TRANSPORT RESEARCH
FOURTH FRAMEWORK PROGRAMME
STRATEGIC TRANSPORT
DG VII -

Bridges

A middleware technology to develop open multi-software support systems, bridging advanced transport models to GIS and database managers.

Mcrit , National Technical University of Athens NTUA, MKmetric, Danish Technical University DTU, SOFRES, Marcial Echenique & Partners, TRT.

TABLE OF CONTENTS

TABLES

FIGURES

1. Partnership

Full Partners

• Mcrit	Spain	Andreu Ulied
• National Technical University of Athens NTUA	Greece	Dimitris Tsamboulas
• MKmetric	Germany	Benedikt Mandel
• Danish Technical University (DTU)	Denmark	Otto A. Nielsen
• SOFRES	France	Christian Delavelle
• Marcial Echenique and Partners	U.K.	Ian Williams
• Trasporti e Territorio TRT (TRT)	Italy	Angelo Martino

2. Executive Summary

Bridges is a software technology for experts to develop open multi-software support systems, particularly in the field of strategic transport planning. As a “technology”, Bridges includes a number of tools (a “toolbox” so to speak), methods and procedures for using them, and the overall scientific know-how and vision behind them. Resulting from the use of such a technology, several support systems have already being developed and are operational in hands of transport planners and decision-makers at European and local level (e.g. ICONgis for the European Investment Bank, BRAX, SIMU and SIET for different local planning administrations). Even if none of these applications cover the whole range of Bridges capabilities, in total, they demonstrate the usefulness of Bridges research outputs. Bridges was developed within the Strategic Transport element of the 4th Framework Programme, between 1997-1999. Bridges software tools are research outputs, 100% owned by the European Commission.

Bridges research was defined in the context of the ideal user requirements for a European Transport policy Information System (ETIS) and the problems and opportunities presented by already existing or expected software applications, data formats and transport strategic models needed to fulfil the user requirements of an ideal ETIS. The ETIS concept, and its various components, is also being developed as an element of the strategic tasks of the EC's Transport research programme.

In broad terms, user requirements for advanced systems such as ETIS can be summarised as "maximum capabilities with minimum complexity". Since there is no relevant experience of transport software support systems as demanding as ETIS, the starting point for Bridges research was the assumption that the best system architecture to meet the ETIS requirement was a multi-software architecture open to the integration of external advanced support tools ("maximum capabilities") and to be driven from fully personalised and user friendly interfaces ("minimum complexity"). Moreover, this kind of modular architecture is required because there is enough empirical evidence to show that the rapid evolution of Information and Communication Technologies means that only highly decentralised and strongly interconnected systems are flexible enough to be continuously adapted and improved. In the particular case of ETIS, this open system architecture becomes indispensable: new databases and more advanced models are expected to emerge in the next few years, mainly from the European research programmes, for the assessment of current and new policy questions using both main-stream and innovative scientific theories. ETIS users are likely to become more demanding as they gain experience. ETIS development will therefore be an endless “process” rather than a fixed “product”, so that the software architecture supporting it must be flexible enough not to act as a block.

Needless to say, in this inclusive and dynamic system vision, the "bridges" making efficient connections among all system components and, subsequently, between these components and the different users, become crucial elements in making the ensemble behave as an integrated system under user control. Bridges is about designing, developing and testing efficient software solutions to build such software "bridges".

There are no commercial software tools supplying all the "bridges" needed for efficient transport-oriented open multi-software systems[1] like those envisaged, and even the available Windows-compatible tools to link independent applications are sub optimal for many ETIS-related specific purposes. Therefore, Bridges research has developed an entirely new set of tools to fill the most important gaps.

Bridges technology has largely been programmed for a Windows NT environment in Borland C++ 5.02 but with some complementary Visual Basic 5.0 routines. Other programming languages are also used, such as Amzi Prolog (e.g. the Expert System), or component software such as MapObjects (e.g. the GIS_GTF translator, the Expert System) or Microsoft Graph (e.g. to carry on some graphic displays within NISystem).

The main Bridges technology components (the “bridges”) are:

Digital Data Guide (DDG): A directory of available information sources relevant for ETIS

Generalised Transport Format (GTF): A proposed standard data format for transport database exchange, aimed at the transport forecast and evaluation models area.

GTF/Arcinfo Translator (GTF/GIS): An application for transferring data from Arcinfo GIS formats to a GIS version of GTF.

Expert System/Decision Support System (ES/DSS): An application to define rules and criteria to simplify the interface between end users and complex transport models.

NISystem (NIS): A set of routines able to handle advanced transport topologies and carry on graphs analysis.

Communication System (CS): A technology to manage the transmission of commands between independent applications integrated into an open system by the use of multiple customised user interfaces (user work spaces) in an Intranet environment.

Many of these tools have already been successfully tested by developing operational systems, currently in use in a number of transport planning administrations at both local and European levels.

[1] Transport common data formats for example

3. Bridges Vision

The aim of Bridges is to solve as far as possible the lack of compatibility and openness of advanced modelling tools and also between models and other software tools (e.g. those supporting database structures and facilitating interactive viewing of results). Instead of a new transport modelling tool including GIS and database management, the outcome of the Bridges research is a software technology (a "toolbox") "to bridge" models, GIS and database managers in productive and user friendly multi-software support systems. Therefore, Bridges' major goal was to enhance the productivity of information and modelling tools by "bridging" them, including the building of multi-software platforms.

By closing the "software gap" between transport models and other software tools, Bridges technology can help to reduce the more fundamental gap between advanced modelling tools and end users (e.g. experts, decision makers). Systems developed using Bridges tools can put advanced analytic tools in hands of people who are not software specialists; highly customised interfaces can be programmed to give users friendly interaction with advanced tools.

Bridges research is a Windows NT compatible technology, but its Communication System was designed to allow future Internet compatibility and some promising avenues towards such compatibility have already been explored. In particular, a Bridges Server, able to receive remote messages from Internet browsers and interact with Bridges Intranet Communication System is under development as a follow up to the research. The development of such a Server may require modification of some elements of the whole Bridges technology (e.g. the Communication System and NISystem), but the fundamental Intranet design concepts presented in this Final Report will remain as they are.

3.1. From single closed applications to open multi-software support systems

The fact that end users are demanding more and more advanced software is moving the software industry from closed and proprietary tools towards open systems and public formats. No single application can optimally meet all user requirements by itself and therefore open multi-software platforms, making computer aided design (CAD), database managers (DBM), geographic information systems (GIS), statistical applications and other modelling tools work together, are needed to carry out more and more specialised work with higher productivity. Current applications with a limited degree of openness will tend to be replaced by specialised intercommunicating applications and software components, stand alone routines and controls open to data sharing with other applications and to being driven externally by fully customised user interfaces. Needless to say, this emerging trend in the software industry is providing better software tools to develop open multi-software support systems. However this if true only to a certain extent, since most research efforts are nowadays concentrated in key business areas such as electronic commerce or personal communication through Internet, with much less effort to fulfil the specific needs of scientific-oriented systems (those requiring the use of advanced computation tools) or policy-support systems (those making the advanced tools accessible to decision makers).

A support system, or more generally, a Decision Support System (DSS), is a software system under the control of one or many decision makers that assists in decision making. It includes information management, modelling and decision support. Bridges "technology" is a set of communication tools, data models and format protocols needed to develop transport oriented DSSs as open multi-software systems.

The fact that transport models and transport databases are relatively complex and non-standard software tools, with compatibility problems in working with more advanced GIS applications makes the integration of advanced transport models into productive and user friendly support systems an extremely difficult task, almost always requiring specific solutions for each model and each database. These technical problems, as well as other institutional and organisational factors, explains why very few transport administrations have developed such policy support systems despite their clear advantages given the growing complexity of the transport decision making process, especially at European level. While there are commercially available technologies to develop support systems for business management (now linked to electronic commerce) and almost all major companies already have or are presently developing such systems both for management and strategic planning, nevertheless in the policy field, especially in the transport planning sector, specialised technologies are not available and there are only ad hoc, often sub-optimal solutions, unsuitable for general application.

A Software SYSTEM for end users is built by implementing links ("bridges") between MULTIPLE independent software applications. Links ("bridges") between these multiple applications supporting databases and model formulation are needed to make all of them to work together in an integrated and as interactive as possible a manner, using common data formats where this is feasible.

This multi-software system approach of Bridges is becoming ever more necessary given the increasing requirement to deal with more complex questions, which no single database, model or software application can satisfy completely. The first concern of a software technology converting multiple stand-alone applications into an integrated system is, then, improving the productivity (e.g. automating data import and export between applications, reducing running times by interconnecting routines etc.). For this reason, many modellers have already developed their individual ad-hoc solutions to integrating their models with other tools.

A System is OPEN when any additional module or independent external application can be included easily and it is possible for a substitute to be provided for any internal one without changing the overall structure of the system. Openness is a key characteristic in facilitating the evolution and sustainability of the system. Openness requires the existence of standardised formats and communication protocols supporting the links between system modules. This leads to a second reason for developing a Bridges technology, that is the need to increase the capacity for continuous improvement and modernisation of information and modelling systems. There is empiric evidence suggesting that rigid systems and obsolete software solutions are key problems blocking modelling improvement.

An information and modelling system can be considered a SUPPORT system when it is adapted to the solution of end user problems with a "knowledge" interface able to translate information and modelling outputs into meaningful answers for the end user and, at the same time, translate legitimate end user questions into suitable inputs for models. For advanced models, such an intelligent interface has to be developed as an

Expert System; for simplified models, access to policy indicators and database management, it is sufficient to understand end users' analytic and decision processes and design user interfaces adapted to these requirements. While there are many software tools to design and develop such user interfaces, two crucial elements are missing: advanced analytic tools working at the end-user level (to be used also by expert users directly as in the design process) and tools specialised in defining and developing Expert Systems for advanced transport models in a systematic and productive manner.

Finally, an Open Multi-Software Support System can be considered USER FRIENDLY when all modules are driven by fully personalised user interfaces and the whole system is able to engage in an INTERACTIVE dialogue with the user, whenever this is feasible. Within a user friendly system, there may be modules either with their own user friendly interfaces or with no user interface as such. These latter modules remain internal to the system, accessible only to expert users and/or the system developers. There are many multimedia, virtual reality and 3D, desktop mapping and graphic design commercial applications for building such user interfaces. Bridges adds to these standard tools advanced transport modules (NIS), some of which can be run by end users themselves. Systems developed using Bridges can be simultaneously driven by multiple user interfaces which govern a number of applications and databases (a user workspace is this group of applications and databases with a user friendly interface).

The final, and probably most important reason for developing Bridges technology is to narrow the gap between end users and advanced assessment tools, converting the latter to actual "learning and communication tools" for the mutual benefit of end users and scientists (modellers, system developers).

There are no commercial software tools for all the "bridges" needed to build up efficient transport-oriented open multi-software systems like the ones envisaged here, and even the available Windows-compatible tools to link independent applications are sub optimal for these specific purposes. Therefore, Bridges research has developed an entirely new set of tools to fill the most important gaps.

3.2. The software components of multi-software support systems

As discussed in the previous section, advanced transport information and modelling systems ideally require the integration of CAD, GIS, DBM, modelling and Desktop Mapping capabilities, knitted together behind a user friendly, fully personalised, interface:

- CAD applications (such as AutoCad or Microstation) facilitate the digitisation of physical networks and reference environmental and geographic features. With additional GIS oriented routines (programmed in AutoLisp or UCM-DML, or more generally in Visual Basic), simple graphic databases can be supported by CAD applications as well as basic graph utilities (such as checking the consistency between nodes, links and polygons)
- More sophisticated GIS applications (such as ArcInfo, Small World, MGE, Geomedia or Geographics) have specialised network modules with many graph oriented routines (albeit incomplete and in many cases sub-optimal). They provide complex geographical routines for projection change and adjustment, raster analysis and even remote sensing, superimposing both vectorial and raster information. Alphanumeric database management and interactivity with external model algorithms and specialised database managers (DBM) are among the main factors arguing for their general use in transport database management. However, because they are unable to handle advanced transport topologies, they are often used simply to view the final modelling results.
- Desktop Mapping applications (such as Mapinfo, ArcView) enjoy basic GIS options and provide easy graphic display of data and results. They tend to be open to external applications, such as Microsoft Excel or Access, where data and basic calculations can be more conveniently handled, or to complementary GIS applications.
- DBS (such as Access, Dbase, Oracle, Informix etc.) are indispensable for the storage and management of increasingly large data sets coming from many different sources. ODBC drivers and SQL language provide interconnectivity options between database managers. Recent Oracle developments allow the expert user to send "spatial queries", so that Oracle already supports advanced GIS capabilities. Other expert solutions for database management could be, for instance, to program specific ad-hoc routines in database management-oriented languages, such as Clipper, using DBF as data format; since DBF can be read efficiently by advanced statistical tools (e.g. SPSS) and programming languages such as Borland C++, a complete, fully personalised and highly productive software architecture for modelling can be developed without any specific database manager application.
- Spreadsheets, statistical and mathematical packages[2] provide useful routines and programming languages for building mathematical models. Models developed with these tools often have good links to database managers and even provide internal graphic applications (including 3D views, dynamic mapping etc.).

[2] For example, Stastica, SPSS, GAMS, Mathematica, Mathlab etc.

- Transport Modelling tools (e.g. SATURN, TransCad, EMME2, Transplan, Trips etc.) internalise a minimum number of CAD and GIS facilities and are becoming more and more open to CAD, GIS and DBS applications. Because of the inherent rigidity of modelling formulations, they are less likely to be used in research. At European scale also, the heterogeneous and incomplete nature of the available databases often requires ad-hoc formulations).
- For the design and development of multimedia user interfaces there are an increasingly number of applications, currently providing Intranet and Internet viewing[3]. On the other hand, relatively advanced programming languages, such as Microsoft Visual Basic or Borland Delphi, are very efficient, integrating text editors (Word etc.), spreadsheets (EXCEL etc.) and database managers (ACCESS, SQL Server etc.) under fully personalised interfaces.
- Since the late 70s there has been growing activity aimed at integrating all these disparate software elements required to meet the specific decisional needs of particular users to convert software systems into functional "decision support systems". To move from information and modelling systems to "support" systems, it is not enough to develop customised user interfaces; first a deep understanding of the decision process is needed to make the interface intelligent so it can translate users' questions into legitimate inputs to predictive models and return information and modelling results as meaningful answers to users. Artificial Intelligence systems, learning-systems and other kinds of tool have been developed over recent decades, often using programming languages such as Prolog instead of commercial packages.

All in all, ongoing software research and development is moving towards greater openness, scalability and user friendliness. Commercial transport models enjoying GIS options are emerging (such as TransCad by Caliper) and GIS applications well linked to sophisticated DBS applications with relatively good data warehouse capabilities (such as Intergraph Geomedia). Database managers (e.g. through the Internet JDBC extension), are in the process of achieving productive data format exchange and, to some extent, intercommunication through the Internet. Electronic commerce through the Internet is inducing software companies to produce new software technologies capable of developing communication systems which can, to some extent, link remote stand alone applications managing different databases.

The vision of Bridges is to bring software innovation to the transport planning field. In particular, bearing in mind the ETIS requirements, the objective is to adapt and/or develop efficient software solutions to build open information, modelling and support systems capable of becoming efficient decision support systems in hands of experts and policy makers, as close a possible to the ideal of “maximum capabilities with minimum complexity”.

[3] For example, Macromedia Director, Microsoft Front Page and Publisher – even Power Point – and many others

3.3. Specific Bridges research areas were:

- "Data links" (to access transport and GIS database formats)
- "Command links" (to drive stand alone applications), integrated into a "Communication System" built on Windows 95'NT and managed by a System Manager on Intranets.
- Tools to build “Expert systems” for advanced transport models, helping to formulate legitimate queries to them and translate their outputs into meaningful policy questions.
- “Core utilities”, currently missing or sub-optimal for transport planning purposes in the software industry, for transport applications, in standard commercial software tools. Basically, they are GIS routines linked to transport data management and modelling. These components make Bridges a very productive software technology for building transport oriented multi-software support systems. Core utilities are encapsulated into stand-alone components that are linked to any system similarly to commercial applications.
- Customised “user friendly and intelligent interfaces”, helping multiple users to interact with the system according to personalised sub-systems or user workspaces.
- A Directory of European Transport Data Sources

The succeeding chapters will present each Bridges tool in depth, as well as the research process leading to them.

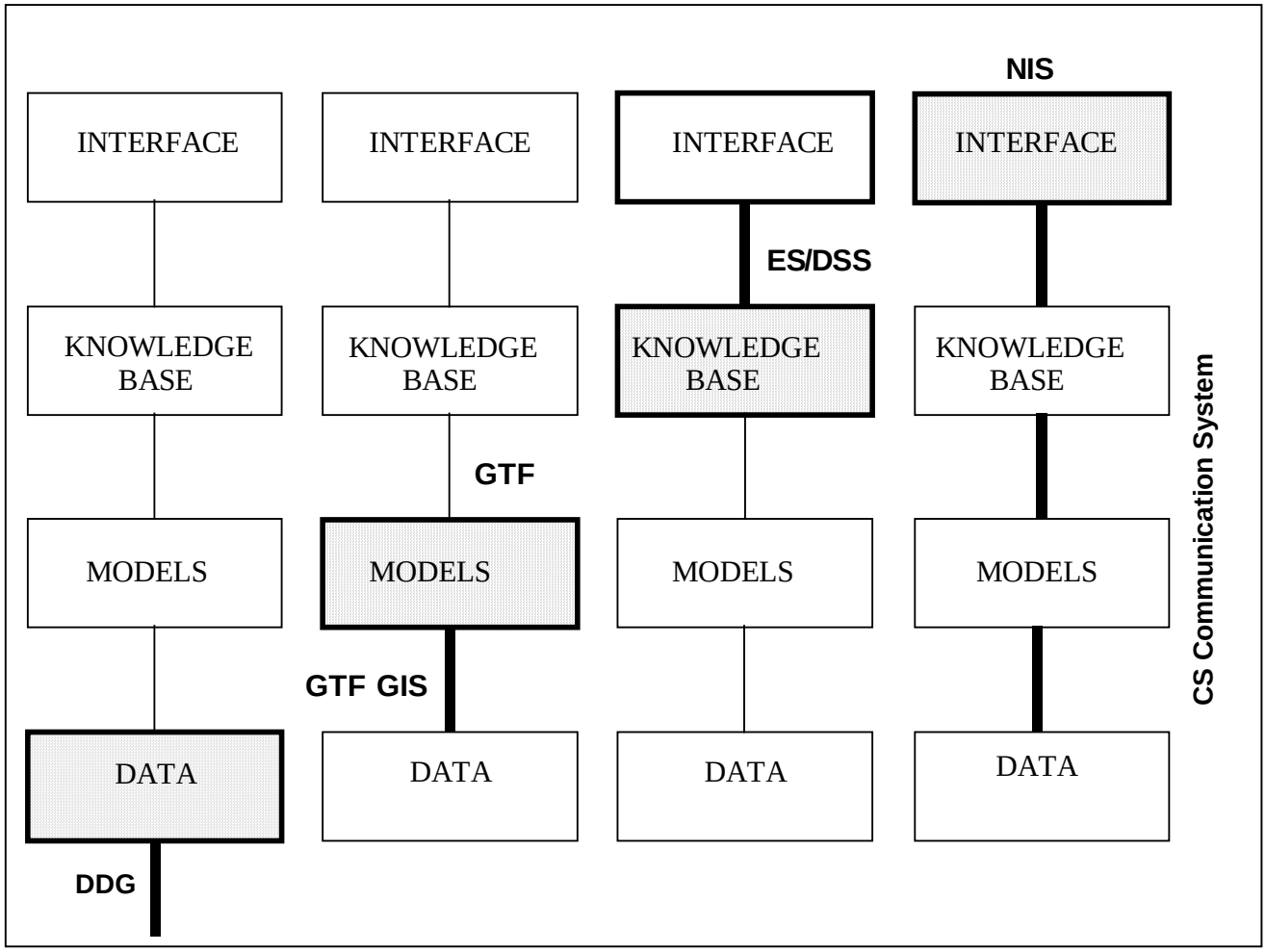
4. Bridges: A set of interconnected tools

4.1. A prototyping development process

Bridges research followed a highly interconnected and de-centralised development process, based on continuous feedback. As a result, all Bridges elements share fundamental concepts[4] making them fully compatible with each other.

According to the "Value Stream" approach adopted, each Bridges Work package was led by one research partner who had to produce a specific deliverable defined to fulfil a precise user need, as illustrated in Figure **Error! Unknown switch argument.**. Bridges partners were urged by the coordinator to produce "throw away rapid prototypes" to provide a view of each partner's ideas and thoughts at an early stage of the research. Based on the analysis of these prototypes, and the spontaneous cross-fertilisation provided by the simultaneous viewing of all final deliverables, successive "evolutionary prototypes" were developed. This process allowed all partners to "begin with the end in mind", an essential requirement for highly innovative (so very risky) research projects.

Figure Error! Unknown switch argument. **Bridges Technology for building**



[4] For example, the same object-oriented and modular architecture, the same data structure etc.

transport policy support systems

In more conventional "Stovepipe" management systems, work packages respond to the internal logic of the developer rather than that of the final user (consumer or client), and each one can be divided into different activities with many developers working with complex interrelations and input-output dependencies. While this management structure may be well suited to mature areas and mature partnerships (with little risk of failure at intermediate input-output steps), this is not the case for immature technologies (such as software development) and immature partnerships (based on heterogeneous multinational organisations) as in Bridges. For these cases the "value stream" approach adopted is much more suitable. This is illustrated in Table **Error! Unknown switch argument.**.

A Bridges brochure has been produced with a precise and synthetic description of the final research outcomes. This Final Report constitutes another step forward from that brochure in that it provides a clearer picture of what Bridges technology is because it includes the experience of partners developing real support systems using Bridges. Only the personal interaction with policy makers and experts, the practical observation of how much Bridges technology helps to solve their requirements in a productive manner, has helped the Bridges partners to communicate what Bridges technology is all about more efficiently.

4.1.1. Introducing Bridges main components

As most Bridges tools remain at the expert level, they have to be used by system's developers to produce OPEN MULTI-SOFTWARE SUPPORT systems, fully customised to the needs of end users.

Some Bridges components can be used directly by end users because they have friendly interfaces (e.g. the DDG, NISystem routines, GTF-GIS Translator), some tools enjoy friendly interfaces even if they will be used typically by developers (e.g. the ES/DSS) and others have no friendly interface as such (e.g. the Communication System has to be configured by writing ASCII files according to an script language, self-explanatory for developers with experience but cryptic for most end users). A number of systems intended for very different end users has already been developed using Bridges tools for a variety of European and Local Public Administrations[5]. For each one of these systems (see Section 11.2) a specific combination of tools has been applied.

In order to understand the technology, the Bridges tools can be differentiated by the different levels of expertise needed for their use:

- Specialised transport routines and applications, not available or poorly implemented in the market place, which can be used by non-expert users (e.g. DDG, some NIS routines),
- Ready-to-use applications and formats to be used by expert users to link advanced transport models to pre-existing systems (e.g. ES/DSS, GTF Format)
- Other tools to enable multiple software modules to work together in an integrated system (e.g. Communication System). They also allow the organisation and

[5] For example, Ig: Transport Infrastructure Assessment System for the EIB, SIMU: An Environmental System for the Local Council of Barcelona

management of a number of personal user workspaces within the overall system and distributed via a Local Area Network. These are tools for expert users and system managers.

Table Error! Unknown switch argument. **Comparison between the efficiency of two different project management paradigms:**

	Value-stream approach (prototyping)	**Stovepipe management approach**
Overall administration (e.g. reporting on time)	Medium	Good
Quality of intermediate products	Bad	Medium
Quality of final outputs	Good	Medium
Partnership motivation	Good	Bad
Risk of administrative failure	High	Low
Risk of scientific failure	Low	High
Recommended for:	Research of innovative products by immature partnerships composed by knowledgeable and committed people under efficient scientific leadership.	Development of products in well established areas by mature partnerships composed by not necessarily knowledgably nor committed groups, under disciplined administrative co-ordination.
Recommended for:	In the Information and Communication Technologies field (ICT), where prototypes are cheap to produce and disseminate and meaningful feedback is feasible and the scientific environment rapidly changing.	The manufacturing field, where prototypes are expensive, difficult to duplicate and improve and may even be misleading.

It is a goal of Bridges developers to make the use of Bridges tools as easy as possible for all end users, expert users and system developers. As Bridges developers gain more experience in developing systems with Bridges and the software industry continues to provide more advanced software and hardware systems, it is expected that user friendliness will improve.

Bridges components are described below in more detail.

4.2. The Bridges Communication System and System Manager (CS)

The Bridges Communication System harnesses OLE/COM technology to integrate stand alone applications. The Bridges Communication System is based on managing flows of command messages between stand alone applications. Bridges Communication System was designed to work in an Intranet but its decentralised architecture allows the addition of a new bridge to the Internet.

The Bridges Intranet Communication System facilitates simultaneous on-line dialogue with commercial applications[6]. Transport models, even those supported by the few available OLE/COM compatible modelling packages, pose additional problems for "bridging" such as when they are based on independent non-interactive modules or they use incompatible data models. This requires more specific tools for example common data formats, such as Bridges/GTF.

In addition, "bridges" which allow different applications to share the same database were developed. However, the simple import and export of databases is not the main problem to be solved: The main problem is to define a common data model and format protocol for the exchange of complex transport oriented databases, which can grow into a standard for transport modellers.

The Bridges System Manager helps to personalise an open systems architecture to the specific needs of each user connected to it. Because of the modular and scalable character of systems developed by Bridges technology, each user has their own "personal Work space", which may contain different applications and different databases. "Friendliness" is therefore achieved by customising the architecture of the system to the needs of each specific user and problem. The Bridges System Manager allows an unlimited number of users to access the system, each with multiple customised workspaces. The Bridges Manager takes care of user passwords, work space maintenance and confidentiality issues.

Finally, the Bridges Common Interface allows the development of fully personalised menus and options, even beyond the design constraints of Microsoft Windows.

The Bridges Communication System, Management and User interface utilities have been programmed in Borland C++. Looking ahead, major improvements could be achieved by achieving Bridges communication through the Internet. Needless to say, this achievement would be a major step, placing Bridges technology in the forefront of software innovation.

[6] Word, Excel, Access and other database managers, CAD and GIS applications such as Microstation J, MapInfo, ArcView etc.

4.3. Specialised "bridges" to transport models (GTF) and to GIS and Database managers (GTF_GIS)

A GTF (Generalised Transport Format) has been defined as a format to store and exchange any transport oriented database. Data exchange between transport models requires that models share a common data model. The GTF data model was defined to be sufficiently open to cover all strategic models at European scale. Bridges' aim is to propose GTF as standard format to be adopted by EC and international institutions.

The GTF data model is able to deal with the complex graph structures and topologies used by transport models (including demand, supply and specific modelling aspects). Following the IDEFIX approach to data modelling, the following entities were defined: node, link, interchange, route service, zone, main mode, persons/goods, carrier, modal stage, and flow/movement. The GTF data model includes the categorisation of relationships. In addition, the adopted GTF format description is public, compatible with international standards (UN/GESMES) and complementary to other database and GIS standard formats. It includes both topological information related to transport entities and the statistical data attached to them.

GTF has been tested with MKmetric VIA and MEAP MEPLAN modelling tools, and it has been extended to cover GIS in cooperation with the GTF_GIS work. A GTF Translator to and from VIA Model formats to GTF has been programmed, with a separate version in Java to allow Internet accessibility.

The GTF_GIS data model is based on adding transport topology to conventional GIS structures, according to the GTF definition (e.g. routes-route segments, stops. The format description is very intuitive with simple ASCII files.

GTF_GIS entities are compatible to GTF but can enrich public transport entities: links, nodes, turns, routes-route segments and segments, stops, changes, terminals, zones, fictive links, matrices-flows, vessels, units-persons/goods and milestones. Topological information related to entity relationships (e.g. allowed vessels and units on links, routes and flow matrices) is also included. Names of data tables containing statistical data and metadata information are listed within GTF_GIS but the actual data is not included.

Arc/Info, one of the most advanced GIS, is unable to handle the complex network topologies required for transport modelling (e.g. defining centroid’s connectors, public transport routes and services etc.). Because of the high GIS productivity of Arc/Info tool, and its widespread use within European institutions, a specific "bridge" from Arc/Info's encrypted data format to GTF has been developed. The translator converts the Arc/Info GIS format into GTF_GIS by adding transport topology. A GTF_GIS-Arc/Info translator has been programmed based on MapObjects libraries, the only way to overcome the Arc/Info encrypted data format. The unavoidable use of such libraries makes the translator not totally royalties free.

Translators to standard CAD (DGN, DXF), GIS and Desktop mapping (MIF, SHP, GEO) formats have also been programmed. Given the limited GIS capabilities of these tools, they are unable to support advanced transport database structures. The interest of bridging these formats lies in the import of graphs developed on CAD, simple databases on GIS and the export of results to be displayed graphically. These translators have been programmed in Borland C++. Translators to DBS applications have been programmed based on ODBC drivers and Data Access Objects (DAO).

4.4. Bridges Core Utilities (NIS): Bridging database structures attached to transport graphs

Bridges Core Utilities (Network Information System, NISystem utilities) have been developed as external stand alone applications to be linked to any system by the Bridges Communication System, just like any other stand alone application. This guarantees that systems developed by Bridges are fully scalable and independent even from Bridges own Core Utilities. Any Core utility can be removed and/or substituted by others when needed.

The paramount goal of NISystem Core Utilities is to complement transport modelling, GIS and DBS applications with missing utilities, particularly in the context of the ETIS reference requirements. NISystem provide:

- Specialised routines for harmonising heterogeneous transport oriented databases and graphs. Examples are "Graph matching" utilities to "bridge" databases attached to graphs with different link segmentations covering the same zone, "Graph checking" utilities to check topological properties on CAD files, "Graph editing" utilities to modify graphs and add topology to CAD files etc.).
- An original GIS binary format (MGS), based on GTF and GTF_GIS data models has been created to increase the productivity of CAD-GIS core utilities and also to overcome possible confidentiality concerns for data dissemination.
- Graphic management routines for CAD, Desktop mapping and GIS have been programmed and included in the system as "core utilities", mostly to support "bridges" which required more than simple data format exchange (e.g. the above-mentioned Graph matching). Moreover, core utilities facilitate maximum flexibility in customising user interfaces and also make Bridges technology not completely dependent on commercial applications but complementary to them.
- A complete database management application has been programmed in Visual Basic 5.0, allowing an optimum degree of user interaction with large and complex databases as well as enjoying several management utilities. Bridges database manager uses Microsoft MDB as default data format but is open to other formats through standard ODBC drivers and DAO.
- Operational research and statistical algorithms for analysing transport databases have been plugged into the system to provide Bridges users with specialised easy-to-access tools for checking, exploring and exploiting databases.

Core Utilities have been programmed in Borland C++ (CAD, GIS and Operational research algorithms) and Visual Basic 5.0 (Database management, EXCEL macro-language).

NISystem utilities have been encapsulated in a number of modules (e.g. the DATABASE module contains all Visual Basic DAO-based functions to manage databases, as well as Microsoft Graph; VIEW contains all C++ routines to provide GIS visualisation capabilities; GRAPH contains all C++ routines providing edition, quality control and analysis of transport networks as graphs etc.). Each NIS module is an executable file which enjoys its own user-interface. It can be activated in the user-interface of the system similarly to any stand-alone application, simply by placing a button linked to the execution of the module.

4.5. Bridges Expert System: Towards a Decision Support System (ES/DSS)

ES/DDS has been developed and integrated into Bridges' toolbox to provide a tool to build intelligent translators between end users questions and sophisticated transport model outputs. The Bridges Expert System helps users to establish legitimate queries to models and interpret the model results. The ES has been tested by building the expert translators to MKmetric VIA, MEAP MEPLAN, and with the Spata Airport environmental impact model. ES/DSS has been defined and programmed with a medium and long-term ambition: the more sophisticated and numerous are the models "bridged" to ETIS, the more useful the ES/DSS tool will become in providing the expert interface.

ES/DSS architecture is composed of a user-interface (Object Oriented Interface, OOI) and the Expert System (ES). The OOI comprises a set of menus, forms, dialogue boxes etc. which are either pre-set or generated at run-time to facilitate user dialogues. The design part of OOI guides the expert user though the standard options of calling up a DSS module (data dictionary, template parser, expert system etc.) or a utility (Visdata, DSSview etc.).

The main task of the Expert System (ES) is to analyse user queries, decompose them into queries to be passed on to other modules, and to combine results into a meaningful form for the user to understand. In doing so, the ES may apply default values, assumptions and rules to fill in any missing data required for running other modules. ES comprises a Template parser, a Query processor, an ES Shell and a Multicriteria evaluator.

ES/DSS has been programmed in MS Visual Basic 5.0, Visual C++ 5.0, Access 7.0, ESRI's MapObjects 1.2 and Amzi! Prolog 4.0, according to OLE/COM. DSS is an OLE container application, therefore easily "bridged" to the rest of the components of a multi-software system by using the Bridges Communication System.

4.6. Bridges Data Sources Directory: The Bridges Digital Data Sources Guide (DDG)

DDG has being developed as a stand alone application which contains European data sources and database descriptions (based on ETIS definitions). DDG follows a similar approach to the "Directory of Transportation Data Sources" published each year by the US Department of Transportation (Bureau of Transport Statistics).

The rational for including DDG in Bridges Toolbox was the assumption that ETIS (as with any other support system) would be limited to a minimum core of data useful for most studies, with links to additional data source providers as needed. On the other hand it was important to develop a public directory of transport information sources in Europe in the framework of Bridges to demonstrate the value of such a tool in the frame of policy support systems.

Currently, DDG is a data sources directory which contains updated information related to the main European Data providers and Data products, as well as available transport models and software products. Initially, DDG was focused on information databases and sources available in electronic format (CD-ROM) and sources accessible throughout the Internet but it was subsequently enriched by including sources available in conventional paper-form. It was also extended to cover available models/modellers and the major software products/providers.

DDG contains

558 European database products, providing information such as availability, cost, periodicity, legal reference, data etc.,

732 European information providers with contact information and Website summaries,

85 modelling products with information based on previous studies undertaken by DGVII,

76 modelling providers with contact information,

71 software products and

74 software providers have also been included.

DDG has a user friendly interface to define queries and achieve customised answers, as well as to establish direct links to data sources through their Internet Websites. It also allows users to update the directory themselves; therefore, it includes a precise codification of fields and checks to validate any new information introduced. DDG has been programmed in MS Visual Basic 5.0, and information has been obtained from a variety of sources.

5. Common Bridges (CS, Communication System)

5.1. Objective: tools to build open multi-software systems

The software architecture to be adopted for any complex support system (including information and modelling, as well as decision support) should ideally be defined as an "open, decentralised and highly intercommunicated network of autonomous components, most of them able to work on an stand alone basis". Only such open and scalable architectures are able to satisfy both today's requirements and have the potential to evolve over time so as to be adaptable to the changing needs of users and ongoing software and hardware innovations. A complex information and modelling system dependent on a single software application has more problems meeting all user requirements while still being flexible enough to be updated. For these reasons, the software industry is moving towards open and decentralised network like structures, bypassing centralised hierarchical structures[7].

Software providers are increasingly making it possible to drive their applications from external user friendly interfaces. However, because only a part of the functionality provided by each application is actually applied by users, the industry is beginning to develop such applications as independent modules or objects (Component Software) which can be called up from external applications[8].

While the software industry is providing a large number of applications, components, routines etc., options to "bridge" these into integrated multi-software systems make use of Microsoft OLE/COM/ActiveX protocols in the Windows Intranet environment and largely Java-based options for both Intranet and Internet. For large corporate databases there are a number of software solutions available to program communications systems but these options require very intense and expensive programming work and a great deal of software expertise.

Bridges has developed its own royalties free original Communication System in C++ for Intranet (Windows NT) based on OLE/COM/Active X. This technological solution was considered the most suitable for a transport information and modelling system[9]. In principle, the Bridges Communication System was designed to be linked to the Internet as well as interconnecting to other Communication Systems but the implementation of these options requires further research. In Bridges technology no single component is indispensable. Even the Communication System could be substituted although this is not recommended.

The Bridges Communication System is composed of two main modules:

System Manager (SM)

Messaging System (MS)

---

[7] The evolution from single isolated main-frames to the Internet, clearly illustrates this process.

[8] For example, Microsoft Chart is the Microsoft component responsible for generating EXCEL graphics

[9] Or any other advanced scientific field, with very different requirements from those of corporate databases

5.2. The Bridges System Manager

The System Manager (SM) takes care of the development and maintenance of all personal work spaces linked to the system and provides registration, status declaration and control of users' priorities and confidentiality for the applications and databases linked to each personal work space. It also provides help to system developers in modifying the system. A work space is a particular set of applications and databases driven by a personal user interface within a specific multi-software system developed using Bridges Communication System. The Bridges System Manager (SM) controls user work spaces, applications and DBS in an Intranet supported by Windows NT.

The Bridges System Manager has three main characteristics.

- A hierarchical structure. Many databases could be linked to a single work space, which is made available to users according to their status (open, restricted to certain user categories, confidential).
- Within a single work space, information is organised in "Projects". A project is an arbitrary list of DBS (Databases and Tables) and CAD files, attached to common classes of network objects, geographical layers and thematic levels.
- Bridges System Manager is located in the Server of a LAN controlling all common databases and applications, maintaining the registration and status of work spaces and users, and, optionally, taking charge of the development of new work spaces.

The SM works on a Windows NT LAN of multiple users, work spaces, applications and database sets. A hierarchical organisation is established as a first step of management strategy. The LAN will have a Server and a Central SM responsible for maintaining common database sets, applications and work space.

- SM keeps a register of users and categories of users
- SM keeps a register of work spaces and the status of work spaces.
- SM keeps a register of common applications registered for LAN use
- SM registers common and specific database sets, together with their availability for application (depending on formats), work spaces (depending on status) and users (depending on their category)
- The SM facilitates definition (and modification) by the work space developer of the application composing the work space.

5.3. Definition of Bridges Messaging System (MS)

The Messaging System (MS) receives, processes and transfers the "Bridges messages" between applications (stand alone system modules) included in personal work spaces. The Bridges Messaging System consists of the following components:

- A library of Bridges messages, which contain information, data or addresses of memory structures where data is located; they also contain the name of the application sending and the application receiving the message (a reference to this can be found in Bridges User-Guide)
- Messages are stored and transferred as simple "strings" using the "SendMessage (Resize)" function, a Windows API Core function[10]. Bridges Messaging System is the module taking care of the transmission of these messages.

Third, the actual translation between Bridges messages and the specific functions required to drive each commercial application, is carried out by independent Bridges modules (translators) linked to each commercial application. The software provider of each commercial application declares a number of "OLE compatible functions" a developer can invoke from an external application to drive the application; usually only a limited number of functions can be invokes by using the application's programming language.

Two different kinds of Bridges messages have been defined:

> System messages [Sys]: Basically, they are warnings and help messages between the MS and the user interface of a WKS.
>
> Application messages [App]: They are sent by one application to another in the same work space and contain all the information needed for channelling it to its target through the messaging module, to be translated by the translator module and executed by the receiving application.

For example, "OPENGIS (file)" is a standard Bridges message requesting a GIS application to display a file. This message leaves some freedom to the translator of the receiving application: each GIS application will use its own default commands to determine the zoom level of the file it is displaying after receiving OPENGIS (file). Bridges Mapinfo translator (Bridges TTAB) would request Mapinfo to use the latest zoom level defined by the user. OPENGIS (file, zoom, x, y) is a standard message which controls the scale and position of the file ([x,y] are the file's central coordinates). The message "WSELECT (book, sheet, cell)", when it is translated by the EXCEL translator (Bridges TXLS) opens a particular sheet from an EXCEL book and focuses on a particular cell.

There are messages which do not require any additional information: for example, "REGENGIS" regenerates all active views in a GIS application, "RESET_ELEMENT" removes the selection, if any object has been selected. There are messages with information attached, limited to a maximum of five parameters, for example: "MAP_RSELECT (number of fields, name of file, command)". Other messages contain information addresses. The message includes, together with the command and any necessary parameters, the address of the information. The

[10] Application Programming Interface

information is stored on DBF Exchange Files to allow it to be read easily from the C++ based Core applications. DBF default files contain a single table with a predetermined maximum number of variables in columns. The first two columns contain identifiers for the objects, which belong to the same class and therefore share attributes (variables). The number of objects (files) is unlimited. The message contains the address of the DBF file (directory path).

In conclusion, Bridges messages [e.g. OPENGIS (file) or WSELECT (book, sheet)] are sent between Bridges MS and all translators, and the actual translation to the internal functions of each application is the responsibility of each translator (e.g. TXLS for EXCEL, TTAB for Mapinfo etc.). Translators have been implemented for most common CAD, GIS and DBS applications and the methodology can be applied to develop others. For original stand-alone applications (e.g. NISystem modules) the executable files have to contain the translator capabilities.

The Bridges communication approach is, to some extent, similar to the SQL library of messages that different Database manager applications can execute after being translated through ODBC drivers ("translators"): The Bridges Messaging System channels the Bridges messages to the translation module of each application and then the translator "translates" the message to (and from) the specific functions of the application.

In practice, the Messaging Module works as follows: it receives messages from the applications included in the system, stores them in a FIFO queue (if they have no special priority), reads them, identifies the receiver and transfer them to the translator of the receiver application, assuring channel synchronism. Each Translator Module is programmed as an OLE container of its application, so it is able to drive the application to execute the translated message automatically. Both the Translator Module, and the application contained by it, are active simultaneously in the work space and remain in a standby situation, keeping the communication channel open.

As has already been described, the Bridges Messaging System was designed to perform two fundamental roles: Message translation and Message transmission.

- The first role is needed to store Bridges messages and translate them to and from the specific OLE functions of each application.
- The second role is needed to notify to the translators that a given translated message is stored, waiting to be read.

The following sections describe the software solutions adopted to implement each role.

5.3.1. MS First Role: Message storage and translation

Once a given application has to send or receive a Bridges standard message, a translation is needed to (or from) its internal OLE/COM automation functions.

This translation is carried out by a translator, built using Visual Basic, with one translator for each application. The translation of a given Bridges message may require several steps and checks, and therefore the Translator Module is not just the list of Bridges standard messages followed by their OLE translations. Often, the internal macro-language of each application (e.g. Mapbasic for Mapinfo) has to be used also. For example, the TTAB.EXE translator translates standard Bridges

messages to Mapinfo Pro by an extensive use of Mapbasic language, given the relative low level of Mapinfo OLE functions.

To explain how the translators work, consider the translation of a particular standard Bridges message: "MAP_RSELECT (number of fields, name of file, command)".

Four separate commands have been implemented within this message:

1 Show the objects selected,

2 Generate a thematic map considering the first field as a parameter to be viewed,

3 Generate a bar graph (each bar represents one field) and

4 Create new arches and assign information from the transferred file to them.

To translate this message to Mapinfo, many Mapinfo OLE functions (each with several Mapbasic functions as attributes) need to be used. The fourth command (Creation of new arches) requires Mapinfo "DO OLE [...]" function, together with the following MapBasic functions: "CREATE TABLE [...]", "CREATE MAP FOR TABLE [...]", "ADD MAP LAYER [...]", "SET MAP LAYER (table) EDITABLE (on/off)", "CREATE LINE [coordinates origin [x,y], coordinate destination [x,y], pen type]".

In the case of "DO CREATE LINE", coordinates [x,y] from points of origin and destination are obtained by using Mapinfo "EVAL OLE function". Depending on the previous use of the OLE DO function, Mapinfo may consider different queries, which makes the translation a little more complex and requires double checking.

This sample illustrates how in a given OLE functions and the programming language of each application (e.g. MapBasic) need to be used to translate a common Bridges Message. To summarise, the translator contains the list of Bridges messages and corresponding application messages. The Translator Module of each application included in the work space remains in a RAM standby situation once the work space is opened by the user. When the translator module receives a message from the central work space Messaging Module (see second Role) it processes the message as an event, generates the OLE translation which its contained OLE application executes.

5.3.2. MS Second Role: The Transmission of messages

The message transmission process consists of the following steps:

- The user clicks an available option in the interface which requires the work of one or many applications, and then the MS automatically establishes the communication between these specific applications linked to the work space.
- The Messaging Module receives the click as an event which causes it to write a message in the Translator Module mail box of the application to be activated. The particular message written in the box depends on the command assigned to the "click".
- The Messaging Module sends a warning to the Translator Module of the application (through a "SendMessage" API function) to ask it to read its box
- The Translator Module reads and translates the message to the OLE server application it contains, causing it to execute the command.

- Depending on the specific message involved, the Messaging Module may send it to one or more other applications.
- If needed, the Translator Module writes a message in the Messaging Module box. The Messaging Module will read it immediately it is received.
- The Translator Module sends a message back to the work space Messaging Module telling it "the process has been completed successfully".

The transmission process is actually channelled by the MS as follows:

- The Messaging Module declares to Windows NT its "hwnd" (the window where it receives and send messages).
- The Translator Modules for all the applications registered in the work space declare their "hwnd". Because translators are programmed as "OLE containers" of their respective applications, they will drive their user interface according to the messages they receive (e.g. open the application in a fixed window situation - hidden or visible, maximised, minimised etc.- execute the tasks ordered, close or remain open etc.).
- All hwnd windows (from the MS and translators) are hidden from the screen in order to free the user from viewing information which is of no interest to him, but remain open in a RAM standby situation until the work space is closed

Messages between applications can be sent automatically by a given Translator Module if it has been pre-programmed this way.

The process described above could follow, for example, the following sequence:

- The user selects a range in EXCEL
- The user clicks a button in the work space interface to display EXCEL data as a MapInfo thematic map. The button belongs to Bridges Messaging Module. The click is processed as an event which initiates the following sequence:

  Send the message "XLS_WSELECT (book, sheet)" to the EXCEL Translator, TXLS.

  TXLS writes the selected range of the mentioned book and sheet in an Exchange File

  TXLS sends the message "SYS_NWSELECT (number fields, name of the exchange file)" to Bridges Messaging Module.

  Bridges sends to TTAB the message "GIS_RSELECT (name fields, file, operation code). The operation code depends on the work space configuration (in the work space interface different buttons may be linked to different operations, in this example different types of thematic maps (bars, arches etc.).

  TTAB translates the message into a series of OLE functions (based on the OLE functions MapInfo provides plus the MapInfo language: MapBasic).

  The message is executed: MapInfo displays the required thematic map with all the objects selected on EXCEL

5.4. Software implementation of Bridges Messaging System

Based on the previous definitions, two alternative ways of implementing MS have been simultaneously developed:

- The first one is based on a conventional mailbox system built on the Windows O.S. and managed by Bridges. It is composed of several static memory structures located in disks ("boxes") capable of holding a single message and a dynamic queue in RAM where messages from boxes are transferred immediately and classified according to priority for execution [for example, the message "Stop workspace" will have the maximum priority, while most other messages will follow the FIFO processing].
- The second one is based on the Win32 API Atoms concept, a shared dynamic memory on RAM managed by Windows O.S. itself, which removes the need for static boxes.

Both approaches have been developed and implemented for testing during the research.

The obvious alternative to both approaches would be an OLE application (or several) with infinite loops checking boxes permanently. This would represent a waste of memory and has therefore been rejected. However, in some cases where multiple messages are sent with great frequency, this could be useful and save processing time.

5.4.1. First alternative: Mail Box System on disk managed by Bridges Communication System

This alternative is based on the following modules:

- "WkS Box" is a simple ASCII file where the sender applications write their messages. The WkS Box contains only one message. The sender opens the file (therefore preventing other applications from writing to it) and writes its message. In order to view and read it easily, the WKS Box has been specified as an ASCII file.
- "Wks Queue Box of messages: this is a space reserved in memory, where Bridges Messaging System transfers messages from the WkS Box and where they wait to be processed in FIFO priority ("First In is First Out").
- "Application WkS boxes": There is one box for each application in the work space. Applications read their messages in their WkS boxes.

MS is provided with a timer, an API function which every 250 milliseconds sends a message from the queue. Timer signals are continuously processed as the event "Read Queue" (in FIFO order unless otherwise stated) for Bridges Messaging System. Once the message has been read from the Queue, it is transferred to the Application Box together with a "SendMessage (Resize)" as a warning to activate the application.

For each application, a specific Translator Module has been developed (for example, TXLS for EXCEL, TTAB for MapInfo), which, after being activated by receipt of the "Send Message (Resize)" notification, reads its box, translates the message and executes it according to its own rules, afterwards deleting the message from its box and leaving room for the next message to be written.

According to this first approach, the design of the Messaging System is based therefore on three crucial points:

- The use of "SendMessage ("Resize Window") as API function for warnings. The usefulness of "Resize Window" is that it will be always linked to a request to read a box if the windows are defined as hidden (the user can not resize a hidden window).
- Synchronising the flow of messages through an intermediate queue box. Synchronisation is essential because of Windows NT's capacity to carry out simultaneous processes (applications must be activated and stopped according to the messages they are sending and waiting for). In addition, the communication system has to be fast enough not to delay processes already running and it has to be as simple as possible so as not to interfere or be dependent on the applications (see below).
- The establishment of specific modules to translate Application Boxes messages to the language of each application, as OLE container applications.

5.4.2. Second alternative: Dynamic RAM structures based on "atoms".

An "atom" is a unique 16-bit value that is associated with a constant value string. Using an atom as a reference to a string is similar to using a memory handled to reference a block of shared global data. Just as an application can pass a memory handled to the Win32 API to obtain a pointer to the contents of a block data by using GlobalLock (...), an application may call the Win32 API to obtain the value of a string associated with an atom. The string value that an atom represents is known as its "atom name". The use of atoms to exchange strings between applications is one of the foundations of Dynamic Data Exchange (DDE). Atoms are also used to store variable-length strings in Object Link Embedding (OLE) object descriptors.

In order to use Global Atoms for Data Exchange (in particular, strings representing commands between applications), the Translator Module of the sender application adds a string to the global atom table. Using GlobalAddAtom (), Windows fixes a particular address for the string in the table, which is transferred as a parameter to the Translator Module of a receiver application. The receiver uses Global FindAtom (...) to find the contents of the received data and deletes the atom using GlobalDeleteAtom (...).

The main advantage of atoms is the speed of the process (managed on RAM), its simplicity (because Windows O.S. takes charge of box creation and management) and easy C++ programming. However it may require more sophisticated Visual Basic programming, and reduce the Bridges developers' options to control the priorities and flow of messages.

Bridges Messaging System allows the simultaneous use of both approaches (static boxes and dynamic RAM atoms), and could also permit the option of infinite loops to substitute for notification of messages, if needed.

6. Bridges to Models: Generalised Transport data Format (GTF)

GTF is an acronym for "Generalised Transportation data Format" specification. The goal of GTF is to standardise the information used by transport modelling software for the purpose of electronic data interchange (EDI).

The conceptual structure of Bridges places GTF on the borderline between external models and the system.

In the current situation, there is no homogeneous data format for transport models. The lack of such a standard makes the process of exchanging data between models extremely difficult and time consuming. Even worse, there is a risk of confusion and misunderstanding of both the terminology and the topological structures used by each transport modeller. Therefore, the success of a standard format depends on two elements:

- First, the "data model" behind the format must be complex enough to cover the database structures used by most models and
- Second, the actual format used must be compatible with accepted standards.

Achieving standard data models and formats is the first indispensable step in the process of integrating advanced models into decision support systems.

6.1. GTF Definition

The GTF specification uses previously defined standards wherever possible in order to maximise its acceptability.

To accomplish this, GTF comprises the following parts:

- A standardised definition of transport information but without limiting the information possible to any specific sub-set. This is called the "GTF data model.
- A standardised set of commands to run models and retrieve relevant data. This is called TIP (Transportation data Interchange Protocol). (This was not part of the Bridges project.)
- A standard format for arranging data in a file used for EDI and a standard protocol for exchanging the data file. For this UN/EDIFACT's GESMES message is used.

To be able to exchange data electronically, it must be in a form that can be processed automatically. This is achieved by specifying the arrangement of the data within an EDI file and the protocol for the interpretation of each section 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 use of UN/EDIFACT's GESMES message. UN/EDIFACT is the specification of ASCII-based EDI messages.

The general requirements are the following ones.

6.1.1. Homogeneous Data Model

To make sure that no valuable information is lost during the transmission of data between models/between models and other software, a homogeneous and agreed upon data model needs to be designed. This will enable a non conflicting interpretation of the data between the sender and the receiver, i.e. important information like "waiting time", "storage time", components of travel time, "check-in time", "access time", "egress time", "taxiing time" etc. will be understood exactly in the same way by the sender and the receiver.

6.1.2. Cross Platform/Human Readability

The consequence of the cross platform requirement is that a non-binary code must be used. The ASCII code is used, because this format presents the fewest problems when being exchanged between heterogeneous platforms (for examples see SGML, PDF, RTF). ASCII has the added benefit that a file can be viewed directly using any editor.

6.1.3. Segmented and Self-Describing

As the data and control information of a model needs to be put together by the 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 data interchange in a structured and segmented manner. In this way, the system will have a "language" to describe the structure and contents of the exchange file. Using the building blocks and grammar (defined by the data model) of the GTF specification, the file becomes self-describing to a translator. This again has the added benefit of being readily understandable.

Also, the important requirement, the capability to transfer survey data and not only model specific data, is of relevance to this specification. The solution provided here is that the GTFGESMES definition developed in Bridges is complementary to the standard GESMES message defined by UN/EDIFACT. Therefore, the way to transmit both model information using GTFGESMES and survey data using GESMES is to use at least two messages, one being a GTFGESMES message and the other being a GESMES message. This will be possible, because the GTFGESMES message follows the format specifications defined in GESMES. This provides for a homogeneous EDI message, by enhancing implementations of GTF Translators by the GESMES specification, limiting the purpose of GTFGESMES to the interchange of model data and that of GESMES to the interchange of statistical, time series data. The two messages are therefore conceptually separate but within the same common and homogeneous framework of the UN/EDIFACT message interchange concept.

All this will enable the GTFGESMES message to be submitted to the standardisation board of UN/EDIFACT to include the message in the list of standard UN/EDIFACT messages, like GESMES.

6.2. Data needs for advanced transport models

In order to understand the basic concepts and ideas behind GTF, one has to know about modelling scientific paradigms and their data structure requirements. This section gives a brief overview of these problems and the resulting need for a "Generalised Transportation data Format" for Electronic Data Interchange.

Generally speaking, transport models use the following information items for their computations:

- Zonal data: any kind of zonal description, e.g. socio-economic data, ecological data, zonal boundaries, transport data, indicators, transport matrices etc.
- Network data: data describing 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.
- Geographical data: for viewing purposes the information which needs to be exchanged with GTF should contain information typically needed by GIS, e.g. the underlying projection of the node and its co-ordinates.

With these different kinds of information in mind, a more detailed view of the information/data categories for transport information can be developed.

Models in general, even if they are not, for example, discrete choice models, are very demanding in terms of the amount and quality of input and calibration data.

The main problems with current data and databases at European level are the following:

- Data required by the model, e.g. for estimation, is not available. For example, a pan-European passenger transport model requires homogeneous input data from all countries at the same level of aggregation. This kind of database is not currently available and when data (or information of interest for the model) is found, not only does the format not correspond to the other data but also the information contained in the new data lacks some essential element.
- The specification of the data required by the model does not match what is available and re-specification is not possible. Often, a database that was acquired for a model, holds the data at an aggregation level that cannot be matched to the one needed by the model. For example, if the model requires a NUTS zonal division of the data but the acquired data has a different regionalisation (e.g. CIP-Codes, Telephone Number or Car Registration System), the structure of data, in the first case a NUTS zone and in the second case a zone in another zoning system, do not match exactly, and a transformation of the data from the other system into data fitting the NUTS regionalisation (or vice versa) will be necessary. This could only be done, if further information concerning the amount of the data for a zone from the other zoning system that fits into the equivalent NUTS zone (i.e. the percentage of the data for the zone from the other zoning system that fits into the NUTS zone) is available. This is seldom the case. Indeed, what usually happens is that you need to take different percentages from a set of zones from another zoning system to create an equivalent NUTS zone (or vice versa). This makes the transformation a tedious and error prone task.

- The level of aggregation of the available data does not match the required level. This often means, that the acquired data is aggregated to a higher level than required and cannot be disaggregated to the level needed.

Because of the problems described above, the structure of a "Generalised Transportation data Format" should cover the following aspects:

- Instead of having disparate and manifold software applications and databases, a GTF should contain all necessary elements and provide one single and homogenous data specification and format
- Instead of having incompatible proprietary formats and informational contents, a "Generalised Transportation data Format" (GTF) should be used throughout the whole system, by providing translators to/from the proprietary formats to GTF. GTF consists of a generalised data model (GTF-DM), and a standard exchange format (GTF-GESMES).

6.3. GTF Main concepts and Data Model

With GTF, the structure of the numerous software applications and databases are accessible in a homogeneous and compatible manner. A set of GTF Translators will provide a single access point to all models and data. The problems discussed previously of non-homogeneous software and data/informational structures and definitions is overcome by using the GTF Data Model (GTF-DM) specification to structure and flesh out data bases and for information exchange (by using GTF GESMES). 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.

The main concept for the development of a GTF-DM 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. 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.

The main focus for the development of the GTF specification and subsequently the GTF Translators was: The goals of the GTF research are to define an abstract view of transport model information for the purpose of implementing translator software to exchange data electronically between modelling software and other software (e.g. database systems).

The primary goal therefore was the definition of a data model for transport 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).

Here are introduced in more detail the fundamental information classes that are the foundation of the GTF Data Model. This will describe the general structure from which the GTF Data Model was developed.

The transport data that is covered is primarily that which is used in strategic transport models. Thus, it covers interurban, regional and 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 although GTF can be extended to handle such issues or combined with more specialised data models, e.g. GDF.

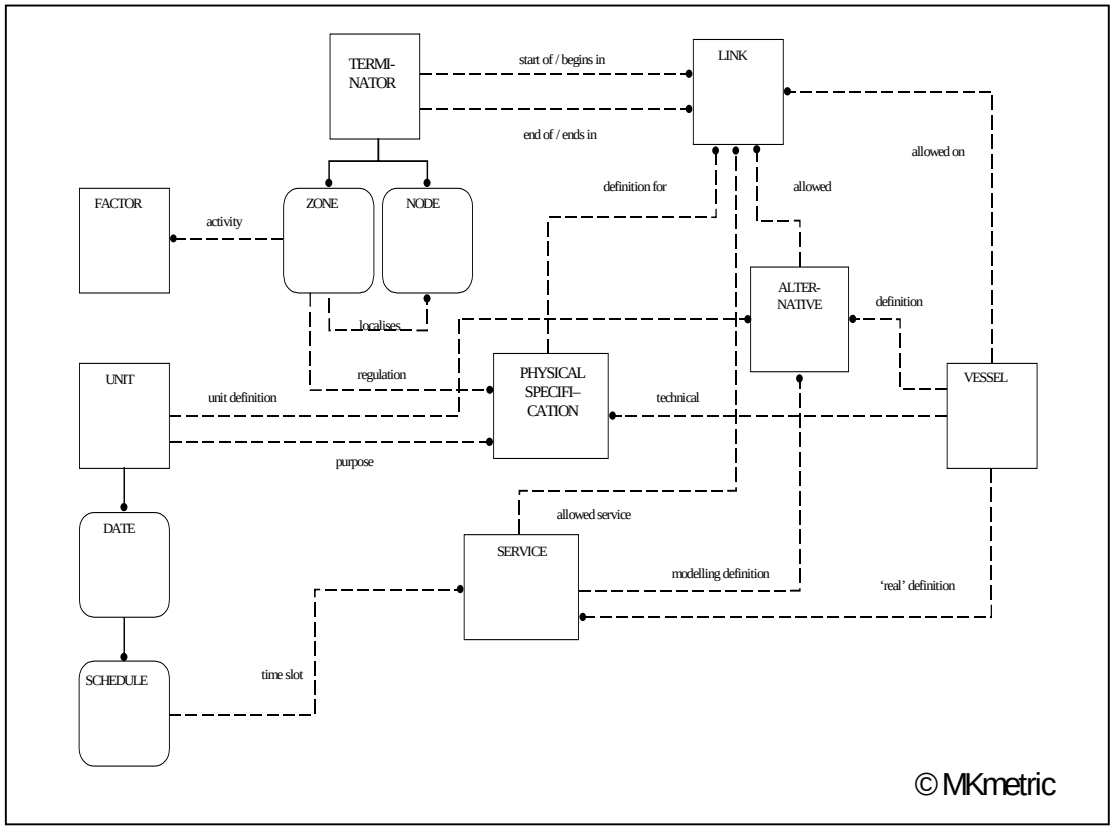
The basic information captured in the GTF-DM is the infrastructure elements of networks: NODE, INFRASTRUCTURE-LINK, and the zonal elements needed for flow assignment: ZONE, FLOW and CONNECTOR-LINK

These types of basic information are further sub-divided in the GTF-DM until the full level of detail required is reached. Also, 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. coordinates, vertices attached to LINKs etc. The "TYPE" part contains the information relevant to models, e.g. INFRASTRUCTURE-LINK and FLOW-LINK.

Many usual elements found in the inputs used by models, e.g. node, link, are generalised in the GTF-DM[11] to provide atomic informational units.

A brief overview covering the GTF entities and relationships is depicted in Figure **Error! Unknown switch argument.**.

Figure Error! Unknown switch argument. **GTF Entities & Relationships Overview**



[11] For example, a LINK is a connection between two TERMINATORS, a TERMINATOR is a starting point or ending point of a LINK, a LINK can be an INFRASTRUCTURE-LINK, a FLOW-LINK or a CONNECTOR-LINK etc.

6.4. Fundamental Design of GTF Translators

From the description of the requirements of the system it follows that modelling data needs to be transferred across different platforms, mainly Windows and UNIX. This is because many modelling software applications are implemented on UNIX platforms and the envisaged default platform for a user is a PC with Windows NT as operating system.

The consequence of the cross-platform requirement is that a non-binary code must be used. ASCII code has been chosen because it is the format which presents the least problems when being exchanged between heterogeneous platforms. ASCII also has the additional benefit that a GTF file in ASCII can also be read using a standard editor, should any problems occur.

As the data and control information of a model needs to be put together by the 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 manner. In this way, the system will have a "language" to describe the structure and contents of the exchange file

With such a language/protocol, the handling of sparse arrays, for example, would become much less complex, as a translator just has to read the positional information attached to a data element of the array in order to assign this data element correctly.

A problem might arise if there is a need to compress the data. As compressed data is usually binary data, most compression utilities are of no use to the GTF translators. Only compression utilities that write ASCII output as a result of compression can be used in order to meet the cross-platform requirement.

6.5. GTF data model & GTF-GESMES message specification

This section briefly introduces the standard format used for the actual exchange of a GTF file. The format is based on UN/EDIFACT's GESMES message.

More information concerning interchange structures to be used with UN/EDIFACT messages is available in the "UN/EDIFACT standard directory (Part 4, Chapter 2.2 to Chapter 6. Structures"). Information can also be found at the UN website: http://www.unece.org/trade/untdid/. The UN/EDIFACT specification defines the allowed character sets to be used in a message transmission. The character set chosen for GTF is the Level A character set defined in the UN/EDIFACT standard directory.

6.5.1. Using the GTF data model and the GTF GESMES format

Basically, the GTF data model is a framework which can be used to define the information that is contained in data. The difference between 'data' and 'information' is crucial. It is clear that 'data' is not always equal to "information". Therefore, 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. In this way, one can make sure that the input data to a transport model fits the model's information requirement. This is crucial for a model to compute valid results. If the input to a model does not fit the assumptions that were made about the information carried in the data (i.e. the meta-data associated to the input data), then the model is unlikely to produce valid results.

Here, the GTF data model comes into play. It forces 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 can thus be used to check whether data fits in with the model's philosophy or not.

The GTF data model provides a standard set of information pieces that can be used for wrapping data into them and combining the pieces of information into a larger chunk of information, just like building something using LEGO[12]. With LEGO one can build many different things without always having to buy different components. One can use the standard set of LEGO pieces and still build a wide range of different things. One only needs a new piece of LEGO, if one wants to build something that does not fit into the concepts of the currently available LEGO pieces. For example, with only square LEGO components it is impossible to build something round. 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. So, one uses a new "round" component 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 piece of LEGO is an entity instance. A construction made of LEGO is a GTFGESMES file with the entity instances and the definitions of the relationships between them. The main advantage of this kind of

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

thinking is the relatively small number of different abstract concepts used to cover a very wide range of concrete objects.

The data model described in this document is complete in the sense that all parent entities required to define a child entity are also defined as separate entities in the model, although the parent entities are abstract and often included to complete the framework. The entities that are actually used in a GTFGESMES transmission have a definition of a GESMES segment following the tabular 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 a "NODE" which is "a point in an infrastructure network". Thus, in a GTFGESMES only NODEs should be transmitted, because the TERMINATOR information is known automatically. This information is implicit in the GTFGESMES file but explicit when looking at the NODE definition in this specification. This means, that when a NODE is transmitted, the receiver (who also knows this specification) automatically knows that a TERMINATOR is the parent of the NODE. One can transmit a TERMINATOR, if needed, but a TERMINATOR is not concrete and can, therefore, be either a NODE or a ZONE. It is not possible to determine which, solely on the basis of the TERMINATOR information. The information whether a TERMINATOR's role is that of a NODE or a ZONE is explicitly contained in the corresponding entities in the data model. Thus, one TERMINATOR can be used as parent of a NODE and a ZONE, if the information to be conveyed is: "This starting/ending point has the role of an infrastructure node and it is also the input/output point of a zone, i.e. the zone's centroid, in this network".

The large number of defined entities (approx. 200) in the data model arises because of a combinatorial explosion. But only 8 basic entities were used to create the data model, namely:

FACTOR	TERMINATOR	LINK	PHYSICAL_ SPECIFIACTON	VESSEL	SERVICE	ALTERNATIVE	UNIT

These are called "topmost entities" or "top-levels". The top-level entities ("top-levels" = all entities in a tree from the topmost entity to a "concrete") are abstract entities and usually only to be used as the beginning of a structural tree. At the end of the trees one can find the concrete entities ("concretes"). The GTF specification defines which combination of entities should be used for actual transmission by defining GESMES Segments, e.g. the NODE's segment specification or the PHYSICAL_SPECIFICATION-TECHNICAL-ENGINEERED's segment specification etc. But one can also use the top-level entities in a transmission but if a transmission contains both "top-levels" and "concretes" then there is redundant information in the transmission, because a "concrete" always implies a parent "top-level". The "top-levels" can be transmitted without breaking any rule in this specification but it would be unnecessary. On the other hand, it is plausible to only transmit "top-levels", if one only wants to define a very abstract network without connection to a concrete transport network. Even the "concretes" have been kept logically abstract enough to be able to define many kinds of networks. For example, one can use the data model to define a network used only by trucks or one only used by cars.

The "top-levels" and the "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. For example, the "is in ZONE.ID" attribute of the NODE entity is an attribute that is migrated to the NODE, because of the "is in ZONE" relationship between NODE and ZONE. This relationship can be used to associate the information within which ZONE a NODE is, e.g. a NODE "Hanover" (where the Expo 2000 will take place) is within the ZONE "Nordrhein-Westfalen". In this data model, a relationship per se does not carry any information in the way an entity does. All information in this data model is enclosed in an entity. The relationships only associate two entities. This association is the only information a relationship adds to the data model. But the actual place where data which describes the relationships (i.e. the two associated entities) is stored is as an entity's attribute. Also, a relationship only defines exactly one attribute. 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. The top most relationships are:

Relationship name	Relationship between
date stamp	DATE X (all other entities)
Activity	ZONE X FACTOR
is in	NODE X ZONE
defined by	ZONE X PHYSICAL SPECIFICATION
technical (specification) by (vessel)	VESSEL X PHYSICAL SPECIFICATION
purpose (defines)	UNIT X PHYSICAL SPECIFICATION
Characteristics	PHYSICAL SPECIFICATION X LINK
allows usage by	VESSEL X LINK
begins in/ends in	TERMINATOR X LINK
time slot	SCHEDULE X SERVICE
'real' definition	VESSEL X SERVICE
allowed service	SERVICE X LINK
modelling information	SERVICE X ALTERNATIVE
Definition	VESSEL X ALTERNATIVE
Allowed	ALTERNATIVE X LINK
unit definition	UNIT X ALTENATIVE

6.6. GTF GESMES Format

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

A GTFGESMES interchange always

- Starts with a mandatory "UNB" segment
- Followed by any number of messages
- And ends with a mandatory "UNZ" segment.

Anything before or after these two segments is ignored by GTF Translators. Between these two segments any number of messages can be defined.

A message

- Starts with a mandatory "UNH" segment
- Followed by an optional "FNT-FTX" segment group,
- Followed by any number of "segment blocks"
- And ends with a mandatory "UNT" segment.

A "segment block" comprises

- An optional "DSI" segment,
- Optional "FTX" segments (up to 5, as defined by the GESMES specification)
- A mandatory "ARR" segment and
- A mandatory "IDE" segment

These are the only segments needed to transmit data structured as specified in this document:

The "UNB" segment is a beginning-of-interchange marker,

The "UNZ" segment is an end-of-interchange marker,

The "UNH" segment is a beginning-of-message marker,

The "UNT" is an end-of-message marker,

The "FNT-FTX" segment group is a group with textual TIP command information,

The "DSI" segment is a dataset identification segment for identifying the data in the subsequent ARR segments assigned by the sender of the interchange, e.g. data/time/project/scenario etc.,

The "FTX" segment is a textual comment to the following ARR segment,

The "ARR" segment is a segment containing the data - the segment's structure is defined in the entity definition,

The "IDE" segment is a structure identifier which identifies the structure used in the previous ARR segments, for example, if the previous ARR segments are NODE definition segments etc.

A transmission, e.g. a file, structured in this way, is a valid GTFGEMSES transmission. A GTF Translator that reads such a file needs to reconstruct the underlying GTF data model filled with the data in the transmission. Once this is done, the complete information is available again at the receiver's side, which was the goal of the GTF specification.

The following GTF message is an example that defines a piece of data according to Infostat's "Area of the zone (km2), NUTS 3" specification. It also defines two ZONEs, "Karlsruhe Stadtkreis" and "Greater London", further information and a flow between the two, LINK-FLOW. This latter is specifically a link with computed attributes in the PHYSICAL-SPECIFICATION entities, that specify the link attribute "distance" to be 1000 km (UNIT-DIMENSION-LENGTH - value = 3).

GTF message:

Transmission:

<connection establishment>

```
UNB+UNOA:2+MKmetric+MCRIT+971126:1510+GTFTIP-TEST-1'
UNH+TEST-MESSAGE-1+GESMES:0:27:M6'
BGM+ZZZ+GTFTIPV1.0'
ARR+5:DE122:Karlsruhe, Stadtkreis'
ARR+6:UK55:Greater London'
IDE+Z07+15' // ZONE
ARR+10:::2:Area of the Zone:5+1:C:50'
ARR+11:::2:Land-use type:5+2:1:17'
IDE+Z07+1' // FACTOR
ARR+7+3:1:10'
ARR+8+2:74:37'
IDE+Z07+136' // UNIT
ARR+7+3'
ARR+8+2'
IDE+Z07+184' // UNIT-DIMENSION
ARR+8+3'
IDE+Z07+192' // UNIT-DIMENSION-LENGTH
ARR+7+2'
IDE+Z07+196' // UNIT-DIMENSION-AREA
ARR+73639++3:2:5:6'
IDE+Z07+118' // LINK
ARR+73639+1'
IDE+Z07+132' // LINK-FLOW
ARR+37+3'
IDE+Z07+40' // PHYSICAL_SPECIFICATON
ARR+37+1'
IDE+Z07+79' // PHYSICAL_SPECIFICATON-MOVEMENT
ARR+37+1000'
IDE+Z07+82' // PHYSICAL_SPECIFICATON-MOVEMENT-COMPUTED
UNT+72+TEST-MESSAGE-1'
UNZ+1+GTFTIP-TEST-1'
```

<connection termination>

For a concrete proof-of-concept, ME&P implemented a translator in Visual Basic and MKmetric implemented a translator in Java. UML was used in the design of the Java translator.

Other formats have been scrutinised, e.g. GDF, XML, ICE etc, to check whether the chosen GESMES format can be replaced by any of these more modern, formats. The examination has shown possibilities which would lead to using one of the other formats instead of GESMES. For example, GDF, which is a very detailed format for capturing road features and which therefore has its own detailed data model, can potentially be used as an addition to GTF. The addition would specify in much more detail the "road" mode of the GTF data model, models normally use aggregated information. But since, GTF is a data model that comprises generic model features, like "mode", it also specifies modes other than "road", e.g. "rail", "waterway", "air". This implies, that for consistency, a mode specific "GDF" would have to be defined, or GTF uses the GDF definition for road and other mode specific definitions for all other modes (which would include by default the GTF definition of each mode). What should be noted however, is the fact that both data models, GTF and GDF, seem to be compatible. Hence, it is only a matter of mapping concepts from one data model to the other.

Another format that was analysed is XML. This specification defines a flexible and extensible format. It does not define any specific data model. Consequently any data model can use XML as a concrete definition of the format to store and retrieve the data in the data model. In the case of GTF this means, that XML can be used instead of GESMES. But one has to note, the disadvantages: XML is not as compact as GESMES, i.e. it uses tags to mark-up ("attach") the meta-data information for a piece of data. Depending on the definition of the mark-up tags, an XML file can be much larger than an equivalent GESMES formatted file. In view, of the fact, that model communication, generically implies a wealth of data to be transmitted, it should be clear, that XML in most cases will result in larger amounts of data. Also the modelling language UML[13], Unified Modelling Language, that has emerged in recent years, was considered as a data model definition notation instead of IDEF1X. At the time it was rejected, because UML did not seem to be stable enough as a data modelling language.

Because of these promising, emerging formats it is advised to follow the development and when necessary/favourable to add another format to possible formats for concrete GTF files.

[13] “Modelling” in this context is not to be confused with transport models/modelling. Transport modelling refers to econometric formulations for forecasting or explaining phenomena. While UML refers to a language and notation initially to define object oriented software systems. The power of UML however has proven adequate for the definition of data models, too. Often also referred to as “modelling” the data relationships and structures.

7. Bridging GIS (GTF_GIS)

7.1. Objective and Conceptual Approach

The objective of "bridging" GIS applications is to integrate specialised tools into transport support systems which can provide both clear graphic viewing of transport databases and spatial analysis.

There is a conceptual gap, however, between the data models used in GIS and Transport databases: Even the network modules of most advanced GIS present incompatibilities and inefficiency in relation to the ideal data structure required by transport databases.

Three work areas to "bridge" GIS applications and database formats were adopted as starting points:

- Developing translators for GIS data formats (such as DXF, DGN, SHP, MIF etc.)
- Developing command exchange links though COM/OLE using BRIDGES/CS
- Developing a new GIS format (GTF_GIS), based on the GTF data model, and its translator to/from ArcInfo[14].

Of these areas of work, great emphasis was laid on the first and third. Command exchange formats have proved not as efficient as expected because commercial GIS software providers do not support sufficiently powerful COM/OLE functions. Only with Microsoft Office 97 applications (WORD, EXCEL, ACCESS, PowerPoint) and to some extent with Mapinfo also have COM/OLE links (a Microsoft technology) proved to be effective.

GIS formats and conventional capabilities do not completely cover Transport information and modelling requirements (e.g. in terms of network topology). Therefore, a special effort to bridge more advanced GIS formats and applications (e.g. ArcInfo coverages) and Transport modelling formats has been made. This effort involves more than a data format translation process: it requires consideration of a transport compatible data model (GTF) and development of mechanisms to implement it -although only partially- in GIS formats. As a result, translators for the import/export of data to/from the most relevant public CAD and GIS formats were developed. The usefulness of these translators and GTF_GIS is not in the actual data translation they provide[15], it is rather in the link they offer between transport and GIS topologies (GTF_GIS) and maximum flexibility and independence from commercial applications for Bridges NIS Core Utilities.

[14] One of the most advanced GIS tools and currently in use by Eurostat and DG TREN

[15] Other software tools, e.g. Intergraph Geomedia, also provide good translation capabilities, including ESRI ArcInfo coverages

7.2. Bridges translators to GIS formats

Bridges translators include a set of C++ routines programmed to import/export databases stored in any public CAD and GIS format (all of them based on ASCII). Translators to the following formats have been developed: DXF, DGN, SHP, MIF, GEO, MGS (and ArcInfo Coverages through MapObjects libraries).

7.2.1. Translator to DXF (Standard CAD from AutoDesk Inc.)

A DXF can be binary or ASCII but more usually it is generated and exchanged as an ASCII file. It is structured in five sections:

5 HEADER provides general information related to the drawing; each parameter has a variable name and an associated value,

6 TABLES defines the elements listed: types of lines (LTYPE), table of levels, table of styles (STYLE), table of views, personal coordinates SCP (UCS), windows configurations (VPORT), dimension style (ACOEST) and identification of applications (APPID),

7 BLOCKS contains the definitions of blocks (set of linked elements) in the drawing,

8 ENTITIES contains all entities in the drawing, including blocks and

9 END OF FILE.

7.2.2. Translator to DGN (from Bentley Inc. Microstation SE)

In October 1990 Intergraph Corporation made its formerly proprietary design file formats common to Microstation and Interactive Graphic Design System (IGDS) available publicly. In particular, the description of binary DGN files was made public. The structure of a DGN is similar to DXF but it is more restricted in the actual organisation of data and therefore simplifies warnings and flags. However, different applications are required for DXF and DGN translators.

7.2.3. Translator to MIF/MID (from MapInfo Inc.)

This is a rather simple format, so simple that with some previous experience on DXF and DGN, no additional information other than an ASCII file is needed to write the translator. For example, instead of codes describing the type of elements it simply uses the full English word (e.g. POLYLINE). Furthermore, the complexity of many CAD elements (blocks etc.) is avoided, and only basic graphic elements are permitted. Therefore it is an intuitive and easily understandable format.

7.2.4. Translator to SHAPE (SHP from ESRI Inc. ArcView)

Shapefiles are also similar to DGN but with some particular features. An ESRI shapefile consists of a main file, an index file, and a dBASE table. The main file is a direct access, variable-record-length file in which each record describes a shape with a list of its vertices. In the index file, each record contains the offset of the corresponding main file record from the beginning of the main file. The dBASE file contains feature attributes with one record per feature. The one-to-one relationship

between geometry and attributes is based on a record number. Attribute records in the dBASE file must be in the same order as records in the main file.

7.2.5. Translator to GEO (from US National Transport Administration)

The GEO format is a very simple format, organised into different files (for transport networks: Link file, Node file, Geography file; for transportation points: Point file, Area file, Geography file). The position of the field name, type, length, beginning and end position and general comment is clearly stated for each file, and therefore it is relatively easy to write a translator.

7.2.6. MGS: Bridges internal binary format

In addition to the previous formats, an internal Bridges format (MGS) has been defined as a binary structure providing more efficient on-line interaction for data storage and screen regeneration. MGS format has proved to be very efficient for displaying graphic data on screen, reaching faster speeds than commercial Desktop mapping tools such as MapInfo and ArcView, and even sophisticated CAD tools such as Microstation SE. Much more important than these advantages is the necessity to have more efficient access from transport modelling algorithms to data attached to transport networks. The running times of traffic network assignment algorithms, for example, rises rapidly if access to data is not extremely efficient.

7.3. Command links based on COM/OLE for GIS

Three different implementations have been developed within Bridges research:

1 Use of COM/OLE to create MapInfo Bridges Command Exchange Translator (MapInfo can be integrated into a Bridges Work Space in the same way as all Microsoft Office 97 applications).

2 Use of MapObjects components for developing ES/DSS own GIS/Mapping capabilities.

3 Use of MapObjects for the translator from ARC/INFO to GTF_GIS).

However, given the level of development of the internal GIS/NIS core utilities (and the limited support for the COM/OLE functions provided by the commercial software companies), this line of research has been limited to these three cases.

7.3.1. COM/OLE for MapInfo

A COM/OLE Container application of MapInfo was developed at early stages of the research (together with EXCEL, WORD and ACCESS) as a part of BRIDGES Communication System.

7.3.2. Use of MapObjects components to support ES/DSS and ARC/INFO translators

As a part of model's Expert System graphic interface (see Chapter 8), MapObjects components have been used. They have proved to be very cost effective in bringing mapping capabilities to user friendly interfaces. When the user interface is connected to ESRI applications (specially ArcInfo) MapObjects components become almost indispensable for easy import facilities from Arcinfo converters. This is the main reason why an ArcInfo GTF translator has been programmed using MapObjects components.

7.4. Linking GIS and Traffic models

The focus of this section is to provide a theoretical discussion of problems and solutions regarding the link between GIS formats and traffic models.

Transport models demand large amounts of data, including:

- Traffic network topology,
- Traffic network data,
- Zone data and
- Trip matrices.

GIS is a natural tool for handling most of these as it can ease processing and improve quality control. However, traffic models demand a complex topology which is not covered well by conventional GIS topology. Thus, in terms of data exchange, there are serious problems when GIS based data are used as input to traffic models and also when trying to use a GIS to illustrate traffic model results.

Despite these difficulties, there are still many advantages from integrating traffic models and GIS, since data handling and the presentation and quality control of results is easier. For this reason, traffic modelling is the field in traffic planning that has made most use of GIS. But, as with data exchange, the poor coverage by GIS of the complex network topology that traffic models demand means that GIS has primarily been used for displaying results from traffic models. As a result, GIS is not used actively as a tool for providing a data foundation for traffic models.

7.4.1. Scope

A major element of Bridges is the integration of the different types of software package used in transport planning and analysis into an open system framework. This section describes the theoretical contributions of ScanRail Consult and the Technical University of Denmark to the integration of GIS with both the Bridges Communication System and to models directly. This is all outlined in Figure 6. Because of its widespread use in the transport field in the Commission and more broadly elsewhere, there is most interest in developing tools for ARC/INFO. Instead of just creating some binary exchange routines between ARC/INFO and the core utilities, a more open exchange format, focusing on the public transport aspects of GIS and transport models has been created. This facilitates easier exchange with other software packages and other types of software.

Apart from the ARC/INFO translators, a set of translators for TPSchedule[16] has been created. This has been done in order to demonstrate the potential for exchanging data between traffic models and GIS as well as to test and validate the implemented exchange routines.

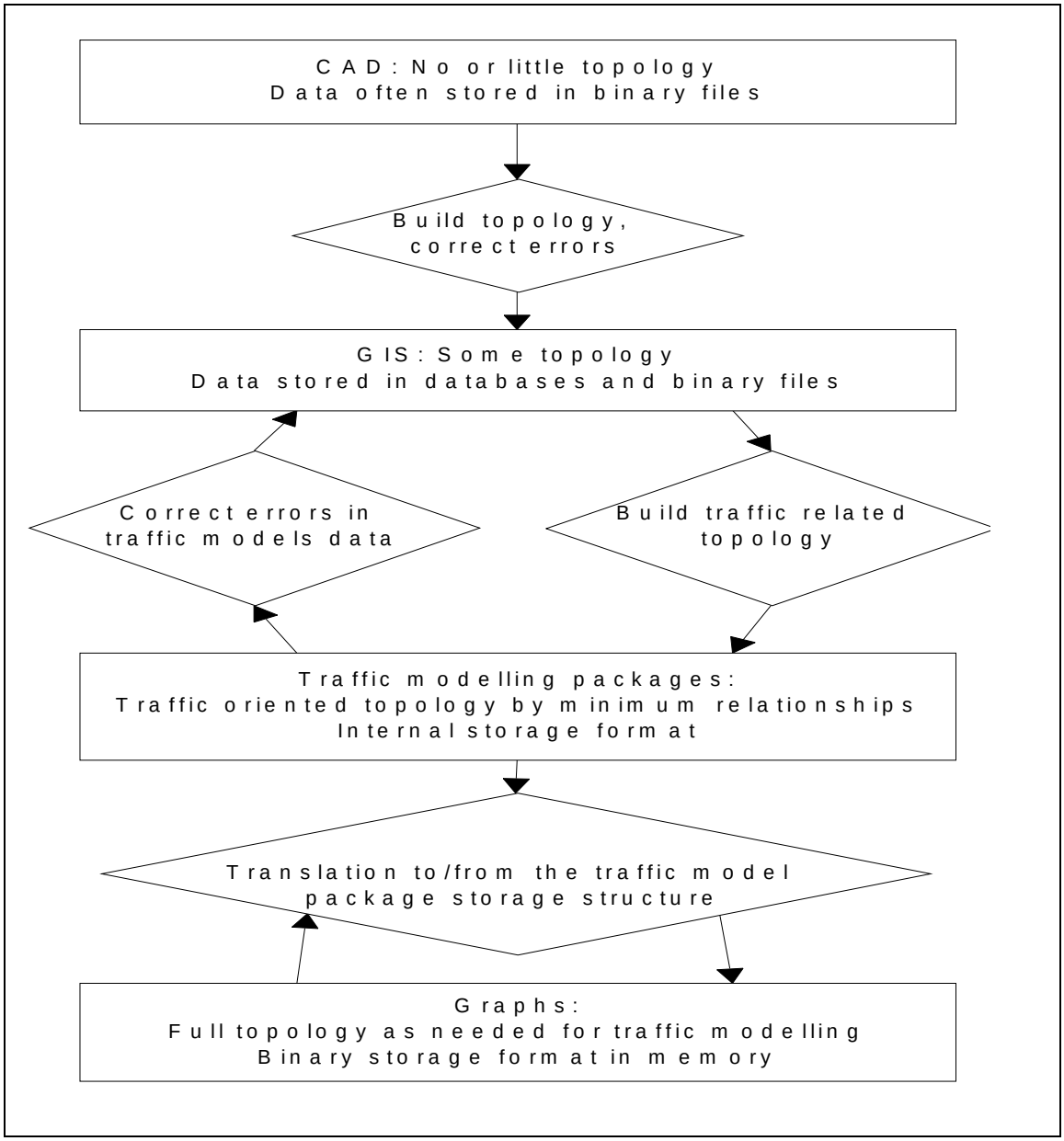
7.4.2. Different types of software - advantages and drawbacks

Different software products for geographic data have different strengths and weaknesses as illustrated in Figure **Error! Unknown switch argument.**. While CAD

[16] A transport modelling package, developed by TetraPlan

is excellent for editing geographic data (e.g. for digitising a map), it can only handle very simple topologic relationships. At the other end of the scale, the graph topologies used within calculation algorithms are often so complicated, that it is almost impossible to maintain them manually. Thus, they are normally built automatically as part of modelling packages.

The trends in GIS developments are for them to become more user friendly and include more CAD like editing facilities. Only very limited attempts have been made in the last 5 years to include more traffic related topologic elements in 'mainstream GIS' (such as trip matrices and transfers). On the other hand, traffic modelling packages seem to include more and more GIS facilities: Not only graphic user interfaces but also real GIS facilities such as in TransCAD. Also the topologic models in traffic models are improving, e.g. MEPLAN which can describe very complex



networks and TRIPS and EMME/2 which can describe a full public transport network.

Figure Error! Unknown switch argument. **Translation processes between different software packages**

7.4.3. Similarities and differences between CAD, GIS and Transport models

Graph theory represents traffic networks as graphs in order to feature an optimal implementation of the all-or-nothing algorithm. However, a similar structure is seldom used in commercial GIS packages. In some GIS and most CAD systems, nodes and links are not connected at all in a topologic sense as outlined in graph theory. In these cases, it is very difficult to implement an efficient all-or-nothing algorithm, since it would demand a new algorithm to interpret the topological relationships from the co-ordinates of the links and nodes. Other GIS maintain a link-node topology (like Arc/Info and TransCAD). Some GIS include procedures to translate the topology to a Forward Star Vector or separate tools to build graph topologies externally to the GIS (such as ESRI's NetEngine). Some GIS or programming tools even include path-finding algorithms as part of their development languages (e.g., Calliper's GISDK or ESRI's Arc/Info and ArcView). However, none of them seem to maintain a full traffic network topology directly as a graph.

7.4.4. Why integrate GIS and traffic model packages?

Most GIS include some path-finding algorithms and must therefore include functions for translation to graphs internally. However, this translation tends to be carried out every time the algorithm is run and it is usually implemented in a non-efficient way for traffic modelling purposes. Consequently, traffic modelling algorithms implemented by GIS macro programming tools usually result in lengthy calculation times. Thus, the easiest approach to using GIS data for traffic modelling is to transmit the network to an external modelling package for the calculations, rather than programming the algorithm with the GIS' own macro tools.

Data exchange causes several problems due to the differences between CAD, GIS and traffic modelling packages storage formats and the mathematical graphs used for the calculations.

7.4.5. Translation between traffic models databases and graphs

The translation between the network representations of traffic modelling packages and graphs are usually carried out automatically when requested. As an example, Trans-CAD can store the network as an efficient graph - a so-called forward star vector file - in order to avoid translation each time the algorithm is run. This file must be rebuilt when the network database is changed within the GIS.

7.4.6. Translation from GIS to traffic models

When using GIS for network maintenance and transport models for calculation, it is often necessary to build a public transport topology in the GIS. In most projects, modellers choose to maintain this topology directly in traffic models or external databases. As traffic-modelling packages have a less user friendly interface, this complicates the process of quality control. The physical network and the organisational network being maintained in different software products is also a

potential source of errors. To avoid this some modellers have recently begun work on a way to maintain the entire topology in a GIS

7.4.7. Translation from CAD to GIS

Finally, one might consider importing data created in CAD systems (e.g. Microstation and AutoCAD) into GIS in order to use such data for traffic modelling purposes. However, as a CAD system does not normally maintain topologic relationships, it demands substantial effort to translate from the CAD format to the GIS format. Although the more advanced GIS such as ARC/INFO and MGE include many functions to assist topology building there is a problem, because some of the attribute data in CAD are often stored as part of the graphics (e.g. layers, colours and line styles reflecting information on road types). Such data are lost in the conversion.

7.5. Practicalities of linking GIS and traffic model packages

7.5.1. Basic Physical Networks

The basic 'building block' in all GIS representations of traffic networks should be the physical network. If this network is not present in the model, the model does not represent geographic information in a GIS sense.

Non-physical objects, such as routes, should be maintained by database tables - not as digitised objects.

7.5.2. Simple Link-Node Networks

A traffic network representation must at the very least include links and nodes. The links only contain information about "totals" e.g. total capacity, both directions included but no information about each individual direction. In the graph, each link should contain information about its ID, the from- and to-nodes, and of course data on the link. Each node should contain information about its ID and data on the node.

7.5.3. Link-Node Networks, With Two Directional Links

A network without link information for each direction is sometimes used in traffic models. This should be avoided in local and regional traffic models, due to the limitations regarding many types of traffic (e.g. traffic in the rush hour and trip-chains) and link attributes (e.g. a different number of lanes in each direction). However, network models that only describe information in each direction do not allow for a modelling of the interaction of the two directions (such as overtaking). Thus, the network should both allow for information on each direction individually and for both together. The topology needed for a traffic network containing two-directional links is identical to the one for one-directional links. The two structures differ only in the associated data. This data should contain separate columns for the forward and backward directions.

7.5.4. Milepost Dependent Data

Data on the traffic network are often stored by mileposts (e.g. an accident at milepost 6,456 or 4-lanes until milepost 14,234 then 2-lanes). This type of data is handled by Dynamic Segmentation in GIS. However, in traffic models such data are usually described by splitting the links at the mileposts concerned. This reflects more closely the structure of the final calculation graph. In order not to make the exchange between GIS and traffic models too complicated, it may be better not to use dynamic segmentation to store link based data in projects that involve traffic modelling.

However, it is noted that dynamic segmentation only containing a sequence of links (no segmentation within a link) can be used as a work-around to describe public transport routes as described in a later section in this chapter. Such a use of dynamic segmentation can also be used to store link data more efficiently and thus to store traffic modelling networks.

7.5.5. Link-Node-Turn Network

A more complete model of physical networks contains an extra topologic element: The Turn. In urban areas, the inclusion of turns is recommended when modelling traffic at an operational or tactical level. Even at the strategic level, it can be necessary to model turns. This is especially the case in networks with congestion problems. In Copenhagen up to 30% of the travel time is spent at intersections. This includes both geometrical delays (deceleration, turn, acceleration) and delays caused by waiting to give way or in queues. At a regional level it is usually not necessary to model turn delays. However, some models describe the time used in passing a city by 'pseudo turns'. This might be more properly described by interchanges.

Figure 8 illustrates the topologic relationships for a traffic network with turns. The turn structure (TURNS) should contain information on which link the turn comes from (FROM-LINK), which it is going to (TO-LINK) and at which node the turn is located (AT-NODE).

Turns are quite easy to describe in a GIS, since they relate to the already described links and nodes. These relations can be described directly in many GIS by using a feature known as 'turntables'.

Some traffic models use more complicated queue models to describe delays at intersections. In this context, it is necessary to have information on the priority of different turn-movements and based on this calculate all points of conflicts between turns. It is noted that this more complicated topology can be interpreted from the simpler data structure above.

7.5.6. Interchanges

In some circumstances, one might want to generalise the topologic element 'turn' to an 'interchange'. The interchange can for example be the movements between different link types, such as from road to rail. Interchanges can in principle be handled in two ways:

1 The interchange is described by turns.

2 The interchange is described as a 'hub' of 'pseudo links and nodes'. Each of the NODES and LINKS can be described as these topologic archetypes.

The first definition is very common for intersections in most GIS and many traffic models but is also used as a work-around to describe stations and airports for example in some GIS representations. 'Turntables' are a simplified version of this definition.

The second definition might be better in cases where the interchanges actually have a physical extent, such as several bus stops within walking distance, or transfers at stations, airports and other big terminals. If the interchanges do not have a substantial physical dimension (compared with the scale of the model), the use of pseudo links and nodes to describe the interchanges is not recommended. If these are used in traffic models, they should be translated into the first definition when imported to a GIS.

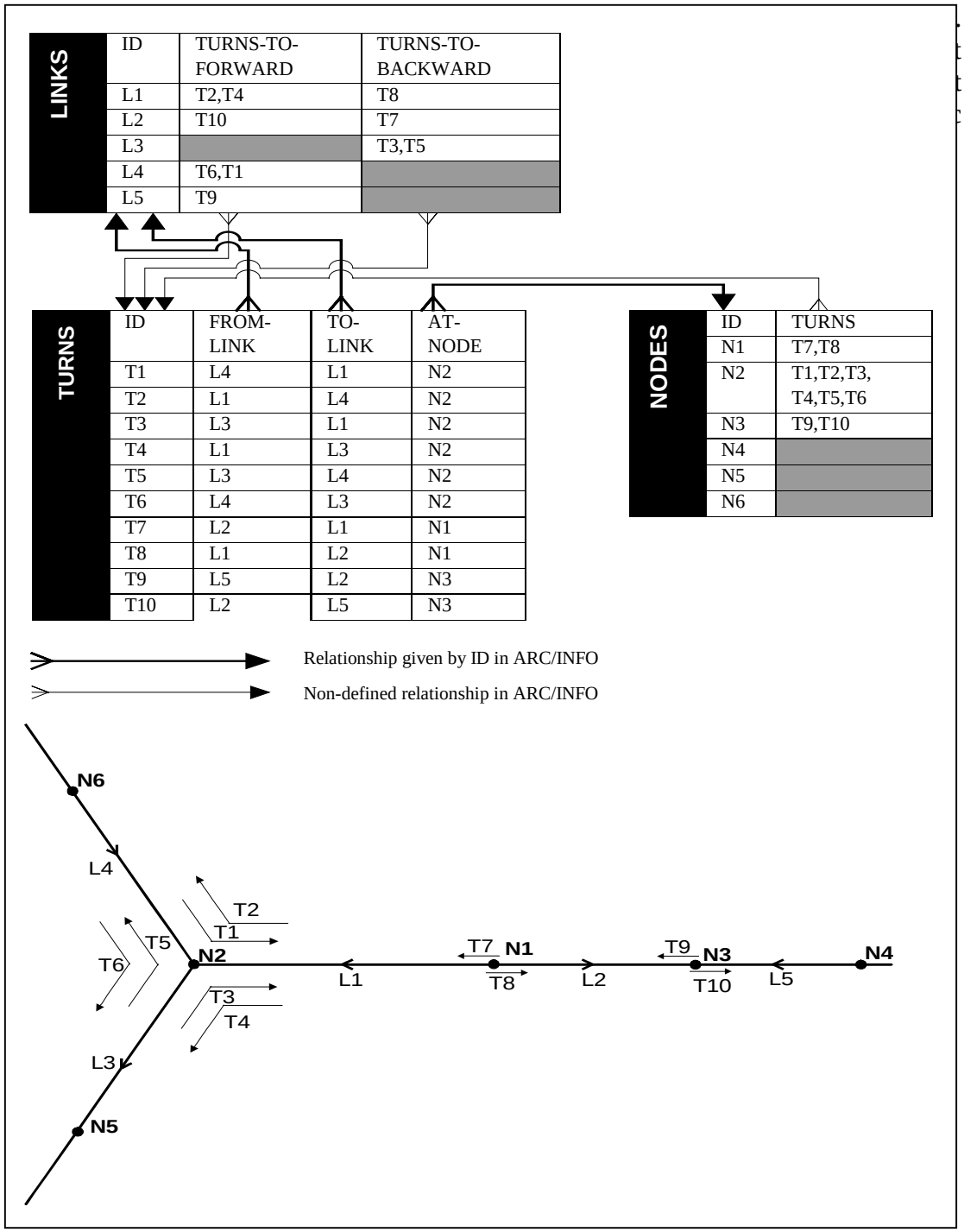
A special data type is the transfer table in TransCAD. This describes transfers from one bus stop to another. Each of the bus stops can be defined for different links as events along different routes (similar to the dynamic segmentation by mainstream GIS). Stops are grouped together by assigning them to a network node. The transfer

table topology is similar to the 'interchange as hub' type, where the links have been simplified to a transfer and the network node to the interchange ID.

7.5.7. Public Transport Networks

This section deals with public transport networks, which describe the organisation of vehicles and passenger flows. Describing public transport networks by 'pseudo links' a and 'pseudo nodes' as in some traffic models is not recommended, because it demands

Figure Error! Unknown switch argument. **Topologic relationships in a network of links, nodes and turns**



Between two bus stops or between two stations, passengers have no points of access to the bus or train. Such a route segment might consist of many links, nodes and turns in the physical network (in contrast to the organisational network, where it is one segment). Public transport might interact with the other modes of traffic at each link along the route segment. Some data might be represented at the link level but not at the route segment level, e.g. travel speeds, capacities and lengths as attributes to the many links along one route segment.

However, from the passengers' point of view there will be an organisational network of route segments, terminals and changes. A change in the topologic sense might be to remain in the bus or train at a bus stop or railway station. Thus, both data on the physical network and on the organisational network are needed.

It is noted, that not all the recommended network descriptions can be implemented in the major GIS'. Thus, work-arounds to reduce this problem are given.

7.5.8. Fundamental network topology

A public transport network topology is best described by ROUTES, CHANGES and TERMINALS referring to the physical network consisting of LINKS, TURNS and NODES. Figure 10 gives an example of the underlying topology.

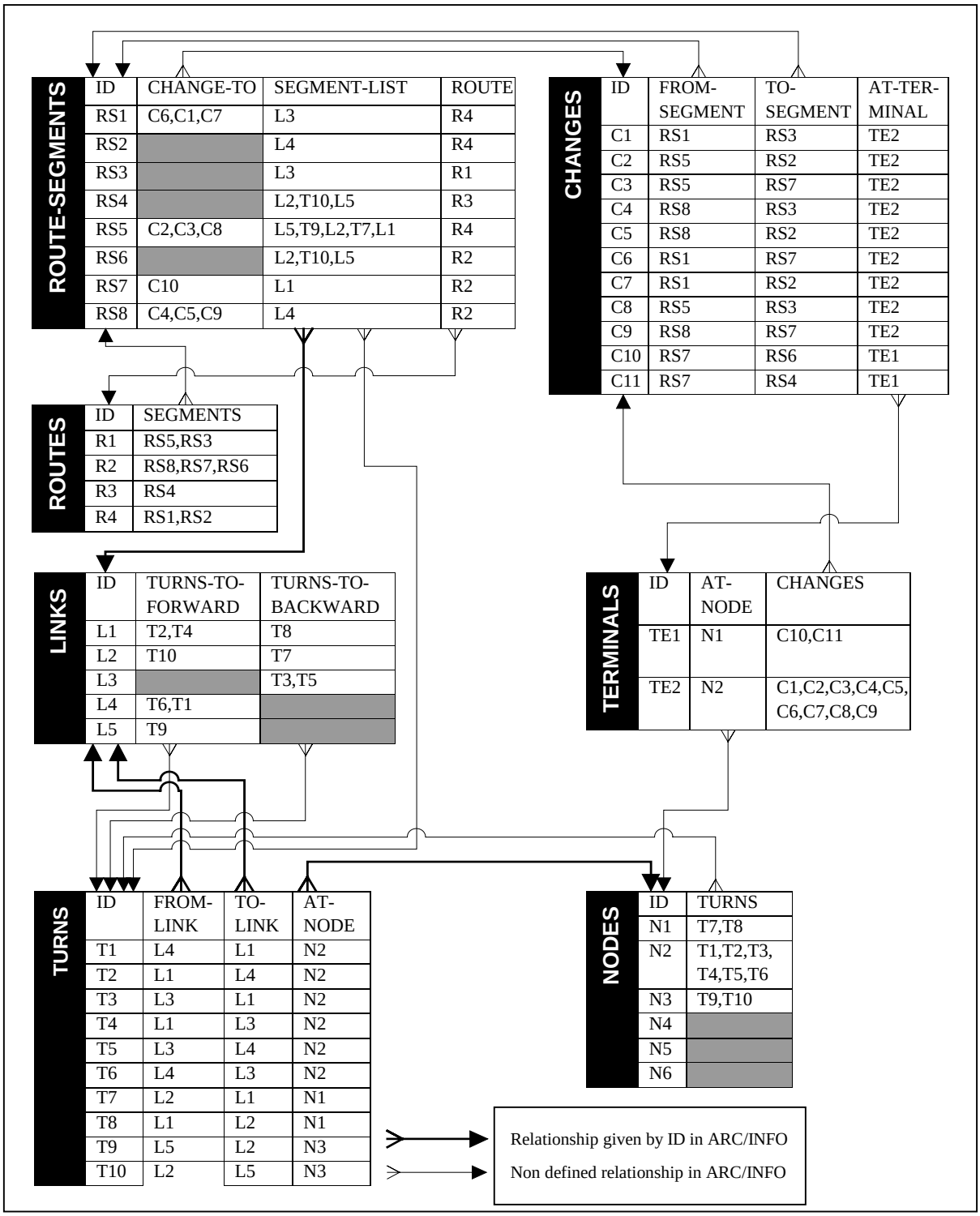
Figure 10 Topologic relationships in a public transport network (b)

The LINKS, TURNS and NODES are organised in the same way as in the car-network. The ROUTE-SEGMENT describes the path going from one terminal to another. Each route segment should contain a ROUTE-ID as well as a ROUTE-SEGMENT-ID. Each link can contain several routes and each terminal can contain several CHANGES between routes. The from-route segment (FROM-SEGMENT), the to-route segment (TO-SEGMENT) and the terminal (AT-TERMINAL) define the changes. There are two main types of CHANGES: To continue at the same route or to change to another route. To continue will typically need information on the waiting time at the terminal. To change to another route will need information on waiting time for the next mode of transport, including walking time within the terminal, discomfort caused by changing etc.

Each TERMINAL is located at a node (AT-NODE). A node can be a terminal for some routes and an ordinary node for others (e.g. through trains, express busses not stopping at local bus stops).

For assignment models, route segments simplify the network as they can be considered to be a link by the all-or-nothing algorithm. However, in large public transport networks with many routes and terminals, the calculation network is often much larger than the car-network for the same area of analysis. It is noted that route segments terminals and changes can be used as a tool for aggregating a traffic network as they also can be interpreted as links, nodes and turns. This can be used to utilise the results from a national traffic model in a regional model. If proper tools are developed (e.g. to thin out the number of shape points), this approach could be an excellent tool for the integration of models at different levels of aggregation. Zone data can similarly be aggregated by using the concept of regions (e.g. as in ARC/INFO).

Figure Error! Unknown switch argument. **Topologic relationships in a public transport network**



7.5.9. Work-arounds to describe Public Transport Networks in GIS

As mentioned, the ideal public transport network topology as discussed earlier cannot be maintained by mainstream GIS (although TransCAD is a big step forward in that

direction). Therefore, different work-arounds are needed to describe public transport networks in GIS.

The simplest approach is to describe the topology by pseudo links and nodes. This approach is very similar to early traffic models, and is straightforward to implement. However, in practice it demands a substantial work effort to enter and maintain the network by 'multiple digitisation' of links. Thus, tools need to be implemented to enter, maintain, aggregate and disaggregate data. Another approach is to describe the routes by digitising each route and describe transfers by turntables. This eases the handling of transfers, since tools are available in most GIS to maintain turntables. For routes, this approach has the same disadvantages as the first.

Describing the network by dynamic segmentation and transfer tables make it easy to aggregate and disaggregate data and easier to exchange the data between GIS and traffic models. Routes are also more easily maintained than by the other two approaches. However, there is no automatic method in mainstream GIS to relate transfer tables to routes. Thus, this relationship must be maintained by some rather complex GIS application work or by importing and exporting the whole network to an external application.

8. Bridging GRAPHS (NISystem Core Utilities)

8.1. Objective: Advanced network oriented utilities

The purpose of developing "core functionalities" or internal utilities, in the Bridges context, was not to contradict the open multi-system strategy by duplicating and replacing external applications (such as commercial Database Managers, GIS, Statistical packages, Spreadsheets, Modelling packages etc.) by core utilities. Instead, the paramount goal was to develop specialised graph management utilities lacking in commercial GIS applications, for example utilities providing graph editing, checking or matching etc. A second objective was to achieve maximum flexibility in customising powerful and intelligent user interfaces, independent from any particular software application. The third objective was to get a number royalties free utilities which could be used for the dissemination of the default options of existing systems to a larger number of potential users (e.g. connecting all PC's of an Intranet or making hundred of copies in CDRom at no extra cost). Finally, ongoing analysis of how to make Bridges technology Internet-compatible largely rely in the capacity of Bridges to send to any web visitors free executable files containing default functionalities and encrypted databases. All this could greatly improve user's interactivity with Bridges systems through the Internet.

As development criteria, five paramount goals were adopted to develop Core functions:

1. Use of mature technology, procedures and algorithms,
2. Focus on specific transport needs, e.g. management of the supply of data attached to graph structures,
3. Compatibility and complementarity with the GTF data model,
4. Emphasis on providing on-line interaction with the user, so developing tools able to be integrated in the user interface level and
5. Use of royalties free software development tools, whenever feasible.

These development criteria resulted in the following software design decisions:

- For Database management functions, easy viewing and common management tools have been developed in Visual Basic using Microsoft ACCESS as the engine. Efficient access to large or remote database sets may require these functions to be supplemented with more sophisticated software, such as SQL Server, ORACLE or others. For example, Bridges GIS functions are supported by specific binary structures. The only exception to the use of royalty’s free tools is the implementation of Microsoft Graph (a Microsoft component able to produce graphics) together with Bridges database routines.
- For CAD functions, no digitalisation procedures were developed but advanced tools, not available in any commercial application known to us, for editing graph entities and checking graph properties were implemented.
- For GIS functions, no raster image processing or automatic geographical projection manipulations have been developed. This simplification of conventional GIS has made the Network specialised tools much more efficient and powerful at managing transport graphs than other commercially available

applications. However, some GIS capabilities, such as geographical analysis and grid structures, have been implemented to enhance graphic analysis related to graphs.

- For Thematic mapping, transport specialised mapping (e.g. mapping of flows) has been developed but conventional mapping facilities are available as well, because of the low additional programming effort required.
- For statistical modelling, only conventional descriptive indicators and basic linear regression tests (e.g. T-student, Kolmogorov) have been developed, supplementing those available in standard spreadsheet applications. Other algorithms (e.g. for optimisation based on Genetic Algorithms, linear programmes etc.) have also been developed.
- For transport modelling, network assignment procedures based on standard methodologies (DUE (Deterministic User Equilibrium), Frank-Wolfe and MSA convergence methods) were implemented, and a large number of alternative path and routing algorithms, as well as other algorithms for network analysis. The structure of these network routines allows linkage of other modelling routines in the future (e.g. for network calibration based on optimisation routines, modal split as part of assignment process etc.)

NISystem utilities are organised into a number of "modules", each one is a stand-alone application enjoying its own user-interface, which can be included in a system like any other stand-alone application. However, each module is modular in itself, so it is relatively easy for Bridges/NIS developers to build new modules by aggregating already available functions or upgrading existing modules by adding new functionalities. The main Bridges/NIS modules are as follows:

- To define "projects" (the data structure of a system: create, remove or modify classes and type of objects)
- GIS/NIS visualisation (to view cartographic elements, assign graphic attributes, manage maps etc.)
- CAD management (to draw and erase elements, adjust projections, carry on geometric analysis etc.)
- Database management
- Graph analysis (to edit graphs, apply quality controls, generate automatically graph elements subject to conditions, carry on analysis such as shortest path algorithms, shortest path algorithms under node and link constraints, assign flows etc.)
- Statistical analyses (to compute basic descriptors, linear regressions, Pearson coefficient and t-student, error tests, Kolmogorov analysis etc.)
- Modelling
- Mathematic algorithms (e.g. optimisation processes such as Genetic Algorithms, mathematic functions etc.)
- Thematic mapping (to provide desktop mapping such as pies, bars, circles and network-related ones, such as flows. etc)

- Decision support (e.g. Multicriteria, decisional-trees, cost-benefit)

Not all NISystem modules are at the same level of development. While some are relatively advanced (e.g. Graph analysis, providing utilities which are sub-optimal or non-existent in commercial packages), others, which provide utilities that can be obtained from existing software or are less relevant in fulfilling the Bridges Vision, are less developed (e.g. Modelling, because advanced models are intended to be developed by external experts and then "bridged" to the system, needing only basic capabilities).

The following sections contain the description of these modules. It is divided into three sections to make the description more synthetic: first the database module is presented in Section 8.2 and then the links it has with other database managers (developed in Visual Basic and based on Microsoft Data Access Objects) in Section 8.3; finally, all other modules are presented in Section 8.4. All of them are developed in Borland C++, and share some common functions and data structures internally.

8.1.1. UTS/GIS as starting point for Bridges: The UTS+ functions

Bridges core functions were partially based on reprogrammed routines and functions from the UTS/GIS System developed by Mcrit (1996) for EC/DGVII. UTS/GIS remains a fruitful inspiration source for the development of Bridges user interfaces since it was a friendly application with powerful GIS visualisation capabilities. Generally speaking, UTS+ DOS functions have been extremely useful for defining and advancing the early stages of Bridges development, in particular in relation to graph management tools; however, there is an essential conceptual difference between the two approaches:

- UTS was a closed C++ stand alone pre-Windows application to provide user friendly GIS viewing of regional impact indicators based on strategic modelling results.
- Bridges is not a single application focused on a particular user but a set of components able to support open systems. Bridges provides links ("bridges") between different software applications (such as database managers and GIS) and transport models, so that they may be assembled and personalised according to any potential user needs.

UTS/GIS, because of its integration of Database management, GIS and modelling options in a user friendly interface, can be characterised as an individual Bridges Work Space with limited analytic capabilities but relatively good viewing facilities.

8.2. NIS Database management module

Bridges database utilities are based on the ACCESS DLL engine. VB5.0 basically provides all the royalties free functions needed to drive the ACCESS engine even if the actual ACCESS application is not installed.

The thirty Bridges Database Core routines, integrated into the DATABASE management module, provide all those capabilities needed to define and manage network oriented databases, in particular transport related ones. These core routines enable the user:

- To define the structure of a database: number and type of classes or entities, name and location of attached files, metadata etc.
- To establish the hierarchy and topological relations between objects from different entities or classes
- To import databases from ACCESS, EXCEL and DBASE with direct DAO links
- To import databases from any DBM application through ODBC drivers (the user has to indicate the ODBC driver needed when declaring all the Databases of the project).
- To view actual data attached to each class in each database and table, in grid form, even from very large databases (a dynamic system for viewing large databases has been implemented to save viewing time)
- To modify the structure of any table by adding or removing variables
- To define personalised queries to any set of tables included in the base of each class of objects
- To personalise formulas and calculations
- To analyse basic statistical indicators (a C++ module was developed including basic statistical descriptors and histograms as well as analysis of variable independence by multiple regression: R^2, t-student, Kolmogorov test to check the normality of error distribution etc.)
- To produce graphics and tables (Microsoft Graph)
- To create new database structures on graph matching options

8.3. Database links

The common goal of Bridges Database DAO/ODBC data links has been the development of efficient import-export links, database viewing and management, capable of working as a stand alone application and open to other DBM applications and Bridges components.

The use of Internet technologies to access any remote database (such as Java database Connectors - JDBC and others) has been tested but remains non-operational. It is however one of the most promising areas in which the research could move ahead. The opportunities to use Internet technologies will be explored in the near future in relation to those Bridges components in which the Internet connection is, in principle, of special interest.

Four work areas were defined:

- Direct links (using Borland C++ routines to read/write in DBASE and ASCII formats).
- Direct links (using DAO on Visual Basic 5.0 routines, to read/write ACCESS, EXCEL, LOTUS and DBASE)
- Indirect links when direct links are not feasible (using ODBC calls, to read/write ORACLE, SQL Server etc.)
- Links to remote databases (using JDBC, IDC etc.)

Beyond these four areas, and closely co-ordinated with them, already existing Bridges Database utilities have been improved and extended. More specifically, the Bridges database engine has been designed as a Visual Basic stand alone application linking bases stored in their original DBASE, ACCESS or EXCEL formats through DAO and all other formats through ODBC (each class of objects has three attached files: a single alphanumeric base with an unlimited number of tables, a "linkage" file with graphic identifications of objects to establish the GIS/NIS relation, and a CAD file containing the graphic elements attached to each object, if any).

8.3.1. Direct links using C++ routines

The main value of using C++ routines is to provide efficient, fast access to large databases. Algorithms and analytical routines included as core utilities, all developed in C++, will use these routines to import and export data from/to databases. Algorithms and analytic routines cannot access large databases through queries to the database manager (in VB5.0) because this would require the transfer of many complex BRIDGES messages between the components involved in the process, causing serious delays when managing very large datasets. On the other hand, C++ routines will be very useful for import/export of ASCII files with transport data structured according to the GTF format.

Most of these routines were developed internally by Bridges partners during previous work and only minor adaptation was needed to integrate them into the Bridges framework.

8.3.2. Direct links using VB 5.0 routines.

Routines to link BRIDGES Core to the following applications have been developed in Visual Basic: ACCESS (.mdb), EXCEL (.xls) and DBASE (.dbf) taking advantage of DAO functions (VB 5.0). BRIDGES Core has thirty specialised VB5.0 functions to facilitate direct links to these three applications and to carry out specialised Database management tasks.

These thirty VB 5.0 functions use the full functionality of Windows Data Access Object (DAO). DAO is composed of a set of Microsoft royalties free Dynamic Linked Libraries (.DLL), for example MSEXCL35.DLL for EXCEL or MSXBSE35.DLL for DBASE. Therefore, all DLLs used to establish direct links between Bridges Core and DB applications will be included together with the remaining Bridges components when Bridges System is disseminated (DAO library files are usually installed together with the Visual Basic language compiler; users without this compiler installed will need to receive these specialised DAO DLLs with the remaining BRIDGES software). In addition to the already mentioned DLLs, other more general DLLs must be included within the BRIDGES software package.

It is important to note that DAO functions allow viewing and management of the data stored on the commercial DBM applications listed. They maintain active links to the server applications where the data is actually stored. Therefore, any Bridges user modification in the database will be automatically updated by the server DBM applications Some DBM applications may constrain the user's capacity to modify the data, even if Bridges provides the capacity to do so. For these cases, data has to be imported to ACCESS using BRIDGES interface options before it is managed (subsequently, the modified database could be exported to the original DBM applications if necessary).

DAO provides functions to link more powerful Database Managers, such as ORACLE or SQL Server, only through ODBC. On the other hand, Borland C++ functions can only directly read ASCII and Binary fields. Difficult programming work is required to define import and export functions in Database manager formats. VDBT (Visual Database Tools) was tested to establish links to Paradox and DBASE but MCRIT's own C++ lower-level libraries were faster and more efficient for BRIDGES purposes. MCRIT C++ routines manage DBASE files (the closest format to ASCII), and this format has been used as default exchange format between BRIDGES components; however, the greater productivity and ease of VB 5.0 in creating Data links through DAO strongly recommended its use instead of Borland C++ or even Microsoft Visual C++ (which uses DAO as VB 5.0).

8.3.3. Object Database Connectors (ODBC)

ODBC is composed of a set of drivers (one for each Database Manager application) which facilitate the transfer of data between DBM applications (import/export options according to predetermined arbitrary user queries). Each ODBC driver is a DLL developed by the software provider of the DBM applications. Microsoft Windows includes all drivers together under a single user interface which allows any user to meet all the parameters that each driver requires: Each call to a particular file of any Database Manager applications will therefore need a specific configuration of the ODBC driver to be properly channelled.

Each Bridges user will have to configure the ODBC driver of the DBM application. For the source of the data files, to meet the particular requirement, BRIDGES Database utilities check if the proper driver is available and then allow viewing of the database file. Once the Bridges Core has received the database file (coming from ORACLE for example), it can be exported to any Database Manager with direct links (such as ACCESS, EXCEL and DBASE).

8.3.4. Network Integration: LAN, WAN and the Internet

DAO provides links to multiple users working on Local Area Networks (LAN) and World Area Networks (WAN) but it does not provide Internet links.

The use of still immature and low-productivity technologies, such as Microsoft Internet Database Connector (IDC) or JDBC (Java Database Connector) has been tested to develop Internet links but still remains non-operational.

8.4. From geographical oriented to network oriented

As stated in previous chapters, GIS utilities are insufficient to manage transport graph structures efficiently. To overcome several of the identified problems, a set of specific utilities was developed. NIS utilities can be understood as re-programmed GIS utilities able to handle transport topologies as defined by GTF.

8.4.1. NIS Definition for Transport Databases

The need to define Transport Databases in NIS arises from the constraints on managing the complex topologies of Transport Systems within the framework of conventional GIS applications and led to useful indications on how to overcome these constraints, particularly in relation to "formats". This led to the definition of the GTF.

Transport Systems' database structures cannot be properly handled by existing GIS formats and utilities. However, GIS utilities facilitate a large number of useful transport information management processes, especially those related to transport infrastructure and environmental impacts, aspects directly linked to geographical features. These utilities constitute the starting point for the NIS definition.

Transport entities (or Classes of objects) are defined by topological structures which can only be partially associated with their infrastructural and geographical dimensions. Information related to categories of transport users, type of services, available vehicles and carriers, turns on intersections, infrastructure sections, terminals etc. must comply with specific interdependency rules in order to guarantee that it is consistent and represents the real system. Transport entities and rules must follow "graph topologies" (abstract network representations) in order to model the networked interrelations of transport systems information.

8.4.2. Defining Classes of Objects (Entities) in a Transport oriented NIS according to GTF

From a data model point of view, NIS is fundamentally different from GIS. NIS is based on "Classes of objects" (defined by database structures and topological attributes) rather than on "Levels" (with graphic attributes). Often, "classes of objects" and "levels" may coincide, and all objects belonging to the same class could be associated to the same level and vice-versa but there is a fundament difference between them that leads to a very different internal software architecture:

- All objects in a class share database structures and topological rules among themselves and other classes. Classes cannot have any attached graphic information or viewing attributes.
- All symbols in a level are viewed or printed together (according to the order in which they were created) in the sequence of all levels in the map. Symbols in a level cannot have any database attachments.

This conceptual change implies a trade-off: NIS has an easier graph database organisation but less intuitive mapping and graphic viewing than GIS and, of course, Desktop Mapping. Mapping "levels" may contain elements with particular symbols but no topological rules or even data attached, and only share the order of viewing in relation to other levels of the map. Levels usually become items in the legend of the

map (this will be used as the foundation for "NIS thematic levels", independent from "NIS classes of objects" but linked to them.

For the sake of consistency, it is important that all elements belonging to a "NIS Class" share the same visual attributes and topological properties. In order to personalise the symbols of a group of objects belonging to any class, "thematic levels" must be created. A series of thematic levels could be associated to a class and be viewed instead of or in addition to the "graph levels". While any graphic editing of thematic levels has no implications for the database structures (only the display is affected), the editing of "graph levels" implies changes in the database structures and therefore protection is provided.

Generally speaking, an NIS will have special problems in being fully efficient in dealing with raster images (it would have to include, in addition, normal GIS utilities to develop raster processing tools, not essential from a purely NIS point of view) or in being linked to GIS raster processing routines if needed.

Table **Error! Unknown switch argument.** presents a simplified representation of the characteristics of NIS in relation to GIS. Needless to say, there is a long list of GIS applications and options for personalisation and linking to DBS managers, so the following table should be seen in terms of facilitating a conceptual description of NIS rather than specific operational explanation of it.

Table Error! Unknown switch argument. **NIS Conceptual Definition**

GIS	NIS (according to Bridges definition)
Graphic elements organised into Levels (according to the order to be viewed or mapped)	Objects organised into classes (they share database structures)
Levels may have links to databases	Classes are linked to a graph topology
Levels (which facilitate CAD and mapping utilities) can be associated with "classes" to facilitate attachment to databases	"Classes" (which facilitate DBS and topological descriptions) can be associated with levels to personalise their viewing and mapping characteristics.
The graphic elements in a level may have different symbols.	All objects in a class share the same graph attributes (to personalise graphic attributes for arbitrary groups of objects, additional "thematic levels" must be created)
Levels can be understood as all graphic elements aggregated in the same item of a map "legend"	Each object is a single register in a table. All objects capable of being included in the same table belong to the same "class"

8.4.3. Elements of NIS architecture. Basic definitions

The definition of a given NIS database project includes the following general elements:

- Classes of Objects: They constitute a "network topology" or a "graph" and have databases (MDB, XLS, DBS etc.) [close to Database managers]

- Thematic or Symbolic Levels: Symbols attached to objects with graphic attributes depending on arbitrary selection of variables. They are created through "Thematic mapping" options in a library of symbols. A predetermined "default" symbolic level is defined to be attached to the objects database itself. [close to Desktop mapping applications].

A given thematic or symbolic level is defined as a group of different type of symbols linked to objects of any set of classes. The thematic mapping interface asks the user to define, one by one, the classes of objects to which symbols are to be attached. All the objects of a given class will share the same type of symbol but the parameters defining it (width, colour, size etc.) may change from one object to another (depending on the value of the variable selected or created for viewing). Each symbol in a thematic level is itself a "virtual object" linked to a real "object". A symbol can be moved and clicked, opening the table of the database linked to the object where the thematic variable is stored. The Thematic levels are stored following the same binary internal structures of classes of objects, up to 50 different levels, and can be viewed in any window together with any "Object class", "Vectorial layer" or "Raster layer" (see below).

Vectorial Layers: They have no information attached (DGN, DXF etc.) [close to CAD applications]

Raster Layers: They have no information attached. Cannot be superimposed over other elements (BMP, TIFF) and require particular utilities for zooming, scaling, printing etc. [close to Graphic design applications]

GIS is based on the same elements but manages them in a different manner to NIS, as has been already explained. Let us define "Class of Objects" more precisely:

Class of objects: Any group of objects ("such as Airports") sharing the same topological attributes and rules. All objects in a class have common graphic attributes as a default, called "by_class".

Classes: Any group of objects belonging to a class ("French Airports", "Regional European Airports" etc.). Individual objects may belong simultaneously to different classes. Each class usually has Databases coming from different sources. [In order to personalise the way each class or even each individual object is viewed, additional graphic attributes "by_object" are allowed; these personal graphic attributes of each individual object can be produced and stored through the options included within "Thematic mapping"].

Tables: Composed of any group of objects belonging to a class associated to a list of variables (ideally, all tables within a class database should have the same objects). A Database may have an unlimited number of tables.

Hyperclasses: aggregations of classes belonging to different classes under arbitrary conditions ("Airports" and "Roads" with private financing). As classes, they may have classes and tables.

Superclasses: aggregations of classes which share fundamental topological attributes (nodes, links, polygons)

Hierarchy of objects (based on classes): Superclasses, classes and hyperclasses, classes and table-contained objects follow hierarchical dependence relations.

Hierarchy of variables (based on themes): All variables contained in all tables of the whole Database can be organised according to their thematic similarity.

Metadata tables: Each database project must have metadata information. This metadata information is related to the table itself and should be linked to the source which provided it. Link to Source providers: each table is associated to a single source provider. Its address and full reference must be contained in a separate "Database of Sources" (supplementary and/or associated to the Digital Data Guide, DDG).

8.5. Software implementation of Bridges NIS Utilities

8.5.1. Database Management of Transport objects

Because its openness, as a default database manager Microsoft ACCESS has been used (SQL Server in Microsoft Office'2000 is being analysed currently to substitute ACESS, and Oracle to add spatially-related queries and efficient access to remote databases through Internet).

In ACCESS a "Database" is contained in a file (DBSNAME.MDB) which contains an unlimited number of "Tables". Each table contains an unlimited number of registers (rows) and a maximum number of attributes (columns). The capacity constraints of ACCESS are 1 Gigabyte for each database and 125 columns in each register. In addition to tables, a database file contains an unlimited number of "Queries". There are two basic types of possible query: queries to select all objects verifying a given condition and queries to create new attributes and store them in already existing or new tables. Queries cannot be made between different databases.

ACCESS can be easily integrated into open platforms since its Visual Basic 4.0 programming language uses it as a default Database manager. A stand alone Visual Basic application may enjoy ACCESS utilities even if the ACCESS application is not available on the hard disk or LAN itself.

Links to ORACLE and other more powerful Database managers will be analysed later on but ACCESS is an unavoidable first choice for development of the BRIDGES environment.

Within Bridges NIS, a "DBS Project" is defined therefore as a set of databases, tables and queries adapted to the catalogue of transport entities and objects:

8.5.2. Basic DBS/GIS concepts within NIS

The "DBS project" is constituted by a series of databases plus a "Project DB" which contains "Reference Tables" (RF) related to the addresses of all databases in the project and "Dynamic Tables" (DY) as data transfer tables between ACCESS files and the remainder of the NIS or external applications.

Within each database, different types of elements are considered: Source Tables (SR), Code Tables (CD), Queries (QR), Working Tables (WK) and Metadata Tables (MT).

Source Tables (SR) contain data as supplied from each source, with the same codification and attributes.

Code Tables (CD) contain equivalences between the object codification of different sources and the official standard ones (ISO, UN, IATA etc.).

Queries (QR) are pre-programmed routines needed for integrating source tables into working tables using the equivalences in the code tables. Conditions for attribute assignment, checking etc. are included in the query tables

Working Tables (WT) are those tables for use in modelling and other work.

Metadata Tables (MT) contain information related to the sources providing the information stored in each source table, and queries applied for each working table. Further information related to the characteristics of each particular

variable etc. must be obtained through access possibly to the source itself via the Internet.

Each Table contains data for a single class of object and it has a single source: Each register in the table represents a particular object and each column an attribute. Three attributes are required in all cases: Name, Code and CAD_Link (where it makes sense).

Bridges NIS utilities are based on two separate families of routines: graphic-oriented [MGIS] (C++) and alphanumeric [MDBS] (Visual Basic). Linkage between both sets of routines is assured by interactive calls through the Bridges Communication System.

8.5.3. CAD, GIS and NIS Management of Transport objects

NISystem contains a number of routines, programmed in Borland C++ allowing the management of transport entities as such. The following sections provide for a reference to their capabilities.

Editing graphs

A number of pre-defined graph elements are available for editing. They are based on three objects: "Nodes" (associated to a single point, x, y, z co-ordinates), "Links" (associated to lines defined by two points) and "Groups" (associated to a list of nodes and links).

Based on these fundamental entities, transport specific entities, such as "services" can be defined (e.g. as a groups). Bridges NIS interface provides the user with a list of transport entities he can select when editing a graph. Bridges NIS automatically generates the data structures related to these entities.

Graph Generation

Procedures to create networks automatically, based on predetermined conditions (for example based on identifying missing links, on pre-existing lines such as roads from CAD maps).

Graph Checking

A number of utilities have been programmed to verify topological conditions between NIS entities automatically (e.g. checking whether all rail stations are connected to rail and road infrastructure links and identifying those connected to other types of link, e.g. rail stations within airport terminals connected to air services). Checking utilities display the identified objects on screen, facilitating correction by the user.

A list of checking utilities is available for the user to check imported graphs.

Graph Matching

Two matching files are needed to transfer data attached to one graph to another with different segmentation: CAD match (equivalence between elements in the origin graph and the target graph) and DBS match (logical or analytical procedure to transfer variables attached to the elements in the original graph to the equivalent elements in the target graph, as the in the target graph, the elements will get the average, weighted average, minimum or maximum values of the equivalent elements in the original graph.

Graph Analysis: Minimum Shortest Paths

- Dijkstra Algorithm
- Ford Algorithm
- Always Forward (one step back) Algorithm
- Always Forward (two steps back) Algorithm
- Shortest path with node and links requirements
- Shortest path with crossing points and transhipment requirements
- Trees from a given centroid
- Circles (Salesmen problem)

Flow assignment

- All or Nothing assignment
- Deterministic User Equilibrium (Frank-Wolfe algorithm)

Statistical and mathematical functions

General statistical functions have been programmed: Normal, Lognormal, Weibul, and basic utilities to adjust histograms to any of these functions

A set of data (x, y) can be loaded from the Database manager in order to adjust it to a mathematical function with predetermined parameters. Errors are statistically analysed: for a level of significance to be fixed by the user the module evaluates whether the errors can be characterised by a Normal distribution. It provides the Pearson coefficient (R^2).

A histogram can be loaded in order to adjust statistical functions (Normal, Binomial, Poisson, Uniform, Exponential, Lognormal, Weibul, Gamma, Beta, Fisher, Student, Chi square and Rayleigh). Parameters can be adjusted statistically (in the case of Normal, Exponential, Uniform) or predetermined by the user. For a level of significance, it calculates whether a given statistical distribution models the loaded data acceptably.

Mathematical functions editor

An editor of mathematical functions y=f(x) (polynomial, trigonometric, exponential, logarithmic, potential and statistic). The user can fix the free parameters of each type of function and obtain a graphic representation of it (the scale of representation and a number of

Only basic combinatory functions can be calculated: Variations, Permutations and Combinations, with or without repetition, according to the number of elements, positions and groups chosen by the user.

This function applies the well-known "Simplex" algorithm to obtain the maximum or minimum of a linear function under linear conditions (or constraints). The user interface facilitates the process of defining both the objective function (to be maximised or minimised) and the constraints.

A Markov process is based on three elements: a number of alternative "states", the probability of each state initially (matrix of probabilities, to be fixed for the first step) and the probability of a change from one state to any other (transition matrix of conditional probabilities). The parameters are the matrix of initial probabilities and the of transition matrix.

A Queue algorithm analyses a channel (single or multiple) with a (limited or unlimited) number of arrivals following a Poisson distribution. Given the ratio of arrival and departure in the channel, there is a probability that a number of those arrivals remain in queue while others pass through it. The algorithm calculates the intensity of traffic through the channel and the average time in the queue.

Basic entropy measures (simple, conditional and joint entropy) have been programmed, as well as interactive parabolic and logistic functions to explore the emergence of chaotic behaviour from the dynamic iteration of very simple functions.

Genetic algorithms have been applied to support some internal routines (for example to create Eulerian tours, to allocate a set of individuals to groups, to optimise the set of parameters needed to achieve equilibrium between trip generation and distribution in "four steps models"). This option will give users the possibility of defining a "genetic process" in each of its steps (reproduction selection, crossover and mutation) under different rules (elitism, ranking).

Thematic mapping

The following options (with similar capabilities of Mapinfo and Arcview) are available:

Highlight
Bar
Pie
Pattern
Flows
Figures
Isozones
Lines

Elements to be represented can be grouped according to a Discriminant Analysis (based on Genetic Algorithms).

Thematic maps are stored as independent classes and remain alive even if the data and formulation creating them are removed.

Cost Benefit

Cost Benefit Analysis provides a complete analysis, including the financing and interest rates of intermediate cash flows. From a list of costs and benefits the routines provide a list of standard CB indicators

Multicriteria algorithms

After defining a list of alternatives, criteria and weights to evaluate them, the module provides three aggregation methods: Qualitative Concordance, Numeric Interpretation and Permutations, to be used depending on the kind of valuations established (quantitative, qualitative or mixed).

Decision trees

Decisions are related to "actions" to be taken in different "states". The best action is the one with maximum "utility". There are initial states (each with its probability of occurrence conditioned by each action) and secondary states (with probabilities conditioned by initial states). Utilities are a function of actions, states and secondary states to be fixed depending on the problem.

9. Towards a Decision Support System: (ES/DSS)

9.1. Objective: From an "Open System" to a "Decision Support System"

This chapter presents the development of a key Bridges tool, the software application developed to create Expert Systems (ES) for advanced transport models. The conversion of an "information and modelling" system into a "decision support system" requires intelligent intermediation between the system outputs and the end user requirements to be provided by an Expert System (ES). An ES consists on a set of tools that facilitate navigation by users through all the available system resources (databases, maps, models etc.). Bridges ES/DSS is a friendly application allowing the user to define such Expert Systems for advanced models.

For simple systems, the user interface itself may have enough expert capabilities within to make the development of an Expert System unnecessary. In the case of complex systems, however, when many advanced models are linked to it, an independent module specialised in channelling the knowledge exchange between model answers and end users' specific questions, is indispensable.

The Bridges tool to develop Expert Systems has been empowered with complementary tools such as desktop mapping, GIS and database management. Therefore, once particular Expert System for a given model has been created, it can be tested in a complete Decision Support System frame.

9.1.1. The users of Bridges Expert System/DSS

Three categories of possible Expert System/DSS users have been considered:

1. Expert users (modellers, program developers, consultants etc.) who have sophisticated data and models and want to make them accessible, in a controlled and limited manner to other, non-expert users.
2. Novice users that do not necessarily know all the intricacies of each and every model but want to ask specific questions within the range covered by one or more models.
3. An ES/DSS administrator, who maintains a base of models, data, templates, knowledge files and other resources which are necessary to support the questions that the ES/DSS can answer.

9.1.2. What is an Expert System?

An Expert System is a program that behaves like an expert in some, usually narrow, domain, or application. Expert systems have to be capable of solving problems that require expert knowledge in a particular domain. They must possess that knowledge in some form.

Therefore they are also called knowledge based systems. However, not every knowledge based system can be considered to be an expert system. An expert system also has to be capable of explaining its behaviour and its decisions to the user, as human experts do. Therefore, expert systems have to have a friendly user interaction capability that will make the system's reasoning transparent to the user.

It is convenient to divide the expert system into three main modules:

1 A knowledge base

2 An inference engine

3 A user interface

A knowledge base comprises the knowledge that is specific to the domain of application, including such things as simple facts about the domain, rules that describe relations or phenomena in the domain, and possibly also methods, heuristics and ideas for solving problems in this domain. An inference engine knows how to use the knowledge in the base. A user interface caters for smooth communication between the user and the system, also providing the user with an insight into the problem-solving process carried out by the inference engine. It is helpful to view the inference engine and the interface as a module, usually called an expert system shell.

The foregoing scheme separates knowledge from algorithms that use the knowledge. This division is suitable for the following reasons: the knowledge base clearly depends on the application. On the other hand, the shell is, in principle at least, domain independent. Thus a rational way of developing expert systems for several applications consists of developing a shell that can be used universally, and then to plug in a new knowledge base for each application, or extend the existing knowledge base with more information. Of course, all the knowledge bases will have to conform to the same set of formal characteristics that is 'understood' by the shell.

9.1.3. Software languages and tools used to build up Bridges ES/DSS

The Bridges Expert System module has been developed using multiple software languages and tools. The most suitable development tool/language was selected for each one of the different sub-modules, as follows.

1 Microsoft Visual Basic 5.0. Used for development of the main Expert System program, templates, database programming, application calls, and mapping utilities.

2 Microsoft Office 97. It is the "default" environment. In particular Access 97, and Excel 97 were used for database applications and testing of small-scale models.

3 ESRI's MapObjects 1.2. MapObjects is a collection of mapping and GIS components including an ActiveX control (OCX) for development environments such as Visual Basic, Visual C++, Access, and others.

4 Amzi! Prolog 4.0 + Logic Server. The Amzi! Prolog implementation is very close to the "Edinburgh Standard" Prolog language. It was used for the development of the ES shell and its interfacing to Visual Basic.

Amzi Prolog was used, as it is fully integrated with the rest of Visual Basic code.

9.2. The concept of Bridges Expert System

According to the above technical description, the objective of an ES is to become the module that will allow a user to formulate questions to the whole information and modelling system, transform them into a series of calls to transport models and databases, and present the end results either through a specific user friendly interface, or through the user's own routines.

Since the ES must also be used as an independent module, special emphasis was placed on the development of direct access to database utilities and GIS mapping functions from within the ES. Thus, such functions are either called directly when needed from the Visual Basic code, or they are available as separate tools from the ES/DSS menu (VisData for databases, and ES/DSSview for maps).

Note that since there is an infinite variety of models and data that could be used within a given open system, the ES/DSS provides the tools for automating the preparation of such menus and requests. By using the ES/DSS tools the model developer, who knows all the details of his/her model, can prepare suitable questions that an unsophisticated user may safely ask the model. This is particularly true for complex models, such as those built with VIA and MEPLAN. Typically, they involve huge amounts of data (hundreds of Megabytes, or Gigabytes), which reside in various files, require calling various sub-modules, making assumptions, and knowing which data to select for presentation under given conditions. Having the ES/DSS, the modeller, who knows what valid questions one may ask the model, can prepare a template (an ASCII file) for each one of the questions, test them out, and then submit just the necessary data with the template to the ES/DSS administrator. The ES/DSS automatically parses the templates and based on them generates simple menus containing the questions for the novice user. In cases of missing data or logic assumptions concerning the model's execution, the modeller may prepare an additional knowledge base file stating the preconditions for each assumption needed to run a model, or to select a model parameter.

For example, for a question like "What are the flows on Spata bypass, during airport construction?" the answer may be obtained through an SQL statement properly phrased in the template. The modeller knows in which tables the data are located, and under what names, knows the SQL syntax (or the simpler template syntax) so he can write a template for the question. The ES/DSS parses the template, and shows on its menu the much simpler question "What are the flows etc.". When the novice user starts the ES/DSS he may find this, as well as other related questions, grouped under the "Spata airport construction" category of models. When the question is selected, the ES/DSS fetches the results and displays them in the appropriate form (e.g., a listing, a graph, a map etc.).

ES/DSS will facilitate the following options:

1 Database utilities for querying, updating, obtaining reports etc., through an ETIS interface link;

2 Comparative answers to user questions relating to alternative scenarios/ options;

3 Establishment of expert rules to estimate missing data;

4 Initiation of transport models appropriate to the functionality requested by the user; and

5 Evaluation of model results according to multi-criteria models (e.g. Should I recommend this corridor as opposed to another?).

The ES/DSS is a combination of an Object Oriented Interface (OOI) and an expert system (ES). Options (1) and (2) can be handled directly by the OOI. Options (3), (4) and (5) can be provided through an expert system module that will "know", depending on the question, which model data to use, how to fill in data gaps and what kinds of results to display to the user. Interaction with the ES is also achieved through the OOI utilities. "Knowing" means that a knowledgeable person has prepared ahead of time the necessary template or knowledge base files, and has provided the corresponding data, so that the template(s) can be parsed and executed correctly. The facilities offered by the ES/DSS to support the above options, are explained in the following chapters of the report.

Once a set of questions has been properly set up and tested, then a "user" may go directly to the "Q&A" menu of the ES/DSS, browse through the available questions, select one, provide any additional values that may be needed (e.g., display ranges, desirable colours etc.) and get the result.

9.3. Bridges Expert System Interface

The ES/DSS has its own Interface (OOI) for communicating with its users. The OOI comprises a set of menus, forms, dialog boxes etc. which are either preset or generated at run-time to facilitate the user dialogues.

The preset part of the OOI guides the user through the standard options of calling an ES/DSS module (data dictionary, template parser, expert system etc.), or a utility (Visdata, ES/DSSview etc.).

On the other hand, certain parts of the OOI are generated based on the information contained in the templates and knowledge base files available during ES/DSS execution. It comprises menus, list boxes, input boxes, maps, graphs etc. that are generated either in order to prompt the user or to display the answers to user questions. Essentially, the ES/DSS provides a shell for generating this part of the OOI. The expert user prepares templates, defining the questions that will be included in the OOI as well as the answers that will be displayed.

Thus, the run-time OOI of the ES/DSS is actually "developed" by the expert users that prepare and test their templates and then make them available, through the ES/DSS to the novice users (simply "users").

For a given model that we want to be included in the ES/DSS, the OOI development work starts off by defining:

1 What kinds of questions a user may ask ETIS

2 What kinds of problems ETIS can solve

3 What kinds of information will be accessible

4 Which models will be used

5 Which databases can be accessed

6 Which application programs will be interfaced

The starting point of course is the existing ES/DSS interface, with the sample models, data, maps etc. that it already contains. However, using the ES/DSS facilities, an expert user can extend this interface so that it can connect to distributed databases, diverse software models, and even commercial packages. This follows an Object Oriented approach, and, at this point, it has been programmed in the ES/DSS using the ActiveX technology available through Visual Basic, Access, MapObjects and Amzi Prolog.

At the user level, there is an effort to keep these implementation details of the ES/DSS, invisible as far as possible. All systems peripheral to ETIS are made accessible through this single interface in a uniform and simple manner. For this reason we have included two important utilities in the ES/DSS program:

- The Visdata program, which provides most of the MS Access functions to the ES/DSS users, without any royalties or additional costs. In this way ES/DSS users can examine and update all kinds of database files (Access, dBase, Excel etc.) as well as files needed to run the ES/DSS models.
- The ES/DSSview program, which is a simple GIS application using ESRI's MapObjects. It also provides ES/DSS users, free of charge, with basic GIS

functions for viewing various kinds of GIS files (ARC/INFO, MapInfo, shapefiles, bitmaps etc.), as well as basic editing and shapefile creation capabilities.

- The integration into Bridges/Communication System of Bridges/ES makes all other capabilities included in a system accessible from Bridges/ES.

A critical aspect of the ETIS system is that with its ability to interconnect with various databases, models etc., it becomes very complex in terms of knowing what to do and why. The user does not need to know what all the various databases and models are but only what kinds of question the system can answer. Therefore, the OOI of the ES/DSS was designed to comprise two layers:

- A higher, user oriented layer; and
- A lower, application specific layer.

The user oriented layer is the one, which is visible to the user. The application specific layer is not directly visible to the user but it comprises the tools that make models, databases, GIS utilities etc., available to the end user.

9.3.1. User layer of the OOI

The user layer of the Object Oriented Interface enables the user to supply the necessary information for the problem at hand, and, also allows the ES/DSS to present the user with the available information, the problem formulation and the results.

The ES/DSS, through the use of hierarchical menus, boxes with grouped input, standard models etc. guides the user through a series of questions such as:

1 Do you want specific flow/socio-economic/ land use data?

2 Do you want to run a specific transport model?

3 What is the geographical area that you want to evaluate?

These are examples of the infinite number of questions that can be prepared through the ES/DSS templates and knowledge based files and presented to the user in the simple manner described above, through the user layer of the OOI.

The system applies the user choices from among the set of questions/options to build a formulation of the problem, select appropriate databases, query certain data from a given database, select the appropriate transport model and prepare the necessary input data set for running the model(s).

Also, the ES/DSS through the OOI may obtain additional information about the user's preferences in order to derive the appropriate indicators and present the results in the most suitable format, e.g., which colour, on what map, with graphs, charts, tables etc.

The user may have the option to supply information on problem parameters, e.g., transport mode and network alternatives. Alternatively, the system may find/ propose default values for certain problem parameters, e.g., transport, socio-economic, land use etc.

The user interface is menu/form driven. All model or standard query related information is grouped under a single menu path along with the necessary selection and data input boxes, command buttons etc.

Thus, the OOI serves as a graphic and textual tool for query input and result output. It is based on menus, browsers and graphers for result curves and so on, complemented by a graphical interface for presentation of geographical data.

9.3.2. Implementation of the OOI

The main task of ES/DSS is to offer a User friendly Interface where Expert or Novice users can pose questions and the ES/DSS will try to find the answers by calling Databases or Transport Models, either locally or throughout Europe via Internet Servers. Finally the ES/DSS offers a number of Viewing techniques for the interpretation of results.

1 The ES/DSS OOI is Menu Driven with linkage/GIS/ database capabilities.

2 The main concept is the Query which is expressed either through a Template or with the Visual Query Selection (menus, list boxes etc).

3 There are GIS capabilities, Zoom, Pan, Mark, Display Map, Hide Map, which can be activated in two ways: through Menu options or through a GIS palette.

4 The User must define Thematic Layer(s), Driving Variables and Properties in order to form the question.

5 The User chooses the Map to display and select Areas of Interest.

6 Thematic Layers, Driving Variables and Properties belong to Methods which are organised in Class Hierarchies.

7 A Collection of Methods is contained in a Template. Templates are saved in ASCII files.

8 When the User prepares a Template for the ES/DSS, the Template Compiler is activated. The output of the Template Compiler is specific SQL code and a Visual Path where the User may select the Answering Path that the ES/DSS will follow in order to answer the question. The Expert User may change part of the solution path by rewriting one or more methods and by applying Testing and Debugging techniques.

9 The whole Query process can be Saved so the User can call it again to modify it or just evaluate the results.

10 When the ES/DSS has the answer to the Question, the User must supply Viewing output parameters. The ES/DSS has the capability to display results through GIS palette, Excel, Access or a GIS package, such as ARC/Info.

11 The User may supply Criteria and Weights in order to apply multi-criteria evaluation to a given problem.

9.4. Expert System (ES) modules

The ES comprises (or makes use of) the following modules:

- A Template parser that interprets user queries (specified by means of the OOI).
- A Query processor that executes SQL queries. It also calls the appropriate modules (e.g., ES/DSS core, transport model, database) and/or inspects the corresponding parts of the ES knowledge base.
- An ES shell that applies rules in order to determine which modules to use under given conditions or to apply realistic assumptions for missing data. The ES shell processes Rule files (or knowledge based files). A rule file is an ASCII file containing simple rules for processing of models.
- A Multi-Criteria Evaluator that combines results from the diverse sources above to be used in a multi-criteria model.

These ES/DSS modules can be called separately or in combination with each other. Thus, the first two modules are called during Template parsing and execution. The ES shell module is used specifically for processing Rule files.

The main module is the Query processor which allows the user to form queries for the Manager systems of the OOI to process. Its main task is to analyse and separate user queries into sub-queries and to pass the sub-queries to other modules. Additionally, it is able to answer some kinds of queries by itself. This holds in particular for queries which focus on items already stored in the ETIS database or that can be deduced by means of default values or the user's current assumptions. Before passing information to other modules, the Query Manager checks the consistency of data and results, keeps track of the user's current assumptions, and applies rules to attempt to provide realistic assumptions for missing data.

The ES modules (1), (2) and (4) have been developed in Visual Basic and the ES shell module (3) has been developed in Prolog, whereas all its screen I/O is passed to Visual Basic and handled by the ES/DSS's interface.

All access to the above ES/DSS functions is provided through the Object-Oriented Interface (OOI) described previously.

9.4.1. Initiation of transport models

While transport models exist in all varieties, simple and complex, they mostly require large amounts of data and time for their preparation, calibration, testing, running, and presentation of results. Once they are set up considerable expertise is required to understand what they really do, and what answers they can safely provide to an external user. However, in order to be useful to the user community outside of the model developers group, models need customisation and careful design to control what the inexperienced user may do with them.

The ES/DSS provides a solution to the above problem, of model integration and results dissemination. By writing a set of Templates the modeller can control what kind of questions a user may safely ask his model. Also by providing the subset of data the modeller wants to distribute (ASCII files, database tables, maps etc.) he can provide a useful package for information retrieval and processing that can be highly valuable for administrators or other interested parties.

This purpose is achieved in the ES/DSS by two means:

- Preparation of Templates, one for each type of function that the modeller wants to make available to end users.
- Generation by the ES/DSS of a Query path, after parsing Templates registered in the ES/DSS database.

The process of Template preparation involves the following:

1. Selection of a data set (ASCII files, database files, GIS files) to be used along with the template.
2. Writing the Templates. For each question that a user might ask the model, a different Method is written. Methods can be organised in one or more template files. Also they are grouped for easier classification of menus and retrieval by the user.
3. Testing-debugging each method. Using the ES/DSS the modeller can run each method separately and check whether it produces the desired results.
4. Submitting the templates to the ES/DSS end user, or to a central ES/DSS administrator. The ES/DSS has the option of applying a password to restrict template use, and of locking the template source file from unauthorised use, or tampering by non-expert users.

Once a template is available to the end user, all its methods appear as entries in a list table, where the user may select which methods to register for further use. When a method is registered it is also placed automatically in a set of hierarchical menus that guide users through its execution.

From then on, a user may just browse through these menus, select a method, specify further parameters that the method may need, provide display options, and view the results. This sequence of menu selections guides the user through the valid options to final completion of his request. Each path on the menu hierarchy constitutes a Query path which is available to all ES/DSS users.

Additionally, a modeller may prepare a Rule file which contains rules for selection of models under a set of conditions, for filling data gaps, for conditional execution of program files and other options. Each rule file is considered as a knowledge base of the ES shell. It is an ASCII file that contains a set of rules written in an "IF conditions THEN action" format. It is checked-tested-debugged in a similar manner to the Template file, by the modeller. The result of processing the Rule file is that the ES/DSS may ask the user additional questions as these are encountered in the knowledge base rules.

Finally, the user may provide a Data Dictionary (DD) for all the parameters used in the above Template and Rule files. The DD is an Access database file that contains the hierarchies, names, definitions, units, default values etc. of all the parameters of the model.

The implementation of "initiating transport models" in the ES/DSS is as follows.

1. The OOI application layer handles the Template and Rule files.
2. The OOI user layer generates and processes the Query paths.
3. The ES/DSS modules (Template parser - Query processor - ES shell) are executed in sequence.

4 Depending on the output of each module the corresponding ES/DSS (Core-Model-Database-Application) Manager system is called.

9.4.2. Templates

Template: An ASCII file that contains a Model's Methods submitted by the Modeller. Each Method consists of reserved keywords that the ES/DSS engine tries to parse, analyse and store. It guides the ES/DSS engine on how to execute a specific method and act on the model's data.

The purpose of Templates is to give the modellers a tool that allows them to make some of their model's functions available to outside users in a controlled, yet user friendly manner.

Templates are a well known solution that has been adopted by many commercial applications such as MS Word and MS Excel. The basic idea of a Template is an ASCII file with commands, arguments and possibly some data. It is not a substitute for Macros or a Scripting language or a data file, it just "drives" the application providing definitions and orders.

As we have already mentioned, the Modeller cannot normally submit the real Model but just a set of data output results. ES/DSS can further process the data in order to produce answers to certain specific questions posed by the users. But how can the ES/DSS act on those data? What rules and restrictions must ES/DSS follow? If the Modeller were always present he/she would apply some special touches to produce good results. But now we need a "driver" that will guide ES/DSS on how to act on those data. This "driver" is the Template.

The Modeller may also submit some independent programs along with a data set. Again ES/DSS needs to know how to "call" those programs in order to activate the Model. The "driver" is again the Template.

The Template is an ASCII file that can be written in an Operating System, using any editor or word processor that supports the ASCII format. Because the file is in ASCII format it can be exchanged between heterogeneous operating systems like Windows NT, UNIX, VAX VMS etc. without the intervention of any filter. This ASCII file will be read by the ES/DSS application and parsed line by line.

Each Template file consists of one Module and one or many Methods. The Module can represent a whole Model and each Method a procedure or function of a model. The ES/DSS application supports many Modules in one Template but this is not advisable for reasons of clarity. Each Method consists of a set of predefined keywords and their arguments

9.5. Expert System Software Architecture

The ES/DSS Core Engine is a Windows Menu-driven Application, written in Visual Basic 5.0. The back-end database is ACCESS and the linking library is DAO 3.5. ES/DSS Core Engine is also a Shell where independent programs - utilities have been incorporated.

The Administrators have the possibility of adding further independent programs into this Shell in the future.

The Modellers can have a copy of ES/DSS and test their Model Templates. So ES/DSS can serve as a workbench for Modellers where they can try many variations of Methods just by writing a simple Template file. This is much simpler than programming the execution sequence of those methods independently.

The End Users sit in front of an ES/DSS application copy and enjoys a rich set of Tools and Utilities all presented through a friendly User Interface. They can pose a question to the Expert System, search for a specific Method using keywords, browse database tables, and examine results in a digital map.

9.5.1. Expert System Shell

Internally, ES/DSS stores Methods in memory Objects and all of them are tied together into a Collection. Keywords of the Method are stored as a Collection inside the method-object. The combination of Object Oriented Programming and Collections gives great flexibility for Method manipulation and storage. All the actions that ES/DSS performs have been coded as Classes that can expose Methods and Properties. A Collection operates like a memory storage device, we can add and remove "elements", dynamically at run time, easily. Imagine what happens when those "elements" are Objects, which means memory structures that store other sub-elements and so on. This gives great flexibility to store and manipulate keywords and their context lines.

9.5.2. Parsing

Reads an ASCII template file and analyses it line by line for syntactical and logical validity. Loads a new Method into memory as an object structure.

Initially the User selects a pathname for the Template file. ES/DSS reads this file from disk and tries to "parse" (i.e. analyse it line by line) the ASCII file. If a syntax or logical error occurs the Parser displays an error message and terminates.

When all Method's keywords have been analysed successfully, the Parser checks for existence of this Method in memory. If the Method does not exist in memory, Parser loads Method as a new Object into Methods Collection.

Execute one or all or a combination of Methods loaded in memory. Store Methods in a Database and execute them in the future at any time without parsing. When the Parser finishes its job, all the available Methods in Memory can be executed sequentially or can be registered into the Methods Database. One Method may call another Method and this gives great flexibility for Modellers in creating scenarios. In addition, Methods that are stored in the Database can be executed directly at any time in the future avoiding the parsing process. This can be achieved because Methods

stored in the Database are locked, so end users cannot change them. This guarantees that Methods have no syntactical or logical errors.

9.5.3. ES/DSS: Data dictionary

This is an Access database with Data Dictionary Tables, Queries, Forms and Reports. Modellers should complete the Forms with all the necessary data that best describe the characteristics of their Model. There are sample data in the database that will help the Modellers understand the nature of the data that has to be supplied.

The purpose of Data Dictionary is to allow recording and updating of all the necessary parameter information that is needed in order to run an ES/DSS model.

The Data Dictionary (DD) describes in a Table format all the variables external to the module, which the ES/DSS should know about. An external variable is a variable that could be used or referred to by another module. Variables internal to a module (that the ES/DSS does not have to know about) are not described in the DD.

The Templates describe in a standardised format (combined with pseudo-code when necessary) all public functions of the module, which we call methods. A method is any function of the given module that can be called or accessed from another module. Functions internal to the module are not considered as methods (the ES/DSS does not have to know about them) and there is no need to describe them with Templates.

10. Bridging Databases (DDG)

10.1. Objective: A Directory of European Transport Database Sources

Bridges Digital Data Guide (DDG) is a computerised tool having two main functions: the identification of the sources of information and electronic databases dealing with transport in Europe, and description of the characteristics of these sources and databases.

Initially, the objective of Bridges was to develop a DDG focusing on international information providers and database products available in Europe (transport ministries, statistical offices, professional organisations, sectoral associations) in electronic form. Bridges users could use DDG to find the information they need to carry out a given analysis. Bridges DDG was envisaged as a "reference directory" which could be the basis for other necessary directories (in the ETIS building process):

- Directory of data sources (DDG)
- Directory of models available
- Directory of experts
- Directory of policies

Obviously, not all data and all models available can be integrated within a decision support system. Therefore, a more dynamic strategy has to be adopted: all existing sources can be properly identified and monitored, but not necessarily all their available databases can be included in the system. In its present version, Bridges DDG contains about 1,500 relevant information providers or products. Complementary to database sources, some models and software products have been included to illustrate the overall aim of the "directories" approach but DDG remains focused on data sources.

Bridges DDG has been designed as a "living" tool. Through its software interface, users can introduce additional information, update and remove existing data. Bridges DDG has also been progressively extended to a large number of key "national" information providers, such as transport operators, research institutes or market research companies, even if their databases are neither supplied electronically nor accessible through the Internet.

DDG has been developed in Microsoft ACCESS and its user interface and main management options (e.g. search of particular sources under predetermined and free queries) programmed in Visual Basic. Because of the simplicity of the database structure, migration from ACCESS to other database systems is easy.

As a maintenance mechanism, DDG must be regularly updated by a central body (e.g. EUROSTAT, DGVII specialised cells etc.) and core data sources be disseminated through Intranet and the Internet to all users. An automatic procedure for automatic updating from Internet sources, and quality control check of user's modifications could also be designed.

10.2. DDG Structure

The DDG is structured in six modules:

- Information providers
- Database products
- Modelling providers
- Modelling products
- Software providers
- Software products

Various information sources are included in the first module called "information providers", with the exception of "modelling providers" and "software providers" which are included in separate modules because specific parameters are necessary to describe them.

The DDG focuses on electronic databases (on-line access, diskettes and CD-ROMs). However, many databases are "paper" based but represent a significant proportion of transport data available in Europe. For this reason, it was also decided to include the most relevant "non electronic" databases existing in Europe in the DDG.

A system of queries allows the DDG user to identify:

- All information providers covering a specific geographical field,
- All information providers engaged in a specific activity,
- All information providers dealing with specific ETIS variables.

In its current version (revision 1), the number of providers and products listed in the DDG is shown in Table **Error! Unknown switch argument.**:

Table Error! Unknown switch argument. **Current State of the DDG**

Nature of data	**Number**
Information providers	772
Database products	559
Modelling providers	76
Modelling products	85
Software providers	69
Software products	71

It should be stressed that some information providers may offer up to 20 database products. Conversely, several information providers have no corresponding database product. The "information providers" are defined by their country of location. The database products are defined by the geographical area covered by the data (national or international).

10.3. Typology of information providers by activity

Each "information provider" is characterised by its main activity, according to the following typology:

- Transport operators
- Ministries of transport
- Statistical institutes
- Environmental agencies
- Road administrations
- Civil aviation administrations
- Professional associations
- International organisations
- Research institutes and universities
- Transport training organisations
- Market research/consulting
- Transport directories
- Transport equipment and systems manufacturers
- Global distribution systems
- Observatories, monitoring centres
- Web directories
- Magazines, newspapers

10.4. Sources of information

The following sources of information were used to prepare the DDG:

- Interviews with the statistical offices in the 17 European countries
- Interviews with the ministries of transport in the 17 European countries,
- Identification and analysis of 180 Websites at world level
- Data available through the APAS project "databases and scenarios" and the RTD projects INFOSTAT and ASSEMBLING,
- Data from the "Intermodal Expert Group (IMEG)" project.
- Desk analysis of the Eurostat electronic databases,
- Interviews with UNO, ECMT and other international sources.

The DDG also contains information about Eastern Europe but it is less comprehensive than for Western Europe.

10.5. DDG User interface

10.5.1. General menu

The general menu of the DDG proposes six categories of information:

- Information providers (sources of data)
- Database products
- Modelling providers
- Modelling products
- Software providers
- Software products

10.5.2. Search for an information provider

The menu "information providers" contains four items: All, European interest, Favourites and Query.

10.5.3. Sub-menu "All"

The list of all information providers contained in the DDG is presented in alphabetical order.

1. If you cannot easily find the name of the information provider you are searching for, you can use the automatic search system. Click on the search area located at the bottom of the list and type the first few letters of the name being searched for
2. Click twice on the name to open the corresponding window
3. Click on the "product" button in order to see the "database products" corresponding to the information provider selected
4. Click on the cross at the top right of the dialogue box to close the "information providers" window

10.5.4. Sub-menu "European interest"

This sub-menu proposes a pre-established list of information providers that are considered as key sources of information at European level. The "European interest" characteristic can be added to a source by checking the corresponding box in the information provider's window. A selection has been set but can be modified at any time by the user.

10.5.5. Sub-menu "Favourites"

The DDG user to an "information provider" can add the "Favourites" characteristic by checking the corresponding box in the sources' window. When clicking on "Favourites", the list of sources will be narrowed to "Favourites" sources.

10.5.6. Sub-menu "Query"

By choosing some criteria in the query (such as the ETIS variable, geographic area etc.), it is possible to narrow the list of "information providers" proposed by the DDG

1. In the general menu, select "Query" in the list at the middle top of the general menu
2. Select some criteria
3. Click on the "execute" button
4. Close the query dialogue box by clicking on the cross at the top right
5. The number of "information providers" which correspond to the criteria chosen in the query is indicated at the bottom of the list
6. Open any source by clicking twice on it

10.5.7. Search for a database product

The sub-menu "database products" contains three items: All, Favourites and Query. The list of "database products" is presented in alphabetical order.

In order to see the "information provider" corresponding to any database product, click on the "source" button in the database product window.

10.5.8. Add a new information provider (or modelling or software provider)

1. Select the kind of provider you want to add in the general menu
2. Click on the "add" button on the tool bar
3. Fill the form with information
4. Save the information with the "save" button. "Name", "activity" and "country" have to be filled before saving data (if not, it cannot be saved)
5. Close the window, either by clicking on the "product" button to see the products managed by this provider, or on the cross at the top right of the dialogue box

10.5.9. Modify an information provider (or modelling or software provider)

1. Select the kind of provider you want to modify in the general menu
2. Open the window of the provider by clicking twice on its name
3. Modify the information. "Name", "activity" and "country" must be filled prior to saving (if not, it cannot be saved)
4. Save the information with the "save" button

10.5.10. Delete an information provider (or modelling or software provider)

When deleting an information provider, the next one you enter will automatically replace it with the same code number. So, when deleting an "information provider", it is essential to delete all "database products" attached to this "information provider", in order to avoid automatic linkage between the new "information provider" and the old

"database products". If this problem appears, opening each "database product" and deleting it with the "delete" button can solve it.

1 Select the kind of provider you want to delete in the general menu

2 Click on the name of the "information provider"

3 Click on the "delete" button of the tool bar

10.5.11. Add a new database product (or modelling or software product)

1 Select the kind of provider you want to add in the general menu

2 Open the provider to which you want to add a product by clicking twice on its name

3 In the "information provider" window, click on the "product" button

4 Click on the "add" button of the tool bar

5 Supply the information required. The field "product name" must be filled prior to saving (if not, it cannot be saved)

6 Save by clicking on the "save" button

7 Quit the window by clicking either on the "source" button to go back to the corresponding information provider or on the cross at the top right of the dialogue box

10.5.12. Modify a database product (or modelling or software product)

1 Select the kind of product you want to modify in the general menu

2 Select the product to modify in the list and click twice

3 Modify the information. The name of the database product cannot be modified

4 Click on the "save button". If you want to cancel the modification, click on the cross on the top right of the dialogue box

10.5.13. Delete a database product (or modelling or software product)

1 Select the kind of product you want to delete in the general menu

2 Select the name of product to delete in the list

3 Click on the "delete" button of the tool bar

10.5.14. Add information about web sites of information providers

The "WEB" button opens a dialogue box that allows you to specify the Website of an information provider. In the dialogue box, enter all the items corresponding to the contents of the Website. Then click on OK.

10.5.15. Add (or modify or delete) the ETIS variables characterising a database product

1 Select the "ETIS" button to add/modify/delete ETIS variables for a given product.

2 Choose first an "ETIS family" in the list at the top at the dialogue box, then an "ETIS sub-family" and then an "ETIS variable" in the third list.

3 Click on the "Save" button to add (or "delete" button to delete) your selection.

4 Repeat operations 2 and 3 until you have added/modified/deleted all the ETIS variables you want

5 Quit the dialogue box and save modifications by clicking on the cross at the top right of the dialogue box.

11. Conclusions

11.1. Bridges Technology: Efficient tools to build up productive transport policy support systems

Bridges set out to fulfil the most demanding software requirements of an ideal ETIS but Bridges technology can also be applied for building up effective support systems in transport related fields (e.g. spatial development or environmental assessment) at various geographic levels (local, regional, national). This is demonstrated by the already available experience of implementing Bridges technology (see Section 11.2).

Bridges technology was designed as a set of highly interconnected tools: communication systems, data models and protocols, specialised transport routines, applications to build intelligent user interfaces, component software routines and stand alone applications etc. Each Bridges tool can work alone and none is indispensable, and optimum implementation will always be through integration with other tools and external applications. This approach is consistent with (and complementary to) current strategies adopted for management support systems and electronic business by leading software companies such as ORACLE and Software AG.

11.1.1. Potential benefits of Bridges technology for different users

At the time of writing (late 1999), Bridges is a productive software technology which has been successfully applied to building of advanced support systems.

Systems developed using Bridges have the following advantages from the end-user point of view:

- They are scalable multi-software systems: supported by the Bridges Communication System, a Windows'NT/98 compatible technology, able to integrate multiple and specialised software applications, in particular sophisticated transport models, database managers and GIS.
- They are open to the integration of any Windows OLE/COM compatible application and component software. Advanced transport models (usually non-interactive applications, often not OLE/COM compatible) can be integrated through a new common data model and format (Bridges GTF), as well as transport database structures managed by GIS applications (Bridges GTF_GIS)
- They are highly productive systems, with a set of Bridges Core Utilities specialised in transport database management (e.g. network oriented database and GIS routines to harmonise information linked to heterogeneous graph structures).
- They are intelligent systems, able to facilitate intermediation between policy questions and outputs resulting from scientific models.

They are user friendly systems, with specific Bridges Core Utilities supplying the place of previously missing or sub-optimal data management and mapping tools needed to develop powerful user interfaces

From a system developer's point of view, Bridges technology has the following impacts:

- The time and cost of building up an advanced multi-software support system is dramatically reduced by Bridges Communication System (CS) (this explains why so many Bridges implementations have been possible even before completion of the research).
- Bridges Network Utilities (NIS) provide a number of independent modules, for instance new graph management tools (e.g. graph editing, matching, analysis etc.) which have already proved to be very efficient to develop precise and topologically rich graphs of Europe. This set of transport oriented database management tools is closely linked to transport modelling (including network assignment routines) and may be the starting point for building interactive models based on the formulation and results of more advanced external models.
- For the first time, Bridges data models and formats (GTF) provide a real possibility for transport modellers to exchange of databases. Currently, heterogeneous data models (even terminology) and individual formats make the data exchange process extremely costly, when it is feasible. This problem is blocking modelling research to some extent, since cross-fertilisation and interaction between models is very difficult (e.g. using the same database and comparison of results obtained with two different models is far from easy)
- Bridges data models and formats linking transport models and advanced (GTF_GIS), as well as the ArcInfo translator, allows use of GIS not only to display modelling results (as a mapping tool) but as a database management tool as well. These capabilities can become especially useful in a data warehouse approach.
- Bridges Directory of Transport Data Sources (DDG) demonstrates the actual feasibility of building simple and highly useful tools, with potential in the transport planning sector. DDG demonstrates the necessity of developing a set of electronic directories (for data sources, models, policy questions, experts and advisors etc.) in the ETIS context.
- Bridges Expert System (ES/DSS) is the first attempt to develop a methodology and, based on it, a software tool able to provide intelligent intermediation between advanced models and end users (decision makers). Complex tasks, such as writing model templates to interpret model results and translate user queries to model parameters, can be carried out relatively easily by system developers.

All considered, already existing experience proves that Bridges technology is an innovative software technology in the following senses:

- It brings new capabilities (e.g. new data models and formats to import data from advanced transport models, a new Communication System to drive many commercial applications without being dependent on any of them, new GIS utilities for network and transport database management etc.)
- It optimises and refines already existing software tools (e.g. mapping tools for transport networks, easier interfaces to key database management utilities, fully personalised interfaces etc.)

11.2. First systems developed using Bridges technology

Many Bridges components are already being implemented by the Bridges partners, in collaboration with other companies and institutions. These experiences cover a variety of areas, from systems actually implemented in institutions and in use by their personnel (e.g. SIMU), systems in the process of being implemented (e.g. ICONgis - see www.mcrit.com/icongis-, BRAX, SIET etc.), acting as support to dissemination drives (Phare Toolbox) or educational initiatives (PIE), systems used in consultancy work by Bridges partners (PSCUT, SCANRAIL) or systems being developed within European research projects (e.g. Pyrenees, ATIS). Each one of these systems is different, uses different Bridges tools and commercial applications, and is at a different level of development. Despite the fact that not all Bridges tools have yet been used, especially the more advanced ones, all these support systems constitute excellent opportunities for testing and improving fundamental aspects of Bridges technology.

Even before the formal end of Bridges research, a number of real applications have been already implemented and work in hand of their users. For example the ICONgis/ EIB for the European Investment Bank to assess large transport infrastructure projects in Europe, and the SIMU/ IMU, Urban Information and Modelling System for the Institut Municipal d'Urbanisme of Barcelona, designed for environmental impact analysis at local level. These two policy support systems and several others under development helped to test software tools and validate the overall approach of Bridges. Even more important, they have been extremely useful helping Bridges technology to start the process of moving from the research laboratory to the of policy makers' and experts' personal computers.

In the context of technology oriented research such as Bridges it is always important to remember that a Policy Support System, or a more general Decision Support System (DSS), such as PJGIS or SIMU, is a software system under control of one or many decision makers that assists but does not replace them. Therefore, the development of a decision support system is about people, not about computers. Although computers and software play an integral role in the DSS world, the study of DSS is about how people think and make decisions. The definition and implementation of a DSS must integrate future users as much as possible, since for them a DSS represents both a challenge and an opportunity to improve their working processes. In any case, a DSS will induce organisational changes which cannot be successful unless they are clearly perceived and desired from the outset. Recent developments in the DSS field tend to integrate the multiple decisions being taken by the institution, so they become Organisational DSSs. A successful ODSS will be a participative rather than a mandatory process.

Because of these user and institutional requirements, only decentralised and highly interconnected modular technologies can support a DSS. Moreover, only modular technologies are flexible enough to be updated and improved according to rapidly and unexpectedly changing conditions for both information and technology.

One important implementation advantage of Bridges technology is that being 100% financed by EC it is royalties free for all EU institutions (and others with EC agreement), and therefore an unlimited number of users can be connected to systems developed on Bridges with no licensing fees. For large institutions this advantage is crucial at least in the short term to stimulate their interest to be involved in the

development of their own support systems, a process which anyway would require a serious organisational effort. The following list exemplifies the on-going experience in building up friendly and open multi-software systems based on Bridges technology (in some cases including decision support capabilities, in other cases just modelling and/or information):

1. ICONgis in the European Investment Bank (Transport Infrastructure Assessment, May 1999-October 1999)
2. SIMU in the Urban Planning Department of the City of Barcelona (Environmental Impact, January 1998-October 1999)
3. Phare Toolbox for DG7/Phare Countries (Dissemination of data, formulation and results of traffic forecast models, July 1999- end planned November 1999)
4. BRAX in the Barcelona Regional Agency (Urban Development linked to Public Transport, starting October 1999- end first phase February 2000)
5. PSCUT for JAE Junta Autostradas de Portugal (Mapping/Strategic Modelling, January 1999-July 1999)
6. SIET for the Metropolitan Plan of Barcelona (GIS/Strategic Transport Modelling, February 1998- January 2000)
7. SCANRAIL-Denmark (GIS/Transport Modelling links through Bridges ArcInfo translator, July 1999)
8. ATIS (Alpine Policy Information System demonstrator) for DG7 as a pilot case-study for ETIS (December 1999)
9. Pyrenees Information System (for ASSEMBLING research, 4th EU FP, October 1999- July 2000)
10. PIE GIS/Mapping system for educational purposes (Educational Software Program, Catalonia). Free GIS/Mapping for 2,000 schools (500,000 students between 16-18 years).

These applications (for further information please visit www.mcrit.com) demonstrate the feasibility of using Bridges technology to build up advanced policy support systems in transport or other closely related policy fields, as well as their productivity.

11.3. Looking ahead: a never-ending research

The Bridges research effort, by its very nature, could not reach a final, definitive conclusion. Each one of the systems mentioned requires the improvement of existing tools and even, in same cases, the development of new ones. For example, the "bridges" between transport models and Virtual Reality and 3-Dimensional applications, which were not part of the original Bridges research because their limited interest for European transport planning, are more important when developing support systems at urban scale. Other “bridges” with advanced modelling tools such as GAMS, TRIO, SPSS etc. will also be of greater interest and are expected to be developed as required by new applications). More importantly, many advanced Bridges technologies (e.g. GTF format for data exchange, and Expert System/DSS) have not yet been applied in real cases. It is expected that many of these more advanced technologies will be taken forward in further research activities.

Finally, the development of a “bridge” between Intranet multi-software systems build with Bridges and Internet is the area of most interest from a software development point of view. In the context of ASSEMBLING, another 4th Framework Programme project, which aims to develop a pilot Internet executive support system for European transport policy-makers, this is currently being explored. Once achieved, the integration of the results of both projects may provide for an optimum use of forefront software innovation to support a friendly access (both Internet and Intranet) to advanced transport information and modelling systems.