

# Transport Data Management

*Applications of Software Engineering  
Best Practice to Transport*

---



**CITY SCIENCE**  
endless possibilities

## Contents

Contents .....	1
1 Introduction .....	3
2 The Cost & Quality Challenge .....	4
3 Quality.....	7
4 Collaboration .....	11
5 Efficiency.....	12
6 Conclusion .....	15

## 1 Introduction

This paper sets out the opportunities afforded to the transportation industry from the adoption of techniques, practices and data processing methods established within the software industry as ‘best practice’.

Developing software is a complex undertaking that often involves whole teams of people. At the same time, software failure is expensive, with numerous high-profile examples of implementation projects running over budget, being delayed or being shelved entirely due to functionality not being delivered. Given the requirement to reduce risk and manage software delivery projects, the software industry has invested considerable effort to reduce the cost of development and improve software quality. Overall, these best practices are shown to reduce risk and life-time costs of software projects.

Transport modelling exercises share many similar features with large-scale software development projects. For example, transport modelling projects often require:

- Collaboration across large expert teams;
- Management of complex model structures and data flows;
- Detailed expertise including data processing and coding;
- Multiple versions and scenarios to be managed simultaneously;
- Continuous tracking and audit to ensure adherence with user expectations of quality (e.g. DfT WebTAG guidance).

In addition, the world of modelling is changing with growing expectations of what models will be able to do. For many authorities this includes the potential to link transport models with wider transport datasets (including real-time datasets) and use models for a range of purposes beyond traditional ‘static’ strategic scheme appraisals.

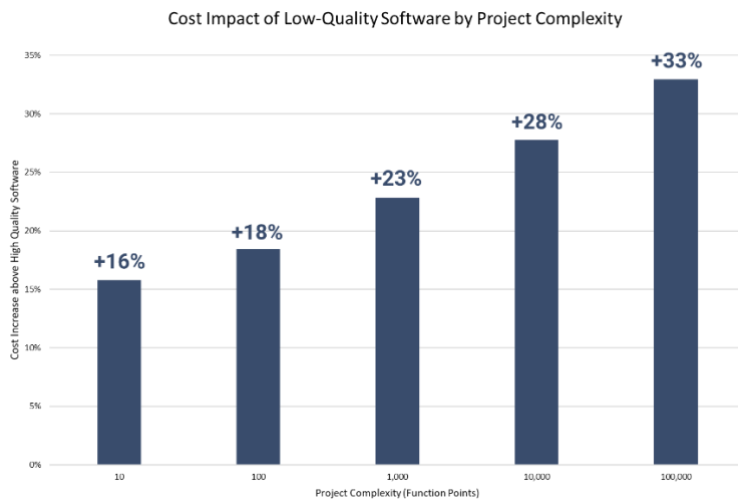
In this document we explore the potential benefits that can be afforded to transport modelling through the adoption of software engineering best practice. Overall, we conclude there are considerable opportunities to utilise these best practices to delivery three key benefits:

- **Quality** – Using best practice to strengthen quality assurance, improve model transparency and reduce risk;
- **Collaboration** – Making best use of existing tools to enhance collaboration within teams, with external partners and stakeholders; and
- **Efficiency** – Using best practice to streamline processes and improve performance and productivity.

## 2 The Cost & Quality Challenge

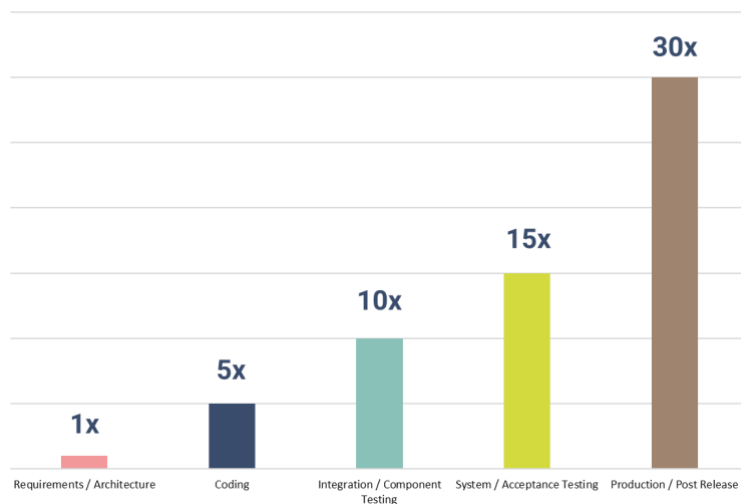
While software is among the most widely used products in human history, it also has one of the highest failure rates of any product due to the complexity of developing software of sufficient quality (Capers, 2011). Software failure is expensive, with numerous high-profile examples of implementation projects running over budget, being delayed or being shelved entirely due to functionality not being delivered. The cost of poor software quality has been shown to increase as complexity increases. Analysis shows that for the most complex projects, low quality in software development can increase costs by a third (Capers, 2011). Even for the most simple projects, low quality software development can increase overall costs by 16%. In addition, the costs of fixing flaws in software projects has been shown to increase exponentially the further the project progresses without the flaw being identified (Figure 2).

Reducing the cost of software development and improving software quality have therefore been critical objectives for the software development industry to reduce risk and manage delivery projects.



*Figure 1: Cost Impact of Low-Quality Software by Project Complexity (Adapted from Capers, 2011)*

The Relative Cost of Fixing a Flaw At Different Stages of the SDLC



*Figure 2: Relative Cost of Fixing a Flaw, Source: National Institute of Science & Technology, US Dept for Commerce 200*

In addition, given the complexity of modern software, products are necessarily developed by large teams often with specialist expertise. Since many computer programs remain in use for extended periods of time, development practices need to facilitate both initial development and subsequent maintenance and enhancement by people other than the original authors. Software engineering practices therefore also developed to facilitate teamwork and co-ordination to improve the management of systems that cannot be completely understood by one person.

### **Transport Modelling – a Complex Software Project**

Transport modelling shares many similarities with complex software development. It is common for transport model development to require a number of components to be brought together based on the needs of the users. Detailed network coding and data processing is then required to develop the models. Through this process the modelling team will need to deal with a range of data and work collaboratively across the model, while ensuring quality.

#### **Similarities include the following:**

- **The need for collaboration across large expert teams.** Transport models are increasingly built by teams spanning multiple individuals, sites, and even organisations. With some models developed using offshore subcontractors, across multiple consultant engineers, or using remote working practices, the need for tools to enable collaboration has never been stronger.

- **Management of complex model structures and data flows.** In many cases transport models might include multiple components including a variable demand model, a public transport assignment and a highways assignment among others. Models are often developed on extensive data collection which needs to be processed. Changes to underlying data released also need to be tracked and systems in place to avoid rework.
- **Detailed expertise including data processing and coding.** Different modelling systems often require different expertise, in particular to code up the system representations within the modelling system. Extensive data processing is also often undertaken – this could include gravity modelling, techniques for data fusion or for processing model outputs.
- **Multiple versions and scenarios to be managed simultaneously.** During a model development process the model will go through a large number of iterations with multiple team members working on the project at the same time. In addition, once a model is complete, many future scenarios will then be prepared, often involving multiple schemes and time periods. Increasingly, there is a desire to also stress-test future scenarios, potentially creating many more output data points that all need to be managed.
- **The need to ensure adherence with user quality expectations.** Since transport models need to be relied upon by local authorities to support detailed strategies and business cases for investment, the quality of model development must be maintained. In the UK, the DfT provides best-practice guidance through WebTAG. It is often the responsibility of the development team to evidence that best-practice has been followed.
- **Increasing requirement to engineer solutions around the core model.** Increasingly there is demand from stakeholders to the transport model for greater transparency, more open data and potentially even new functionality to enhance its use. This creates new requirements for integration with wider software systems such as geospatial tools, APIs and web-based applications.

The software engineering industry has developed a range of best practices that can address these issues. In particular, there are considerable opportunities to utilise software engineering best practices to delivery three key benefits to transport modelling:

- **Improved Quality Assurance** – Using best practice to strengthen quality assurance, improve model transparency and reduce risk;
- **Enhanced Collaboration** – Making best use of existing tools to enhance collaboration within teams, with external partners and stakeholders; and
- **Streamlined Efficiency** – Using best practice to streamline processes and improve performance and productivity.

How software engineering best practice can enable these benefits is considered in the following sections.

### **3 Quality**

#### **Managing data and model versions**

Transport authorities and their modelling teams want to ensure that versions of models can be tracked alongside their input data and assumptions throughout the life of the build and operation of the model. In software engineering, best practice is to use a system of automated version control to manage changes. This technique can be applied to transport modelling with a range of benefits.

One of the key ways that errors can be detected and model owners can be assured of the authority of their models is through automated change-tracking and automated audit/change logs. Change control responds to the fact that coordinating complex projects is difficult. Without the ability to control changes, development teams can introduce inconsistencies, complexity can grow and, as a result, the risk of introducing errors can increase.

At its most basic function, version control enables users to establish a clear release process for model assets. Within a version control system, a 'Master Model' can be stored on a release branch, clearly identifiable and readily available to users. This branch can include the associated input and output datasets in a prescribed format so that users always have access to the latest official version of the model. Using a web-based system, a wide-number of authorised users would be able to access the model. Users would then be able to make changes on their own branches and submit these back to the owner of the Master Model for approval and inclusion. Any changes are then clearly identifiable, fully described and with a full digital audit trail.

Future scenarios can also be managed in this way – transport modellers can take a branch of the future year model, make changes and then submit the changes back to the owner of the repository for acceptance as a new scenario.


#### **Reviewing Changes and Minimising Unwanted Alterations**

Version control also provides a system to review every change, just as you would in a word document. Figure 3 presents an example 'change log' within a version control system. The system summarises the number of lines added and removed and then highlights the changes (shaded red and green). This means that every change is tracked and that both users and managers have full transparency over what has been coded and, in particular, enables teams to identify any unwanted changes. This works especially well for long scripts or code such as .dat files.

**Remove Default Flag and rename Variants** [Browse Source](#)

Removed the default flag from the AM Variant and renamed the variants to be uppercase period with dashes instead of spaces to be consistent with the other models.

🔖 master

 **Alex Dawn** <alex.dawn@cityscience.com> 1 year ago parent [8f93465af5](#) commit [cf64c568a0](#)

📁 1 changed files with 3 additions and 4 deletions [Split View](#) [Show Diff Stats](#)

+ 3 - 4 stats-config.json [View File](#)

```

@@ -17,8 +17,7 @@
17 17     }
18 18     },
19 19     "variants": {
20 -     "am": {
21 -     "default": true,
22 +     "2013-AM": {
23     "network": {
24     "file": "OSM_BaseA.dat",
25     "cached": "network_am.json"
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 -     "ip": {
36 +     "2013-IP": {
37     "network": {
38     "file": "OSM_BaseI.dat",
39     "cached": "network_ip.json"
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 -     "pm": {
50 +     "2013-PM": {
51     "network": {
52     "file": "OSM_BaseP.dat",
53     "cached": "network_pm.json"
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

*Figure 3: Example Version Control 'Commit Message' and Change Log (from City Science Cadence Software)*

### Understanding the Design Rationale for Model Components

In a version control system, every change is accompanied by a 'commit message' which documents what's changed and why. This means the project benefits from having a traceable description of all changes.

There are a range of areas in transport modelling where this is likely to be beneficial. For example, WebTAG states the following regarding network development: "Documentation of network coding assumptions is essential... Whilst it is recognised that assumptions can be changed (during calibration and validation), the need to record a consistent and sensible starting set of assumptions is important." Version control allows the development team to consistently document assumptions underpinning every change that is made.



## **Error Minimisation**

Since flaws in software are expensive, the software development industry has developed a range of tools and practices to aid error identification. These techniques can be applied to transport modelling to also minimise errors. Techniques are set out below:

**Linting:** Linting operates directly on code and identifies deviations from the formal structure of the coding language. This helps the programmer identify issues as they are developing code. In transport modelling, files parsers could be used to validate coding inputs at the point of upload to ensure the structure is correct. This will help avoid syntax errors in the coding.

**Testing:** In software development, Unit tests are written around each fragment of code to ensure the code functions as expected. This also ensures that when changes to the code are made, developers are notified whether the change has impaired the intended functionality. In transport modelling, tests can be developed that automatically run on transport modelling code – for example, automated testing against a coding manual. This enables continuous automated quality checking and ensures consistency between different analysts or teams.

**Code Reviews:** By providing a systemised process to review changes version control simplifies the process of reviewing code. In a transport modelling context, this can be enhanced by using visualisation tools to enable managers to quality assure what has been coded. As WebTAG states: “All key junctions and intersections should be formally reviewed. The review should seek to ensure that junction types are correctly defined and that the representation of key characteristics is appropriate and accords with available data sources.”

**Error Correction: Version control also simplifies error correction.** Since version control stores every change, the system enables a project director to track back to previous working versions of the model if a material error is ever introduced.

## Managing Spreadsheet Errors

The informality of spreadsheet development offers the benefit of flexibility, low initial cost and low barriers to entry across a wide range of industries and applications. However, research has shown that in general within spreadsheets, errors occur in a few percent of all cells, meaning that for large spreadsheets, there is almost near certainty that error exists somewhere (Panko, 2008).

Software development best practice follows strict development disciplines to mitigate the occurrence of errors, whereas spreadsheet development does not. Practices are likely to differ between teams and individuals. This leaves spreadsheets likely to hold residual errors and liable to change or deviation over time.

A range of transport model development processes currently rely on spreadsheets which play key roles around traditional modelling software. Within systems such as Excel managing consistency becomes difficult due to the potential for users to make manual changes, overwrite formulae or introduce other errors. Using an extensible version control system however, process and model development can be managed in the same way. A so-called 'pipeline' can be used to provide testing around all the end-to-end components of the transport model, reducing risk of error while at the same time allowing for greater flexibility.

### Overall:

Overall, integrating version control, validation and error checking support into the transport model development process can provide the following benefits:

- **Reduced risk of error from spreadsheet-based systems;**
- **Consolidated data, providing a useful and valuable asset in its own right;**
- **Improved audit trail around changes introduced through the calibration process;**
- **Improved documentation, going beyond documentation specifically for the LMVR;**
- **The ability to manage a consistent standard and approach across multiple suppliers;**
- **Future ability to automate data updates and production of downstream reports.**

**Example – Automated Route Choice Monitoring**  
WebTAG provides specific guidance on the number of routes that should be examined - "The number of pairs of zones which should be examined and displayed will be dependent on the size of the model. The following rule of thumb should be used: Number of OD pairs = (number of zones) x 0.25 x the number of user classes." (Department for Transport, 2014). Using an automated system, practitioner sign-off could be built into the workflows to monitor and report on the quality of the route choice review (i.e. by providing aggregate measures of quality akin to test coverage employed in software engineering). This would include appropriate coverage of HGV routing behaviour where specific additional tests may be warranted. These types of systems would ensure that all required checks are undertaken to the desired level of accuracy and coverage, reducing risk of error.

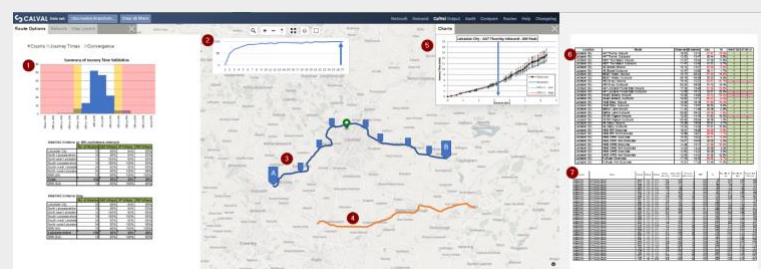


Figure 4: Example Route Choice Calibration Dashboard

## 4 Collaboration

A range of barriers exist within existing transport software systems which limit the degree to which users can share models and their data. This impedes both collaboration and transparency. Some of these barriers are set out below:

- **Knowledge Barriers:** Transport models include an extensive set of data assets that have relevance both to transport professionals but also potential users outside the transport industry. Knowledge of this data or the processes used is not widely available, and official reports (e.g. LMVR) do not aid in making these assets discoverable.
- **Cost Barriers:** License costs for software can inhibit the number of users that can have access to models and their data. As a result, access to transport models is often via third party custodians which can drive up cost and the time taken to access the model.
- **Access Barriers:** Access barriers include the inability to view and visualise the model data and outputs online; difficulties accessing relevant data from the models; and the limited interoperability of modelling assets with other systems (e.g. open source).

Many of these barriers can be overcome using software engineering best practices and, over time, through the use and development of additional tools to promote a more open and accessible modelling environment.

### **Overcoming Knowledge Barriers through Enhanced Visibility**

To counter issues such as complexity and ambiguous communications in software development a range of best practice processes have been developed. These range from visual systems to understand software components and dataflows, visual systems to enable individual users to identify issues more easily, systems to share workflow and systems to enable teams to collaborate on complex projects.

Once again, version control can act as a backbone, increasing visibility of the work being conducted and enhancing transparency of the process. Beyond this, in a transport modelling context, visualisation tools can then be used to enable consistent understanding, improved communication and greater transparency. Interactive dashboards can enable users to explore and explain all the inputs and outputs within transport models, fostering communication within teams and with stakeholders. Sharing functionality can help highlight model issues or communicate specific impacts. Together these tools can help share knowledge and promote an understanding of what is in the model and how it is being used.

### **Overcoming Cost Barriers through Web-Based Sharing**

Through web-based tools, the ability to share information between teams increases materially. Cloud technologies are breaking down barriers within and between organisations by delivering services and applications that can be accessed from any device. Reduced deployment costs and multiple users are reducing the barriers of traditional software. This, in turn, enables more users to view the model and increases transparency. Web-based tools, combined with appropriate authorisations and workflows, can also enable

models to be shared across organisations and with the public, for example as part of consultations. These approaches can improve collaboration and reduce costs

#### **Enabling Self-Service to Overcome Access Barriers**

Often, distribution of transport models between teams is facilitated by sending individual files via email, larger zip files, or sent via other file transferring systems (SendFiles, WeTransfer, FTP etc...). This can take up to two weeks, and if files are missing or further requests required, the delivery time could be even longer. Files passed around in such way are also subject to untracked changes and divergence.

A modern version control system, with appropriate adaptations for transport models, can provide self-service for authorised consultants, saving time and cost transferring these. This makes it much easier for a client to view their models. An online, version-controlled system that allows users to self-serve model components can also facilitate improved use of the model in downstream applications.

## **5 Efficiency**

One of the challenges levied at transport modelling is the cost and time taken to build and refresh models. The introduction of software engineering best practice could enable considerable automation of workflows through the use of 'pipeline's, reducing risk and improving efficiency. A pipeline is used to combine a version control system, a testing framework, and the processes used to create the end deliverable (Merron, 2018). In a transport modelling context this would be running and producing the model outputs and analysis as an end-to-end process from the underlying datasets.

In the short-term, setting up projects as a pipeline, could enable changes discovered later in the process to be incorporated much more easily. Over the longer-term end-to-end pipelines could enable much more frequent update cycles, going beyond the current frequency of 5-yearly updates, potentially at reduced cost. Time and resource spent producing the model could be reinvested into use of the model and development of a wider range of scenarios and what-if analyses.

#### **Using Version Control to Streamline Changes and Reduce Build Effort**

Figure 5 shows the potential benefits of version control where an authority is maintaining multiple scenarios associated with a base model. Traditionally, if a change is made to the base year model, every other model would need to be changed manually. Using a version control system, changes can be fed through to all the scenarios digitally saving time and effort making the bank of models much easier to manage.

In order to create new models under a traditional process, users will often start from an old or donor model. While models are regularly shared across organisations there is considerable difficulty managing changes and historic errors may be brought forward. Models from other organisations are already being used to streamline development in some cases. With a version control system changes across organisations can also be managed digitally. This could enable automated updating of changes between organisations saving further time and effort.

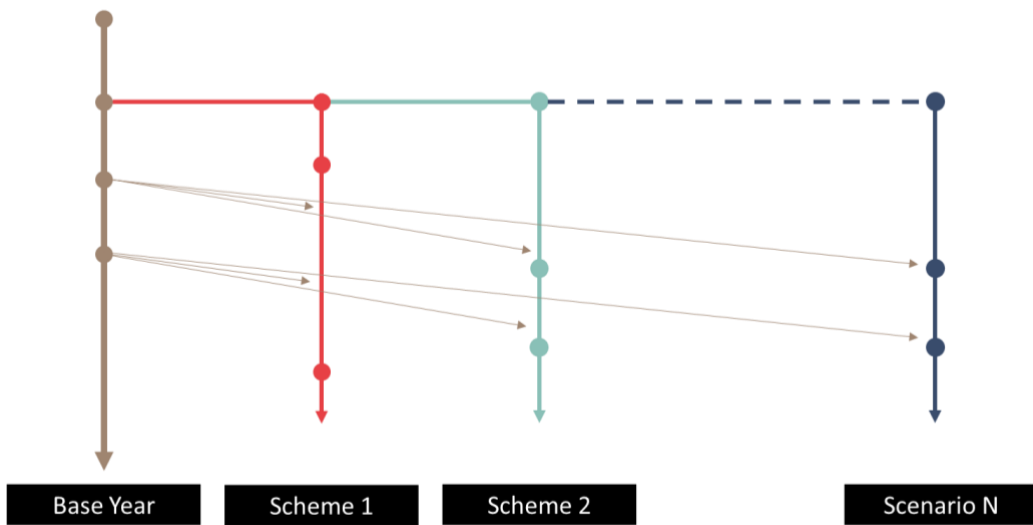


Figure 5: Benefits of Version Control, Example 1

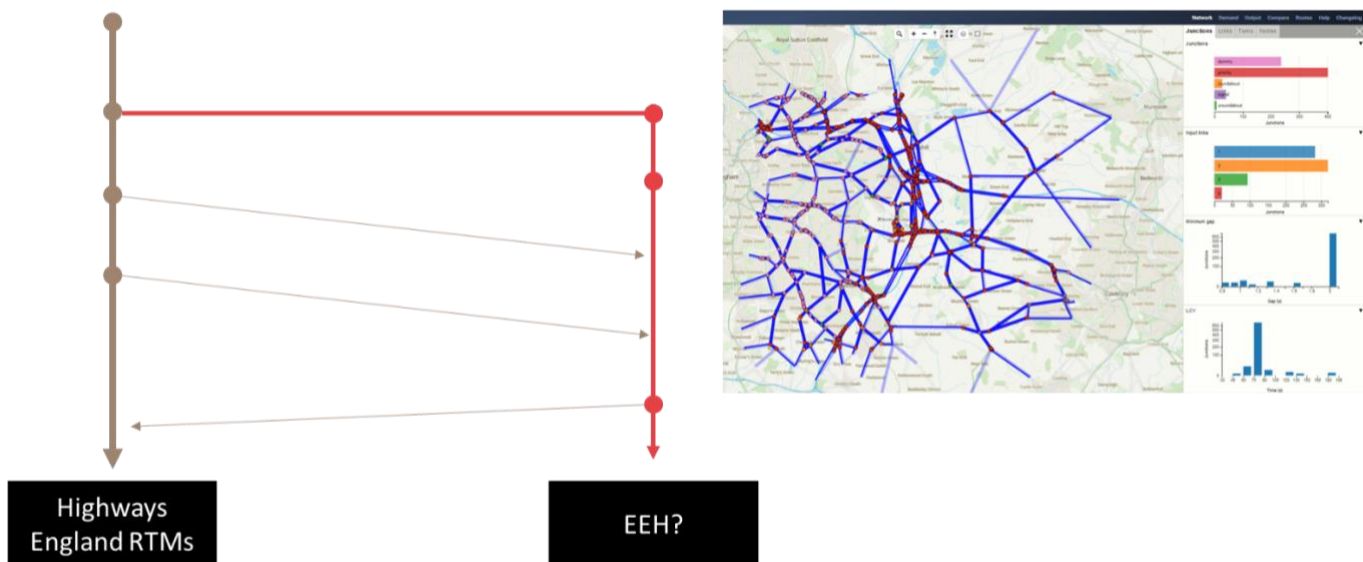


Figure 6: Benefits of Version Control, Example 2

### Streamlining Data Processing

Standardised data processing can be developed to process raw data prior to use within the model. For example, count point data could be standardised enabling it to be automatically converted into an appropriate format for viewing and analysis alongside the transport model.

Linking every step of the pipeline to an underlying data source could have a range of benefits. For example, designing the topological structure of the network first in GIS could enable model design decisions to be made prior to more costly coding, but would also allow for the ability for the model to map to OS link ids and capture data from national datasets (e.g. the road centre line). Not only could this save the practitioner from measuring the length of each model link, this additional data (e.g. road shape) could be essential for downstream uses (such as air quality modelling).

### **Calibration & Validation**

Calibration and validation is widely considered one of the most time-consuming and risky parts of model development. According to TfL guidance “Calibration can be a lengthy and time-consuming exercise.” Using pipelines, automated processes can be built around calibration and validation to accelerate the process while reducing risk.

### **Calibration & Validation Data:**

All models are developed using calibration data, which needs to be collected in the form of site observations and real-world measurement. The quality of the final model, and any analysis derived from it, depends on the data used during model development (TfL, 2010). Data used is often stored across a series of files and only consolidated in Local Model Validation Reports. This can make data difficult to find or validate and limits its wider use. In contrast, using a digital data management system, all relevant modelling data could be stored within the same environment as the model. Source data could be specified for calibration purposes and linked to the calibration model branches. Further automated tooling can then be introduced to support calibration and validation more widely.

### **Automating Pre-Calibration Checks:**

One of 'quick wins' at the calibration stage is to enhance the pre-calibration checks conducted automatically by software. Pre-calibration checks to the network and route choice, recommended by WebTAG, could be undertaken by automated tooling. With the count and calibration data stored in the same system as the model, automated tooling could then be used to undertake the key calibration checks at the network stage, for example:

- Automatically identify and visualise turning movements where the modelled capacity is less than the count;
- Automatically identify and visualise areas where calculated delays are significantly greater than observed delays;
- Automatically identify and visualise turning movements where modelled flows are significantly below observed flows;
- Automatically identify and visualise areas where modelled delays are materially lower than observed delays.

This would strengthen confidence in the network coding prior to commencing calibration while reducing the time and resource required for manual checks of the network. In addition, automated stress-testing could be brought forward (WebTAG Unit 3.1 para 9.2) to identify potential network issues early on in the process.

### **Trip Matrix Calibration & Validation:**

The Calibration of Trip matrices is concerned with two main tests – firstly that the prior trip matrices are validated by comparing total screenline and cordon modelled flows and counts by vehicle type and time period; and secondly to monitor changes brought about by matrix estimation. In both cases, automated tooling could be used to streamline these workflows and reduce the risk of errors that might be introduced through spreadsheets.

### **Assignment Validation:**

Finally, assignment validation could be undertaken within the software itself for a chosen model variant and data base-year. This could include both convergence checks as well as key comparison of flows, journey times and turning movements.

### **Streamlining Downstream Processes**

Finally, transport models often support a number of downstream tasks such as economic modelling, air quality modelling or reporting. By adopting software engineering best practice, an increasing number of downstream processes could also be brought into an end-to-end pipeline, essentially eliminating manual interfaces and updating these outputs automatically whenever input data changes.

## **6 Conclusion**

Transport modelling is changing. Driven by an explosion in data; the growing complexity of models; changing demands of customers and the public; an increasing desire to link strategic and real-time models; and the need for more granular predictions through time and space, transport modelling practitioners are having to reassess the traditional tools.

Robust data management and version control is a fundamental pre-requisite to support clients in this new world. But embracing these new technologies does not have to be costly. Quite the contrary – new technology, developed from software engineering best practices will reduce risk and cost while creating better outcomes for clients. By improving quality assurance, enhancing collaboration and streamlining efficiency, data management tools will help you maximise value from your modelling activities.



## 7 References

- Atlassian. (2019). *Software Development: Modern practices and where it's headed*.
- Capers, J. B. (2011). *The Economics of Software Quality*. Addison-Wesley.
- del Aguila, I. P. (2014). *Milestones in Software Engineering and Knowledge Engineering History: A Comparative Review*. Scientific World Journal.
- Department for Transport. (2014). *TAG Unit M3.1, Highway Assignment Modelling*.
- Gartner. (2018). *Gartner Market Databook Q4:18*.
- HM Treasury. (2018). *The Green Book: Central Government Guidance on Appraisal and Evaluation*. OGL Crown Copyright.
- Krasner, H. (2018). *The Cost of Poor Quality Software in the US: A 2018 Report*. Consortium for IT Software Quality.
- Merron, D. (2018). What is a Pipeline in Software Engineering? Intro to Deployment, CI, & CD Pipelines. BMC Blogs. Retrieved from <https://www.bmc.com/blogs/deployment-pipeline/>
- Panko, R. (2008). *What We Know About Spreadsheet Errors*. Journal of End User Computing's Special Issue.
- Roggio, B. (2011). *Software Engineering*. University of North Florida.
- Standish Group. (n.d.). *Chaos Report*.
- TfL. (2010). *Traffic Modelling Guidelines - TfL Traffic Manager and Network Performance Best Practice*.
- TfWM. (2019). *Transport Modelling Development Specification*. West Midlands Combined Authority.
- Zakheim, B. (2017). *How Difficult Can It Be to Integrate Software Development Tools? The Hard Truth*. InfoQ, available at: <https://www.infoq.com/articles/tool-integration-hard-truth/>.