22nd Australasian Transport Research Forum
Sydney, September 1998

Computer modelling in transport planning: an investigation of the current issues and the potential of component object model (COM) approach

Tu Ton
Institute of Transport Studies
The University of Sydney

Abstract:

The past twenty years has been the period of rapid technological change and there is no question that computing technology has been enthusiastically adopted by transport professionals and has brought about new concepts such as Intelligent Transport Systems (ITS). However, this has not necessarily shown itself in software quality growth – especially from a transport modelling viewpoint. There has been a gap between the conceptualisation of a model and its implementation since 1960s. Transport software developers have been equipped with more powerful computer hardware and software than before. However, the increasing complexity of land use/transport and environment system together with the constraint of limited resources allocated to the development task of transport modelling are putting pressure on the transport modelling task. Among fast growing computing technologies in recent years, object-oriented concept and component object model (COM) have continued to demonstrate their capability in laying fundamental layers for significant computing framework such as Microsoft Windows System. The use of such tools in implementing models of land use/transport and environment system opens up an opportunity to consider the extent to which a component object model framework can be constructed, re-used and customised to flexibly represent more complex models in a more resource effective way. This paper investigates the capability of the object-oriented concept as a candidate for structuring a COM framework for implementing models of land use, transport and environmental impact system.

Contact author:

Dr Tu T Ton
Senior Research Fellow
Institute of Transport Studies (C37)
The University of Sydney NSW 2006

Telephone: (02) 9351 0072 Fax: (02) 9351 0088
Email: tut@its.usyd.edu.au

Introduction

The past twenty years has been the period of rapid technological change and there is no question that computing technology has been enthusiastically adopted by transport professional and has brought about new concepts such as Intelligent Transport Systems (ITS). However, this has not necessarily shown itself in software quality growth - especially from a transport modelling viewpoint. There has been a gap between the conceptualisation of a model and its implementation since 1960s. As noted by Harris (1968, pp.4-5) and Boyce (et.al., 1970, p.25), in their review of major land-use and transportation studies that were carried out in the United States in the 1960s, they found that:

"Considerable difficulties and delays were experienced in operationalising the theoretical models, and these were compounded by appreciable data management problems. Consequently, the number of alternatives prepared and the differences among them were severely limited and work schedule were drastically revised, often resulting in a loss of credibility for the planning effort".

These quotes depict the difficulties of implementing theoretical models in the 1960s No further reports or publications on specific problem areas of the implementation task are released to public. However, literature review from computer science field indicates that the programming problems are in general due to the inherent complexity of software. Booch (1991) found that this inherent complexity derives from four elements: the complexity of the problem domain, the difficulty of managing the developmental process, the flexibility possible through software, and the problem of characterising the behaviour of discrete systems.

The question is what about the 1970s, 1980s and 1990s? Have they got the same programming problems of the 1960s? Transport software developers have been equipped with more powerful computer hardware and software than before. However, the increasing complexity of land-use/transport and environment system together with the constraint of limited resources allocated to the development task of transport modelling are putting pressure on the transport modelling task. Among fast growing computing technologies in recent years, object-oriented concept and component object model (COM) have continued to demonstrate their capability in laying fundamental layers for significant computing framework such as Microsoft Windows System The use of such tools in implementing models of land-use/transport and environment system opens up an opportunity to consider the extent to which a component object model framework can be constructed, re-used and customised to flexibly represent more complex models in a more resource effective way.

This paper investigates the capability of the object-oriented concept as a candidate for structuring a COM framework for implementing models of land-use, transport and environmental impact system. The paper is structured around six sections. Next section reviews current issues and the potential of COM approach. Section three describes the development process of a COM framework for supporting the development of transport models. Section four presents different methods of using COM objects to develop new applications The flexibility and reusability features of the COM framework in building five models for five selected case studies are reported in Section five. The paper concludes with a summary of main findings in terms of the feasibility and practicality of the component object model framework together with direction for future research.

Current Issues and the Potential of COM Approach

Literature review and evaluation of commercially available packages for transport reveal a trend in developing transport software from *problem-oriented approach* towards *problem solving* or *tool-box approach*. The main difference between the two strategies is in the requirements of the skill (ie. computing and transport) of the users of the transport packages. The problem solving approach requires users with strong background in both computing and transport. It provides the user with a software tool set so that transport modellers can use to construct a model to their requirements. The design of EMME/2, a transport software package developed by INRO (1994), partly reflects this strategy.

Matrix tool is an important tool provided by EMME/2. This matrix tool was specifically designed for EMME/2 users to construct a number of matrix-based models in transport planning. The advantage of the problem-solving approach is the flexibility and reusability in building and testing the models. These features could explain why EMME/2 has received a considerable amount of attention from transport community worldwide soon after it was released. Flexibility is in the providing its user a matrix manipulation tool and letting the user builds the first three-steps of the four-step transport modelling process (eg. trip generation, trip distribution and mode choice). Reusability is in terms of reusing the scrip file written in EMME/2's macro language.

However, from the viewpoint of software construction, EMME/2 still has some drawbacks. First, EMME/2 cannot be easily expanded to include new computational tools; matrix is the only reusable tool available in EMME/2. Secondly, the dependence on EMME/2 environment (user interface and library setting) make it inefficient and uneconomical to integrate computerised models developed from EMME/2 with other models or systems. Thirdly, the EMME/2 macro itself is not designed as a programming language or even a problem-oriented language (ie. its vocabulary for matrix manipulation functions can not be customised to any specific model).

The problem-oriented approach is more user-friendly than the problem-solving approach in terms of providing the required computational tools and a fixed user interface describing a specific problem such as the four-step transport models. Therefore, this approach requires the user with a moderate background in both computing and transport. The design of most commercially available packages such as TRIPS, TRANPLAN, QRS 2, TMODEL, TRANSTEP reflects this strategy. The advantage in adopting this strategy is, through an user interface, the models can be represented as a problem-oriented system. The disadvantage in using this strategy is the lack of flexibility that the problem-solving approach has - for example, in terms of constructing a new model and/or employing a new alternative to an existing modelling function.

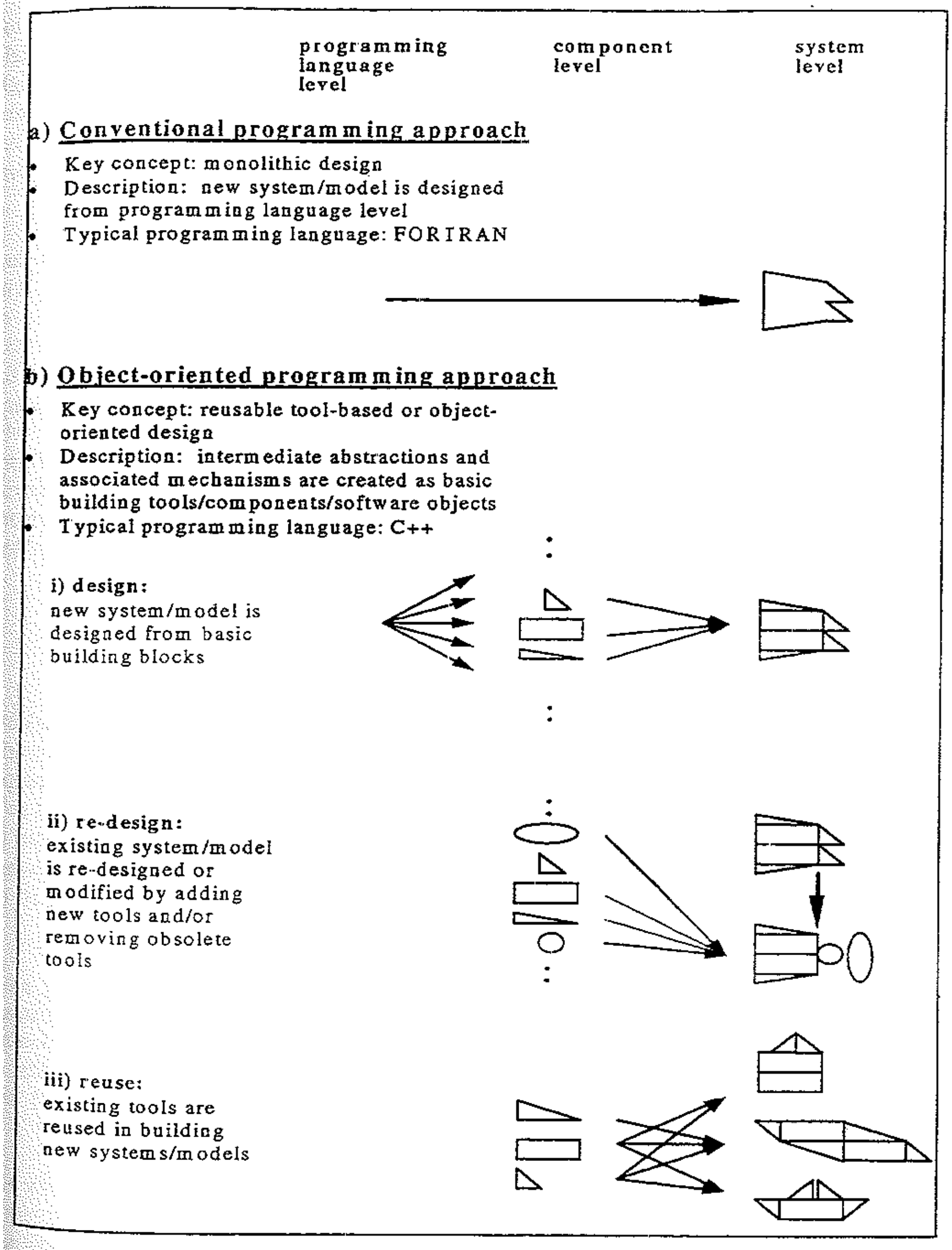
Each strategy has its own advantage and disadvantage. Even there exist some particular software user groups to share the experience on using particular software. Users of most of current transport packages are still relying on their developers and it is up to the software developer to provide enhanced solution to an existing problem. Transport software packages are still developed as a complete aggregated tool. Users do not have choices of obtaining more disaggregate tools that making up an aggregated tool. Linear algebra (ie. matrix based), statistical functions and estimation, network, four-step transport models, choice-based models tool are example of disaggregate tools. This flexible feature demands for a reusable tool-based approach in the development of transport software. More generic tools such as matrix and statistical function tools can be stocked to support problem-solving strategy. The generic tools can then be reused and customised to support problem-oriented strategy. The development of ODMatrix (origin destination matrix) tool is an example. ODMatrix tool can be developed by reusing Matrix tool and more specific functions such as functions for calculating trip length distributions, etc. are the only new code that are required to add. Software reuse is maintained in this development. ODMatrix tool can still have the functionality of a normal numerical matrix but its own additional functions are specific to transport users.

This reusable tool-based or component object model approach is similar to the tool-based mathematical approach which has been part of the transport modellers' knowledge (eg. linear algebra, statistical techniques, etc.) in the conceptual modelling task. The consequence of using the tool-based programming approach is that the cost of programming would be reduced due to the minimisation of the duplicated code for most frequently used models or abstractions and mechanism.

The COM approach can be best explained using the illustration in Figure 1 in contrasting two methods of design: "monolithic design" and the "reusable tool-based design" representing the conventional approach and the COM based approach, respectively.

The monolithic design as being used by the conventional procedural approach is oriented towards the original specification of the problem (Figure 1a). The monolithic design is not flexible in terms of supporting the redesign and/or reuse existing software system. This is because the conventional approach cannot implement the component as a complete software module encapsulating both data and the associated functions.

In contrast, the COM approach supports the reusable tool-based design (Figure 1b). In other words, from the original specification of the problem, the system is designed to consist of a number of components. These components once implemented as prefabricated software tools can then be used as basic building blocks for developing the required model (Figure 1b-i). The resulting COM system would function similarly to that of conventional approach. However, it offers three additional features. Firstly, the resulting structure of the COM system is more flexible than that of conventional system. In consequence, the COM system would be easier for supporting user interface and maintenance. As shown in Figure 1b-i, the resulting COM system is made up of six components which are constructed from the three separate tools/objects (ie. long triangle, short triangle and rectangle tools). In contrast to the monolithic design system



a) Conventional programming approach

- Key concept: monolithic design
- Description: new system/model is designed from programming language level
- Typical programming language: FORTRAN

b) Object-oriented programming approach

- Key concept: reusable tool-based or object-oriented design
- Description: intermediate abstractions and associated mechanisms are created as basic building tools/components/software objects
- Typical programming language: C++

i) design:
new system/model is designed from basic building blocks

ii) re-design:
existing system/model is re-designed or modified by adding new tools and/or removing obsolete tools

iii) reuse:
existing tools are reused in building new systems/models

Figure 1: The COM Approach versus Conventional Approach

by the conventional programming approach centralises the user interface through a top level function, the user interface support in COM system is delegated to the right component object in the system. For example, if a user queries on the first rectangle object in Figure 1b-i, this particular object can handle itself. In terms of system maintenance, as shown in Figure 1b-ii, the COM system can be redesigned by just replacing the unwanted components (ie. two triangle objects) with new ones (one oval object and one circle object).

Secondly, stricter software quality control can be achieved with COM approach than that of conventional approach. It is because defective components in COM (ie. with possible errors or bugs) are always easier to be detected for debugging, replacement or modification. Thirdly, the prefabricated software tools are the saving of the programming effort invested in the COM approach. These tools can be reused for building completely new systems/models (Figure 1b-iii) in a more economical and efficient way. To support the development of new models, a number of basic well-established abstractions and mechanisms needs to be implemented as basic software units so that new models can be built from these basic building blocks (eg. Matrix, Minimum Path, Gravity Model, etc.). Next section describes how a COM framework can be established with a number of basic building blocks identified and generalised from transport problem domain.

Developing A COM Framework for Transport Modelling

The proposed framework is aimed at providing the user with a number of basic building blocks for use as is or can be reused for customisation or as part of a larger framework. The framework is built on object-oriented (OO) technology and software reuse concept. Object-oriented concept is used to represent a system or model and the associated sub-systems from transport and related domains. The software reuse concept helps to organise and relate objects in an efficient way to minimise duplicated representation. Commonly used models can then be abstracted and prefabricated as software objects or tools to support the development of more complex objects.

Figure 2 describes a development process of a COM framework for transport area. Starting with the analysis phase, the major tasks are to identify basic and commonly used transport models; to identify basic and commonly used models from supporting domains (eg. mathematics, statistics, etc.); and to identify the current trends in technology (eg. integrating demand with geographic information systems (GIS) and knowledge-based expert system (KBES)).

In order to provide the input to the analysis phase, five case study applications are selected based on Taylor's (1991) definition of five levels of planning: sketch planning, strategic planning, local planning, road element analysis and traffic impacts. They are journey-to-work census data analysis, four-step transport models, traffic noise modelling, site impact traffic analysis and traffic count analysis. These models represent the major steps in the transport planning and traffic impact assessment process.

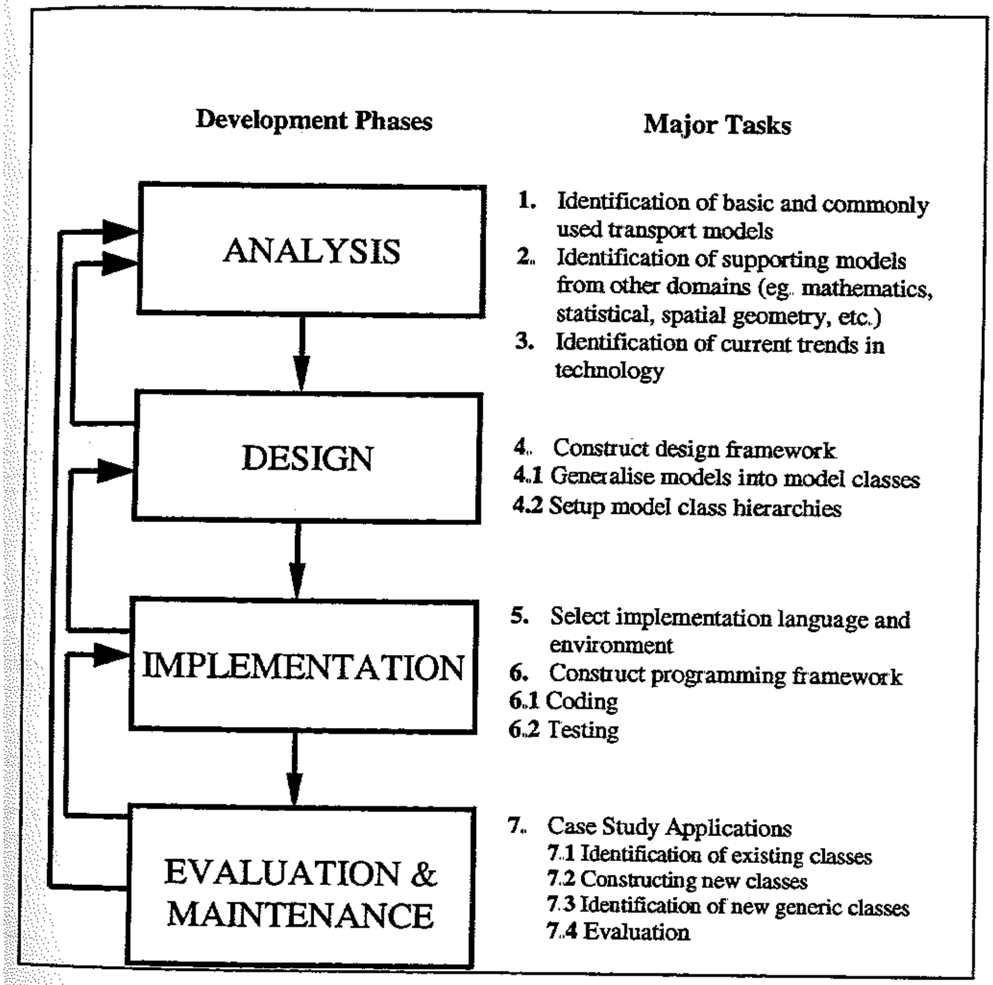

Development Phases
Major Tasks
ANALYSIS
DESIGN
IMPLEMENTATION
EVALUATION & MAINTENANCE
1. Identification of basic and commonly used transport models
2. Identification of supporting models from other domains (eg. mathematics, statistical, spatial geometry, etc.)
3. Identification of current trends in technology
4. Construct design framework
4.1 Generalise models into model classes
4.2 Setup model class hierarchies
5. Select implementation language and environment
6. Construct programming framework
6.1 Coding
6.2 Testing
7. Case Study Applications
7.1 Identification of existing classes
7.2 Constructing new classes
7.3 Identification of new generic classes
7.4 Evaluation

Figure 2: Development Process of A COM Framework for Transport Modelling

Figure 3 depicts the interrelationship between the five case study applications in terms of the input and output data flow process. Starting with case study number 1 (journey-to-work analysis), the census journey-to-work data (ie. number of trips made between origin and destination) and travel cost data (ie. distance or travel time or cost to travel between origin and destination) are used as input to the journey-to-work analysis application. The journey-to-work analysis system can be used as a fact finding facility about travel behaviour of a particular study area with further interrogation. There is also a need to integrate the software application developed for this case study with GIS technology to maximise the graphic and mapping presentation in a relational database context. The main output from this case study, which includes origin-destination trip tables and mean trip lengths, can be used to calibrate the trip distribution models developed in case study number 2.

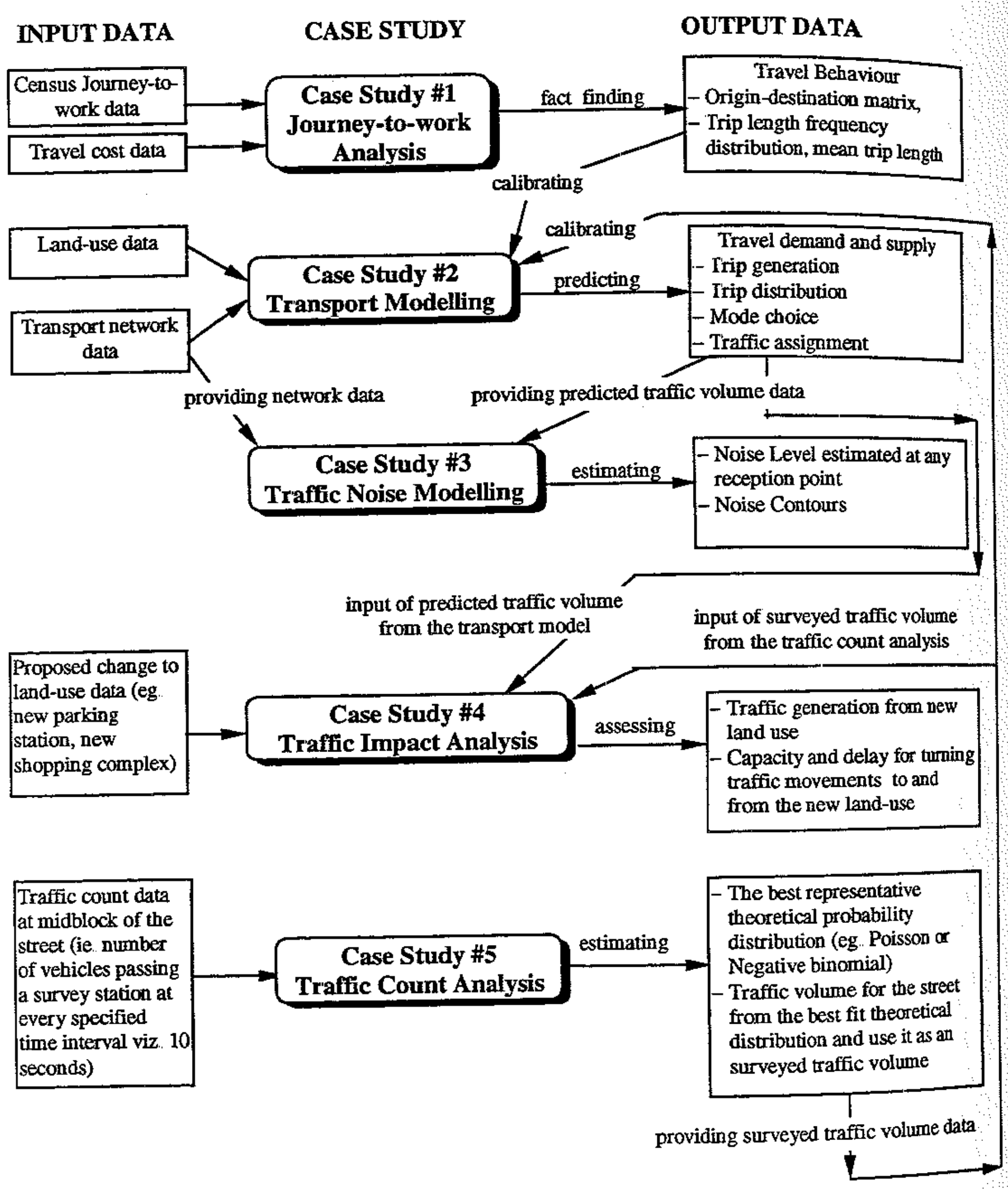

INPUT DATA
CASE STUDY
OUTPUT DATA
Census Journey-to-work data
Travel cost data
Case Study #1 Journey-to-work Analysis
fact finding
Travel Behaviour
– Origin-destination matrix,
– Trip length frequency distribution, mean trip length
calibrating
calibrating
Land-use data
Transport network data
Case Study #2 Transport Modelling
predicting
Travel demand and supply
– Trip generation
– Trip distribution
– Mode choice
– Traffic assignment
providing network data
providing predicted traffic volume data
Case Study #3 Traffic Noise Modelling
estimating
– Noise Level estimated at any reception point
– Noise Contours
input of predicted traffic volume from the transport model
input of surveyed traffic volume from the traffic count analysis
Proposed change to land-use data (eg. new parking station, new shopping complex)
Case Study #4 Traffic Impact Analysis
assessing
– Traffic generation from new land use
– Capacity and delay for turning traffic movements to and from the new land-use
Traffic count data at midblock of the street (ie. number of vehicles passing a survey station at every specified time interval viz. 10 seconds)
Case Study #5 Traffic Count Analysis
estimating
– The best representative theoretical probability distribution (eg. Poisson or Negative binomial)
– Traffic volume for the street from the best fit theoretical distribution and use it as an surveyed traffic volume
providing surveyed traffic volume data

Figure 3: Interelationship Among Five Case Studies

The main function of the applications developed in case study number 2 is to implement a flexible modelling tool for the four-step transport planning model of trip generation, trip distribution, mode choice and traffic assignment. Flexibility in this case study means that different variations of each transport modelling step can be represented and compared in the same application. Land-use and transport data are the two major types of input data to the applications developed in this case study. The output from this case study is in terms of estimated traffic volume on the network. The traffic volume information on the network provide basic input data to the traffic noise modelling and site impact traffic analysis applications developed in case studies number 3 and 4, respectively.

With the transport data and the predicted traffic volumes available from case study number 2, the traffic noise application (case study number 3) can estimate the noise level at a single reception point as well as at the network level in the format of noise contours. The integration with GIS from the software application developed in this case study is also being required for supporting the mapping and plotting of noise contours and other associated data. This facility adds an extra dimension, the traffic noise impact, to the transport modelling application developed in case study number 2.

In case study number 4, the focus is on a finer level of geographic detail - traffic impacts and the assessment of road access to and from a new land-use development (eg. proposals for parking stations or new shopping complex). The main function is to estimate the trip generation from a new land-use development and estimate the traffic capacity and delay to the traffic movements when accessing to and egressing from the new development.

In the final case study, the traffic count analysis application processes raw data from a roadside traffic count, produces the observed frequency distribution, applies curve fitting from a number of given theoretical probability distributions and finally estimates the traffic volumes from the best fit distribution. The estimate of traffic volume is considered to be the surveyed traffic volume. The process can be applied to any locations on urban or rural roads. The resulting surveyed traffic volumes also provides a feed back loop for supporting the calibration procedure required in case studies number 2 and 4 and a new transport modelling process can start.

A literature review on the commercially available software packages (detailed in a separate section of each case study) has found that the software systems for every single case study were developed from different software developer. However, no one software package implements applications that would support all five case studies. Consequently, there exist problems of data and model incompatibility among those software packages making them cost ineffective to integrate them.

Therefore, in this section, the framework is constructed as a single modelling environment to develop all five case study applications with two major aims. One aim is to represent models (both theoretical and physical models) with the "object" to test if duplication effort in software development can be reduced due to the reuse of software objects. The other aim is to solve the problem of data and functions incompatability by using "object" as the unit for the transport modelling environment. Detailed descriptions of the development of the software system is in Ton (1995).

The analysis of data (both input and output) and functions required in the five selected case studies identified a number of basic models and abstractions and functions potentially used as generic supporting tools. For classification, they are grouped under eight domains. These domains are land-use, transport, traffic, mathematics, statistics, environmental impact, spatial geometry, and computing utility. It should be noted that the current scope of the framework can be expanded in terms of new domain or new models added to existing domains. The land-use, transport, traffic and environmental impact COM objects represent the main domains of COM framework as they support the evaluation of the interaction between land-use, transport in terms of traffic and environmental impacts. The mathematics, statistics, spatial geometry and computing utility domains are the four important supporting domains in terms of the use of mathematics (eg. matrix manipulations and linear algebra), statistics (eg. basic statistics, probability distributions, random number generators, etc.), spatial geometry (eg. representing x, y, z coordinates, calculates distance between noise source and noise receiver in the traffic noise model) and some utility functions (eg. iterative process control and other software support utilities).

The boundaries on a domain (the extent of each domain) are open-ended. They are defined more by the author's experience and practical considerations - such as available resources - than by formal definitions or limits. The intention is to analyse a sufficiently wide area so that related concepts are identified; these may influence the development of our understanding of the concepts central to the application. This broad area is also insurance. As transport modellers might change their minds about the requirements for a system, the broad scope of the analysis will likely have anticipated these changes and further analysis will not be needed.

Similar models identified in the analysis phase are generalised into model classes in the design phase. Related classes of objects are then organised into class hierarchies. These class hierarchies are implemented directly by the implementation phase involving the tasks of selecting implementation language and environment; and constructing programming framework.

C++ is selected as a programming language to implement the framework. The selection of a suitable programming language is not really an issue in comparing with the communication between COM tools. With the client/server technology together with the support for implementation of COM by Microsoft, COM objects can actually communicate with each other regardless these COM tools reside on the same computer or not at run time. This kind of communication opens a new era in developing and using transport software via internet. COM tools can serve as a component object in a larger framework or as a stand-alone software unit ready to be used by users. In other words, users do not necessarily have the required COM tools located on their local computer, they can be on a remote computer somewhere on the internet. Obviously, if all the required COM tools are located locally, the performance will be better in terms of compile and run time. Therefore, in this paper the COM framework is constructed and tested for running on local computers.

Case study approach is used for the evaluation and maintenance phase. This phase involves the identification of existing classes of objects for supporting the development of new classes of objects required in the case study. The evaluation of the framework support in the development of case study applications is to find out how effective has the framework been in the support of the development of these applications. The identification of new generic classes from a case study in the maintenance phase provides a feed back to the analysis phase to refine and/or expand the existing framework.

Two-way communication between different phases allows for an *incremental (iterative, evolutionary, prototyping) and adaptive* development process to be achieved. With incremental development, the existing system can be evolved into a future system when more objects/classes are added to the framework software base. With adaptive development, existing design can be further refined to respond to changes in system specifications. The inclusion of this feature in the development process supports the software flexibility, which is an important feature in software development. Next section describes the procedure for using the framework in the development of new applications.

COM Based Development Methods

It is quite flexible in using the COM framework to develop applications. COM framework supports all four methods of reuse. They are abstract reuse, instance reuse, custom reuse and source code reuse (see Figure 4).

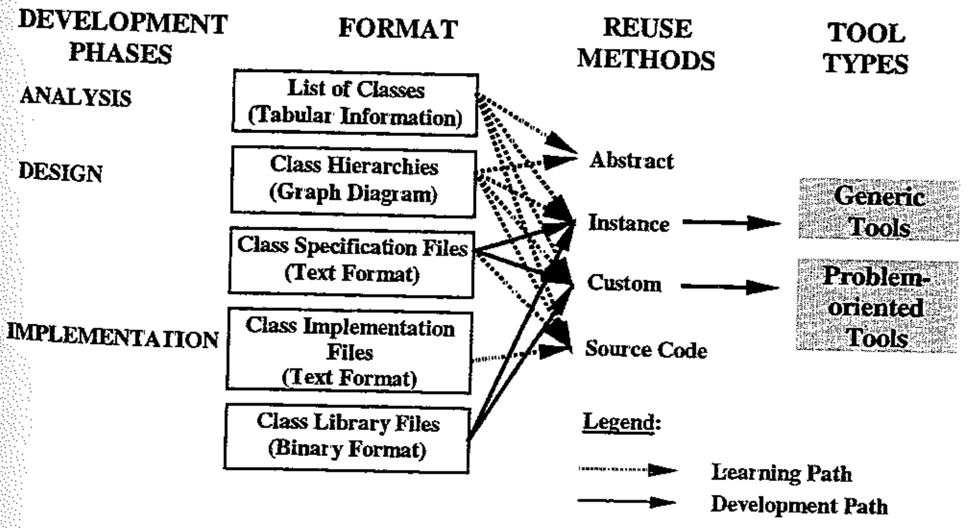

DEVELOPMENT PHASES
FORMAT
REUSE METHODS
TOOL TYPES
ANALYSIS
DESIGN
IMPLEMENTATION
List of Classes (Tabular Information)
Class Hierarchies (Graph Diagram)
Class Specification Files (Text Format)
Class Implementation Files (Text Format)
Class Library Files (Binary Format)
Abstract
Instance
Custom
Source Code
Generic Tools
Problem-oriented Tools
Legend:
Learning Path
Development Path

Figure 4: COM Based Development Methods

Users of the framework can study the structure of the framework by following the learning paths. These learning paths are indicated by dotted lines and associated with all four methods of reuse whereas the development paths indicated by continuous lines are associated with the instance reuse and custom reuse. Abstract reuse is reusing the concept and design of the framework. It is an important step for new framework users in understanding framework. In other words, it shows how the basic models from the eight domains of the framework were abstracted, organised and related to each other in an object-oriented design framework.

Having studied this type of reuse, the framework users might proceed to the other three methods of software reuse: instance reuse, custom reuse and source code reuse.

Instance reuse is the most usual form of the framework reuse. The user can use directly the available classes from the framework by creating instances of the required classes. The C++ class specification (or class interface) file provides the vocabulary (syntax) of a particular class being reused (data and functions available from the class). The C++ class library files provide the implementation method for building executable programs. In fact, the C++ class library is a packaged version of the C++ class implementation files. These files are used for implementing the functions defined in the corresponding class interface files. They are documented in Ton (1995) for supporting the final method of reuse - source code reuse.

In summary, in COM based development process, all that a user as a model developer needs to do is to write a new application source code using the vocabulary available from class interface file. A compiler is then used with a compiler run time library of interface to compile the new application source file in order to have it translated into object module (binary machine format) and linked with the class library. This is the easiest, quickest, and most economical form of reuse.

For example, in representing a minimum path algorithm, a developer can use instantly class MinPath and deliver it to end users. However, as a developer may want to add more specific functions to an existing class, for example, a user may want to represent a specific minimum path algorithm such as Floyd's algorithm (Floyd, 1990), class MinPath is reused to develop class MinPathFloyd. With this custom demand, the custom reuse is more suitable.

With custom reuse, users do not have to rewrite the code for an existing class. They are only required to write additional code which is specific for the new functionality. This feature is supported through the use of inheritance to create a new class from an existing class (incremental development). Reuse is not an either/or proposition as it is in the procedural programming style. A class in COM framework is customised with far less effort than a set of procedures in the conventional approach. A reuser can create a new class without changing the existing class by inheriting information from the existing class, overriding certain methods, and adding new behaviours that are required.

Case Study Evaluation of COM Framework

This section reports the performance of COM framework in terms of software quality measures in the development for the five case studies. Problem-orientedness, software modularity, and software reusability are used as the criteria for assessing the framework. Problem-orientedness evaluation assesses the ability of the framework in specifically representing the problem to be modelled in a specific context. Software modularity evaluates how the framework supported this feature in the development of new applications. Software reusability measures the level of reusing the framework in the development of the five case study applications.

The construction of problem-oriented systems was supported by the framework in both aspects of system modelling: structural and functional. In terms of structural representation of the theoretical models, The framework supported the construction of specialised tools for use in structuring new application. Objects such as ODMatrix, and MinPathFloyd are derived from object Matrix and object MinPath in the framework for representing origin-destination matrix and a specialised algorithm implemented by Floyd to support minimum path procedure, respectively. These specialised tools were used in structuring the five case study applications. The use of descriptive vocabulary for naming functions of the framework facilitated the functional problem-oriented representation. Functions GetAPath, GetNoiseSource, etc. are examples of problem-oriented functions.

In terms of software modularity support, the framework's objects were used as software modules. The provision of direct abstraction by the framework's objects made the software modularity support more efficient than the conventional programming. It is more efficient because an object is the only module used for representing a model whereas in the conventional programming, two different types of modules are required: data and function modules. In addition, the transport modeller has to handle the linkage between data and function modules that are not relevant to the task of modelling transport entities.

Literature review of the software reusability in computer science and engineering area found that there has been no reference or guidelines on how to quantify software reuse. Nevertheless this criterion is amenable to rigorous analysis. Two software reusability measures are proposed in the evaluation of the COM framework. They are the *total reusability* and *reusability index*.

The total reusability is defined as the total number of the framework classes being reused in the development of a particular application. As different classes, depending on how comprehensive or complexity their design are, might be significantly different in their sizes which can be measured by a number of programming lines. Therefore, total reusability can also be measured in terms of number of reused programming lines. In addition, the use of number of reused lines as an alternative unit to number of reused classes helps putting the software reuse in a format that can be comparable to the conventional programming.

The reusability index represents the percentage of software reusability. It is defined as a ratio between total reusability and total development. In the development of the five case studies, the average reusability is a useful measure in identifying the most common supporting domain in COM framework. It is defined as an average number of reused classes or average number of reused lines in a supporting domain (eg. mathematics or statistics, etc.) by all applications.

Table 1 presents these results of these software reuse measures for the five case studies. The first column of this table lists out the names of the eight domains. The last three rows of this table are the total reusability, total development and reusability index measures. The last two columns in this table are the average reusability measured in both number of reused classes and number of reused lines. The calculated values of these software reusability measures reflect the software reusability support of the supporting classes, the size and the complexity of the case studies.

Starting with case study number 1 of the journey-to-work analysis, the reusability index is measured as 87.5 percent in which seven (three mathematics classes, one statistics class and three computing utility classes) out of a total of eight classes were reused. The equivalent number of reused lines are 4195 lines out of a total development of 5879 lines (71.4 per cent). Matrix processing and manipulation; histogram setup; and look-up table were the major tools required in the application of this case study.

Reusability indices of 76.7 percent of classes and 84.1 per cent of programming lines were measured in the development of four applications representing the four steps of transport modelling in case study number 2. Mathematics, statistics, spatial geometry, transport and computing utility were the five domains providing direct support of required modelling tools (ie. the framework classes).

The spatial geometry and the environment impact are the two domains providing the support required by case study number 3 of the traffic noise modelling. Reusability indices of 75 per cent of classes and 89.9 per cent of programming lines were achieved. The land-use, mathematics and traffic domains were used in the development of application in case study number 4 of site traffic impact analysis. Reusability indices of 85.7 per cent of classes and 83.4 per cent of programming lines were measured.

The traffic count analysis application in case study number 5 requires the support of the mathematics and statistics domains. Reusability indices of 90.9 per cent of classes and 91.7 per cent of programming lines are the highest measures of software reusability amongst the five case studies.

Table 1: Estimated Software Reuse Meatures in Development of Five Case Study Applications

Supporting Domains	Case study # 1		Case study # 2		Case study # 3		Case study # 4		Case study # 5	
	Number of reused classes	Number of reused lines	Number of reused classes	Number of reused lines	Number of reused classes	Number of reused lines	Number of reused classes	Number of reused lines	Number of reused classes	Number of reused lines
Mathematics	3	3180	3	3180	2	2870	2	2870	2	2870
Statistics	1	332	2	592	-	-	-	-	8	1478
Spatial Geometry	-	-	4	808	4	808	4	808	-	-
Land-use	-	-	-	-	-	-	-	-	-	-
Transport	-	-	13	4568	-	-	-	-	-	-
Traffic	-	-	-	-	-	-	6	1783	-	-
Environmental Impact	-	-	-	-	6	2075	-	-	-	-
Computing Utility	3	683	1	141	-	-	-	-	-	-
Total Reusability	7	4195	23	9289	12	5753	12	5461	10	4348
Total Development	8	5879	30	11040	16	6394	14	6547	11	4743
Reusability Index	87.5%	71.4%	76.7%	84.1%	75.0%	89.9%	85.7%	83.4%	90.9%	91.7%

An inspection on Table 1 reveals that transport, spatial geometry, mathematic, statistics were the four most commonly used domains in the development of all five case study applications. However, the mathematics domain was the most popular one among the four as this domain was used by all five case study applications. The land use domain was not reused in the development of five case studies. In fact, two the framework classes representing two specific Lowry models in the land use domain were not suitable to support the development of the land use model of shopping centre in case study number 4. Insteads, a new class, class TShopGen was developed from the software reusability support of class Matrix in the mathematics domain. This indicates the genericity of class Matrix help address the specificity of land use model.

In summary, as shown in the last row of Table 1, the reusability index values for all five case studies are high ie. from 71.4 to 91.7 per cent of classes and from 75 to 90.9 per cent of programming lines. On average, reusability index values of 81 and 83.9 percent of classes and lines, respectively, were reused from the framework. In terms of line coding, only 16.1 per cent of application specific development were required as a result of 83.9 per cent saving due to software reusability from the framework. The following formula is used in converting the 83.9 per cent saving in the development to time unit.

$$\text{Estimated Time Saving (hours)} = N \times R \times U \qquad (1)$$

where:

N is number of reused coding lines;
R is rate of program coding (number of coding lines per hour); and
U is component/class utilisation factor (%).

The N value of 29046 lines was selected from the sum of the number of reused lines from all five case studies in Table 1. The R value is based on the author's own coding rate of 6.5 lines per hour, but different values could be determined from surveys of work rates performed by commercial code developers. This coding rate is recorded from 850 hours spent in coding 5557 programming lines. The component utilisation factor represents how much the software development for a component (a class) was reused in the development of new application. It can be defined as the ratio between a number of functions being used in the development of new application and the total number of functions being implemented for a particular component/class. For example, there are 28 functions implemented in class Matrix, but only 14 functions are required by the new application. In this case, class utilisation factor is 50 per cent (ie. 14/28). The measurement of this class utilisation factor is time consuming as it requires the manual counting of functions in every framework's class that provide direct support to the applications. For the five case study application development, a value of 25 per cent is assumed as an average class utilisation factor in the five case study applications. Therefore, given a total of 29046 reused lines, coding rate of 6.5 line per hour and class utilisation factor of 25 per cent, a total of 1117 hours (= 29046 lines / 6.5 line per hour x 25 %) time saving over conventional programming techniques are estimated.

Table 2 provides a summary of recorded development and estimated time saving in the development of five case study applications.

Table 2: Development and Estimated Time Saving in the Development of Five Case Studies

Coding	Total Number of Lines	Recorded Development Time (hours)	Component utilisation factor	Estimated Time Saving (hours)
Reused Code	29046	4468	0.25	1117
Application Specific Development Code	5557	850	0.25	213
Total	34603	850		1330

34603 lines were coded in the development of five case studies in which 5557 lines were application specific code and 29046 lines were shared and reused by all five case study applications. A total amount of application specific development time was 850 hours (equivalent to 5 months) and total time saved due to software reuse of 1117 hours (equivalent to 6.5 months) was estimated. In summary, these estimates indicate that the object-oriented programming and COM approach have satisfied basic programming requirements (ie. problem-orientedness, software modularity, and software reusability) in transport computer modelling. A total of 1330 hours of time saving can be achieved for any future development which will be based on the development of five case study applications.

Conclusions

The problem being investigated is the practicality of component object model (COM) approach when applied to land-use/transport/environmental modelling. Current issues in the using and developing transport software packages were identified together with the potential of the COM approach in the development of transport software. The development process of a COM framework for transport modelling was presented. Five case study applications representing different levels of transport planning were selected as an input to the identification and construction of basic tools for eight supporting domains in the COM framework. Evaluation on the implementation of models of the five case studies has resulted in favour of the COM approach. In summary, the research found that the COM approach has supported the development of COM tools that highly complying with basic software quality factors, particularly and most significantly – software reuse factor.

However, the research is still in progress to answer the following question. What are the implications of adopting COM approach in developing more large-scale complex transport models incorporating other domains or frameworks such as economic-based and behaviour-based modelling framework?

References

INRO (1994) EMME/2 User's Manual, (INRO: Quebec).

Booch, G. (1991) Object Oriented Design with Applications (The Benjamin/Cummings Publishing Company: Sydney).

Boyce, D., Day, N. and McDonald, C. (1970) Metropolitan plan making, Regional Science Research Institute Monograph Series 4, Philadelphia.

Harris, B. (1968) Conference summary and recommendations; in Hemmens, G.C. (ed.) Urban development models, Highway Research Board Special Report 97, Washington, D.C.

Meyer, B. (1988) Object-oriented Software Construction (Prentice Hall: Sydney).

Taylor, M.A.P. (1991) "Traffic Planning by a 'Desktop Expert'", Computers, Environment and Urban Systems, Volume 15, pp.165-171.

Ton (1995) An Investigation of the Analytical of the Analytical Capability of Object-Oriented Programming in Transport Modelling, an unpublished PhD thesis, School of Civil Engineering, University of New South Wales, Kensington.

Traczt, W. (1988) "Reusability Comes of Age", in W.Tractz (Ed.), Tutorial: Software Reuse Emerging Technology, (Computer Society Press: Washington, D.C.), pp.23-32.

Traczt, W. (1988) "RMISE Workshop on Software Reuse Meeting Summary", in W.Tractz (Ed.), Tutorial: Software Reuse Emerging Technology, (Computer Society Press: Washington, D.C.), pp. 33-35.

Traczt, W. (1988) Tutorial: Software Reuse Emerging Technology, (Computer Society Press: Washington, D.C.).