1

GENERALISED TRANSPORTATION-DATA FORMAT (GTF): DATA, MODEL AND MACHINE INTERACTION

Dr. Benedikt Mandel, Eduard Ruffert, MKmetric Gesellschaft für Systemplanung mbH, Karlsruhe, Germany

Abstract

Exchanging data and information between (strategic transport) models and between models and other software, e.g. GIS, is always a very tedious, if even possible, task. There is always the problem of loss of information because the exchanged data only seemingly contains the information required and there is also always the problem of inhomogeneous and proprietary data formats forcing the users of the data to re-format and re-combine the data from scratch every time.
The solution to these problems is to understand that not only data needs to be transferred, but also the precise meaning of the data (meta-data). For the definition of information typically needed by strategic transport models a data model which precisely defines the data and information to be exchanged and for the exchange per se a standard format for electronic data interchange (EDI) is needed. The data model must be an information model which enables a user (typically an applications programmer) to use the building blocks specified in the data model, to define precisely the data and the information contained in the data to transfer. The exchange format should be based on a standard. In addition, an interchange protocol should be defined in order to make the whole process of running a model and retrieving results automatic.
This paper defines such a data model ("Generalised Transportation-data Format" GTF) and proposes an exchange format (GTF-XML) based on standard XML which allows a software application (called a "GTF Translator") to exchange information and data between models and between models and other software and proposes an interchange "language" to run models and retrieve results (TIP).

Keywords: exchange, interchange, format, protocol, EDI, data, information, XML, transportation, model, strategic

Contents

List of Figures

INTRODUCTION: MODELS, DATA, SOFTWARE AND POLICY SCENARIOS

Models are used to represent a view of a part of the real world in order to reduce the complexity of the interrelationships of the elements in the view and subsequently to make it simpler to derive conclusions and logical relationships between the elements in the view. Once satisfied with the model, i.e. the results from an input impetus are regarded to be sensible and realistic, these results are translated from the simplified view, back to the real-world part thus having results, e.g. a forecast, of the changes that would occur for an equivalent real-world impetus.

Very basically, a model is a formulation (e.g. mathematical and / or verbose – the mathematical formulation has the advantage of being able to be processed by computer) of the real-world view being analysed. "Things" that can be analysed could be for example, the passenger flow between a set of different cities (passenger transport model), the freight flow between a set of different production and sales points (freight transport model) or the price of stocks and bonds of a market, e.g. the DAX in Germany or the NASDAQ in the United States. These examples already show that models and their formulations – since they are used to describe the real-world – can be as diverse and complex as the real-world itself. Nowadays, the only limitation to the practical use of complex models is the sheer computational performance required when using complex models on very disaggregated data.

A very successful way and the current state-of-the-art of modelling parts of the real-world is by using mathematical formulations (and computer data structures) to describe the elements and their relationships of the view of the real-world being analysed (– currently there are also a number of other concepts, e.g. using neural networks for modelling). And then to use aggregated or disaggregated data – depending on the specification of the model – to flesh-out the data structures and to run result procedures which in turn serve as input for the estimation of the model's parameters.

For a disaggregated model (e.g. passenger transport models) the basic input data required are socio-economic data of the elements (i.e. passengers), the sets of sources and sinks used (called zones, i.e. the areas of generation of transport demand, e.g. a zonal structure like the administrative zones of a country, which are areas populated by the elements) and structures representing the supply for transport (i.e. the infrastructures that support the ability for mobility, which are usually represented by network systems, e.g. a representation of road, rail etc. infrastructure). This information (the input data-set) is used to feed an estimation procedure of a model which in turn is used to describe the factors that influence the decisions taken by the elements. The output from the estimation are the parameters of the model which completes a model's practical specification – the mathematical formulation plus the estimated parameters based on the input data-set.

When one speaks of transportation modelling one is talking about two models that are used simultaneously to solve (or answer) a problem: a model of the transportation infrastructure (i.e. a model describing the supply side of transportation) and a behavioural model (i.e. a model describing the demand side of transportation). These models reduce the complexities of the real world into manageable chunks and, in principle, can be handled separately.

The infrastructure model defines networks (infrastructure supply side or abstract networks), vehicles (e.g. cars, trains, aeroplanes etc.), services (facilities for loading and unloading at a port) etc. that are based on the real world (observable) "things".

The behavioural model defines 1. abstractions of zones, zone features, choice alternatives etc., either in an aggregated or in a disaggregated fashion 2. the way that the formulated actors of the problem domain react and decide, given sets of choice options. This model is usually based on survey data. The more disaggregated the model (and therefore the required survey data), the more complex the model becomes mathematically. Disaggregated models have the advantage of being more accurate in forecasts and in their analysing behaviour. (At this level there are many connections to social science, because both try to explain differences in behaviour of groups based on their social, economic etc. differences.) The GTF data model [CHEN76] [NIST93] is another reduction of complexity, making the modelling information manageable for EDI. The reduction is done by grouping and classifying the modelling information. For example, the concepts centroid node and intersection node are very different in the problem domain (and the usual models), but they share a common function of being ending points of links: centroids being ending points of flow-links and intersection nodes being ending points of infrastructure-links. These kinds of abstractions are the gist of data modelling and the contents of section "Generalised Transportation-data Format (GTF)".

The current practice is to gather data (or databases) for models, to re-format the data, re-compose the information contained in the data and to apply the result as input to the model. This very often leads to problems because the information contained in the different databases used for the model aren't compatible, meaning that after re-formatting and re-composition, the data set shows that vital information is missing. After the model is run and the outputs generated the practice is more and more to use a GIS (Geographical Information System) to visualise the results [BRIDGES]. This also almost always leads to problems, because the structure used by a typical GIS doesn't contain information useable by a typical model. Also, there is the problem of matching the GIS structures and information to the model's informational structure before being able to visualise the results.

This paper defines an information structure (which includes information for models and information for GIS) and an exchange format that ensures that information isn't lost and that all the information contained in data is defined explicitly (– if the specification of the data model is followed accurately). The conclusions of this paper can primarily be applied to strategic passenger transport models but can be applied just as well to other models because all conclusions are derived from an abstract view of models, data, software used to implement models and software to view results, i.e. software to view the explained or forecasted flows between the modelled zones.

Current Situation & Problems

The usual use of a strategic model (e.g. for forecasting) is to define changes in the input data for each scenario to be analysed. The usual inputs that define "Policy Scenarios", like economic, demographic and spatial developments as well as network changes and changes in prices and fares for the use of transport [APAS96] supply are depicted in *Fig. 1*.

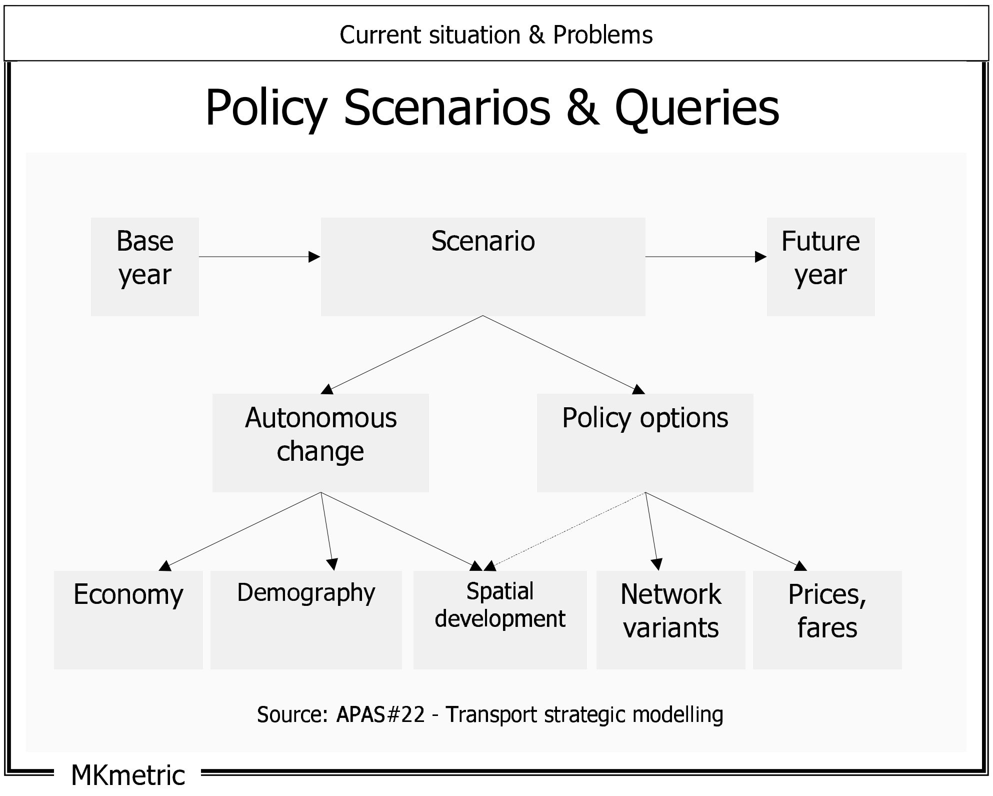

Fig. 1. ***Policy & Scenario Queries***

Fig. 1 shows that models are demanding on the amount of data needed. What also has to be ensured is the quality and informational contents of the data used for the model for estimation or calibration purposes. These two points are the main problems when trying to exchange input or output data for models. The exchange of data also includes acquiring data for models and transforming this data into the format and contents required by the model.

Another main aspect of using data or databases for different models is the informational contents of the data. Because, “data” can be defined as “the concrete value of a piece of information”. For example, a piece of data is “50” but the piece of information required by the model is “50 km/h”. The component describing the concrete value is a “fact”. The component describing the informational definition is the “meaning”. And only both together give a complete piece of information. Thus “information” can be defined as “fact” plus “meaning”. For the example, if one only has the fact “50” without the meaning “km/h” one would need to make assumptions about the meaning of the fact, which usually leads to errors. Because, “50” could also mean “miles/h” which would imply a different level for the data, as “miles/h” is about 1.6 times greater than “km/h”. This simple example shows that a model is extremely dependent on the correct information and not only the data.

Policy Queries & Models

The usual workflow, currently, is depicted in *Fig. 2*. The policy to be analysed using models is “translated” into model specific commands such that the concrete model, that is chosen by the user, is run and produces results – that are hopefully the results required for the user to make an objective decision for the problem at hand.

The figure shows that the definition of scenarios to be analysed by a model strongly depends on the manipulated information (and data) for the definition of a scenario. Using the example in the previous section, an increase of the speed for a type of an infrastructure (network) link (e.g. a road section) from “50” to “60” in “km/h” would be from “50” to “57” (approx.) in “miles/h”. This problem cannot be overcome even if the scenario defines the increase as a percentage and not by an absolute value, e.g. increase by 10%.

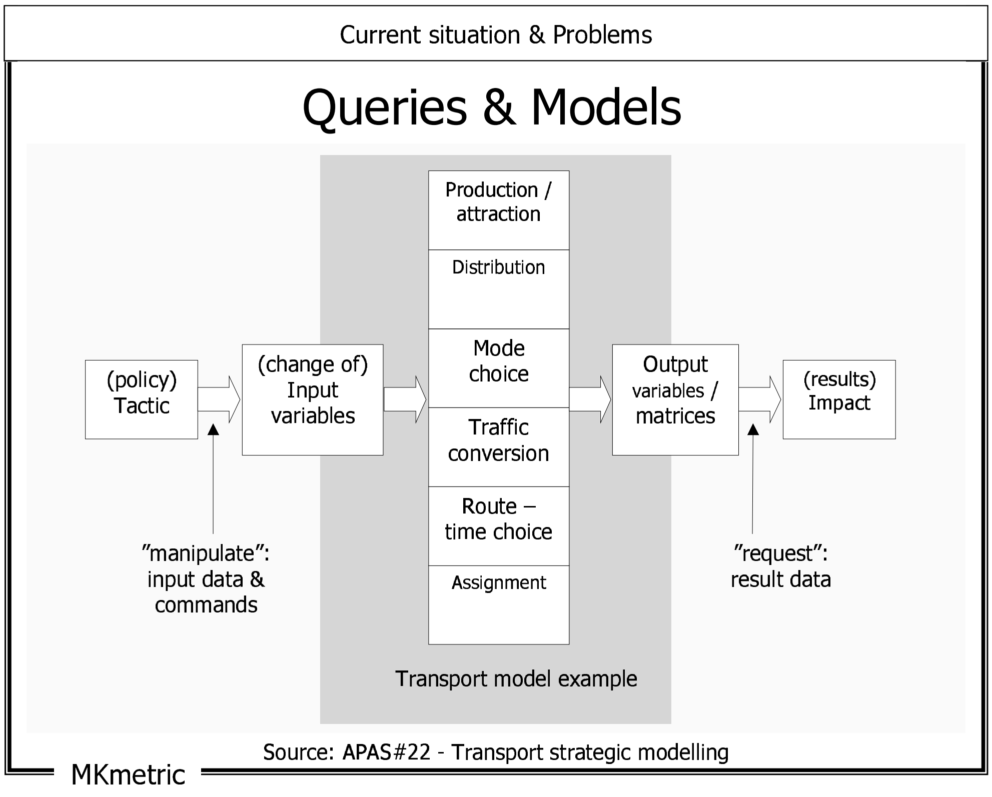

Fig. 2. ***Policy Queries & Models***

Current Data Pool

Currently, the structure of the numerous software applications and databases is inhomogeneous and largely incompatible with each other. Which, very frequently, leads to the problem (– rather the impossibility) of comparing results from scenarios based on different software applications and databases (see *Fig. 3*).

The problems mentioned in the previous sections also apply to the databases of the results from models (– not only on the input databases). Most models have specific database formats for the output. The output information is aggregated or composed according to the needs of the model package or the modeller using the model. The problems also apply, because it is often a wish to compare results from different models (from different projects) or to use the output from one model as input to another model. In these cases, the same problems of re-formatting and re-composition of the data arise.

The problem of trying to compare results from different software applications and databases lies in the fact that the underlying data-models (DM) and information structures for the model software applications and databases are incompatible or only made compatible with extreme difficulty and loss of information. This is because the informational units used by the data-models are to “large”, meaning one unit contains too much information. For example, the variable “time” is a typical variable used in (transport) modelling. But what does “time” mean exactly? The exact meaning is normally only known to the modeller himself and when this data (information) is transferred elsewhere, the problem of the exact definition of this variable arises, because “time” can mean “total travel time”, meaning for example for an air mode: network access / egress time + taxiing time + flight time. Mostly this information is implicit to the variable “time” used. What is needed is an explicit handling of this information.

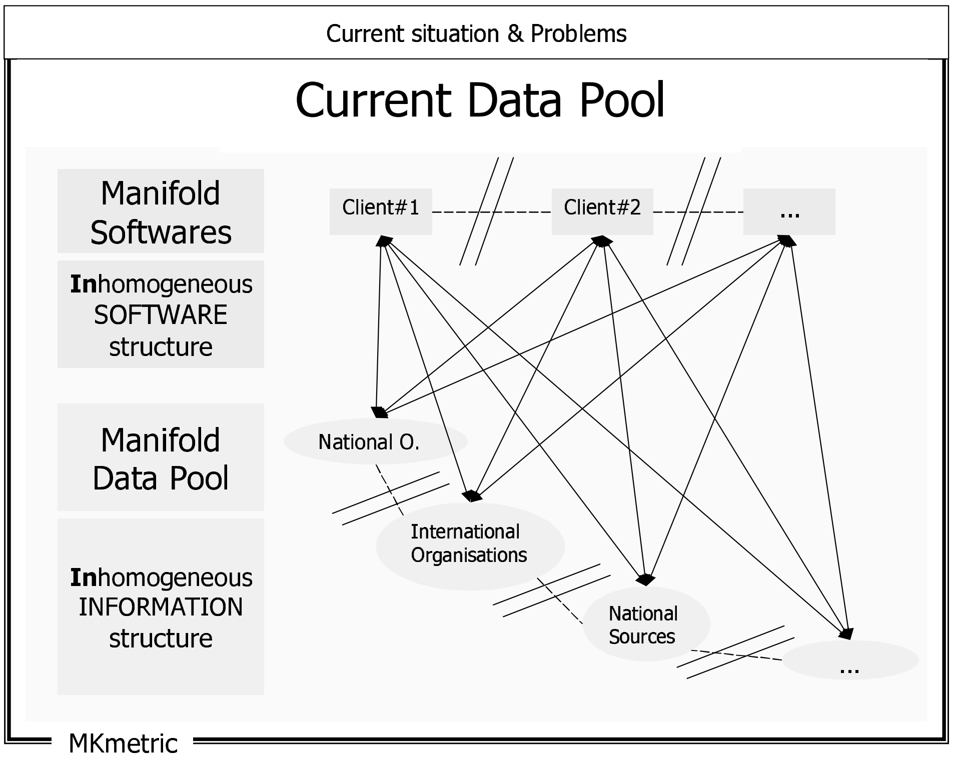
Current situation & Problems
Current Data Pool
Manifold Softwares
Inhomogeneous SOFTWARE structure
Manifold Data Pool
Inhomogeneous INFORMATION structure
Client#1
Client#2
...
National O.
International Organisations
National Sources
...
MKmetric

Fig. 3. ***Current Data Pool***

Matching Data Models

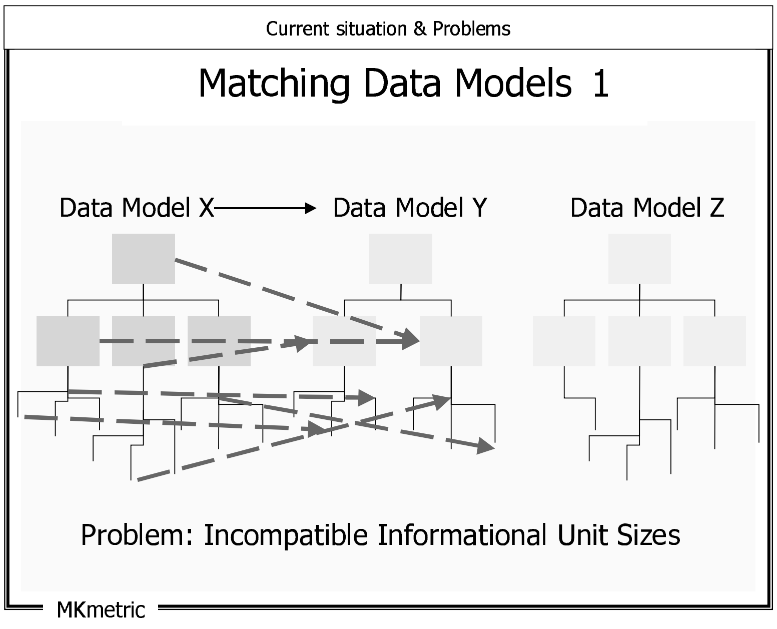
Current situation & Problems
Matching Data Models 1
Data Model X
Data Model Y
Data Model Z
Problem: Incompatible Informational Unit Sizes
MKmetric

Fig. 4. ***Matching Data Models Step 1***

Due to the “large” sizes of the information units, it is virtually impossible to manage the task of matching informational units contained in different databases without loss of information.
When more than one matching of informational units in different databases is required, there is a loss of information in each step of transformation, which very often leads to the fact, that the output of all the transformations won’t contain all expected and relevant information, see *Fig. 5*.

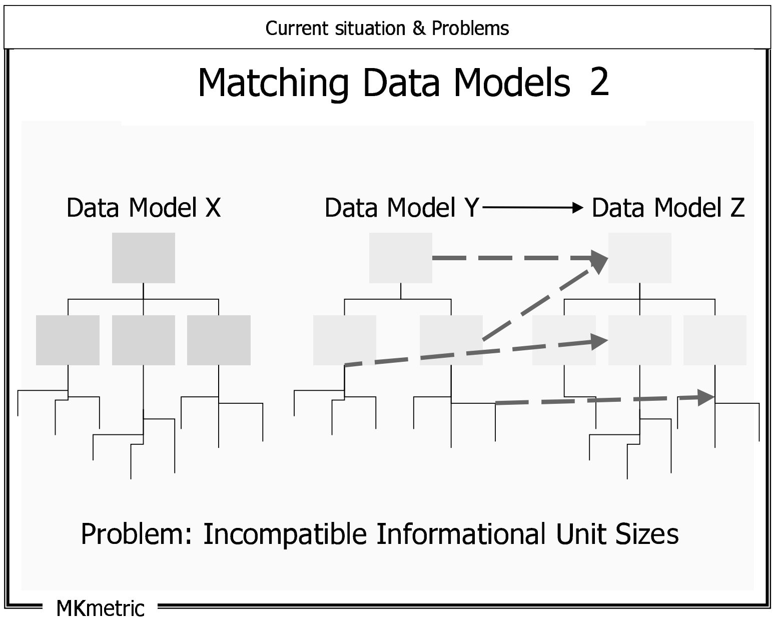
Current situation & Problems
Matching Data Models 2
Data Model X
Data Model Y
Data Model Z
Problem: Incompatible Informational Unit Sizes
MKmetric

Fig. 5. ***Matching Data Models Step 2***

Models & Data

A broad look at information & data for transportation models

One needs to examine the kinds of data and information used by different transport models, to approach the task of specifying an information / data model for GTF. Very generally speaking, transportation models use the following information items for their computations: 1. zonal data: any kind of zonal description, e.g. socio–economic data, ecological data, zonal boundaries, transport data, indicators, transport matrices etc. 2. network data: data describing the relations between the elements, e.g. link characteristics, a link has a starting node and an ending node (i.e. topological characteristics), link/network clusters etc. 3. GIS data: the necessary information for visualisation purposes, e.g. the underlying projection of the node and its co–ordinates. All these types of information must be must be part of the data exchanged with GTF. With these different kinds of information in mind, a more detailed view of the information / data categories for transportation information can be developed.

Typical Problems

Models in general are very demanding on the amount and quality of input and calibration data (e.g. even if they aren't discrete choice models). The main problems with current data and databases are 1. data required by the model, e.g. for estimation, isn't available. For example, a pan-European passenger transport model requires homogeneous input data (i.e. equal in structure and information contents) from all countries at the same aggregation level. This kind of database isn't available currently and when data (or interesting information for the model) is found not only the format does not correspond to the other data, but also the information contained in the new data lacks essential parts. I.e. the description of the data sounds good, but in detail there's missing a vital piece of information (or data). 2. the composition of the available data required by the model doesn't match and re-composition isn't possible. For example, a database that was acquired for a model, has the data in an aggregation that can't be matched to the one needed by the model. If the model requires a NUTS zonal division [EUROSTAT94] [EUROSTAT95] of the data but the acquired data has a different regionalisation (e.g. ITP) the units of data (in the first case a NUTS zone and in the second case an ITP zone) do not match exactly, thus re-composition of the ITP data into data fitting the NUTS regionalisation is necessary. This can only be done, if further information concerning the amount of the ITP data for a zone that fits into the equivalent NUTS zone (i.e. the percentage of the ITP data for the zone that fits into the NUTS zone) is available. Also, it is usually the case, that you need to take different percentages from a set of ITP zones to create an equivalent NUTS zone (and vice versa). Making the re-composition a tedious and error prone task. 3. the aggregation of the available data doesn't match. This often means, that the acquired data is aggregated to a higher level than required and can't be disaggregated to the level needed, because e.g. the disaggregation procedure isn't known.

Conclusion

Because of the problems mentioned, there is the potential of significantly increasing the value of models' databases by not only homogenising them but by defining a generalised structure with which information contained in (current and future) databases and the required information (not just the data) for models can be described precisely [SPOTLIGHTSTN]. The first (and main) advantage would be to have databases which can be exchanged, enriched, corrected and used by models in a homogenised manner, drastically reducing the amount of redundant work, e.g. by having to reformat and re-arrange pieces of data in order to fit the data requirements of a specific model. Secondly, it can be ensured that the required information is actually contained in the data. The previous sections, gave a brief overview of the problems that arise when having to do major re-formatting tasks. Therefore, a "Generalised Transportation-data Format" Data Model will be developed in the following sections. Also a specific exchange format will be defined to be able to process (exchange) concrete GTF Data Models by electronic means.

GENERALISED TRANSPORTATION-DATA FORMAT (GTF)

Basically, the GTF data model is a framework which can be used to define the information that is contained in data. The GTF data model framework allows a user of the GTF specification to wrap data into information entities. These entities contain the basic data and the necessary supplementary information (meta–data) to give a meaning to the basic data. Like this, one can make sure that the input data to a transport model fits the model's information requirement. This is vital for a model to compute valid results. If the input to a model doesn't fit the assumptions that were made concerning the information carried in the data (i.e. the meta–data implicitly associated to the input data), then the model is very likely not going to produce valid results. Here, the GTF data model comes into play. It enforces a user of this specification to make the implicit information explicit by wrapping the data – data with implicit information – into entity structures – data with explicit information. These entities are then combined to represent the complete implicit information that a piece of data carries. In this way all the data's information is made explicit and thus can be used to check whether data fits the model's philosophy or not.
How can one know which entity to use to wrap data into? The first step is to make it clear to oneself which information a piece of data is carrying. For example, a piece of data could be "60", but the information it carries for a user of the GTF data model (typically a model provider that needs to package the model's results into a GTF-XML file according to this specification) is

“speed”, “maximum”, “kilometres per hour”, “administrative regulation for zones in Germany” and “on the Link 87321 between Node 983 and Node 1001”. Now, one could define a new and separate attribute “maximum speed in km/h for Germany” for the Link between Node 983 and Node 1001. But to do this for every possible combination of these pieces of information is not possible due to the combinatorial explosion. It would also be very ineffective to define always new attributes for a piece of new information. The question is also, what happens if a new piece of information is associated to the data, e.g. “for cars not for trucks”. Shall a new attribute be defined? This doesn’t seem very elegant.

Therefore, the GTF data model provides a standard set of information pieces that can be used to wrap data into and to combine the pieces of information into a larger chunk of information, just like building something using LEGO[1]. With LEGO one can build many different things without having to buy always different pieces of LEGO. One can use the standard set of LEGO pieces and still build a very large number of different things. One only needs a really new piece of LEGO, if one wants to build something that doesn’t fit into the concepts of the available LEGO pieces. For example, with only square LEGOs it is impossible to build something round (– at least it’s very difficult). In this case, the concept of “round” is totally different to the concept of “square”. The “square” concept is covered by the available pieces of LEGO, the “round” concept is not. Thus, one uses a new “round” LEGO to bring the concept to life. In the case of GTF, all this applies equally, the different concepts of LEGO pieces, e.g. “round”, “square”, are defined as entities. Each specific real piece of LEGO is an entity instance. A construction made of LEGO is a GTF-XML file with the entity instances and the definitions of the relationships between them. The main advantage of this kind of thinking is the minimal amount of different abstract concepts used to cover a very wide range of concrete things.

Coming back to the question “how can one know which entity to wrap data into”. The answer is fairly simple. Think about what information is implicit in a piece of data and choose the correct combination of entities from the data model and follow the pre–defined relationships of the data model.

The GTF data model is a complete data model in the sense that all parent entities required to define a child entity are also defined as separate entities, although the parent entities are abstract and mostly mentioned to have a compete framework. The entities that are actually used in a GTF-XML transmission have a definition of a XML segment following the definition of the attributes associated with the entity. For example, the entity Terminator which only captures the concept of “something that is the beginning or the end of something else” is abstract, the concrete concept is an intersection Node which is “a point in an infrastructure network”. Thus, in a GTF-XML only Nodes should be transmitted, because the Terminator information is automatically known. This information is implicit in the GTF-XML file, but explicit when looking at the Node definition in this specification. This means, that when a Node is transmitted, the receiver (who also has to know this specification) automatically knows that a Terminator is the parent of the Node. One can transmit a Terminator, if needed, but a Terminator isn’t concrete and can, therefor, be either a Node or a Zone. It isn’t possible to determine which, according only to the Terminator information. The information whether a Terminator’s role is an intersection Node or a centroid Node is explicitly contained in the corresponding entities in the data model. Thus, one Terminator can be used as parent of an intersection Node and a centroid Node, if the information to be conveyed is: “This starting / ending point has the role of an

[1] LEGO is a trademark and copyrighted by LEGO Systems AIS, Denmark

intersection Node and it is also the input / output point of a zone, i.e. the zone's centroid, in this network".

One might be flabbergasted by the sheer number of defined entities (approx. 200) when looking at the list of entities in the data model. But the basic entities that were used to create the data model are a total of only 9, namely SpawnFactor, Terminator, Link, Specification, Vessel, Service, Alternative, Unit and Meta which are called "topmost entities" or "top–levels".

The top–levels and their children (called "concretes") can be combined using the defined relationships. These relationships are defined by a user of this data model by filling out an attribute in an entity that was migrated to the entity through the relationship. This means that the entity attributes defined in the data model are either generic to the entity or have been added to the entity because of a relationship.

Once it is clear which entities to use for the transmission of a piece of data, one needs to generate the corresponding GTF-XML segments and construct a complete GTF-XML interchange transmission file.

Definitions

More specifically, what is meant by "Transportation Entity" in this paper is:

1. Transportation = "The act of moving passengers or freight in space."

2. Transportation Entity = "All that spawns, enables or hinders movement of passengers or freight."

3. Transportation Relationship = "The connection between two transportation entities."

4. Transportation Attribute = "A quality or feature of a transportation entity that is a central part of its nature distinguishing its instances."

For example, concerning the entities Zone & SpawnFactor, this means, what is meant by SpawnFactor are those information that are not essentially part of a Zone's nature, e.g. age distribution etc.

The things that are part of a Zone's nature are attributes of the entity, e.g. barriers (mountains, lakes etc.) separating one Zone from another. This kind of information is kept within the Zone entity. Also, the definition of SpawnFactor in GTF not only contains the raw data, but also the meta-data. I.e. each SpawnFactor instance contains the raw data, e.g. 5, and the necessary meta-data, e.g. "statistical source = EUROSTAT, type = statistics".

Obviously the definitions above cannot be used for direct implementation. The goal of these definitions are to be able to define a data model of transportation and not to implement this data model. The implementation is left to eventual providers who have to adopt GTF as one of the exchange formats of their software/model.

These definitions provide a good and general basis to derive a data model for GTF.

Now reading these definitions one might argue "since a pile of leaves might hinder movement, by definition it should be an entity of transportation. Does that make sense?" Basically, yes, it does make sense. Although practically, no. Since GTF was conceived as a data model for (mainly) strategic transportation models which are long-term (e.g. 10-50 years forecasting flows, impacts etc.). If say one would devise a model that takes "piles of leaves" into account, then it is an entity instance (i.e. an object from a class [BRWON97] [BUDD97] [RUMBAUGH91]). And in fact, the GTF data model does contain objects called "SpawnFactor"s that are used to store all the spawn-factors which generate, attract (basically make or influence) movement and the data model also contains the entity “Barrier” (associated with Zone) to describe all objects that are in the way of free movement (mostly, e.g. mountains, rivers etc.). So the answer is, “if the pile of leaves hinders movement, say on Link L, for the totality of the period being analysed with the model, then yes, these leaves have to be considered an entity of transportation, just like you would consider a river flowing between two Zones, also being a transportation entity which hinders movement.”

Structure Overview

Due to all the problems described above, the structure of a “Generalised Transportation-data Format” data model should cover the following 1. instead of having disparate and manifold software applications and databases, GTF contains all necessary elements and provides one single and homogenous data specification and format 2. instead of having incompatible proprietary formats and informational contents, GTF should be used throughout any computer system, by providing translators to / from the proprietary formats to GTF.
GTF consists of 1. a generalised data-model (GTF-DM), 2. a standard exchange format (GTF-XML) and 3. generic commands to run models and retrieve results (TIP).

GTF Data Pool

With GTF, the structure of the numerous software applications and databases are accessible in a homogeneous and compatible manner. A set of GTF Translators could provide a single access point to all models and data. The problems discussed previously of inhomogeneous software and data / informational structures and definitions is overcome by using the GTF Data Model specification [MKMETRIC99] [EUROSTAT96] to structure and flesh-out databases and for the exchange of information (by using GTF-XML), see *Fig. 6*.

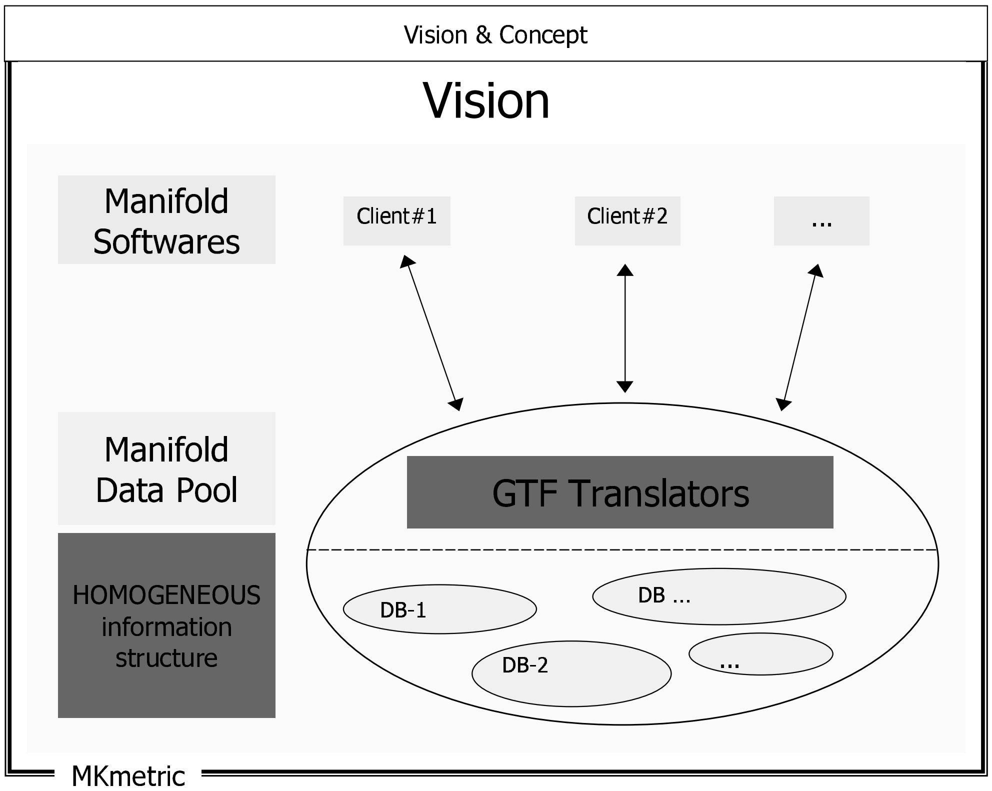

Fig. 6. ***GTF Data Pool***

The numerous databases can either be restructured according to the GTF Data Model or a specific GTF Translator for each database can be developed thus providing a homogeneous and single access possibility (see *Fig. 6*). Software like GIS can be enhanced by a GTF Translator to be able to access the information of GTF databases.

Matching Data Models using GTF Step 1

The main concept for the development of a GTF Data Model is that the informational units of GTF are “atomic”. Therefore the informational units (- the data) of any other DM (DM-X) can be decomposed according to the GTF-DM (see *Fig. 7*).

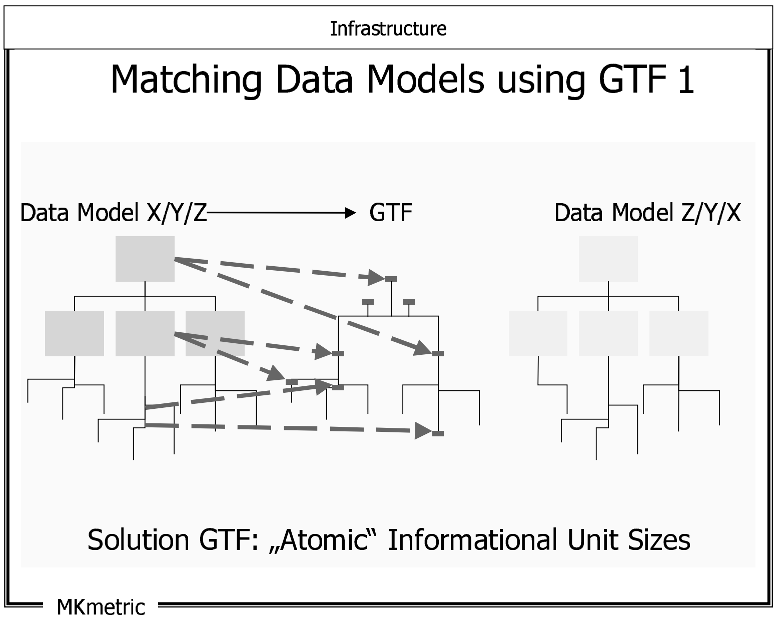
Infrastructure
Matching Data Models using GTF 1
Data Model X/Y/Z
GTF
Data Model Z/Y/X
Solution GTF: „Atomic“ Informational Unit Sizes
MKmetric

Fig. 7. ***Matching Data Models using GTF Step 1***

For example, an information unit of some data-model DM-X of “flows”, say “time”, is “50”. This is a non atomic information because one can’t know whether there is much implicit information or not, without having the exact definition of the data-model (and thus the exact definition of the attribute “time”). “Time” might be an aggregated value meaning the “Total travel time”. This, too, is aggregated and could mean f.i. the sum of access / egress, taxiing time, flight time, etc. Already this example shows that implicit information can be numerous. To avoid this confusion, the GTF data-model offers containers of “atomic” information which must be used to describe aggregated information as in the example above.
In the GTF-DM, all pieces of information that qualify a piece of data are kept in separate entity instances which are linked through relationships to the entity instance containing the piece of raw data. In the case of the example, this means, that “access/egress”, “flight time”, “total time” etc. are qualifiers (“flags”) attached to the raw data “50”. In this way, the implicit information is made explicit, because each implicit piece of information is reflected in an entity in the GTF-DM.

Matching Data Models using GTF Step 2

Following the example in the previous section, the decomposed information from DM-X can be re-composed differently than in the original DM-X using the GTF informational units, creating the data according to a different DM (DM-Z). I.e. assembling of aggregated information units from the GTF-DM units can be done in analogy to the decomposition of aggregated information units into GTF-DM units (see *Fig. 8*).
The main focus for the development of the GTF specification and subsequently the GTF Translators is

to define an abstract view of transportation models’ information for the purpose of implementing translator software to exchange data electronically between modelling software and other software (e.g. model packages, database systems).

The primary goal therefore is the definition of a data model for transportation information (a GTF specification of information structure) and the definition of a format and syntax for electronic transfer of information (a GTF Translator syntax and the format of information data).

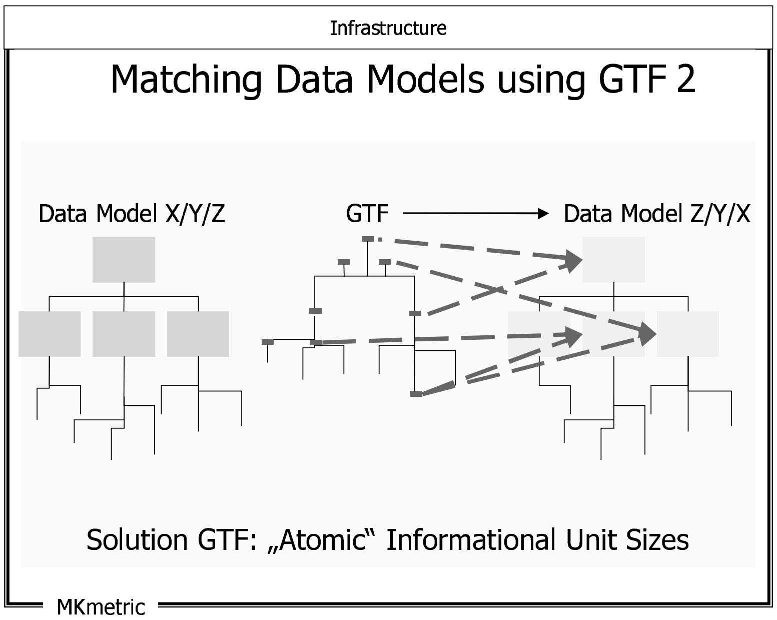
Infrastructure
Matching Data Models using GTF 2
Data Model X/Y/Z
GTF
Data Model Z/Y/X
Solution GTF: „Atomic“ Informational Unit Sizes
MKmetric

Fig. 8. ***Matching Data Models using GTF Step 2***

The GTF Data Model

This section introduces in more detail the fundamental information classes that are the foundation of the GTF Data Model. The transport data that is covered is primarily that which is used in strategic transport models. Thus it covers interurban, regional or international travel on all transport modes for both passengers and freight. It does not cover detailed local traffic issues, such as the representation of road junction geometry [MOELLERING97]. But this can be described using GTF, too.

GTF Data Model Overview

Some basic concepts from economic theory (i.e. supply side determinants, demand side determinants and the market where supply meets demand) were used to develop the concepts for the data model [BUTTON93] [ORTUZAR90]. *Fig. 9* depicts the main conceptual entities used in modelling information (without going into details).

A number of SpawnFactors (not depicted) determine the generated or attracted movement, which together induce the demand for movement and transport. SpawnFactors are for example the GDP, age distribution, level of income etc. for one Zone, a group of Zones or an aggregation of Zones.

A centroid Node is a virtual point for input & output (source & sink) of movement in networks. It is associated with Zone which contains the SpawnFactors of an area. For transportation models a Zone is a description of socio-economic and other information of a geographical area. The geographical connection between a Zone and the area it describes is used to relate specific SpawnFactors and their values to specific input and output points in infrastructure networks. In this context the virtual input & output points are called centroid Nodes. These kinds of Nodes are not to be confused with infrastructure (network) Nodes, that are descriptions of junction points which in turn are an abstract description of some part of a physical transportation network.

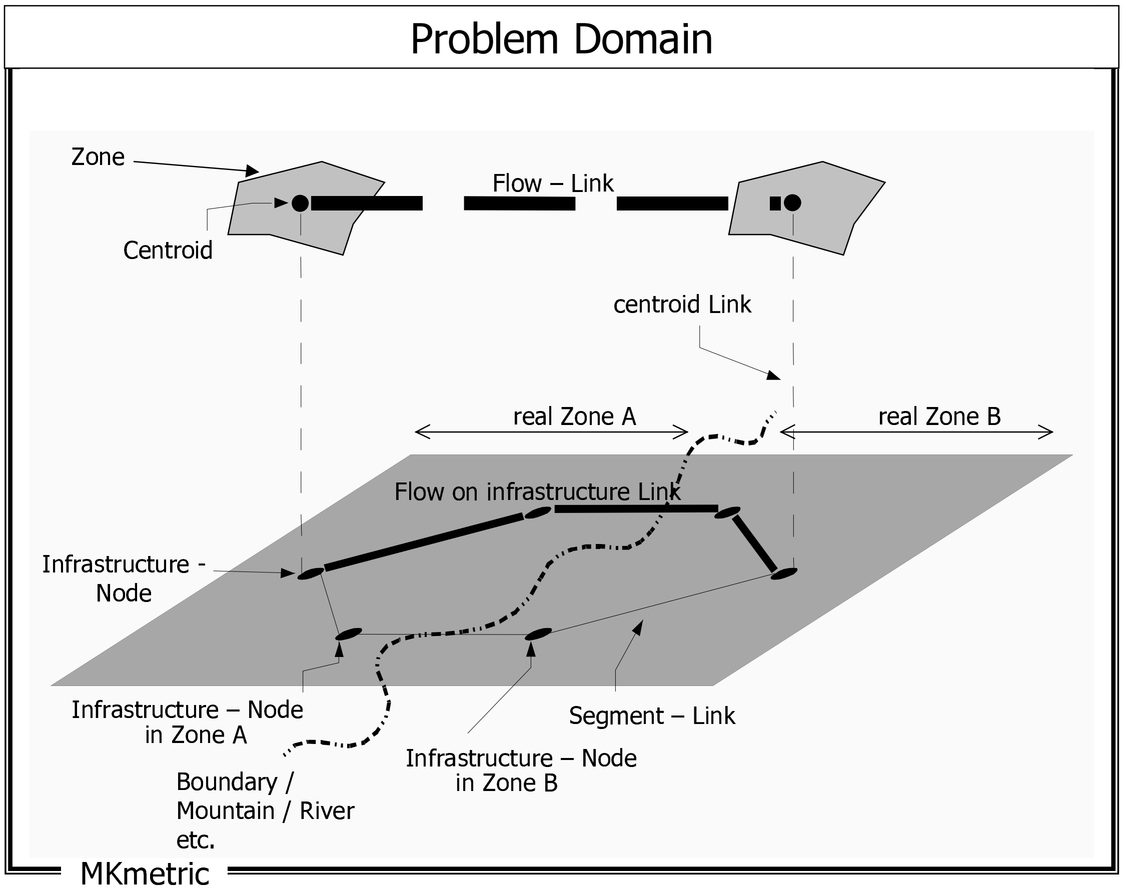
Problem Domain
Zone
Centroid
Flow – Link
centroid Link
real Zone A
real Zone B
Flow on infrastructure Link
Infrastructure - Node
Infrastructure – Node in Zone A
Segment – Link
Infrastructure – Node in Zone B
Boundary / Mountain / River etc.
MKmetric

Fig. 9. ***GTF Data Model Overview***

The concept behind SpawnFactor and Zone is the following: an area itself (e.g. 10 km^2) cannot be the reason why traffic or movement is produced or attracted. The reason why there is movement to and from an area, is that the area has a number of features that the travellers to an area are interested in. In this paper these features, e.g. people, industry etc., are called SpawnFactors in analogy to the concept of production factors in economic theory. A Zone then, is the combination of an area (either in physical space or in the modelling space) and the SpawnFactors, that are located in the Zone's area. The specifics of the Zone's area are described by the combination of SpawnFactors and a Zone represented by the relationship "activity", because a SpawnFactor in a Zone generates some sort of activity, either attracting movement or producing it.

A centroid is connected to an infrastructure network through a centroid Link. The centroid Link is the virtual description of the impedance(s) that is needed in average to enter / leave a Zone and thus creating inter-zonal transportation / movement called LinkFlow.

A LinkFlow is the result of SpawnFactors generating & attracting movement across the limits of Zones. It can therefore be described as a connection (relationship) between two Zones. This is "flow" in the sense of demand for transportation. A LinkFlow is the container of information that exists when two specific Zones are connected, because there is demand for movement between them. Thus, a LinkFlow is a connection between centroid Nodes with information about the amount and types of flows (vehicles etc.) that go into and come out of a Zone.

A "node" performs two functions in transportation modelling. The first function is to relate (connect) a Zone to some point in the network as access and egress points for mobility. This function determines the Node as being a "centroid Node". The other function is that of being a junction point, which determines the Node as an “intersection Node”. These junction points describe topological aspects of infrastructure networks, i.e. which Nodes are connected to which other Nodes. The Nodes and the connections (branch, arc, edge etc. called Link) between them are the topological description of the networks.

A Link is a topological relation between two Nodes. The Nodes in turn usually are associated to specific geographical co-ordinates in real world space. But this is mostly needed for visualisation and presentation purposes.

An “activity” is a term for “determines demand of”. This term was chosen, as it describes better the concept of the entity SpawnFactor. “activity” can also be seen as abstraction of the attractiveness of Zones or the potential for a visitor of the Zone to see sights etc. An “activity” describes everything that induces movement/transportation to/from a Zone.

Following this kind of logic of decomposition and analysis, 9 toplevel[2] entities are defined in the GTF: Terminator, Link, Vessel, Service, Alternative, SpawnFactor, Specification, Unit and Meta. The next table gives definitions for each toplevel.

Entity Name	Description
Terminator	This entity is the generalisation of the concept “start or ending point of Links” and thus a generalisation of the centroid and intersection Node concepts usually used in modelling and graph theory; and its function is to act as the starting / ending point of Links. The Terminator entity is the generic entity for both types of Node. The generic entity Link therefore is determined by exactly two Terminator entities which can be either a centroid or an intersection Node. In this way a more homogenous view on the problem domain and especially the similarities between centroid and intersection Node with respect to their function defining Links is achieved. FUNCTION: the starting / ending points of Links
Link	The Link entity is not only an abstraction for all types of infrastructure network links, but it incorporates the connections between two Zones (through their centroid Nodes) when modelling flows. Thus, a (zonal) Link is specified by exactly two centroid Nodes (of two Zones) which are the starting and ending points of a flow. Centroid and intersection Nodes in different combinations act as Terminators to define Links. The three possible types of Link are (depending on the combination of centroid and intersection Node): 1. the LinkInfrastructure is a Link between two intersection Nodes that is used to describe the supply-side of transport, i.e. infrastructure elements that supply the possibility of movement / transport, e.g. road links etc. 2. the LinkConnector between an intersection Node and a centroid Node is a Link that describes the avg. travel-times, costs, speeds describing the avg. disutility to reach (any) point in the Zone. 3. the LinkFlow between two centroid Nodes is a Link that holds the flow information that results when two Zones are connected to describe the movement between two areas in space. FUNCTION: the possibility of movement between two Terminators
SpawnFactor	SpawnFactors determine the generated or attracted movement of a Zone, which together induce the demand for movement and transport. SpawnFactors are for example the GDP, age distribution, level of income etc. for one Zone, a group of Zones or an aggregation of Zones. A SpawnFactor is a piece of data (aggregated or disaggregated), e.g. socio-economic or other statistical data, that is used to compute / describe the potential for transportation demand that an actor / group of actors generate / attract. A SpawnFactor is

[2] A toplevel entity is a parent from which a hierarchy is derived. The toplevel entities and their relationships are the foundation of the complete GTF Data Model.

Entity Name	Description
	unique to a Zone, because a SpawnFactor has an explicit value for some demand SpawnFactor of the Zone, e.g. GDP=5000, which is Zone specific. SpawnFactor objects can be seen as a container of the attributes of the associated Zone object. Because of its importance it was defined explicitly as an own class. FUNCTION: it is used to describe actors (or group of actors) attributes that are used for Zones in the transportation model. Actors are the reason why movement and flows are generated or attracted.
Specification	Specifications are characteristics which can be attributed to Links. The Specifications of a Link can be defined by Zones, Vessels and Units. For example, information concerning speed-limits or tolls are defined by a Zone's location and its political / administrative regulations. The Specifications can be dependent on a Zone's country attribute value, because of the relationship "defines regulations for". These specifications are regulatory / administrative or defined by engineering science. The interesting point to note here is the fact that the "specification" (e.g. maximum-speed limits from an engineering point-of-view) of a Vessel like a car can depend on the Zone's location (e.g. Germany, UK) and the type of Link (e.g. motorway, 2-lanes, 4-lanes), because the law and regulations for the different types of Vessel that are allowed to use different types of Link differ by country. FUNCTION: this entity associates all the technical, statistical and movement specifications that come from Zone, Vessel and Unit and defines (physical) characteristics of a Link
Alternative	Models use choice alternatives (e.g. usage of road or rail or air mode for transportation etc.) to describe the situation individuals (or the behavioural units being modelled) face in certain situations. The model then "decides" which option the individual chooses by taking into account different aspects (socio-economic, economic, psychological etc.). From a modelling point-of-view the Networks (i.e. the groupings of Nodes, Links etc. which form a logical whole) need to be distinguished according to different "main modes" (or Alternatives), because models use these "main modes" to differentiate elements of sets of choice alternatives. As the term "main mode" intuitively implies one single mode (i.e. Vessel), which isn't the proper concept, because a model's "main mode" can imply any number of modes, the more precise term Alternative, short for "choice alternative" is used. The instances of the Alternative entity define choice alternatives (or choice of combinations of available means of transport) for a model, through the combination of Units (person/good/business/private/holiday ...), Services and Vessels. These choice alternatives are "main modes" when associated to Links. Certain Links can only be used by some of the Alternatives, because e.g. the Link can't cope with a Vessel of some given tonnage used in the Alternative definition or some other restriction due to the definition in the model, i.e. an Alternative's main mode might not be allowed to use a Link, but otherwise (physically) it is allowed. Note, the term "main mode" is an alias for "choice alternative" and doesn't imply only one single mode. Because an alternative can be defined using any number of "modes" (Vessels). The entity Alternative comprises all the definitions for a Link that are derived from the modelling-side of the information, e.g. the "main mode" might be "road". This actually comprises road links as well as car ferries etc. The Alternative entity associates this information to a Link. The difference to the entity Specification (and Vessel) is, the Alternative describes a logical (modelling) use of a Link while Specification describes the "real world" usage of a Link, e.g. a Link might be a ferry link, but if it is a car-ferry the modelling usage

Entity Name	Description
	would say that the link is a road link. FUNCTION: a container for information pertaining to the definitions of choice alternatives for a model
Service	A service provides a traveller with the means to travel with relevant choices already made in advance by the service operator. The Service entity is a container for information pertaining to services, e.g. public transport. This entity is a definition of a type of service, the used carrier Vessel(s), the level of security attributed to this type of service and the time-table for the service. The instances of Service are used by Link and Alternative. The relationship “real definition” to Vessels, associates the Vessels that a Service uses to support their service. The entity Service is the container for specifications concerning services, carriers etc. that use a Link. With this entity a Link also has “usage” information apart from the “physical” usage. FUNCTION: bundling of assistance to a user (=traveller) for travelling purposes
Vessel	Vessel is the abstraction of everything that increases the flow-count on any Link. In transportation models typical Vessels are cars, trains, aeroplanes, trucks etc. A vessel is a logical view of objects / entities that can use links to travel / transport some person / good from one point (Terminator) to another. A vessel description contains all that characterises a vessel object. A Vessel is anything that moves and uses infrastructure entities, e.g. cars, planes, persons that use roads, rails, airways etc. There is also the virtual Vessel like a “human” that uses the mode “walking”. For example for the access / egress points (where the car is parked) of a railway station or airport to the points where one actually enters the railway station or airport, one has to walk. Thus one is using the mode “walking”. FUNCTION: a container of information of that that travels / moves on Links
Unit	Units define the type of unit being moved or transported, the purpose of the movement or the date / time schedule of a movement. This entity contains all such information and associates this information with Alternatives and Specifications etc.
Meta	Metas are objects to define meta-information which do not pertain to modelling or network or other information, but are rather complementary information describing units of measurements etc. The Metas can also be used to associate dimension information with all other entity instances, because each Meta instance has two attributes “for entity code” and “for instance”, which uniquely associate the Unit instance with the instance of some other entity (with that entity code).

Using these high-level definitions approx. 200 entities are defined in the data model.

It is important to note that an <u>entity instance’s function</u> in the data model is either to actually <u>hold raw data</u> or to serve as a <u>qualifier to another entity instance</u> holding the raw data.

Fig. 10 depicts the toplevel objects and their relationships.

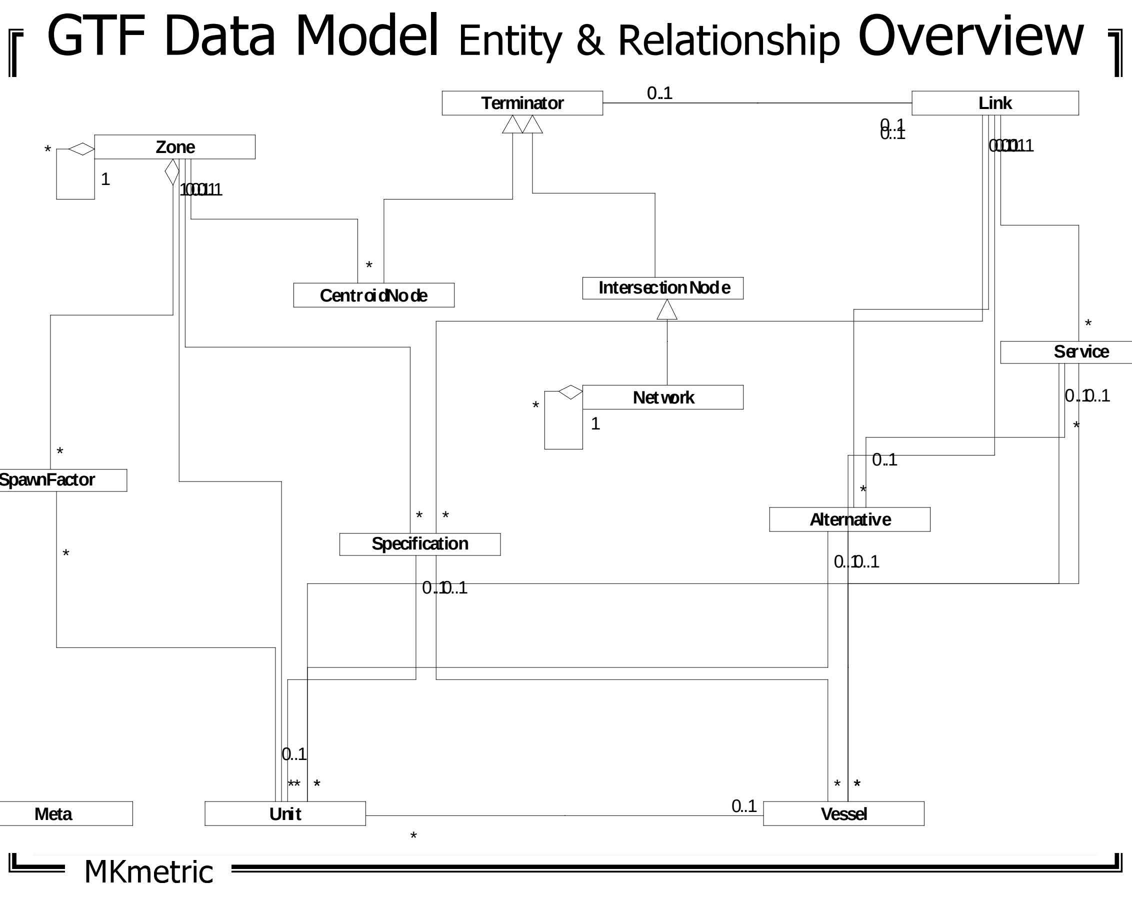

Fig. 10. *Overview GTF-DM*

These basic types (entity or class) of information are further sub-divided in the GTF-DM until the level of detail required (one bit) is reached. *Fig. 11* [UML] gives a bit more detail about the data model structure at a high-level.

All entities of the GTF-DM have a "GIS" part and a "TYPE" part. The "GIS"-part is used to capture graphical information, e.g. co-ordinates, vertices attached to Links etc. The "TYPE" part contains the information relevant to models, e.g. modes that are part of an alternative. To explain the principles of the data model structures as an example, the informational unit "Terminator" structure is depicted in *Fig. 12*.

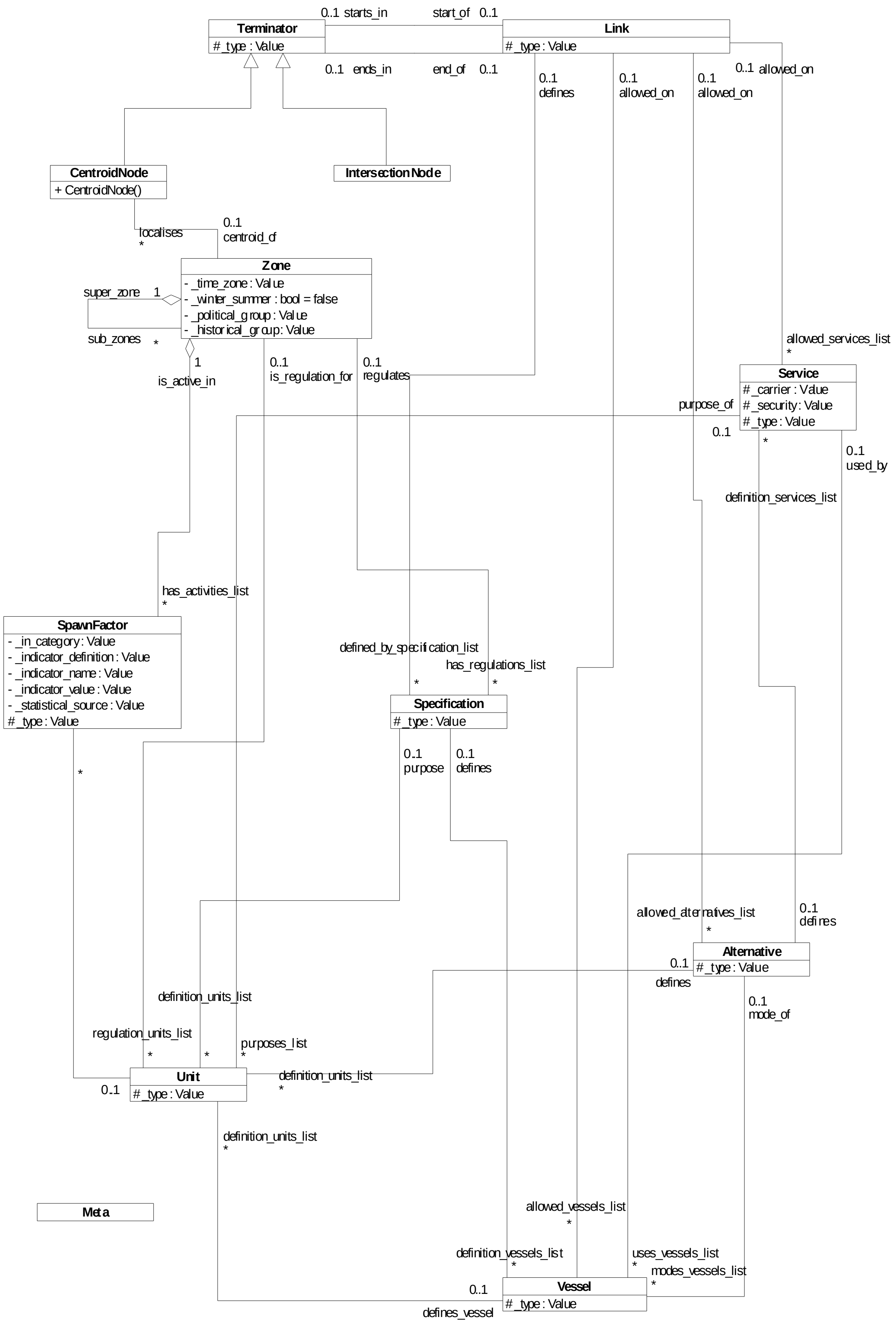

Fig. 11. *UML diagram of toplevel entities*

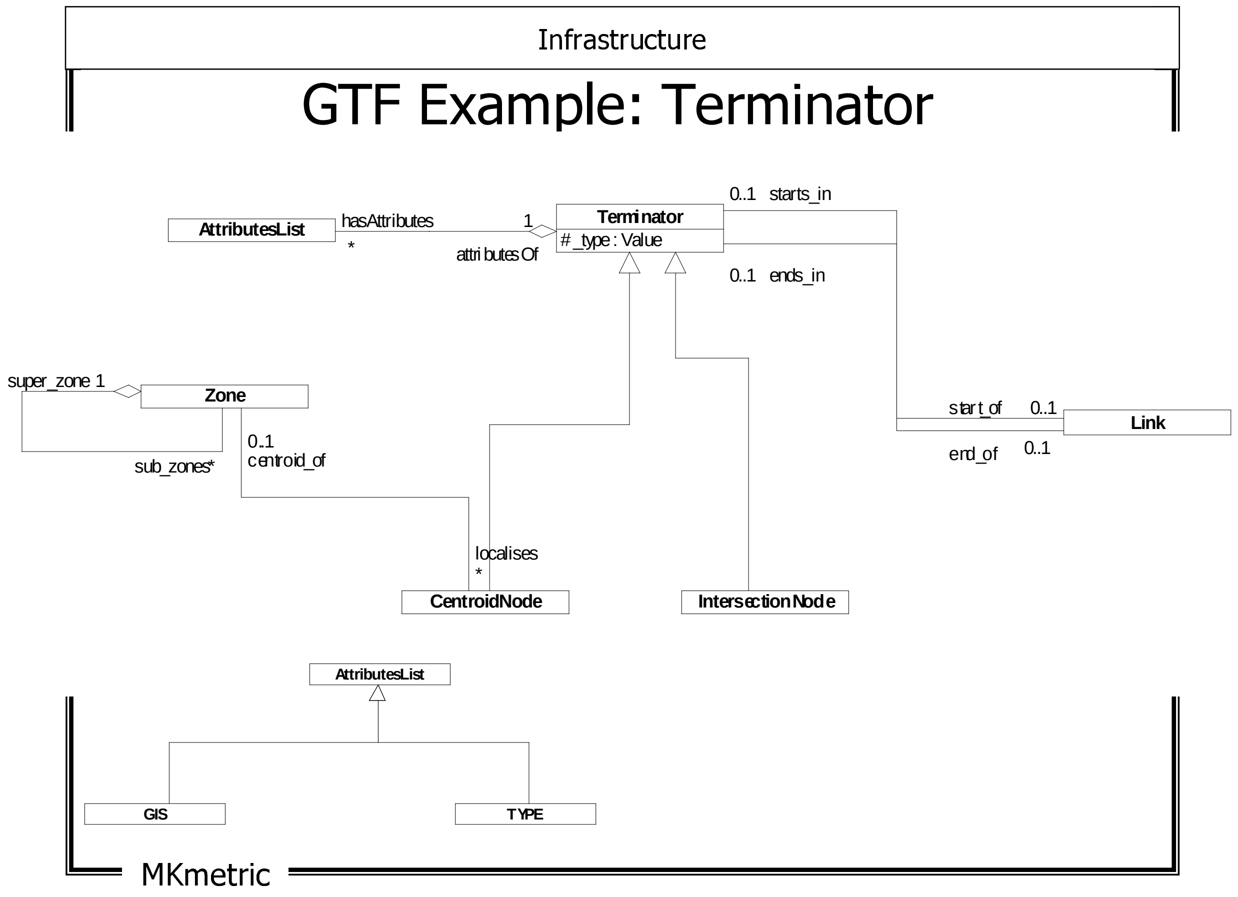

Fig. 12. ***GTF Example: Terminator***

All Terminators (all entities) have a Type entity of information and (optionally) a GIS entity of information associated to them.

The GIS entity information of a Terminator is for example Symbol entity = definition of the symbol to use when displaying the Terminator graphically in say a GIS. This entity has a relationship to a Shape entity which captures the co-ordinates defining the Shape.

The Type entity information of a Terminator are for example Infrastructure entity, defining the Terminator as a network intersection Terminator without further structure. (In contrast, the Type being (sub-)Network entity would define the Terminator as a sub-network, which itself is a grouping of (possibly) all other entities in the GTF-DM. This allows for different levels of detail in a network.)

All Terminators have a relationship to a Zone (called “is in”, not depicted), defining the Zone in which a Terminator is contained. All Zones have relationships to centroid Nodes (called “localises”) defining the centroid Nodes which connect the Zone to the network. Intersection Node and centroid Node are children entities of the Terminator entity. The function of Nodes is to represent (in combination with Links) the graphical and modelling topology of networks, the Zones to group factors of activity.

For example, the Terminator–Node–GIS entity is the container of graphical Node information like projection, zoom–level of the co–ordinates associated with Node. The entity Terminator–Node–GIS–Symbol-Shape is a container of lists of co–ordinates describing the shape of a symbol used to display Nodes.

The function of a Node as an element in the topology description of a supply side model makes a Node of type “infrastructure”. If the Node is used as an aggregation container of other entities, then the Node is of type (sub–)Network entity. And one uses this kind of Node to “zoom in” and to “zoom out” of a Node in order to see its internal structure. The concept of “zooming in” is to show further **topological** details (not only graphical details) associated with the Node. The further detail a (sub–)Network can associate with a Node is, that the Node is made up of other entities, e.g. a group of Nodes and Links that describes a railway station, an airport or generally terminals and their access and egress points as well as their “turns” and “changes” between Links (or Links of different modes).

Fundamental Design of GTF Translators

Requirements

From the description of the requirements of the system follows that modelling-data needs to be transferred across different platforms, mainly Windows and UNIX platforms. This is because many modelling software applications are implemented on UNIX platforms and the default platform for users is (usually) a PC.
The structural system requirements [MKMETRIC/MESUDEMO99] are depicted in *Fig. 13.*

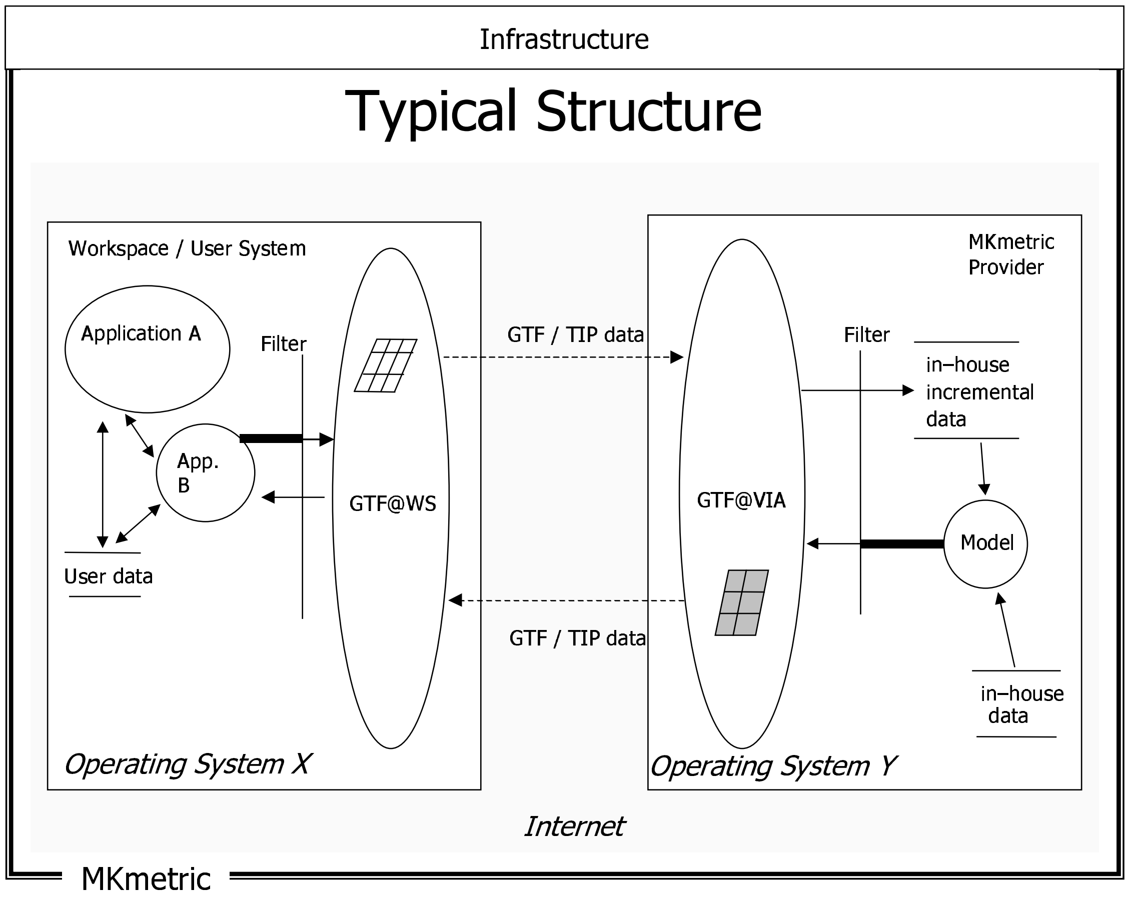

Fig. 13. ***Typical Exchange Structure***

A user modifies his local “User data” through his system. He then formulates a request for a model. The user specifies the model to use. The user also specifies the request and the data to be used. A filter is used to make sure that only relevant data (or data not unknown to the model

provider) gets translated by the GTF@VIA Translator (VIA = MKmetric's model package). The resulting GTF file is transferred to the user's account at the model provider's server. There the data from the GTF file is extracted and incorporated as incremental data into the data already available at the provider's (in–house data) site. The complete data is then fed into the chosen model according to the TIP information in the GTF file and the requested computations are done. The requested results are extracted by the filter and translated into GTF by the GTF@VIA Translator. The user's system gets notified that the requested results are ready for download at the provider's site. The user downloads the data. The user can then view the results with his favourite applications.
The consequences for the actual structure of a GTF file are:

Cross-platform / Human-readability

A non–binary code must be used. The choice has fallen to the ASCII code, because this format has the least problems when being exchanged between heterogeneous platforms. ASCII also has the additional effect that a GTF file in ASCII can also be read and understood by a human, in case of problems.

Segmented & Self-describing

As the data and control information to a model needs to be put together by the user's system the exchange format must be very flexible and powerful. The best way to achieve these two goals is to design the format and protocol for interchanging in a structured and segmented file. Like this, the user's system has a "language" to describe the structure and contents of the GTF file.

GTF data model & GTF–XML message specification

Ad-Hoc-Format

A simple ad-hoc format is defined based on XML [MARCHAL98] to be able to describe some examples of how to use the GTF data model.

This is an ad-hoc format, because the development of the GTF specification didn't have the goal of defining a universally accepted computer exchange format for the data model. This ad-hoc format was defined in order to make concrete examples of how to use the data model. It is based on XML, but isn't a validated and formally correct XML format.

The next two lists define 1. the elements of the format 2. the organisational structure. It is assumed that the XML format and syntax are known. (An introduction to XML can be found at [MARCHAL98].)

List of toplevel entities for the GTF-XML:

Block-level Entity (Object)	XML Tag	Attributes	Comment
GTFDB	<GTFDB> </GTFDB>	id, name	Top most element. Hierarchical starting instance of the information network using the other entity instances.
Terminator	<T ></T>	id, name	Default attributes
		start_of, end_of	Ids of the Link entity instance of which this Terminator is the start / end of
		type	value from code list
Link	<L></L>	id, name	Default attributes
		starts_in, ends_in	Ids of the Terminator instances which are the start / end of this Link instance

Block-level Entity (Object)	XML Tag	Attributes	Comment
		defined_by_specification_list	list of Specification entity instance Ids
		allowed_vessels_list	list of Vessel entity instance Ids
		allowed_alternatives_list	list of Alternative entity instance Ids
		type	value from code list
Zone	<Z>	id, name	Default attributes
		time_zone	value from code list
		winter_summer	value from code list
		historical_group	value from code list
		political_group	value from code list
		has_activities_list	list of SpawnFactor entity instance Ids
		localises	list of centroid Node entity instance Ids
		super_zone	Id of the Zone entity instance of which this entity is a part of
		sub_zones	list of Ids of the Zone entity instances which compose this Zone entity instance
		separated_by_barrier_on_the_left	Id of the Barrier entity instance
		separated_by_barrier_on_the_right	Id of the Barrier entity instance
Intersection Node	<N></N>	id, name	Default attributes
		type	value from code list
SpawnFactor	<F></F>	id, name	Default attributes
		in_category	value from code list
		indicator_definition	value from code list
		indicator_name	value from code list or TEXT
		indicator_value	TEXT indicating the value
		statistical_source	value from code list or TEXT
		is_active_in	Id of Zone entity instance
		type	value from code list
Specification	<SP></SP>	id, name	Default attributes
		regulates	Id of Zone entity instance
		definition_units_list	list of Unit entity instance Ids
		definition_metas_list	list of Meta entity instance Ids
		definition_vessels_list	Id of Vessel entity instance Ids
		type	value from code list
Alternative	<A>	id, name	Default attributes
		definition_services_list	list of Service entity instance Ids
		mode_vessels_list	list of Vessel entity instance Ids
		definition_units_list	list of Unit entity instance Ids
		definition_metas_list	list of Meta entity instance Ids
		type	value from code list
Service	<SE></SE>	id, name	Default attributes
		allowed_on	Id of Link entity instance
		uses_vessels_list	list of Vessel entity instance Ids
		defines	Id of Alternative entity instance
		purpose_list	list of Unit entity instance Ids
		schedule	list of UnitDateSchedule entity instance Ids
		type	value from code list
Vessel	<V></V>	id, name	Default attributes
		allowed_on	Id of Link entity instance
		mode_of	Id of Alternative entity instance
		definition_units_list	list of Unit entity instance Ids
		definition_metas_list	list of Meta entity instance Ids
		type	value from code list
Unit	<U></U>	id, name	Default attributes
		information	TEXT representing the actual data

Block-level Entity (Object)	XML Tag	Attributes	Comment
		for_entity_code	TEXT, name of a class
		for_instance	Id of any entity instance of class “for_entity_code”
		type	value from code list
Meta	<M></M>	id, name	Default attributes
		information	TEXT representing the actual data
		for_entity_code	TEXT, name of a class
		for_instance	Id of any entity instance of class “for_entity_code”
		type	value from code list
Network	<NE>	id, name	Default attributes
		super_network	Id of super Network entity instance Id
		sub_network	list of Network entity instance Ids
		components	list of entity instance Ids
Comment	<C></C>	id, name	A textual comment, about the associated entity instance.
Shape	<SH>	id, name	Default attributes
		SHAPE	SHAPE=shape of region=[rect | circle | poly | default]
		COORDS	COORDS=coordinates of region.
XML Comment	<!-- TEXT -->		Textual XML comment

As can be seen, all the explicit attributes of each class as well as the implicit ones (- which are part of a class due to the relationships of the class)[3] are sub-entities of block-level[4] entities of the XML entity (of the equivalent class).

Organisational Structure of XML entities for the GTF (excerpt):

Entity (Object)	XML Tag	Sub-Tags	Comment
GTFDB	<GTFDB>		block entity, Toplevel entity enclosing all other entity instances of one data base
		<T></T>	
		<L></L>	
		<F></F>	
		<SP></SP>	
		<V></V>	
		<A></A>	
		<SE></SE>	
		<U></U>	
		<M></M>	
	</GTFDB>		
Terminator	<T>		block entity
		<N></N>	A Terminator can be the parent of an intersection Node
		<C></C>	A Terminator can be the parent of a centroid Node
	</T>		
Link	<L>		block entity
		<LI>	

[3] For example: the class Terminator has a relationship „start_of“ / „starts_in“ with Link. This relationship specifies which centroid Node or intersection Node (which are Terminators) is the starting point of a Link. Because of this relationship, each Terminator object (thus each centroid Node or intersection Node object, because these are derived from Terminator) have an attribute pointing to the Link object of which the Terminator is the start of. This specification will refer to these implicit attributes by the name of the relationship, e.g. the relationship „start_of“ / „starts_in“ between Terminator and Link implies an attribute „start_of“ in the Terminator entity and an attribute „starts_in“ in the Link entity.
Inheritance relationships are mapped as sub-tags, e.g. Node is a sub-type of Terminator (Node „inherits“ from Terminator), therefore the Node-tag <N></N> is only allowed as a sub-tag of the Terminator-tag <T></T>.

[4] A block-level entity is an entity that can contain other entities. An inline entity (inline element) is an entity that isn't allowed to contain other entities.

Entity (Object)	XML Tag	Sub-Tags	Comment
		<LF>	
		<LC>	
	</L>		
Zone	<Z>		Inline entity
Intersection Node	<N>		block entity
		<NE>	
		<NI>	
	</N>		
SpawnFactor	<F>		block entity
		<FA>	
		<FAR>	
		<FE>	
		<FG>	
		<FP>	
		<FL>	
	</F>		
Specification	<SP>		block entity
		<SPM>	
		<SPS>	
		<SPT>	
	</SP>		
Alternative	<A>		Inline entity
Service	<SE>		block entity
		<SEF>	
	</SE>		
Vessel	<V>		block entity
		<VA></VA>	
		<VRO></VRO>	
		<VRA></VRA>	
		<VW></VW>	
	</V>		
Unit	<U>		block entity
		<UDI>	
		<UDA>	
		<UP>	
		<UG>	
	</U>		
Meta	<M></M>		block entity
Network	<NE>		inline entity
Comment	<C> TEXT </C>		block entity Text of Comment
Shape	<SH>		Inline entity, all information of this entity are in the entity attributes: id, name SHAPE, COORDS

Example: "Airport Network"

A (simple) Network consisting of an origin Node O, a destination Node D, both linked to an airport Node A:

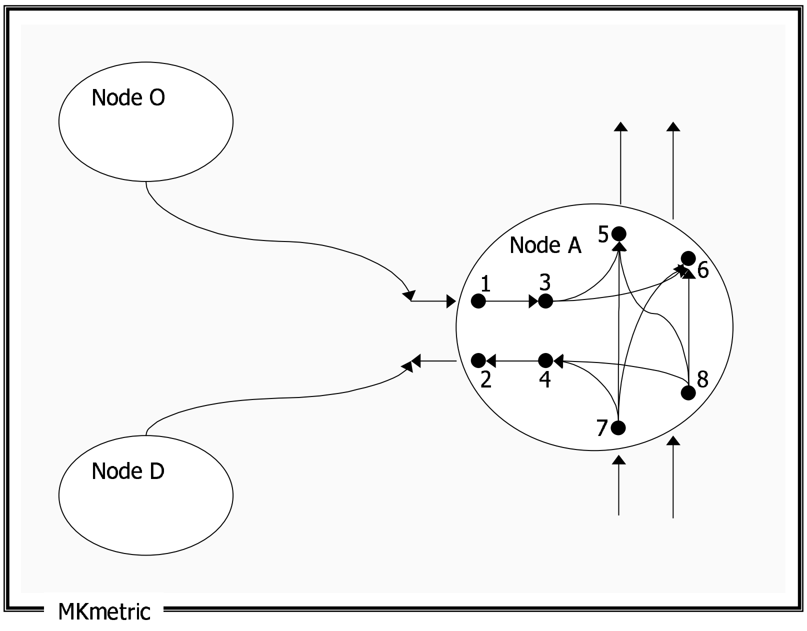

Fig. 14. ***Airport node***

This structure would be represented in a GTF data model using the ad hoc XML format as:

```
<GTFDB id=1 name= "Airport Network Example">
<T id=2 type= "CentroidNode">
            <N id=3 name= "O" type= "1">
            </N>
</T>
<T id=4 type= "CentroidNode">
            <N id=5 name= "D" type= "1">
            </N>
</T>
<!-- definition of the internal Nodes -->
<T id=6 type= "IntersectionNode">
            <N id=7 name= "A1 airport access" type= "1">
            </N>
</T>
<T id=8 type= "IntersectionNode">
            <N id=9 name= "A2 airport egress" type= "1">
            </N>
</T>
<T id=10 type= "IntersectionNode">
            <N id=11 name= "A3 check-in counter" type= "1">
            </N>
</T>
<T id=12 type= "IntersectionNode">
            <N id=13 name= "A4 check-out counter" type= "1">
            </N>
</T>
<T id=14 type= "IntersectionNode">
            <N id=15 name= "A5 departure national" type= "1">
            </N>
</T>
```

```
<T id=16 type= “IntersectionNode”>
            <N id=17 name= “A6 departure international” type= “1”>
            </N>
</T>
<T id=18 type= “IntersectionNode”>
            <N id=19 name= “A7 arrival national” type= “1”>
            </N>
</T>
<T id=20 type= “IntersectionNode”>
            <N id=21 name= “A8 arrival international” type= “1”>
            </N>
</T>
<!-- definition of the link from Node O to Airport A -->
<L id=10000 name= “Route 66 to Airport A” type= “LinkInfrastructure” starts_in= “2” ends_in= “22”>
</L>
<!-- definition of the link from Airport A to Node D -->
<L id=20000 name= “Highway 928” type= “LinkInfrastructure” starts_in= “22” ends_in= “4”>
</L>
<!-- definition of the internal links of Node Airport A-->
<L id=30000 name= “to check-in” starts_in= “6” ends_in= “10”>
</L>
<L id=30001 name= “from check-out” starts_in= “12” ends_in= “8”>
</L>
<L id=30010 name= “to departure national” starts_in= “10” ends_in= “14”>
</L>
<L id=30011 name= “to departure international” starts_in= “10” ends_in= “16”>
</L>
<L id=30020 name= “from arrival national” starts_in= “18” ends_in= “12”>
</L>
<L id=30021 name= “from arrival international” starts_in= “20” ends_in= “12”>
</L>
<L id=30030 name= “from arrival national transfer to departure national” starts_in= “18” ends_in= “14”>
</L>
<L id=30031 name= “from arrival national transfer to departure international” starts_in= “18” ends_in= “16”>
</L>
<L id=30040 name= “from arrival international transfer to departure national” starts_in= “20” ends_in= “14”>
</L>
<L id=30041 name= “from arrival international transfer to departure international” starts_in= “20” ends_in=
“16”>
</L>
<!-- definition of the Airport Node-Network-->
<T id=22 name= “Airport A”>
            <N id=23 type= “5”>
                <NE id=24 name= “Airport A” components=
“6,8,10,12,14,16,18,20,30000,30001,30010,30011,30020,30021,30030,30031,30040,30041”>
            </N>
</T>
<GTFDB>
```

TRANSPORTATION–DATA INTERCHANGE PROTOCOL (TIP)

Section “Generalised Transportation-data Format (GTF)” established the specification for a GTF Data Model. Now a specification concerning the available commands to a user’s workspace is required. These commands will be part of a GTF file and will enable a model provider to process the GTF data file so that the requested answers are computed. This is necessary, because a GTF Data Model alone doesn’t contain any information on what shall be done with the data. This is where TIP is necessary. TIP is a generalisation of "usual" commands (queries) to a transportation model. The development of TIP is based on the classic four step transportation model: generation, distribution, modal split and assignment. Within these four stages, a number of commands (independent of the actual model or the model's philosophy) can be issued to the model in order to produce intermediate data or final model results. These results can then be passed through a filter defined in a TIP command file that is part of a GTF-XML file. The filter extracts the data relevant to the user's query out of the model results and notifies the user's system that the requested results are available for download from the model provider.

Classification of possible queries

The categories of possible (transportation–)information exchange are:

- pricing policies
- regulatory policies
- investment policies
- co–operation of models

The following types (per category) are feasible – also ensuring that meaningful results could be produced:

modification of model input

1. input modification, e.g.
   - proportional modification of a variable's value on a whole network or a specific sub–set (for pricing policies, regulatory policies)
   - modifications of networks (for investment policies)
2. output queries, e.g.
   - modal split effects (e.g. high speed train vs. air; alternative i vs. alternative j)
   - generation and distribution effects (e.g. on airport choice results)

communication between models

1. Model 1→ Model 2: output of Model 1 (e.g. passenger movements or OD–flow matrix) as input to Model 2

2. Model 2 → Model 1: output of Model 2 (e.g. modal split matrix) Model 2 as input to Model 1

With respect to scenario definitions and future projections the described options fit into the following framework (please refer to *Fig. 1*). For each of the components in the last level of the hierarchy two commands must be available.

Input modification

1. explicit change of variable values, e.g. variable X = 100
2. functional change of variable values, e.g. variable X = (variable Y * 2) + variable Z (all mathematical standard operators and functions are allowed for manipulation, e.g. log(), sin(), +,–,*,/, exp() etc.)

Output query

3. output matrix to be calculated, e.g. modal split for all available modes, assigned road network – Germany

4. definition of extracted variables, e.g. modal split of mode road and air, travel–time on link 1152, travel–time of shortest path between zone 51 and zone 894
The variables available in 1., 2. and 4. are the attributes of entity instances defined in the GTF data model.

TIP commands

The commands needed are split into two categories 1. manipulation of variables (selecting & setting / updating) 2. creating, requesting matrices (selecting & calculating). These categories are based on the usual structure of transport models, please refer to *Fig. 2*.

1. selecting & setting / updating

The variables available for manipulation are those defined in the GTF Data Model, e.g. "entity SpawnFactor – Population Class 79 – INSTANCE 34923" for a *single* manipulation or "entity SpawnFactor – Population Class 79" for manipulation of *all* instances. The semantics for the manipulation commands is based on SQL, because the manipulation of GTF variables is a manipulation of relational data. The manipulation commands (i.e. manipulation of model input data) always refer to data already located at the model provider[5]. The commands have the following syntax:

```
UPDATE <entity>.<ALL|SINGLE> SET <variable>=<value>
UPDATE <entity>.<ALL|SINGLE> SET <variable>=<function>
UPDATE <matrix>
```

Where "function" is a mathematical function of any variables in the GTF data already at the model provider. A user can either modify single data elements (e.g. travel–time on link between node 42873 and node 42192 multiplied by 1.2) or lists of data elements (e.g. all the travel–times in a network multiplied by 1.1). <matrix> is one of those specified in the paragraph below.

2. selecting & calculating

The requests for calculation are related to the usual phases that a transportation model comprises: generation (production / attraction), distribution, mode choice (modal split), traffic conversion, route choice and assignment (see *Fig. 2*). The request commands (i.e. model output data) are introduced by the keyword "CREATE"
Command syntax: "CREATE <matrix> {MODE|PURPOSE|SEGMENT|PRODUCT}". Followed by the these keywords

KEYWORD	**Matrix contents**
GENERATION	Matrix: ZONE (x PURPOSE or SEGMENT / PRODUCT) Cell contents: ZONE number of TRIPS or amount of FREIGHT
PRODUCTION / ATTRACTION	Matrix: ZONE x ZONE (x PURPOSE or SEGMENT / PRODUCT) Cell contents: number of TRIPS or amount of FREIGHT
DISTRIBUTION	Matrix: ZONE x ZONE (x PURPOSE or SEGMENT / PRODUCT) Cell contents: number of TRIPS or amount of FREIGHT
MODAL SPLIT	Matrix: ZONE x ZONE x MODE (x PURPOSE or SEGMENT / PRODUCT) Cell contents: amount of FLOW / PERCENTAGE (trips / tons)
TRAFFIC CONVERSION	Matrix: ZONE x ZONE x MODE (x PURPOSE or SEGMENT / PRODUCT) Cell contents: number of VEHICLES
ASSIGNMENT	(loaded) network: NODE x NODE Cell contents: LINK attribute(s)

[5] To manipulate data located in the GTF file doesn't make sense, as the result of the manipulation can be computed beforehand, at the user's site.

The keyword defines which output matrix shall be computed and transmitted (after filtering) back to the user. The specification of MODE, PURPOSE, SEGMENT / PRODUCT is optional. If one is specified it must follow the keywords preceding.
For example: “CREATE DISTRIBUTION BUSINESS”.
The output filter is defined with

```
FILTER <matrix> <variable 1> ... <variable N>
```

The meaning of this line is: ”Filter from the output matrix <matrix> the variables <variable 1> through <variable N>”. Where <variable> is the fully–qualified[6] name of an entity attribute.
TIP commands are defined in the XML segment <TIP></TIP>.

IMPLICATIONS / RAMIFICATIONS OF GTF

The impact of GTF software might have many ensuing commercial and practical ramifications:
1. people dealing with problems appearing in different working areas can exchange information, e.g. analysing side effects when changing from a higher to a lower aggregation level
2. synergetic effects can result from the possibility of transferring knowledge between systems and points of view
3. it will be possible to compare different models’ results (and their quality) as the models can be used on the same data(–base)
4. model users won’t always have to (re–)create their own databases over and over again like in the past, but will have access to standard data(–bases)
5. data–(bases) will gain in quality as time passes, because the data providers will have an incentive to update their databases regularly and properly, since only the “good” databases will be used
6. the fast pace at which telecommunication and telematics are developing in respect to the pace at which models are developing, suggests that within a few years users will be able to use even remote models interactively
7. researchers from different countries can work on the same database and exchange knowledge about their findings
8. users will request new models or combination of models, which previously could have been denied by the consultants, because of lack of transparency on the supply–side of the business
9. the clients / users will have the possibility of choosing and combining models by choosing model outputs to be fed as input to other models to compute further results. This might even be done independently and automatically. Like this the client will not be bound to any specific model but will have the freedom to choose the “best” models for the task / question at hand.
10. all these effects will have a vigorous impact on research in the modelling and other fields

6 A fully-qualified name consists of the name of the entity preceded by the names of all parent entities.

SUMMARY

GTF is an acronym for “Generalised Transportation-data Format” specification. The goal of GTF is to standardise the information used by transportation modelling software for the purpose of electronic data interchange (EDI). The GTF specification uses already defined standards wherever possible in order to maximise acceptance and to minimise redundant work.
To accomplish this the GTF specification comprises the following parts 1. a standardised definition of transport information, but without constraining the possible information to any specific sub-set. This is called the “GTF data model”. See section “The GTF Data Model”. 2. a standardised set of commands to run models and to retrieve relevant data. This is called TIP (“Transportation-data Interchange Protocol”). See section “Transportation–data Interchange Protocol (TIP)” 3. a standard format for arranging data in a file used for EDI and a standard protocol for exchanging the data file. For this XML is used. See section “GTF data model & GTF–XML message specification”. The goal that was defined for GTF was “GTF is for the exchange of information between models and other models & softwares.” it does not entail “... exchange of model databases.” The difference is crucial as the first definition doesn't require the GTF to be in anyway optimised for database handling. GTF is only for a definition of a data model that can be used for a specific database implementation. In contrast, the second definition implies this optimality. What must be clear is that taking a GTF file and putting it into a database such that the access of the data in the database is optimal (i.e. the table definitions are such that, e.g. a minimum of select statements are needed) must be done by a “GTF Translator”. This means that providers adopting GTF need to define 1. their optimal database configuration for their needs 2. to implement a translator that takes the GTF file and converts it into the optimal database tables. Since different people / users have different optimality criteria, GTF defined in this paper is not for optimal database use, but for completeness of the description of the transportation problem domain.

In GTF, the term “data” is simply a fact like “the number 50”, while the term “information” is the association of “data” with a meaning, e.g. “the number 50” and “speed in km/h” which together give “a speed of 50 km/h”. This kind of association between data and meaning is needed to specify exactly what the data signifies. For example “the number 50” when associated to “speed in miles / h” has a completely different sense compared to “speed in km / h”. The former means that “the number 50” is approx. 1,6 times larger than in the latter case. In general, “information” is data with a meaning. By associating data with meaning, the more general “information” can be described. “Data” is fact without exact meaning. Transport models are very sensitive to the information used as input. For example, if a model runs using the information “speed is measured in km/h”, the same model most probably won’t produce valid results, if it is fed with data based on the information “speed is measured in miles / h”. For sophisticated models, it is not enough just to send “data” when exchanging information between models, but the exact meaning must be transmitted, too. To accomplish this, GTF specifies a generalised framework of information in the context of transportation. Within this framework a set of defined pieces of information can be taken and arranged in order to convey the proper meaning for the data being exchanged in EDI, somewhat like building something with LEGO. These pieces of information are contained in “entities” (class, type of information) and “relationships” defined in a so called “data model” by the “GTF specification” which defines the “GTF Data

Model". In order to convey the meaning of a bit of data, different entities can be linked through predefined relationships. The entities and relationships used are then transmitted in an EDI together with the data. GTF is not an imposition of a format, but is a flexible way of describing transportation information.

Once the information is transmitted to a model provider one wants to use the model on the information and one wants to retrieve the results. For this purpose, GTF specifies a set of commands. These aren't related to the workings of any particular model, but are related to the retrieval of information results once the model has computed the input. In the transportation context, models produce results at some stage or the other during the computation. These results can be classified in a standard way using usual requests by clients of model providers. These requests are for example: a flow matrix, modal split, shifts in service levels etc. GTF specifies a set of standard commands relating to such requests, which can be satisfied by most (any) model(s). The set of commands is called TIP (Transportation-data Interchange Protocol). The term "GTF specification" implies both "the GTF data model" and "TIP".

To be able to exchange data electronically, the data must be in a form that can be processed automatically. This is done by specifying the arrangement of the data within an EDI file and the protocol to interpret the sections in an EDI file. Furthermore, to ensure maximum portability across very different hardware and software platforms (e.g. the sender uses UNIX/Solaris and the receiver uses PC/Windows) the transmission files must be in ASCII. This is resolved by using XML and defining a GTF-XML.

The structure of GTF is open and by following the defined rules it can be enriched, detailed and extended in nearly any direction concerning transport. This specification doesn't cover everything in detail, but tests showed that models of urban transport, freight and passenger models, special models for shipping, road specific information on load or damages, schedules as well as indicators or indexes can be handled by GTF. It's a matter of effort to enrich the initial specification developed in [MKMETRIC99].

In the wider context, GTF is for linking (mainly) passenger transportation models to any system. The GTF specification was developed to enable model providers to offer their transportation models' results in a standard fashion. Subsequently, this enables computer systems to present the results in the form a user wishes. Most commonly, users want to view the results using standard applications, e.g. Spreadsheets, Desktop Mapping etc. Therefore, computer systems should also offer links between GTF translators and standard applications that are previously registered in the system. A complete system furthermore should assist the user with the tasks of finding appropriate data and appropriate model providers to answer a user's transportation query. *Fig. 13* shows the typical environment computer structure where GTF is to be used.

DEFINITIONS, ACRONYMS, ABBREVIATIONS AND USED SYMBOLS

Acronyms

GTF	Generalised Transportation-data Format
TIP	Transportation-data Interchange Protocol
EDI	Electronic Data Interchange
UML	Unified Modelling Language
OO	Object-Oriented
ITP	Intraplan
NUTS	Nomenclature of territorial units for statistics

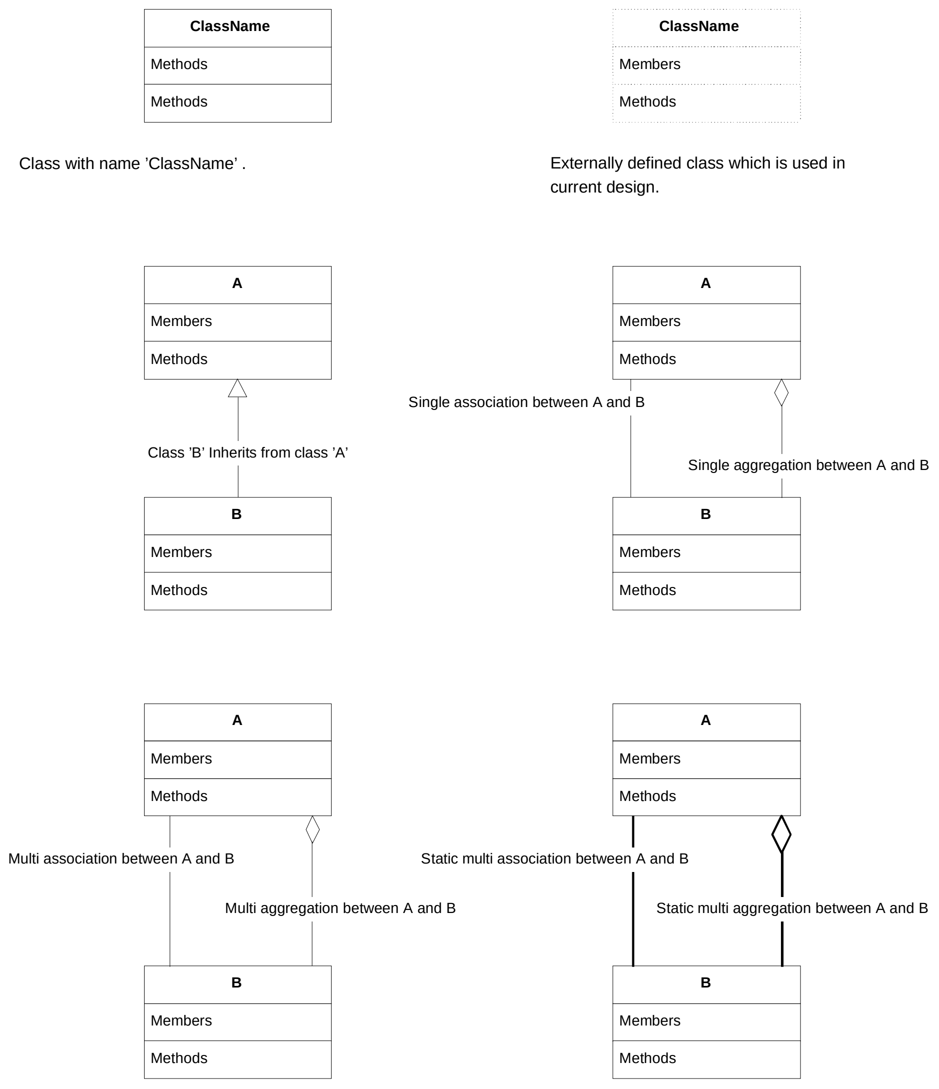
ClassName
Methods
Methods
ClassName
Members
Methods
Class with name 'ClassName' .
Externally defined class which is used in current design.
A
Members
Methods
Class 'B' Inherits from class 'A'
B
Members
Methods
A
Members
Methods
Single association between A and B
Single aggregation between A and B
B
Members
Methods
A
Members
Methods
Multi association between A and B
Multi aggregation between A and B
B
Members
Methods
A
Members
Methods
Static multi association between A and B
Static multi aggregation between A and B
B
Members
Methods

Fig. 15. ***Explanation of the symbols used***

REFERENCES

Literature

Brown, David (1997), “An Introduction to Object-Oriented Analysis: Objects in Plane Language”, John Wiley & Sons
Budd, Timothy (1997), “An Introduction to Object-Oriented Programming”, Addison-Wesley
Button, Kenneth J.(1993), “Transport Economics”, Edward Elgar Publishing Limited
Chen, P.P., “The Entity–Relationship Model — Toward a Unified View of Data”, ACM Trans. on Database Systems, Vol. 1, No. 1, March 1976, pp. 9-36
European Commission - Directorate General Transport (1996) “Transport Research APAS Road Transport VII - 34, Network Architecture”
EUROSTAT (1996) “GESMES 93 –Exchange of Multidimensional Statistical Arrays and Time–series Data. Volume 1: Guidance to Users, Volume 2: Reference Guide”, EUROSTAT, Luxembourg.
EUROSTAT (1994) “Glossary of Transport Statistics”, EUROSTAT, Luxembourg
EUROSTAT (1995) NUTS. EUROSTAT, Luxembourg.
Gamma, Erich et al. (Gang-Of-Four) (1994), “Design Patterns”, Addison-Wesley
Mandel B., Ruffert E. (1999), “GTF Final Report”, MKmetric GmbH
Mandel B., Ruffert E. (1999), “Steps towards the infrastructure of ETIS (European Transport Information System) from a user’s point of view – The way towards an operational ETIS using synergies between 4th FP outcomes”; Mesudemo Workshop 2; Rotterdam; 06/1999, MKmetric GmbH
Martin, Rhiele, Buschmann (1998) “Pattern Languages of Program Design”, Addison Wesley
Marchal, Benoit (1998), “XML by Example”, Que; ISBN: 0789722429
Moellering H & Hogan R (1997) “Spatial database transfer standards 2”, Elsevier Science Ltd., Oxford.
Ortúzar, J. de, Willumsen, L.G. (1990), “Modelling Transport”, John Wiley and Sons
NACE, http://www.cdnet.at/internetpages/cgi/webc.exe/german/nace.htm, ISO 3166 Maintenance Agency
National Institute of Standards and Technology, “NIST (1993). Integration Definition for Information Modelling”, Federal Information Processing Standards Publication 184, NIST Gaithersburg, MD.
Rumbaugh J, Blaha M, Premerlani W, Eddy F & Lorensen W (1991) “Object–Oriented Modelling and Design”, Prentice Hall, New Jersey.
UML, resource (documentation etc.) http://www.rational.com/uml/index.jtmpl

Projects

BRIDGES, “Building Bridges between Digital Transport Databases, GIS Applications and Transport Models to Develop ETIS Software Structure” (contract no. ST-96-AM-1138), on behalf of the Commission of the European Community – DG VII, 1997-1999
Spotlights(TN); “Scientific forum for making advanced transport models fully transparent to end-users, open and more integrated into policy-making”; on behalf of the Commission of the European Communities– DG-Energy and Transport; Actual Cost Contract No.: 1999-TN.10941; 2000-2003