements may refer to objects

Who would know the official language abbreviation for Romantsch? rr? rm! Within the scope of the National Tourist Office it is obvious which languages are in consideration for the designations of railway companies. In most cases when collecting data of one line, only one abbreviation has to be taken into account. This is not difficult to retain and hence the National Tourist Office has built its model as described above.

If this were not the case, a model would have been chosen where languages would feature as actual objects. Then the language object would contain both abbreviation and e.g. name as a text in their own language and in English.

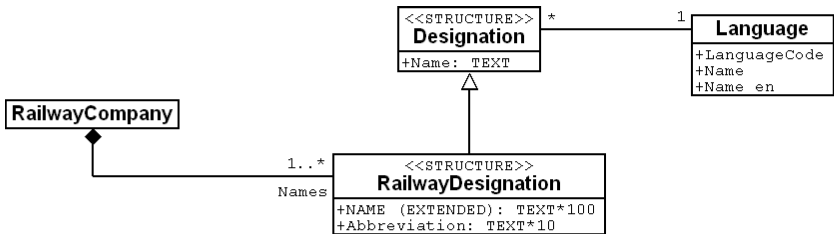


Figure 46: In this variant the language designation (a structure) refers to the language (a normal object class).

Thus the designation refers to the language. However this reference is not a full relationship (cf. paragraph 6.13), since these designations have no identity. Consequently from the point of view of a language object there is no direct access to the designation elements. This would first have to be established by means of a view (cf. paragraph 6.17).

6.12 How do clocks tick in Ilis Valley? – Modeling time

6.12.1 Sufficient for modest pretensions

The National Tourist Office provides a simple solution in the form of an attribute for the validity of ticket types, which shows the number of days (with one decimal).

```
Validity: 0.0 .. 1000.0 [d];
```

If – like people in Ilis Valley – one is more concerned with details, several questions arise:

- A ticket valid on the day of issue has not the same validity as one that is valid for 24 hours.
- A month may have 28, 30, or 31 days.
- A year may either have 365 or 366 days.

Upon their inquiry, the National Tourist Office offered the answer that the following rules applied to questions of validity:

- 0.9: on the day of issue;
- 30.0: one month;
- 365.0: one year.

⚠ Such makeshift solution may at times seem appealing, because they appear to be simple. But what if 30.0 days really means that exact number of days and not one month? It pays to proceed with precaution!

But what then would be a better solution?

6.12.2 Length of time as structure

It is not always possible to describe with sufficient precision object characteristics such as validity by one single value. Sometimes a whole group of attributes is necessary, sometimes it makes sense to plan several extensions. That is where structures provide a convenient solution.

```
STRUCTURE LengthOfTime (ABSTRACT) =
END LengthOfTime;

STRUCTURE LengthOfTimeToday EXTENDS LengthOfTime =
END LengthOfTimeToday;

STRUCTURE LengthOfTimeInDays EXTENDS LengthOfTime =
  Duration: MANDATORY Days [d];
END LengthOfTimeInDays;

....

CLASS TicketType =
  Validity: LengthOfTime;
END TicketType;
```

The validity of one particular ticket type is described by means of an instance (a structure element) of the structure LengthOfTimeToday, LengthOfTimeInDays, LengthOfTimeInMonths etc. We could even go further in our modeling and make sure that the units of an explicit duration (day, month etc.) always have to be a length of time and define an enumeration for implicit duration (week, season etc.):

```
STRUCTURE LengthOfTime (ABSTRACT) =
END LengthOfTime;

STRUCTURE LengthOfTimeImplicit EXTENDS LengthOfTime =
  Duration: MANDATORY (Day, Week, Month, Year);
END LengthOfTimeImplicit;

STRUCTURE LengthOfTimeExplicit (ABSTRACT) EXTENDS LengthOfTime =
  Duration (ABSTRACT): MANDATORY NUMERIC [TIME];
END LengthOfTimeExplicit;

STRUCTURE LengthOfTimeInMinutes EXTENDS LengthOfTimeExplicit =
    Duration (EXTENDED): MANDATORY 0 .. 200 [Units.min];
END LengthOfTimeInMinutes;

STRUCTURE LengthOfTimeInDays EXTENDS LengthOfTimeExplicit =
  Duration (EXTENDED): MANDATORY 0 .. 1000 [d];
END inDays;
```

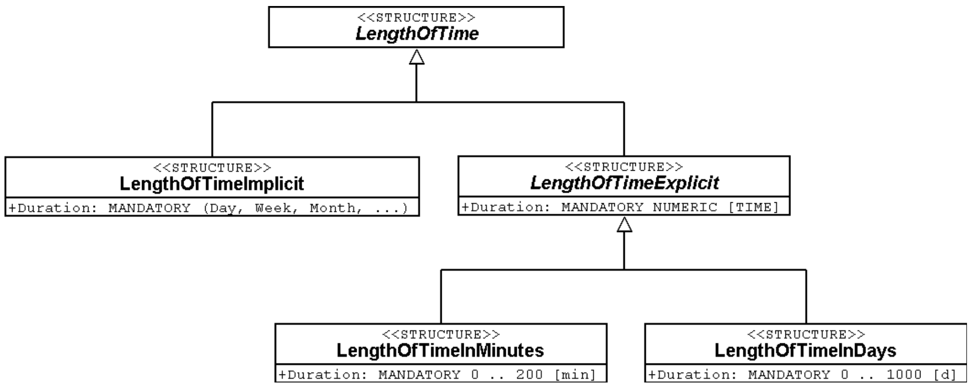


Figure 47: Length of time in detailed modeling with structures. Thus it is possible that if required the validity of a ticket can be either one month (LengthOfTimeImplicit; left) or exactly thirty days (LengthOfTimeInDays; right).

Basically we aim at a precise, detailed and appropriate modeling. However one should always keep in mind that this only makes sense if it can also be translated into action. What does this mean for program packages? And moreover: What does it mean for the people that gather and deal with data? And vice versa: What does it mean if we differ from the most correct of all possible models? All things considered it may pay to be satisfied with a more simple solution.

6.12.3 Exact length of time

Length of time does not only exist for tickets. Every Friday Ilis Valley organizes a ski race for their guests. Racing times are measured in minutes, seconds and their fractions. For this we could define a structure, which features the attributes hours, minutes and seconds: It seems natural to define a structure featuring the attributes minutes and seconds:

```
STRUCTURE LengthOfTimeInMinutes EXTENDS LengthOfTime =
  Minutes: 0 .. 9999.99 [min];
  Seconds: 0.00 .. 59.99 [s];
END LengthOfTimeInMinutes;
```

To express the relationship between minutes and seconds, the following additional solution presents itself:

```
STRUCTURE LengthOfTimeInMinutes EXTENDS LengthOfTime =
  Minutes: 0 .. 9999.99 [min];
  CONTINOUS SUBDIVISION Seconds: 0.00 .. 59.99 [s];
END LengthOfTimeInMinutes;
```

This does not determine in what form such a length of time would be memorized in a computer. It simply is a means of describing as precisely as possible what we really want.

6.12.4 Formatted representation of structures

The traditional ski race for guests is always designed in a way that even skiing instructors will take more than 30 seconds to do the course. Participants that take more than 3 minutes and 30 seconds are offered a cup of tea at the finish, but their result will not be registered.

How do we record the admissible domain (from 30 seconds to 3 minutes and 30 seconds)? The solution is provided by formatted domains:

```
DOMAIN LengthOfTimeinMinSec = FORMAT BASED ON LengthOfTimeInMinutes
                                       ( Minutes ":" Seconds );

CLASS RaceTime =
  FirstName: TEXT*50;
  Surname: TEXT*50;
  Runtime: FORMAT LengthOfTimeinMinSec "0:30" .. "3:30";
END RaceTime;
```

A formatted domain refers to a structure and determines the design of a string of symbols composed of the individual attributes of the structure and text constants representing the value. In this form it is possible to define restrictions for the domain. This formatted representation will also be used for data transfer. Thus it may be possible to directly support certain representation forms demanded by external authorities. This is of special interest fort the XML-conform representation of duration and points in time.

6.12.5 Moments in time

Reports regarding weather conditions, waiting times, conditions of the ski runs in Ilis Valley shall always state the time at which these conditions were observed. First thought: Time in hours and minutes. Yes, and in order to establish statistics, the respective date. That should be enough!

Really? On good nights with a full moon the Mount Ilis Alpine Transports do extra runs up to Mount Ilis, where then the popular Dracula-Party takes place. So even in the middle of the night reports on conditions are issued. Even at 2.30 am. Also on that early Sunday morning when the hands of the clock were switched over to daylight saving. However that was quite chaotic. Suddenly the latest report dated further back than the last! Naturally enough: That night any time between 2 am and 3 am appeared twice.

▶ With moments in time it is always important to know the respective reference system.

Are we speaking of summertime, wintertime, UTC? The more international we get, the more important it is to know. It is a short step to the idea to report everything in UTC and to leave it up to the computer to present the data to the user according to his current time zone.

INTERLIS 2 not only offers the possibility to describe domains and units, but also reference systems. For UTC-times already formatted domains in accordance with XML-rules have been predefined (XMLTime, XMLDate, XMLDateTime).

Then again opening and operating hours preferably are described in local time. After all midnight is at 24.00, whether it is summertime or wintertime. These are not moments in time in the true sense of the word; they actually describe differences to the midnight hour according to the currently valid time.

⚠ Wherever time, and above all precise moments in time are of importance, we have to proceed with outmost care.

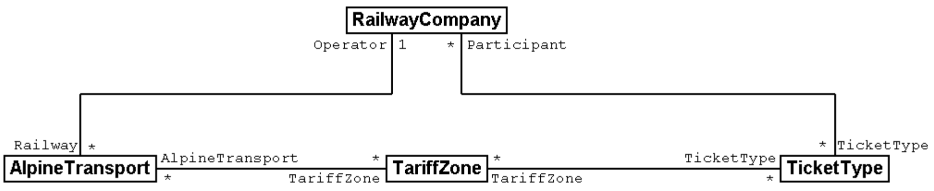
6.13 Tariff zones, reports on conditions – Relationships

6.13.1 Roles

What exactly is a railway company for one particular alpine transport? Proprietor? Operator!

In the relationship between RailwayCompany and AlpineTransport the railway company fulfils the role of operator.

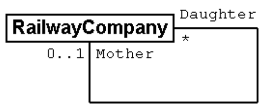
In the graphic representation the role name appears at the end of relationship line on the side of the holder of the role. However if it is no different from the class name, then in most cases the role name is omitted.



```
ASSOCIATION =
  Operator -- {1} RailwayCompany;
  Railway -- {*} AlpineTransport;
END;
```

Figure 48: According to this model it is possible to inquire after the operator of an AlpineTransport. «Operator» is a *Role* that the class «RailwayCompany» holds in view of the class «AlpineTransport». Below this relationship between RailwayCompany and AlpineTransport is rendered in INTERLIS notation.

It is quite common to select role names that do not differ from the class name. For instance in the relationship AlpineTransport – TariffZone there is little sense in introducing further names. Nevertheless the need for additional names is quite obvious if a relationship exists between objects of the same class. We might like to display the fact that one company owns other railway companies as subsidiary companies.



```
ASSOCIATION =
  Daughter -- {*} RailwayCompany;
  Mother -- {0..1} RailwayCompany;
END;
```

Figure 49: A railway company may be parent company but also subsidiary of another railway company. In such cases the class name is not suitable as role name. This example is displayed on the left the graphic notation of UML, on the right in the textual notation of INTERLIS.

6.13.2 Force of a relationship

Association, Aggregation and composition express the difference in force of relationships.

- **Association** – The relationship between tariff zone and alpine transport is rather loose. Two objects are linked without being sub-ordinate to the other. An association is a relationship between equals. Very often in a data model the greater number of relationships are common associations.
- **Aggregation** – An alpine transport is a rather independent object. Yet there always has to be a railway company to run it. The railway company is always superior to the alpine transport.
- **Composition** – A very close relationship exists between an alpine transport and its pylons. In actual fact a pylon only makes sense in connection with a certain alpine transport. A composition is relationship between a whole and its (mainly physical) parts.

It is not always easy to classify according to these forces. From the view point of IT there are other rules, which sometimes will simplify this decision:

- **Delete** – If a railway company is deleted this will means that the assigned alpine transports are now without a manager. However if an alpine transport is deleted, all pylons will also be deleted. Deleting a whole also means removing all its parts that are connected via a composition.
- **Copy** – If we copy a railway company (in real life of course not as simple as on the computer), we will at the same time establish copies of all assigned alpine transports which then will be assigned to the new railway company. Accordingly copies of all the pylons are established for each of these alpine transports. Copying an object also means duplicating all those objects assigned by a common association.

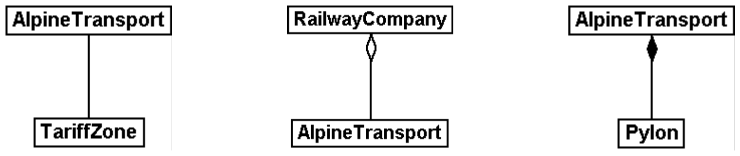


Figure 50: Association (left), aggregation (middle) and composition (right) are different types of relationships. They differ in their binding force: A Pylon is so closely tied to its AlpineTransport that it can be considered a part of it. In comparison with a composition, both association and aggregation are weaker.

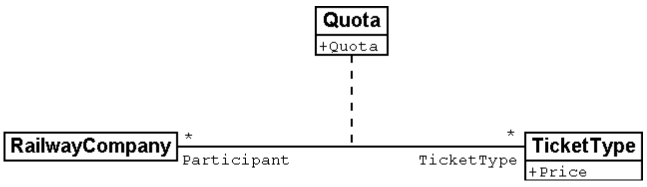
The INTERLIS notation is copied from the graphic representation. However the role name has to be written even if does not differ from the class name.

```
ASSOCIATION =
  AlpineTransport -- AlpineTransport;
  TariffZone -- TariffZone;
END;

ASSOCIATION =
  Operator -<> RailwayCompany;
  AlpineTransport -- AlpineTransport;
END;

ASSOCIATION =
  AlpineTransport -<#> AlpineTransport;
  Pylon -- Pylon;
END;
```

6.13.3 Relationships with attributes

Various ticket types entitle to a ride on alpine transports run by different railway companies. This brings us to the question how the proceeds of the ticket sale should be divided amongst these companies. For instance the national general yearly season ticket also entitles its owner to rides on the Mount Ilis Alpine Transports. Based on an agreement the Mount Ilis Alpine Transports receive 0.13% of the turnover a general yearly season tickets in return.

Relationships can also feature attributes and hence have the nature of special classes.

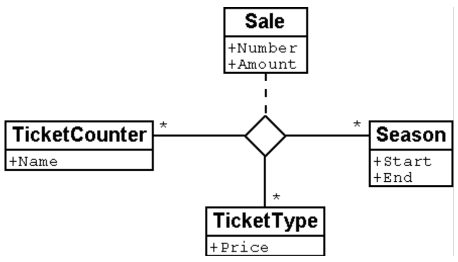


```
ASSOCIATION Quota =
  Participant -- {*} RailwayCompany;
  TicketType -- {*} TicketType;
ATTRIBUTE
  Quota: 0.00 .. 100.00 [Units.Percent];
END Quota;
```

Figure 51: A RailwayCompany has a predetermined Quota in the profits from the sale of a particular TicketType. The percentage agreed upon is neither a characteristic of the railway company nor of the TicketType. Instead we deal with a characteristic of their relationship. Such situations are modeled with relationship classes.

6.13.4 Multiple relationships

In order to gain a better overview of all the ticket sales, the National Tourist Office would like to record in the future which ticket counter has sold how many of one ticket type in which season.



```
ASSOCIATION Sale =
  TicketCounter -- {*} TicketCounter;
  Season -- {*} Season;
  TicketType -- {*} TicketType;
ATTRIBUTE
  Number: 1 .. 999999 [Units.CountedObjects];
  Amount: 0.00 .. 9999999.99 [Ahland.Sovereign];
END Sale;
```

Figure 52: The Sale is captured per TicketCounter, TicketType and Season. We deal with a multiple relationship between three equal partners (the classes TicketCounter, TicketType and Season). In contrast «Sale» is a relationship class, which captures characteristics of the relationship (e.g. the number of tickets sold as well as the amount).

Thus there is an equal relationship between ticket counter, ticket type and season, which also captures in the form of attributes the number of tickets sold plus the turnover. So this relationship does no longer link two but three classes.

So then what do the indications of cardinality exactly mean in multiple relationships? Cardinality e.g. with the season (*) means that for a particular combination of ticket type and ticket counter there may be any number of assignations to season objects. Were we to indicate cardinality 1, then a certain ticket type could only be sold for one season by one specific ticket counter.

Slightly complicated. Do we really need multiple relationships or could we reduce them to the common one-to-one relationships?

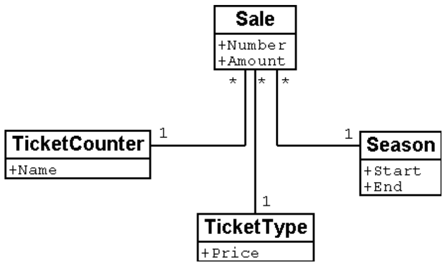


Figure 53: Relationships between more than two parties can be reduced to common one-to-one relationship. The former relationship class (in this instance: Sale) becomes an equal partner and now all the parties concerned are only related to the former relationship class.

However this model will express less clearly the fact that the three classes TicketCounter, TicketType and Season are related as a group of three.

6.13.5 Directed relationships

Looking at all the alpine transports assigned to the company Mount Ilis Alpine Transports, we observe that there is no certain order. The question whether in an assignation an aerial cable car should appear before or after the gondola does not really make sense.

Of course we could list all means of transport of one company in alphabetical order.

But this sorting would not be a characteristic of the relationship between company and alpine transport but merely a question of representation. Under different circumstances a sorting according to investment costs, travel time etc. could be interesting.

But wouldn't it make sense if this list captured the order in which the relationships were established? To start with the aerial cable car was inaugurated, then the ski lift, followed by the gondola etc. Then again in this case it would be better to supply the relationship with the attributes establishment and closure. Then it would even be possible to record the different managers in the course of time. So in this case it would no longer make sense to consider the relationship as an aggregation.

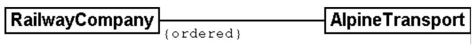


Figure 54: To record the order in which alpine transports of one company have started operating, we could use a directed relationship. However the model in the figure below is better.

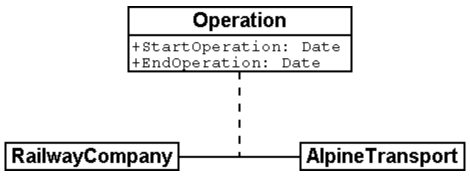


Figure 55: The model with a relationship class is clearer because it will permit further evaluations. For instance it allows the sorting of one company's means of transport according to their shutdown and a computer program may display past managers of one alpine transport.

Similar considerations apply to the relationship between alpine transports and pylons. By putting in order this relationship we could express the succession of bottom to top station. But from the conceptual point of view it is preferable to introduce a position attribute with a pylon and then to derive the succession from this position and the course of the track.

Before declaring a relationship ordered, consider carefully whether the order could not be derived from attributes of classes concerned or from the relationship itself.

So where do ordered relationships really make sense? The gondola from Ilis Bath to Mount Ilis has individual gondolas that are not mounted fixedly on the cable. They can be taken off at either the bottom or top station and, when needed, be replaced. At present which gondolas are mounted in which order on the cable?

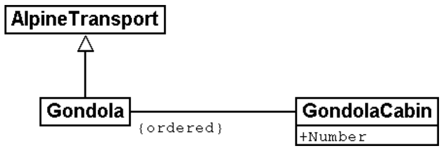


Figure 56: Gondola cabins may have numbers but these will not determine their order on the cable. In this instance an ordered relationship makes sense.

For once order is of interest. The number of a gondola cannot be used for establishing order. It simply identifies one specific gondola. It has nothing whatsoever to do with their current order on the cable.

6.13.6 Extending relationships

A railway company is related to a number of persons. Some are employed by it, others have quotas in it. Analogous to the different kinds of alpine transports there are various possibilities for modeling.

One possibility consists of defining two different relationships between railway company and persons: one for employment, one for participation. In case occasionally this differentiation should not be of interest (perhaps when sending little chocolate trains before Christmas), an application would have to concern itself with both relationships.

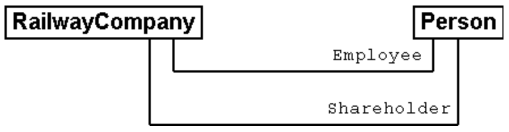


Figure 57: A Person may be employee and/or shareholder of a RailwayCompany as modeled above with two different relationships. Should the RailwayCompany intend to treat either of them to a Christmas surprise, both relationships would have to be evaluated.

Another possibility of modeling would consist in primarily defining a relationship (contact), which then would be extended into employment or participation. As long as the type of contact person – railway company is irrelevant for an application, it uses the contact-relationship and consequently obtains everybody that in some way has contact with the company. An application where only employees are relevant would use the extended relationship Employment and thus would only obtain employees.

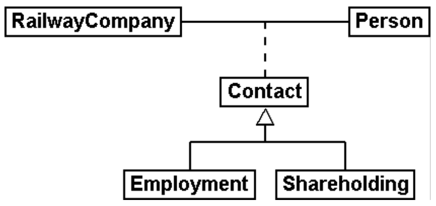


Figure 58: In this variant the relationship between RailwayCompany and Person is modeled in a general way with the relationship class «Contact», Employment and Shareholding are special cases of a contact. Whoever inquires after the contacts of a company will automatically also obtain employees and shareholders. Hence in a similar way as object classes relationship classes can be extended, which in the diagram is shown by a white arrow.

This employment-relationship could be further extended and for instance a relationship «Management» could be introduced.

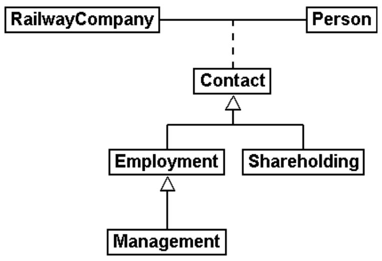


Figure 59: The relationship between a RailwayCompany and its managing director («Management») is a special case of the relationship «Employment».

Extensions of relationships often go hand in hand with the extension of object classes. Instead of stating right from the beginning that an alpine transport possesses pylons, to start with we only speak of rolling stock. These would be loosely assigned, i. e. by association to the means of transport. Since pylons are an important feature of different kinds of alpine transport, the class «AlpineTransportWithPylons» will be introduced. This class will have a relationship with the pylons. However it will be introduced as an extension of the relationship between alpine transports and rolling stock. Since pylons – opposed to a vehicle – directly belong to an alpine transport, this relationship becomes a composition. Note that in an extension the force of a relationship can only be strengthened but not loosened, so as not to contradict the definition in the basic definition.

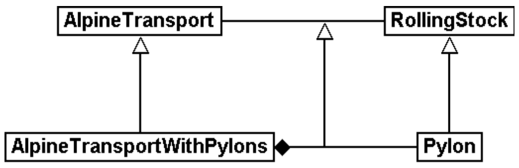


Figure 60: AlpineTransport and RollingStock lead a general relationship, strengthened into a composition by specialized classes.

6.13.7 Derivable relationships

If your stomach rumbles, you tend to go for a ski run that passes a hotel. This does not mean that ski runs and hotels necessarily have to be on a constant, explicit relationship. It is enough to know that there is a hotel near the ski run. A statement that can be derived from the position of the hotel and the course of the ski run (both in projection coordinates)

⚠ Not everything belonging together within the scope of evaluations necessarily needs to be linked by relationships. Especially with spatial data coordinates are an ideal tool to establish connections when needed.

There is no point either in adding all derivable relationships to the conceptual model. Consequently you will not find the derivable relationship between hotels and ski runs in the conceptual model.

⚠ In a conceptual model we only want to describe those implicit relationships that are of conceptual importance. In addition programs can establish further relationships by skilfully comparing attributes of the objects (not least of all according to their position).

Not least of conceptual importance are relationships that in some cases have to be defined explicitly and in other cases can be derived. Their derivation may depend on the geography or other characteristics. For instance Ilis Valley has introduced a special tariff zone described as a surface which comprehends all alpine transports whose bottom and top station lie within this surface.

```
CLASS TariffZoneInRegion EXTENDS NatTour.TicketsZone =
  Area: AhlandSurface;
END TariffZoneInRegion;
```

The relationship between this special tariff zone and the alpine transports in the corresponding tariff zone can be automatically established by means of views (cf. paragraph 6.17).

6.14 Unique MountIlisAlpineTransports – Consistency constraints

6.14.1 General remarks

We recall that the Mount Ilis Alpine Transports want to report information on current conditions for each of their alpine transports, amongst others the weather at the top station:

```
CLASS InformationOnCondition =
  Temperature: MANDATORY -50 .. 50 [oC];
  WindDirection: (N, NE, E, SE, S, SW, W, NW) CIRCULAR;
  WindSpeed: MANDATORY LengthOfTimeInMinutes;
  Captured MANDATORY MomentInTimeCET;
END InformationOnCondition;
```

With this definition even a report stating an undefined wind direction and a wind speed of 60 km/h would be acceptable. This is not our goal. An undefined wind direction should mean calm. But then of course wind speed would be 0. And vice versa with a wind speed greater than zero the wind direction should always be defined.

▶ Situations where a certain connection has to exist between different attributes of an object or even between different objects are described by means of **consistency constraints**.

As a rule consistency constraints are described by a formula whose interpretation will tell whether the condition is fulfilled or not. Thanks to such a logical expression we will come to terms with a lull:

```
CLASS InformationOnCondition =
  Temperature: MANDATORY -50 .. 50 [oC];
  WindDirection: (N, NE, E, SE, S, SW, W, NW) CIRCULAR;
  WindSpeed: MANDATORY 0 .. 200 [kmh];
  WaitingTime: LengthOfTimeInMinutes;
  Captured: MANDATORY MomentInTimeCET;
MANDATORY CONSTRAINT
  DEFINED (WindDirection) == (WindSpeed > 0);
END InformationOnCondition;
```

Precisely when the wind direction is defined, then the wind speed must be greater than zero. If no wind direction is defined, then speed has to be zero. Hence «Definiteness» of the wind direction must equal (==) «Positiveness» of the wind speed.

Very often consistency constraints could be omitted if only the model were structured differently. If we wrap up all indications regarding wind in a structure, and then state that this structure, there is no need for a consistency constraint. With no wind the structure element is missing. If it is blowing then forcibly there must be both wind direction and wind speed.

```
STRUCTURE IndicationOfWind =
  WindDirection: MANDATORY (N, NE, E, SE, S, SW, W, NW) CIRCULAR;
  WindSpeed: MANDATORY 0 .. 200 [kmh];
END IndicationOfWind;
```

```
CLASS InformationOnConditions =
  Temperature: MANDATORY −50 .. 50 [oC];
  Wind: IndicationOfWind;
  WaitingTime: LengthOfTimeInMinutes;
  Captured: MANDATORY MomentInTimeCET;
END InformationOnConditions;
```

⚠ With consistency constraints – especially if they are complicated – we always suspect that the optimal model has not yet been found. On the other hand there is no sense in making an extra effort to render a basically simple model complicated only to avoid a consistency constraint.

6.14.2 Plausibility constraints

Any employee of the Mount Ilis Alpine Transports makes good money, but still the managing director's salary is something different.

In general consistency constraints apply to all objects of the corresponding class. In INTERLIS 2 they are called MANDATORY CONSTRAINT. Occasionally we speak of «hard» conditions, because they always have to be fulfilled. But there are also conditions that on principle, but not always are complied with.

With attributes such as monthly salaries and body-height the generally admissible scope must be relatively. However the values of most objects will lie below a considerably lower limit, but exceptions may occur, for instance the salary of the managing director.

```
ASSOCIATION Employment =
  ...
  MonthlySalary: MANDATORY 0 .. 20000 [Sovereigns];
  ...
CONSTRAINT >= 95%
  MonthlySalary < 10000;
END Employment;
```

We estimate that in a minimum of (>=) 95% of all cases the monthly salary is below 10000 sovereigns. If a model comprises such «soft » condition, there is the possibility to check data with regard to plausibility and to examine it statistically during input.

6.14.3 Uniqueness constraints

How can we identify persons that either work for the company or are participants? It might seem obvious to use names and first names but this would be quite unsuitable:

```
CLASS Person =
  Name: TEXT*50;
  FirstName: TEXT*20;
UNIQUE Name, FirstName;
END Person;
```

Consequently it would become inadmissible to list to persons with the same combination of name and first name. Hence the new engine driver John Smith could only start his job after his namesake the accountant has been sacked.

What would be a better uniqueness constraint? Why even bother with uniqueness constraints?

▶ A **uniqueness constraint** does not serve the identification of an object within the program package. It describes instead what combination of attributes must be technically unique.

▶ Program-internally and during data exchange an object is marked by a technical **object identifier**. It is of no importance whatsoever for the application.

Hence we do not need a uniqueness constraint for every object class just to be able to identify the object. It is sufficient if the data object that corresponds to the actual person can be found during data input. For this purpose we can use attributes, relationships etc. without a combination having to be unequivocal.

However if it is a system external identification we are aiming at, which e.g. would be comprehensible to human beings, we need an attribute or a combination of attributes whose values are unequivocal with regard to all objects. Very often artificial attributes are created (insurance number, client numbers, article number, etc.)

⚠ Wherever possible avoid artificial identifiers. If nevertheless they are necessary, make sure they do not contain items of other attributes in redundant form.

For the Mount Ilis Alpine Transports a simple solution has been found for this problem: A number for every employee has been introduced. Anybody new at the Mount Ilis Alpine Transports will be assigned a number that has not been bestowed on before.

```
CLASS Person =
  Name: TEXT*50;
  FirstName: TEXT*20;
  Employee'sNumber: 1 .. 9999;
UNIQUE Employee'sNumber;
END Person;
```

The whole thing would become more tricky if the class that describes these persons were not defined by the Mount Ilis Alpine Transports but by the national association instead. The number of any person within the scope of the association would have to be unequivocal – even if they were gathered decentrally. If there happened to be two numbers (e.g. one at the Mount Ilis Alpine Transports, another at the Blue Mountains Alpine Transports*)*, then the constraint would be violated.

▶ Uniqueness constraints always apply to all objects that correspond to the class for which this constraint has been drafted – even if they only correspond indirectly (in the form of an extension of the class.

One railway company may possess several names. However per language there should only be one single designation, hence the Mount Ilis Alpine Transports may not have a second German name. Then again this constraint only apllies locally, in other words per company.

After all the Blue Mountains Alpine Transports also have a German name. With regard to all companies there definitely is more than one name in the same language. The language of these designations must be unequivocal for one specific railway company.

▶ If an object features sub structures, uniqueness – as opposed to the actual objects – as a rule should not apply «globally» to the elements of all sub-structures. In most cases it only refers «locally» to the sub-structure elements of one single object.

```
STRUCTURE Designation =
  Name: TEXT*100;
  Language: TEXT*2;
END Designation;

STRUCTURE DesignationOfRailway EXTENDS Designation =
  Abbreviation: TEXT*10;
END DesignationOfRailway;

CLASS RailwayCompany =
  Names: BAG {1..*} OF DesignationOfRailway;
UNIQUE
  (LOCAL) Names : Language;
END RailwayCompany;
```

But how can we avoid collisions between the abbreviations of various railway companies? Both the Blue Mountain Alpine Transport as well as the Black Mountain Alpine Transports would in the first place like to be known as BMT. In INTERLIS 2 consistency constraints cannot only be formulated for object classes respectively local structure elements but also for views (cf. paragraph 6.17). By means of a certain view we can make structure elements into basically independent objects. Then in turn we can formulate uniqueness constraints for them.

6.14.4 Existence constraints

As opposed to cog rails and funiculars, the course of the tracks of aerial cable cars, gondolas, ski lifts etc. is not arbitrary but linked to their bottom and top station as well as their pylons.

We wish to express this context. However the lines of INTERLIS 2 connect vertices, which primarily are coordinates, and are devoid of any reference to model objects such as pylons for instance. The connection between the course of the track and other objects may be formulated as a consistency constraint.

Using the following definition every point within the course of the track has to rely either on the position of a pylon (Pylon:Position), the position of the bottom station of an alpine transport (AlpineTransport:PosBottomStation) or (OR) the position of the top station of an alpine transport (AlpineTransport:PosTopStation).

```
CLASS GroundIndependentTransport EXTENDS AlpineTransport =
EXISTENCE CONSTRAINT
  TrackCourse REQUIRED IN
    Pylon:Position
      OR
    AlpineTransport:PosBottomStation
      OR
    AlpineTransport:PosTopStation;
END GroundIndependentTransport;
```

Such existence constraints cannot only be formulated in connection with lines but also with ordinary attributes. In conceptual terms they can always be considered a weak form of a relationship.

6.14.5 Inheritance of consistency constraints

Already with the alpine transport itself a consistency constraint had been formulated: The course of the track has to start at the bottom station and end at the top station. In other words, the first point of the course of a track (Tracks -> Segments[FIRST] -> SegmentEndPoint) must coincide with the position of the bottom station and (AND) the last point of the course of a track (Tracks -> Segments[LAST] -> SegmentEndPoint) must coincide with the position of the top station.

Paragraph 7.3 explains the structure of polylines. It also deals with the attribute SegmentEndPoint, which stands for the end point of a line segment.

```
CLASS AlpineTransport =
  PosBottomStation: Ahland.NationalCoord3;
  PosTopStation: Ahland.NationalCoord3;
  TrackCourse: Ahland.LineNormal;
MANDATORY CONSTRAINT
  Tracks -> Segments[FIRST] -> SegmentEndPoint == PARENT == PosBottomStation
    AND
  Tracks -> Segments[LAST] -> SegmentEndPoint == PARENT == PosTopStation;
END AlpineTransport;
```

What does such a definition mean when it comes to possible extensions of this class?

▶ Class extensions cannot rule out consistency constraints. Extensions can only define additional constraints.

6.15 How close are operating decisions and means of transport? – Independent topics

6.15.1 General remarks

Operating decisions always refer to a certain alpine transport. Consequently these two classes are linked by an association.

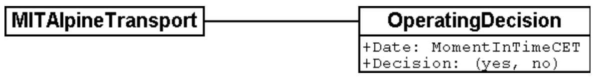


Figure 61: The classes MITAlpineTransports and OperatingDecision are linked by an association.

Nevertheless the objects of these two classes are quite different. It takes quite a lot to build an alpine transport and to alter any of its characteristics. Such modifications (and also those of the tickets) are always decided by the management. Operating decisions are a daily occurrence and are up to the works manager.

Condition reports are even more extreme: In the case of the more important alpine transports they are generated automatically every twenty minutes. For the input and processing of data sometimes different program packages are in use. This matter ought to be stated in some way in the concept.

- **Topics** put the model definition into order with regard to **responsibilities and occurrences in time**.

This offers the possibility that not all data have to be on hand on a certain computer system, or that certain topics are only read, but never altered.

- Several **baskets** may exist in connection with one topic; these contain data related to this topic.

The computer system of the Mount Ilis Alpine Transports for instance comprises one basket each for the alpine transports, the tickets and other different operational aspects. The National Tourist Office also keeps one basket each for alpine transports and tickets. The Mount Ilis Alpine Transports always transmit modifications within the baskets of alpine transport and tickets to the National Tourist Office. The Blue Mountain Alpine Transports and all other railway companies transmit their modifications or periodically a copy of their data baskets. Upon receipt the National Tourist Office integrates the data contained in these baskets in their own.

Thanks to the organization of these models in different topics data can be transmitted specifically. Only the baskets containing topics of relevance to the receiver need to be transmitted.

6.15.2 Independence of topics

If an alpine transport is pulled down, then consequently the data object is deleted. This modification will be made known to the Tourist Office. However if only the basket of alpine transport is transmitted, a contradiction will result within the data of the National Tourist Office. There will be tariff zones that still are connected to the alpine transport, even though it has been deleted. Obviously enough relationships that span topic limits are particularly tricky.

- Topics should not, or as little as possible, depend on one another Relationships that are topic spanning should be avoided whenever possible. If they occur they have to be marked especially within the model.

In graphic representations of models such relationships are relatively easy to spot, provided the representation clearly depicts topics and relationships. In the textual representation of INTERLIS 2 must be marked with the keyword EXTERNAL. Furthermore they are only admissible if the topics have been declared dependent (DEPENDS ON). Mutual dependencies (even if indirect) are inadmissible.

But how can we avoid relationships beyond the bounds of topics without consequently having to limit ourselves to one single topic?

6.15.3 The responsibility of sender and receiver

Of course the relationship between an operating decision and the alpine transport it refers to simply cannot be avoided. Nevertheless it still makes sense to keep alpine transports and operating decisions in different topics. And with this relationship there really should be no problem about things not matching. Both topics and their corresponding baskets are updated by the Mount Ilis Alpine Transports. But mainly with fast moving objects that are referred to via relationships in different topics conflicts cannot always be precluded.

INTERLIS 2 lays down the following regulation:

▶ Correctness of the relationships within one basket is the responsibility of the sender. The receiver has to deal with the fact that objects belonging to a topic-spanning relationship may not be known at a given moment. Then again the receiver may proceed on the assumption that even topic spanning relationships are correct if there are matching versions of the corresponding baskets.

The first rule whereby a basket internally has to be correct must be obeyed even if for some reason a basket is divided.

6.16 All that is good comes from above – Using existing models

The people from Ilis Valley are smart: Instead of reinventing everything, they are using existing models (Units, Ahland, Addresses, NatTour). All the same they have committed some sacrileges. For instance an AhlandLine surely is not typical for Ilis Valley and its alpine transports, but should rather belong to the customary data model of Ahland. The same goes for WGS84Coord and the TimeOfDay, which cannot count as Ilis Valley specialties.

By normal standards those in charge in Ilis Valley have proceeded sensibly in as far as they added things they couldn't find anywhere else to their own model. Understandable, but not optimal.

⚠ Missing or incorrect definitions at a more general (higher) level should not simply be accepted; it would be better to get together with the responsible authorities and to improve these models.

This is why it is also sensible to predefine fundamental types at various levels. INTERLIS itself puts some basic models at your disposal. Other models will be provided by user communities (e.g. associations). There are others that are very specific such as the model of the Mount Ilis Alpine Transports.

Paragraph 3.3 names some sources for standardized data models.

6.17 Tariff zones are of no interest – Views

6.17.1 General remarks

If within the scope of modeling we speak of views of course we do not think of the view from Mount Ilis with its spectacular sight of Twisted Peak. But nevertheless here are some similarities between the two types of views. On the topographical map we find Mount Ilis, Twisted Peak and all the other mountains, valleys and villages, their altitude being illustrated by means of numbers and contour lines. The map does not show the view from Mount Ilis as such. But it does contain all necessary information that will allow a practiced map-reader to derive the view from Mount Ilis. Studying the map, it becomes clear that the peak that can be seen to the left of Twisted Peak, must be the «Black Tooth».

In analogy object classes, structures and relationships of a model correspond to a map. They are appropriate replicas of reality without predefining a specific purpose. The views of a model correspond to the view from Mount Ilis. They serve a certain purpose. To that end they refer to fundamentals or other views and convert them in such a way that the purpose may be served as well as possible.

But then why should such views be part of the model? We do not want to anticipate in the model whether the view should be enjoyed so to speak from Mount Ills, Twisted Peak, the Black Tooth or from the spa in the garden of the kurhaus.

Above all for special consistency constraints (cf. paragraph 6.14.3) and or derivable relationships (cf. paragraph 6.13.7) views will make sense even within the scope of the model. But views are also helpful when processed data for a specific purpose have to be supplied, for instance in the case of the data transfer to the Ilis Valley web service. Moreover INTERLIS 2 offers you the possibility to define graphics. In many cases such graphic definitions will not be based upon original data but upon views.

▶ **Views** are built upon object classes or other views and combine in different ways primary objects into new view objects.

The views of INTERLIS can be compared with the views of data base systems.

6.17.2 The formation law of views

Every detail of the course of a track may not be interesting, but its entire length definitely is. With persons we may sometimes be more interested in their age than in their year of birth. These characteristics can be derived from others. If such «redundant» characteristics were collected as normal data they would be more than liable to be out-dated. After all a person's age changes every year!

▶ The most essential characteristic of a view is its **formation law**. It determines how derived view objects will be created from primary objects.

For instance the view «PersonWithIndicationOfAge» is derived from the object class «Person» the so called basis. One individual PersonWithIndicationOfAge will possess the same characteristics (ALL OF) with exactly the same values as the original person. In addition the view supplements one more characteristic «Age». The age results from (:=) the difference between year of birth and current year.

```
VIEW PersonWithIndicationOfAge
  PROJECTION OF Person;
=
  ALL OF Person;
  Age: 0 .. 150 [y] := Difference (Person -> YearOfBirth,
                         PARAMETER CurrentYear);
END PersonWithIndicationOfAge;
```

In this example there is exactly one virtual view object for each object, in other words a corresponding PersonWithIndicationOfAge for every person.

▶ The most simple formation law of a view is the **projection (PROJECTION)**. It is built upon the basis, accepts individual (or even all) attributes in any order and can add other, derived attributes. Hence its main purpose is to put attributes of already existing objects into a user-friendly form.

The National Tourist Office has defined an abstract class «TariffZone». In Ilis Valley however they do not want to list individually which alpine transport belongs to which tariff zone. Instead they have limited tariff zones that are described with the class «TariffZoneInRegion». This class is a characteristic «Zone» for the geographically limited tariff zone of validity.

A means of transport whose bottom and top station lies within the area of such a spatial zone automatically accepts its tickets.

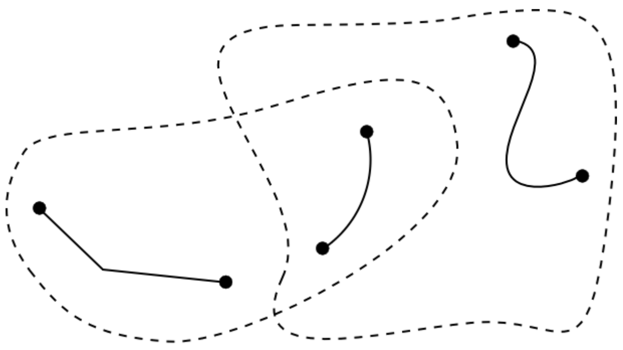


Figure 62: Which alpine transport is situated in the area of which tariff zone? People in Ilis Valley are interested in any pair of alpine transport At and ZoneInRegion Z which comply with two conditions: The bottom station of At must lie within the area Z, and the top station of AT must also lie within the area of Z.

But now which alpine transport actually does lie within the area of which tariff zone? By taking recourse in a view this connection between alpine transport and tariff zone can also be derived.

▶ Possibly the most important formation law of a view is the **join (JOIN)**. It combines several basic objects into a view object. Especially as a basis for derived relationships the join is of major importance.

```
VIEW AlpineTransportsInRegion
  JOIN OF At ~ AlpineTransport,
          Z ~ TariffZoneInRegion;
  WHERE InSurface(At -> PosBottomStation, Z -> Region) AND
        InSurface(At -> PosTopStation, Z -> Region);
=
END AlpineTransportsInRegion;
```

To start with by introducing a join all possible pairs are formed. Each object of the class AlpineTransport is joined to every object of the class TariffZoneInRegion to form a virtual view object.

By introducing the WHERE-part the set of all view objects is reduced to those that comply with both conditions. Hence we are left with those pairs of alpine transport At and tariff zone Z where bottom and top station of At lie within the area of Z. In the figure above with six possible pairs (three alpine transports x two tariff zones) four pairs comply with this condition.

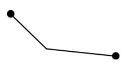
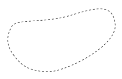
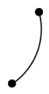
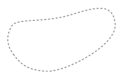
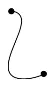
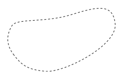
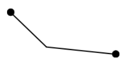
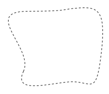
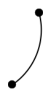
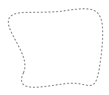
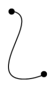
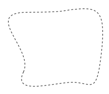
Alpine transport *At*	Tariff zone *Z*	Bottom and top station of *At* in the area of Z?
		yes
		yes
		no
		no
		yes
		yes

Figure 63: Looking at all combinations of alpine transport *At* and tariff zone *Z* in the last figure, we realize that with only four pairs out of six bottom and top station of *At* lie in the area of *Z.*

In a last step we decide in a projection which characteristics the view objects should possess and how their values are determined. In the INTERLIS-definition above the part after the equation mark is used to that purpose.

If there is no tariff zone that corresponds to one particular alpine transport, then it will not appear in that view. By introducing a special join (a so-called «**Outer Join**») we require that a view object should exist even if there is no corresponding tariff zone for one alpine transport. Then again with regard to the concrete application of alpine transports and tariff zones this will hardly make sense.

If we should wish to have a register of all coordinates of bottom and top station we are confronted with the fact that these coordinates are captured as individual attributes of the alpine transports. By using a **union (UNION)** they can be gathered into a set of equal view objects.

```
VIEW StationCoordinates
  UNION OF BottomStation ~ AlpineTransport, TopStation ~ AlpineTransport; =
  Coordinates: Ahland.NationalCoord := BottomStation -> PosBottomStation,
                                       TopStation -> PosTopStation;
END StationCoordinates;
```

Here the set of all view objects equals the double set of all alpine transports. Once they are evaluated under the aspect of bottom station, and once under the aspect of top station. The attribute is selected according to the position attribute of either bottom or top station.

Aggregation (AGGREGATION) and **inspection (INSPECTION)** deal with structure attributes. An aggregation unites objects that have the same characteristics into one single object. Within the scope of the view object already existing objects are available as elements of a structure attribute (cf. paragraph 6.17.3). On the other hand an inspection makes sure that structure elements become independent view objects (cf. paragraph 6.14.3).

6.17.3 Building views step by step

In order to check tickets, every alpine transport has to know what ticket type is valid. So they would still like a list of all alpine transports that indicates for every line which ticket types are valid. Independently of all basic data they would like to define something like the following model:

```
CLASS TicketType =
  Names: BAG {1..*} OF Designation;
  Price: 0.00 .. 5000.00 [Ahland.Sovereign];
  Validity: LengthOfTime;
END TicketType;

CLASS AlpineTransport =
  Names: BAG {1..*} OF Designation;
  ValidTicketTypes: BAG OF TicketType;
END AlpineTransport;
```

But how can this be derived from the original data? This is not quite as simple. Several tariff zones can be assigned to one alpine transport; then again several ticket types are assigned to one tariff zone. Furthermore there are tariff zones, which comprise all alpine transports within one area.

Luckily this last aspect has already been dealt with because there is an abstract relationship between alpine transport and tariff zone, «Validity». On the one hand it is realized by means of an explicit relationship between the two classes («ValidityExplicit»). On the other hand we can derive from the view «AlpineTransportsInRegion» which lines on grounds of their position accept the tickets of one tariff zone.

On this basis we can define a view which links alpine transports and ticket types:

```
VIEW AlpineTransportAndValidTicketType
  JOIN OF At ~ AlpineTransport,
          Z  ~ TariffZone,
          Tt ~ TicketType,
          V  ~ Validity;
  WHERE (V -> AlpineTransport == At) AND (V -> TariffZone == Z) AND
        (Tt -> TariffZone == Z);
=
  TransportNames: BAG {1..*} OF Designation := At -> Names;
  TicketNames: BAG {1..*} OF Designation := Tt -> Names;
  Price: 0.00 .. 5000.00 [Ahland.Sovereign] := Tt -> Price;
  DurationOfValidity: LengthOfTime := Tt -> DurationOfValidity;
END AlpineTransportAndValidTicketType;
```

This combines alpine transport and ticket type. It takes into consideration the validity relationship and the fact that a tariff zone is assigned to every ticket type, which has to be in keeping with the validity relationship. So we have almost achieved our goal. The admissible combination of alpine transport and ticket type are available as view objects. Now we would like to unit them per alpine transport:

```
VIEW OnAlpineTransportValidTicketType
  AGGREGATION OF AtVT ~ AlpineTransportAndValidTicketType
                EQUAL (AaVT -> At);
=
  TransportNames: BAG {1..*} OF Designation := AtVT -> At -> Names;
  TicketTypes: BAG OF AlpineTransportAndValidTicketType := AGGREGATES;
END OnAlpineTransportValidTicketType;
```

This result is achieved by means of an aggregation. Thereby all objects of the basic view which comply with a certain condition (i.e. that they belong to the same alpine transport) are combined into one view object. The set of all primary view objects that has been combined to form a whole is available for structure attributes (AGGREGATES).

6.17.4 Inheriting views

The national association has already defined the view that lists all valid ticket types for every alpine transport (view «OnAlpineTransportValidTicketType», see above). In Ilis Valley they also want to use this view. But they also want to include the attribute TrackCourse in this view, which they have defined in their own extension of the class AlpineTransport.

```
VIEW MITAlpineTransportAndValidTicketType
EXTENDS AlpineTransportAndValidTicketType
  BASE At EXTENDED BY MITAt ~ MITAlpineTransport
=
  TrackCourse := MITAt -> TrackCourse;
END MITAlpineTransportAndValidTicketType;
```

With the definition of an additional basis (must be an extension of an already existing basis) its attributes are available. If a view object is not based on this extension (i.e. it is not a MITAlpineTransport), the attribute is undefined.

▶ An extension of a view allows the user to acknowledge extensions of the classes of the basic view and to make use of their attributes. However we cannot alter the formation law of the view in any major way. It is merely possible to define additional selections.

6.18 What's in a name? – Schemata in a foreign language

An alpine transport is no different in French than it is in German. It still has the same characteristics; it entertains relationships with the same classes etc. We might argue whether terms in different languages express the same meaning, but where data models are concerned it holds: The actual concepts are the same in whatever language. The only thing that changes from one language to the next is names.

Whoever wants to translate a model into a foreign language only needs to exchange designations and remarks. The structure itself – in UML-diagram the little boxes and lines – remains the same. For INTERLIS-descriptions there is a tool (the so-called INTERLIS-compiler), that checks whether a translation in actual fact only uses different names or if during the translation accidentally the structure of the model also has been changed.

7. Ilis Valley systems in focus

7.1 What are standard conform systems? – System neutrality

Various program packages, e.g. NatTourSys used by the national association, but also Lift-Sys employed by the Mount Ilis Alpine Transports, have implemented the models defined by INTERLIS 2. They are capable of establishing and reading corresponding files. Consequently are they INTERLIS-systems? To put the question more generally: When does a system comply with a certain standard?

A description at conceptual level does not aim at dictating the way a computer system implements the idea portrayed. As a rule systems have their own possibilities to realize concrete applications. Ideally the internal data model can be derived directly from the descriptions but this is no prerequisite. It is quite conceivable that a system cannot directly process data. In order to be compatible it is good enough for a system to be able to process data directly or indirectly according to a (e.g. by means of supplementary conversion programs. Whenever possible a norm for data modeling should not dictate the concrete manner in which the systems convert the application. Creativity is what we are after, there should be room for free enterprise. There must be system independence of the standard; otherwise desirable system changes may fail because of the incompatibility of data.

Then again a system need not possess the entire range of capacities that are conceivable in connection with a description. For instance WebSys can do nothing but collect INTERLIS 2-data. More extreme still are those systems that send condition reports from the individual alpine transports to the central office. In a very rudimentary way report data are inserted into prepared files that are structured according to the rules of the standard.

If an application makes very special requirements, we cannot expect them to be covered by a generally applicable norm. In such cases support is a matter of contract with the system manufacturers. However exercising restraint is recommended, because at the same time you will lose part of your system-independence and limit the circle of possible system providers.

Named below two examples depict how system dependent aspects can be standardized without in a major way influencing the systems internally.

7.2 The exchange rate of the Euro varies daily – Parameters and functions

For each ticket type its price is known in the national currency. In the interest of foreign tourists it should also be published in Euro and US-dollars. Then again it is unpractical to alter the prices for all ticket types following every fluctuation in the corresponding exchange rates. They should rather be computed from the national currency.

```
CLASS TicketType=
  Name: Text * 100;
  Price_SOVEREIGN: 0.00 .. 5000.00 [Ahland.Sovereign];
  Price_EUR: 0 .. 4000 [EUR] := SOVEREIGNtoEUR (Price_SOVEREIGN);
  Price_USD: 0 .. 4000 [USD] := SOVEREIGNtoUSD (Price_SOVEREIGN);
END TicketType;
```

Whenever data are presented in the system or are prepared as data for the WebSys, the prices in Euro and dollar are newly calculated from the prices in the national currency. The conversion results from the two functions SOVEREIGNtoEUR respectively SOVEREIGN-toUSD. But how should LiftSys in particular, or any other system know what the essence of this function should be?

From the conceptual point of view the most important information consists of the fact that the two prices in foreign currency are treated as independent data but will be derived from the price in sovereigns. It is enough to determine names and their parameters for a certain function.

```
FUNCTION SOVEREIGNtoEUR (Sovereign: NUMERIC [Ahland.Sovereign]): NUMERIC [EUR]
                        // Conversion in Euro //;
```

The actual performance of the function is only indicated as an explanation between //. Its implementation is still up to the systems. Such definitions should be limited to some few basic models because they have to be prearranged with the systems manufacturers. In INTERLIS 2 the existence of such an agreement is remarked with a contract. Functions can only be defined in models with such a contract.

```
CONTRACTED MODEL IlisTour AT http://www.interlis.ch/models/ahland
  VERSION "2008-01" =
  ...
END IlisTour;
```

If we aim at keeping the number of functions as small as possible, they have to be formulated in as general a manner as possible. For instance instead of using SOVEREIGNToEUR, a universally applicable function for the division could be defined.

```
FUNCTION Division (Dividend: NUMERIC, Divisor: NUMERIC): NUMERIC;
```

It must be admitted that thus we renounce certain possibilities of control because there is no certainty that the first indication is a sum in the national currency. Using the form stated below this could be guaranteed.

```
FUNCTION ToCurrency (Sovereign: NUMERIC [Ahland.Sovereign],
                     ExchangeRate: NUMERIC [Ahland.Sovereign]):
                    NUMERIC [MONEY];
```

But where does the exchange rate come from? For such cases we might intend certain values, e.g. the exchange rate for Euro or US–dollars, to be available as parameters within the systems. As this again has some impact on the systems, the definition of system parameters is only permitted within the scope of contracts.

```
PARAMETER
  EURExchangeRate: 0.000 .. 5.000 [Ahland.Sovereign]; !! Price of one euro
                                                      !! in sovereigns
  USDExchangeRate: 0.000 .. 5.000 [Ahland.Sovereign];

CLASS TicketType =
  Name: Text * 100;
  Price_SOVEREIGN: 0.00 .. 5000.00 [Ahland.Sovereign];
  Price_EUR: 0 .. 4000 [EUR] := ToCurrency
                              (Price_SOVEREIGN, PARAMETER EURExchangeRate);
  Price_USD: 0 .. 4000 [USD] := ToCurrency
                              (Price_SOVEREIGN, PARAMETER USDExchangeRate);
END TicketType;
```

7.3 On crooked ways – Line forms

Maybe somebody would come up with the idea to describe the course of the ski run other than by means of straights and arcs. Maybe he would rather use clothoides, splines or Bézier-curves instead. INTERLIS 2 does not supply these forms directly but it permits the definition of new forms for line segments.

A line consists of an ordered set of line segments. They are considered a concrete extension of the abstract structure *LineSegment*. If you choose to use additional forms besides the predefined types of line segments (straights and arcs), then you can extend *LineSegment* with a suitable structure.

Again such a definition is subject to an agreement with the manufacturers. After all the systems will have to come to terms with these line forms. Above all it is desirable that these forms will be represented correctly on the screen and on paper.

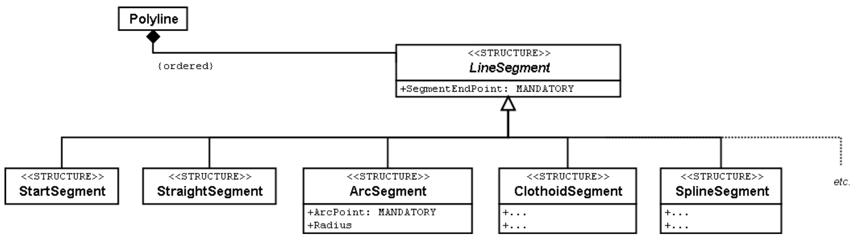


Figure 64: INTERLIS-lines are composed of individual segments. Straights and arcs are predefined. The abstract structure for line segments can be extended by additional forms.

The end point of each segment is at the same time the start point for the next. That is why the start point is not part of the line segment. A special start segment defines where the first segment begins.

An end point does not sufficiently define an arc. That is why arcs not only have an end point but also an auxiliary point that is also on the line. It should be approximately in the middle between start and end point, because this will render calculations more precise.

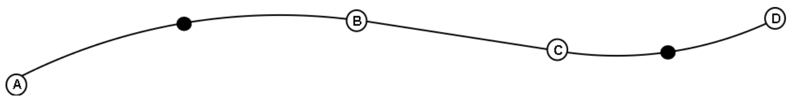


Figure 65: This line consists of four segments: A start segment with end point A, an arc segment with end point B, a straight segment with end point C, and a second arc segment with end point D. the auxiliary points of the arcs are on the line and displayed in black.

Of course the radius of an arc can always be computed from the coordinates of the vertices. However mathematical inaccuracies may lead to calculated values that diverge from the one intended. If the radius is of conceptual importance for the application, this is unacceptable. Hence arc segments optionally can feature a value for the radius.

If the radius is indicated, the exact position of the line is defined with this value. In this case the auxiliary point would only serve to select one of the four possible connecting lines.

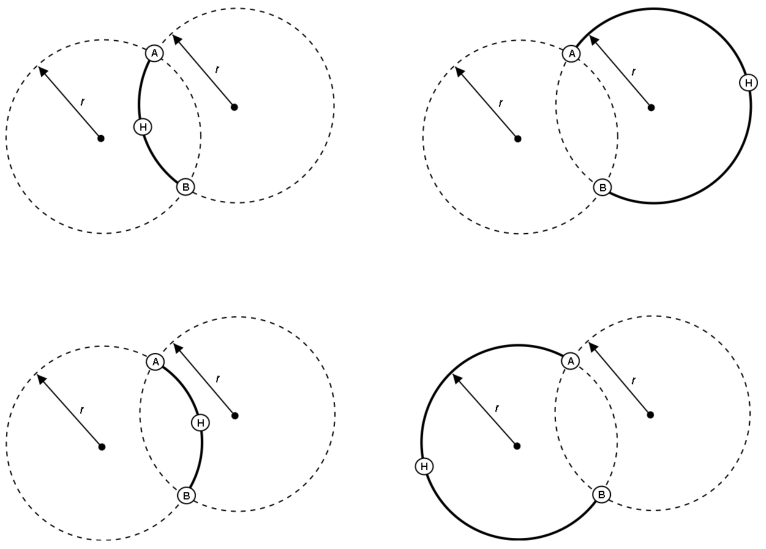


Figure 66: If the radius *r* is indicated, the auxiliary point H only serves to select one of the four possible arcs, which link points A and B.

8. Ilis Valley data on their way

8.1 Out of mind, out of sight – Full transfer

Let's recall: In Ilis Valley they had carefully considered which data was needed for their projected application and had documented their ideas with a data model. In the course of their discussions it had become apparent that working with both graphic representation and detailed textual description was helpful.

After many a lengthy discussion those in charge in Ilis Valley had gotten there: The standard program package «LiftSys» was acquired and arranged according to the Ilis Valley data model. Even though things did not go quite as smooothly as promised by the LiftSys-sales representative, these initial difficulties were not hard to overcome. And finally the data of all transport lines, railway companies, ski runs etc. in Ilis Valley had been collected.

Proudly the IT-expert demonstrated how with one single mouse click the entire Ilis Valley data could be exported into an INTERLIS 2-file. Some spectators were not easily enthused and critical questions were asked: All these meetings for one miserable file? Had that really been necessary?

All objection was quickly squashed once the accountant explained just how many sovereigns the data exchange had cost them per year up until now. Every time data had had to be transferred to the national association there had been trouble. How many hours of work had been time wasted!

From now on it would be possible to send one file with all the Ilis Valley data to the national association and they would be able to process them without any problems.

▶ The simplest type of transfer is the **full transfer**, whereby all data are transferred as a whole.

The fact that a transfer goes smoothly even if not all the parties concerned use the same data model is due to inheritance as explained in paragraph 5.5.

8.2 Brackets as pointed as Mount Ilis – Transfer rules based on XML

What was actually in this file? It was opened with a normal text editor.

```
<?xml version="1.0" encoding="utf-8"?>
...
<BASKET BID="xAHTOUMIT01234567" TOPICS="IlisTour.AlpineTransports">
  <IlisTour.MITAlpineTransports.MITAlpineTransport
   TID="xAHTOUMIT04231336">
    <Names>
      <NatTour.Designation>
        <Name>Pony lift Ilis Ville</Name>
        <Language>en</Language>
      </NatTour.Designation>
    </Names>
    ...
    <PosTopStation>
      <P>
        <C1>8020.60</C1>
        <C2>13188.62</C2>
        <C3>1789.04</C3>
      </P>
    </PosTopStation>
    <TravelTime>
      <Ahland.LengthOfTimeInMinutes>
        <Duration>3</Duration>
      </Ahland.LengthOfTimeInMinutes>
    </TravelTime>
    <Kind>SkiLift</Kind>
    ...
    <PictureTopStation>
      http://www.ilishornbahnen.com/webcam?bahn=pony4
    </PictureTopStation>
    ...
  </IlisTour.MITAlpineTransports.MITAlpineTransport>
</BASKET>
...
```

Figure 67: The Ilis Valley pony lift as part of the full transfer for Ilis Valley data. The INTERLIS-standard defines the precision with which such a transfer file has to be structured for a certain data model.

Quite a few pointed brackets, at the very beginning something about XML (that had been mentioned somewhere earlier along the way!), even though with question marks... Aha, something simple for a change: The pony lift Ilis Ville (more towards the beginning, printed in bold).

Depending on the data model a transfer file has a different structure. But of course it is not completely arbitrary: The INTERLIS-standard states precise rules as to how the corresponding XML format is to be derived from a specific data model.

Slightly complicated, this whole thing – but luckily LiftSys could be used and all these brackets never had to be entered by hand. At least even without a special program it was evident that the travel time on the pony lift is 3 minutes.

The IT-expert explained that even in forty years' time they would still be able to work with these data. Even supposing that by then the manufacturers of the original programs had gone bankrupt, fifteen new program versions had had to be installed and all the hardware had been exchanged. This statement appeared to be quite plausible because even without any documentation and on paper it was possible (though rather awkward) to understand these data.

But what about this funny text «TID="xAHTOUMIT04231336"»? It designates the pony lift within the transfer file. The value is only of importance for the transfer: This object identifier appears wherever in the transfer file there is a reference to the pony lift. During installation of

LiftSys it had been possible to indicate that all identifiers should begin with xAHTOUMIT, exactly as had been demanded by the National Tourist Office in their letter.

Object identifiers begin with a country code in accordance with ISO 3166 (ch for Switzerland, fr for France, etc.) and six other symbols uniquely assigned by the appropriate, central authority. The eight remaining symbols are automatically assigned by the computer system.

For explanations about object identifiers see appendix E of the INTERLIS-reference manual.

8.3 Once and over and over again – Incremental update

As much as seventeen minutes on the pony lift? Those poor kids will be frozen stiff before they will even have attempted their first stem turn! This line will hardly survive its first season, how could anybody still be interested in its data forty years from now?

Finally it turned out that the pony lift was not tremendously oversized – somebody had simply made a typing-mistake while entering data: Instead of seventeen minutes the ride would only take all of one minute.

Of course it did not take long to change that figure in LiftSys. But did this mean all the data had to be sent once more to the national association?

▶ Thanks to **incremental update** it is no longer necessary to send all data after a modification, it is enough to transfer the modifications themselves. This saves transferring data but the real advantage lies in the fact that it provides at the same time a documentation of the exact nature of modifications.

By means of the LiftSys-program an update for the national association had been generated. In contrast with the full transfer this update did not comprise all data of the entire Ilis Valley but only data of modified objects. Consequently it turned out to be much smaller and easier to handle.

Yet it does not always make sense to do an incremental update. For instance the National Tourist Office does not offer itself the internet-service where tourists can find information about the prices of tickets at the various lines, instead they have delegated this task to one of their sister companies. Periodically they are sent a complete copy of all data in INTERLIS 2-format. These data are loaded by a server-program, which answers any concrete question. An incremental update would increase requirements on the server-program without any visible profit.

8.4 The Blue Mountains also are touristy – Baskets, replicas and polymorph reading

Ilis Valley is not the only resort that sends data to the National Tourist Office; there are 162 other areas that do the same. They do not want to be bothered with looking after every areas data. They are happy to simply receive regional updates with current data from time to time.

In the individual regions data are kept in the databases of the systems employed. Within the scope of INTERLIS we proceed on the assumption that the data of every topic of the data model are stored in one (or maybe several). Hence the alpine transport data of the Mount Ilis Alpine Transports are in one basket, those of the Blue Mountains Alpine Transports in another. Now if these data are sent from the Mount Ilis Alpine Transports or the Blue Mountains

Alpine Transports to the National Tourist Office, then the corresponding basket becomes visible in the transfer file. The computer system of the national association (NatTourSys) reads these data and updates the data-base NatTourDB. Simultaneously they record where the objects come from.

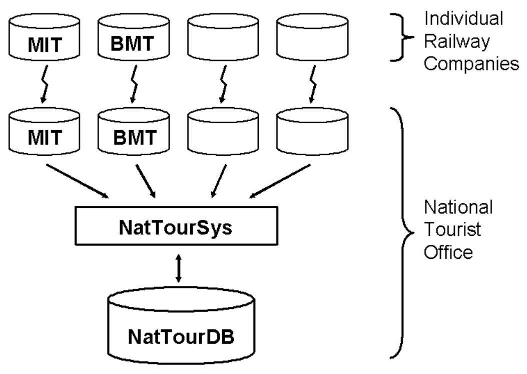


Figure 68: Occasionally the National Tourist Office receives an update of their tourism data from the Mount Ilis, the «Blue Mountain Alpine Transports» and many other railway companies.

Hence the data concerning the Ilis Valley pony lift exist twice: Once with the Mount Ilis Alpine Transports, once with the National Tourist Office. Of course this does not mean that from now on the children in Ilis Valley have got an extra ski run. We have only copied data and not built a new ski lift!

Even from the electronical point of view everything is clear; these two data objects possess the same object identifier. This makes it obvious that we are dealing with replicas that stand for one single really existing pony lift.

Other terms meaning replica are: substitute, duplicates, proxy-objects.

It is important that an object identifier (such as «xAHTOUMIT04231336» in the example above) definitely is unequivocal. Otherwise it might accidentally happen that the Mount Ilis and the Blue Mountains Alpine Transports use the same identifier for two different objects. Consequently it no longer would be clear for the National Tourist Office whether when receiving an incremental update they are dealing with a modified object from Ilis Valley or from the Blue Mountains.

An administrative authority of Ahland («AH») has assigned the index code «AHTOU» to the National Tourist Office. Subsequently the National Tourist Office determined the first part to be used in an identifier for every railway company (e.g. «AHTOUMIT» for the Mount Ilis Alpine Transports and «AHTOUBBB» for the Blue Mountain Alpine Transports). For the remaining part of the identifier the company itself respectively the program employed is responsible.

With a full transfer object identifier do not have the same significance as with an incremental update. They need not be maintained; they merely serve the re-establishing of relationships between different objects (e.g. tariff zones and ticket types).

8.5 The pony lift in the «Tal der gelben Murmeltiere» – Foreign languages in data transfer

Just behind the Black Tooth lies the «Tal der gelben Murmeltiere». Disregarding the fact that German is spoken over there and that the indigenous marmot have an intensively colored coat, it is hardly any different from Ilis Valley.

Above all their local pony lift is also an all-time favorite with the kids. But how will the national tourism association find out about its travel time? After all the designations used in the data model will reappear in the structure of the transfer files. That is how come the Ilis Valley data feature lines such as <Duration>3</Duration>. If the data model is translated into another language, then of course the corresponding transfer format changes as well.

So how does the National Tourist Office handles the fact that for instance the transfer file from one valley contains the line <Duration>3</Duration>, but the one from the neighboring valley says <Dauer>3</Dauer>?

The National Tourist Office does not have to buy separate software for everyone of the native languages. INTERLIS makes sure that in spite of multilingual applications a smooth transfer is guaranteed, on the sole condition that due to translation the data model has not undergone any changes in its structure. As already mentioned in paragraph 6.18 a tool (the so-called INTERLIS-Compiler) is available that checks the translation of a data model with regard to its structural conformity with the original.

9. Data modeling beyond Ilis Valley

While in the preceding chapters we have examined many details of data modeling, this chapter takes up some of the more fundamental aspects. So its lecture might even be of interest to those who do not want to deal with full particulars.

9.1 There are many routes up to Mount Ilis – The correct model does not exist

In Ilis Valley they are quite aware of the fact that their model is not above all criticism. Since in reality Ilis Valley does not exist we have not bothered to question the model again and again from different angels. It was our primary intention to introduce the different aspects of modeling and their corresponding tools.

Nevertheless with every model the question arises: Is this model correct? Or actually the question might rather be: Is this model good enough?

It is clearly short sighted to consider a model correct or good if it formally meets all the requirements of the description language. Just because a model is accepted by the INTERLIS-compiler does not mean that it is in actual fact good.

Going by the proverb «Many roads lead to Rome» – or in our case to Mount Ilis – is more to the point. Evidently there are many possibilities of modeling one specific circumstance. But not every possible model is as satisfactory as another. After all there are also wide, narrow, steep, passable, rough and scenic paths up to Mount Ilis. Today the best way may be different from the one we took yesterday. Needs, opinions and criticisms can vary.

In the following paragraphs we look at different verdict criteria for the modeling of data.

9.2 Mount Ilis seen from different angles – Original data and views

If we only go by our immediate needs when modeling, we run the risk of collecting data in an unsuitable way. For instance it is not suitable to use the age of a person as an attribute since it will change continuously. It is preferable to define the date of birth as an attribute and then to derive the age by means of a view from the date of birth and the current date (cf. paragraph 6.17.2).

If the structure of a data model corresponds to the actual facts rather than to a certain point of view, we are more flexible with regard to concrete uses that can be derived from the original data. These data will contain Mount Ilis rather than the view from Ilis Ville or Ilis Bath. These and various other perspectives can be derived as views from our data.

Sometimes the derivation of desirable views from «ideally» modeled data can be quite demanding. Above all views containing, die formation laws with geometrical calculations are not always easy to realize. The decision when to swerve from virtue and to adjust a data model to a view is not always an easy one and in general depends on various criteria. Even if you decide on a model that does not describe the actual data but a view on them, it is worthwhile considering which would be the «nice» model. It will be easier to return to this model at a given time and thus to overcome the disadvantages of a view-oriented model.

The biggest disadvantage of a view-oriented model will become apparent in the collecting and above all in the up dating of data. For instance if for every house entrance we state the street name, the house number, the zip-code and town as a direct attribute, there is a certain risk that mistakes occur e.g. due to different spelling. Furthermore this in a major way increases the costs of collecting data. Worse still when it gets to incremental update, for instance if the new settlement Ilis Mill will be introduced as a village with its own zip code and the lower part of Ilis Ville will be classed with this new village. Do all house entrances have to be muted It would be much better if this new village with its validity domain could be introduced and then the validity domain of Ilis Ville was adjusted accordingly.

Thanks to good models one modified fact in real life will lead to exactly one modification in our data. All the rest can be derived by means of views.

9.3 Time is ticking by at Mount Ilis – Life cycles

Good, in other words adequate data models that are independent of any immediate use become more important the more durable and of general interest are the facts described by these data. Data that are intended for private purposes or for a project of clearly defined duration processed with a specific system, the data model is less important than with long-lived data.

However as a rule data that describe real life are long-lived. The Mount Ilis itself only changes within geological periods. But even houses, railway companies and legal position have a life span of several years or decades. That is why real life objects will out-live the computer systems that describe their mappings.

In order for these mappings of real life (i.e. the data objects) to out-live the specific computer systems (hardware and software) we need to have a clear perception of these data objects. We need a data model.

This data model should be as system independent as possible because it is most likely that the next system will have different capacities. Maybe views will be generated automatically. Considering the relatively short life span of software packages, we should not be too concerned when structuring data with the present functionality of one specific software package.

9.4 Not seeing the wood for the trees – Degree of specification

Beyond question we will not model real life as if the individual atoms that form real-life items were our data objects. That would be far too awkward. Nor should a building be composed of its individual bricks and tiles. But when do we speak of individual trees, and when of a forest without listing each tree as a data object?

Of course this depends on what we want to achieve with the system. What is the use in finding the best possible model if it cannot be realized? Is the model comprehensible? Is it possible to define a dialog for both data-collection and update, which the people intended for the job, can handle?

Especially in this respect a simple model whose object classes are not overly tangled up, may have a positive impact.

The more complicated a model, the more important an intuitively understandable user surface becomes. In such situations it is dangerous to extend a basic model in such a way that not only attributes are added but new relationships are defined as well. There is a considerable risk that the user dialog designed for the basic model loses its elegance and comprehensibility.

Once again we should keep in mind: «As simple as possible, as complex as necessary». But as we all know the adequate realization of this maxim is not quite as easy.

9.5 The Kurorchestra of Ilis Bath strikes up – Data modeling is tricky

When drums and trumpets, violins and flutes all combine to offer us a musical treat, we tend to forget all the effort that has been put into this performance. Just imagine all these years of hard work, dating back to the first attempts on a recorder, until finally they reached the stage of virtuosity. And even now daily practice is the order, alone or with the orchestra.

It is to be hoped that as much serious work goes into data modeling. Whoever wants to understand models must at least have a passive knowledge of the different instruments. Whoever wants to define models is well advised to keep on practicing until he or she is quite familiar with the effects of the instruments (classes, attribute types, relationships, inheritance etc.). Even if the result seems perfect it is worthwhile thinking it overt, obtaining other people's opinions and reconsidering it from different angles. This is the only way to achieve good models that are sufficiently simple and comprehensible to stand the test of time.

10. Conclusion for authorities and management

Naturally enough the satisfactory outcome of this new solution was duly celebrated in Ilis Valley. Nevertheless the president of the town council insisted on receiving further information concerning technical details. What had promoted their success? Why had they been able to solve the problem using their old systems? Why should they run little risk of having to revise everything if one of the systems involved had to be? So when preparing her short speech she had been reading the report written by the secretary of the department of construction and the technical manager of the Mount Ilis Alpine Transports.

* * * * *

I. Remembering the starting shot

It is still easy to recall the day of the first representation of possible new Ilis Valley Web-Pages. After a few impressive demonstrations participants nearly got into each other's hair. Luckily enough several experts were present. Otherwise all the amateurs had never realized just how controversial were the opinions of specialists, just how badly matched their arguments and counter arguments – or rather their catchwords and counter catchword – actually were. Thanks to the ensuing discussion, which very nearly became a dispute, we have pricked our ears.

II. Items, mappings and the ravages of time

To call in one more expert had definitely been a good move, even if the first talks had turned out to be quite surprising. Few technical catchwords were mentioned and mostly swept aside; instead a thorough analysis of the entire matter was conducted.

First conclusion: Facts are not what is in a computer or even what it spits out in the way of pictures, tables or other presentations, facts are what exists and happens in real life.

What is in the computer can be compared to a wooden model or a sandpit. The pictures we might take of them from different angles and with different lighting correspond to the screen or to prints. And one thing that is also obvious with wooden models or a sandpit: to copy real life for the first time is a giant task. Just what it takes to build a simplified version of all the houses, streets, railway companies that exist in reality! Even more complicated than the first input is the update of it all. Hardly completed, and already there are first alterations to be made in order to keep the representation up to date. And time is not kind to sand or plaster structures. Computers are comparatively easy to handle! Of course we have to make updates on changes of the reality, but at least the remaining information does not become antiquated.

Really? No, definitely not as fast as in a sandpit. But after a few years we would like to buy a new, more powerful computer, and a little later still the software employed will no longer be the best on the market. Assuming one would like to use new programs, how would one transfer the electronic replica of real life from the old onto the new system? How would we shift the sand from the old pit to the new without it running through our fingers and all our work having to be recommenced?

III. System independent description of the data structure

In order to transfer electronic mappings of real life – in other words data – from one system to another, we have to have a definite notion of what these data look like, how they are structured. The hard nut to be cracked being: These structure descriptions also have to be independent of any concrete system. They have to be conceived in such a way that they can be equally interpreted and understood by application experts as well as technical specialists of all authorities concerned. Ideally this description would even have a structure that not only human beings but also computer programs can deal with. What we need is a standardized Esperanto of data description.

In everyday speech Esperanto has remained without the slightest significance. Something livelier, possible also more convenient prevailed. Matters are slightly different where technology is concerned. Who does not aspire to system independence will be ruled by systems. And systems are relatively short-lived. Hence the realization that in spite of many de facto-standards *Unified Modeling Language* (UML), a graphic language within the scope of data modeling has spread worldwide.

In intensive discussions a model of real life has been sketched. To start with we had to make clear what the relevant items were, how they had to be named in a comprehensible way, how they inter-acted and what other characteristics they possessed. The protocol of these discussions always recorded new findings in the UML-graphic language and added comments and explanations. Since it became apparent that the pictures were loaded with too many details and in some cases did not have the required precision and additional description-tool was used in the form of INTERLIS: Similarly to programming language it permits the precise description of data structure.

An important characteristic supported by both UML and INTERLIS 2 is the possibility to also use data descriptions made at other locations. Thus it had been possible to profit from definitions of cadastral surveying for instance in the case of building addresses. Data descriptions of the National Tourist Office were not only used but complemented according to local needs, too.

IV. System independent data transfer

The case of data as such is quite similar to that of the description of data structure. Consequently we do not only need standards for the description of data but also for their transfer. When transferring data from one system to another we kind of use Esperanto, within the individual systems the respective mother tongue is spoken.

Having applied such intensive care to the data structure, one no longer wants to be bothered with these details. So INTERLIS serves a double purpose: Along with the description of data structure the transfer of the corresponding data has been defined as well.

It is of utmost importance that this data description cannot only be read and understood by human beings. Because of its formal language definition computer programs can also read it. Based on data description and transfer rules it is evident how a subsequent data transfer will have to be structured.

V. Different effects on the systems concerned

Within the scope of the global solution different computer systems were in use. Above all it was possible to continue using systems already installed. UML and INTERLIS do not prescribe the way systems have to be structured. They are only concerned with the final delivery of data, which has to be in accordance with their conceptual conception.

The system of the department of construction was immediately capable of generating data that had to be transferred and of reading data received. With the system of the Mount Ilis Alpine Transports a converter was employed, in other words a program that converted system specific data into a neutral transfer format. With the small collecting systems at the bottom stations it was only a matter of introducing the simplest of programs, which ensured that based upon the measurements a correct data set was sent as an incremental update to the central office.

It is also interesting to have a look at the system of the National Tourist Office. It was possible to send them files containing supplements defined by us. Thanks to the principle of polymorph reading this was of no consequence.

One reason that rendered this practicable may have been the fact that INTERLIS 2 is based upon the international standard of the *Extensible Markup Language* (XML).

VI. Outlook

Looking back on the experiences gained with all participating systems we are confident that it will be possible to also integrate new wishes and new systems without jeopardizing our entire solution. This will allow us to achieve an important goal: Our solution is no flash in the pan but of enduring value.

* * * * *

The president of the town council mentioned the most important points in her speech and stated quite appropriately that last but not least the technical success was due to their initial detachment from technology. And during the reception that followed she had every right to be pleased with herself: Once again she had had a good nose on the day of that first presentation at the town hall when she had remembered that discussion with her colleague and opted for a very careful procedure.

Register

References to comprehensive explanations in chapter 4 are set in *italics.*