

Author: Sven Maerivoet
Last modified: 5 January 2009

FLEXSYS: Operational Context

Deliverable for WPs 2.9/2.10/2.12/2.13/2.17/3.2

Document history

<u>Date</u>	<u>Author / Input from</u>
11/11/2007	Sven Maerivoet (TML), Chris Tampère (KUL-V&I)
12/05/2008	Sven Maerivoet (TML)
25/09/2008	Sven Maerivoet (TML)
20/11/2008	Sven Maerivoet (TML), Wim Michiels (Tritel)
05/01/2009	Sven Maerivoet (TML)

Table of contents

1. Introduction.....	3
1.1 A brief note on the UML modelling language.....	3
1.2 Top-level module decomposition view of the FLEXSYS platform	4
2. Data management module	4
2.1 Data input.....	5
2.1.1 Detector conformity	5
2.1.2 Types of traffic flow measurements.....	5
2.1.3 The stochastic nature of the measurements.....	7
2.2 Data storage.....	10
2.2.1 Overview.....	10
2.2.2 Construction of a historical database	10
2.2.2.1 A note on clustering.....	10
2.2.2.2 Methodology for defining typical days.....	11
2.2.2.3 Defining special days	12
2.2.2.4 Creating typical daily profiles.....	14
2.3 Data validation.....	15
2.3.1 Methodology for validation	15
2.3.2 Detecting outliers	15
2.3.2.1 Different kinds of outliers.....	15
2.3.2.2 Testing against physical outliers.....	16
2.3.2.3 Testing against statistical outliers.....	16
2.3.3 Reconstructing faulty measurements.....	18
3. Traffic management module	18
3.1 Overview of the work flow.....	18
3.2 Traffic model.....	20
3.2.1 Off-line part of the model	21
3.2.2 On-line part of the model.....	23
3.3 Incident detection module.....	23
3.4 Operator's interaction with the traffic management module.....	24
3.5 Data specifications for the traffic management module.....	26
3.5.1 Input to the traffic model	26
3.5.1.1 Road network description and traffic demand	26
3.5.1.2 Input with detector locations and zone definitions	28
3.5.2 Output of the traffic model to the GUI.....	29
References	30

1. Introduction

This report describes the operation context of the model used in the FLEXSYS project. It is based on the requested information contained in the FLEXSYS IBBT Technical Annex version 1.4.1, 2007-06-11.

We first give a small explanation of the UML modelling language, used to draw various diagrams detailing the project's software architecture. Next, we provide a top-level view on two aspects of the FLEXSYS platform components, i.e., the data management and traffic management modules. Each of these two modules is subsequently discussed in detail. The last part of this report gives an overview of the expected data input and output for the traffic management module.

In this Section, we provide the reader first with a reminder of the modules that this report discusses. Next, we present a clear overview of the work flow within this part of the FLEXSYS project. But first, we shed some light on the use of UML diagrams encountered in the various formal specification documents in the project.

1.1 A brief note on the UML modelling language

Up until halfway the nineties, there was an abundance of formal specification languages. The goal of these languages was to provide programmers and software architects with a common method for exchanging information, user specifications, programmatorical aspects, et cetera. Within this existing plethora of different types of such methods, the need arose for a single scheme that represents the software specification process, usable in an industrial context. The non-profit organisation *Object Management Group*¹ (OMG), founded in 1989, started a task force "Object Analysis and Design" in 1996.

In 1997, the efforts of this task force resulted in the release of a unified industrial standard, i.e., the specification of the *Unified Modelling Language*² (UML) v1 standard. It has since then become the formal and de facto standard. UML provides a standard frame work for modelling software, i.e., designing software applications before they are programmed. With UML, the structure and design of software can be specified, visualised, and documented, all this in a programming-language independent fashion.

Within the UML community, people adopt the use of views and diagrams [Wuy05]. Views can be considered as projections of the complete system, whereby each view highlights a particular aspect of the system. Views can be catalogued as follows:

- **Use case view:** shows the functionality of the system as perceived by external parties. In any software project, the use cases drive the development as they integrate customers, designers, developers, and testers.
- **Logical view:** shows how the functionality of the system is designed and provided.
- **Component view:** shows the organisation of the code components and their dependencies.
- **Concurrency view:** addresses the problems with communication and synchronisation for a concurrent system (e.g., real-time databases).
- **Deployment view:** shows the deployment of the system into the physical architecture with computers and devices.

¹ <http://www.omg.org/>

² <http://www.uml.org/>

In order to describe these views, a modeller can use a certain number of diagrams, for example:

- Use-case diagrams.
- Class diagrams.
- Object diagrams.
- State diagrams.
- Sequence diagrams.
- Collaboration diagrams.
- Activity diagrams.
- Component diagrams.
- Deployment diagrams.

Within the FLEXSYS project, the architecture of the system is specified using UML diagrams that provide us with a clear representation of the requirements and format of the complete model. For more information on the use of software specifications, we refer the reader to the work of Bass et al. [BCK03].

1.2 Top-level module decomposition view of the FLEXSYS platform

The information contained in this document mostly refers to the data management and traffic management modules of the FLEXSYS project, as given by the software specifications of the DistriNet research group [HM07]. Figure 1 gives a recapitulation of their relation by means of an UML component diagram.

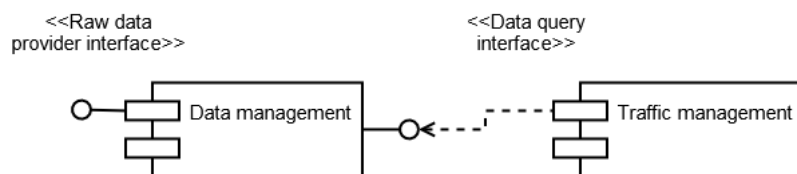


Figure 1: A part of the top-level module decomposition view of the FLEXSYS platform, showing the data management and traffic management modules together with their respective interfaces.

2. Data management module

In this Section we discuss the different aspects surrounding the data management module. An overview based on the component diagram is given in Figure 2. It illustrates the interaction between the validation and collection of the data.

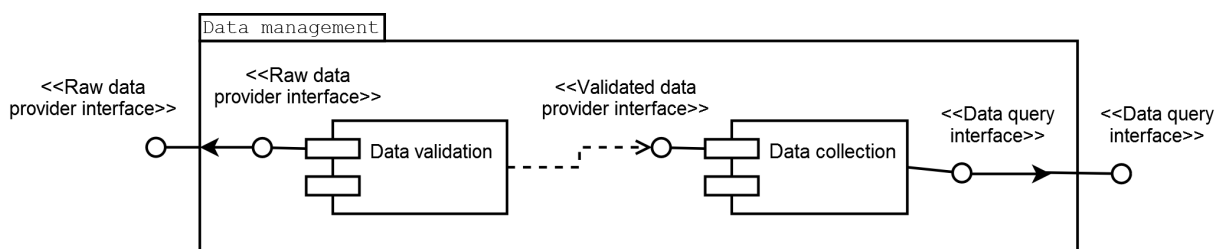


Figure 2: A component diagram illustrating the data management module within the FLEXSYS project.

Discussing the data management module is done by dividing it into three parts:

- input of the data,
- storage of the data (including the construction of a historical database),
- and validation of the data.

The next Sections go into more detail for each of these aspects.

2.1 Data input

2.1.1 Detector conformity

Given the setup as explained in the system architectural specifications of the DistriNet research group, we make a distinction between on the one hand detectors that give output conforming to the FLEXSYS standard, and on the other hand detectors that do not. In this latter case, we believe that the responsibility for giving suitable output measurements for a new detector provider wishing to participate in the FLEXSYS project, lies with the provider himself. As such, these detectors need a proxy that translates their output so that it can be used within the FLEXSYS project; these proxies define so to speak the rules of the game in order to conform to the FLEXSYS standard [HM07].

There are two types of information entering the system: normal data and incident data. Normal data stems from detectors like radars and cameras. The data is sampled at an interval of one minute, and consists for each minute of:

- the total number of vehicles detected during the last minute,
- and the average speed of all these vehicles detected during the last minute.

The incident data entails journalistic information, video footage, signals provided by an automatic incident detection camera, ... As opposed to the continuous stream of normal data, the incident data enters the system on an event-based schedule.

When such measurements enter the system, there are two possibilities:

- either we first validate them, after which we store them,
- or we store them in their raw form, validating them afterwards.

At the moment, we select the first option. The benefits of having the raw data available have not been determined yet.

2.1.2 Types of traffic flow measurements

All traffic flow measurements obtained by detectors, can be considered as being temporal or spatial in nature, or a combination of both [Mae06b]. Figure 3 gives an overview of such measurements, depicted in a so-called time-space diagram. In this diagram, the positions of all the vehicles trace out distinct vehicle trajectories (for multi-lane traffic, the trajectories cross each other). As the time direction is horizontal and the space direction is vertical, the vehicles' respective speeds can be derived by taking the tangents of the trajectories: accelerating vehicles have steep inclining trajectories, whereas those of stopped vehicles are horizontal.

Instead of considering each vehicle in a traffic stream individually, we now ‘zoom out’ to a more aggregate macroscopic level (e.g., traffic streams are regarded as a fluid). This allows us to measure some macroscopic traffic flow characteristics based on the shown time-space diagram. To this end, we define three measurement regions:

- R_t corresponding to measurements at a single fixed location x_* in space, during a certain time period T_{mp} . An example of this is a single inductive loop detector (SLD) embedded in the road’s concrete.
- R_s corresponding to measurements at a single instant t_* in time, over a certain road section of length K . An example of this is an aerial photograph.
- $R_{t,s}$ corresponding to a general measurement region. Although it can have any shape, in this case we restrict ourselves to a rectangular region in time and space. An example of this is a continuous sequence of images made by a video camera detector.

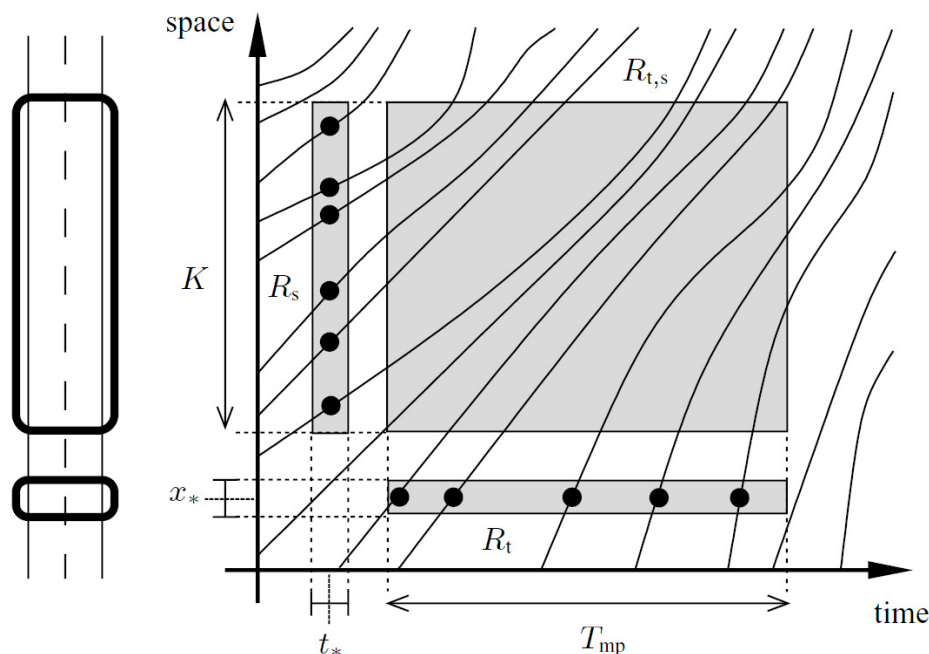


Figure 3: A time-space diagram showing several vehicle trajectories and three measurement regions R_t , R_s , and $R_{t,s}$. These rectangular regions are bounded in time and space by a measurement period T_{mp} and a road section of length K , respectively. The black dots represent the individual measurements.

Taking into account the previous remarks with respect to measurement regions in the time-space diagram, we consider the most important measurements to be:

- **flows** (the number of vehicles/hour passing by a specific location),
- **mean speeds** (expressed in km/h),
- and **travel times** (e.g., expressed in minutes).

Also important, but to a lesser extent, are:

- **occupancies** (the fraction of time a measurement location is occupied by vehicles),
- and if possible, **densities** (the number of vehicles/km on a certain section).

In traffic flow literature, flows are typically denoted by q , mean speeds by \bar{v} , travel times by T , occupancies by ρ , and densities by k . When talking about measurements without being specific as to what measure exactly, we use the symbol z in the remainder of this document.

2.1.3 The stochastic nature of the measurements

Raw traffic flow measurements stemming from, e.g., cameras, radars, loop detectors, ... are typically quite noisy in nature. Consider for example the flows (i.e., the total number of vehicles passing by a certain location during one hour) depicted in Figure 4. Here we can see a typical daily profile, with low traffic during the night, a more or less broad peak during the morning rush hour, moderately traffic during the bulk of the day, and a less pronounced peak during the evening rush hour (the measurements were obtained from single loop detectors embedded in three adjacent lanes on the E40 motorway). As can be seen, the flow exhibits large stochastic fluctuations.

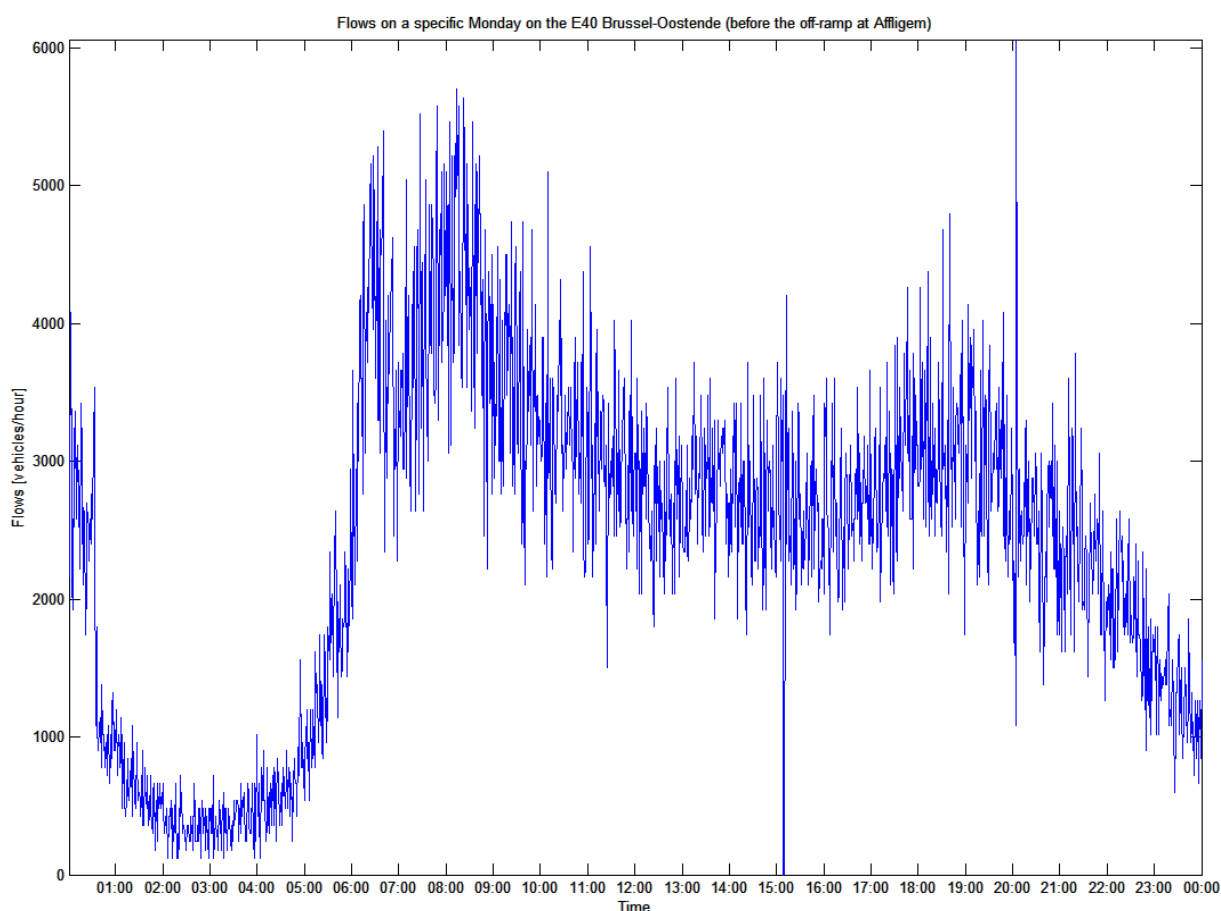


Figure 4: The flows over all three lanes on a specific Monday in 2001 at the E40 (Brussel-Oostende) near the off-ramp at Affligem.

In order to get a more quantitative view on these fluctuations, Figure 5 shows the standard deviation of the daily flow for each half hour of the day. The Figure clearly illustrates how the within-day dynamics lead to a very clearly pronounced peak during the morning rush hour.

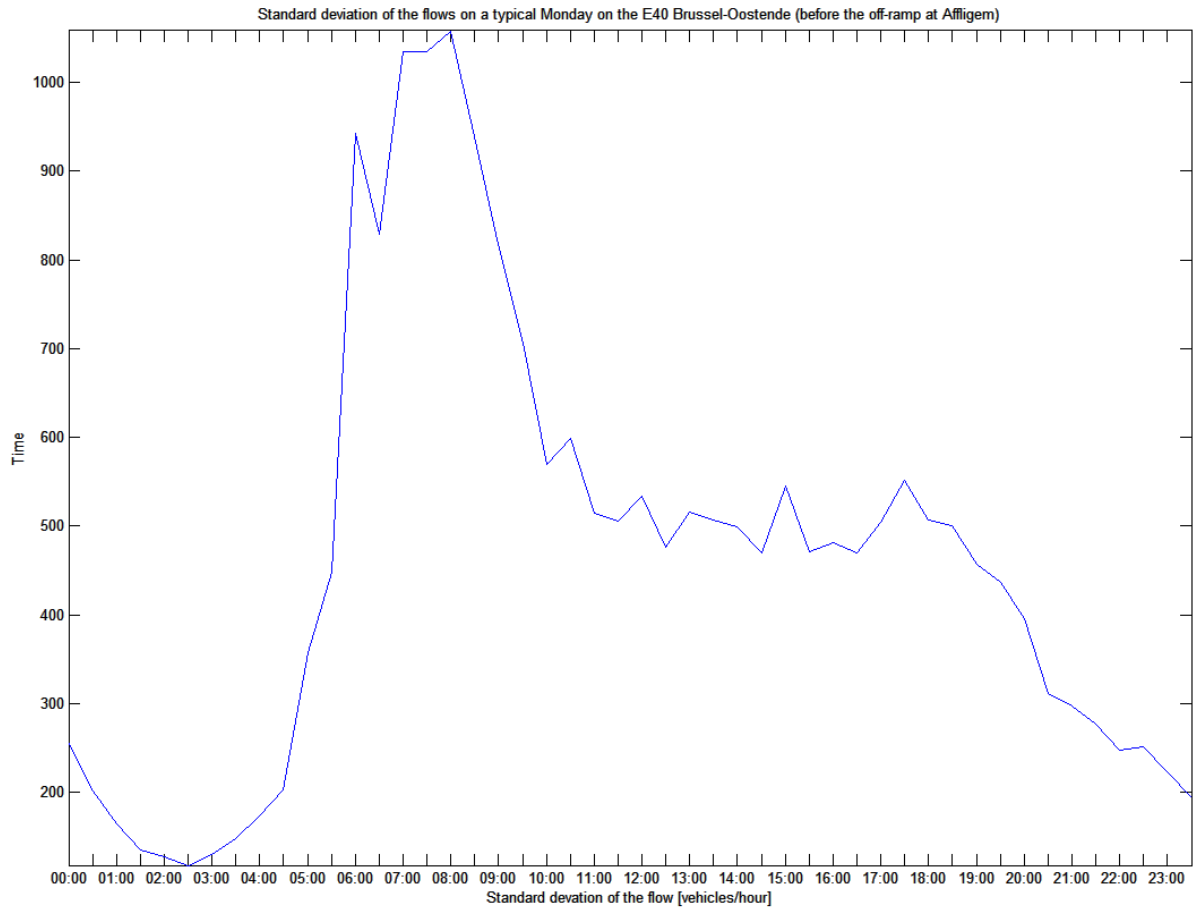


Figure 5: The standard deviation of the total flows over all three lanes on a typical Monday in 2001 at the E40 (Brussel-Oostende) near the off-ramp at Affligem. The curve gives us an indication of the within-day variability.

Furthermore, considering multiple days as in Figure 6, we can see how the day-to-day variability is quite high during the bulk of the day, with night-time traffic almost without large fluctuations.

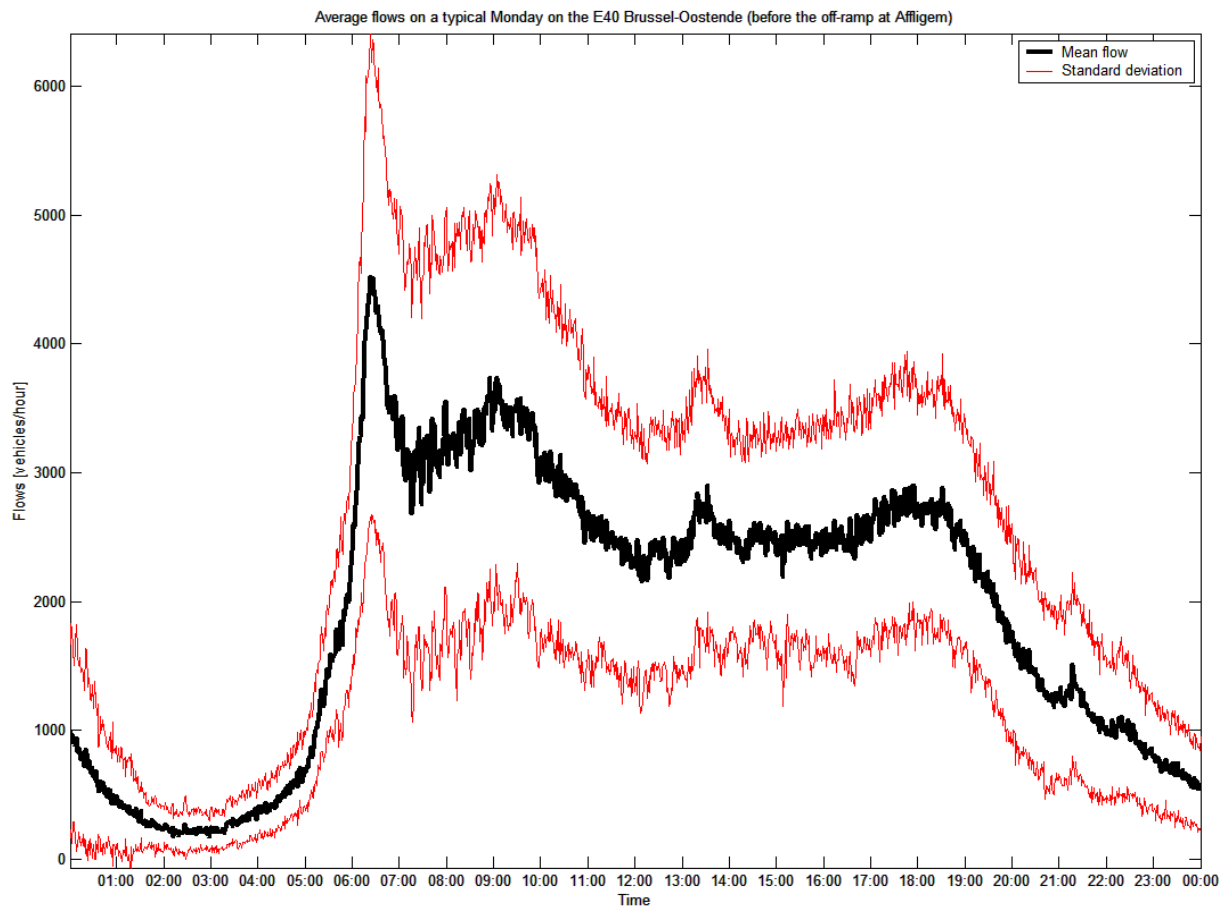


Figure 6: The average total flows over all three lanes on a typical Monday in 2001 at the E40 (Brussel-Oostende) near the off-ramp at Affligem. The thick black solid line denotes the average of all Mondays, whereas the two thin red lines envelope the average with its standard deviation, giving us an indication of the day-to-day variability.

2.2 Data storage

2.2.1 Overview

Within the FLEXSYS project, local traffic measurements initially stem from detectors (e.g., intelligent cameras, radar detectors, ...), and are aggregated into a database containing the raw data. However, before this data can be used by the models in FLEXSYS, it has to be validated (see also Figure 7 for a schematic overview).

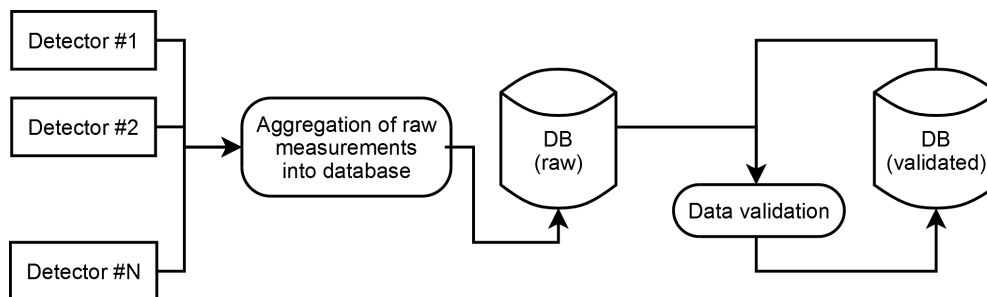


Figure 7: Schematic overview of transforming raw input data into a database with validated measurements.

2.2.2 Construction of a historical database

Each time when new data enters the system, it can be considered as historical information. It is then possible to cluster the data (i.e., grouping them logically according to a predefined criterium of likelihood), so that specific typical days are available. Broadly defined, there are two methods for clustering of data:

- clustering without a priori knowledge,
- and clustering with a priori knowledge.

In the remainder of this Section we first look at both clustering methods, after which we give an example of clustering with a priori knowledge applied to detector measurements stemming from the R0 ring around Brussels. To this end, we first consider so-called special days that have traffic patterns deviating from the normal observed behaviour, after which we

2.2.2.1 A note on clustering

Clustering without a priori knowledge

In this case all measurements are considered ‘blindly’, implying that it is up to the clustering method itself to ‘discover’ the groups in the data. There exist several techniques to do this, each time looking at several aspects such as:

- How similar are measurements from the same group?
- Are the various groups sufficiently different from each other?
- Can we say anything about the number of measurements in each group?
- ...

A more mathematical formulation of the above questions addresses the pairwise distances between the measurements in a group and between the groups themselves. The concept ‘distance’ should be interpreted in a broad sense; it is actually a metric that is used to compare the similarity of data. Examples are the minimisation of the least squares of the differences between the measurements, or the automatic determination of the optimal number of groups such that all groups contain

approximately the same number of measurements and yet are sufficiently different (which is not always possible).

There exist quite a number of ways to do clustering. For a good overview we refer to the work of Xu and Wunsch [Xu05] and Eick et al. [Eic].

Clustering with a priori knowledge

In case there is predefined knowledge, and we thus a priori know to which group the measurements belong, than the clustering process can be simplified. It boils down to placing the measurements in the correct group.

For the detector measurements in this project we can identify the day of week the measurements occur, thus automatically differentiating between them (weekdays, weekends, and (special) holidays). Note that the construction of typical days is always detector specific, as the detector output closely resembles the dynamics of the traffic driving past it.

2.2.2.2 Methodology for defining typical days

In this Section we use clustering with a priori knowledge. To this end we first look at so-called ‘special days’, because they can not be used when constructing a typical weekday. Afterwards, we apply this method to detector measurements stemming from single loop detectors on the R0 ring around Brussels (see also Figure 8).

Along the R0 ring around Brussels, we consider the single loop detectors at Halle, Huizingen, Beersel, Ruisbroek, Anderlecht, St.-Pieters Leeuw, Anderlecht-Pede, Dilbeek, Groot-Bijgaarden, Merchtem, Strombeek, Vilvoorde, Machelen, Zaventem, Wezembeek-Oppem, Tervuren, Oudergem and Groenendaal. For all these detectors we calculate the total flows at each post (i.e., a group of neighbouring detectors in adjacent lanes), as well as the mean speeds (weighted with the flows).

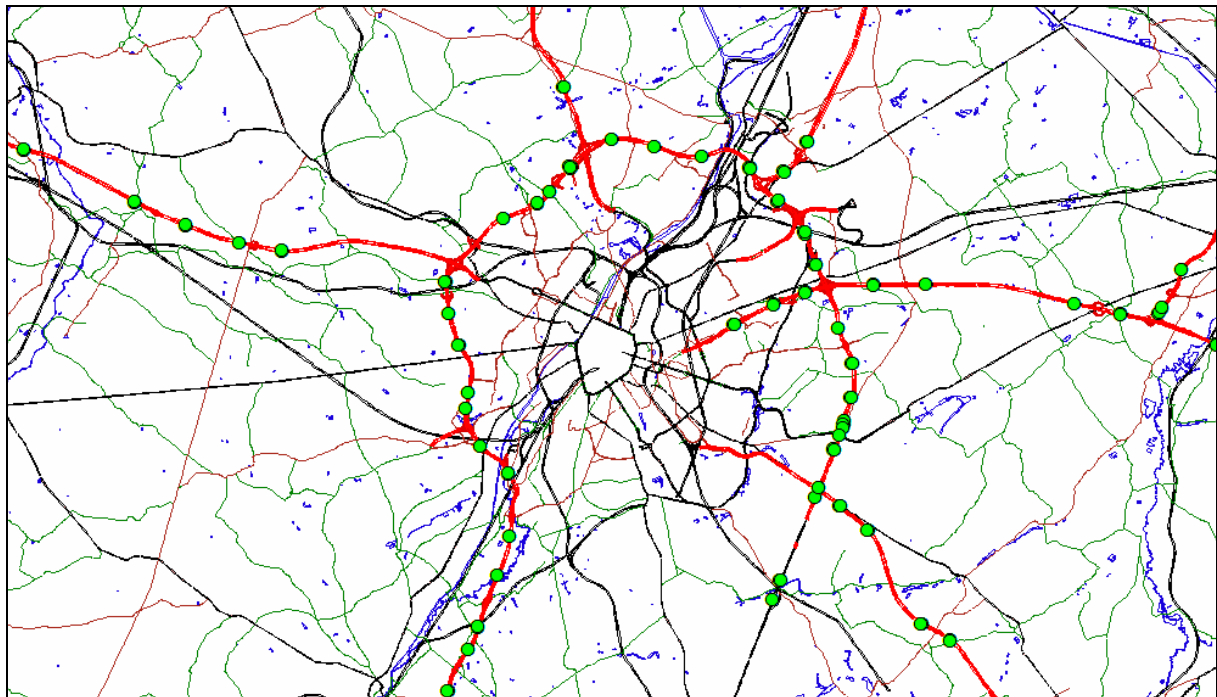


Figure 8: The R0 ring around Brussels; the green dots denote the locations of some single loop detectors.

2.2.2.3 Defining special days

When defining special days, the criterion used is that a special day has a significantly different traffic flow pattern in comparison to normal days (e.g., Easter Monday which has a different traffic load on the road network as opposed to a regular Monday). The first days that come to mind are the official holidays. They are predefined for each year. Next to that, we also take into account the bridge days (e.g., Mondays or Fridays when the holiday is a Tuesday, respectively Thursday).

Vacation periods (e.g., July and August) are not considered as special days, because collective leave is not an issue here (which on the contrary is the case for official holidays when more people are simultaneously on leave). An exception to this rule is the Christmas vacation, which is considered as special days in its entirety.

Given the above reasoning for defining special days, we also check if these days do have a different traffic flow pattern. In the remainder of this Section, we consider the total traffic flow pattern that is recorded on the R0 ring around Brussels (clockwise direction). This area is characterised by strong, structural congestion during the morning rush hour near the bridge of Vilvoorde, and during the evening rush hour near the Vierarmenkruispunt, Tervuren, and Wezembeek-Oppem.

Consider a time-space diagram of the mean speeds as recorded by the various detectors. In such a time-space diagram, time runs horizontally from left to right; the driving direction is vertically from bottom to top. Congestion waves that propagate upstream give rise to blocks with slower mean speeds. An example of this can be seen in Figure 9; the yellow/red blocks denote congestion during the morning and evening rush hours.

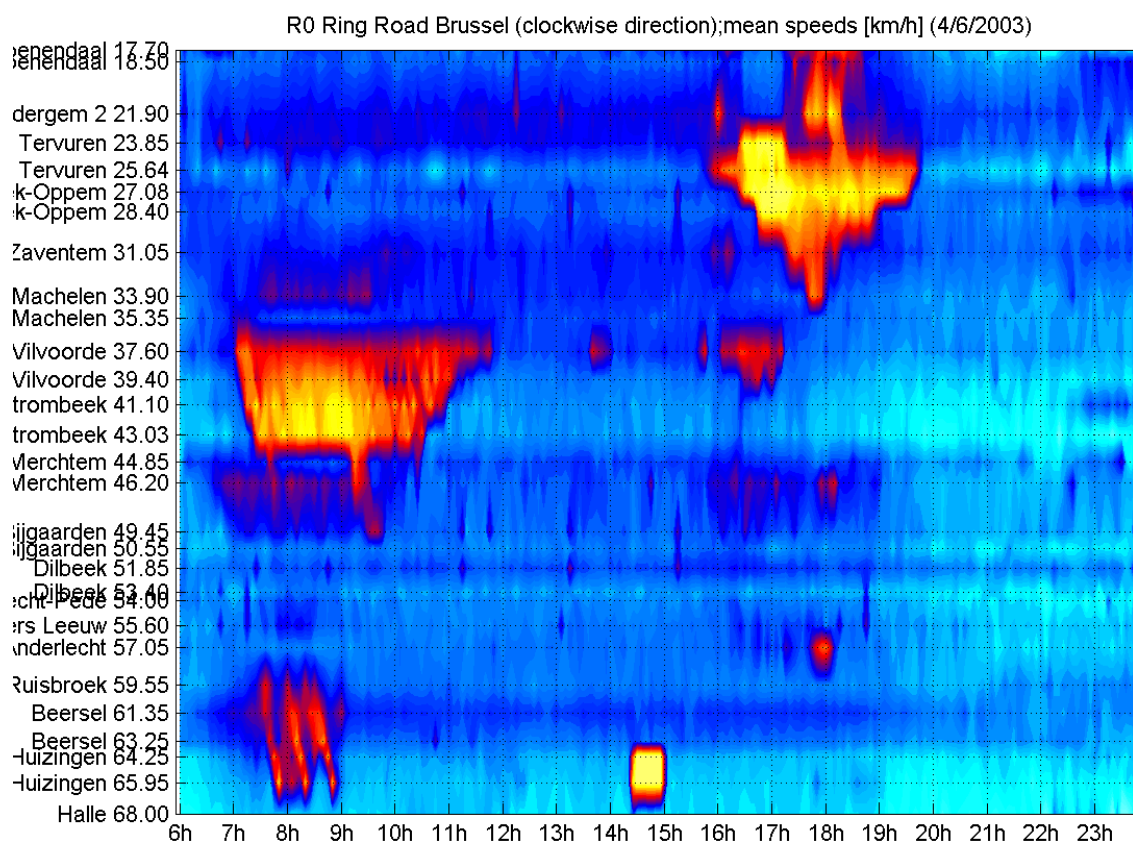
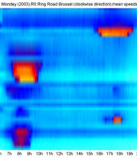
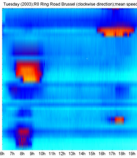
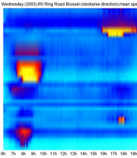
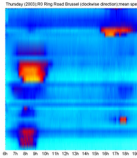
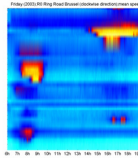
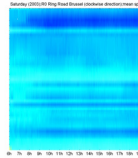
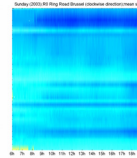
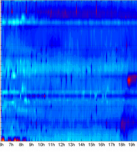
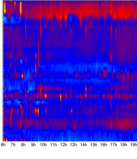
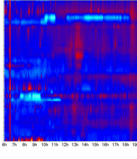
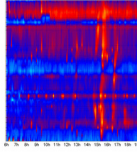
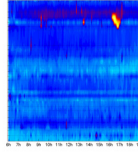
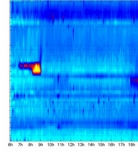
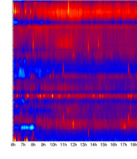
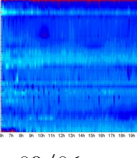
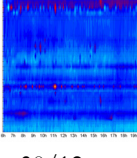
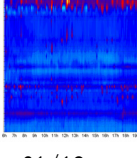
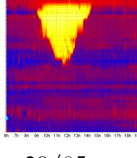
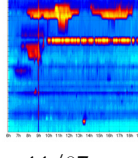
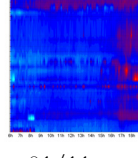
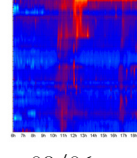
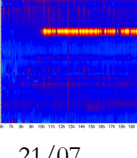
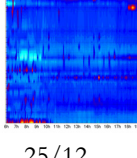
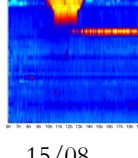
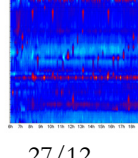
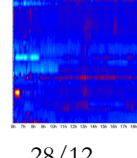
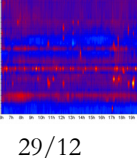
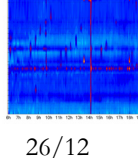
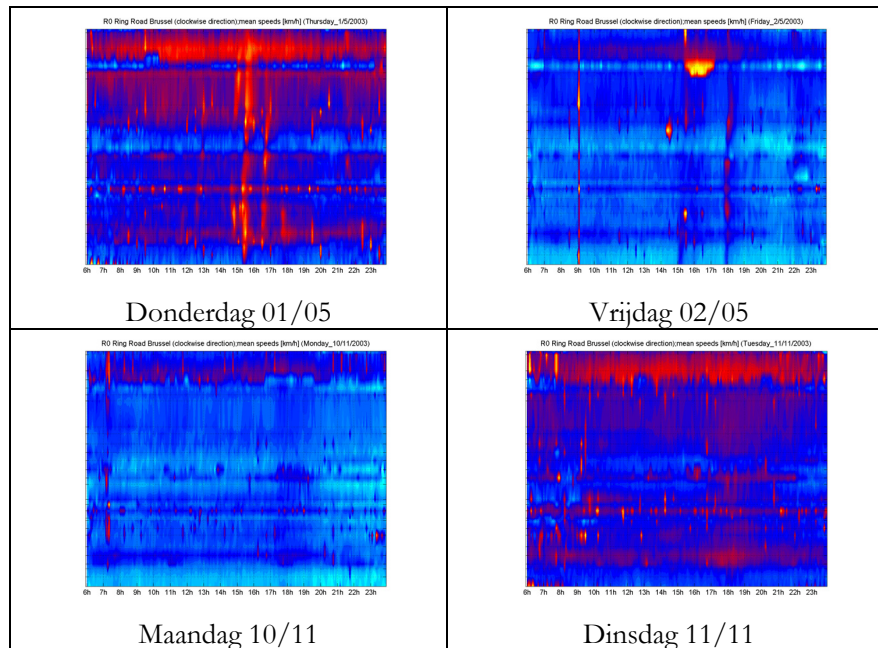


Figure 9: A time-space diagram that shows the mean speed on Monday, 4 June 2003 on the R0 ring around Brussels.

Based on the list of special days, we can now give the time-space diagrams for all these days in chronological order. The results can be seen in the Table below (the upper row each time corresponds to a normal weekday).

<i>Monday</i>	<i>Tuesday</i>	<i>Wednesday</i>	<i>Thursday</i>	<i>Friday</i>	<i>Saturday</i>	<i>Sunday</i>
 <i>normal</i>	 <i>normal</i>	 <i>normal</i>	 <i>normal</i>	 <i>normal</i>	 <i>normal</i>	 <i>normal</i>
 21/04	 11/11	 01/01	 01/05	 30/05	 27/09	 20/04
 09/06	 30/12	 31/12	 29/05	 11/07	 01/11	 08/06
 21/07			 25/12	 15/08	 27/12	 28/12
 29/12				 26/12		

Some special days occur on Tuesdays or Thursdays; it is logical to assume that many people will then take leave on the corresponding Mondays or Fridays. Looking at these days, gives the time-space diagrams in the Table below.



2.2.2.4 Creating typical daily profiles

Given the information from the previous Sections, we can now look at a typical weekday, e.g., a typical Monday and a typical Friday in 2003, as can be seen in Figure 10.

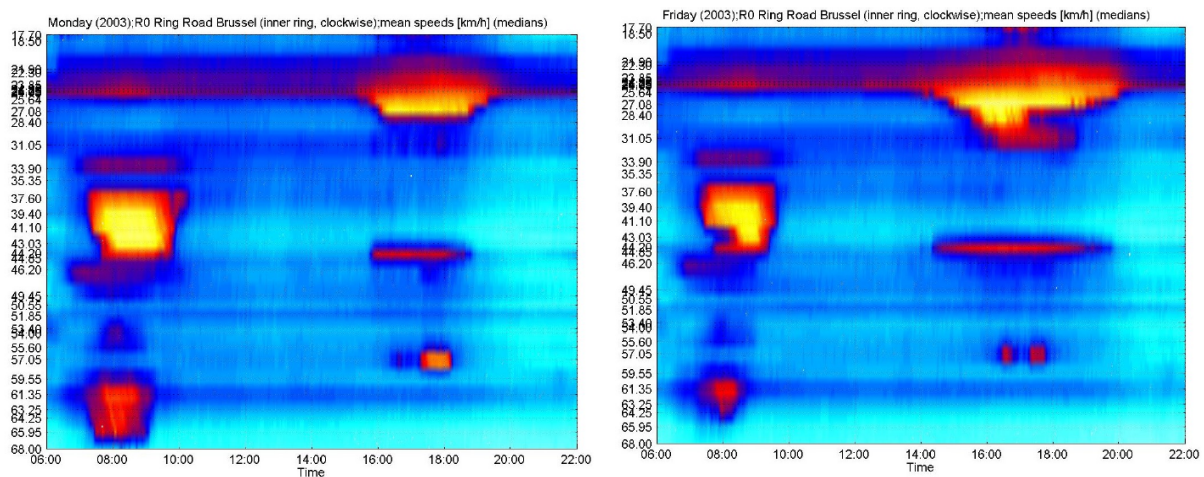


Figure 10: Congestion on the R0 ring around Brussel on a typical Monday (left) and a typical Friday (right) in 2003.

For the definition of these daily profiles we considered the median of the mean speed of all similar weekdays (without the special days). The median gives a good indication of the default traffic pattern, without taking into account exceptionally local circumstances.

Based on such typical daily profiles, we can use them to construct typical time series for the detectors, which can be stored in a historical database and used when reconstructing missing traffic flow measurements.

2.3 Data validation

The traffic model detailed in the traffic management module (Section 3) assumes that all data fed to it is correct. To this end, all the data should be validated which entails some rudimentary checking: can the given data be considered as being ‘correct’, and if not (i.e., data flagged as outliers), with what kind of data do we replace it. With respect to normalisation of the data, we assume this is not necessary, as all data entering the system is supposed to be compliant to the FLEXSYS standard.

2.3.1 Methodology for validation

The detection of incorrect data (i.e., outliers) is done in an on-line fashion, implying that we only have access to the current incoming measurement and its (recent) past, as opposed to off-line outlier detection for which we also can look at future time periods of each measurement.

Our validation is based on a process that comprises two stages:

- (1) Test for physical outliers.
- (2) Test for statistical outliers.

Physical outliers correspond to measurements that are physically impossible (e.g., negative speeds and flows), whereas statistical outliers correspond to measurements that are unexpected given the current dynamics of traffic. Once an incoming measurement is flagged as an outlier, it is reconstructed as best as possible.

2.3.2 Detecting outliers

When performing data validation, we try to identify possible outliers and correct them as best as possible. In the following, we consider an incoming real-time stream of traffic flow measurements (with a constant frequency of one minute).

2.3.2.1 Different kinds of outliers

When considering faulty measurements from detectors, we can in general distinguish between structural failures versus occasional errors. The former can be due to a miscalibration, resulting in consistently faulty data (e.g., over- and underestimations of flows, detectors that get stuck in an on-/off-position, ...). Spotting and correcting these failures is not a difficult task (it requires, e.g., a recalibration), in comparison with the latter class of occasional errors. These can have very different causes, such as detector cross talk, chattering, transmission failures, ... As a result, the detector logic can report incorrect data, for example, values that can easily be spotted are the sentinel values that get stored in the central database due to transmission failures [Mae06b].

Considering these ‘strange values’, we can look at them from a statistical perspective; as such, they are called outliers. From this point of view, “*outliers are observations that appear to be inconsistent with the remainder of the collected data*” according to Iglewicz and Hoaglin [IH93]. The phrase “*being inconsistent with the remainder*” can be given a more mathematical characterisation by taking into account the distributions of the measurements. Values that fall outside these distributions, or those that occur in the tails of them, can then be considered as outliers. Note that from a statistical point of view, outliers are not necessarily bad values, as it is possible that these data points might come from another population/distribution [VH05].

2.3.2.2 Testing against physical outliers

Consider an incoming measurement $z(t)$ at time instant t . This measurement is checked against several thresholds to judge whether or not it is physically sound; if it is found to be invalid, a flag is set. The decision is straightforward:

- If $z(t) < 0$ or $z(t) > z_{\Delta, \max}$ then set **FP = 1**.
- Otherwise, set **FP = 0**.

The threshold $z_{\Delta, \max}$ is defined depending on the nature of the measurement. For flows this can e.g., be equal to 7000 vehicles/hour, for mean speeds it can be set to 150 km/h. Note that in any case, the thresholds should be set by observing the detector output for a while, as the local traffic situation plays an important role.

2.3.2.3 Testing against statistical outliers

For each measurement $z(t)$ at time instant t , we consider the measurement itself, including the recent past. This latter can be modelled as a sliding window W_z with a fixed size N_W . In this way, we can make a distinction between different kinds of presence of outliers, e.g.:

- outliers that occur as single instances (e.g., the left graph in Figure 11)
- and outliers that occur in patches (e.g., the right graph in Figure 11).

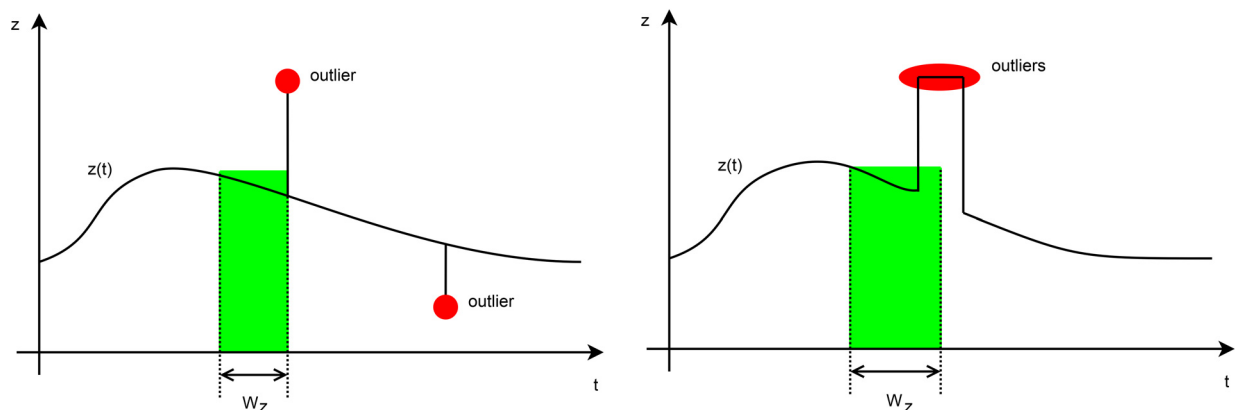


Figure 11: Two types of outlier occurrences: the left graph shows two single, isolated outliers denoted by the red circles, whereas the right graph shows a small patch of outliers as indicated by the red ellipse (note that they do not necessarily have to be consecutive). In each graph, $z(t)$ denotes the measurement at time instant t ; W_z is the width of the sliding window (i.e., the green rectangles) used for detecting outliers based on information from the recent past.

With respect to these patches, we can make a distinction between small and large patches. Outlier detection in a small patches can suffice with simple statistical tests; in order to detect outliers in larger patches, the small amount of information contained in the window is not sufficient, requiring us to resort to more sophisticated methods.

Furthermore, considering the occurrence of outliers in a patch, they can either be a set of isolated outliers, randomly distributed over the window (with the rest of the measurements still intact), or they can be a set of consecutive outliers forming a solid block among the other measurements. The test is devised in such a way that it does not matter how the outliers are temporally distributed over the window.

The goal is now to detect statistical deviations, more specifically so-called ‘*spikes*’ (also known as ‘*detector shot noise*’). The detection is based on the work of Menold et al. and their data-cleansing filter [MPA99].

We assume the measurements are accompanied by additive noise, corresponding to the following model:

$$z(t) = z_*(t) + \xi(t),$$

with $z_*(t)$ the nominal value of a measurement, $\xi(t)$ a deviation and $z(t)$ the measured value, each time at time instant t .

We also consider all measurements from the recent past, i.e., the $N_w - 1$ measurements from the sliding window $W_z(t)$. Outliers are now defined as values that disturb the local pattern of measurements. A possible metric for this is the average. As this is however heavily sensitive to outliers, we instead use a robust estimator such as the median $m(W_z, t)$.

Because we work with the median, up to half of the measurements in the sliding window can be allowed to be ‘faulty data’ and still have a valid detection of outliers. In order to deal with groups of outliers, the window size must be at least twice as large as the size of the largest expected group of outliers. Note that a larger window has the disadvantage that more variation of the local pattern is spread out, resulting in a lower success-rate at detection.

As a distance metric we use the absolute difference between the current measurement $z(t)$ and the median $m(W_z, t)$. If this difference is larger than a certain given threshold $z_{\Delta, \max}$ value, then the measurement is considered an outlier, and the flag **FS = 1**; otherwise, the flag **FS = 0**.

Some remarks

There are some important aspects related to the choice of a suitable threshold:

- (1) The sensitivity of the threshold: this should be considered by looking at the measurements themselves, as each detector is different.

In order to set the threshold, we need to take into account the fact that by using the median as a replacement of outlying measurements, the threshold should not be set too high. A better strategy is to make it varying, relative to the variation that is measured within the window. For this latter, we can take, e.g., the standard deviation of all ‘good’ measurements within the sliding window. To do both simultaneously, we can use the ‘*median absolute deviation*’ (MAD).

- (2) The size N_w of the sliding window $W_z(t)$.

Setting the size of the sliding window is best done on a trial-and-error basis, as each time series has its own specific structural variation (e.g., the jumps in the minute values of the flows).

2.3.3 Reconstructing faulty measurements

Once both tests for physical and statistical outliers have been carried out, we can now consider what to do in case at least one of both flags is set (i.e., **FP** = 1 or **FS** = 1).

The idea is to substitute/replace the outlying measurement with a candidate value. To this end, we have several methods at our disposal:

- We can use the median $m(W_z, t)$ itself.
- As an alternative we can also use the last valid measurement (note that this not necessarily corresponds to $z(t-1)$).
- Yet another possibility is to use a scaled-up value based on a typical dag (constructed for the historical database). This scaling can be done straightforward as follows:

$$z(t) = \frac{z(t-1)}{z_{\text{historical}}(t-1)} z_{\text{historical}}(t),$$

with $z_{\text{historical}}(t)$ the historical measurement obtained from the typical day at time instant t .

3. Traffic management module

The traffic management module consists of two related models. On the one hand there is the traffic model (discussed in Section 3.2) that, based on input data from the detectors, tries to give an estimation of the current state of the traffic flows in the complete network. On the other hand there is the incident detection module (discussed in Section 3.3) that inspects the time series of incoming measurements, as well as measurements from the historical database, and uses them to detect possible incidents.

Before both models are discussed, we present an overview of the work flow in Section 3.1, showing where the data input, traffic state estimation, and operator control take place and how they interact with each other within the FLEXSYS project. We end with Section 3.4, which illustrates the interactions between the traffic management module and an operator in more detail.

3.1 Overview of the work flow

In order to get a good feeling with the operation of the FLEXSYS platform, this Section provides an overview of the work flow. It indicates where measurements enter the system, how they are processed, which modules are involved, and how the information leaves the system. Having a clear view on how this part of the FLEXSYS platform operates is crucial in order to gain a deeper understanding of its usability and possibilities.

In Figure 12, we present a graphical depiction of some of the operational aspects of the FLEXSYS platform, mostly centred around the traffic management module.

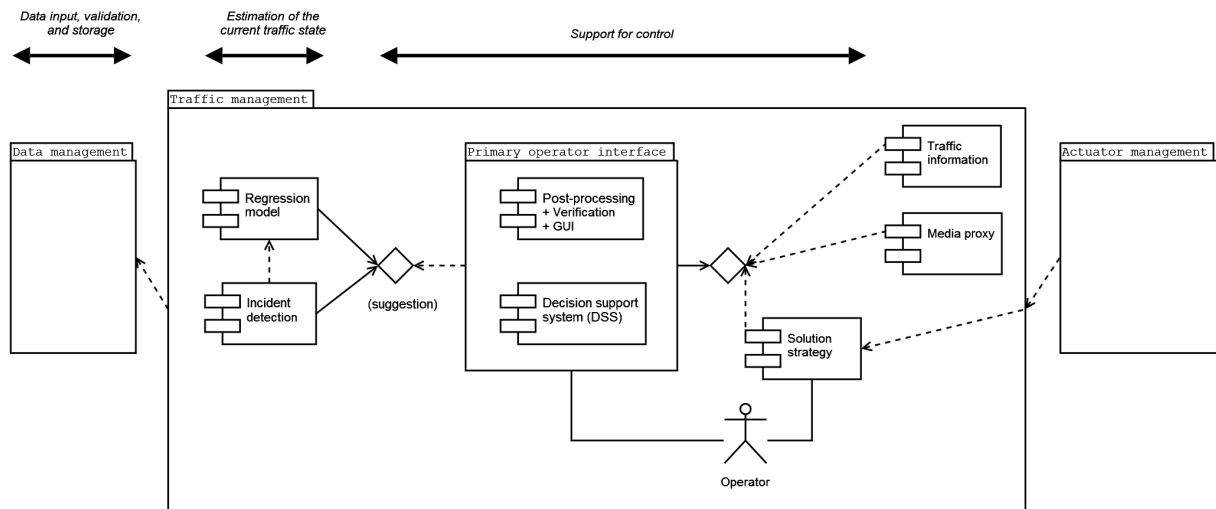


Figure 12: A graphical depiction of some of the operational aspects of the FLEXSYS platform, mostly centred around the traffic management module.

The depicted operation of this part of the FLEXSYS platform can roughly be subdivided into three different, consecutive blocks:

- Data input, validation, and storage:**
 The first block contains the data management module’s aspects. In here, all data stemming from the radars, cameras, et cetera enters the system, where it is validated and stored into a database. Details for this block are given in Section 2.
- Estimation of the current traffic state:**
 The block dealing with the estimation of the traffic state actually consists of two inter-related modules, i.e., the traffic model and the incident detection module. The former is responsible for taking the detectors’ measurements currently obtained from the network, and using them to estimate the current state of traffic flows on all the roads within the network. The latter is coupled in that it can receive information from this traffic model, allowing a better detection rate of incidents occurring within the network. More information on the traffic model is given in Section 3.2; the incident detection module is briefly mentioned in Section 3.3.
- Support for control:**
 The last block entails operator support for controlling the actuators present within the road network. To this end, the operator can rely on the output of the traffic model and the incident detection module. The operator has the availability of a user interface that gives a clear overview of the current (estimated) status of traffic in the network. Details for this operational part of the FLEXSYS project are provided in Section 3.4.

3.2 Traffic model

The core of the FLEXSYS system is given by the traffic model. Note that we do not use the nomenclature of a traffic model; in our view, this latter type of model describes the physical propagation of traffic flows (e.g., microscopic and macroscopic models) [Mae06a]. Putting it concisely, the traffic model merely provides an estimation of the current state in the road network.

The task of the traffic model is to extrapolate information from several points in the road network (i.e., the detectors) to all the links (this for all state variables). As such, the output should be considered as a suggestion for the current traffic state; it can be wrong.

The goal of the traffic model is to give an accurate and complete description of the status of the road network. The model largely splits into two separate parts, i.e., an off-line and an on-line component. Both parts are illustratively depicted in Figure 13; the next two Sections 3.2.1 and 3.2.2 give more details on the operational aspects of both parts.

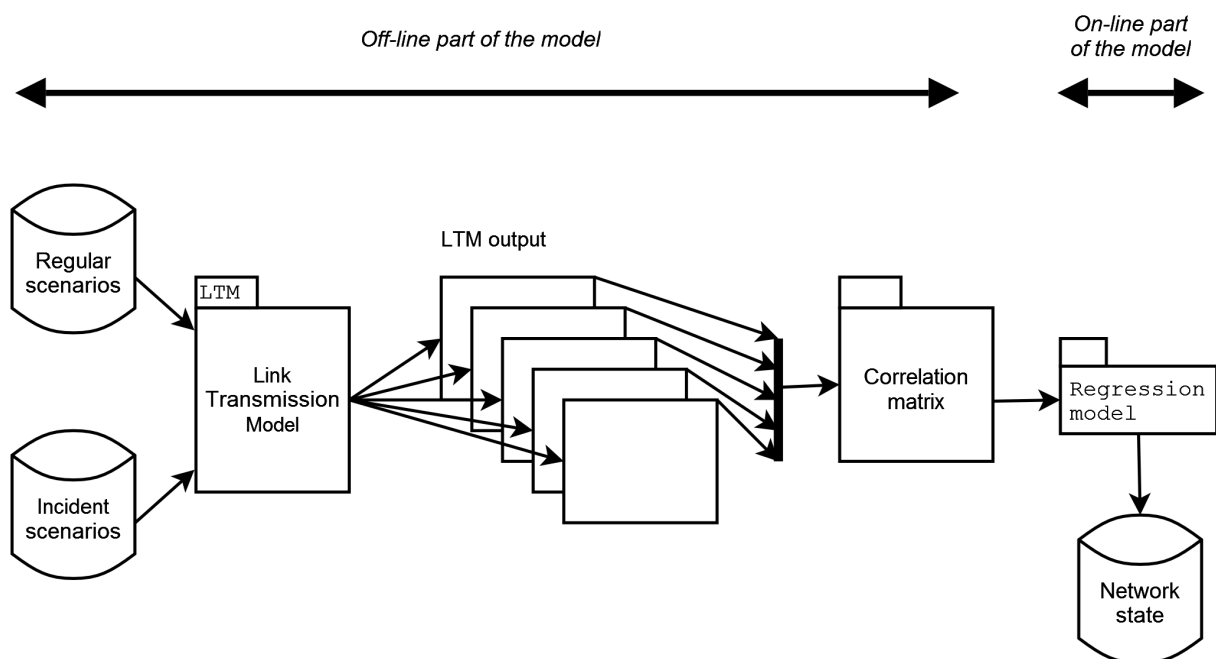


Figure 13: A schematic overview of the off-line and on-line parts that form the context of the traffic model.

3.2.1 Off-line part of the model

- (1) This part of the model first considers all the input data. The basic input data consists of the following information:
 - A network description: all nodes together with their X and Y coordinates, and all links attributed with their capacities, free-flow speeds, and jam densities (see [Mae06a] for a definition of these variables).
 - Origin-destination matrices (static and/or dynamic).
 - Route assignments (through the route choice model).
- (2) Based on this input data, a set of scenarios is constructed. These scenarios define simulation inputfiles that vary according to some predefined user specifications. They encompass two different kinds: regular scenarios and incident scenarios. Examples of the former are changing the capacities of some specific or random links, changing some origin-destination relations, ... For the latter, we can choose for, e.g., lane closures, strongly reduced outflows, ...
- (3) For each of the scenarios previously defined, the link transmission model (LTM) is executed (see [Ype07] for an overview of this model). This model is actually a computational efficient traffic flow propagation model, based on a macroscopic first-order hydrodynamic model. Within the context of dynamic traffic assignment, the LTM model forms the dynamic network loading (DNL) component [Mae06a]. It gives a realistic description of the propagation and dissipation of queues.
- (4) When the LTM executions have finished, the results are given in the form of a set of output files that describe the traffic conditions on all the links in the road network. The output is then used to construct a correlation matrix that contains the correlations between flows and travel times on all pairs of links.
- (5) Before the traffic model is executed, there is also a step involved that calculates the optimal detector positions in the road network; from these optimal positions, the road network can be divided into distinct zones (see Figure 14 for an example of such a division).

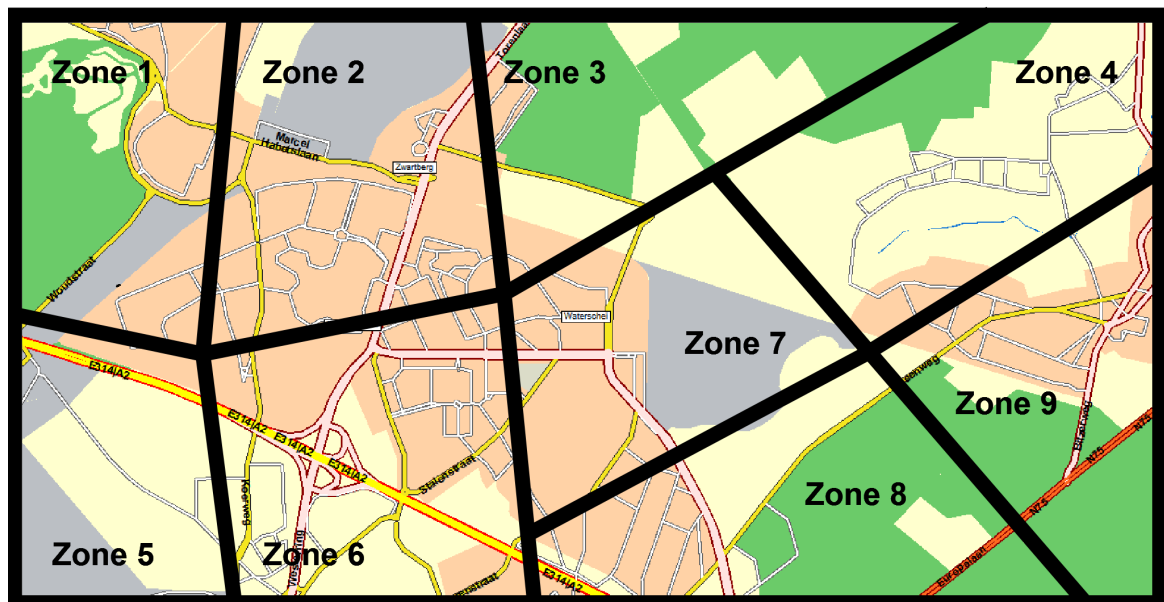
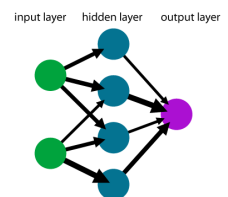




Figure 14: *Top*: an example of the division of the road network into several distinct zones, each containing its own set of links. *Bottom*: a satellite image of the location is the vicinity the ‘Ei’ of the city of Kortrijk.

In this description, we assume the detector positions and zones are given. As such, the traffic model takes as inputs both the correlation matrix and the current detector positions. This leads to the construction of a series of artificial neural networks³ (ANNs), i.e., one for each zone in the road network. The ANNs are trained based on the information contained in the output of the LTM model (which itself is based on the regular and incident scenarios). These ANNs actually comprise the traffic model.

³ An artificial neural network (ANN) can be seen as a non-linear statistical data modelling tool, which models complex relationships between inputs and outputs (e.g., for clustering purposes). It is very suitable for finding patterns in data structures. The ANN itself consists of an input layer, an output layer, and one or more so-called hidden layers that perform the actual computations. Note that before an ANN can be used, it has to be ‘trained’. This training defines the weights that get assigned to each of the nodes in the layers.



3.2.2 On-line part of the model

Based on the incoming measurements from all the detectors deployed throughout the network, the on-line part of the traffic model then uses the ANNs to get an estimation of the current prevailing traffic conditions everywhere in the road network. This implies that, based on the correlations derived earlier, the on-line part of the model estimates the flows and travel times on all the links in the road network, time and again for each time step.

Note that if any changes occur in the definition of the underlying road network, or changes in the detector positions, ... than the off-line part of the model has to be recalibrated (including a retraining of the ANNs). During the deployment phase, the traffic model is constructed off-line, with its resulting executable deployed on-line; it interacts with the incoming and outgoing data streams by means of configuration files (Section 3.5 provides an overview of the necessary information).

3.3 Incident detection module

Based on the time series of incoming measurements, as well as measurements from the historical database, the incident detection module tries to detect patterns that signal the possible occurrence of an incident (e.g., sudden drops in the speed, a large increase in the occupancy, or a difference between a section's in- and outflow measurements). Another source of information stems from the video images that get relayed from Traficon's 'incident detection' cameras. Other sources of information are notifications from outside the system (e.g., from a local observer of the traffic situation); they enter the system through telephone calls or other methods of communication. For more information, we refer to the work of Tritel in this respect.

Note that in the current setup, information flows from the traffic model to the incident detection module, as opposed to a bidirectional setup, as depicted in Figure 15 in which the black arrows indicate the flow of information. Here we can see that the output from the traffic model is considered by the incident detection module, and that the database with incident data is not considered by the traffic model.

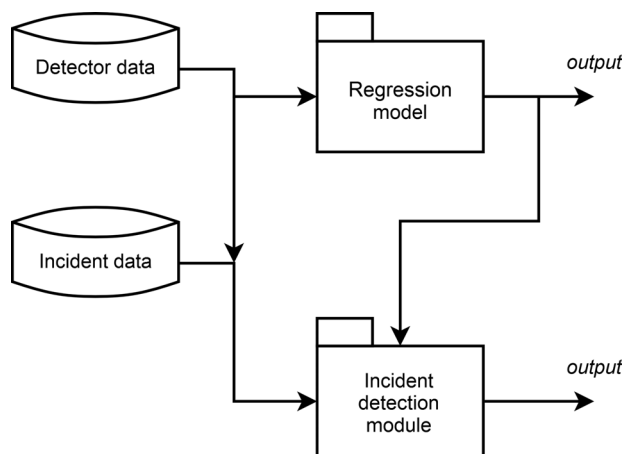


Figure 15: A summarised overview of the interaction between the traffic model and the incident detection module.

In this setup, the idea is that when an incident is detected, a flag is set and shown in the operator's interface.

3.4 Operator's interaction with the traffic management module

As explained earlier in the beginning of Section 3.2, the resulting output from the traffic model and the incident detection module are suggestions, proposals, implying they can be completely wrong. This means that some form of verification remains necessary. By this we mean for example a post-processing step by means of an operator. To this end, we provide the functionality, e.g., user specifications which state that the system can run stand-alone, or with interventions. The operator can then look at the situation, either in situ or by means of a video image or some other method of communication (this to get verified notifications of reported incidents).

As already hinted at in Section 3.1 and Figure 12, the operator can interact with the traffic management module in two ways: first, he can consult the graphical user interface (GUI) and its accompanying rudimentary decision support system (DSS), and second he can interact with the solution strategy module that provides the actuators with their correct settings.

The output from the traffic model is given to the operator as a table containing information on all the links in the road network. This table is visualised on a map containing the geographical area with the road network and all its details.

Besides this standard information, the traffic model also provides the operator with an indication of reliability of the calculations for a complete zone (i.e., how good does the model think its own output is?). These reliability indicators are categorised according to four different types:

- **+** (reliable estimation),
- **0** (neutral),
- **-** (not reliable),
- **N/A** (no output was possible, e.g, because no detector measurements are available).

Once the operator has chosen certain control settings, the output of the FLEXSYS system no longer considers these reliability indicators (i.e., they are stripped away from the output as they only serve as an internal indication towards the operator). If a line of measurements is declined in this post-processing stage, then all its values are set to N/A (i.e., no zeros or other special sentinel values, as they are discarded anyway).

In the GUI, the road network is shown; at this point, the different zones can be coloured according to, e.g.:

- a certain indicator that captures, e.g., the amount of congestion within the region,
- or the reliability indicator that the traffic model assigned to its estimation of the traffic state in that region (e.g., green for +, yellow for 0, red for -, and grey for N/A); see Figure 16 for an example of this colouring scheme.

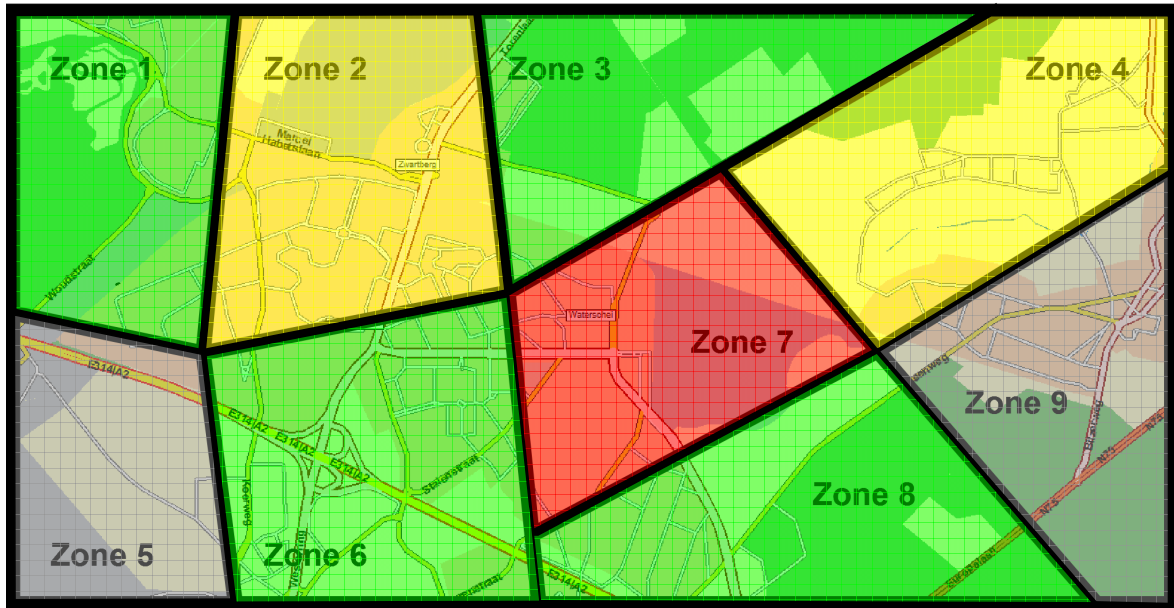


Figure 16: An example of colouring the zones according to the reliability given by the traffic model. The estimations in zones 1, 3, 6, and 8 are trusted by the model (green), those in zones 2 and 4 are less trusted (yellow), the estimation in zone 7 is not trusted (red), and for zones 5 and 9 no estimation could be made (grey).

The operator can inspect and edit detailed information relating to zones, as well as for individual links (for example, by means of popup windows):

- For individual links, the GUI can show the time series of the individual measurements (e.g., average speeds as shown in the left part of Figure 17). Besides that, the GUI can also show the location of detected/suspected incidents, the position and settings of the actuators in the road network, ...
- With respect to zones, the GUI can show a sort of global indicator of congestion within the region. An example of this is information based on historical data, e.g., the severity of congestion as shown in the right part of Figure 17.

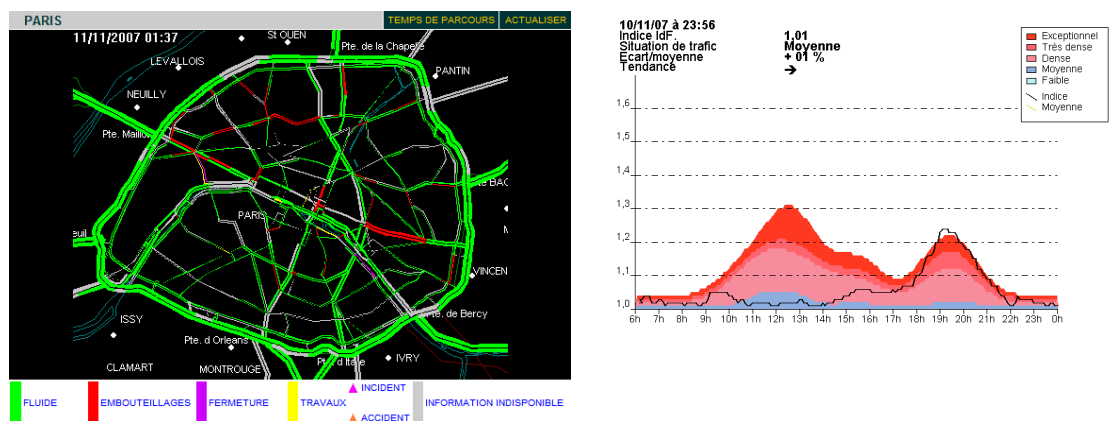


Figure 17: An overview of congestion within a region/zone. *Left*: the Paris area with links coloured according to the average speed. *Right*: a global indication of congestion in the Île de France, relative to the observed historical patterns. Images taken from the Sytadin system (<http://www.sytadin.tm.fr/>).

Based on the information shown, an operator can choose to distrust the traffic model's estimation for a certain zone. To this end, he disables its output by selecting the zone from the GUI and flicking a switch in the popup window for that zone.

Besides the information contained in the GUI, we foresee that the operator can also be supported by means of a small decision support system⁴ (DSS). This provides him with suggestions, at which point he chooses to go along with the DSS proposition, or choose to entirely discard the DSS's advice. Taking the operator out of the control loop, it is also possible to let the DSS run in a stand-alone fashion.

3.5 Data specifications for the traffic management module

The traffic management module on the one hand expects information with respect to the road network geometry, traffic demand, detector locations, and zone definitions (see Section 3.5.1), and on the other hand sends the relevant information to the GUI (see Section 3.5.2).

3.5.1 Input to the traffic model

The input to the traffic model falls into two distinct aspects: on the one hand there is the road network description and the traffic demand that is fed directly to the LTM model (see Section 3.5.1.1), on the other hand there is the input with respect to the locations of detectors and zone definitions (see Section 3.5.1.2).

3.5.1.1 Road network description and traffic demand

The input for the off-line part of the traffic model consists of a file that is fed to the link transmission model (LTM); the structure is taken from the specifications document [Ble05] and the Cube→LTM converter methodology [Mae09].

The input file is based on keywords; these are the following ones:

- nodes=
- links=
- movements=
- lanes=
- signals=
- timeslices=
- routes=
- odmatrix=

These keywords have to be mentioned in the same order as indicated. When the software detects one of these keywords, all the following lines are expected to have a certain syntax. Schematically, this syntax goes like this:

- after nodes=
nodeID, xCoordinate, yCoordinate, type

All input is separated by commas. Different nodes are to be written on different lines and node IDs should succeed each other.

⁴ A decision support system (DSS) can be considered as a system that presents a choice between alternatives, based on the values of certain input parameters.

- after links=

linkID, fromNodeID, toNodeID, jamDensity, capacity, freeFlowSpeed, length

where from node ID and to node ID should refer to the IDs you used after the nodes= keyword. One of the two is allowed to be 0. If so, the link will be treated as an origin or destination link, respectively.

- after movements=

**nodeID, localMovementID (node-related), incomingLinkID,
outgoingLinkID, conflictingHigherRankedPriorities (node-related)**

- after lanes=

**nodeID, link ID, localLaneID (node-related), local movement ID(s) (node-related),
capacity**

- after signals=

nodeID, phaseID, phaseTime(in seconds), laneIDsWithGreen

- after timeslices=

0, endTimeOfSlice1, endTimeOfSlice1, ..., endTime

If you want to simulate a time-varying OD matrix, you need to give in the length of the different time slices. Notice that this line of input should always start with 0.

- after routes=

routeID, originID; linkID; linkID; ...; destinationID, fraction1; fraction2; ...

where each route consists of a set of successive links (successive in path, not in ID or something). origin ID is the ID of a link with an empty begin node and destination ID the ID of a link with an empty end node. The frags at the end represent the fraction of the demand between the origin and destination link that uses this route. There should be as much of these fractions as there are time slices (one fraction per time slice, thus per OD matrix, thus per OD relation).

A small feature of the software: if you do not fill in lines with this syntax, but only “Floyd”, then the program calculates the shortest paths between every origin and every destination (thus 1 route per OD pair).

- after odmatrix=

$OD(11)T_1, OD(12)T_1, \dots, OD(1n)T_1$
 \dots
 $OD(m1)T_1, OD(m2)T_1, \dots, OD(mn)T_1$
 $OD(11)T_p, OD(12)T_p, \dots, OD(1n)T_p$
 \dots
 $OD(m1)T_p, OD(m2)T_p, \dots, OD(mn)T_p$

where n is the amount of destinations, m is the amount of origins and p is the amount of time slices. The first origin link considered in the OD matrices is the first origin link that was inserted in the links= section of the input file and idem for the destination links. $O(i)T_k$ is the demand between the i -th inserted origin link and the j -th inserted destination link in time slice p . The matrices of the different time slices should be cumulative ! If $OD(i)T_1=10$ and $OD(i)T_2=30$, then there are 20 vehicles that want to travel from origin link i to destination link j during the second time slice.

Extra necessary information:

For all the intersections in the road network, the LTM model expects node delays. These imply the following necessary information:

- For setting up lanes for every incoming direction, we need to have the possible outflow directions for each lane, the capacities, and the correct right-of-passage rules (priority roads, or right has priority).
- In case of traffic lights controlling the intersections, we also need the phases and their durations.

Some more comments:

- Spaces or blank lines are of no importance, as long as all info that belongs together (e.g., ID, begin node, end node, jam density, capacity flow and free-flow speed of a link) is inserted on the same line.
- A line is ignored if it is preceded by // . This way, comments can be added.
- All data should have same units: if free-flow speeds are indicated as kilometre per hour, then x- and y-coordinates should be in kilometres, time slices in hours and so on.

3.5.1.2 Input with detector locations and zone definitions

For all the zones in the road network, we need to know which links they are comprised of, i.e.,

Zone1: Link1, Link2, ..., Link5
Zone2: Link6, Link7, ..., Link9

Or more formally:

zoneID, linkID, linkID, ..., linkID

We also need to know the detectors' locations in each zone (note that this information is only available at the end of the off-line construction of the traffic model):

Zone1 *Detector 1* *Link1* *Position*
 Detector 2 *Link3* *Position*
 \dots
 Detector 5 *Link5* *Position*

Zone2 *Detector6**Link7* *Position*

```

Detector 7      Link9  Position
...
Detector 9      Link6  Position
    
```

Or more formally for all detectors in all zones:

zoneID, detectorID, linkID, positionFromStartOfLink

When measurements are entering the system, the traffic model expects the following information in a flat text-file:

```

Zone1  Detector1Flow  AverageSpeed  Occupancy
Zone1  Detector2Flow  AverageSpeed  Occupancy
...
Zone2  Detector5Flow  AverageSpeed  Occupancy
    
```

Or more formally for all detectors in all zones:

zoneID, detectorID, flowValue, averageSpeedValue, occupancyValue

As such, there are (number of detectors * number of measurements per detector) measurements recorded each minute.

3.5.2 Output of the traffic model to the GUI

For all the links in the road network, the traffic model gives at each minute all the information, resulting in the following output table:

```

Zone1  Link1  TotalFlow  AverageSpeed  AverageTravelTime  Flow/Capacity  Active
Zone1  Link2  TotalFlow  AverageSpeed  AverageTravelTime  Flow/Capacity  Active
...
Zone1  Link6  TotalFlow  AverageSpeed  AverageTravelTime  Flow/Capacity  Active
Zone2  Link7  TotalFlow  AverageSpeed  AverageTravelTime  Flow/Capacity  Active
Zone2  Link8  TotalFlow  AverageSpeed  AverageTravelTime  Flow/Capacity  Active
...
Zone2  Link12 TotalFlow  AverageSpeed  AverageTravelTime  Flow/Capacity  Active
    
```

Or more formally:

zoneID, linkID, totalFlow, averageSpeed, averageTravelTime, flowCapacityRatio, isActive

Information from links is aggregated for the different zones that contain them; this entails average speeds, average journey times, total flows, ... This is incorporated into the GUI, but can also be made publicly available as an outgoing information stream.

```

Zone1  TotalFlow  AverageTravelTime  AverageTravelTime/FreeFlowTravelTime  Reliab.
Zone2  TotalFlow  AverageTravelTime  AverageTravelTime/FreeFlowTravelTime  Reliab.
...
Zone5  TotalFlow  AverageTravelTime  AverageTravelTime/FreeFlowTravelTime  Reliab.
    
```

Or more formally:

zoneID, totalFlow, averageTravelTime, averageTTFreeFlowTTRatio, reliabilityIndex

The reliability index of the suggestion made by the traffic model can be +, 0, -, or N/A, as explained in Section 3.4.

The same output format is used for disseminating information from the FLEXSYS system to third parties (e.g., the media, traffic centres, ...). But note that, as mentioned before, if a line of measurements is declined in the post-processing stage, then all its values are set to N/A (i.e., no zeros or other special sentinel values, as they are discarded anyway); this holds true for all the links, as well as all the zones.

References

- [BCK03] Bass Len, Clements Paul, and Kazman Rik, **Software Architecture in Practice**, 2nd edition, Addison-Wesley Professional, ISBN 0321154959, 2003.
- [Ble05] Art Bleukx, **Program Specifications LTM**, Technical report, Katholieke Universiteit Leuven, Department of Civil Engineering, 28 June, 2005.
- [Eic] C. Eick, N. Zeidat, and Z. Zhao, **Supervised Clustering – Algorithms and Benefits**, Department of Computer Science, University of Houston.
- [HM07] Alexander Helleboogh and Sam Michiels, **FLEXSYS Deliverable D2.1/2 Architecture Study**, version 1.0, October 2007.
- [IH93] B. Iglewicz and D.C. Hoaglin, **How to Detect and Handle Outliers**, American Society for Quality Control, Milwaukee, Wisconsin, 1993.
- [Mae06a] Sven Maerivoet, **The Physics of Road Traffic and Transportation**, in Modelling Traffic on Motorways, PhD dissertation, Chapters 2 and 3, Katholieke Universiteit Leuven, June 2006.
- [Mae06b] Sven Maerivoet, **Data Quality, Travel Time Estimation, and Reliability**, in Modelling Traffic on Motorways, PhD dissertation, Chapter 6, Katholieke Universiteit Leuven, June 2006.
- [Mae09] Sven Maerivoet, **Cube – LTM Intersection Converter Methodologie**, FLEXSYS Deliverable 2.13, January 2009.
- [MPA99] P.H. Menold, R.K. Pearson, and F. Allgöwer, **Online outlier detection and removal**, in Proceedings of the 7th Mediterranean Conference on Control and Automation (MED99), Haifa, Israel, June 28—30, 1999.
- [VH05] Sabine Verboven and Mia Hubert, **LIBRA: a MATLAB Library for Robust Analysis**, Chemometrics and Intelligent Laboratory Systems, Volume 75, Pages 127—136, 2005.
- [Wuy05] Roel Wuyts, **UML: History and Overview**, INFO025: Analyse et Méthodologie Informatiques, Université Libre de Bruxelles (ULB), 2005.
- [Xu05] R. Xu and D. Wunsch, **Survey of Clustering Algorithms**, IEEE Transactions on Neural Networks, Volume 16, Nr. 3, Pages 645—678, May 2005.
- [Ype07] Isaak Yperman, **The Link Transmission Model for Dynamic Network Loading**, PhD dissertation, Katholieke Universiteit Leuven, Department of Civil Engineering, June, 2007.