OPUS

Optimising the use of Partial information in Urban and regional Systems

Project IST-2001-32471

WP6: Database Systems

Title : Provenance and Reliability: Managing Metadata for Statistical Models

Creator (Author): Andrew Westlake — Survey & Statistical Computing, AJW@SaSC.co.uk

Contributor :

Identifier : AW PR SSDBM
Status : As published, but reformatted
Type : Conference Paper, presented at SSDBM 18, Vienna, 2006
Version : Final
Date.Created : 17 June 2005
Date.Modified : 14 April 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.
This paper draws heavily on Deliverable D6.2.

Contract Date : April 2003
Publisher (Project Co-ordinator) : 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

Provenance and Reliability: Managing Metadata for Statistical Models

Andrew Westlake
Centre for Transport Studies, Imperial College London
A.Westlake@Imperial.ac.uk

Abstract

In this paper we present a general model of the information (meta-data) needed to represent statistical modelling and its results. We discuss the design process for this model (in terms of both objectives and methodology), along with the resultant design. However, we recognise that it is not enough to design a static structure for the meta-data, we also need functionality to support the end users of the statistical results in exploring the provenance and reliability of the results. We have identified some generic functionality that is useful in this, but also recognise that its presentation must be tailored to the domain of interest and the level of understanding of the user.

1. Introduction

1.1. Overview

The past decade has seen significant advances in the acceptance and use of the concept of meta-data, auxiliary information that informs about other information. General standards such as the Dublin Core [1] are widely discussed, and have led to specific extensions, such as the UK governments e-GMS standard [2].
In the statistical field there have been many useful activities. The MetaNet project [3] brought together many practitioners and produced much useful material, though without recommending any particular meta-data models for applications. Data Description is increasingly well served. The basic triple-s standard [4] is used for data exchange among many market research and survey systems, the DDI Codebook standard [5] is gaining acceptance in the social science and data archive community, and the SDMX initiative [6] is continuing to develop a highly engineered (UML-based) set of standards for data interchange between government and international agencies.
For the Opus project [7] we needed to take the idea of meta-data beyond that of data description, into the area of statistical models. Our aim is to support the user of results from statistical modelling exercises by providing information about the structure and technical processes used in calibrating a statistical model (its Provenance), and about the quality of the results obtained from it (its Reliability). To this end we have designed a structure for the representation of this meta-data, and have proposed (and explored) functionality to support its use. Amongst other things, this can be thought of as providing an audit trail to the statistical process that underpin statistical results. There is scope (and a recognised need) to extend these ideas into the area of the design, development and exchange of the statistical models. We have not explored this path within the pro-

ject, though we believe that the structure is strong enough to support future development in that area.

1.2. The OPUS Project

The OPUS project is developing a methodology for the integration of information from multiple data sources about complex systems. The approach is centred on the construction of one or more statistical models which represent components of interest from the real system, followed by the application of existing statistical ideas such as Bayesian Methods and MCMC [8]. We are not attempting to develop any new Statistical Methods. Rather, the 'Opus Methodology' aims to show how to use these methods in a consistent and coherent way with complex systems.
Our main target domain for trials of the OPUS methodology is Transport, with particular emphasis on the transport system in London. Transport Engineers (as others) are often sceptical about results from 'models'. They are concerned about the assumptions used and their implications, as well as about the quality of any results or conclusions obtained. This is entirely reasonable, and so we accept the responsibility of providing information and functionality that can be used to provide reassurance and create confidence related to any results from our methodology. We characterise these issues as being about the Provenance and Reliability of any results from a statistical model, and see this as a meta-data problem.
Our aim is to provide information for users' of statistical results. We are not attempting to capture the data analysis process that leads to the final formulation of the statistical model – that is a fascinating but different area to which contributions have been made by various authors including early SSDBM papers [9, 10] and in Statistical Expert Systems [e.g. 11].

1.3. Statistical Models in OPUS

Each statistical model will be a generic representation of (part of) the application domain, and will include variables, parameters and mathematical relationships, the latter both deterministic and stochastic. Stochastic terms are used to represent both variability in observations and uncertainty about parameters.
Once a (generic) model has been constructed, information about the model is extracted from datasets by various model fitting processes. We assume that different datasets relate to different (though usually overlapping) aspects of the system, or are at different levels of aggregation, as otherwise it would be better to simply combine the datasets directly. Information from the data has the effect of reducing the uncertainty associated with parameters. Once all information has been extracted and combined into the model, results from the final state of the model are reported in terms of the uncertainty distributions of the parameters (often summarised into best estimates and confidence limits) and of measures derived from the parameters. The methodology is generic in that the statistical models can relate to any domain.

1.4. Terminology for Models

The term Model is used widely, and means different things in different contexts. As statisticians we usually use the term Statistical Model to refer to a specification which is specific about the variables, parameters and relationships involved, but generic in that it does not need to have been calibrated against data. Indeed, if more data arises the statistical model is not changed. In con-

trast, users tend to use the same term to refer to the calibrated version, from which statistical results are derived, so that the model changes if more data arrives that can improve the fit. (We refer to this as a *state* of the model.)
In computing, the term model is used to refer to abstractions that arise during design processes for computing systems. It is in that sense that we use the term when we refer to meta-data models.

2. Meta-Data for Statistical Models

2.1. Objectives

Our objective has been to design a structure (which we refer to as StatModel) for the representation of information about statistical models, together with appropriate functionality for the presentation of that information.
The data structure must include information about

- the structure of the statistical model used;
- the processing steps used to calibrate the model against data (model fitting);
- datasets used in model fitting steps;
- parameter estimates and uncertainty resulting from fitting (a model state);
- results derived from a model state

This information 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. This requires exploration of the mappings from application-specific views to the StatModel view.
Our immediate purpose for this information 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.

2.2. 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 [12] 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.

2.3. Structures

The structure and semantics of the classes used to hold information about a statistical model are specified in UML using the *hyper*Model Workbench application [13]. The main components of this structure for a single model are shown in Figure 1, and are elaborated in section 4. Information about multiple models can be stored together and they can make cross-references.
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, though perhaps changing over time), 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.
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.
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.

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.

2.4. 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, the model falls into the class known by statisticians as Graphical Models [14].
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’ or ‘free’. Uncertainty distributions must be supplied for them in any model state.

2.5. 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

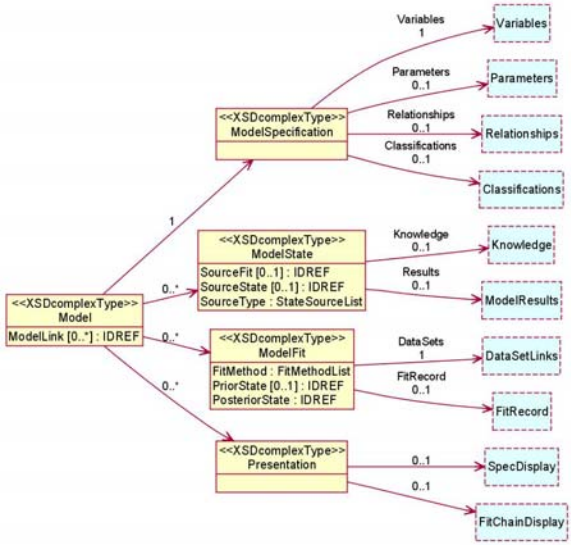

Figure 1. Main components of a model

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 meta-data about statistical models. Instead we have concentrated on the development of generic presentation components that can be used as building blocks for such systems.
A generic facility for listing all the components of a StatModel instance is directly useful to specialists, and provides a resource 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 [15, 16].

3. Development with StatModel

3.1. Construction of the model

The static structure for meta-data about statistical models (StatModel) was designed using the *hyper*Model Workbench tool. This is a plug-in for the Eclipse IDE, and supports the building of UML class diagrams. It also includes a profile for XML Schemas, and can generate the equivalent schema (XSD) files from the class diagram. It uses the standard XMI format for model storage, so the resulting model can be transferred to another more complete UML package for further elaboration.
Working with UML rather than an XML schema design tool allows us to use object constructs (such as specialisation) more easily, and allows us to extend the design beyond static structures into manipulation and presentation functionality.
The resultant XML schema files can be used with any validating XML editor to create XML documents that are instances of the StatModel structure. We have used the XMLSpy application from the Altova XML Suite [17] for this. We have also used the StyleVision application from Altova for the construction of style sheets for generic presentation of the model instances.

3.2. Construction of model instances

A StatModel instance is an XML document containing the meta-data about one or more statistical models. For the moment we are building these by hand in the XMLSpy editor 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.
We have built a stylesheet in StyleVision that can be used with the Altova Authentic editor for the entry and editing of core parts of StatModel instances. Most of the statistical models we are exploring are being designed initially in the WinBUGS [18] application, and we are exploring a converter from WinBUGS script that would produce the main parts of a StatModel instance.

3.3. Use in Modelling Packages

In the event that the StatModel proposal is found acceptable we would expect the developers of appropriate statistical modelling software to provide an interface to StatModel, both saving mod-

els in this form and allowing models from elsewhere to be imported. Model design systems would use and create the Model Specification components of the instances, and 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 [19] to at least explore whether our proposals could be compatible with their systems. Other standards, such as PMML [20] are also relevant to the more complex question of model exchange between fitting packages. We see StatModel as complementary to such initiatives, because they focus on the details of the implementation of model fitting, whereas we concentrate on the specification of models at a somewhat more abstract level.

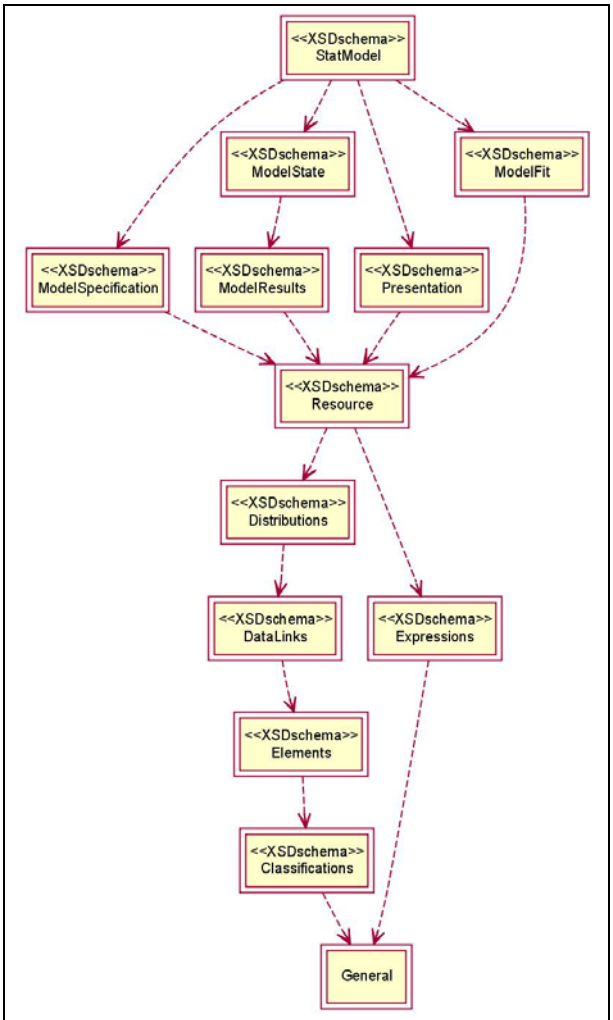

Figure 2. Packages in StatModel UML

4. Documenting Statistical Models

4.1. Introduction

Finally we arrive at the details of the components of the StatModel structure. Figure 1 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 available from the Opus project web site as deliverable 3.1 (version 2) [21], and a more extensive version of the presentation here is available as deliverable 6.2 [22].
In the following sections the term ‘model’ refers to the Model component of the StatModel structure, and to the statistical model that it represents. Figure 2 shows how the model is broken down into packages, and the dependencies between them. Each package corresponds to an XML Schema file, and dependencies are implemented through inclusion.

4.2. Variable and Parameter Elements

The distinction between these components has already been discussed. Both are specialisations of the general Element class (see Figure 3), so can have a Type 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.

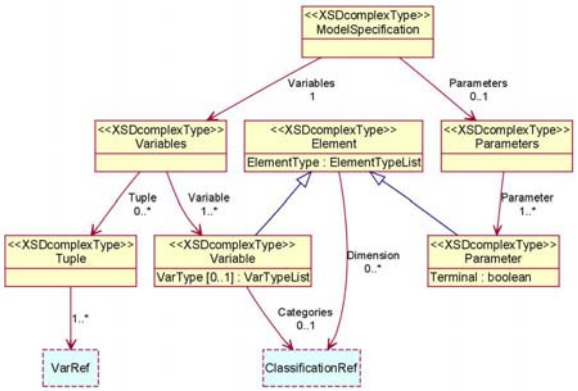

Figure 3. Variables and Parameters

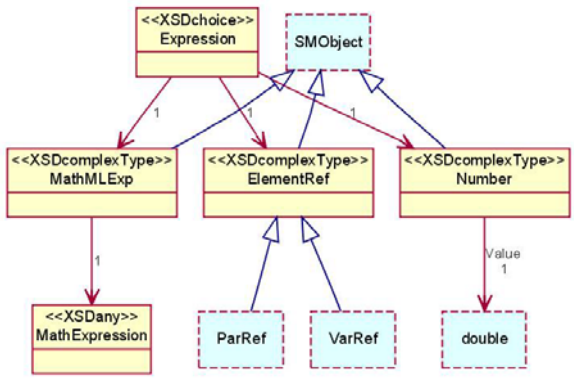

Figure 4. References and Expressions

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

4.3. References and Expressions

References (by Name) to variables and parameters are specialisations the more general Element reference (Figure 4). 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.

The SMObject component shown here is the generalisation from which all classes in the UML model inherit. This provides a centralised mechanism for associating standard attributes with all classes. This is used for Name and ID attributes, and for a general mechanism to associate comments with any object within a StatModel instance.

4.4. Relationships

Relationships can be Derived or Stochastic (Figure 5). A relationship always has an Input component consisting of at least one element (variable or parameter).

The form of the relationship can be omitted – this can 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

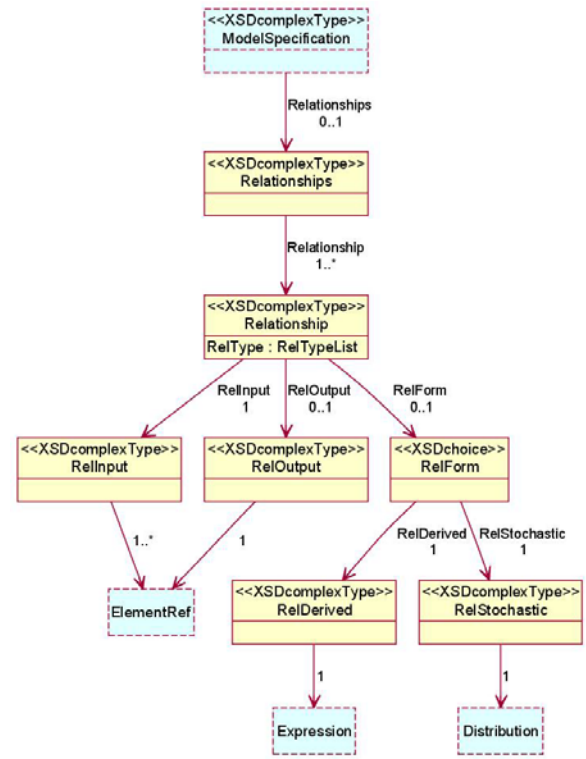

Figure 5. Relationships

to a statistical distribution, which in turn will have expressions for its parameters. Constraints are implemented as derived relationships without an output element – the output value must be 'True'.

4.5. 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 (see Figure 6 – 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.

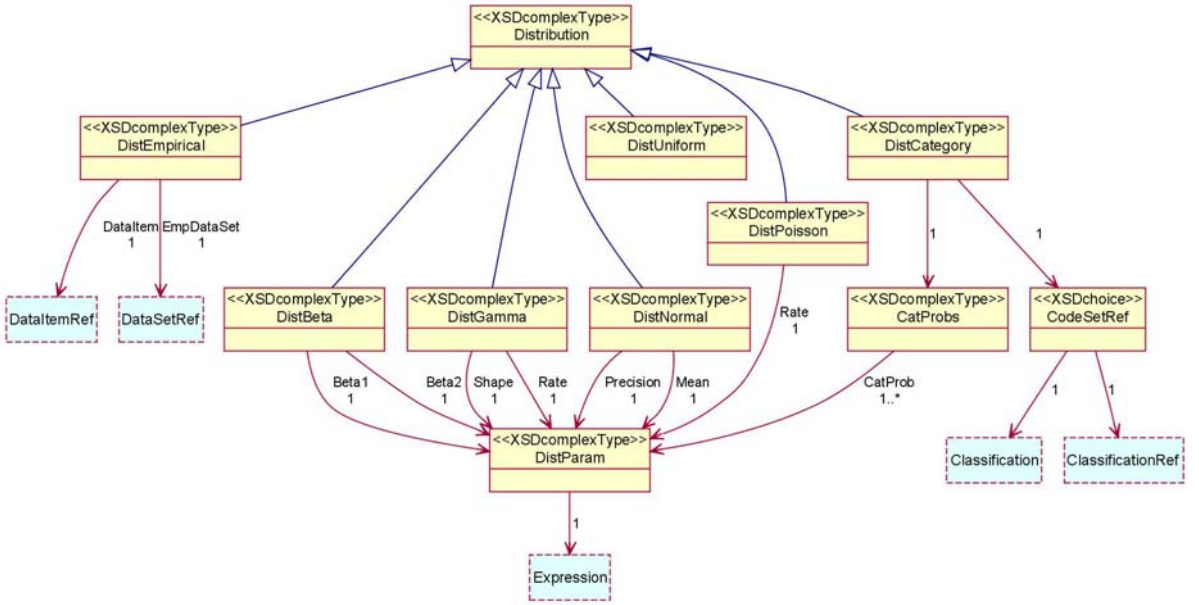

Figure 6. Distributions

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 tuples, 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.

4.6. 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' (or 'free') 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.

With standard Bayesian methodology there will usually only be two states associated with a model, linked by a single fitting step. The Opus methodology extends this and envisages chains of states linked by sequences of fitting steps over distinct data sources. Equally valid, it is possible 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.

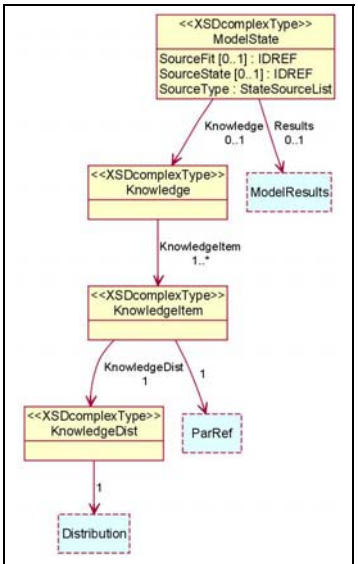

Figure 7. Model State

4.7. 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 representations of uncertainty, in the form of uncertainty limits (equivalent to confidence limits) or some representation of the full distribution. For parameters that are derived from the terminal parameters, corresponding estimates and distributions can be derived.

4.8. 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 metadata 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.

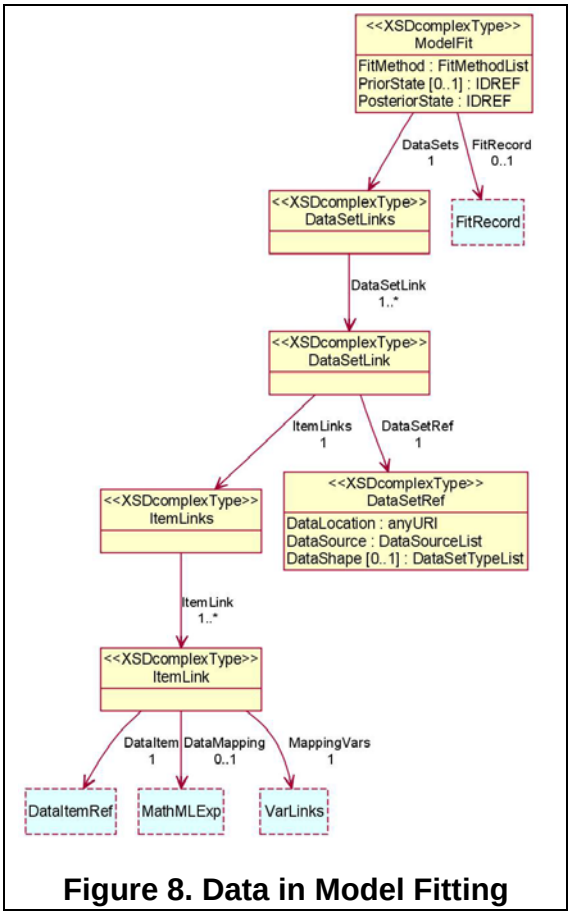

Figure 8. Data in Model Fitting

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!

4.9. 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. 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.

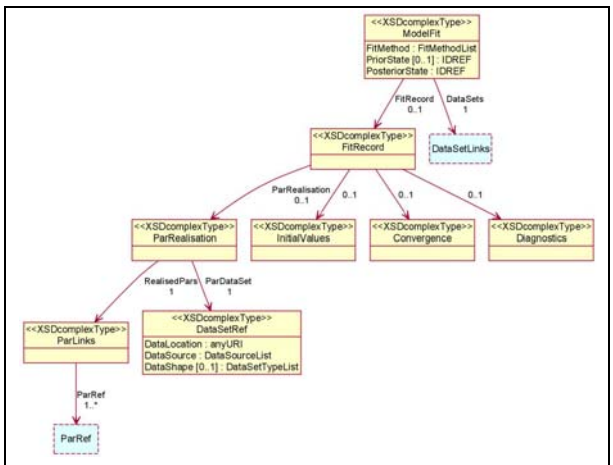

Figure 9. Recording Model Fitting

5. Presentation Examples

5.1. 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 10 shows a fragment of the listing for a single complex model that has been used as a test case. 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 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. Everything in the listing (apart from the italicised headings) is taken directly from the instance document.
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.

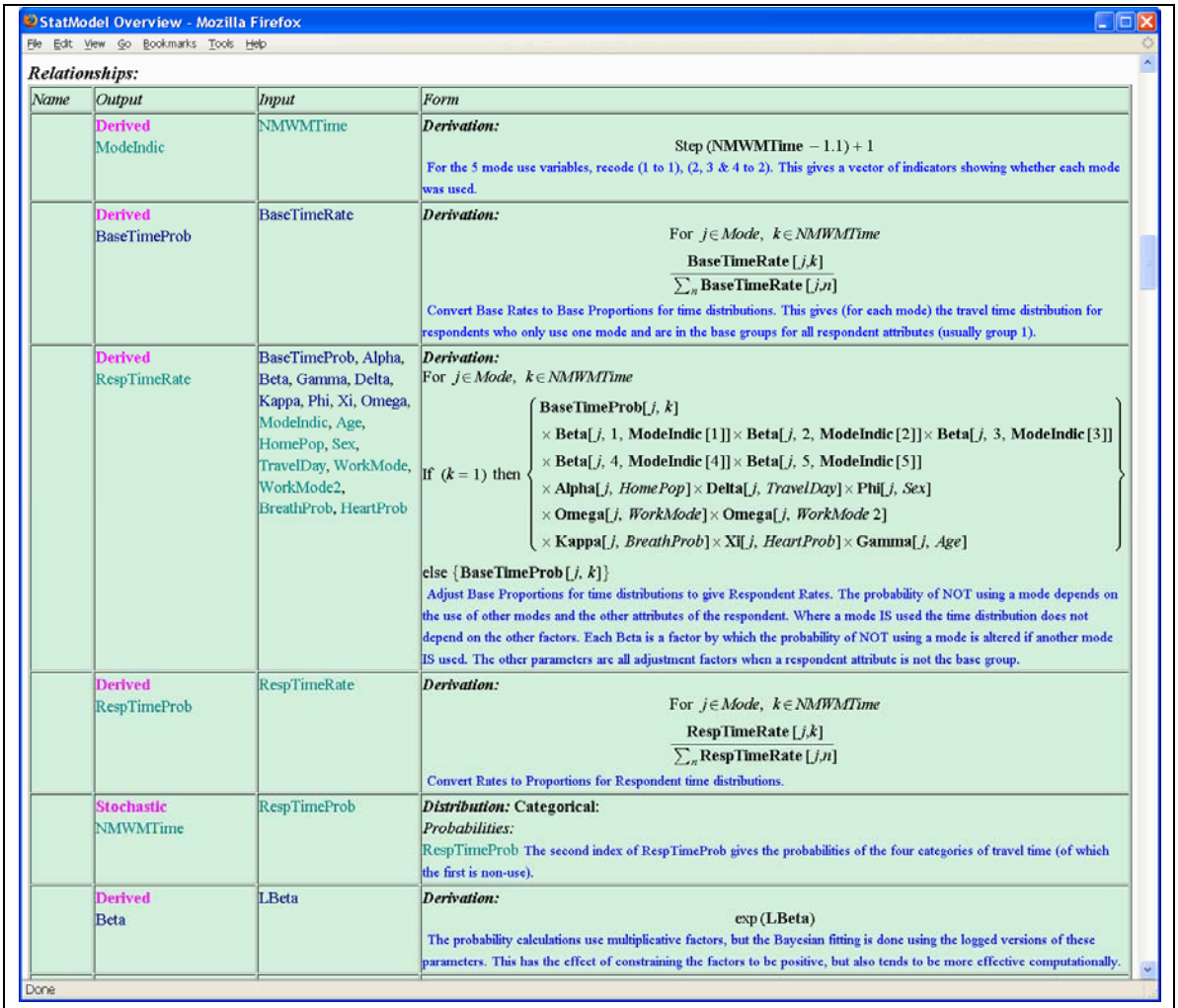

Figure 10. StatModel Listing (part)

5.2. Graphical display of model relationships

Components have been developed to provide dynamic displays of parts of the StatModel instances within a web browser. Figure 11 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 10 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 graph layout algorithms, the user can zoom and pan the display, and can select to manually adjust the location of individual nodes. Various graph trimming and filtering options are provided, and interesting sub-graphs can be pre-defined. Colour coding is used to distinguish between variables (green) and parameters (blue) in the display, and the node shapes distinguish between derived, stochastic and other relationships. Constraints are shown in light grey. Figure 11 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.

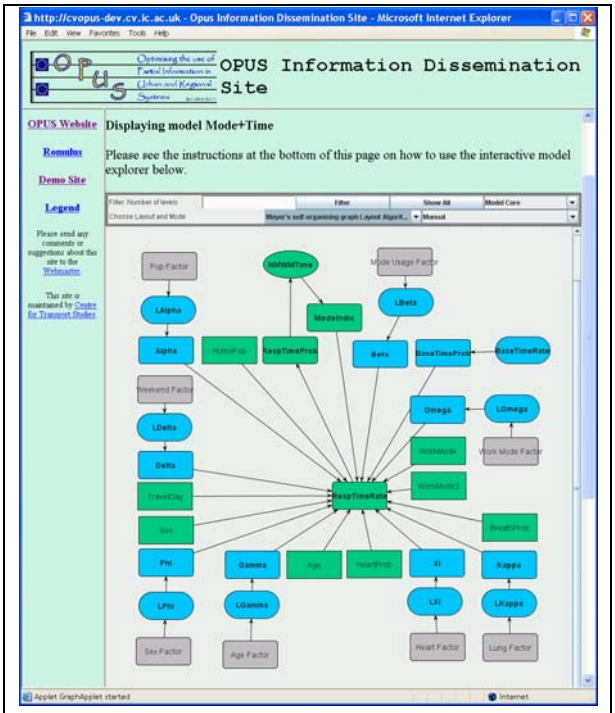

Figure 11. Full influence graph in browser

Figure 12 shows two sub-graphs. The left one extracts just the part that relates to the response variable 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 lozenge shape. These figures are standardised 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 to give probabilities (in RespTimeProb) which are used in the stochastic relationship (indicated by an oval) to define NMWMTime. ModeIndic is a vector of derived indicators for whether each mode was used at all, which feeds back into the calculation of RespTimeRate.

The right-hand part illustrates how some of the covariates contribute to the calculation of RespTimeRate: Age selects a Gamma parameter, Phi relates to Sex, and Beta to Mode Us-

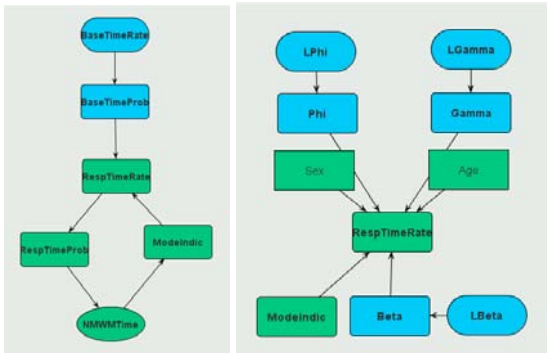

Figure 12. Sub-graphs

age. The mathematics of these relationships can be read from Figure 10.

5.3. Comparison of models

The influence graphs can make it straight forward to see the differences between related models. For example, Figure 13 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 z\ones in a transport system, and a combined estimate of the parameters is desired.
Figure 13 treats the two estimates as independent samples from exactly the same stochastic process. In contrast, Figure 14 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 factors are still shared, but it is recognised that the average level of flow obtained from the two datasets may have been influenced by different biases, so are estimated separately.

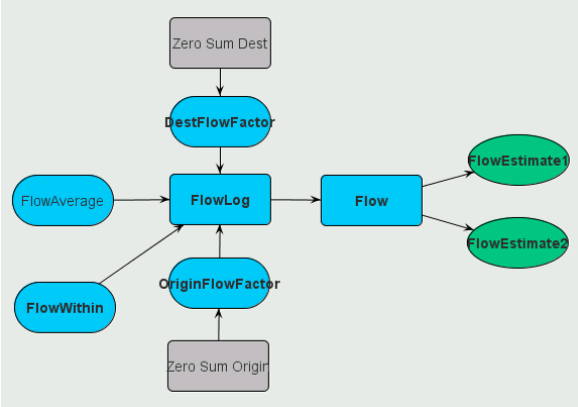

Figure 13. Model with independent estimates

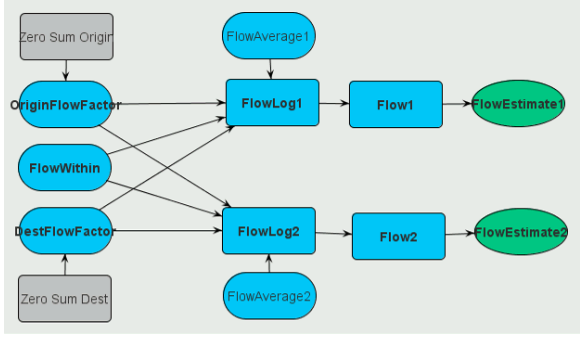

Figure 14. Model with differentiated estimates

5.4. Other displays

A display showing model fitting and state chains (with data sets) is provided, using the same display applet.
Initial experiments using R [23] for the display and comparison of posterior empirical distributions have been successful, but we are currently waiting for the supply of suitable result data from other teams within the project before these are added to the application.

6. Acknowledgements

The work reported in this paper (and the whole Opus project) is funded as Project IST-2001-32471, part of the Fifth Framework Information Society Technologies programme of the European Community, managed through Eurostat.

Thanks are due to Opus colleagues Miles Logie and Saikumar Chalisani in the development of the original ideas for the presentation of meta-data about statistical models, and to Rajesh Krishnan for the implementation of the presentation application. Thanks are also due to a referee for suggestions about related work.

7. References

[1] Dublin Core Metadata Initiative. See dublincore.org
[2] UK GovTalk. *e-GMS, the UK Government meta-data standard*. See www.govtalk.gov.uk/ schemasstandards/ metadata_document.asp?docnum=872
[3] MetaNet: *Network of Excellence for Statistical Metadata*. See www.epros.ed.ac.uk/metanet
[4] Hughes, K., Jenkins, S. and Wright, G. *The Triple-S Survey Interchange Standard*: See www.triple-s.org
[5] DDI Alliance. *Data Documentation Initiative*. See www.icpsr.umich.edu/DDI
[6] SDMX.. *Statistical Data and Meta-data Exchange*. See www.sdmx.org
[7] The Opus Project. See www.opus-project.org
[8] Gilks, W.R., Richardson, S., Spiegelhalter, D.J. (Eds.) *Markov Chain Monte Carlo Methods in Practice* (1996). Chapman and Hall, London, UK
[9] Cowley, P., Nicholson, W. & Carr, D., "Managing the Data Analysis Process", *Proceedings of the 3rd SSDBM Workshop*, 1986, Eurostat, Luxembourg, pp 72-77.
[10] Becker, R., & Chambers, J., "Auditing of Data Analysis", *Proceedings of the 3rd SSDBM Workshop*, 1986, Eurostat, Luxembourg, pp 78-80.
[11] Wolstenholme D.E., Nelder J.A. "A Front End to GLIM". In: Haux, R. (ed.) *Expert Systems in Statistics.* Stuttgart: Fischer, 1986, 155-177
[12] *Nesstar*. See www.nesstar.com
[13] Carlson, D.A. et al. *hyperModel Workbench*. See www.xmlmodeling.com
[14] Whittaker, J. *Graphical Models in Applied Multivariate Statistics* (1990). Wiley.
[15] *MathML*. See www.w3.org/Math
[16] Design Science. *Math Player – display MathML in a browser*. See www.dessci.com/en/products/mathplayer
[17] Altova. See www.altova.com
[18] *WinBUGS*. See www.mrc-bsu.cam.ac.uk/bugs
[19] Centre for Multilevel Modelling. *MLWin*. See www.mlwin.com
[20] PMML, "*The Predictive Model Markup Language*". See sourceforge.net/projects/pmml
[21] Westlake, A. *Proposals for Metadata for Generic Support of Statistical Modelling in Statistical Databases*. Deliverable 3.1 from the Opus Project, www.opus-project.org
[22] Westlake, A. & Krishnan, R. *Implementation Report on Using Information from Statistical Models*. Deliverable 6.2 from the Opus Project, www.opus-project.org
[23] *The R project for Statistical Computing*. See www.r-project.org