



OPUS



Optimising the use of Partial information in Urban and regional Systems

Project IST-2001-32471

WP6: Database Systems

Title : Generic Structures and Functionality for Support of Statistical Models in Statistical Databases –
Implementation Report on Using Information from Statistical Models

Creator (Author): Andrew Westlake Survey & Statistical Computing
Rajesh Krishnan Centre for Transport Studies

Contributor :

Identifier : Deliverable D6.2 of project IST-2001-32471

Status : Final version of programme deliverable

Type : Report

Version : 3.3 – Final

Date.Created : 17 June 2005

Date.Modified : 12 March 2006

Submission Date :

Subject.Category

Subject.Keyword

Source

Relation

This header section draws on the e-GMS structure for document meta-data developed by the UK e-Gov initiative.

Rights.Copyright

The Opus Project

Contract Date : April 2003

Publisher (Project Coordinator) : Imperial College London

Contact Person : John Polak

Address : Centre for Transport Studies
Department of Civil and Environmental Engineering
Imperial College London (South Kensington campus)
London SW7 2AZ
United Kingdom

Telephone : +44-(0)20-7594.6089

Fax : +44-(0)20-7594.6102

e-mail : j.polak@imperial.ac.uk

Consortium : CTS, TFL, KATALYSIS, ETHZ, FUNDP, PTV, SYSTEMATICA, WHO, MINNERVA, SURVEY & STATISTICAL COMPUTING, OXFORD SYSTEMATICS

TABLE OF CONTENTS

Technical Abstract	1
Executive Summary	2
1. Introduction and Framework	3
1.1 About OPUS	3
1.1.1 Background	3
1.1.2 Objectives of the OPUS project	4
1.1.3 Motivation	5
1.1.4 Subject areas	6
1.2 OPUS Project Work Package WP6	6
1.2.1 Objectives	6
1.2.2 Description of work	6
1.2.3 Deliverables	7
1.2.4 In Practice	7
1.3 Objectives of Deliverable D6.2	8
1.4 Structure of the Deliverable	8
2. Introduction	9
2.1 Modelling with Opus	9
2.2 Communication about Models	9
2.3 Models in the Opus Methodology	11
2.3.1 Model Structure	11
2.3.2 Bayesian Approach	11
2.3.3 Models and Models	12
3. Results from the Opus Methodology	14
3.1 Why do we use models?	14
3.2 The form of Results from Models	14
3.3 Results from Opus Models	14
3.4 Provenance and Reliability of Results from Models	15
3.5 The Interpretation of Synthetic Data	17
4. The Role of Meta-data	18
4.1 Meta-data as Audit Trail	18

4.2	Meta-data in Opus.....	18
4.3	Relationship to other software.....	19
4.4	Users.....	19
4.5	Presentation Functionality.....	20
5.	The Stat Model specification _____	21
5.1	Introduction.....	21
5.2	Overall Model	21
5.3	Main Components.....	21
5.4	Model Completeness	23
5.5	Links between Models	23
5.6	Relationships in Statistical Models	24
6.	Documenting Statistical Models _____	25
6.1	Introduction.....	25
6.2	Variable and Parameter Elements	25
6.3	References and Expressions	26
6.4	Relationships.....	26
6.4.1	Principles.....	26
6.4.2	Forms of Relationship	27
6.5	Distributions	28
6.6	Knowledge in Model States	29
6.7	Results from Model States.....	30
6.8	Data in Model Fitting	30
6.9	Records of Model Fitting	31
6.10	Other components.....	32
6.11	Other Meta-data Issues	32
6.11.1	Documentation of Meta-Data Objects.....	32
6.11.2	Structures for Data used by a Model.....	33
7.	Working with Stat Model Instances _____	34
7.1	Introduction.....	34
7.2	Working with XML documents	35
7.3	Construction of model instances	35
7.3.1	Using a generic validating XML editor.....	35
7.3.2	Using a specialised XML editor	36

7.3.3	Creating Mathematical specifications	37
7.3.4	Working with modelling applications	38
8.	Presenting Stat Model Examples	40
8.1	A Model Example	40
8.2	Generalised listing with style sheets.....	40
8.3	Graphical display of model relationships	42
8.4	Comparison of models	44
8.5	Model fitting process	44
8.6	Summary	45
9.	Implementation Options	47
9.1	Requirements	47
9.1.1	Functionality Requirements	47
9.1.2	Soft requirements	47
9.2	Technology options.....	48
9.2.1	Graphical Display of Model Graphs	48
9.2.1.1	XML Graph representations	48
9.2.1.2	Dynamic display of StatModel XML.....	49
9.2.2	Display of mathematics.....	49
9.2.3	Graphical display of statistical properties of synthetic data.....	50
9.2.4	Technology infrastructure for the website	50
10.	WP6 Proof-of-concept site	51
10.1	Objective of the site.....	51
10.2	The Site Map	51
10.3	Sample Screen Shots from the site	53
10.3.1	WP06 Site Homepage	53
10.3.2	StatModel XML Loader Page	54
10.3.3	Model Selection Page.....	54
10.3.4	Model Display Page.....	55
10.3.5	Different Layout Algorithms.....	55
10.3.6	Sub-graph Display	56
10.3.7	Model Fit Process Display	56
10.3.8	A Sample R Graphics page	57
10.4	Re-usable components	57
10.5	Continuing Work	57

11. Further Work	58
References	59
Appendix 1: Graph display conventions	61
Colours for Objects	61
Shapes for Nodes	61
Links	62
Appendix 2: Updating the Presentation Stylesheet for StatModel	63
MathML namespace.....	63
Link to MathPlayer.....	63
Model Bookmarks	63
ID Bookmarks	63
Use existing XML for MathML in MathMLExp nodes	64
Make hyperlinks open a new window	64
Appendix 3: A StatModel Example	65
A simple StatModel XML document	65
Formatted model listing	67

TABLE OF FIGURES

Figure 1 Analyst and Client in a Simple Statistical context	10
Figure 2 Communication in a complex context	10
Figure 3 Meta-data for Synthetic Information	17
Figure 4 Package structure of the UML model	21
Figure 5 Main components of a model	22
Figure 6 An Influence Graph	24
Figure 7 Variables and Parameters	25
Figure 8 References and Expressions	26
Figure 9 Relationships	27
Figure 10 Distributions	28
Figure 11 Model State	29
Figure 12 A model fitting step	29
Figure 13 Data in Model Fitting	30

Figure 14 Recording Model Fitting	31
Figure 15 Inheritance from SMObject	32
Figure 16 Information about a model in a StatModel document	34
Figure 17 Creating StatModel documents	36
Figure 18 Constructing a StatModel instance in Authentic	37
Figure 19 Influence diagram	40
Figure 20 Relationship Listing from the Example Model	41
Figure 21 Relationship Listing (partial) from the WP11 model	41
Figure 22 Full influence graph in browser	42
Figure 23 Data generation sub-graph	43
Figure 24 Model core sub-graph	43
Figure 25 Model with differentiated estimates	44
Figure 26 Multiple processes	44
Figure 27 Conceptual map of the Opus Dissemination web site	52

TECHNICAL ABSTRACT

Implementation Report on Using Information from Statistical Models

This deliverable D6.2 is a result of Work Package WP06 of the OPUS project. Work Package WP06 has as title: “Database Systems”. Its original objective was to explore ways to extend statistical databases to support statistical modelling, with associated meta-data. The ideas proposed were to be implemented in the context of the ‘LATS Database’

However, as already reported in deliverable D6.1, the LATS database does not exist in the form envisaged when the project proposal was written. Instead, we have Romulus, which is based on Nesstar, which is a Federated Database system for the dissemination and analysis of statistical data and results. This uses DDI for meta-data about datasets, with some potential extensions. Nesstar is also used by ETH for Zürich data, where they have proposed various extensions for handling transport-related data structures.

There is no proposal to extend either of the Nesstar implementations to include Opus-style modelling within the dissemination systems, as was originally envisaged. However, there is the intention to use the Opus methodology outside the system, and to then include synthetic (simulated or enhanced) datasets, as well as other results derived using the Opus methodology. So, for the meta-data and database contributions from WP06 we concentrate on the support of the use of the synthetic data and other information obtained from the statistical model.

The term we use for this functionality is *Provenance and Reliability*.

The source of most of this is the meta-data that describes a statistical model and that records (like an audit trail) the processes used to arrive at the posterior distributions in the final state of the model.

In this report we present the progress of the implementation of these ideas. We present a major revision to the original structural model (in D3.1) for meta-data about statistical models. This model (called StatModel) is implemented in UML, and from that a set of XML Schema (XSD) files have been generated to control the construction of instances of this meta-data as XML files. We present in reasonable detail the structure and assumptions of the structural model, and also discuss the processes for capturing information about particular statistical models.

We then outline and illustrate the software design and components of the platform that constitutes the software system built for WP6, which displays Provenance and Reliability information from the meta-data instances produced as a part of other work packages.

EXECUTIVE SUMMARY

Implementation Report on Using Information from Statistical Models

This document is Deliverable D6.2 of the Fifth-Framework project (FP5) OPUS. The OPUS project aims to develop and demonstrate statistically sound methods of combining datasets, where each provides partial information on a single complex of underlying variables.

As discussed in deliverable 6.1, the development of the LATS database environment did not follow the path anticipated in the original project specification for Opus. As a result we have changed the focus of WP6 and, rather than aiming to implement statistical methodology within the existing database system, we focus on functionality to support of the correct use and interpretation of the results of statistical modelling, using information from the meta-data that records the structure and fitting of a statistical model.

This functionality we refer to as *Providence and Reliability* information, and this comes from the meta-data that records the specification and fitting of the statistical models. A general structure for this meta-data was specified in deliverable 3.1, and has since been updated – we review the main components here.

Although the focus of the work package has changed, the structure of the objectives stays much the same, and this deliverable reports on the process of implementing facilities to provide this functionality for use in the contexts of the test case and feasibility study work packages.

In this deliverable we report on:

1. Refinement of the model for meta-data about statistical models, and originally proposed in deliverable 3.1
2. The process for creating XML Schemas and other tools that support the creation of XML documents that record actual instances of meta-data about statistical models
3. The implementation choices made in choosing a suitable environment for providing this functionality (building of proposals from D6.1)
4. The functionality implemented to support use of the meta-data for the exploration of model specifications and results.

1. INTRODUCTION AND FRAMEWORK

1.1 About OPUS

1.1.1 Background

In many areas of transport there are a range of different types of data, all bearing on the same issue of interest, but with different variables, reliabilities and biases. It would be invaluable to be able to combine these different and overlapping types of data to improve our representation of the transport factors of interest.

As an example, the analysis of fatal crashes, can be expanded by integrating additional types of data to improve the reliability of the fatality attribute data.

Opus is designed to explore Bayesian approaches to doing this and other tasks. The objective is to develop an appropriate methodology, and demonstrate its initial application to a number of transport and related issues to determine if the overall approach is effective and workable.

OPUS is a large information management research project, supported by Eurostat as part of the European Commission's Information Society Technologies (IST) Programme. The overall aim of the OPUS project is to enable the coherent combination and use of data from disparate, cross-sectoral sources, and so contribute to improved decision making in the public and private sector within Europe. The research is focused on developing an innovative methodology, incorporating statistical and database systems. Transport planning is a prominent example of a topic that uses multiple sources of data, and will be the main test case for OPUS, but the cross-sectoral nature of the research will be demonstrated through the inclusion of an application in the field of health information as another example.

To meet the needs for comprehensive information on socio-economic systems such as urban and regional transport planning, and in the health services sector, data from diverse sources (e.g. conventional sample surveys, census records, operational data streams and data generated by IST systems themselves) must be combined. There is currently no appropriate developed methodology that enables the combination of complex spatial, temporal and real time data in a statistically coherent fashion. The aim of the project is to develop, apply and evaluate such a methodology. OPUS will develop a general statistical framework for combining diverse data sources and specialise this framework to estimate indicators of mobility such as travel patterns over space and time for different groups of people. The project will undertake pilot and feasibility study applications in London, Zurich, Milan, and on a national level in Belgium. Methods for extending the framework to information aspects of the health domain will also be investigated.

The benefits of OPUS will be:

- Improved estimation of detailed travel demand, using all available information;
- Avoidance of simplified combination of data that can give erroneous estimates;
- Indicators of data quality, to provide guidance for new data collection;
- A framework for managing data from rolling survey programmes;

- Better understanding of the role of variability and uncertainty in results and models;
- Avoidance of confusion from different, apparently conflicting, estimates of the same quantity;
- A generalised methodology for other domains of interest.

The participants in the OPUS project are as follows:

Research Organisations

- CTS (Centre for Transport Studies, Department of Civil and Environmental Engineering, Imperial College London), United Kingdom – Lead Partner
- DEPH (Department of Epidemiology and Public Health, Imperial College London), United Kingdom
- ETHZ (Institut für Verkehrsplanung, Transporttechnik, Strassenund Eisenbahnbau), Switzerland
- FUNDP, Transport Research Group (Facultés Universitaires Notre-Dame de la Paix), Belgium

Practitioners

- Minnerva Ltd., United Kingdom.
- Survey and Statistical Computing, United Kingdom.
- Katalysis Ltd., United Kingdom.
- PTV AG, Germany
- Systematica, Italy.
- Oxford Systematics, Australia: Peer Reviewer

Public Bodies

- Transport for London (TfL), United Kingdom.
- World Health Organisation (WHO), Italy.

1.1.2 Objectives of the OPUS project

To meet the needs for comprehensive information on socio-economic systems such as urban and regional transport planning, and in the health services sector, data from diverse sources (e.g. conventional sample surveys, census records, operational data streams and data generated by IST systems themselves) must be *combined*. There is currently no appropriate developed methodology that enables the combination of complex spatial, temporal and real time data in a statistically coherent fashion.

The overall aim of the proposed project is to develop, apply and evaluate such methodologies, taking as a specific case study the transport planning sector. The specific objectives of the study are:

- To develop a generic statistical framework to enable the optimal combination of complex spatial and temporal data from survey and non-survey sources. This framework will specify how to optimally estimate the underlying population parameters of interest taking into account the structural relationships between the different measured data quantities and the sampling and non-sampling errors associated with the respective data collection processes. It is envisaged that the

framework will be broadly Bayesian in nature. The framework will make no specific assumptions regarding the particular structural and sampling/non-sampling errors and will thus be relevant to a wide range of application domains.

- To apply the generic framework within the field of urban and regional transport planning. This will involve the definition of specific structural relationships amongst measured quantities and the characterisation of sampling/non-sampling errors, based on domain knowledge from the field of transport planning.
- To develop the necessary database and estimation software to enable the application of the statistical framework in a number of case study areas.
- To undertake a major pilot application study in London, focusing on the derivation of indicators of the mobility and the performance of transport policy measures.
- In parallel, to investigate the feasibility of applying the framework and methodologies developed both in other transport planning contexts and in other proximate domains, specifically environmental management and social statistics.
- Based on the experience gained in the pilot application and the feasibility studies, to evaluate the performance of the proposed methods and to define the scope and approach for wider applications in relevant domains including environmental management and health care.
- To disseminate the results to the relevant academic and practitioner communities.

1.1.3 Motivation

OPUS addresses the situation in which the analyst must combine data from a variety of different data sources to obtain a best estimate, or a fuller understanding, of a system. Such a situation can arise for a number of reasons including:

- No single source contains sufficient information by itself; or
- Multiple sources naturally arise (e.g. through observations at different levels of spatial or temporal aggregation or by means of different survey methods), resulting in a need to reconcile potentially conflicting estimations; or
- The need to update or transfer an existing set of data and parameter estimates when additional information becomes available.

Problems of combining data from different sources to produce consistent estimates of underlying population parameters arise in many fields of study including environmental monitoring, epidemiology and public health, earth observation, geographic information and navigation systems, transport and logistics, and economic and social statistics. Although the risks of using *ad hoc* combination rules and procedures are well understood, there are nevertheless many examples from practice in which just such approaches are still used. This reflects the fact that, although relatively straightforward methods exist for simple cases, there does not exist a coherent and well developed set of applicable methods capable of dealing with the full range of data combination problems, including factors such as:

- Data sources that provide both direct and indirect information on the relevant population parameters
- Data that are presented at different levels of aggregation
- Data sources with differing levels of statistical precision or user confidence
- Data that overlap, but that may provide different or conflicting information
- Gaps in the data observations

- The issues raised by the aging of sample survey data and the consequent need for updating
- Accommodating the updating sources
- The effect of sampling and non-sampling errors (including survey non-response and other sources of missing data)
- The opportunities presented by new data streams from IST systems

The key scientific objective of the project is to develop a generic statistical framework for the optimal combination of complex spatial and temporal data from survey and non-survey sources. The framework will be sufficiently abstract to be applicable to a wide range of potential domains.

Associated with this overall objective is the need for a suitable representation of the statistical meta-data that is used for the specification and application of such a framework. That is the immediate objective of this report.

1.1.4 Subject areas

OPUS provides a generic approach but, in each case, it is necessary to make this approach specific to the particular area of interest (whether the area is geographical or topical in nature). A particular test-bed is transport in London, but studies will be made for transport in Belgium, Switzerland, and Italy, as well as health studies.

1.2 OPUS Project Work Package WP6

This section summarises the specifications for this work package, as taken from the project proposal.

1.2.1 Objectives

- *Building on the work done in WP3 and on the previous design and implementation work for the LATS 2001 project in London, to establish designs and implementation criteria for supporting modelling within statistical databases.*
- *To undertake the implementation of the database and meta-data software in support of WP7 and the applications in WP8 and WP9.*

Note that this implementation has to be done in the context of the pilot applications in London and Zurich, (within WP8 and WP9) but we will also investigate whether it is possible to do an implementation in a standard generic environment. The former will be sufficient for supporting work packages 6 and 7, but the latter would be more valuable in that it would be more easily transferred (and extended) for other domains. The database environment plus the modelling software would provide a more general-purpose modelling environment. However, we are well aware that producing general-purpose facilities is very difficult, so, given the time and resources allocated to this work package, we will only attempt this generalisation if a particularly straightforward solution presents itself.

1.2.2 Description of work

The work programme will consist of two inter-related activities.

First, extensions to the existing object model developed for the LATS 2001 project in London to be more generic and to provide support for and an interface to a separate modelling component. This will include enhancement of the LTS model to provide the meta-data extensions de-

veloped in WP3, and design of suitable functionality (including interfaces) to support modelling. This will be done in a more generic way, so that the model is applicable more widely than just in the LATS 2001 (or even transport in general) context

Second, the design and implementation of a test database environment to support the modelling software to be developed by WP7. This implementation has to be done in the LATS 2001 project for London, but this will also include study of whether it is possible to do the implementation in a standard generic environment.

1.2.3 Deliverables

D6.1 Report on the Database Structures and Functionality For Generic Support

D6.2 Report on the Implementation of Modelling Support in Statistical Databases

D6.3 Database System Enhancement

1.2.4 In Practice

The proposal for the Opus project was written shortly after the completion of a design report for a Statistical Database for the results of the 2001 London Area Transport Survey (see [West01]). This report envisaged the construction of a specialised database system.

After a more detailed investigation TfL decided that the full implementation of a new system was neither economical nor practical, and instead have constructed a system (called *Romulus*) which is based on the dissemination package Nesstar¹. This is the system that will be used to host suitable results produced by WP8 (the London Test Case), and so is the target system for WP6. WP9, the Zürich Test Case is also to be used with a system constructed using Nesstar, so the results of WP6 will be usable with both test cases.

This situation has been discussed already in D3.2. We concluded there that the focus of WP06 had to be changed, because the direct inclusion of modelling methodology and meta-data in the TfL statistical database is no longer envisaged. Instead, we have decided to concentrate on issues related to making the results of modelling available in a statistical database context, including synthetic data. Because any information that is fed back into the statistical system will be a result of the Opus methodology, we have decided to focus our efforts in WP6 on supporting the use and understanding of this information. This includes discussion of the way in which meta-data about the model fitting process can be used to inform users' of the model results.

We focus on the broad requirements for making use of the results of the Opus methodology (or other statistical methodology) in association with statistical databases, and on the overall architecture needed to support those requirements. This will be done alongside the existing statistical system, using the same technology where appropriate, and implemented in such a way as to appear as seamless as possible for the user of the statistical system. A specific objective of this enhancement will be to implement new facilities in a way that demonstrates their usefulness and facilitates any later implementation within the Nesstar system.

¹ The Nesstar product was developed under a number of EU framework projects, including the Nesstar and Faster projects. Commercial exploitation is being undertaken by Nesstar Ltd, a spin-off company hosted at the University of Essex – see www.nesstar.com.

We refer to this problem area as *Provenance and Reliability* with the implications of this term being expanded in later sections.

1.3 Objectives of Deliverable D6.2

As outlined above and discussed in detail in deliverable 6.1, the development of the LATS database environment did not follow the path anticipated in the original project specification for Opus. As a result we have changed the focus of WP6 and, rather than aiming to implement statistical methodology within the existing database system, we focus on functionality to support of the correct use and interpretation of the results of statistical modelling, using information from the meta-data that records the structure and fitting of a statistical model.

This functionality we refer to as Providence and Reliability information, and this comes from the meta-data that records the specification and fitting of the statistical models. A general structure for this meta-data was specified in deliverable 3.1.

Although the focus of the work package has changed, the structure of the objectives stays much the same, and this deliverable reports on the process of implementing facilities to provide this functionality for use in the contexts of other workpackages.

1.4 Structure of the Deliverable

In this deliverable we report on:

1. Refinement of the model for meta-data about statistical models, originally proposed in deliverable 3.1
2. The specification of this model in UML and its conversion to XML schema for documents that record actual instances of meta-data about statistical models
3. The implementation choices made in choosing a suitable environment for providing this functionality (building of proposals from D6.1)
4. The functionality implemented so far.

2. INTRODUCTION

2.1 Modelling with Opus

The Opus methodology is based on statistical modelling, using Bayesian methods to integrate information from multiple sources.

This is a new level of abstraction in many areas (such as transport modelling), as it requires formal specifications of models and the statistical distributions associated with variables and the parameters of model components. These models cover the statistical distributions needed to represent measurement error and respondent variability, the parameters of these distributions and their associated uncertainty (also represented as distributions), in addition to the parameters of the equations and relationships that the term 'model' is usually restricted to in transport planning and analysis.

In transport we have specific needs to integrate multiple, partial datasets, but similar problems arise in many domains, and environmental health is a discipline included in the Opus team. Applications in any area always require domain knowledge, and these require specialised models, approaches and assumptions. The Opus project is about the methodology, not the applications, but we treat specific applications as case or feasibility studies in which we explore the problems – and the outcomes that arise when the generic Opus methodology is applied.

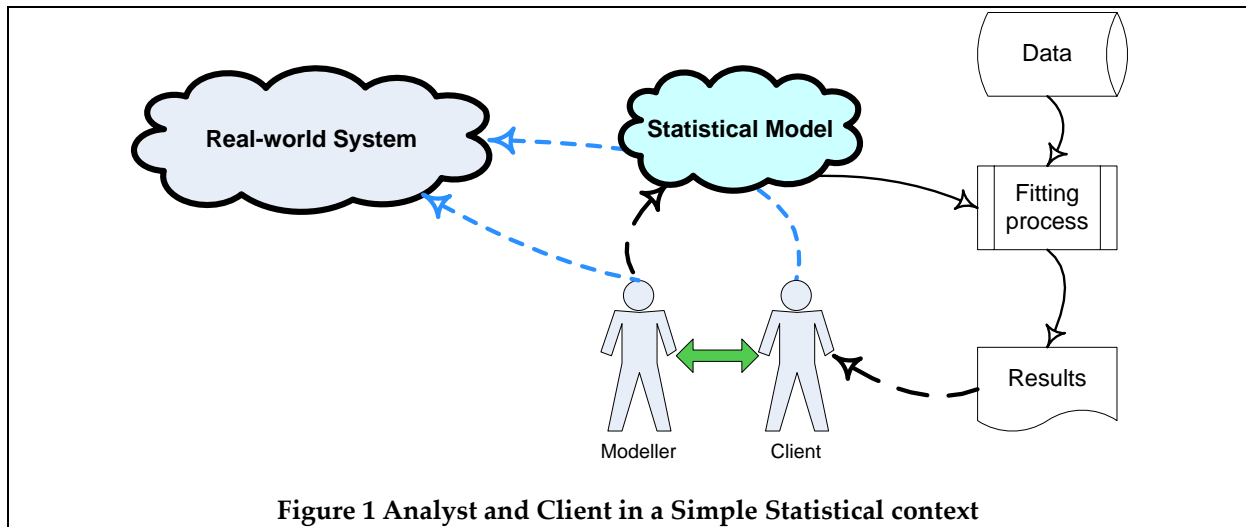
Opus approaches data integration through the use of statistical models of the domain and problem of interest. The formulation of such models is clearly specific to the application domain and the particular objectives of an analysis. Where multiple, disparate data sources contain information about the domain we use Bayesian methods to combine information about the model extracted from the data sources. Users of results based on statistical models should ask questions about the form and quality of the models used, so we have developed a meta-data-based approach that records the structure of the model and the fitting processes used (as a type of audit trail), together with functionality to present this information to users of the results.

This deliverable concentrates on the meta-data structures and functionality of the Opus project, but starts with an outline of the modelling issues, in order to provide context. The structural realisation of this context is carried within the Opus methodology as it is a key feature of the formulation and estimation processes, and needs to be retained with the data and estimation outcomes of Opus applications to secure the full benefits of the approach.

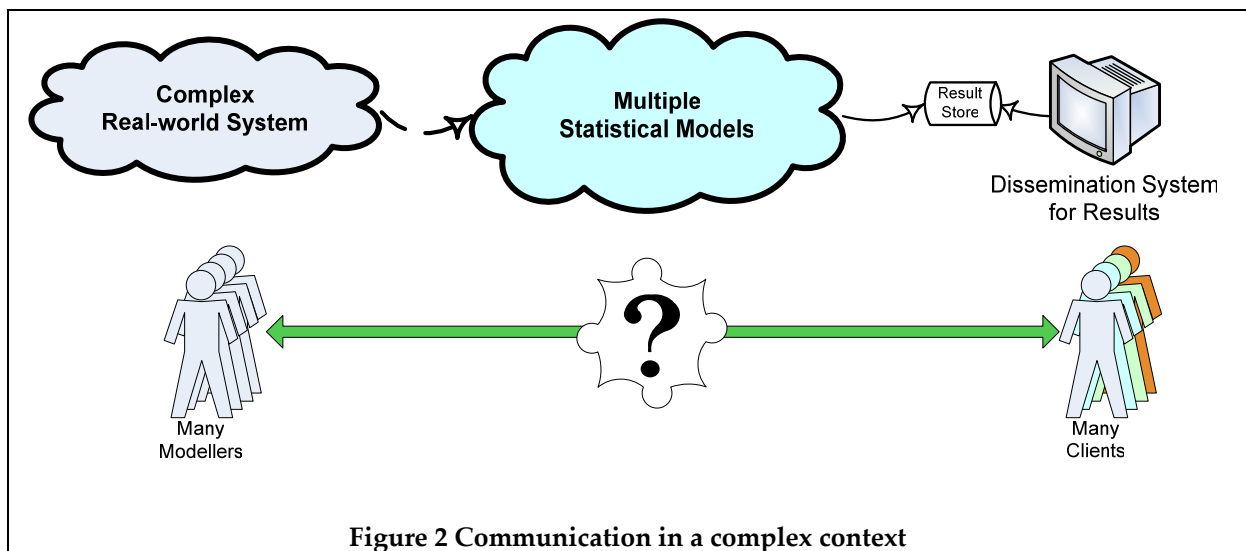
2.2 Communication about Models

Figure 1 illustrates the communication involved when a modeller and client work closely together. The modeller will formulate a statistical model by reference to the real world and in discussion with the client. From the model a fitting process will be initiated which uses data and produces results. These are communicated to the client. The client will relate the results back to their own view of the real world, and may have difficulties in interpreting the results or using them correctly. But the close link between the client and the modeller means that they can both talk to each other and so explore and clarify their understanding of and the details of the meaning of the re-

sults, and the relationship of the results to the data and the fitting process. The model used may be complex, but the communication is simple, provided both participants make the effort to understand each other.



The Opus situation is more complex. Rather than trying to answer a single question we are trying to work with models which give a much more complete picture of a complex system. We expect to have multiple datasets, and will need many models and multiple fitting steps to produce an integrated understanding of a system. Probably many modellers will be involved, all covering different (though probably overlapping) parts of the system. We expect to have multiple users (clients) of different types and skills, and, in this complex situation, results will probably be delivered through some sophisticated dissemination system. All this is shown in Figure 2, and in this type of situation the link between modeller(s) and client(s) is almost impossible to maintain. So, how do clients explore the meaning of results, the intentions and assumptions made by modellers, the details of the mathematical forms used in the models, the reliability and applicability of results?



This problem is familiar to some statisticians, in a different context. Those involved in the provision of access to data for secondary analysis know that the provision of adequate **meta-data** is essential, as enabling information for the secondary analyst. In order to work with data produced by someone else, associated information must be made available. Some of this can be in formal form that can be used by software to fa-

cilitate the input of data sets and the identification and labelling of variables and codes. Other formal metadata can refer to the sample design and other aspects of the data capture process by which the information within the dataset was obtained from the respondents (or whatever data subjects were used). And less formal metadata can describe the concepts that underlie the decisions and choices about the sample, the capture method, the variables and the coding for the data. Various standards have been proposed in this area, and the DDI (Data Documentation Initiative) Codebook standard [DDIA] is used by several Opus partners.

The Opus project is proposing a methodology that involves the use of complex sets of statistical models in order to advance data integration and understanding for complex systems. For this methodology to be acceptable we must be able to bridge the gap in understanding (and confidence) for the clients of the results of this methodology. We address this as a meta-data issue, and have defined both structures and functionality for the storage and exploration of information that originates with the modellers and is made accessible to the clients. We refer to this meta-data system as Stat-Model, and present the structures and functionality we have developed in the remainder of this report.

2.3 Models in the Opus Methodology

2.3.1 Model Structure

The heart of a model in Opus is a specification in mathematical terms (i.e. largely algebraic) of the factors that influence a complex system in the real world. In the context of transport this can include traffic flows, transport survey attributes (or some other aspect of the system being studied, such as exposure) and the way in which they interact in their influence. Of course, the particular factors and form of relationships are in each case specific to the problem and domain we are addressing – but the formal specifications of the statistical attributes of each parameter are always needed for all application system parameters.

All the factors will have statistical distributions associated with them (i.e. they are not necessarily assumed to be fixed), and all the distributions and relationships will have parameters.

All the parameters have prior distributions (representing prior knowledge or uncertainty), which will be more or less informative depending on what experience we can bring to the context and the understanding of the model. It is here that the Opus methodology finds its foundation.

2.3.2 Bayesian Approach

In basic statistical analyses we represent the uncertainty associated with an estimate of a parameter by calculating a confidence interval. For different levels of confidence we obtain different intervals (or limits) and we can represent the set of all limits as a distribution over the possible parameter values. In many cases this will take the shape of a Normal distribution, because the Normal distribution is often assumed for the data.

Although we can represent our uncertainty about a parameter as a distribution, this does not mean that the parameter is a random variable. Rather, it is a fixed property of the reality about which we have collected data, and it is our uncertainty that is represented by the distribution. The interpretation of the distribution is that for any pair of possible values for a parameter the area between them under the uncertainty curve represents the confidence that the interval between the values contains the true parameter value.

We can take the idea further, and represent **any uncertainty** with a distribution. Thus we do not require that an uncertainty distribution is necessarily derived directly from data, we can construct it on any reasonable basis. Of course, it is not sensible to do this without some prior knowledge, or justification, to support the particular choices that we make. Where we **do** have knowledge about the parameter we tend to talk about knowledge rather than uncertainty distributions.

With uncertainty represented in the form of distributions, we can draw on Bayesian Methodology to expand the range of data resources that can be used when working with our models.

As well as uncertainty about the **values** of parameters in the model, we may be uncertain about the appropriate **form** for the model. We can cope with this by introducing additional parameters to control the functional form of the model, additional to those that relate directly to the underlying system.

Very general classes of Bayesian models can be fitted using MCMC (Monte-Carlo Markov Chain) methods [GiRS96]. This is the approach used in the Opus project, and details of the fitting methodology are presented in other papers, available from the project web site.

2.3.3 Models and Models

The term 'Model' is very widely used, and can be confusing because it implies different things to different people. Formally, a model is some abstraction (often in mathematical form) representing part of the behaviour of some real-world system, selected in a particular context for a particular purpose. An often quoted remark, attributed to the statistician Prof. James Durbin, is that *all models are wrong, but some models are useful*.

Models are designed to meet a particular need in a particular context. Thus the form and roles of models can be very different. Some examples may help to show some of the range.

Conceptual Models are an attempt to form a frame of reference for some domain or collection of constructs or concepts. Where concerned with terminology or names (and so sometimes called *Ontological Models*) they are often similar to classification structures. Other conceptual models may be concerned with suitable structures for organising ways of thinking about a domain.

Structural Models concentrate on the objects and attributes that are used to represent information structures. This is necessary for the exchange of information between computer systems, but needs to be accompanied by clear specifications of the intended purpose and use (the *semantics*) of the various elements. Inconsistent interpretation by independent users or implementers working with such a structure is a con-

tinuing concern, unless some enforcement mechanism can be specified and implemented. Structural models can be conceptual, in that they provide a way of thinking about the appropriate structures for some context, or they can be physical, and so present the actual structures needed for some particular system. In contrast, *Process Models* (of which *Data Flow Modelling* is an example) are concerned with the operations that are performed on data objects as they are moved between structures or in response to events. Structural and Process (and related) models for software are often represented using the Unified Modelling Language – UML [UML].

Statistical Models are generally representations of some real system that exhibits variability or unpredictability. They use mathematical relationships to specify the form of dependencies between variables, and statistical distributions to express the variability. Such models can be generic (when they are sometimes described as methods or methodologies), or specific to a particular application (when they will often be instances of the more generic model forms). The term *Graphical Models* is used to refer to a class of models that express dependencies between variables in the form of a graph showing conditional independence. *Bayesian Models* use a formulation of uncertainty about parameter estimates that is based on Bayes' Theorem. For the Opus project we are working with statistical models of specific systems, constructed using the generic Bayesian approach. Some of these models are (in part) instances of Graphical Models.

The term *Transport Model* is used to refer to a class of procedures that are used to estimate information about transport systems that is difficult to observe. For example, 'Route-Flow' models are used to estimate the sharing of traffic flow between possible modes and routes when the demand between origins and destinations is known. Such models are often treated as deterministic, but in reality they are usually statistical models in which variability is ignored. For example, the use of 'least-squares' to estimate the relationship between variables observed with 'error' is only optimal under assumptions of independence, symmetry and constant variance. Many such assumptions can exist within the 'black boxes' that are some transport models. Our preferred approach is to open up such models and to make their mathematical and statistical assumptions explicit.

Models exist at various **levels of abstraction**, and confusion can arise from not recognising the level to which a particular construct contributes, or at which a discussion about the model is taking place. In the Opus methodology we explicitly separate out generalised models (GAPMs – Generalised A-Priori Models) which represent the general knowledge about the nature of relationships and influences within a domain, and contrast them to the specific and detailed models that are used to explore our understanding or knowledge about a specific issue or system.

All these forms of model exist because they are (at least potentially) useful. None of them is right or wrong, though certain forms are more appropriate in certain circumstances. When managing models in Opus through meta-data we focus on statistical models, specifically allowing for different levels of abstraction, but also try to be open to the representation of models that have different objectives and motivations.

3. RESULTS FROM THE OPUS METHODOLOGY

3.1 Why do we use models?

Practitioners are often sceptical about results from models, preferring to rely on results derived directly from a particular dataset. While this attitude is understandable, it does ignore the limited applicability of a particular dataset – to what extent can the results be generalised? – or the biases inherent in particular data collection methods – what do we do when different datasets give different results? In practice, all data analysis involves some form of statistical model, even if this is not made explicit. By making the model explicit we are better able to balance information from different sources, understand biases and so generalise to the whole system.

How, then, do we persuade practitioners that models are valid and useful? The Opus project addresses this through the use of meta-data about the statistical models and the model fitting processes. From a philosophical perspective we argue that there is always a model, so it is better to understand it and be able to criticise it than to pretend that there is no model. However, rather than trying to win a philosophical argument we are concentrating on exposing the qualities of a model so that users can make their own judgements as to the usefulness of model results. We focus on providing information to users about the provenance and reliability of results obtained from a model.

3.2 The form of Results from Models

The end result from application of the Opus methodology is a calibrated statistical model. This is specified in terms of a set of mathematical relationships among the variables and parameters of the model, including components that describe the stochastic variability exhibited by the underlying system. In addition, the knowledge about the model parameters that has been extracted from the evidence available in datasets is summarised in terms of posterior distributions which encapsulate the best estimates and our uncertainty about the parameters.

An experienced analyst, familiar with the methodology, can use the model to extract information about the underlying system, covering estimates of measures of interest, their variability, and the uncertainty associated with these estimates. If dealing directly with the mathematics of the model is seen as too difficult, the implications of the model can be presented in the form of simulated (or synthetic) datasets generated from the mathematical specification. A simulated dataset will generally include variability associated with the underlying system, and can also include variability arising from uncertainty about parameter values.

3.3 Results from Opus Models

The Opus methodology is Bayesian, so all the knowledge lies in the model specification plus the posterior distributions of the parameters. That is, all information about the underlying real-world system that is contained in observed datasets and is pertinent to the model formulation has already been extracted by the model fitting process

into the posterior distributions. In theory it is then sufficient to present just this extracted information (the model formulation together with the posterior distributions) to users. In practice, this will be too complex or impenetrable for most users, so, as with most statistical analyses, other forms of interpretation and presentation will be needed.

Notice that we assume that the mathematical formulation of the model has been determined, and the methodology has been applied to give us the best possible calibration of this model, extracting all possible information from the data sources. Clearly there is a previous process by which the mathematical form of the model is developed and decided upon. This may well use the same methodology as part of an intermediate step (and other methodologies and previous knowledge), but results are always derived from the final version of the model formulation, calibrated in the best possible way. Knowing about the intermediate steps can help to explain and justify the choice of the final version of the model, but all the information about the underlying system is in the model – because, if not, the model would be different.

The central role of the model is valuable because it allow us to generalise, from actual data to all situations covered by the model. All information that we present will be valid information about the model, but will only provide useful insights about the underlying system if the model has a valid (and sensible) structure and is well-determined by the available data. Thus a user of information from the model should reasonably ask about the form of the model, the processes by which it was fitted, and the extent to which conclusions are well-determined.

3.4 Provenance and Reliability of Results from Models

We anticipate the presentation of three forms of information derived from a model.

1. **Conclusions.** Summary reports which provide interpretations of the fitted model, based on the experience and judgement of the author. These will be largely textual, but will include illustrative material and links back to the model.
2. **Estimates.** Presentation of the posterior distributions of quantities of interest from the underlying system. This can be done in terms of summary statistics (particularly means and standard deviations) of the posterior distributions, or of complete distributions, presented as histograms or multivariate contour plots (for example). Note that the distribution represents our uncertainty about the true value of the quantity, so it is important to present this in addition to any point (best) estimates.

Population parameters of direct interest to users (for example, in decision making) will be the primary focus, but these are generally dependent on internal (hyper-) parameters, which are the ones directly adjusted by the fitting process. Estimates can be obtained for any derivable measure on the underlying system, with a corresponding derived posterior distribution.

3. **Synthetic data.** Given the model specification and the posterior distributions, it is possible to simulate observations on data subjects. In this way, we can create synthetic datasets which have the same characteristics as the model. These are much easier to analyse for people used to handling real datasets. It is also pos-

sible to generate data for specific conditions, for example by limiting the impact of abnormal events, focussing on particular subsets of the overall possibilities, or assuming away some uncertainty in parameters.

These three types of information have close parallels with information obtained by more traditional methods. The difference is in the central role of the model in our methodology. Instead of presenting information that is directly derived from a dataset, and which is then inferred to be directly about the underlying system, all our information is mediated by the model. The model serves to balance and explain differences in the results obtained from separate datasets, by requiring that differences in the data collection methods or the response processes are made explicit. It also makes it possible to explore the implications of the model for combinations of circumstances for which no data has actually been observed.

For such results from a model to be useful and usable, the user must have confidence in the model. We must be able to explore and ask questions about the nature and qualities of any fitted model. We thus propose that two additional types of information should be available with all results that are derived from a statistical model.

- 4. Provenance.** Information about the structure and objectives of the model (including its mathematical form), and about the model fitting process (the audit trail). This includes information about the fitting methodology (which will apply across a set of related models), together with the datasets used at the various fitting stages and the contribution of each such stage to the final fit. The latter is particularly important in terms of understanding how well the posterior distributions of parameters have been determined by the fitting process.
- 5. Reliability.** This relates to the posterior distributions of the model parameters. But instead of focussing on estimates of quantities of interest in the underlying system, it focuses on the uncertainty that remains about the model parameters themselves. We explore whether the parameters are well-determined, the source of the knowledge about a parameter (ie prior knowledge or particular datasets), and how well the final model reproduces the datasets used. It is important to distinguish between uncertainty about parameters (which should generally decrease as more data is used or as the model formulation is improved) and variability in observed data that is associated with measurement processes or unpredictable behaviour.

The source of most of this information is the meta-data that describes a statistical model and that records (like an audit trail) the processes used to arrive at the final state of the model. Later sections of this report describe a structure for meta-data about statistical models that includes (potentially) all this information (it is in effect a complete audit trail for all the specifications and stages used to produce results – including specific estimates and synthetic data).

We also need to find ways of presenting this additional information that are accessible and comprehensible for different groups of user. Different types of user will expect answers of different complexity and detail. Some answers can be generic, describing the philosophy behind the Opus methodology and Bayesian modelling, or showing (perhaps in UML diagrams) the outline of the model fitting processes used. Other answers will need to be based on the specific components used in the model

from which the data are synthesised, and further ones will make use of the detailed posterior information about the parameters. The same information may need to be presented in different ways for different types of user.

3.5 The Interpretation of Synthetic Data

Because synthetic data looks like real data, no special facilities are needed to add it to existing data management or analysis systems, or to use it within them. However, synthetic data is different, and to use it effectively (and correctly) the user needs to understand this difference and have access to information about the form and quality of the model from which the synthetic data was created.

The problem is that synthetic data is not real, and its statistical properties are not the same as those of real observations on the underlying system. They are different because they come entirely from the fitted model. The challenge is to guide users to appreciate these differences.

The issue of sample size illustrates the difference. With real data, increasing the sample size increases the amount of information available about the real system being observed. However, with simulated data, increasing the size of the simulated sample only provides more information about the model, not about the real system.

There can be no information in synthetic data that is not already present in the calibrated model from which it was synthesized. All available information about the real system has already been extracted into the model and increasing the size of a synthetic sample merely improves the precision of the information about the model, it does not provide any additional information about the real world.

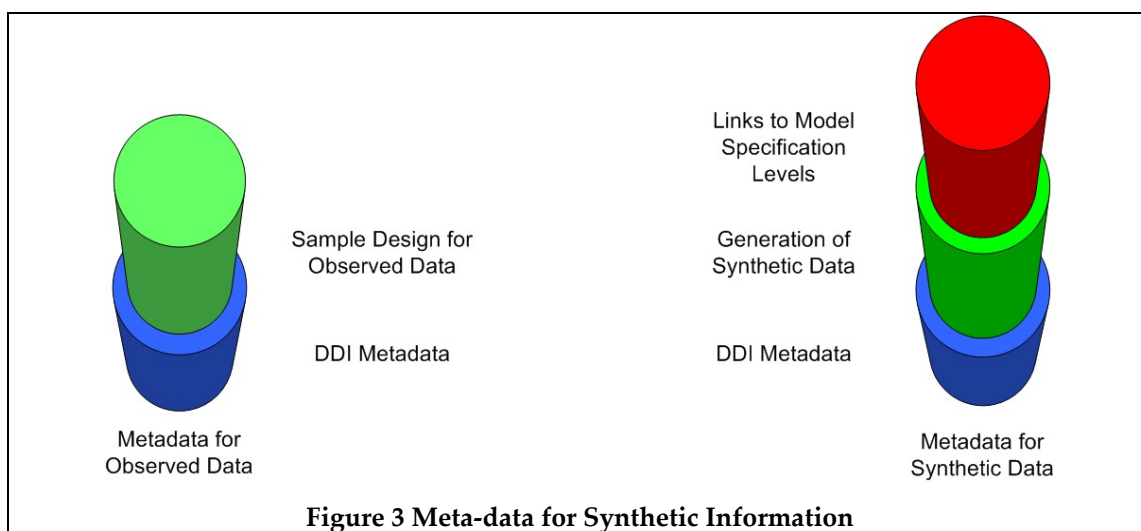


Figure 3 is intended to illustrate the different meta-data requirements for synthetic data (and, in a similar way, for any results from statistical modelling). The file of synthetic data needs to be described, as does a file of real data, and existing structures, such as DDI are quite adequate for this. To understand and interpret the real data adequately an understanding of the data collection process is needed, particularly the sample design. For synthetic data, similar issues arise with the generation process by which the data is synthesised from the statistical model. In addition, the formulation and quality of the model itself should be understood.

4. THE ROLE OF META-DATA

4.1 Meta-data as Audit Trail

Over recent years the concept of meta-data and the recognition of its importance has become widespread in many fields, including transport. However, the general idea of meta-data has many different applications in different areas and so means different things to different people. For example, the Dublin Core proposals [DCMI] (and extensions such as the UK government e-GMS standard [eGMS]) have proved important in the context of resource discovery, especially on the Internet. Related to this is the ISO 11179 standard [11179] for meta-data repositories. Similarly, the DDI Codebook standard for the description of survey datasets has achieved wide acceptance including use by some of the Opus project partners. Several examples of this approach to travel survey data are discussed by Levinson and Zofka [LeZo05].

An alternative thread that has received attention in the statistical domain is that of process meta-data. This is information that describes and documents the processes through which data has passed. This can be seen as providing an *audit trail* so that it becomes possible to discover details about any transformations, adjustments or corrections that have been made to data before it reaches the form in which is published. This approach to statistical meta-data is discussed by Green and Kent [GrKe02] in one of the deliverables from the MetaNet project [MetaNet]. Also from that project, Froeschl et al [FrGV03] make valuable contributions to the concepts underlying statistical Meta-data.

4.2 Meta-data in Opus

In the Opus project we focus on process meta-data, mostly of extensional (formal) form. Our objective is to keep track of the processes that are applied in developing the statistical model from which conclusions are drawn, while at the same time allowing the modeller to record motivations or descriptions.

Details about the meta-data system adopted by the project appear in the following sections, but the main elements are as follows:

- The mathematical specification of the model that is chosen, including all its statistical components
- The model fitting processes that are applied to the model, including all the datasets that are used
- The states of knowledge about the modelled system that is extracted from the data by the fitting processes
- Specifications for the results that are extracted or reported from the final model

The intention is to capture all pertinent information about the model fitting process and link this to any results produced from the model. Any component can have comments or explanations or other textual information linked in (sometimes called intentional meta-data).

However, while the capture of this information is essential, its mere existence is not sufficient. Facilities are needed to present the information in ways that are accessible

to particular groups of user, together with guidance about the types of question that should be asked about the model and the results.

Our objective has been to design a structure (StatModel) for the representation of information about statistical models, together with appropriate functionality for the presentation of that information.

4.3 Relationship to other software

Opus makes use of existing model design and fitting software (such as WinBUGS, R and MLWin), or develops specialised tools for specific problems. It has never had the intention to develop a generic statistical modelling tool (only the methodology for this). So the information that we need to capture and store as meta-data will come from other applications.

Different applications embody different conceptual models of their methodologies and their application domains. Our aim with StatModel is to be sufficiently generic to be able to encompass these different views. However, to support users working within specific domains we do need to think about the mappings from application-specific views to the StatModel view.

Our immediate purpose for the meta-data is to allow users to **explore** the specification of statistical models. However, some developers have suggested a need for a structure to allow the **exchange** of statistical models between applications, and we hope that StatModel may be a contribution to that.

We also hope that applications will want to use StatModel as their native structure for storing information about models and processes. So we allow for structural extensions to the model, both as new generic requirements are identified and so that applications can store information specific to their requirements.

When developing an understanding of a system it can be useful to store models that are incomplete, or that are defined at a level of abstraction above that needed for the computation of results. So the structure must not impose completeness rules unless they are always appropriate. Of course, this implies that applications using the structure must be able to check whether an instance of the structure is complete enough for the purposes of the application.

We are not attempting to produce a complete statistical meta-data system, so assume that meta-data and presentation functionality for other components (such as classification structures and dataset documentation) will be available through external links.

4.4 Users

Many users of statistical results have only limited understanding of statistical models and methodologies. For them, information about the statistical models needs to be presented in ways that relate to the reliability of the results for use in their working context, rather than the more abstract terms of the model itself. Furthermore, different domains build on different conceptual models for the form and description of relationships and dependencies, and an ideal presentation system should work within these.

In contrast, those who actually develop statistical models are more likely to have a good understanding of the more abstract concepts involved. For them a more generic presentation of model information may be adequate, or even preferred.

We thus envisage that significant use of the StatModel approach to support the use of statistical model results in a domain will require a presentation application that is specific to that domain. Such presentation systems are not too difficult to build using web-based methods. As an example, the Nesstar system [Nesstar] has been used in a number of large-scale dissemination projects to present basic statistical results and related materials to groups of users in specific domains.

We have concentrated in creating generic displays that can be used by specialists, and that can also be building blocks for the construction of more specific presentations.

4.5 Presentation Functionality

As discussed earlier, our main use for the information is to give confidence to users of the results of statistical modelling. Since these people will not be statistical specialists the presentations will need to be tailored to the level of understanding of the users, and to their conceptual views of the domain in which they operate. Generic presentations are also useful, but only for specialists or as building blocks.

We take as our model for presentation the Nesstar system. This is a system for building distributed access and dissemination systems for statistical datasets and results. Presentation is through a web interface based on templates and components. Components are provided to give access to any information that is stored within the system, including results derived dynamically from accessible data. But by using the web paradigm the presentation of the results can be customised to the target domain, and any other relevant information accessible on the internet can be incorporated in the presentation.

We have not attempted to construct a general presentation application for the metadata about statistical models. Instead we have concentrated on the development of a number of presentation components that can be used as building blocks for such systems. Most of these components are generic, but some are targeted at the transport domain. These have been implemented within a test environment, a publicly accessible web site which is described later.

We have developed a generic facility for listing all the components of a StatModel instance, through the use of XSL/T stylesheet transformations. This is directly useful to specialists, and provides a model from which appropriate elements can be extracted for more specific listings. A graph display applet is used to explore the influence relationships in the model, and to show the fitting processes used. Mathematical derivations are displayed using MathML [MathML]. Uncertainty distributions are displayed and explored using the graphical facilities in the R statistical system [R] (because this can be used as a service). The specialist transport modelling application Visum [PTV] is being extended to show uncertainty in its transport flow diagrams.

5. THE STAT MODEL SPECIFICATION

5.1 Introduction

The UML model for meta-data about statistical modelling has been significantly refined, updated and extended since we reported in April 2005 in D3.1. It is now referred to as Stat Model. In this chapter we give an overview of the main components of the structural model for Stat Model. Note that this is not a statistical model, but a specification of the data structure needed to store information about actual statistical models.

The model is now developed using the *hyperModel* Workspace tool [hMxml]. This follows the UML standard for the specification of static structures (classes and their attributes), but also contains stereotypes for a mapping to corresponding XML Schema structures, plus the functionality to generate schema (XSD) files from the UML specification. The *hyperModel* system is a workspace that runs within the Eclipse integrated development environment (IDE).

Using UML for the design gives access to various object-based facilities that are not directly included in XML schema. Of particular value are the use of Packages to partition the design, and the use of Inheritance to avoid duplication in the design.

5.2 Overall Model

The overall structure is split into packages as shown in Figure 4. Each package is implemented as a separate XSD Schema file, and the dependencies (shown as arrows) are implemented as 'include' instructions.

5.3 Main Components

The main components of the StatModel structure for a single model are shown in Figure 5, and are elaborated later. Information about multiple models

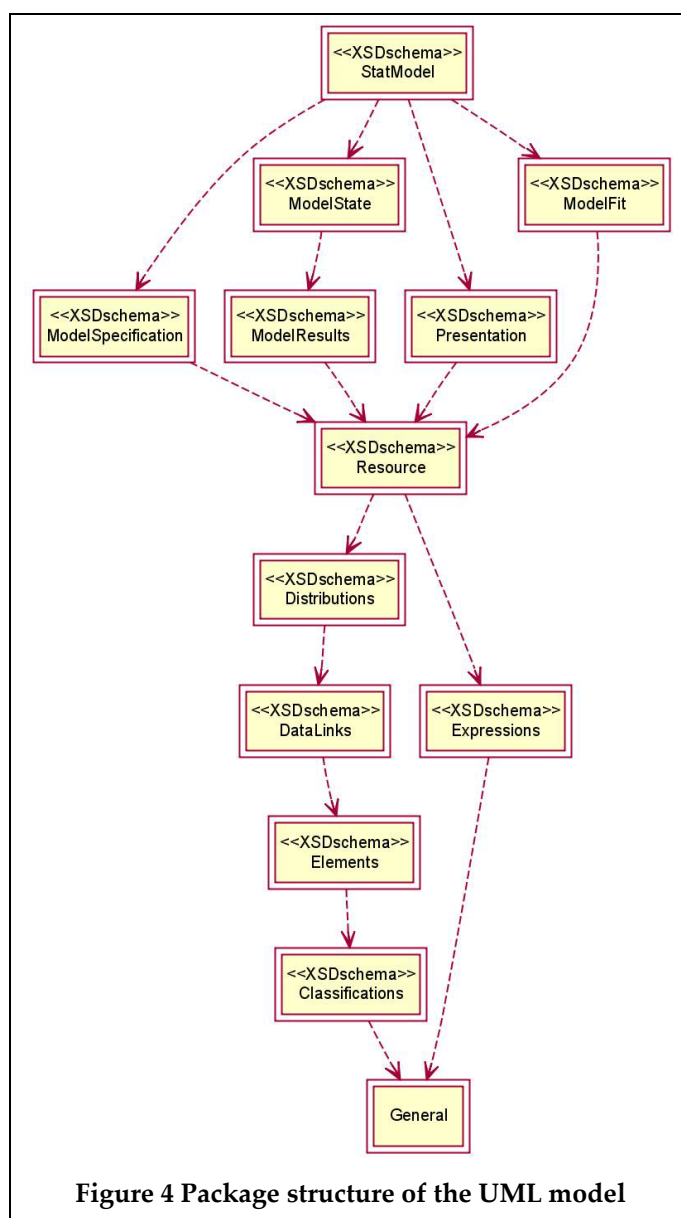


Figure 4 Package structure of the UML model

can be stored together and they can make cross-references.

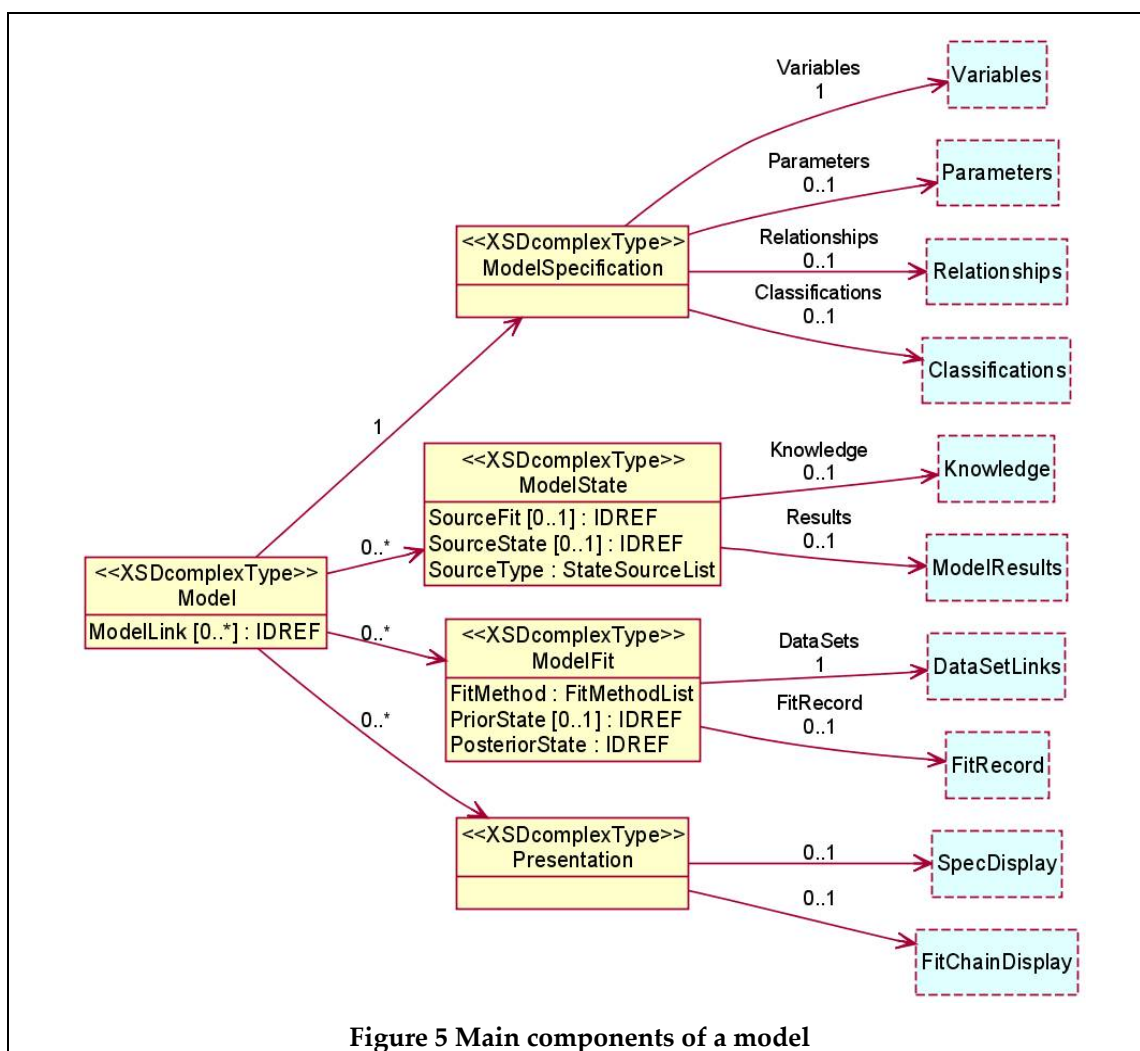


Figure 5 Main components of a model

A *model* must contain a *model specification*, which is where the variables, parameters and relationships which specify the model are stored (the structure of these elements is elaborated later). The variables correspond to the statistical idea of Random Variables, that is they relate to suitable data subjects (for whom actual values may be observable) but we are interested in the stochastic distributions of the values, not the values for individual respondents.

Parameters are properties of the underlying system. They are real (fixed), but they cannot be directly observed. We extract evidence about them from data, but there will always be uncertainty about the true value. This uncertainty is represented by uncertainty distributions, which have the same mathematical properties as probability distributions, but a different interpretation.

Both variables and parameters can be array structures in which all cells are of the same type. Classifications are used to define the dimensions of such arrays.

Relationships specify how variables and parameters influence each other, and can be deterministic or stochastic.

Models can have different purposes (more than one), and different purposes have different requirements. For example, an abstract (or conceptual) model (such as a GAPM) is used mostly to convey the nature of influences between variables, and this is best displayed in a diagram. Such a model will never be the direct specification for

a fitting process, though it may be a source of the conceptual motivation for another, more specific, model.

At the early stages of development of a model it is sufficient to specify variables and parameters at a conceptual level. Their intention must be clear, so that influences between them can be identified, but details of their representation can be left for later. For example, a variable that relates to the income of a respondent should probably make reference to an appropriate definition of income, but does not need to be specific about currencies or whether the representation is exact or grouped. More measurement detail is needed when the details of the relationships are added, as these will include mathematical expressions.

A *Model State* contains knowledge about all the parameters in a model that are not determined by relationships, in the form of their uncertainty distributions. Following Bayesian methods there can be multiple states of the knowledge about a model. Any results derived from a model are based on a particular set of uncertainty distributions, and so can be linked to a specific state.

A *Model Fit* documents a step in which some application is used to extract evidence about the model from data. Such a step usually draws on knowledge from a model state (the Prior state) and always produces knowledge for a new state (the Posterior state). Model fitting processes are thus chains consisting of alternating fits and states, where each state is the output of one fit and the input to the next.

A Presentation component has been added, for storing information about preferred displays for components such as influence diagrams. This will later be extended, including the storage of layouts for individual users.

5.4 Model Completeness

We use this structure to express abstract models (GAPMs) as well as concrete ones. For this we need to be able to store models at different levels of completeness, so the details of the relationship form are optional within a specification. Variables and parameters can be declared but not used.

It is also possible to specify incomplete models. Incompleteness will manifest in the presence of elements which are used but not defined. Of course, these are all needed before a model can be fitted. Any support software should be able to report the location of incompleteness. Model display (certainly influence diagrams) should not depend on completeness. It may be useful to develop some specialised mathematical forms for the representation of more abstract forms of influence relationships.

5.5 Links between Models

Although each model is distinct in terms of its specification and fitting, it may be useful to link to other models which provide input to the formulation of the current one.

It may well be useful to present such information as part of the provenance of the current model. Similarly, another model may provide (prior) information about the distribution of a parameter for a model state.

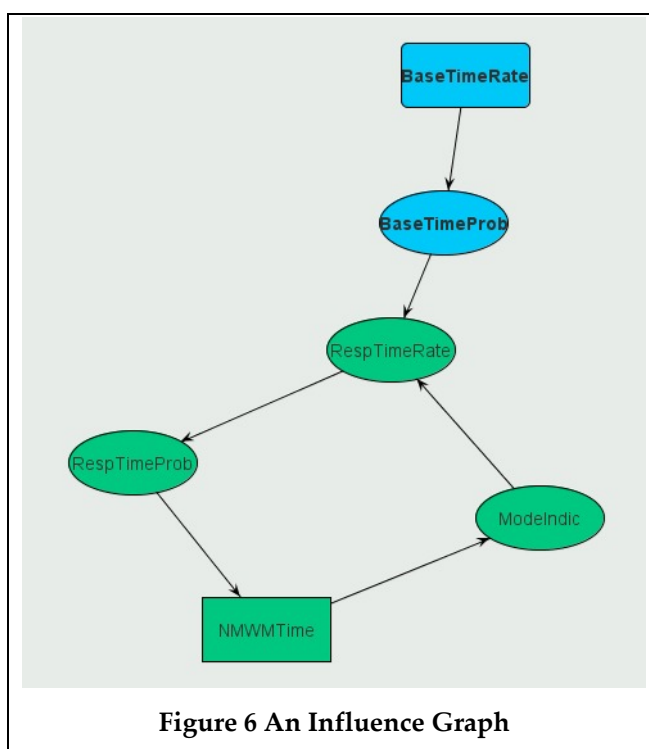
5.6 Relationships in Statistical Models

Relationships show how the various elements of a model depend on and influence each other. The specification of the set of relationships is the essential core at the heart of statistical modelling. This is a highly skilled technical activity that needs to be informed by both statistical ideas and a deep understanding of the domain to which the model is to be applied.

Relationships imply links between the variables and parameters in the model. These can be displayed in an Influence Graph. In this, each element is a node and the links connect from all the input elements of relationships to the corresponding output element. Where the resulting graph is acyclic (does not have loops), the model falls into the class known by statisticians as Graphical Models [Whit90].

A relationship can have multiple inputs. The output of a relationship is always a single element (variable or parameter), and an element can only be the output of (be defined by) one relationship.

All dependent variables must be the output of a relationship with parameters and other variables (in order to specify their derivation or their stochastic properties). Variables that are not so defined are called 'independent' or 'exogenous'. Parameters can also be specified by relationships with other parameters: those that are not are called 'terminal'. Uncertainty distributions must be supplied for them in any model state.



6. DOCUMENTING STATISTICAL MODELS

6.1 Introduction

Finally we arrive at the details of the components of the StatModel structure. This is defined as a UML model (not a statistical model) of the statistical modelling process. The discussion uses UML terms such as ‘package’, ‘class’ and ‘attribute’ to refer to the components.

Figure 4 showed how the model is broken down into packages, and the dependencies between them. Figure 5 has shown an outline of the main components of the structure. Appropriate parts of this are elaborated in the following sections. A full specification of the UML model is being developed as part of an update to deliverable 3.1, and will be available from the Opus project web site.

In the following sections the term ‘model’ refers to the Model component of the Stat-Model structure, and to the statistical model that it represents.

6.2 Variable and Parameter Elements

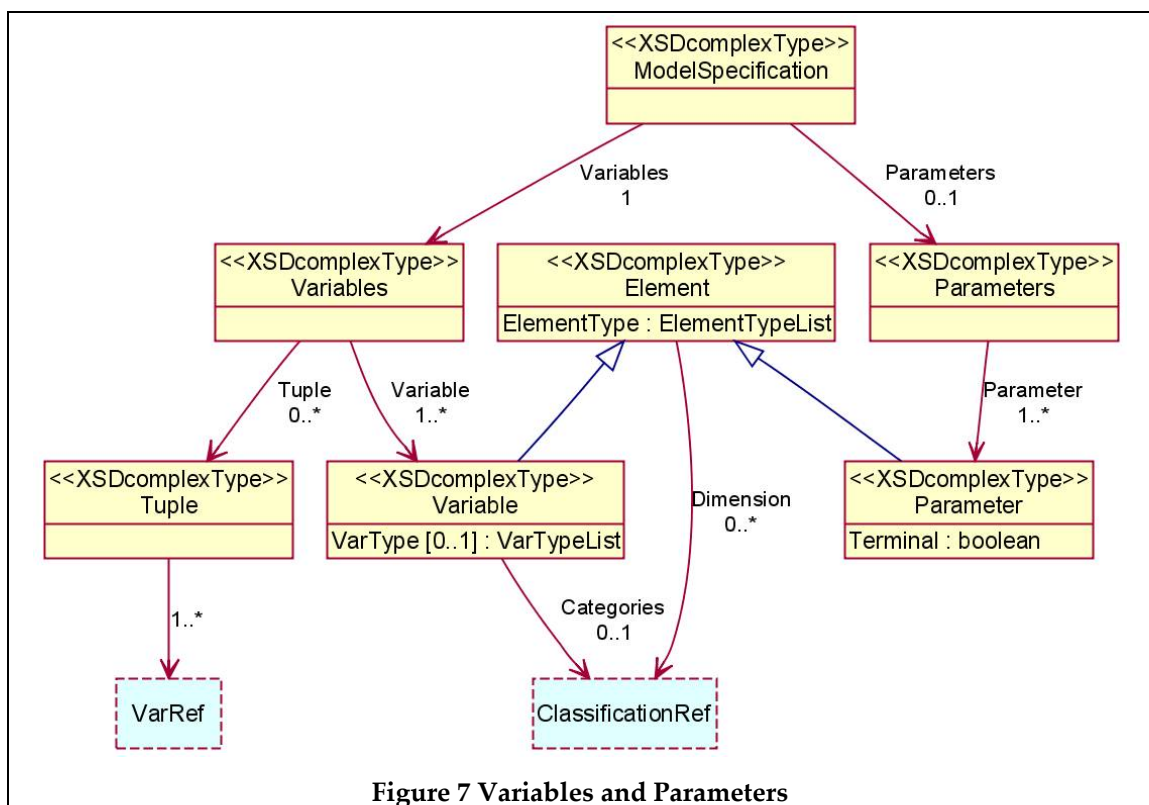


Figure 7 Variables and Parameters

The distinction between variables and parameters has already been discussed. Both are specialisations of the general Element class, so can have a Type attribute which defines their structure (Simple or Matrix) and Dimensions which are based on Classifications. Parameters are always continuous measures (we do not allow for quantum changes in parameters in statistical modelling), but Variables can be of different types, Measures, Categories, etc.

A model must always have at least one variable. Parameters are not required, but their absence is likely to be useful only for models at an early stage of development.

Variables can be grouped into Tuples (the same concept as in relational databases) to show that they relate to the same data entity.

6.3 References and Expressions

References (by Name) to variables and parameters are specialisations the more general Element reference. Where an Expression is needed (as in relationships – to follow) a reference to an element can be used, or a numeric constant, or a full-blown mathematical expression, written in MathML.

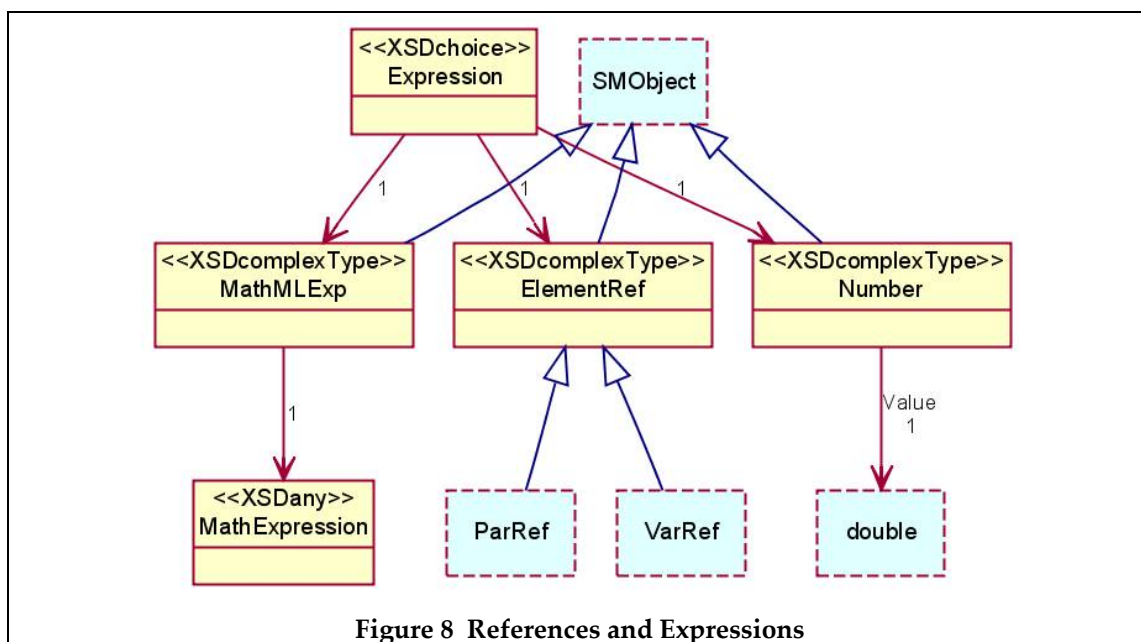


Figure 8 References and Expressions

The SMOBJECT component shown here is the generalisation from which all classes in the UML model inherit, and is elaborated later. It provides a centralised mechanism for associating standard attributes with all classes, such as Name and ID, and allows textual documentation to be associated with any object.

6.4 Relationships

6.4.1 Principles

A relationship lists the input and output elements, and the detailed form of the link between them. The relationship part of a model specification is not required, but a model is unlikely to be useful without some relationships.

Relationships can be Derived or Stochastic. A relationship always has an input component consisting of at least one element (variable or parameter), and generally has one output element. Constraints are implemented as derived relationships which must evaluate to 'True' – the output component is omitted so the constraint applies to the input elements. An example familiar to statisticians is to constrain the set of parameters that represent the differences between the different levels of a factor (when there is a separate overall mean parameter) to sum to zero.

The form of the relationship can be omitted – in the early stages it is often enough to just list the input and output elements without being specific about the form. This can also be useful to document relationships at a more abstract or generalised level.

Where present, the form of a derived relationship is an expression giving the value of the output element, and for a stochastic one it is a reference to a statistical distribution, which in turn will have expressions for its parameters.

All expressions in the form component can only reference the elements listed in the input component. This rule is not enforced by the schemas, but we would expect it to be enforced by a model design application.

6.4.2 Forms of Relationship

Our objective here is to list all the relationships (mathematics) in the model, in a way that:

- is adequate for representing the formulation of the model,
- can be used by other software that needs to work with the model,
- can be presented to users who need to explore the form of the model.

Components

- Input – a set of variables and parameters. This can be empty for a stochastic relationship where all the parameters are constants (very unlikely).
- Output – a single variable or parameter (though it can have a complex structure). Omitted where the relationship is a constraint.
- Type – a classifier for different types of relationship, as needed.
 - Derived – the relationship is deterministic.
 - Constraint – a derived expression that must evaluate to True, so has no output.
 - Stochastic – the inputs affect the statistical distribution of the output.

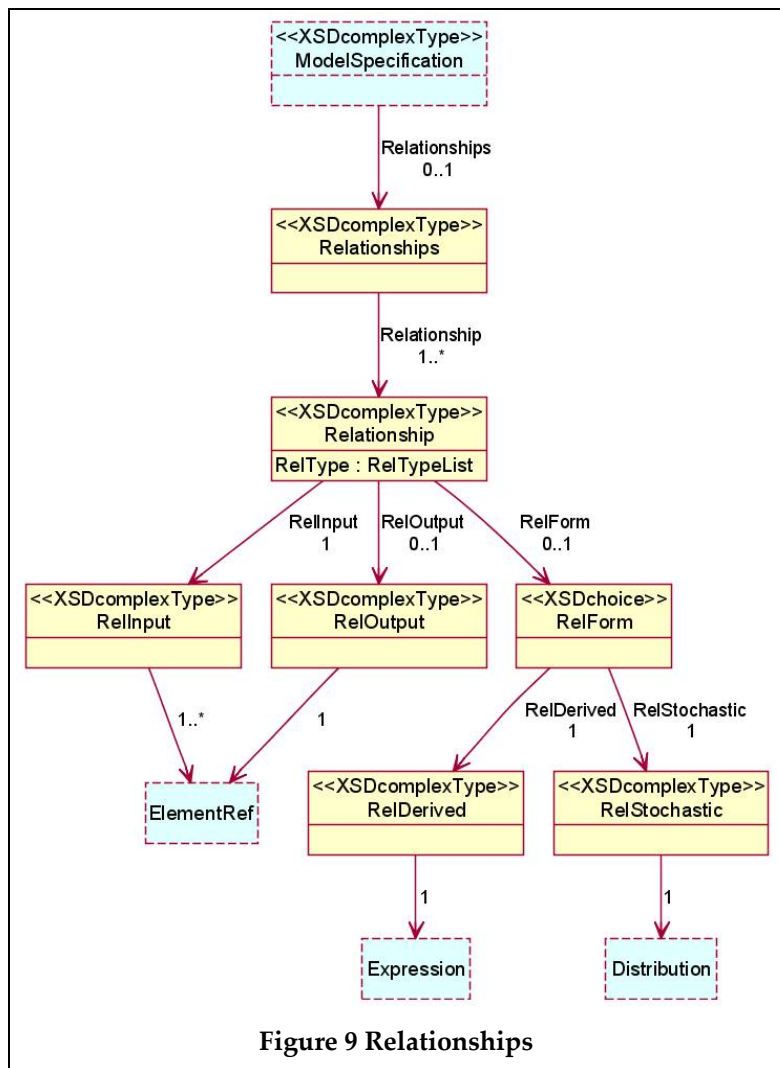


Figure 9 Relationships

- Undefined – the default if the type is not included.
- Formula – for a derived relationship this is the expression, based on the inputs, which yields the output value. For a stochastic relationship it is the type of distribution that yields the output, together with expressions, based on the inputs, which yield the parameters of the distribution.

Semantics

A variable or parameter can only be an output in one relationship, but can appear in as many inputs as needed. Each relationship implies a set of directed links from the inputs to the output. These together form a directed graph. There is no fundamental reason why there should not be cycles in this graph – that implies feedback relationships. If the graph is acyclic it corresponds to a Graphical Model.

Where the output of a relationship is a variable, the inputs can be any combination of variables and parameters. Where the output is a parameter, the inputs will generally only be parameters. We suspect that this is always true, but have not yet been able to establish the truth (or otherwise) of that conjecture.

In theory, the set of inputs in a relationship could be inferred from the formulae used. By making it separate we allow a relationship to be incompletely specified, so that the paths of influence are given, but not the detailed formulation. However, software that supported the construction of relationships could use a prior specification of inputs to check that a formula was complete, or could start with the formula and derive the inputs from it.

6.5 Distributions

Statistical distributions are used to represent both variability in variables and uncertainty in parameters, depending on context. We treat the various standard statistical distributions as primitives and assume that each is supplied with appropriate methods to calculate the information needed for model fitting. The distributions imple-

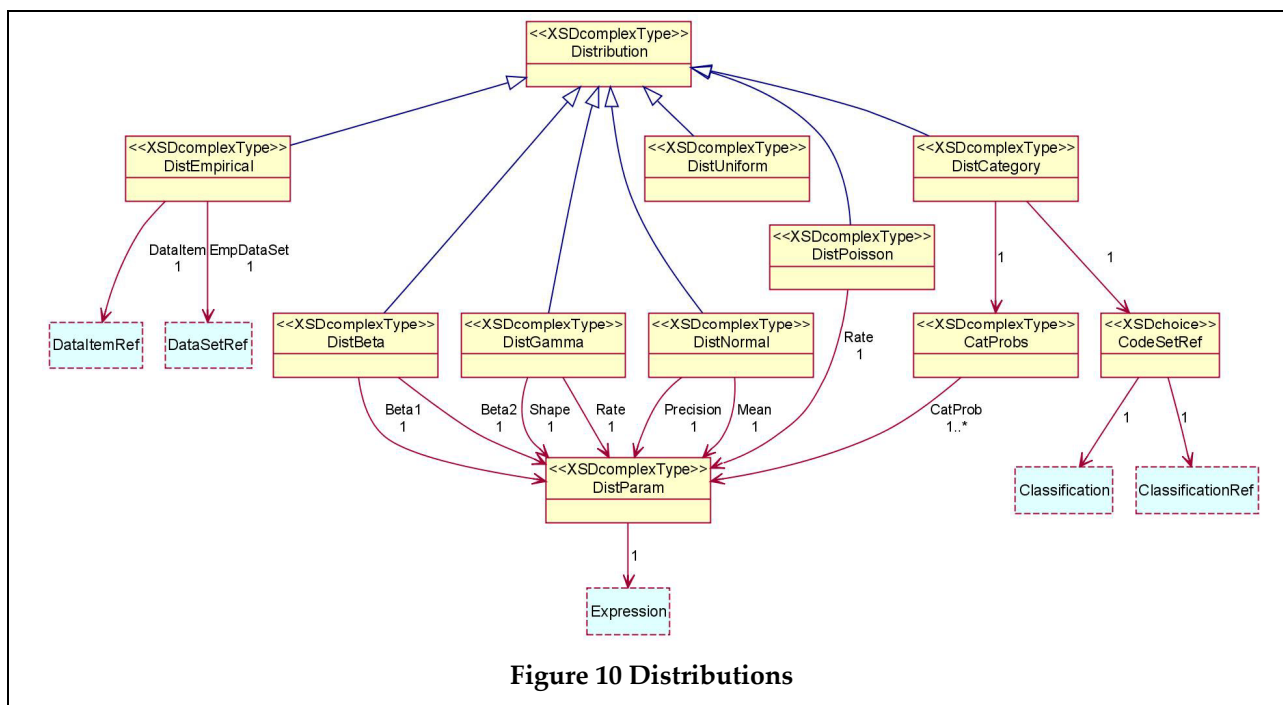


Figure 10 Distributions

mented so far are shown in Figure 10 – further standard distributions will be added as needed.

The parameters of distributions are expressions, so can be mathematical calculations based on model parameters and variables, or simpler forms.

Note the presence of empirical distributions. These are needed particularly for the representation of posterior knowledge from MCMC simulation processes. These methods produce sets of realisations of the target parameters. Such sets are like relational database tuples (data records), and so can be represented as data sets.

The distribution for a parameter linked to a column in such a realisation dataset can then be approximated by fitting a histogram (or some other smoothing method) to the distribution of values. In many contexts it will be appropriate to retain the complete empirical distribution, such as when generating synthetic data.

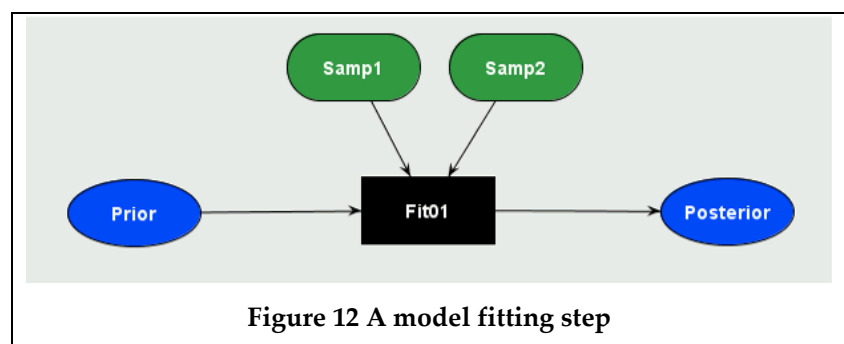
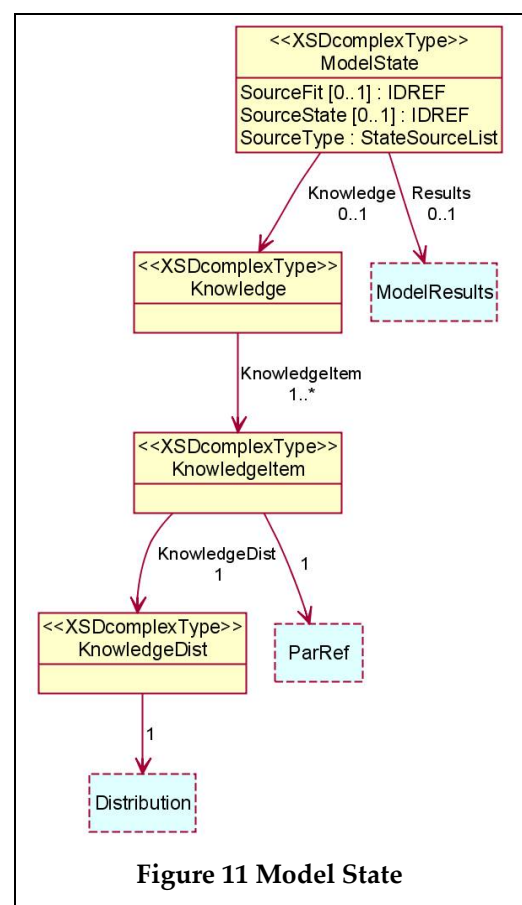
6.6 Knowledge in Model States

The specification part of a model contains a lot of knowledge about the *forms* of relationships between parameters, represented as mathematical expressions and choices for distributions. However, the specification always leaves us with some parameters that are not determined by the model relationships. We refer to these as ‘terminal’ parameters.

Knowledge about the *values* of these parameters constitutes a *State* of the model. A state associates an explicit uncertainty distribution with every terminal parameter in a model.

A model fitting step, in which evidence about parameters is extracted from data, always results in a new state of the model. Bayesian methods (which are our focus for modelling) also require a *prior* state as input. The initial state of a model is usually created manually, based either on guesswork or by importing knowledge from some other context. Figure 12 illustrates the relationship between Data, prior and posterior states and the fitting step.

With standard Bayesian methodology there will usually only be two states associated with a model, linked by a single fitting step. The Opus methodology ex-



tends this and envisages chains of states linked by sequences of fitting steps over distinct data sources.

It is equally valid to define multiple initial states and to use these as inputs to fitting steps that use the same data. Comparison of the resulting states is then the basis for investigation of the sensitivity of the final state to the starting point. Similarly, different fitting methods can be compared, using the same data and initial state.

6.7 Results from Model States

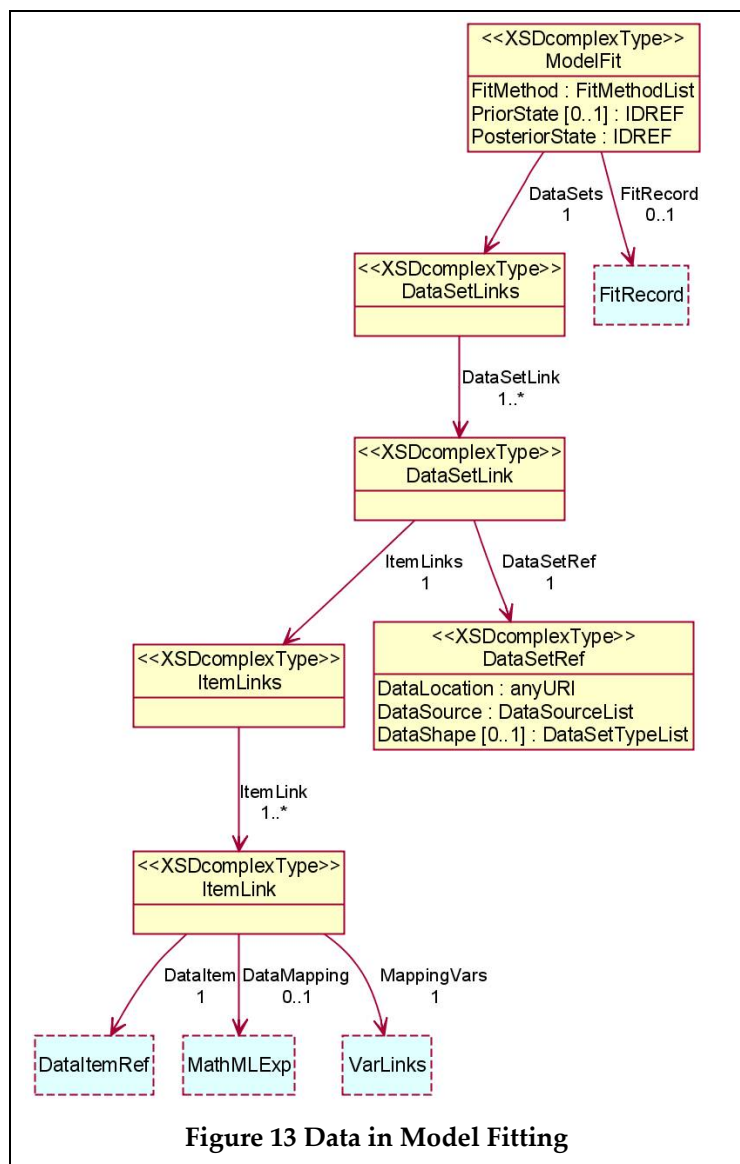
Any results that are derived from a statistical model are based on a particular model state. From the uncertainty distributions we can report best estimates of the parameters and corresponding estimates of uncertainty, in the form of uncertainty limits (equivalent to confidence limits) or some representation of the full uncertainty distribution.

For parameters that are derived from the terminal parameters, corresponding estimates and distributions can be derived. This includes the generation of synthetic data to represent both variability and uncertainty in the model, as favoured by transport engineers.

This component of the structure has not been elaborated yet, but will record any derivations, transformations or simulations used to extract results from the model state.

6.8 Data in Model Fitting

A *ModelFit* component documents a processing step in which evidence about the terminal parameters in a model is extracted from datasets. We document here which items (data variables) were used from which datasets, and how those link to the variables in the model. We assume that meta-data documenting the content of the datasets is available via the dataset link, and make no attempt to reproduce it here: we just make reference to data items by name.



The fitting method is recorded and this (potentially) provides the link for model fitting software to access or provide information for the model. Bayesian fitting will be linked to a prior model state, but other methods may not need this. All steps produce information for the posterior state resulting from the fitting step. This state represents the combination of the prior knowledge with the evidence extracted from the data.

Often the data variables will correspond exactly to those in the model, but there is no requirement that this is true. In particular, there is no particular problem if the data contains aggregate information about variables but the model is conceived in terms of individual observations or if different coding schemes are used. The only requirement is that it must be possible to calculate the likelihood of the data values from the combination of the model and the data mapping,

Note that in the Bayesian methodology it is not necessary to provide data that links to every variable in the model, or for a fitting step to produce evidence about every parameter. If no additional information is obtained about a parameter then its posterior state will be the same as its prior state. Indeed, this implies that it is possible to have components in the model for which no data exists – though this is not particularly useful!

6.9 Records of Model Fitting

Generally, information about the fitting process needs to be recorded, in addition to the posterior distributions. For example, diagnostic information and information about convergence is of value when subsequently exploring the quality of the fitting steps.

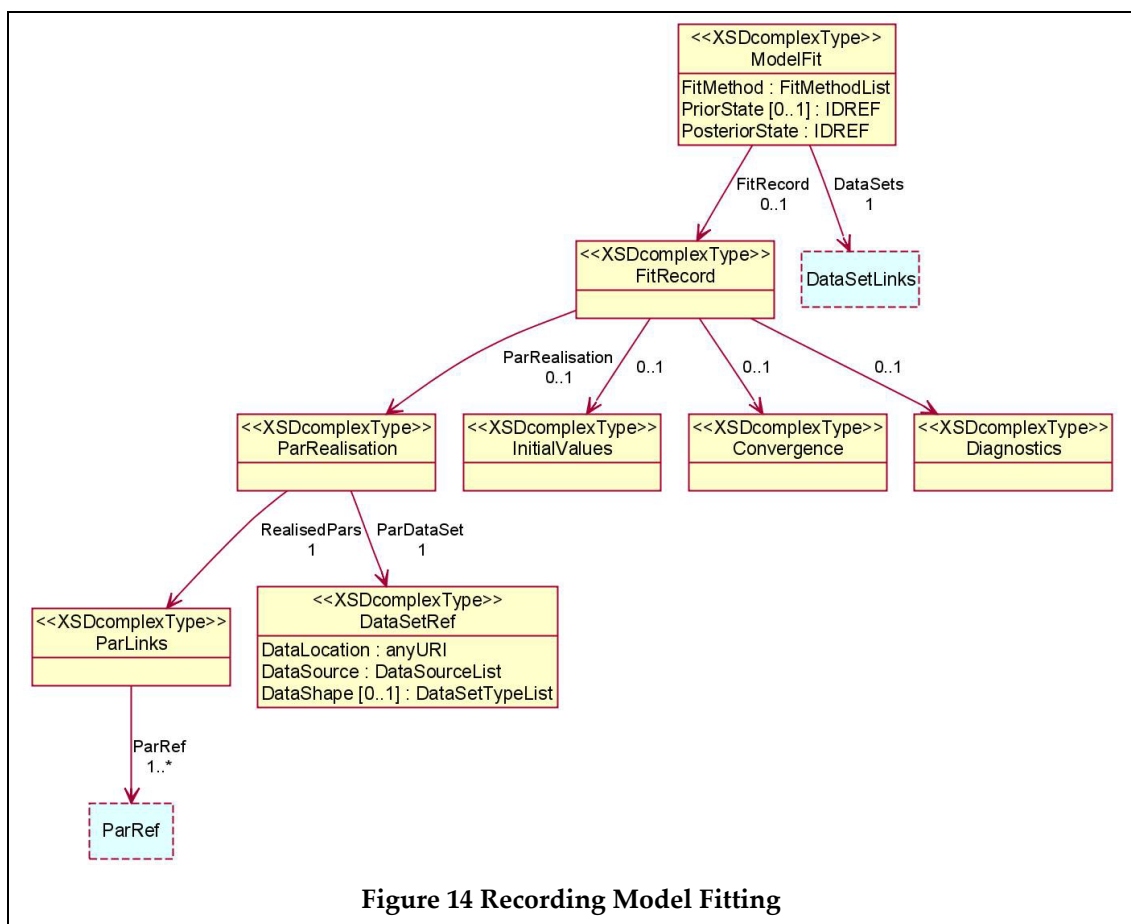


Figure 14 Recording Model Fitting

Most Bayesian methods make use of MCMC simulation steps to estimate the posterior distributions, and these simulations form the empirical distributions that we need to record in the posterior model state. The Parameter Realisation component provides a way to record this information as a property of the fitting step. As discussed previously, we can store these realisations as a dataset. Note that this has the benefit of retaining information about the joint uncertainty distributions of the parameters. The records in such a dataset are also the drivers for realisations of the (empirical) uncertainty distributions of derived parameters, and (with additional stochastic input) of the resultant distributions of variables from the model.

6.10 Other components

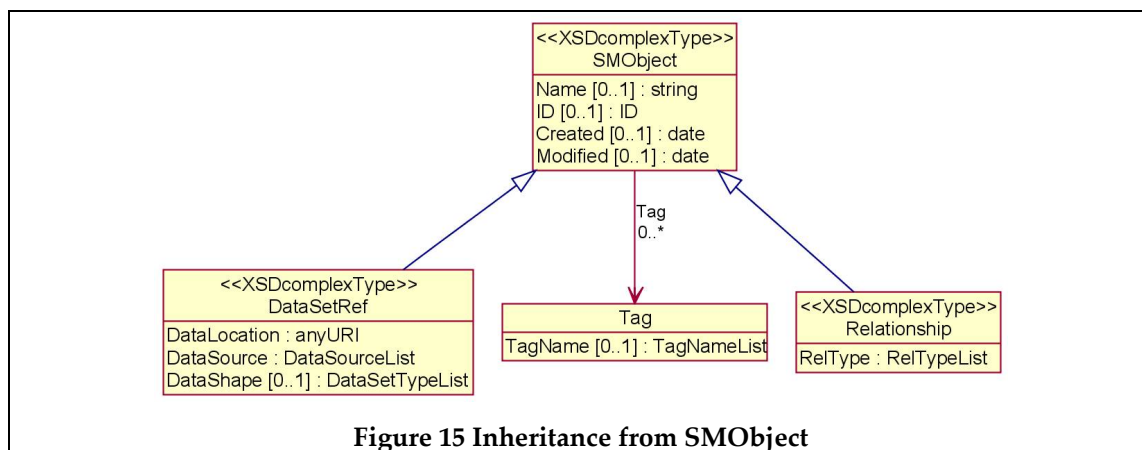
A number of other components exist within the structure. Amongst other things these relate to links to external sources of meta-data, and to information relating to preferred presentations of particular statistical models.

6.11 Other Meta-data Issues

6.11.1 Documentation of Meta-Data Objects

The individual instances of meta-data will (sometimes) have various items of associated (less formal) information, such as descriptions of purpose or intent or precedence, which can be useful to later users of the model to which the object refers. All meta-data classes need to have this facility, and perhaps other generic ones.

The obvious ways to achieve this is to define a meta-data class from which all classes in the meta-data model inherit. This is the most general approach.



This has been done, and an example is shown in Figure 15. A class called SMOBJECT has been defined, and all other classes inherit from this class, mostly directly but sometimes through other classes. The diagram shows the example of both the DataSetRef and Relationship classes inheriting from SMOBJECT. The implication of this is that all object instances in the system can have any of the attributes of an SMOBJECT. This means that every object can have a Name and an ID, plus creation and modification dates, and all can have Tags.

Each Tag has a TagName, which gives its type. Various types have been defined, such as Title, Description, Source and URL, allowing any of these types of information to be associated with any object instance. Further types can be (and have been)

added to the set. The content of a tag is not constrained. At present they are all textual, but the future use of structures is not excluded.

6.11.2 Structures for Data used by a Model

With a complex system we will have a multitude of data sources, related in complex ways. In general, these data relationships can be captured using the relational model for data structures (though there can be complications when some of the data structures are really meta-data, providing classification structures for others).

Actual data may be collected at various levels of the data structure, and this may not be the level at which the fundamental variability of the model is defined. What is essential is that the likelihood associated with any data can be calculated from the model.

It may be that all necessary relationships can be included in the model in mathematical form. This certainly works well when the number of possible groups for an aggregation is known in advance, but feels more problematic where the numbers are data-dependent or random. If this surmise is correct, then aggregation related to the network structure of a transport system (for example), which is known in advance, can be represented in the mathematics, but that related to the travelling individuals may be more problematic. However, such aggregations are of no difficulty within the relational data model.

This suggests that it may be useful to include a relational (or similar, such as E-R) data model alongside the variables component, and in addition to the mathematical specification of the relationships within the statistical model. This idea needs further refinement, and is not yet included in the UML model. What we have included is the relational concept of a *Tuple*, which is in effect a data record relating to a single data respondent. Variables can be grouped together (in a 'tuple') to show that they always occur together. Multiple tuples can be defined, relating to different types of data object, such as 'person', 'vehicle' or 'trip'.

7. WORKING WITH STAT MODEL INSTANCES

7.1 Introduction

The presentation so far about StatModel has been at the structural level, talking in a general way about the generic requirements for meta-data about statistical models, and defining the structures for storing it. We now need to consider the process by which we capture actual meta-data about actual statistical models. Then we can discuss how to make this information available to users of the results from the statistical models.

In order to store information about actual statistical models we need to be able to construct instances of the structure described in the previous sections. These instances are stored in computer-readable form as XML documents, following the XML schema (XSD) generated from the underlying UML structural model. These documents can readily be used by software, using widely available tools and conventions.

A StatModel instance is an XML document containing the meta-data about one or more statistical models. Figure 16 illustrates the various types of information that can be extracted from the design and fitting processes and stored in a StatModel document (which itself will be in an appropriate StatModel store). The purpose of the information is to provide background and understanding for the user of the results (and they may in turn feed back insights as to the substantive meaning of the results). We describe later various presentation methods and tools for exploring the provenance and reliability of statistical models stored as StatModel documents.

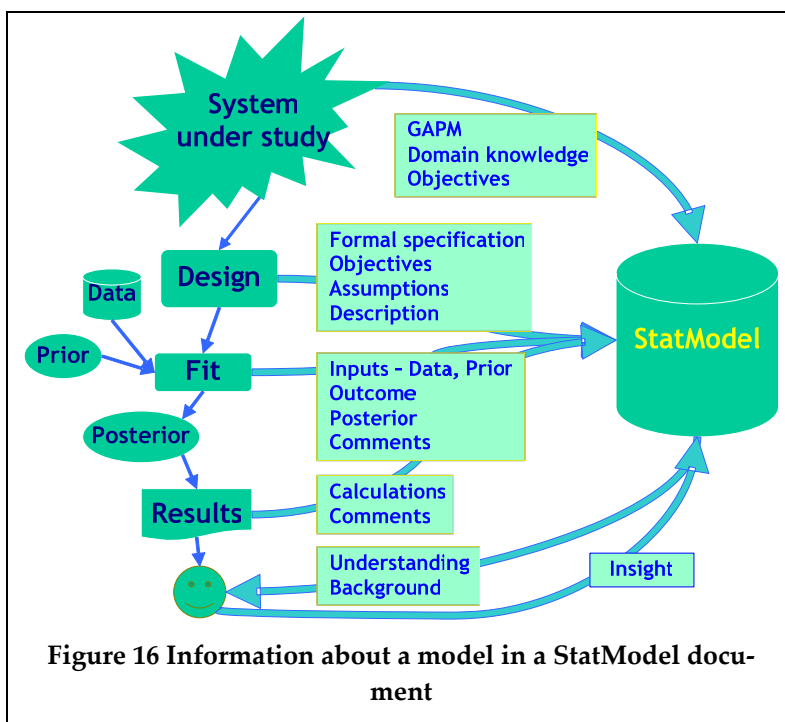


Figure 16 Information about a model in a StatModel document

the results (and they may in turn feed back insights as to the substantive meaning of the results). We describe later various presentation methods and tools for exploring the provenance and reliability of statistical models stored as StatModel documents.

Ideally, both editing and viewing systems for working with this meta-data should communicate with the user in terms of the statistical tasks involved, not generic XML. Wherever possible, the meta-data should be captured automatically, as a by-product of a design or specification process, not entered manually. This point has often been made in past discussions about statistical meta-data, see [Sund03], for example. However, it remains useful to be able to manually edit meta-data in order to add interpretation, description and explanation that may not have been included with the more formal specifications in a design context.

7.2 Working with XML documents

Standard tools exist for working with XML files, constructed using further standards developed by the W3C (World-wide-web consortium). Various programming interfaces (APIs) are available for programmers to read in and manipulate XML documents, and various producers have released suits of tools for their editing and presentation.

XML documents are text files. This makes them easy to exchange and store. They are also easy to view and edit in a text editor. However, because they are a generic way to represent complex structures of information, XML documents are actually rather difficult to understand and are generally very verbose.

The StatModel XML schema files can be used with any validating XML editor to manually create XML documents that are instances of the StatModel structure. Although our work has used commercially developed tools, alternative free tools are widely available.

Standard tools are available for converting XML documents to other forms, such as HTML for web pages or RFT for printed documents. This is achieved through the use of the XSL/T stylesheet and transformation language, supported by standards such as X-Path and X-Query. These are all XML-based standards and so can be constructed using text editors, but it is generally much more effective (and efficient) to use specialised editors for working with these tools.

We have used the Altova [Altova] suite of tools in our work for the construction and editing of StatModel documents. The XML Spy validating XML editor has been used for manual editing, and a free, basic version (the Home edition) is provided by this company.

Stylesheets for presentation have been built using the StyleVision application from Altova. A generic style sheet for presentation of the model instances within web browsers has been constructed – such generic presentation tools are valuable because they allow the content of the XML document to be presented in a way that is related to the objective of the StatModel design, without the distraction of the XML syntax. However, because they are generic they do not contain any contextual concepts to help a reader working in a particular domain. They are also static, and so do not provide any realistic opportunity for altering the presentation or exploring the represented statistical model.

We have also used StyleVision in conjunction with a more specialised editing system called Authentic (discussed below) to construct a more specialised editor for entering meta-data into StatModel documents.

7.3 Construction of model instances

7.3.1 Using a generic validating XML editor

The UML definitions of StatModel can be converted automatically into XML schema (XSD) files. These can then be used by any validating XML editor, because they contain all the rules about what is allowed where within instance documents that conform to the schema.

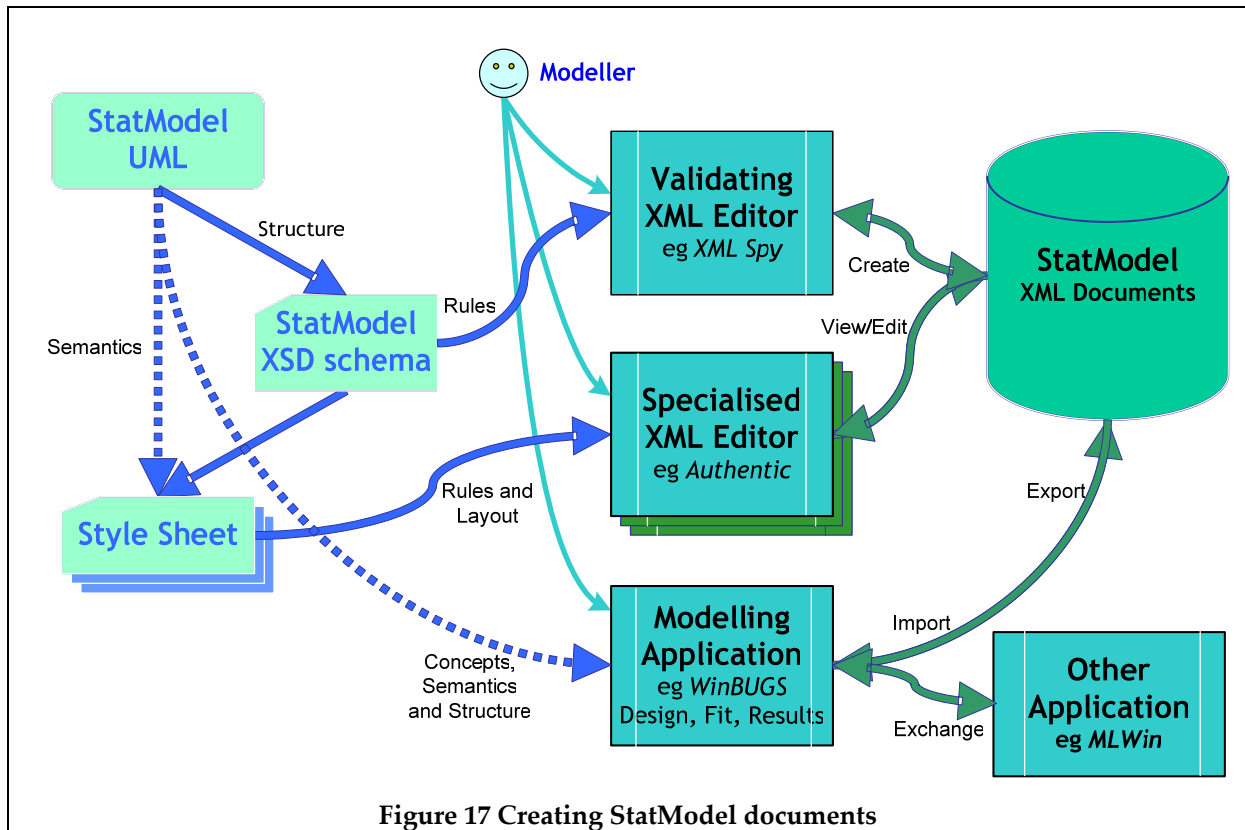


Figure 17 Creating StatModel documents

This gives us our most basic method for constructing StatModel instances, and this is how we have built (in XML Spy) the initial model representations for the example models being explored within the Opus project. Working from a template this is relatively fast for those involved with the model, but is clearly not a long-term solution. This way of working is illustrated in the upper part of Figure 17. The modeller works with the validating editor (which is controlled by the schema) to create (and edit) StatModel instance documents.

7.3.2 Using a specialised XML editor

A better approach is to use a more specialised XML editor, one which produces XML documents but can present the data entry task to the user using forms and terminology more closely related to the general objectives of the task. We have used the Altova Authentic editor for this. It is freely available, but requires layout specifications in a special style sheet which must be produced using the (not free) StyleVision editor. This process makes use of the XSD files that control the content of the XML document, but also draws on the semantics of the underlying model, as implemented by the producer of the stylesheet. This is the middle path in Figure 17.

Using these tools we have built an entry and editing application (called SM SpecEdit) for StatModel instances. This is illustrated in Figure 18, which shows the section of the Model Specification used for entering Variables and Parameters. It does not yet cover all aspects of the model (in particular the mathematics for relationships), but has the potential to become a generic editor for StatModel documents. Because the (person who built the) editor knows about the StatModel semantics, the editor contains some rules that are not explicit in the schema files. For example, when defining a relationship, the editor limits the entry of variable and parameter references to those already defined within the current model. Similarly, classifications used for ma-

trix dimensions and the coding of categorical variables are limited to the currently available set. Input is facilitated through the use of 'combo boxes' based on the allowable entries.

Such editors do not need to cover all components that can be present in the document. Thus it might be useful to have separate editing tools that covered different sections, such as one for the model specification and a separate one for the fitting and state components. Another approach might be to have a simple one that was only sufficient for specifying GAPMs (at a rather abstract level), and another for complete models. Again, we could have a specialised version that could display a complete model but only allowed comments to be added or edited.

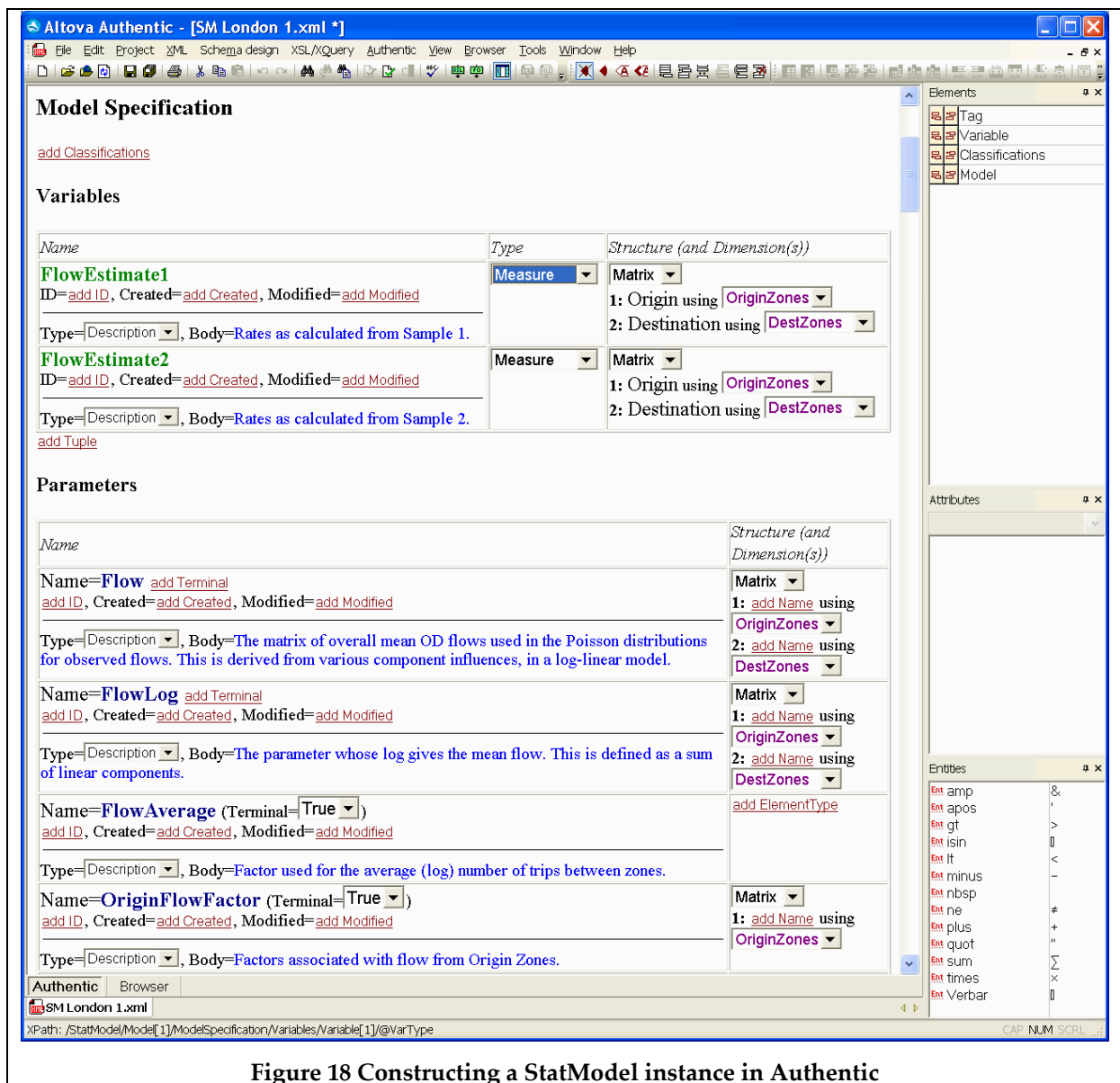


Figure 18 Constructing a StatModel instance in Authentic

7.3.3 Creating Mathematical specifications

Mathematical specifications (the expressions that are allowed in the Expression contexts of StatModel) are written according to the MathML standard. We have used the free Formulator editor [FormML] for this. It allows mathematical expressions to be constructed in a way similar to the Equation Editor in MS Word. The mathematics can be displayed either in mathematical form, or as MathML text, and the latter can

be copied into the appropriate places in the StatModel XML document. The Formula editor can also be integrated more closely with the full (not free) version of XML Spy, which automates the cut and paste steps. Unfortunately, it cannot be integrated with specialised editors based on Authentic, so the specification of the mathematics will remain a specialised step.

An alternative MathML editor is the MathType application from Design Sciences. Unfortunately the version needed to get access to the MathML script for use in StatModel is not free. However, this product has the advantage that it can be integrated into MS Word as a replacement for the standard Equation Editor.

The same company produces the (free) Math Player [MP] product, which is required for viewing the MathML as mathematics within a web browser.

7.3.4 Working with modelling applications

The process of statistical modelling requires the specification of modelling processes in specialised applications, such as WinBUGS [BUGS]. It clearly makes sense to explore whether it is possible to transfer the information needed for StatModel documents directly, rather than going through a manual transfer process.

Note that it is unlikely that such an automatic process would ever produce complete StatModel instances, because the objectives of the StatModel approach are different from those of the modelling applications. So there will continue to be a need for manual editing, particularly for the inclusion of motivational and presentational information.

Most of the statistical models we are exploring are being designed initially in the WinBUGS application, and we are exploring a simple converter from WinBUGS script that would produce the main parts of a StatModel instance automatically. This would not be expected to work all the time, because very complex models can be expressed in WinBUGS. We do not propose to write a complete parser for the WinBUGS language, but will limit ourselves to simpler cases.

Such a converter would be a very basic example of a mechanism to export meta-data from the application in which the statistical modelling is undertaken, into StatModel form. To do this more successfully, it is better for the application developers to take ownership of the problem and to build a specific export mechanism into their products. This requires a good understanding of the concepts and semantics of the StatModel approach, in addition to the structural rules, and is not a trivial task. Some of the underlying concepts in StatModel (such as variable and parameter) are defined in a way which is different from (though not incompatible with) the related concepts in WinBUGS. A further difficulty is the way that references to data are integral to the WinBUGS model specifications, but are treated only in the fitting steps in StatModel. And WinBUGS focuses on single fitting steps, whereas in Opus we see model fitting as a process with multiple steps.

Once model export is established, it becomes reasonable to consider the import of models by modelling applications from a StatModel instance. This is more complex, because now the StatModel instance must contain (almost) all the information required for complete specifications in the application. We accept that extensions to the

existing structure (particularly in the area of the specification of relationships) are likely to be needed, but think that the structure is strong enough to support this.

Once import and export are available, StatModel becomes a possible medium for the exchange of models between modelling packages. This idea could be extended further, with the StatModel structure being extended to become the native format for applications to use to store models. Some interest in this possibility of model exchange between MCMC packages has been expressed (independently) in the past by the WinBUGS team, but no real progress has been made.

In the event that the StatModel proposal is found to be an acceptable proposal for storage and/or exchange of models, we would expect the developers of appropriate packages to provide an interface to StatModel, both saving models in this form and allowing models from elsewhere to be imported. A system focussing on model design would use and create the Model Specification components of the instances. Model fitting software would use this and also use and create the State and Fit components.

The Opus project has no resources to explore this path. In practice, it is unlikely that any existing systems will be changed or extended in this way in the short term, but we have initiated discussions with the developers of WinBUGS and MLWin [CMM] to at least explore whether our proposals could be compatible with their systems. And we accept that any such use would require extensions to the existing StatModel structure.

8. PRESENTING STAT MODEL EXAMPLES

8.1 A Model Example

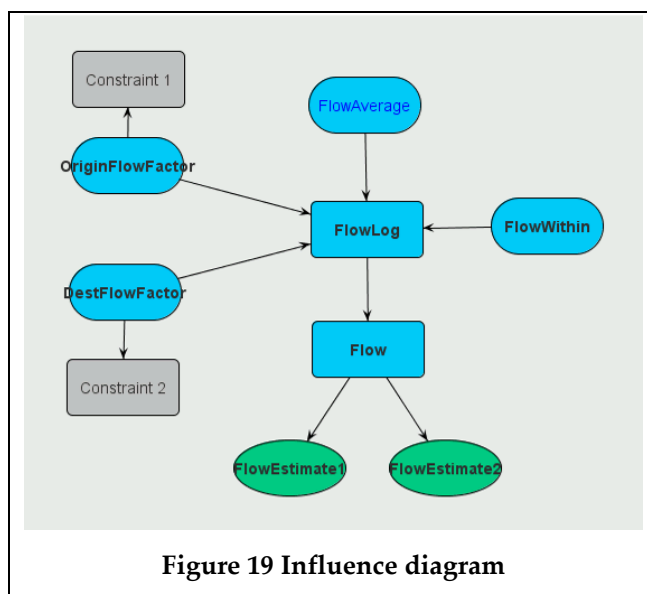
Appendix 3 contains a listing of a StatModel document for a basic part of the London test case. This produces a combined estimate of the origin and destination effects in a log-linear Origin-Destination (OD) model from two independent estimates of the OD flows. The appendix also shows the listing of this model in a web browser. The mechanism used for listing models is discussed in the next section

In addition to simple listings we also need more dynamic functionality for model exploration. A web site has been developed as a test bed for a number of components which allow a StatModel document to be explored in more dynamic ways. The functionality of this is discussed in following sections, and the following two chapters present the technical issues involved in the building of the site.

Figure 19 shows the Influence Diagram for this model. The shape and colour conventions used in

this diagram are listed in Appendix 1. This figure shows how various parameters (in blue) are related together and how they impact on variables (in green). Rectangles show derivations, and ovals are stochastic. In this model the parameters form a log-linear model for the Flow (the OD matrix), and the two Flow Estimate variables are (independent) observations on a Poisson distribution with this mean. The details of the relationships can be read from the listing in the appendix.

Fitting processes are also stored in the StatModel document and can be visualised in the web application. Figure 12 (page 29) illustrates how the two sample datasets are used in a fitting step with the prior knowledge to create the posterior knowledge state. This latter state contains the knowledge from which results are derived.



8.2 Generalised listing with style sheets

An XML style sheet has been developed to display all the components of a StatModel instance document, using a web browser. Figure 20 shows the listing of all the relationships that specify the model shown in Figure 19.

The fragment in the figure is the start of the section that lists the relationships in the model, and it shows the input and output elements used in each relationship, together with the mathematics of the relationships. It also shows how comments that describe the purpose of the relationship can be associated with it. Colour coding is used in the listing to highlight variables and parameters and to identify comments.

Name	Output	Input	Form
Estimate 1 distribution	Stochastic FlowEstimate1	Flow	Distribution: Poisson: Rate=Flow Poisson distribution for observations in first estimate set, based on common rates.
Estimate 2 distribution	Stochastic FlowEstimate2	Flow	Distribution: Poisson: Rate=Flow Poisson distribution for observations in second estimate set, based on common rates.
Derived Flow	Flow	FlowLog	Derivation: $\exp(\text{FlowLog})$ Poisson rates are derived as exponential of (linear) flow function.
Derived FlowLog	FlowLog	FlowAverage, OriginFlowFactor, DestFlowFactor, FlowWithin	Derivation: For $i \in \text{OriginZones}, j \in \text{DestZones}$ If $(i \neq j)$ then $\{\text{FlowAverage} + \text{OriginFlowFactor}[i] + \text{DestFlowFactor}[j]\}$ If $(i = j)$ then $\{\text{FlowWithin}[i]\}$ Linear function for log of flow rates. Inter-zone flow is modelled as an average flow adjusted by origin and destination factors (with no interaction). Intra-zone flows are modelled separately
Constraint	OriginFlowFactor	OriginFlowFactor	Derivation: $\sum \text{OriginZoneFactor} = 0$ Origin factors sum to zero, so product of rate components is one.
Constraint	DestFlowFactor	DestFlowFactor	Derivation: $\sum \text{DestZoneFactor} = 0$ Destination factors sum to zero, so product of rate components is one.

Figure 20 Relationship Listing from the Example Model

Everything in the listing (apart from the italicised headings) is taken directly from the instance document.

Name	Output	Input	Form
Derived ModeIndic	ModeIndic	NMWMTime	Derivation: $\text{Step}(\text{NMWMTime} - 1.1) + 1$ For the 5 mode use variables, recode (1 to 1), (2, 3 & 4 to 2). This gives a vector of indicators showing whether each mode was used.
Derived BaseTimeProb	BaseTimeProb	BaseTimeRate	Derivation: $\frac{\text{BaseTimeRate}[j,k]}{\sum_n \text{BaseTimeRate}[j,n]}$ Convert Base Rates to Base Proportions for time distributions. This gives (for each mode) the travel time distribution for respondents who only use one mode and are in the base groups for all respondent attributes (usually group 1).
Derived RespTimeRate	RespTimeRate	BaseTimeProb, Alpha, Beta, Gamma, Delta, Kappa, Phi, Xi, Omega, ModeIndic, Age, HomePop, Sex, TravelDay, WorkMode, WorkMode2, BreathProb, HeartProb	Derivation: For $j \in \text{Mode}, k \in \text{NMWMTime}$ $\left\{ \begin{array}{l} \text{BaseTimeProb}[j,k] \\ \times \text{Beta}[j, 1, \text{ModeIndic}[1]] \times \text{Beta}[j, 2, \text{ModeIndic}[2]] \times \text{Beta}[j, 3, \text{ModeIndic}[3]] \\ \times \text{Beta}[j, 4, \text{ModeIndic}[4]] \times \text{Beta}[j, 5, \text{ModeIndic}[5]] \\ \times \text{Alpha}[j, \text{HomePop}] \times \text{Delta}[j, \text{TravelDay}] \times \text{Phi}[j, \text{Sex}] \\ \times \text{Omega}[j, \text{WorkMode}] \times \text{Omega}[j, \text{WorkMode2}] \\ \times \text{Kappa}[j, \text{BreathProb}] \times \text{Xi}[j, \text{HeartProb}] \times \text{Gamma}[j, \text{Age}] \end{array} \right\}$ else $\{\text{BaseTimeProb}[j,k]\}$ Adjust Base Proportions for time distributions to give Respondent Rates: The probability of NOT using a mode depends on the use of other modes and the other attributes of the respondent. Where a mode IS used the time distribution does not depend on the other factors. Each Beta is a factor by which the probability of NOT using a mode is altered if another mode IS used. The other parameters are all adjustment factors when a respondent attribute is not the base group.
Derived RespTimeProb	RespTimeProb	RespTimeRate	Derivation: $\frac{\text{RespTimeRate}[j,k]}{\sum_n \text{RespTimeRate}[j,n]}$ Convert Rates to Proportions for Respondent time distributions.
Stochastic NMWMTime	NMWMTime	RespTimeProb	Distribution: Categorical: Probabilities: RespTimeProb The second index of RespTimeProb gives the probabilities of the four categories of travel time (of which the first is non-use).
Derived Beta	Beta	LBeta	Derivation: $\exp(\text{LBeta})$ The probability calculations use multiplicative factors, but the Bayesian fitting is done using the logged versions of these parameters. This has the effect of constraining the factors to be positive, but also tends to be more effective computationally.
Derived Alpha	Alpha	LAlpha	Derivation: $\exp(\text{LAlpha})$

Figure 21 Relationship Listing (partial) from the WP11 model

In contrast, Figure 21 shows a fragment of the listing for a single, much more complex model that has been used as a test case in WP11. This model explores the distribution of the time spent travelling (within a day) on different modes of transport, and the extent to which this is affected by the characteristics of the traveller (including where they live). It includes interactions between different mode choices.

The information presented by this listing is a complete presentation of the contents of the XML document, and is often adequate for exploration by the originator of the model and other specialists. It can be more helpful than the original script used in WinBUGS (the statistical application used for this example) for less experienced users, but in general a user will need more explanatory information and context.

For specific applications the style sheet can be used (by someone familiar with XSL/T) as the basis for a more focussed listing. However, this approach has limited applicability.

8.3 Graphical display of model relationships

Various components have been developed to provide dynamic displays of parts of the StatModel instances within a web browser. Figure 22 shows the testing interface which is

made available to partners and associates of the Opus project. This includes the full influence diagram for the model of which Figure 21 lists a part.

The display is a Java applet, and the graph is created dynamically from the information in the XML document. Controls are provided for a number of different automatic

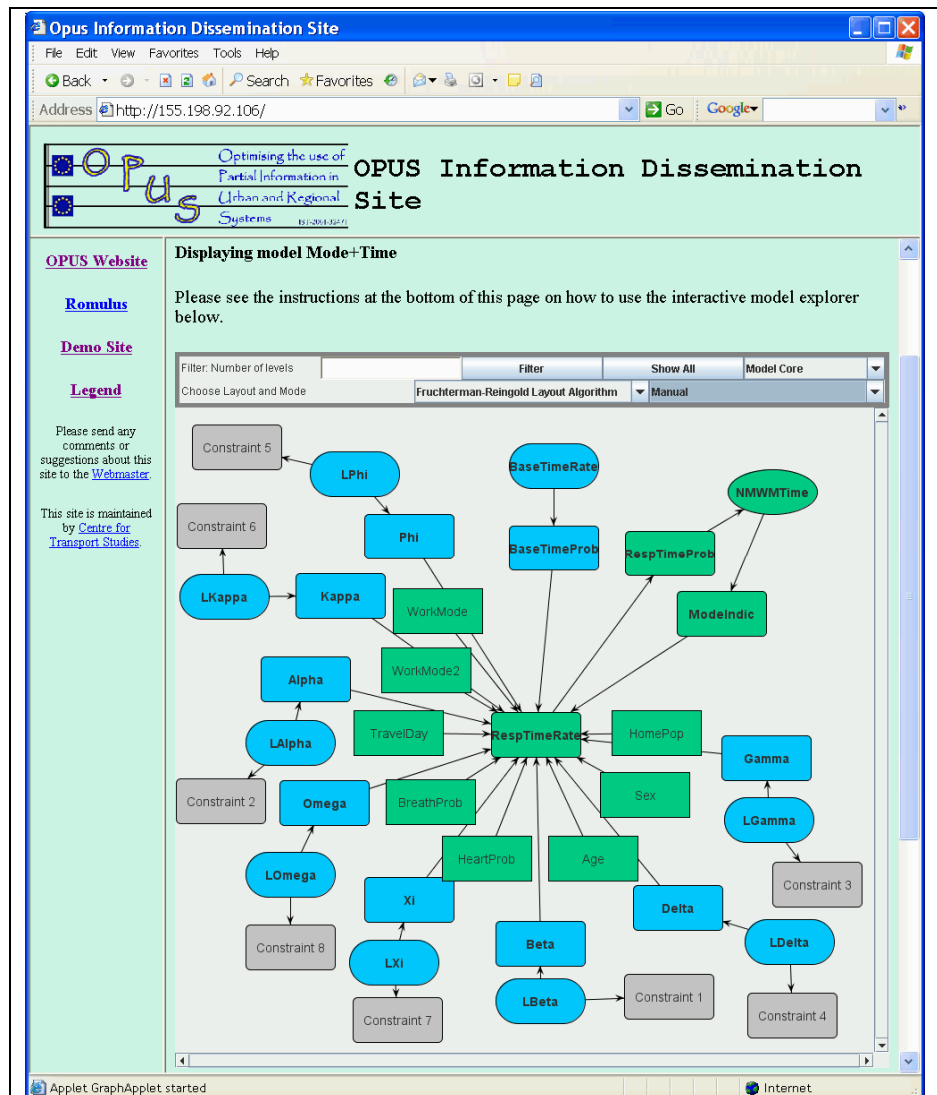


Figure 22 Full influence graph in browser

graph layout algorithms, the user can zoom and pan the display, and can select to manually adjust the location of individual nodes.

Several graph trimming options are provided. The filter option trims the graph to show only nodes connected to a selected node, with the filter level giving the number of links that may be traversed. A pop-up menu accessed with a right-click provides options to hide or expand child nodes or to hide a selected node. And node sets can be predefined for display where the model designer can identify subsets that aid understanding of the model.

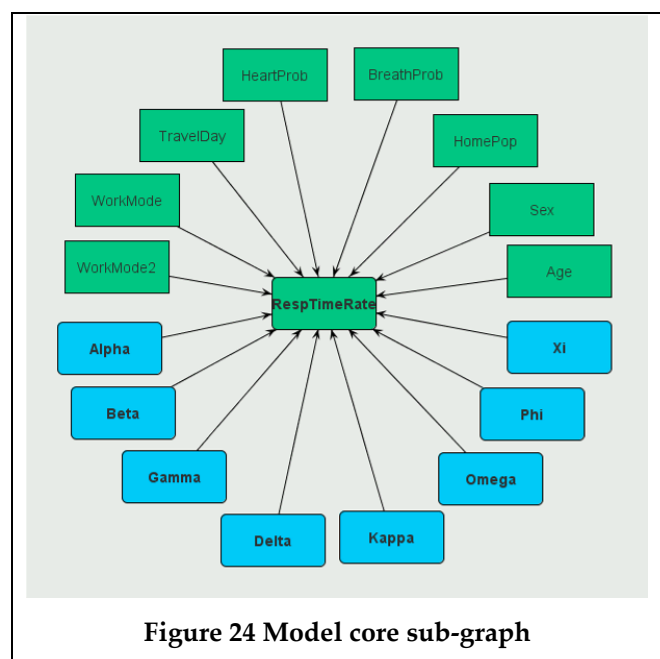
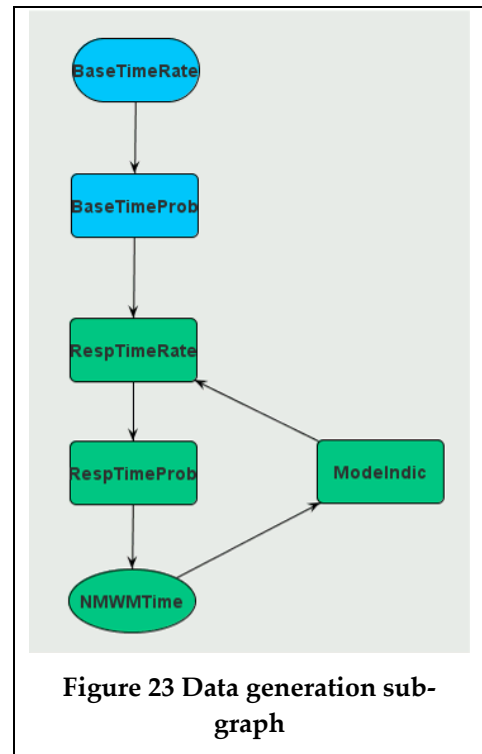
Colour coding is used to distinguish between variables and parameters in the display, and the node shapes distinguish between derived, stochastic and other relationships.

Figure 22 shows all the relationships in this model. The display is complex, but some regularity can be discerned, suggesting that the complexity comes from the number of elements, rather than the basic complexity of the model.

Figure 23 extracts just the part that relates to the response variables in the dataset, the variable NMWMTIME, which is a vector showing time spent on each of 5 transport modes, grouped into four classes.

BaseTimeRate is an array giving the basic usage rates for each mode. This is a terminal parameter, indicated by the use of a lozenge as the node shape. These figures are standardised to probabilities in BaseTimeProb and then adjusted in RespTimeRate to account for the characteristics of a data subject, including their mode usage. These rates are again standardised in RespTimeProb to give probabilities which are used in the stochastic relationship (indicated by an oval) to define NMWMTIME. Actual usage figures for respondents are then processed in ModeIndic to create a vector of indicators for whether each mode was used at all, and these feed back into the calculation of RespTimeRate.

Figure 24 shows how the other parts of the model also contribute to the calculation of RespTimeRate. All the other respondent variables, and a set of corresponding parameters, enter at this point. The mathematics of these relationships can be read from Figure 21.



8.4 Comparison of models

The influence graphs can make it straight forward to see the differences between related models. For example, Figure 19 (page 40) shows the graph for a log-linear model in which two independent estimates are used to calibrate a single model.

This example arises when two quite different methods (such as a household survey and roadside measurement) are used to estimate the flow between zones in a transport system, and a combined estimate of the parameters is desired.

Figure 19 treats the two estimates as independent samples from exactly the same stochastic process.

In contrast, Figure 25 shows the model in which it is recognised that the two different data collection methods may influence the estimates.

Here the origin, destination and within-zone factors are still shared, but the average level of flow obtained from the two datasets may have been influenced by different biases, so are estimated separately.

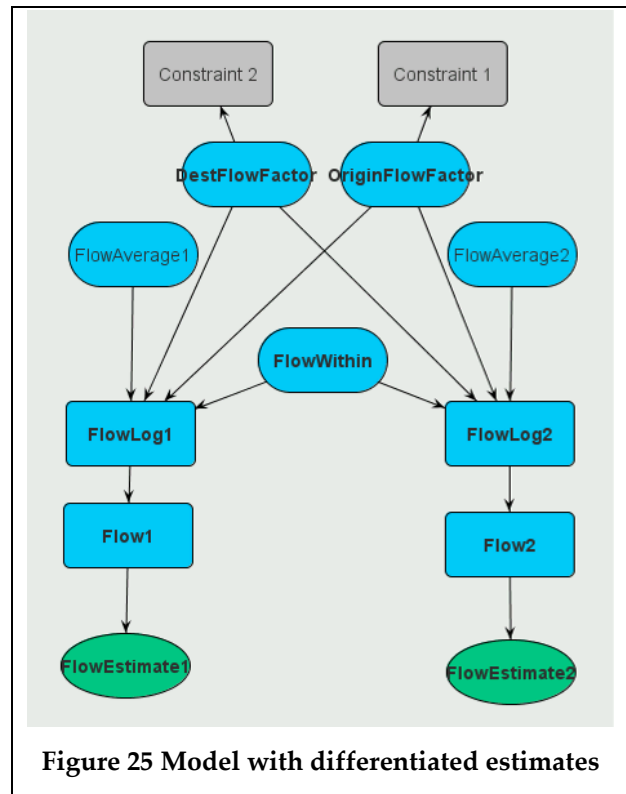


Figure 25 Model with differentiated estimates

8.5 Model fitting process

Model fitting steps use evidence from data to produce a new state of knowledge about the model, usually based on a prior state. Fits and States thus produce chains, decorated with datasets. A simple example of this was shown in Figure 12.

Figure 26 shows a more complex example. Here the same prior state is used in two fitting processes. The first has only one fit step and uses two datasets together. The second process has two steps and uses one dataset at each step. The datasets are intended to be the same, so comparison of the two different final states (posterior and joint) would indicate whether or not the two different fitting processes were equivalent.

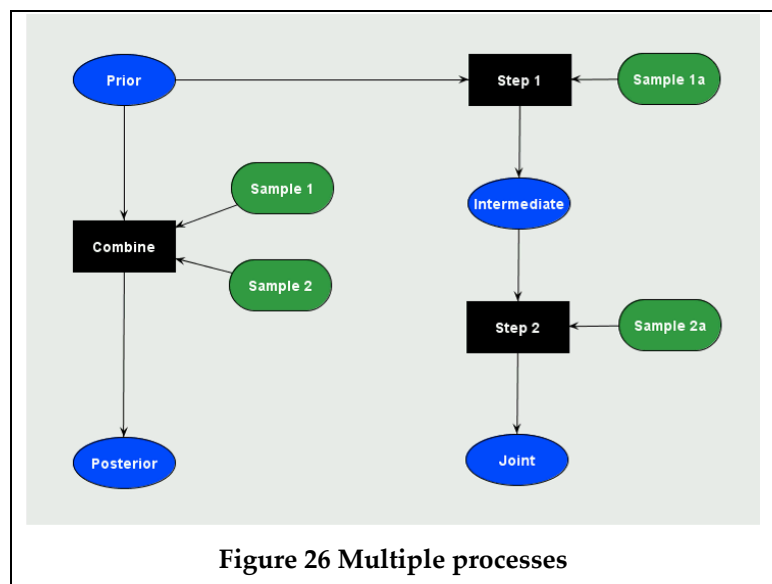


Figure 26 Multiple processes

The display of model fitting and state chains uses the same display applet as the influence diagrams, so has the same functionality and flexibility.

8.6 Summary

Various initial ideas about display and presentation methods were set out in deliverable D6.1, particularly in chapter 5. The work that we have done covers most of the objectives of section 5.2, about model and process representation.

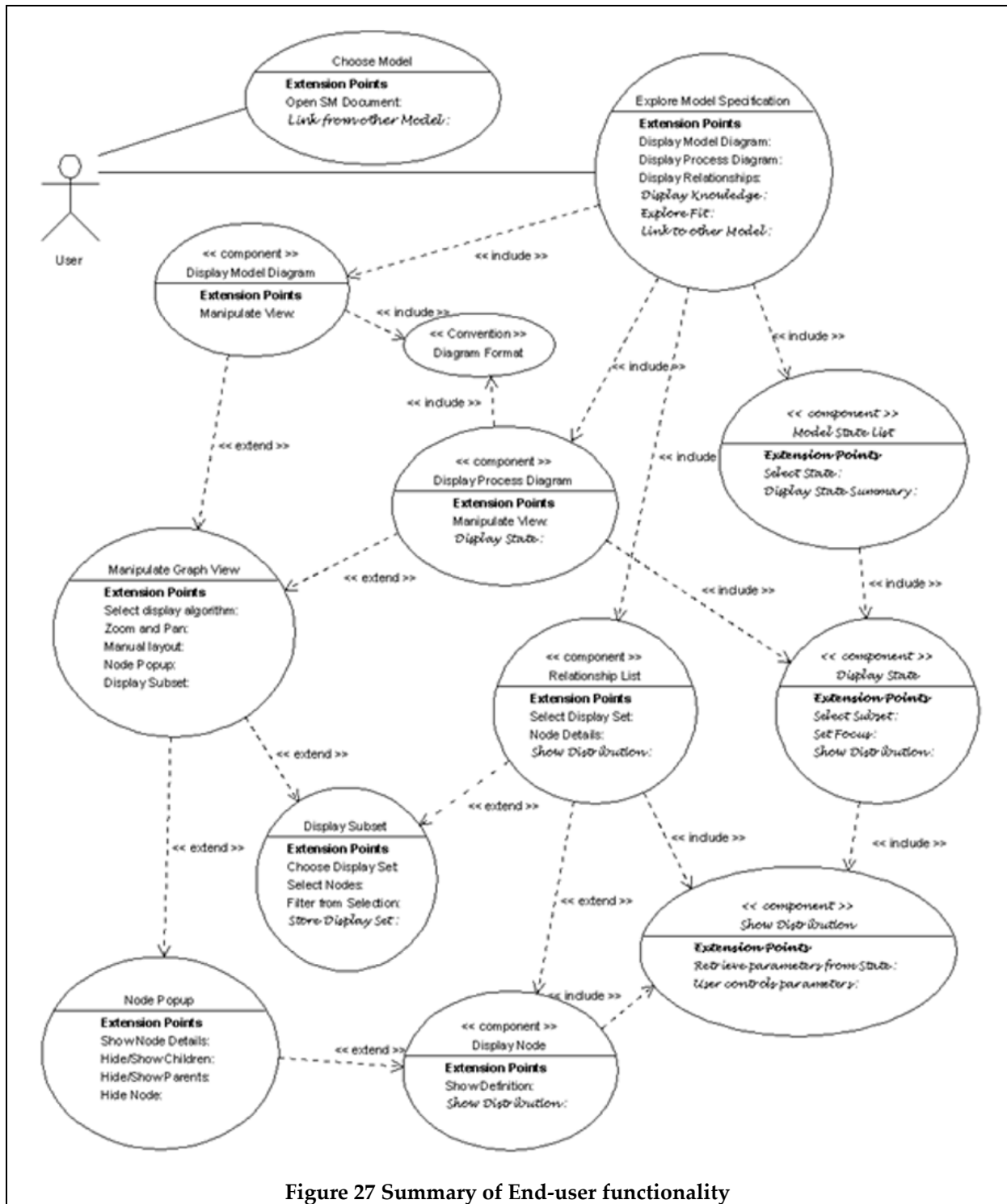


Figure 27 Summary of End-user functionality

Figure 27 summarises (in a form related to a UML Use Case diagram) our plans for the functionality of the end-user environment for testing the StatModel structure and

functionality. The elements in script font have not yet been implemented. Because of various delays within the project we have not been able to make much progress on the presentation of information about model reliability, discussed in section 5.3 of D6.1. Test data for work in this area is only now becoming available, as we finalise this report. However, initial experiments using R for the display and comparison of posterior empirical distributions have been successful.

9. IMPLEMENTATION OPTIONS

9.1 Requirements

In this section we discuss the technical options for the implementation of the presentation functionality discussed previously. The following chapter gives an overview of the actual implementation.

The system has specific functionality requirements as well as soft requirements, as outlined below. The requirements have driven the consideration of implementation options and the chosen technology components for implementation.

9.1.1 Functionality Requirements

- Dynamic graphical display of mathematical models as graphs: A system for representing statistical models in XML documents is developed as a part of WP6 and has been reported above. The web-based system must have a component that is able to graphically display the model defined in the XML document.
- Display of mathematics: Mathematical expressions are needed at various stages in the specification of a statistical model. The mathematics must be stored in a standard format that is amenable to manipulation and evaluation by appropriate software, and that is displayable on the web.
- Graphical display of statistical properties of data: The system should be able to display statistical distributions. This is needed for exploration of the statistical variability and uncertainty properties of variables, parameters and derived results, including synthetic data.
- Integration with other data sources such as Romulus: It is envisaged that the input data for the Opus process will often come from external systems, especially the Nesstar system used by Transport for London (TfL). The system should have a standard component for accessing data and meta-data from the Nesstar system.

9.1.2 Soft requirements

- Web based system: A web-based system based on open standards and components will enable users and researchers at different geographical locations to access the system.
- Usage of open components: The system will use open-source components as much as possible. This will enable a “no-strings attached” system that can be further evolved for specific requirements for different instances of the application of the Opus methodology.

9.2 Technology options

9.2.1 Graphical Display of Model Graphs

The StatModel structure includes representation of the relationships between different variables in the domain of application (mostly transport), and this is best represented as a directed graph. Hence, the technical approach to display of the StatModel graph is to translate the XML to a standard XML-based graph representation format, and use display components already available for that format to drive the web-based display of a model.

Hence, a survey of commonly used graph representation XML formats was carried out to identify the format to be used in the system. Maturity of the XML format as well as the number of open-source technical components that are compatible with the format were considered while short-listing.

9.2.1.1 XML Graph representations

GraphML

GraphML [GraphML] is the successor of the earlier non-XML graph specification GML. GML started in 1995 as a part of Symposium on Graph Drawing and ended in the Symposium on Graph Drawing 1996. The format is used by many graphing software applications. GraphML was initiated by the Graph Drawing Steering Committee prior to Symposium on Graph Drawing 2000. It is a mature standard that is widely used.

An open-source project, JUNG (Java Universal Network/Graph Framework) [Jung], supports input of graphs expressed in GraphML format. It is a Java software library that enables modelling, analysis and visualisation of data that can be represented as a graph or a network. However, there is a bug with the current release of JUNG that affects the visualisation functionality. Hence, GraphML-JUNG combination was not used for the display of StatModel XML.

GraphXML

GraphXML [GraphXML] was presented at Symposium on Graph Drawing 2000, and has been in use since 1999, developed by CWI (Centre for Mathematics and Computer Science in Amsterdam). A Java based open-source framework, GVF (Graph Visualisation Framework), is available to manipulate graphs described in GraphXML format. GraphXML is no longer active, and this format was discarded for use in WP6.

GXL

GXL [GXL] is designed to be a standard exchange format for graphs, and it is one of the most mature XML representations of graphs available. It is supported by JGraph [JGraph], another open source Java based graph visualisation and layout package. It is one of the most widely used graphing packages in the industry, used by 12 of the top 30 Fortune 500 companies. Together, the GXL-JGraph combination forms one of the most mature open source technologies available to represent and visualise graphs.

XGMML

XGMML [XGMML] is an XML based file format for graphs based on GML. A Java based open source library, OpenJGraph [OJG], is available to manipulate graphs in XGMML format. It appears that OpenJGraph project has been dormant since 2002, and there are no updates on the XGMML site since 2001. There is little evidence of widespread adoption of XGMML based on Google web search. Hence, this standard was not used for the purposes of WP6.

9.2.1.2 *Dynamic display of StatModel XML*

It was determined that JUNG provides the most suitable technology for displaying graphs on the web; it is especially easy to create interactive Java applets using JUNG classes. JUNG also comes with a set of different layout algorithms that can be used for graph display. Hence, it was determined that a JUNG driven Java applet will be used to display graphical models.

It was determined that there is no value in converting StatModel XML to a standard graph format, as a mature interface for graph creation could not be found in JUNG for any of the standard XML formats. On the other hand, we could parse the StatModel XML and create the data necessary for graph creation directly using JUNG's graph creation API. Hence, this method was used in creating graphs on the applet. StatModel XML is parsed via XSLT using Xalan API, which is another open source component available from Apache. The relevant information is passed as parameters to the applet to generate and display the graph.

9.2.2 Display of mathematics

There are two commonly used methods to display complex mathematics on the web.

The first method involves converting the display of mathematical formulae into graphics, and including the graphics image on the web page. This method has been in use for a long time, but is cumbersome. Some of the most reputable mathematics sites on the web, such as Wolfram Mathematics Resource page, use this method.

A newer method involves MathML [MathML]. MathML is an XML standard for representing mathematics, and is accompanied by various applications for editing mathematical expressions and displaying them on the web. With the help of the MathPlayer plug-in [MP] for rendering, mathematical formulae can be expressed using XML on a web page, nested within HTML.

The latter method was chosen for the purpose of WP6. This method is dynamic, and allows integration of mathematics from different sources using XML and displaying on the web.

A number of packages are available that support the construction of mathematical expressions in MathML (similar in functionality to the equation editor in MS Word). An additional advantage of MathML is that it is designed to support evaluation as well as display, so can (at least potentially) be used as input by fitting programs.

9.2.3 Graphical display of statistical properties of synthetic data

R [R] is an environment and language for statistical computing, and is heavily used by other work packages in Opus. In addition, R has a rich functionality for displaying statistical properties of data. Hence, R was a natural choice for us to use for displays of distributions.

There are multiple known ways to integrate R functionality with a web-based application. Rcgi [Rcgi] enables web pages to use R using CGI scripts running on a web server. COM calls can be made to R using the Microsoft technology; this will enable integration of R with a web site running in Microsoft IIS server. Another available option is to run R in the server mode using Rserve [Rserve], and use Java technology to communicate with R.

The Rcgi option is a simplistic one, and handling concurrent users is not straightforward, while the COM option requires the use of proprietary Microsoft technology. Hence, the last option is implemented in this project.

Rserve: R was run on the server mode using the Rserve mechanism. This enables R to listen to a specified TCP/IP port for requests. Rserve is a part of the standard R distribution.

Rserve-Java Bridge: A Java API, JRClient, is available from ROSUDA that enables communication with Rserve. This API enables integration of Java applications with R using Rserve.

9.2.4 Technology infrastructure for the website

The components described above have to be hosted in a technology architecture that enables building of integrated websites using these components. Following the guideline of using standard open-source components and software, Apache web server was the web server of choice for hosting the website. It is a mature technology that is widely in use.

In order to enable more complex functionality, Apache Tomcat servlet [Tomcat] container was used to provide a Java-based execution environment. Tomcat is developed as a part of the Apache initiative, and Apache-Tomcat architecture is a widely used configuration.

MySQL database server, the leading open source database software, enables data storage. All software is hosted in a Linux environment. This provides a completely cost-free, open, scalable architecture for the purpose of WP6.

10. WP6 PROOF-OF-CONCEPT SITE

10.1 Objective of the site

The OPUS method produces results (including synthetic data) based on a statistical model and a set of model parameters (with associated uncertainty). While people, especially statisticians, understand sample data when the sampling process is described, more detailed information is required for people to understand synthetic data or a statistical model. Primarily, the users need to understand what the data are, how the data were created, and how reliable the data are. The meta-data, described using an instance of the StatModel XML schema, fulfil this objective. It is a structured XML document that contains all the information necessary to understand the model and generated results. However, it is not easy for users to read and understand a long XML document. Hence, a tool for presenting StatModel XML was created as a part of WP06 package. This tool is a web-based, user-friendly application that can be used to explore the following aspects of a model documented in a StatModel XML file.

- What is the model? How did the modeller represent the underlying physical process/system?
- What are the different model states (representing knowledge about parameters)? How were different model states created?
- How were different data sets used in the model fit process to generate different model states?
- How well-determined are parameter estimates? What information do I have about the precision of various parameters in different model states?
- Where does information about particular parameters come from? Do particular fitting steps appear reliable?

The first three questions address the provenance, and the others address the reliability of the resultant statistical model and derived results.

10.2 The Site Map

The site is designed to address users' questions outlined in the earlier section. Please see below the site-map of the WP06 website that is under construction. Sections of the website are already live, and can be accessed at <http://155.198.92.106>.

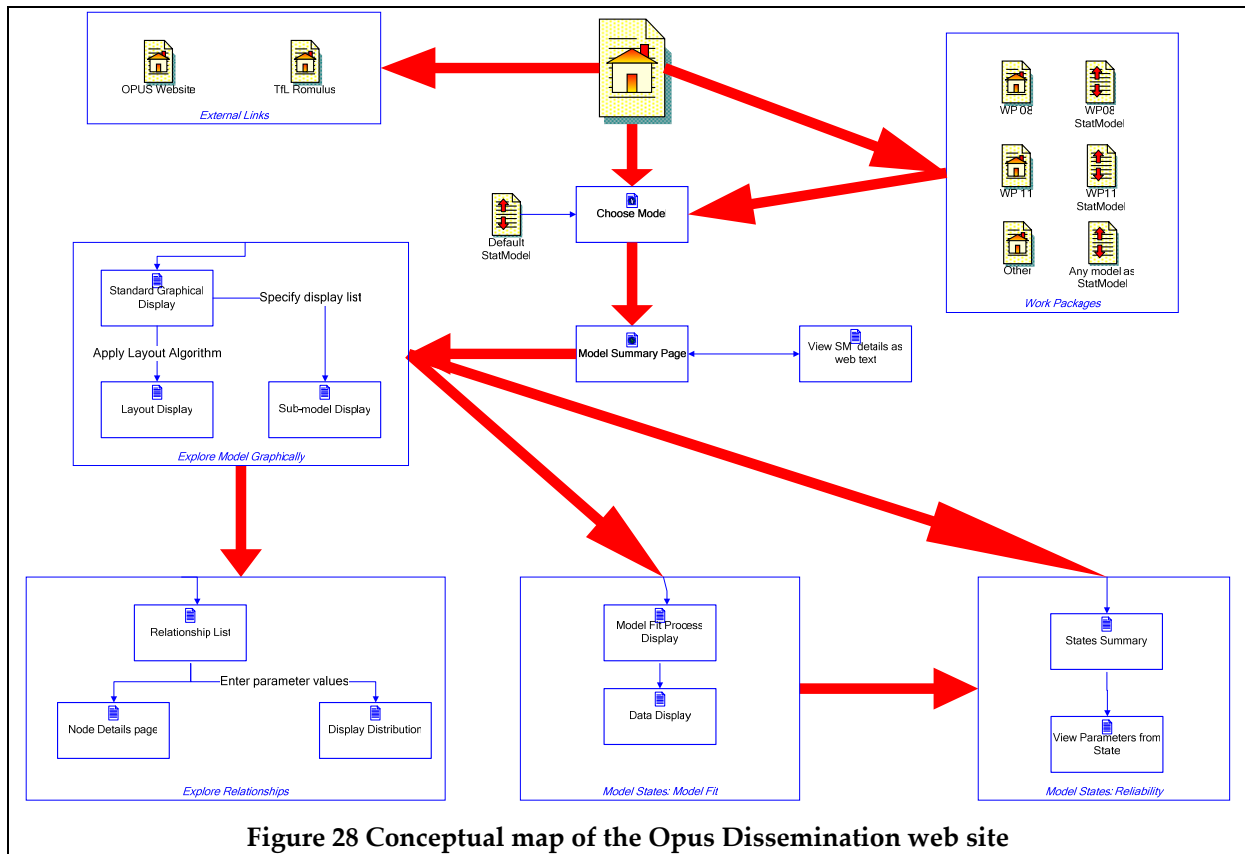


Figure 28 Conceptual map of the Opus Dissemination web site

The page flow or a typical suggested use case of the site is as follows.

1. In order to understand a specific application of the Opus methodology using the site, the first step is to read in the StatModel XML document containing the model specification.
2. A StatModel XML document contains a set of models, representing different abstractions of the underlying physical process. The Model Summary page displays the name and description of all these models.
3. A fully formatted (textual) listing of all the model details is available from this page.
4. The user can select one of the models for further exploration.
 - a. Exploration of the model begins with a graphical display of the full influence diagram of the chosen model.
 - b. The user is able to view the graphical model using different layout algorithms, or can organise the graph manually. A model specification can include display sets selecting a sub-set of nodes. These sub-views focus on specific aspects of a complex model, as chosen by the modeller, to aid comprehension.
 - c. The user also can filter the graph by limiting the display to nodes that are n levels away from the chosen nodes. This allows the user to focus on specific areas of the model.
 - d. The user also can interactively hide or show children or parents of a chosen node. This provides an additional way for the user to reduce complexity of large models.

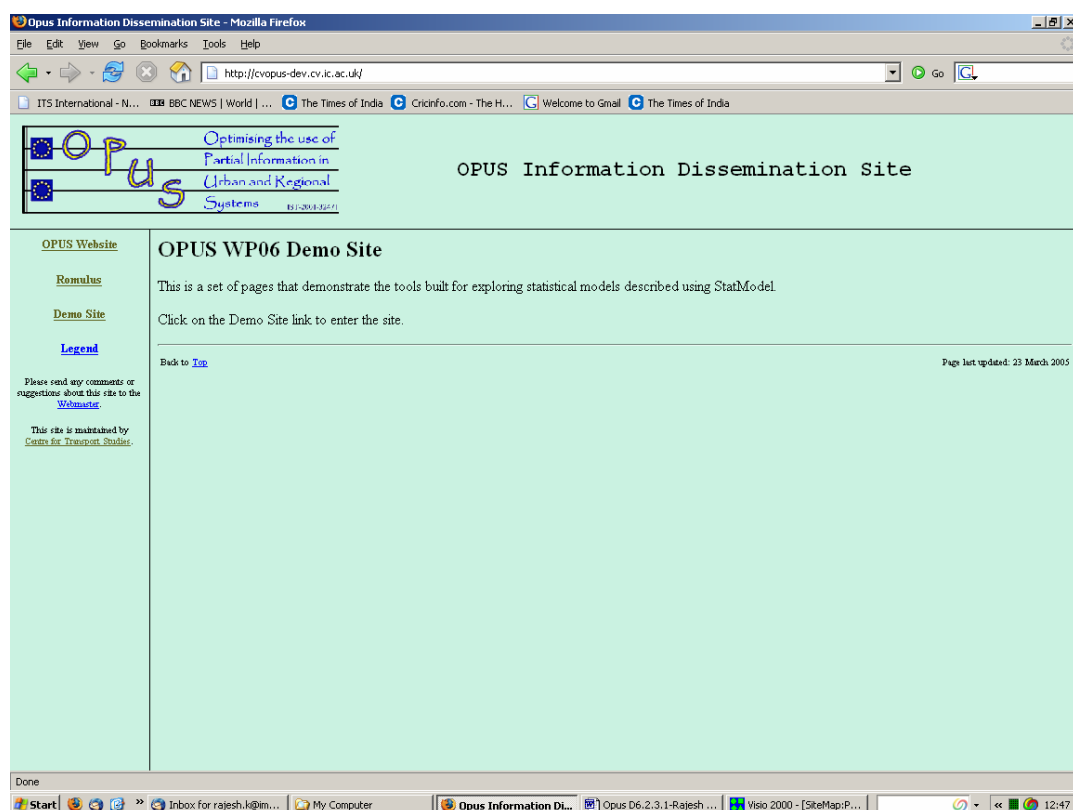
- Once the user has understood the model, mathematical relationships contained in the model can be explored using the relationship page to give more detailed understanding of the underlying model. The user can come back to the main model page from here.
- At this stage, the user is ready to understand how the model was calibrated by looking at the model fitting process. The model starts with a-priori state, and more refined states are generated by bringing in different data sets and combining them using the Opus methodology. The relationship between various datasets, and how they were used to create new model states using the fit process is described graphically on the Model Fit page. The user is able to interact with this process diagram in a way similar to the model diagram.
- The user, at this stage, has understood the provenance of the model, and is ready to understand the reliability of the resultant model. This is done by looking at the parameter uncertainty distributions of different states, and how they change as different data sets are introduced. The user will be able to explore various model states from the Model Fit page, or explore the final model state directly from the Model Display page.

Thus, the website takes the user through the audit trail of the Opus modelling process and helps the user understand how the model is defined, how the data is used, and how reliable are the final model state and any resulting estimates or conclusions.

10.3 Sample Screen Shots from the site

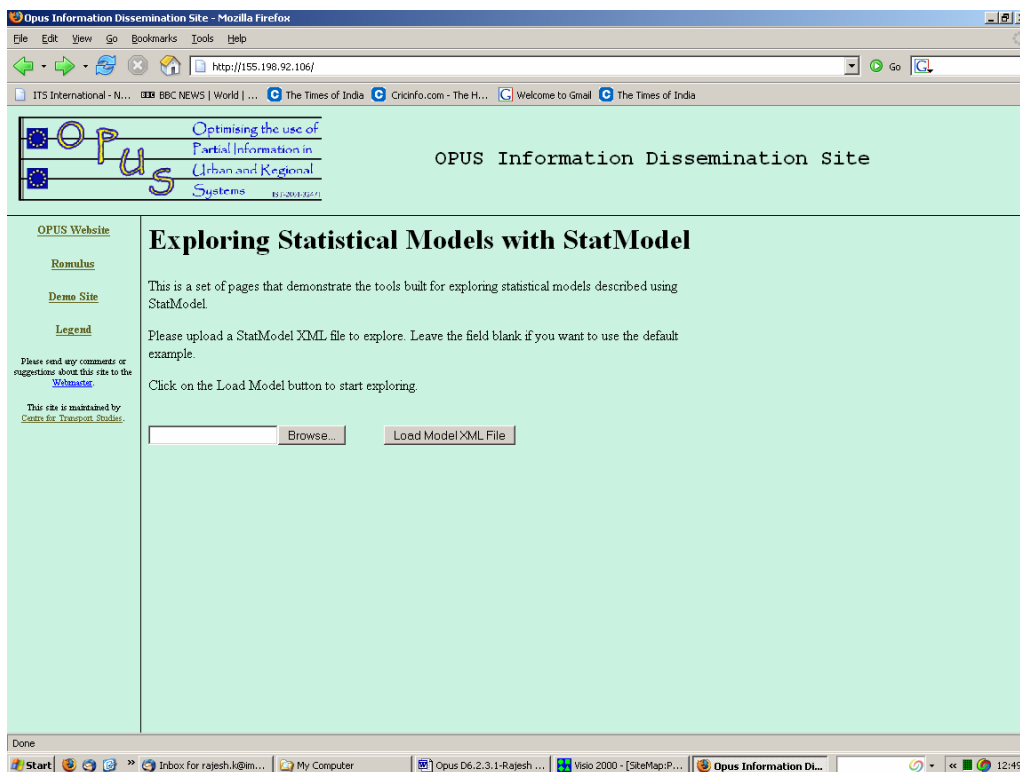
A few screen shots of the site are included in this section. This should give the reader an understanding of the functionality and usability of the site.

10.3.1 WP06 Site Homepage



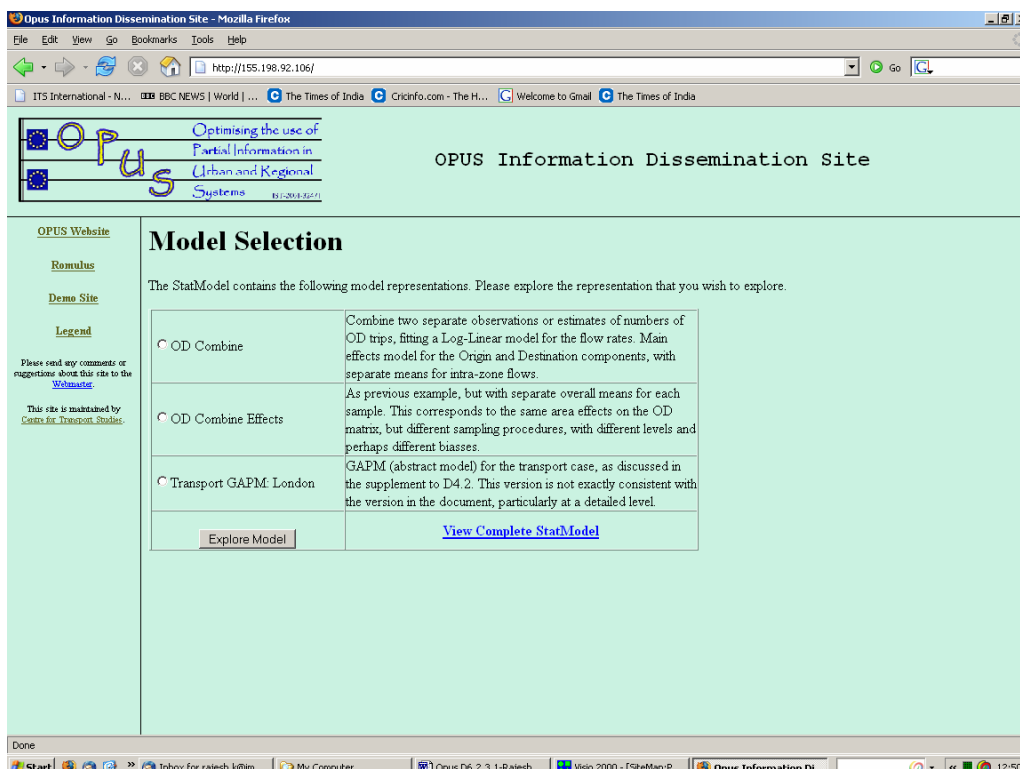
10.3.2 StatModel XML Loader Page

Load a StatModel XML file from this page, or leave the textbox blank to load the default file.



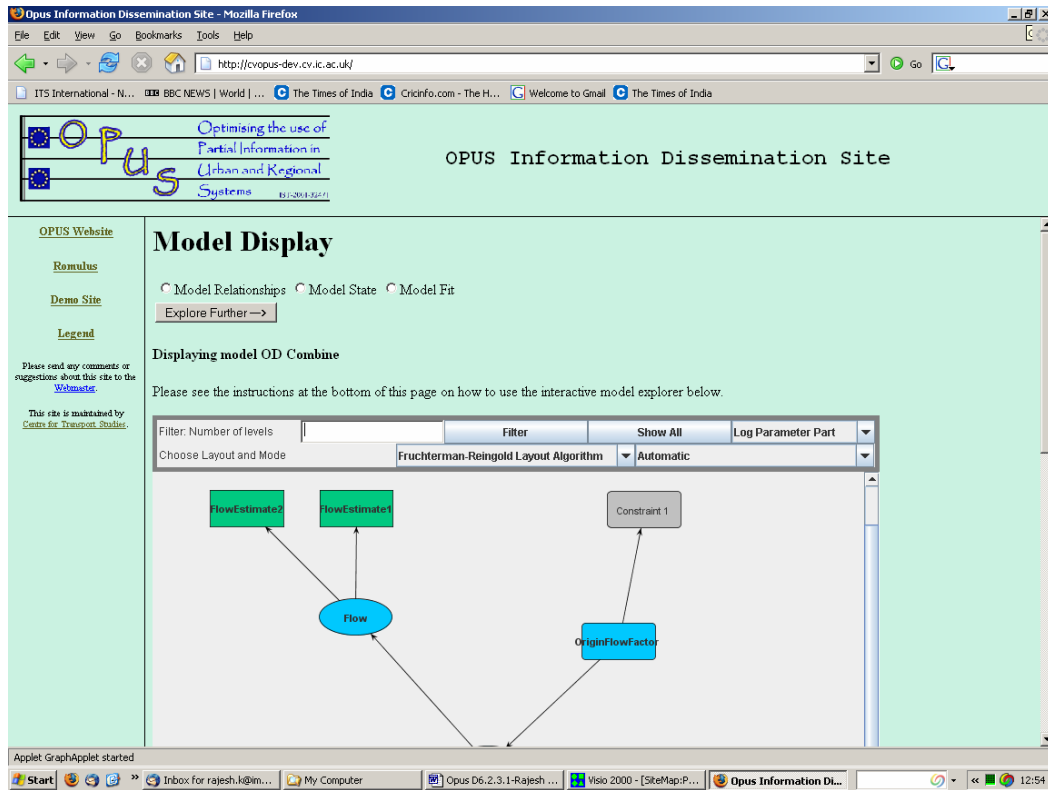
10.3.3 Model Selection Page

This page displays the list of models specified in the StatModel document, and their descriptions. The user can choose a model for further exploration from this page, or view the entire document in the text format by clicking on the link.



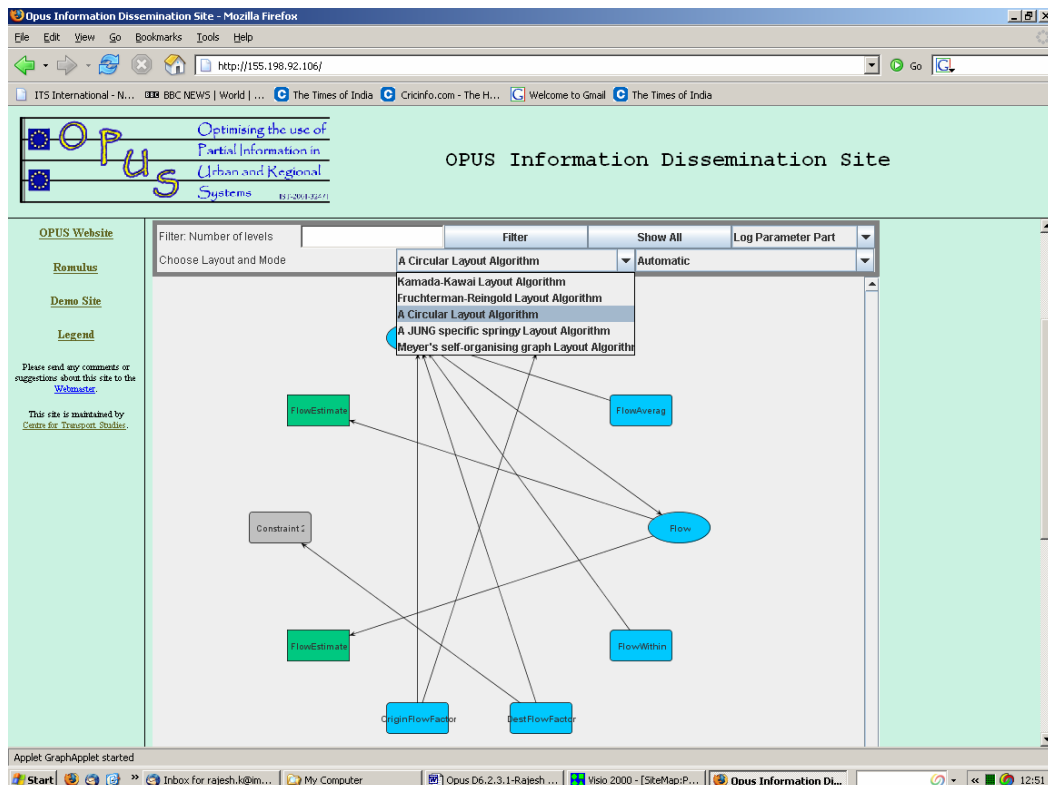
10.3.4 Model Display Page

This is a screen-shot of the model display page with graphical model display applet.



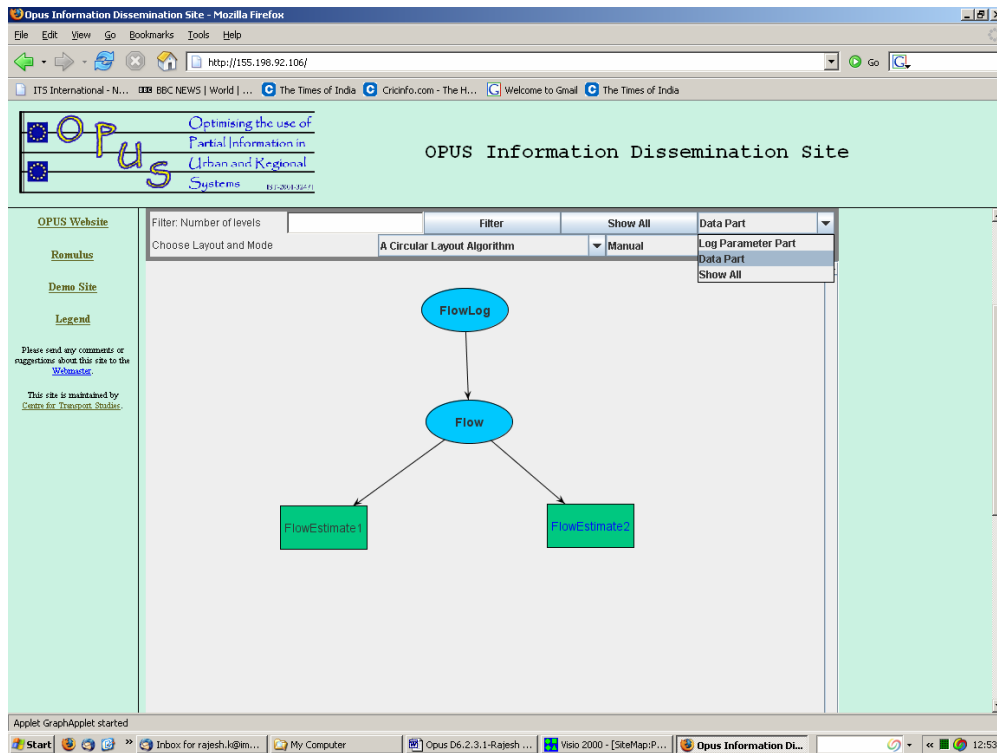
10.3.5 Different Layout Algorithms

The user can select different graph layout algorithms using the drop-down menu.



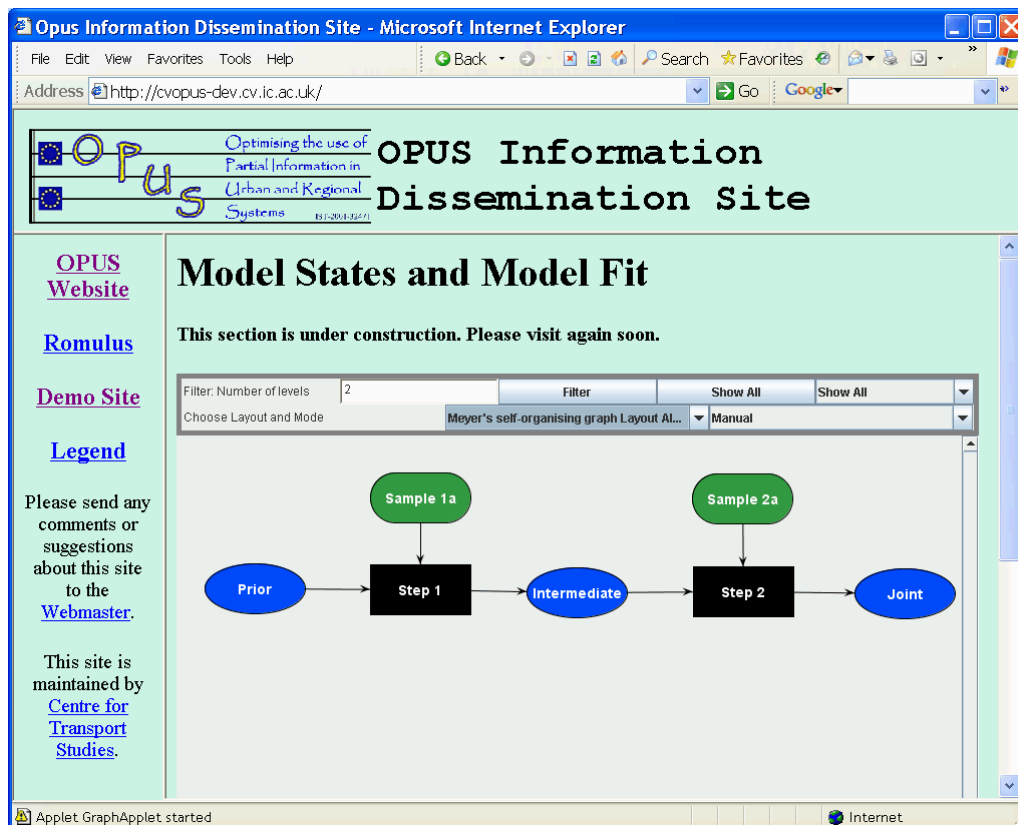
10.3.6 Sub-graph Display

The user also can view pre-defined sub-graphs (defined with the model) using a drop-down menu.



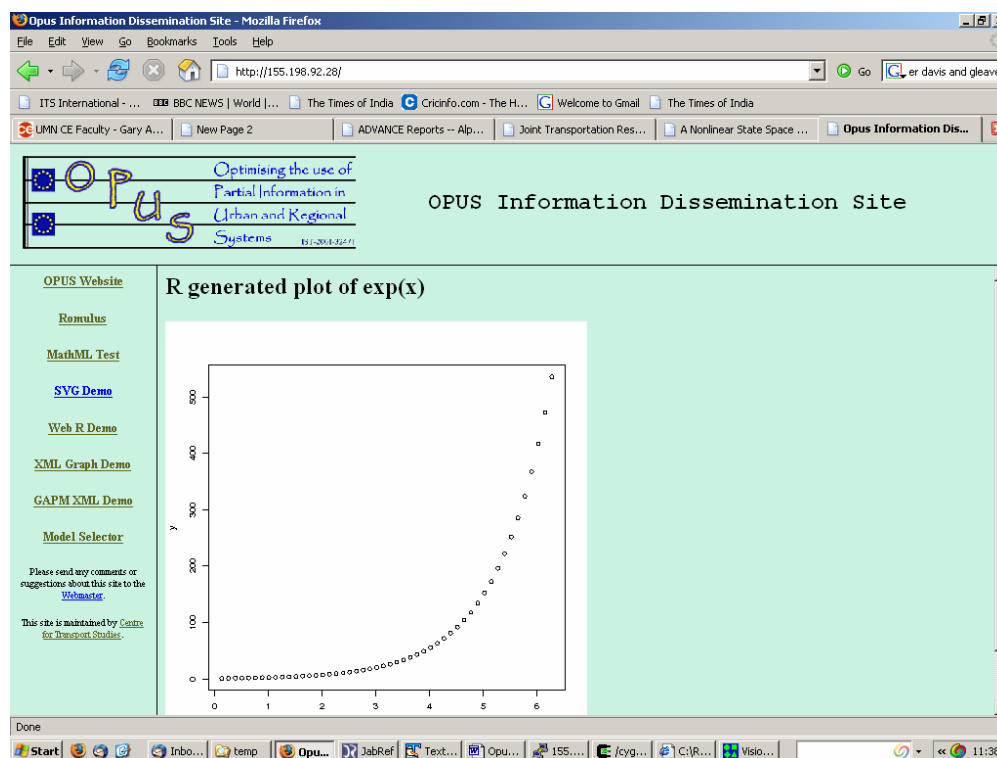
10.3.7 Model Fit Process Display

The process display shows process chains which link model states and fitting steps, and also shows how datasets are used.



10.3.8 A Sample R Graphics page

A technology proof-of-concept page that displays an R-generated graph on a web-page.



10.4 Re-usable components

The site provides a domain-neutral display, but was designed with future extensibility in mind. The components developed for this website could be extended or re-used to build domain specific meta-data exploration applications. The graphical model display and fit process display components are fully re-usable, such that these Java applets could be dropped into other web-pages as long as required parameters are passed to the applet in the prescribed form. The display applets themselves are designed with extensibility in mind, and they can be programmatically extended to change display settings or to add more functionality; in fact, the process display applet is a programmatic extension of the model display applet. All the XML manipulations are done using XSLT transformations, and the XSLT style-sheets could be re-used or modified to build domain specific meta-data exploration tools.

10.5 Continuing Work

The model fit process display is being refined currently. The remaining work includes the design and implementation of a user-friendly way of communicating information about model states.

In addition, feedback on the usability and usefulness of the WP6 site is being gathered from Opus members and others. Improvements to navigation and functionality supporting the exploration process will be incorporated in the final version of the site based on the feedback.

11. FURTHER WORK

Considerable progress has been made in the specification of the StatModel structure and in the elaboration and implementation of presentation functionality. However, there remains considerable scope for further development, and we will continue to add functionality until the end of the project, and perhaps beyond that. We will also try to address other areas discussed in deliverable D6.1, such as presentation to less experienced users and linking to DDI meta-data within Nesstar.

In the very final phase of the project we expect to make some progress on:

- Additional display functionality to support reliability exploration, as discussed in section 5.3 of D6.1.
- Storage mechanism for empirical distributions (as datasets) and links to enable the display of these, including multivariate displays.

Among the areas still awaiting work are:

- Links from model specification to variable and parameter specifications in external repositories, such as DDI.
- Links from the web application to data and meta-data stored in Nesstar data stores. This will include various items, such as:
 - Display of DDI meta-data about variables used in model specification and fitting processes.
 - Links to the specification of datasets used in model fitting steps.
 - Links to results (such as synthetic data) stored in Nesstar.
- Exploration of the integration of the facilities with user-facing analysis facilities (ie Nesstar) based on the results of Opus modelling.

These final issues will only be addressed if additional funding can be found in other projects.

REFERENCES

- [11179] ISO/IEC 11179. Information technology -Meta-data registries (MDR) www.iso.ch.
- [Altova] Altova. See www.altova.com
- [Bugs] WinBUGS. See www.mrc-bsu.cam.ac.uk/bugs
- [CMM] Centre for Multilevel Modelling. *MLWin*. See www.mlwin.com
- [DCMI] Dublin Core Metadata Initiative. See dublincore.org
- [DDIA] DDI Alliance. *Data Documentation Initiative*. See www.icpsr.umich.edu/DDI
- [eGMS] UK GovTalk. *e-GMS, the UK Government meta-data standard*. See www.govtalk.gov.uk/schemasstandards/meta-data_document.asp?docnum=872
- [FormML] *Formulator: Mathematical Equation Editor*. www.hermitech.ic.zt.ua
- [Fowl04] UML Distilled, 3rd Edition. ISBN: 0 321 19368 7. This is an excellent though terse guide to the content and use of UML, aimed at readers with some programming background.
- [FrGV03] Froeschl, K., Grossman, W. and Del Vecchio, V. (2003). *The Concept of Statistical Meta-data*. Deliverable 5 from the MetaNet project.
- [GiRS96] Gilks, W. R, Richardson, S., Spiegelhalter, D.J. (Eds.) (1996) *Markov Chain Monte Carlo Methods in Practice*
- [GraphML] GraphML, Specification available at <http://graphml.graphdrawing.org>
- [GraphXML] GraphXML, <http://gvf.sourceforge.net/>
- [GrKe02] Green, A. and Kent, J-P. *The Meta-data Life Cycle*. Chapter 2: Deliverable 4: Methodology and Tools (2002), Ed. Jean-Pierre Kent, the MetaNet project.
- [GXL] GXL (Graph eXchange Language), <http://www.gupro.de/GXL/>
- [hMxml] Carlson, D.A. et al. *hyperModel Workbench*. www.xmlmodeling.com
- [JGraph] JGraph (Java Graph Visualization and Layout). www.jgraph.com
- [Jung] Java Universal Network/Graph Framework (JUNG), <http://jung.sourceforge.net/index.html>
- [LeZo05] Levinson, D. and Zofka, E. . *Processing, Analyzing, and Archiving Travel Survey Data*. TRB CD Rom. 2005.
- [MathML] Mathematical Markup Language. An XML language for the display and evaluation of mathematical expressions. See www.w3.org/Math.
- [MetaNet] MetaNet: *Network of Excellence for Statistical Meta-data*. www.epros.ed.ac.uk/metanet
- [MP] Design Science. *Math Player – display MathML in a browser*. See www.dessci.com/en/products/mathplayer
- [Nesstar] See www.nesstar.com

- [OJG] OpenJGraph, Java Graph and Graph Drawing Project, <http://openjgraph.sourceforge.net/>
- [PTV] *Visum*. See www.english.ptv.de
- [R] The R project for Statistical Computing. See www.r-project.org/index.html.
- [Rcgi] Web interface for R and Octave. Available at <http://www.ms.uky.edu/~statweb/>
- [Rserve] Interactive Software developed at RoSuDa – Rserve. Available at <http://stats.math.uni-augsburg.de/Rserve/>
- [Rserve] Urbanek S (2003), *Rserve: A fast way to provide R functionality to applications*, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DCS 2003), Vienna, March 2003.
- [SDMX] *Statistical Data and Meta-data Exchange*. See www.sdmx.org
- [Sund03] Sundgren, B. (2003). *Developing and implementing statistical metadata systems*. Deliverable 6 from the MetaNet project.
- [SVG] Scalable Vector Graphics. See www.w3.org/Graphics/SVG/About.html
- [Tomcat] The Apache Jakarta Project. Available at <http://jakarta.apache.org/tomcat/>
- [UML] Object Management Group. UML2.0. A Standard developed under the auspices of OMG (www.omg.org).
- [Whit90] Whittaker, J. *Graphical Models in Applied Multivariate Statistics* (1990). Wiley.
- [XGMML] XGMML (eXtensible Graph Markup and Modeling Language), <http://www.cs.rpi.edu/~puninj/XGMML/>












APPENDIX 1: GRAPH DISPLAY CONVENTIONS

The following conventions have been adopted for the display of model components as graphs. At present this applies to model graphs that show the relationship components, and to process graphs that show the links between Fits, Data and States. Where appropriate, related conventions are used within the generic style sheet display listing.

Colours for Objects

Object	Format	Notes
Variable	Pale Green	Relationship node named with output element name. Add relationship name if present. Bold Text if ElementType='Matrix'
Parameter	Pale Blue	
Constraint	Grey	Use relationship name if present, otherwise 'Constraint <i>n</i> '.
State	Blue	Use name if present, otherwise ID.
Fit	Black, white text	
DataSet	Green	
Tuple	Green	Same colour as dataset
Result	Dark Blue, white text	
Parameter Realisation	Blue/Green	Also for related storage datasets and empirical distribution datasets.

Shapes for Nodes

Node	Shape	Notes
Derived relationship	Rounded Rectangle	
Stochastic relationship	Oval	
Unlinked (input) variable	Rectangle	
Terminal (unlinked) parameter	Lozenge / Rounded Rectangle	More rounded than for Derived 
Constraint	Rounded Rectangle	
State	Oval	
Fit	Rectangle	Process box  would be better
DataSet	Rectangle	Data store shape  would do
Tuple	Rectangle	Same as for dataset, eg 
Result	Oval	Document shape  would do.
Parameter Realisation	Rectangle	Same as Dataset, eg 

Links

Context	Direction	Notes
Relationship	From Input to Output element	
Constraint	From Constraint to Input elements	Because a constraint actually applies to the input element.
State		
Fit	From prior state to fit, from fit to posterior state.	
DataSet	From dataset to Fit	
Tuple	From tuple to variables	
Parameter realisations	From fit to parameter realisation set.	Links to objects that contain empirical distributions based on the realisation would be nice, but are for later.

APPENDIX 2: UPDATING THE PRESENTATION STYLESHEET FOR STATMODEL

The StatModel presentation stylesheet allows a StatModel document to be viewed as a formatted listing in any web browser.

Altova StyleVision is a useful tool for quickly building an XSL/T stylesheet for the display of StatModel instance documents in an HTML browser. However, it does not make use of all possible elements in XSLT, and a number of these are needed for our purposes. This is achieved by manual editing of the stylesheet that StyleVision produces. Most of the steps are about introducing hyperlinks within the document, and activating the display of MathML. The following steps are needed.

MathML namespace

The generated <HTML> statement needs to have the MathML namespace reference added.

```
<html xmlns:m="http://www.w3.org/1998/Math/MathML">
```

Link to MathPlayer

This is needed for Internet Explorer, to tell it to use the MathPlayer plug-in for the display of MathML elements. The following line is added to the <head> section.

```
<xsl:copy-of select="document('SM MathML.inc')" />
```

The text of the link is actually stored in a separate file, and this statement copies it into the target HTML. FireFox does not need this statement, but it does no harm.

Model Bookmarks

At the top of the detailed listing of each model the model name is made into a bookmark, so that it can be the target of a link from the top of the page.

```
<xsl:for-each select="@Name">
  <a><xsl:attribute name="name"><xsl:value-of select="." /></xsl:attribute><span style="font-size:medium; font-weight:bold; "><xsl:value-of
  select="." /></span></a>
</xsl:for-each>
```

ID Bookmarks

All ID entries are potentially the target of links from IDREF attributes, so all need to be flagged. As a convention we always use the string 'ID=' to present such an element, so each needs to be replaced with:

```
<a><xsl:attribute name="name"><xsl:value-of select="." /></xsl:attribute>&#160;ID=</a>
```

Use existing XML for MathML in MathMLExp nodes

The template for the MathMLExp node needs to be changed to copy the existing XML (actually MathML) statements into the HTML, where it will be found by the MathML processor.

```
<xsl:copy-of select="." />
```

Make hyperlinks open a new window

The following line needs to be added wherever there is a line setting an href attribute that is not an internal jump.

```
<xsl:attribute name="target"><xsl:text>_blank</xsl:text></xsl:attribute>
```

APPENDIX 3: A STATMODEL EXAMPLE

A simple StatModel XML document

This is a complete StatModel document, except that the MathML components have been omitted for brevity. A formatted listing of this model follows.

```
<?xml version="1.0" ?>
<StatModel xmlns:m="http://www.w3.org/1998/Math/MathML" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="StatModel.xsd">
<Classifications>
<ExtClassification Name="Zones" ExtClassRef="Zone List">
<Tag TagName="Description">The set of zones used for origins and destinations within these models.</Tag>
</ExtClassification>
<ExtClassification Name="OriginZones" ExtClassRef="Zone List" />
<ExtClassification Name="DestZones" ExtClassRef="Zone List" />
</Classifications>
<Model Name="OD Combine">
<Tag TagName="Description">Combine two separate observations or estimates of numbers of OD trips, fitting a Log-Linear
model for the flow rates. Main effects model for the Origin and Destination components, with separate means for intra-zone
flows.</Tag>
<ModelSpecification>
<Variables>
<Variable Name="FlowEstimate1" VarType="Measure" ElementType="Matrix">
<Tag TagName="Description">Rates as calculated from Sample 1.</Tag>
<Dimension Name="Origin" ClassificationName="OriginZones" />
<Dimension Name="Destination" ClassificationName="DestZones" />
</Variable>
<Variable Name="FlowEstimate2" VarType="Measure" ElementType="Matrix">
<Tag TagName="Description">Rates as calculated from Sample 2.</Tag>
<Dimension Name="Origin" ClassificationName="OriginZones" />
<Dimension Name="Destination" ClassificationName="DestZones" />
</Variable>
</Variables>
<Parameters>
<Parameter Name="Flow" ElementType="Matrix">
<Tag TagName="Description">The matrix of overall mean OD flows used in the Poisson distributions for observed flows. This is
derived from various component influences, in a log-linear model.</Tag>
<Dimension ClassificationName="OriginZones" />
<Dimension ClassificationName="DestZones" />
</Parameter>
<Parameter Name="FlowLog" ElementType="Matrix">
<Tag TagName="Description">The parameter whose log gives the mean flow. This is defined as a sum of linear compo-
nents.</Tag>
<Dimension ClassificationName="OriginZones" />
<Dimension ClassificationName="DestZones" />
</Parameter>
<Parameter Name="FlowAverage" Terminal="true">
<Tag TagName="Description">Factor used for the average (log) number of trips between zones.</Tag>
</Parameter>
<Parameter Name="OriginFlowFactor" ElementType="Matrix" Terminal="true">
<Tag TagName="Description">Factors associated with flow from Origin Zones.</Tag>
<Dimension ClassificationName="OriginZones" />
</Parameter>
<Parameter Name="DestFlowFactor" ElementType="Matrix" Terminal="true">
<Tag TagName="Description">Factors associated with flow into Destination Zones.</Tag>
<Dimension ClassificationName="DestZones" />
</Parameter>
<Parameter Name="FlowWithin" ElementType="Matrix" Terminal="true">
<Tag TagName="Description">Factors associated with flow within Zones (so Destination is the same as Origin).</Tag>
<Dimension ClassificationName="OriginZones" />
</Parameter>
</Parameters>
<Relationships>
<Relationship RelType="Stochastic" Name="Estimate 1 distribution">
<Tag TagName="Description">Poisson distribution for observations in first estimate set, based on common rates.</Tag>
```

```

<RelInput>
<ParRef Name="Flow" />
</RelInput>
<RelOutput>
<VarRef Name="FlowEstimate1" />
</RelOutput>
<RelStochastic>
<DistPoisson>
<Rate>
<ParRef Name="Flow" />
</Rate>
</DistPoisson>
</RelStochastic>
</Relationship>
<Relationship RelType="Stochastic" Name="Estimate 2 distribution">
<Tag TagName="Description">Poisson distribution for observations in second estimate set, based on common rates.</Tag>
<RelInput>
<ParRef Name="Flow" />
</RelInput>
<RelOutput>
<VarRef Name="FlowEstimate2" />
</RelOutput>
<RelStochastic>
<DistPoisson>
<Rate>
<ParRef Name="Flow" />
</Rate>
</DistPoisson>
</RelStochastic>
</Relationship>
<Relationship RelType="Derived">
<Tag TagName="Description">Poisson rates are derived as exponential of (linear) flow function.</Tag>
<RelInput>
<ParRef Name="FlowLog" />
</RelInput>
<RelOutput>
<ParRef Name="Flow" />
</RelOutput>
<RelDerived>
<MathMLExp>
+ <m:math display="block">
</MathMLExp>
</RelDerived>
</Relationship>
<Relationship RelType="Derived">
<Tag TagName="Description">Linear function for log of flow rates. Inter-zone flow is modelled as an average flow adjusted by
origin and destination factors (with no interaction). Intra-zone flows are modelled separately</Tag>
<RelInput>
<ParRef Name="FlowAverage" />
<ParRef Name="OriginFlowFactor" />
<ParRef Name="DestFlowFactor" />
<ParRef Name="FlowWithin" />
</RelInput>
<RelOutput>
<ParRef Name="FlowLog" />
</RelOutput>
<RelDerived>
<MathMLExp>
+ <m:math display="block">
</MathMLExp>
</RelDerived>
</Relationship>
<Relationship RelType="Constraint">
<Tag TagName="Description">Origin factors sum to zero, so product of rate components is one.</Tag>
<RelInput>
<ParRef Name="OriginFlowFactor" />
</RelInput>
<RelDerived>
<MathMLExp>
+ <m:math display="block">

```



```

</MathMLExp>
</RelDerived>
</Relationship>
<Relationship RelType="Constraint">
<Tag TagName="Description">Destination factors sum to zero, so product of rate components is one.</Tag>
<RelInput>
<ParRef Name="DestFlowFactor" />
</RelInput>
</RelDerived>
<MathMLExp>
+ <m:math display="block">
</MathMLExp>
</RelDerived>
</Relationship>
</Relationships>
</ModelSpecification>
</Model>
</StatModel>

```

Formatted model listing

The following demonstrates the appearance of the XML document above when viewed through the StatModel stylesheet in a web browser.

Models in this StatModel document

OD Com- bine	Combine two separate observations or estimates of numbers of OD trips, fitting a Log-Linear model for the flow rates. Main effects model for the Origin and Destination components, with separate means for intra-zone flows.
--------------------------------------	---

Model: OD Combine

Combine two separate observations or estimates of numbers of OD trips, fitting a Log-Linear model for the flow rates. Main effects model for the Origin and Destination components, with separate means for intra-zone flows.

Variables:

Name	Structure	Type	
FlowEstimate1	Matrix (Dimensions: Origin, using Origin-Zones and Destination, using DestZones)	Measure	Rates as calculated from Sample 1.
FlowEstimate2	Matrix (Dimensions: Origin, using Origin-Zones and Destination, using DestZones)	Measure	Rates as calculated from Sample 2.

Parameters:

Name	Structure	
Flow	Matrix (Dimensions: Origin-Zones and DestZones)	The matrix of overall mean OD flows used in the Poisson distributions for observed flows. This is derived from various component influences, in a log-linear model.
FlowLog	Matrix (Dimensions: Origin-Zones and DestZones)	The parameter whose log gives the mean flow. This is defined as a sum of linear components.
FlowAverage (terminal)		Factor used for the average (log) number of trips between zones.

Name	Structure	
OriginFlowFactor (terminal)	Matrix (Dimensions: Origin-Zones)	Factors associated with flow from Origin Zones.
DestFlowFactor (terminal)	Matrix (Dimensions: Dest-Zones)	Factors associated with flow into Destination Zones.
FlowWithin (terminal)	Matrix (Dimensions: Origin-Zones)	Factors associated with flow within Zones (so Destination is the same as Origin).

Relationships:

Name	Output	Input	Form
Estimate 1 distribution	Stochastic FlowEstimate1	Flow	Distribution: Poisson: Rate=Flow Poisson distribution for observations in first estimate set, based on common rates.
Estimate 2 distribution	Stochastic FlowEstimate2	Flow	Distribution: Poisson: Rate=Flow Poisson distribution for observations in second estimate set, based on common rates.
	Derived Flow	FlowLog	Derivation: exp(FlowLog) Poisson rates are derived as exponential of (linear) flow function.
	Derived FlowLog	FlowAverage, OriginFlowFactor, DestFlowFactor, FlowWithin	Derivation: For : $i \in \text{OriginZones}$, $j \in \text{DestZones}$ If ($i \neq j$) then {FlowAverage + OriginFlowFactor[i] + DestFlowFactor[j]} If ($i = j$) then {FlowWithin[i]} Linear function for log of flow rates. Inter-zone flow is modelled as an average flow adjusted by origin and destination factors (with no interaction). Intra-zone flows are modelled separately
	Constraint	OriginFlowFactor	Derivation: $\sum \text{OriginZoneFactor} = 0$ Origin factors sum to zero, so product of rate components is one.
	Constraint	DestFlowFactor	Derivation: $\sum \text{DestZoneFactor} = 0$ Destination factors sum to zero, so product of rate components is one.

[Back to Top](#)

Classifications:

External	Zones from Zone List	The set of zones used for origins and destinations within these models.
	OriginZones from Zone List	
	DestZones from Zone List	