



OPUS



Optimising the use of Partial information in Urban and regional Systems

Project IST-2001-32471

WP7: Software test plan and test results

Title :	Software Components	
Creator (Author):	Charles Lindveld Diego Santapaola	Imperial College London PTV
Contributor :	Miles Logie	Imperial College London & Min- nerva
Identifier :	Deliverable D7.3of project IST-2001-32471	
Status :	Final	
Type :	Software	
Version :	1.0 – Final Version	
Date.Created :	18 April 2006	
Date.Modified :	18 April 2006	
Date.Next version due :	30 April 2006	
Submission Date :		
Subject.Category		
Subject.Keyword	Estimation software	
Source		
Relation	This header section draws on the e-GMS structure for document metadata 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

1. System Architecture and Components	4
1.1 Introduction.....	4
1.2 Software modules and system Architecture.....	4
1.3 Problem partitioning	5
1.3.1 Estimating trip generation.....	6
1.3.2 Estimating the the O-D matrices on basis of surveys	7
1.3.3 Estimation of the O-D matrix using counts.....	7
2. Software Components to be tested	9
2.1 Components to be tested.....	9
2.2 Not tested	9
3. Test plan for the software components to be tested	10
3.1 Test plan for the “R” glue language functionality.....	10
3.1.1 SQL Database access.....	10
3.1.2 Access to VISUM through the COM interface	10
3.1.3 Running WINBUGS from under R and reading the results	11
3.1.4 Running general external programs from R	11
3.2 Test plan for the VISUM generation of assignment matrix.....	11
3.3 Test plan for the WINBUGS scripts	12
3.4 Test plan for the TEBALDI sampler	12
3.5 Test plan for the integrated system.....	13
4. Testing and test results	14
4.1 Testing the “R” glue language functionality	14
4.1.1 SQL Database access.....	14
4.1.2 Access to VISUM through the COM interface	14
4.1.3 Running WINBUGS from under R and reading the results	15
4.1.4 Running general external programs from R	15
4.2 Testing the VISUM generation of an assignment matrix.....	15
4.3 Testing the WINBUGS scripts.....	15
4.4 Testing the TEBALDI sampler.....	16
4.5 Testing the integrated system	16

5. Changes in the software specification and re-testing	18
5.1 Changes due to test results for the “R” glue language functionality	18
5.2 Changes due to test results for the WINBUGS scripts.....	18
5.3 Changes due to test results for the Tebaldi sampler	18
6. Conclusions	21
6.1 General observations	21
6.2 Conclusions	21
7. References	22
7.1 OPUS Deliverable Documents	22
7.2 Other Documents	22

1. SYSTEM ARCHITECTURE AND COMPONENTS

1.1 Introduction

This deliverable D7.3 describes the test plan and the unit testing of the individual software components developed, both the estimation software as listed in D7.2 and certain functionality of the “glue” language “R”.

1.2 Software modules and system Architecture

The system architecture applying to much of the OPUS methodology has been defined in deliverable documents [D7.2], but we will briefly reproduce that description here.

The architecture may be represented as shown in Figure 1.1 below, which shows the main elements of the system.

Part of the system is provided by standard software systems, namely a relational database and the Visum transport modelling package. Besides these elements, the key software components are provided by the:

- R scripts that link and control the operation of the various components
- estimation software (including several MCMC samplers and one Maximum Likelihood estimator)
- data conversion and interface utilities, notably for use in passing information to and from Visum.

The software code, written variously in R, C++, and Visual Basic, is stored on the OPUS web site with links provided in the following section.

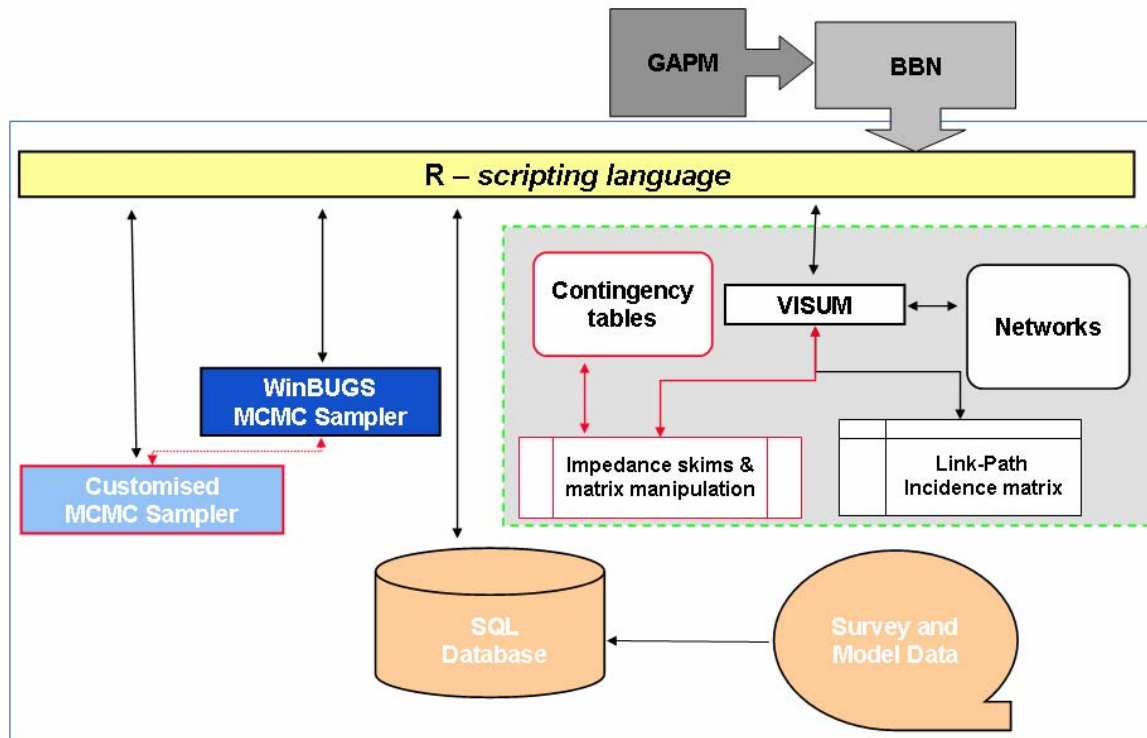


Figure 1.1 Overview of System Architecture

1.3 Problem partitioning

In principle, the estimation problem encompasses the entire model system as shown in Figure 2., but for practical reasons it has been partitioned into the three “boxes” (I), (II), and (III) shown.

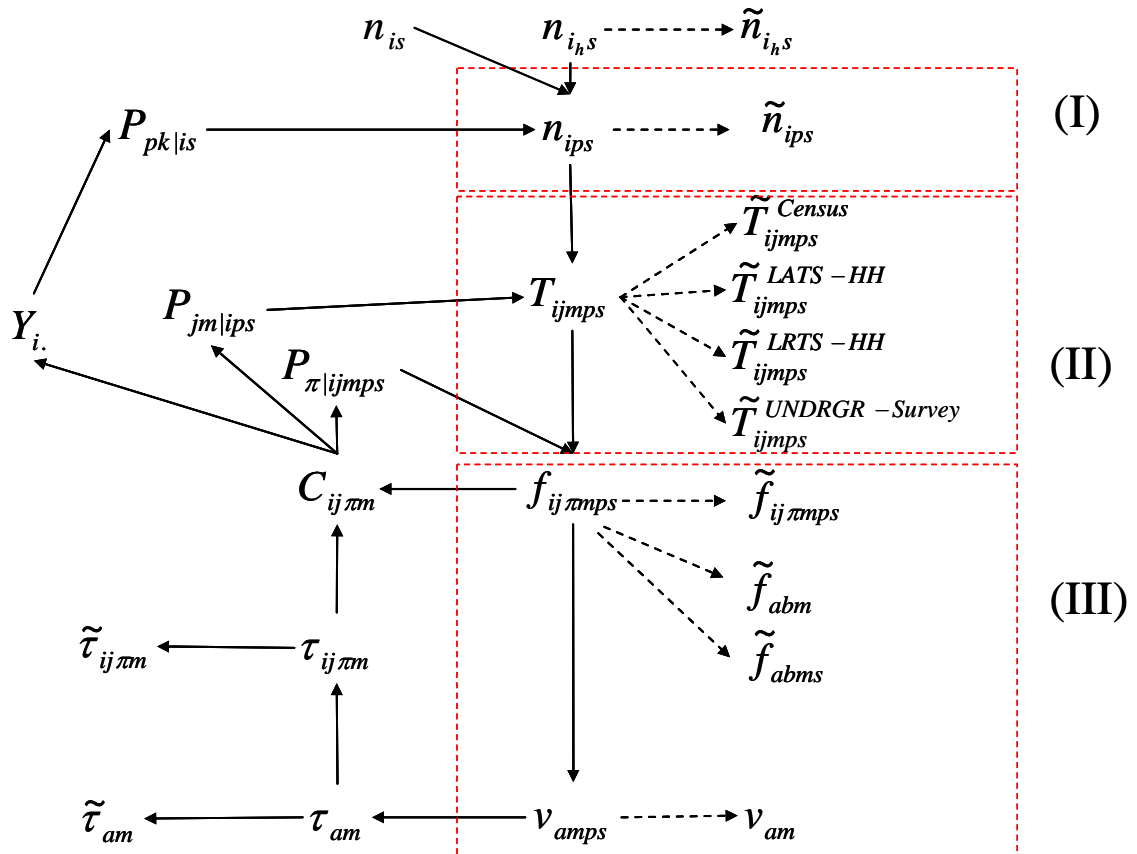


Figure 2: The system and its partitioning

1.3.1 Estimating trip generation

The generation of trips by travel purpose and by traveller segment is addressed in box (I). The BBN for this problem is shown in Figure 3.

The prior distributions are that of n_{is} , the non home-based trips and n_{ihs} the home-based trips. Modelled parameters are coming in as split proportions $P_{pk|is}$ which could either be stochastic or deterministic variables. The observations are trip production totals from the Census and/or the LATS household survey.

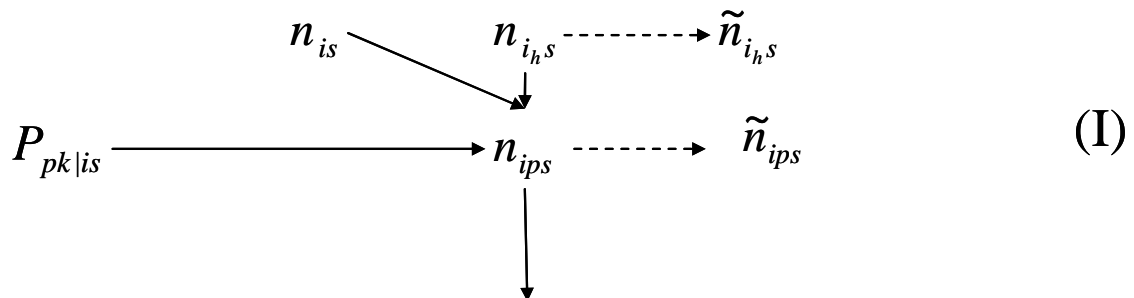


Figure 3: Estimating trip generation

If we assume that trip productions in neighbouring geographic zones are independent, the number of free parameters reduces to one parameter per socio-economic segment

s, with repeated estimations per zone. As the number of free parameters are relatively low, this model could be implemented in BUGS.

1.3.2 Estimating the the O-D matrices on basis of surveys

The trip distribution and modal split are estimated simultaneously on basis of surveys: primarily the LATS household survey and the Census survey, but including the LRTS and the underground surveys.

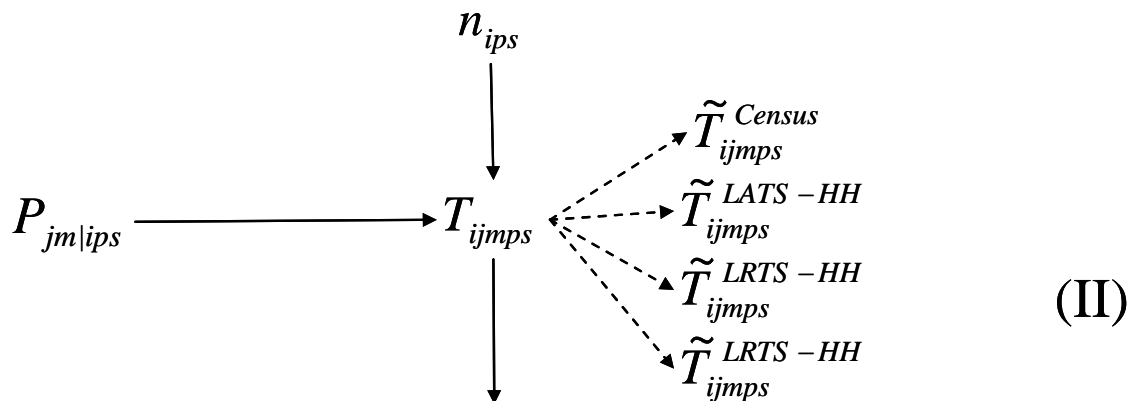


Figure 4: Estimating trip table breakdown mode, purpose, segment

Since the unknown is basically a set of O-D matrices, or a single multi-dimensional table containing hundreds of free parameters, we propose to use the methodology proposed by Dobra (see chapter **Error! Reference source not found.**) for this part.

The number of zones will be an issue, and will be a strong limiting factor. We feel however that this problem could be approached through a hierarchical approach, in which tables are estimated first in a coarse zoning system, and later on refined.

1.3.3 Estimation of the O-D matrix using counts

The estimation problem where the size of the data is likely to be most critical is the third part, shown in Figure 5,. This problem would correspond to O-D matrix estimation done in a statistical sense, for which a method has been proposed by Tebaldi and West.

In this problem too, computational burden would be critically dependent on the number of travel zones. This is not a problem in-principle, but one of practicality.

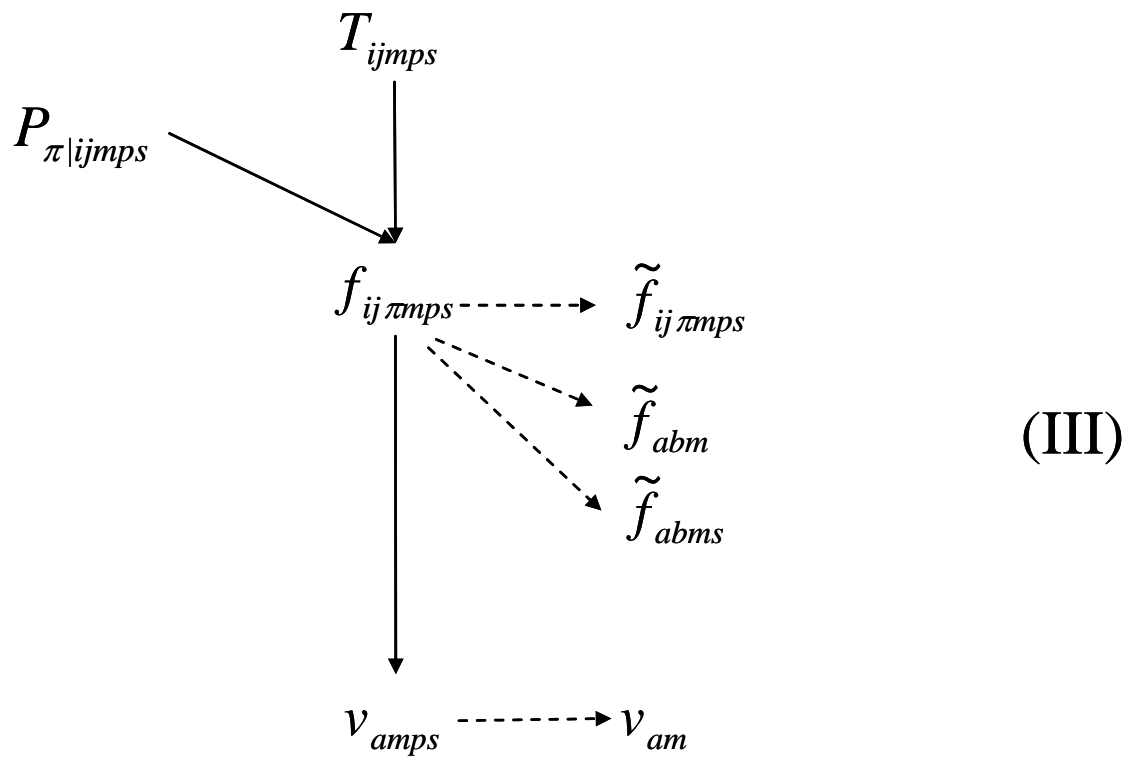


Figure 5: Estimating trip tables conditional on counts

2. SOFTWARE COMPONENTS TO BE TESTED

2.1 Components to be tested

From the system diagrams Figure 1.1 and Figure 2 a number of relationships emerge that are amenable to testing:

- R “glue” language functionality
 - SQL Database access
 - Access to VISUM through the COM interface
 - Running WINBUGS and reading the results
 - Running general external programs from R
- VISUM transport network software
 - Extraction of the assignment matrix
- WINBUGS
 - Testing of WINBUGS scripts
- Tebaldi sampler

2.2 Not tested

The core functionality of commonly or commercially available software packages (such as C-compilers, MS Visual Basic, R, WINBUGS, MAP-INFO, VISUM, MS Access etc.) was not tested on grounds that the packages were selected on basis of the experience that they work.

3. TEST PLAN FOR THE SOFTWARE COMPONENTS TO BE TESTED

3.1 Test plan for the “R” glue language functionality

3.1.1 SQL Database access

The objective of the test was to verify that the R language could (reliably) access SQL databases. This comprises both MS-Access under Windows and the MySQL database on a Linux server.

3.1.1.1 Test

The test consisted of opening the database for the London case study, reading from it, and writing a simple result back

3.1.1.2 Test criteria

The criteria of the test are:

1. Can the database be opened and accessed?
2. Does R read the contents correctly (check against database content as reported by the database itself)
3. After R writes the results, are they really in the database and are they correct (check the changed values that R tries to write against database content as reported by the database)?

3.1.2 Access to VISUM through the COM interface

3.1.2.1 Test

The test consisted of writing an R script that instructs VISUM to open the Innsbruck test network, do a traffic assignment, extract the assignment matrix, and write it to file.

3.1.2.2 Test criteria

The criteria of the test are:

1. Can R access the VISUM COM interface?
2. Can R make VISUM to read the appropriate network?
3. Can R mark a selected list of links as “monitored”
4. Can R make VISUM carry out the assignment?
5. Is the assignment matrix written correctly to file?

3.1.3 Running WINBUGS from under R and reading the results

3.1.3.1 Test

The test consisted of writing an R script that runs a WINBUGS script and reads the results. Use was made of the “R2WinBUGS” package that is available as Open Source.

3.1.3.2 Test criteria

The criteria of the test are:

1. Can R make WINBUGS to read the appropriate files and run?
2. Are the results returned correctly to R

3.1.4 Running general external programs from R

3.1.4.1 Test

The test consisted of verifying that an R script can run a general external program and regain control after the external program finishes.

3.1.4.2 Test criteria

The criteria of the test are:

1. Can R run general external programs?
2. Does R regain control afterwards?

3.2 Test plan for the VISUM generation of assignment matrix

3.2.1.1 Test

The test was complicated by the fact that the assignment matrix reflects the traffic assignment per se, which is a core function of VISUM. It was therefore decided to omit the assignment results per se from the test, and to limit the test to see if VISUM correctly writes out the assignment matrix corresponding to the calculated traffic assignment. It was decided to do this by taking a sample of monitored links and checking if the assigned link flow proportions for a sample of OD relationships were output correctly.

The test was carried out once on a test network that allowed for easy checking, and once on the Zurich network.

3.2.1.2 Test criterion

The criterion of the test was:

- Are the link flow proportions of the sampled links and OD flows correctly written to file

3.3 Test plan for the WINBUGS scripts

3.3.1.1 Test

The core WINBUGS functionality was not tested, but the models built in WINBUGS scripts were. This relates to the models developed for the estimation specified in BOX II (see section 1.3.2).

For the simplest model (the log-linear case): are the model estimation results correct on a test-case?

For the more complex models it was not possible to directly check the estimation results against analytical or otherwise known results, and therefore had to be checked indirectly:

- Are the results reasonable?
- Does the simulation finish in an acceptable amount of time (say 8 hours)?
- Do the estimation results improve the fit of the model compared to the initial data?

3.3.1.2 Test criteria

The criteria of the tests were:

- For the simplest model (the log-linear case): are the model estimation results correct on a known test-case?
- For the more complicated models the criteria were:
 - Are the results reasonable?
 - Does each simulation run finish in an 8-10 hours?
 - Does the estimation improve the model?

3.4 Test plan for the TEBALDI sampler

3.4.1.1 Test

The TEBALDI sampler was developed further to deal with non-integer assignment matrices, and drastically amended to be able to deal with larger-scale networks

3.4.1.2 Test criteria

The criteria of the tests were:

- Does the Tebaldi sampler return the correct answer on a known test network?
- Does the sampler finish in a reasonable amount of time (say 24 hours)?
- Does the sampler return reasonable results on a large-scale network?

3.5 Test plan for the integrated system

3.5.1.1 Test

The integrated system was to be packaged and installed at PTV, and run by PTV as an end-user.

3.5.1.2 Test criteria

The criteria of the tests were:

- Can end-users (personnel not involved in the development) install and run the system?
- Does the system return reasonable results?

4. TESTING AND TEST RESULTS

4.1 Testing the “R” glue language functionality

Due to the continuous development of R during the project period, the functionality tests for R were repeated for each version of R up from version 2.2.0 until the current version R 2.4.0.

4.1.1 SQL Database access

The objective of the test was to verify that the R language could (reliably) access SQL databases. This comprises both MS-Access under Windows and the MySQL database on a Linux server.

4.1.1.1 Test

The test consisted of opening the database for the London case study, reading from it, and writing a simple result back.

4.1.1.2 Test results: *Passed*

The tests were carried out in two cases:

- MS-Access on the same machine as R
- MySQL access on a remote server

The results were identical for both remote access and access on the same machine.

1. R was able to open and access the database in both cases? **Yes**
2. R was able to read the contents correctly? **Yes**
3. R was able to correctly write the results? **Yes**

4.1.2 Access to VISUM through the COM interface

4.1.2.1 Test

The test consisted of writing an R script that instructs VISUM to open the Innsbruck test network, do a traffic assignment, extract the assignment matrix, and write it to file.

4.1.2.2 Test results: *failed*

The criteria of the test are:

6. Can R access the VISUM COM interface? **Yes**
7. Can R make VISUM to read the appropriate network? **Yes**
8. Can R mark a selected list of links as “monitored”? **No**
9. Can R make VISUM carry out the assignment? -
10. Is the assignment matrix written correctly to file? -

4.1.3 Running WINBUGS from under R and reading the results

4.1.3.1 Test

The test consisted of writing an R script that runs a WINBUGS script and reads the results. Use was made of the “R2WinBUGS” package that is available as Open Source.

4.1.3.2 Test results: *passed*

The criteria of the test are:

3. Can R make WINBUGS to read the appropriate files and run? **yes**
4. Are the results returned correctly to R ? **yes**

4.1.4 Running general external programs from R

4.1.4.1 Test

The test consisted of verifying that an R script can run a general external program and regain control after the external program finishes.

4.1.4.2 Test results: *passed*

The criteria of the test are:

3. Can R run general external programs? **yes**
4. Does R regain control afterwards? **yes**

4.2 Testing the VISUM generation of an assignment matrix

4.2.1.1 Test

The test was carried out once on a test network that allowed for easy checking, and once on the Zurich network.

4.2.1.2 Test results: *passed*

The criterion of the test was:

- Are the link flow proportions of the sampled links and OD flows correctly written to file? **yes**

4.3 Testing the WINBUGS scripts

4.3.1.1 Test

For the simplest model (the log-linear case): are the model estimation results correct on a test-case?

For the more complex models it was not possible to directly check the estimation results against analytical or otherwise known results, and therefore had to be checked indirectly:

- Are the results reasonable?
- Does the simulation finish in an acceptable amount of time (say 8 hours)?
- Do the estimation results improve the fit of the model compared to the initial data?

4.3.1.2 Test results: *passed*

The criteria of the tests were:

- For the simplest model (the log-linear case): are the model estimation results correct on a known test-case? **yes**
- For the more complicated models the criteria were:
 - Are the results reasonable? **yes**
 - Does each simulation run finish in an 8-10 hours? **Yes, after change**
 - Does the estimation improve the model? **Yes**

4.4 Testing the TEBALDI sampler

4.4.1.1 Test

The TEBALDI sampler was developed further to deal with non-integer assignment matrices, and drastically amended to be able to deal with larger-scale networks

4.4.1.2 Test results

The criteria of the tests were:

- Does the Tebaldi sampler return the correct answer on a known test network? **yes**
- Does the sampler finish in a reasonable amount of time (say 24 hours)? **No, the simulation on a network the size of Zurich takes 8 hours for 40 iterations.**
- Does the sampler return reasonable results on a large-scale network? **Yes**

4.5 Testing the integrated system

4.5.1.1 Test

The integrated system was to be packaged and installed at PTV, and run by PTV as an end-user.

4.5.1.2 Test results

The criteria of the tests were:

- Can end-users (personnel not involved in the development) install the system? **Yes, but substantial assistance required from developers**
- Can end-users run the system? **Not tested because only an early development version of the sampler was installed**

- Does the system return reasonable results? **Not tested by end-users. Developers tested the individual components and presented them in the case studies.**

5. CHANGES IN THE SOFTWARE SPECIFICATION AND RE-TESTING

5.1 Changes due to test results for the “R” glue language functionality

Due to the failure of “R” to work correctly with VISUM through the COM interface a workaround was necessary. The workaround consists of a separate piece of software, written in Visual Basic, that carries out the task originally planned to be done in R (which is to access VISUM through its command-line interface, has VISUM flag an externally specified list of links as “monitored”, run the assignment, and write the assignment matrix to file).

On further testing it turned out that this function works, but was far too slow to handle real-size networks (such as the Zurich network) due to the speed limitations of the COM interface.

In response, PTV made available a special function in VISUM which would, in a single function call, write the assignment matrix for an assignment to file. This function can be accessed through the COM interface.

5.2 Changes due to test results for the WINBUGS scripts

The WINBUGS scripts that modelled multiple data sources with differing levels of accuracy, measurement error, and a distribution around the a-priori O-D matrix (making it a genuine prior in the statistical sense of the word) revealed numerous problems with:

- the input data,
- and the WINBUGS sampler crashing in an attempt to estimate the model.

The most frequently observed error was that WINBUGS, in the course of its calculations, could not construct the convex hull that it was expecting to.

The problem was in part due to the fact that a multiplicative model was used, which is not robust when the prior specifies a zero value for a cell whereas the data specifies a non-zero value. This led to an additional data-cleaning step.

The sampler would crash for all of the models that stipulate error distributions on the observations, the model, and the prior that don't have compact support. The models were changed to limit the error distributions to compact support. The model also showed sensitivity to error distributions with large variances; as a result the variance was limited to about 15%-20% of the expected values.

5.3 Changes due to test results for the Tebaldi sampler

The Tebaldi sampler presented a formidable challenge in three respects:

- The Tebaldi sampler runs quickly and well on small-scale networks, but requires large amounts of time for realistic networks. Getting the sampler to run in a workable timeframe (i.e. a week) involved major changes.
 - The computations of the original Tebaldi sampler turned out to be numerically unstable to such a degree that the original sampler had to be

revised. The crux of the matter is that the assignment matrix as produced for realistic networks is extremely ill-conditioned (condition numbers exceeding $1.0E20$ were observed). This forced the first rewrite. Specifically a subset of the columns of the assignment matrix had to be determined that contained a square matrix with condition number not exceeding a set value (we choose a condition number of 100)

- After that it turned out that some observations were dependent, leading to the problem of picking independent of the assignment matrix to form the invertible part of the A matrix. This forced the second rewrite.
- Then it turned out that runtimes for the sampler were excessive. Upon analysis it turned out that about half of the O-D pairs affected by the A matrix had negligible assignment fractions, and could therefore be left out of the calculations. In addition the random number generators, and the functions to draw from the Poisson and Gamma distributions were replaced by optimised ones, and all matrix-vector and matrix-matrix operations were coded in terms of BLAS primitives. This constituted the third rewrite.
- As the original code for the Tebaldi sampler ran quickly on the examples provided for it (including the Innsbruck network), it was only realised gradually how much the original sampler had to be rewritten as larger cases were tried. The changes had to be introduced incrementally instead of one at a time, which complicated the coding and led to loss of time. At the end of the project some 29 versions of the sampler were produced, and only the last of these proved reasonably acceptable.
- In order to be able to write a module that could be called directly from under R It was decided to use C as a programming language. The choice of the programming language (C) proved both a blessing and a curse.
 - The advantages are that
 - subroutine libraries for numerically sound and computationally fast subroutines could be found,
 - it was possible to structure the sampler so that it may be paralised
 - using standard ANSI C it was possible to write the sampler in such a way that it could be compiled both under Windows and under Linux and Unix (the parallel computers at Imperial College run under Unix).
 - The disadvantages are that
 - the various subroutine libraries (BLAS, ATLAS, LAPACK, libraries of distributions) used were originally written in Fortran, and are awkward to call from C
 - interfacing with the libraries induced much programming work in which memory management errors crept in. These errors caused large delays, and were only solved after using specialist debugging software (i.e. IBM's Rational Purifier)

- the C language has no protection against array-bound overflows, the use of un-initialised memory, and memory overwrites which leads to hard-to-find bugs.

At the end of the project, the original idea of calling the sampler directly from R was not realised as modifying the sampler to scale up (and debugging the modifications) took up all available time. Instead there is an additional layer of indirection, in which R sets up the control-file and the data for the sampler, runs it as an external self-standing program, and reads in the results for processing and reporting.

With hindsight the decision to use ANSI C as a programming language was a significant mistake. It would have been a reasonable choice if the sampler only had to be translated, but it turned out to be disastrous when the sampler had to be continuously rewritten.

6. CONCLUSIONS

6.1 General observations

The original plan of producing software in a linear fashion once the specifications were clear in outline by and large had to be abandoned in the light of the difficulties with the computational core routines.

The more “standard” data processing work such as the glue language and database access by and large worked as foreseen, and could be tested in a timely fashion.

The development of the computational core routines turned out to be an iterative process, for which sufficient time had been allocated in the original project plan, which unfortunately suffered external delay of a year. In order to meet the formal deadline, project work on the development of the core routines was rushed, resulting in a substantial net increase in development time. It can only be a small solace to note that the total development time is more or less in agreement with the original project estimate but unfortunately did not meet the revised project schedule.

6.2 Conclusions

The components of the software system have been tested and have been found to function. System integration has been carried out to the extent that the system can be run effectively for the case studies in London and Zurich. However the operation of the system still requires some manual intervention. Refinements to the user interface are an obvious next step, but were not within the scope of the OPUS project

7. REFERENCES

7.1 OPUS Deliverable Documents

- [D4.2] Logie, M and Lindveld, C (2005), Transport Domain Method Specification Report, OPUS Project Report Deliverable 4.2
- [D4.2 S] Logie, M and Lindveld, C (2005), Transport Domain Method Specification Supplementary Report, OPUS Project Report Deliverable 4.2 Supplement
- [D6.2] Westlake, A and Krishnan, R (2006), Generic Structures and Functionality for Support of Statistical Models in Statistical Databases – Implementation Report on Using Information from Statistical Models, OPUS Project Report, Deliverable 6.2.
- [D8.2] Logie, M (2006), Report of London Case Study, OPUS Project Report, Deliverable 8.2
- [D9.2] Chalasani, V.S., Axhausen, K.W., Report on Updated Database on Transport Information for Zurich, OPUS Project Report, Deliverable 9.2
- [D10.3] Cornelis, E, Ponti, C and Logie, M (2006), Feasibility Studies: Belgium and Regione Lombardia, OPUS Project Report, Deliverable 10.3
- [D11.2] Martuzzi, M, Mitis, F, Bennet, J *et al* (2006), Feasibility Study Design (Health), OPUS Project Report Deliverable 11.2
- [D11.3] Martuzzi, M, Mitis, F, Bennet, J *et al* (2006), Feasibility Study Results (Health), OPUS Project Report Deliverable 11.3

7.2 Other Documents

- [Dominici00] Dominici, F., (2000) Combining Contingency Tables with Missing Dimensions. *Biometrics* 56, pp. 546 – 553
- [Tebaldi96] Tebaldi, C., West, M. (1996) Bayesian Inference on Network Traffic Using Link Count Data. Internal Report ISDS, Duke University