Modelling Collaborative Systems and Automated Negotiations

Nico Plat[1], Peter Gorm Larsen[2], and Ken Pierce[3]

[1] West IT Solutions, The Netherlands
[2] Aarhus University, Department of Engineering, Denmark
[3] Newcastle University, School of Computing Science, UK

Abstract. Collaboration between different Traffic Management Systems (TMSs) is virtually non-existent. A TMS is a cyber-physical system composed of software and hardware systems under the control of a road authority. Existing road infrastructure is not used in an optimal way, because each road authority tries to achieve its own local goals: minimal congestion and pollution. The performance of the network as a whole is not considered, nor are the interests of individual road users, who travel across boundaries controlled by different TMSs. The TMSs involved do not attempt to help one another to get a better overall result. In this paper we show that collaboration between TMSs can be introduced, by modelling a system of automated negotiations using the Overture toolset. A demonstrator has been made which consists of a VDM model of two TMSs that interacts with a simple traffic simulator. The VDM model receives traffic information from the simulator, then the TMSs negotiate on which traffic control measures to use. The chosen measures are communicated back to the simulator, allowing the effect of the traffic measures to be seen.

Keywords: cyber-physical systems, Overture, VDM, traffic management, automated negotiation, collaborative architectures

1 Introduction

Many countries have dense road networks and significant traffic problems. The flow of traffic on roads is managed by a series of Traffic Management Systems (TMSs), owned and controlled by various local and national authorities. A TMS consists of a collection of distributed systems and devices, usually installed along the roadside, and control logic that decides how these devices operate. Devices can be sensors that collect traffic data, such as cameras, radar detection systems, induction loops, and actuators that give instructions to road users via signs and signals. Current TMS architectures are usually controlled centrally from regional control centres. A low degree of collaboration hinders efficient management of traffic problems that straddle boundaries between authorities, because TMSs cannot communicate between regions and may have competing goals for traffic flow. While cooperation between various road authorities from a governance point of view has improved recently, a technical infrastructure (including the application software) that facilitates collaboration between TMSs, is not yet in place.

The TEMPO[4] project addressed the problem of disconnected TMSs by giving TMS providers collaborative control architectures that engage with each other in automated negotiations. Negotiations are based on policies, which help to reach agreement on which control measures are beneficial for the system as a whole, improving overall network performance.

TEMPO used VDM++ and Overture [6] to perform an initial analysis of both existing traffic management networks and potential collaborative designs, demonstrating the benefits of smart traffic systems. VDM was chosen as way to converge quickly on the core logic in the traffic management domain with limited time and resources. The initial models produced illustrate the core concepts and the possibilities enabled by allowing TMSs to collaborate. It is possible to extend this further to achieve more realistic simulations, for example using either the Crescendo[5] [4] or INTO-CPS[6] technologies [3] to connect the TMS models in VDM to a dedicated traffic simulator. In such extensions it would be necessary to devise a proper protocol for use in negotiations between TMSs. Traffic simulations produce a large amount of numerical data and it is imperative that this data is presented in an understandable way to non-experts, in order to make computer models useful. Overture has been extended with a 2D/3D visualisation library[7] to illustrate the negotiations between TMSs and the overall effect on traffic flow [2].

This paper focuses on the central aspect of the model: the negotiation mechanism used as the basis for collaboration. We start by giving a short introduction in the field of traffic management in Section 2, using the same concepts that will be used in the model itself, in this way making the model more comprehensible. After that a brief overview of the TEMPO project is provided in Section 3. We then explain the idea behind the negotiation process in Section 4, and how this is transformed into a VDM model in Section 5. Afterwards we present initial results of the performance of the model, compared to the original situation, in which no collaboration is facilitated in Section 6. Finally, we conclude the paper and suggest future work in Section 7.

2 Traffic Management

Traffic management involves planning, coordinating, controlling, and organising traffic to achieve efficient and effective use of the existing road capacity. The way traffic management is implemented varies in degree of sophistication, however in all cases the goals are mostly the same:

- Enhancing or optimising the throughput of the network, in order to decrease travellers' journey times;
- Sustainability, ensuring that environmental policies are met;

[4] TEMPO is an acronym for "TMS Experiment with Mobility in the Physical world using Overture". See `http://tempoproject.eu/` for more information.

[5] See `http://crescendotool.org/`.

[6] See `http://into-cps.github.io/`.

[7] The visualisation library is explained further in the latest version of the Overture user manual [8].

– Keeping travellers informed, making sure that travellers get reliable traffic information with respect to their expected journey time and incidents or accidents on the road.

These goals must be achieved with vehicle safety as a vital constraint: making journeys as safe as possible is always a first priority.

In this paper we are only interested in *operational* traffic management[8], i.e. responding to the current traffic situation on (part of) the road network, deciding if the situation requires intervention, and then deploying the instruments available to influence traffic streams or individual vehicles on the network.

To do this one needs:

– *Access to real time traffic data*. Typically this is done using sensors at the roadside or elsewhere. Examples of sensors include induction loops, which measure the speed and density of passing traffic, cameras, and radar systems. Countries like The Netherlands have central repositories in which (soft) real-time[9] traffic data is available from a variety of sources, covering almost the entire Dutch road network[10].
– *Processes for decision making*, to determine whether or not actions need to be taken to influence, steer or guide the traffic streams on the network depending on the current or predicted traffic situation. Decision making can either be done in a Traffic Control Centre (TCC) or along the roadside (local control).
– *Instruments to take action*, to influence traffic using, for example, roadside equipment. At a logical level, we call these *control measures*. Examples include setting up diversions, opening or closing hard shoulders to regular traffic, altering traffic light timings, imposing temporary speed limits and ramp metering (controlling the amount of traffic flowing into an area). At a physical level, such control measures are called actuators. Examples include signs and signals displayed along the roadside or in-car.

A road network will have parts controlled by different TMSs under different ownership. As such, the group of TMSs can be characterised as a System of Systems [10, 11]: it is a system composed of several constituent systems that execute individual control measures on their part of the network, but cannot individually guarantee properties of the whole network without working together. A TMS is also a Cyber-Physical System (CPS) [9], because it consists of distributed, networked physical and software (cyber) components.

Traffic management is traditionally executed by road authorities, where national authorities are typically responsible for a country's motorway network and interurban traffic, and municipalities are responsible for their own urban networks. Other organisations can also influence traffic, and hence act as de facto TMS providers. These include

[8] In addition to operational traffic management we also distinguish tactical traffic management and strategic traffic management. Tactical traffic management addresses planning of resources to be used during operational traffic management and other tactical issues, and strategic traffic management concerns issues related to policy making.

[9] The lowest granularity currently available covers a time interval of one minute, which is suitable for traffic management but not for safety measures.

[10] National Data Warehouse (NDW) for Traffic Information, see `http://www.ndw.nu/en/`.

car manufacturers and manufacturers of navigation equipment, airports and harbours that deal with logistics, and even event organisers that attract large volumes of multi-modal traffic at around the time their events take place.

The means that these parties have to influence traffic vary. In some cases, such as manufacturers of navigation equipment, this is limited to the distribution of traffic information, while official authorities can directly influence traffic through physical measures. Yet all parties can influence traffic on potentially overlapping parts of the road network or across neighbouring parts, for example ramps between urban and motorway networks. Since each party may have individual goals that are conflicting, the final results for individual travellers may not be optimal.

3 The TEMPO project

The TEMPO project was a small scale experiment, in which we attempted to overcome the conflicting interests of TMSs by engaging them in a negotiation process to reach a goal that is acceptable to all parties involved. The concept of "cost" (price) is introduced here to compensate when one TMS needs to give in when global goals get priority over its own goals.

In order to do this we developed a model of the system that treats the road network as a graph, where vehicles move along the edges of the graph in a distributed (but predefined) fashion. The complete model of the system consists of two parts coupled together using the remote access features to the Overture toolset [12]:

– A VDM model that specifies the business logic of multiple TMSs, engaging in a negotiation process to determine which traffic control measures will be taken. We also refer to this model as the "TEMPO engine".
– A traffic simulator written in Java[11], which feeds the TEMPO engine with "traffic situations". Based on the traffic situation, the engine executes calculations (with or without negotiations) resulting in a series of traffic control measures to be applied. These are communicated back to the traffic simulator, which takes the measures into account, performs another round of traffic processing, provides a new traffic situation to the model, and so forth. The simulator provides a simple events language, which allows the user to specify traffic volume for an edge (e.g. simulating rush hours), traffic accidents, and scheduled events such as bridge openings.

The global architecture of the demonstrator is presented in Figure 1.

4 Automated Negotiation

The concept of a "negotiation process" is at the very heart of the TEMPO engine. The idea of working with an automated negotiation process is in itself not new. Beam and Segev published a survey [1] in the mid 90's on the state of the art with respect to automated negotiation. Jennings et al. did the same around the year 2000 [5]. Negotiation

[11] Java was chosen as the implementation language for the simulator for practical reasons. The simulator itself is not in scope of our analysis.

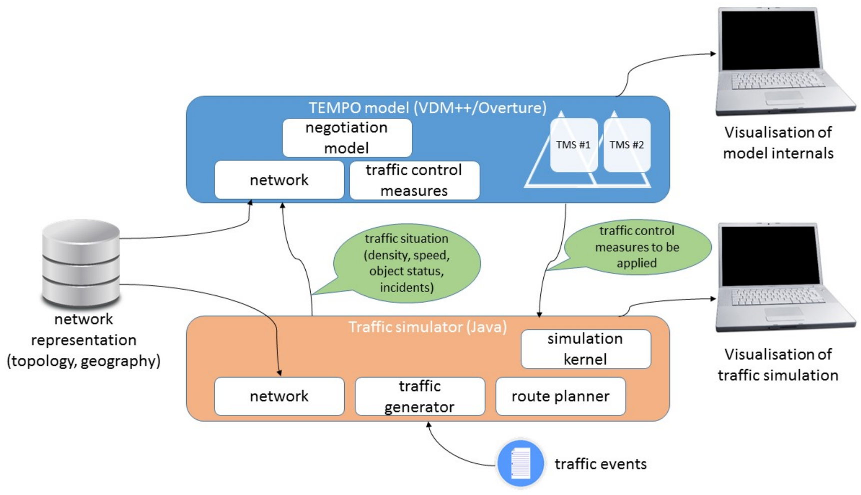
TEMPO model (VDM++/Overture)
negotiation model
network
traffic control measures
TMS #1
TMS #2
Visualisation of model internals
network representation (topology, geography)
traffic situation (density, speed, object status, incidents)
traffic control measures to be applied
Traffic simulator (Java)
simulation kernel
network
traffic generator
route planner
Visualisation of traffic simulation
traffic events

Fig. 1: Architecture of the TEMPO demonstrator

in (electronic commerce) is defined as the process by which two or more parties bargain for mutual intended gain, using the tools and techniques of electronic commerce. Completely automatic negotiations are usually based on intelligent agents. Ontologies (categorising the objects of negotiation such that they are semantically meaningful to software or human agents), strategies for negotiation, negotiation protocols, and decision making models were important research issues at the time. Negotiation support systems are available to help human negotiators make better decisions and negotiate more productively. Online auctions and marketplaces are typical examples of application areas.

The automated negotiation process in TEMPO conceptually uses a rather simple strategy compared to other existing automated negotiation systems[12], although enhancements to make it more powerful are rather straightforward, given the current setup. In that sense, the TEMPO architecture can be regarded as a framework for automated negotiations. Furthermore, automated negotiation as part of a traffic management process is, to our best knowledge, completely new.

The first step in the entire process is not really part of the negotiation itself: it consists of determining whether or not the TMS can "help itself". In order to do this the TMS assesses the traffic situation on all the edges of the network (typically a subnetwork of the overall network) that it controls. A traffic situation typically comprises density and average speeds of the vehicles on the edge, whether or not an incident occurred, and whether or not objects on or close to the network may hinder the traffic (in TEMPO we use the bridges that can be open or closed). Future versions of the traffic

[12] The specification of the negotiation process is less simple though, because of the specific details of the traffic management domain.

situation may include variables like e.g. air pollution which then also can be used as a steering mechanism.

If the TMS has traffic control measures available on a given edge of the network, and if the traffic situation for that edge is such (e.g. a high traffic density, which is a sign of congestion) that measures need to be taken, then that edge is marked as a "problematic edge", and a trigger to activate the measure is generated that will later on be passed on to the simulator.

Then the real negotiation process starts. It consists of the following consecutive steps:

- For each problematic edge a request for help is issues which formulates the service it would like to get from adjacent edges. This service can be either "decrease output" (from an edge in another TMS that is an input edge to the problematic edge), or "increase input" (from an edge in another TMS that is an input edge to the problematic edge.
- Then each TMS checks if it can fulfil the request. It can do this if it has a traffic control measure available that implements the required service (e.g. a ramp metering system is an example of a traffic control measure that implements the service "decrease output"). If it has, it calculates an associated cost which is determined from the severity of the request and a utility function. The utility function values the situation of the edge providing the service, in combination with the priority associated with it: a higher priority increases the costs associated with fulfilling a request. All this information is combined in an "offer" which is sent to the requesting edge.
- The requesting (problematic) edge then determines whether the costs are acceptable, by comparing it to a predefined value. These predefined values are currently arbitrary and static. We regard costs as an abstract notion which in reality could be financial or some other valuable asset (e.g. a guarantee from one TMS to another to "return the favour" at some point). Making predefined cost values dynamic would not be hard to model and could be interesting if, for example, a TMS is willing to pay more during rush hours.
- If the offer is accepted, then it is finalised and turned into a trigger for the traffic simulator. If there is "no deal" then obviously it is not.

At the end of this process all triggers are communicated to the simulator, which takes the effect of each traffic control measure/trigger into account before processing a new simulation step.

5 The VDM Model

In order to explore the different conceptual possibilities in connection with collaboration between different TMSs a VDM++ model[13] has been produced. In general the model contains a number of invariants and other constructs that can only be found in VDM++ and similar modelling notations, but some of these aspects are not included in the paper due to the lack of space. The purpose of this model is to illustrate the potential

[13] The model is available in the standard Overture library with VDM++ examples.

collaboration possibilities between TMSs, so focus has been on these aspects and ways in which it is possible to visualise collaboration for non-technical stakeholders. A class diagram of the most important classes can be found in Figure 2.

The VDM++ model follows the general design principles used for the development of distributed real time systems [7].

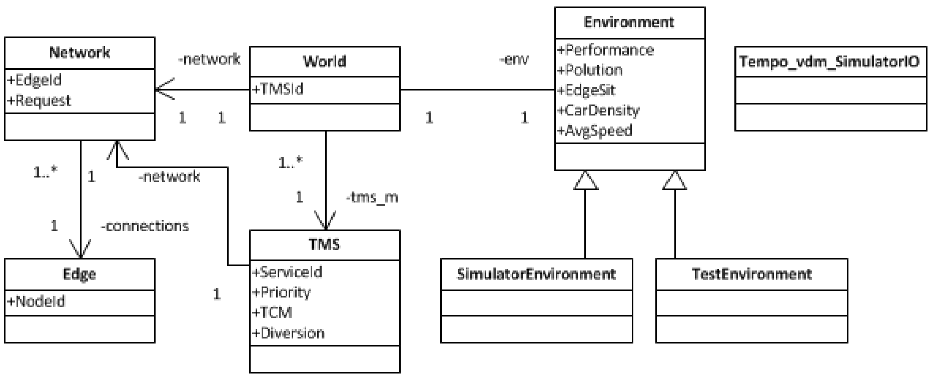

Fig. 2: Class Diagram with the most important TEMPO classes

These classes have the following function in the model:

World: The `World` class is the entry point for the VDM model. It has connections to the `Environment`, to the `Network` and to all the different `TMS` instances (organised in a mapping structure). This class collects the detailed information about the road network and the different TMSs controlling the different parts of the network. At the top-level two different `Run` operations are included in order to enable simulation of the same scenario either with or without collaboration, so that the difference between these two scenarios can be visualised.

Environment: The `Environment` class is a superclass for alternative realisations of the environment to the collection of TMSs that need to collaborate. The role of this class is to act as an interface to the main system, keeping track of the history of the traffic situation measured during a simulation.

SimulatorEnvironment: The `SimulationEnvironment` class is a subset of the `Environment` class and its role is to hook up to the low-level simulation environment realised via the `tempo_vdm_Simulation_IO` class. The latter communicates with the traffic simulator. The simulator visualises the progress of traffic flows and the traffic measures that are deployed.

TestEnvironment: The `TestEnvironment` class is a subset of the `Environment` class and its role is to provide a simpler traffic simulator without the user interface provided by the traffic simulator.

TMS: The `TMS` class represents a single Traffic Management System such that a combination of multiple TMSs can be considered.

Network: The `Network` class provides functionality for the road connections between different points in a graph-like fashion.

Edge: The `Edge` class provides functionality for edges in the graph representing the roads in the network.

tempo_vdm_Simulation_IO: The `tempo_vdm_Simulation_IO` class contains operations that interface to a small application that acts as a front-end to the traffic simulator. The output of this is easy to use for communication by domain experts and it can be seen in Figure 3.

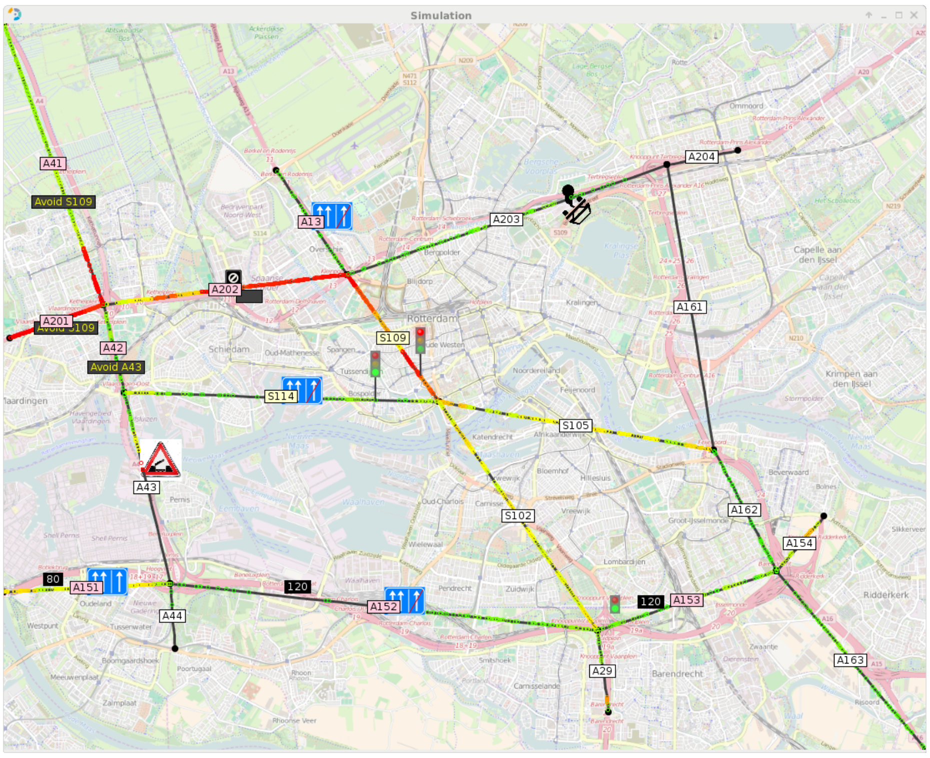

Fig. 3: Screenshot of the TEMPO simulation.

The configuration shown shows the network in and around the city of Rotterdam (i.e. a simplification if it) with two TMSs. The roads marked with an “A” in the figure are controlled by the TMS of the national road authority Rijkswaterstaat. Those marked with an “S” are controlled by the TMS of the municipality of Rotterdam.

In order to investigate the effect of collaboration between the different TMSs both possibilities are made available in the `World` class by having a simple instance variable controlling this:

```
collaboration : bool := true;
```

Inputs also are needed for the configuration of TMSs, for the road network, and for the events that can happen at different points in time during a simulation. All this information is stored in Excel files and read into the VDM model using the CSV standard library.

In each Step every TMS considers the traffic situation, and based on that generates triggers internally in the TMS and subsequently generates requests for help from adjacent TMSs. The requests are then processed by the overall network that has information about all TMSs that have been configured.

```
public Step: Environment`TrafficSituation ==> ()
Step(ts) ==
  (trigger:= {|->};
   trafsit := ts;
   GenerateMyOwnTriggers(ts);
   let requests = GenerateRequestsForHelp()
   in
     network.AddRequests (requests));
```

Inside the SimulatorEnvironment class, if collaboration is enabled, the top-level Run takes such service requests into account and makes offers that are subsequently evaluated before they are finalised, which means that the offers are accepted unless the TMS itself has deeper reasons for helping itself (instead of helping its neighbours):

```
public Run: bool ==> ()
Run(colab) == (
  while not isFinished() do (
    dcl trafsit: TrafficSituation;
    dcl control: TMS`Control := {|->};
    tempo_vdm_SimulatorIO`runSimulator(10);
    trafsit := UpdateSit();
    for all id in set dom tms_m
    do tms_m(id).Step(trafsit);
    if colab
    then for all id in set dom tms_m
           do tms_m(id).MakeOffers();
    for all id in set dom tms_m
    do tms_m(id).EvaluateOffers();
    for all id in set dom tms_m do
      let c = tms_m(id).FinaliseOffers()
      in
        control := control ++ c;
```

The argument passed to runSimulator (10) is the number of seconds of simulation time, so each step in the simulation is 10 seconds.

In order to control the traffic there are different Traffic Control Measures (TCMs) that can be applied:

```
TCM = HardShoulder | MaxSpeed | TrafficLight | RampMeter |
      Diversion | LaneClosure;
```

The type definitions for the various TCMs have parameters that determine specific settings for them. For example, a `TrafficLight` has an associated “green time” (duration of the time that traffic in the given direction can cross the intersection), whereas a `HardShoulder` just has a boolean value (open or closed).

Different thresholds are used to determine if it makes sense to activate some of these TCMs at specific edges at the road network. Here the active controls are gathered in an instance variable called `actctrl` that needs to be updated depending upon whether the given edge is problematic. This can for example be seen in the `SetHardShoulder` operation:

```
SetHardShoulder: Network`EdgeId * Environment`EdgeSit ==>
                  set of TCM
SetHardShoulder(eid, mk_(d,v,-,-,-)) ==
  if v < LOWSPEEDTHRESHOLD or d > HIGHDENSITYTHRESHOLD
  then (problematicedges := problematicedges union {eid};
        if eid not in set dom actctrl
        then return {mk_HardShoulder(true)}
        else return {})
  elseif v > HIGHSPEEDTHRESHOLD and d < LOWDENSITYTHRESHOLD
         and eid in set dom actctrl
  then return {mk_HardShoulder(false)}
  else return {}
```

The first part describes under which conditions a hard shoulder is opened: thresholds for either density or velocity at a given edge are crossed. The edge is tagged as *problematic* if this happens. Thresholds are used here to prevent toggling behaviour of the measures involved. The latter part of this operation deals with closing a hard shoulder when the traffic situation improves.

When an edge is “problematic” it sends out “requests for help” to other TMSs that are known to be able to influence that edge (typically input and output edges). For this an operation `GenerateRequestsForHelp` is used, that broadcasts the request for help to those TMSs that can potentially influence the problematic edge.

The request for help is expressed in terms of a *service* required. The service can either be `<IncreaseInput>` or `<DecreaseOutput>` at the moment[14]. The service called `<IncreaseInput>` is requested from downward edges to “absorb” more traffic so that tension on the problematic edge is relieved, and vice versa for `<Decrease-`

[14] Another future service might be `<IncreaseThroughput>`, a service e.g. provided by a hard shoulder opening.

`Output>`. A TMS asked for help can then see if it has any traffic control measures available that provide that service, and typically there exists a *1:n* relationship between a service and a TCM.

The requests for help are furthermore augmented with a severity indication, calculated by using a utility function:

```
Utility(network.GetMaxSpeed(eid),trafsit(eid), prios(eid))
```

The utility function here uses the maximum speed at an edge, the traffic situation at that edge (to compare with the maximum speed) and the priority of the road (roads with a higher priority get precedence and should be "helped" more quickly). The utility function is a function that in real life situations would be specified by traffic domain experts. The severity is used by the receiver of the request to compare to its own situation: it either determines whether or not help is granted or it influences the "price" that has to be paid for getting help. This is done in the operation `MakeOffers`, where each TMS iterates over the edges it controls to see if it has any traffic control measures available that provide the service requested. The operation also calculates the cost that comes with providing help:

```
let s = {e.service | e in set requestedservice},
    tcm = RequestedServices2PossibleTCMs(requestor, eid, s)
in
  let cost = Utility(network.GetMaxSpeed(eid),
                     trafsit(eid), prios(eid))-severity
  in
    if tcm <> {}
    then network.MakeOffer (eid, cost);
    ...
```

All requestors then evaluate the offers made to them with the operation `Evaluate-Offers`:

```
public EvaluateOffers: () ==> ()
EvaluateOffers () ==
(for all eid in set myedges do
  let offers = network.GetOffers(eid)
  in
    for all provider in set dom offers do
      let reqs = offers(provider)
      in
        network.AcceptOffers(provider,
                        {if request.cost < ACCEPTABLECOSTS
                         then mu(request,accepted |-> true)
                         else request
                        | request in set reqs})
);
```

So, as seen here, an offer is simply accepted when its associated costs are not above a given threshold (ACCEPTABLECOSTS). More sophisticated negotiation models would first proceed by making counter offers, and so on.

When an offer is accepted it is then finalised by transforming it into a trigger to set the traffic control measure that has been offered. The final set of triggers consists of:

– Triggers that are the result of each TMS seeing what it can do itself, i.e. apply the traffic control measures that are under its own control.
– Triggers that result from the negotiation process, i.e. traffic control measures that are part of accepted offers.

In real life, triggers would be passed on to the various local systems that implement the traffic control measures. In the simulator environment they are passed on to the traffic simulator where the effect that they have on the current traffic situation is calculated.

6 Results

The most significant result of the work presented in this paper is that we can demonstrate the working of traffic management with and without negotiation between the TMS involved. The initial results of executing the TEMPO VDM model are reflected in Figure 4, showing a plot monitoring the run-time average speed and density of the two different TMSs using the new Overture extension presented in [2]. We believe that the combination of the live plotting and the GUI from the traffic simulator can be useful in showing traffic management stakeholders that collaborative TMSs have added value.

In the figure the average speed and the average density in the two TMSs are plotted over time. Naturally this is heavily affected by the amount of traffic and the events incorporated in a simulation, so one needs to compare a large collection of cases in order to provide solid proof that negotiations between TMSs has a positive effect on traffic flow.

We also made first steps in comparing the performance of a network with and without collaboration in a more quantitative way by implementing some performance indicators, such as the number of vehicle kilometers produced by the network and the number of congestion kilometers that occur over the entire simulation run. These indicators, however, turn out to be quite sensitive to the specific topology and configuration (which traffic control measures are present and where are they located). For example, if the negotiation results in a better situation on a part of the network where the maximum speed is relatively low, then this will result in a lower number of vehicle kilometers produced, while intuitively the network has performed better. Therefore, it is questionable whether or not the indicators we choose are effective and accepted indicators by experts in the field of traffic modelling.

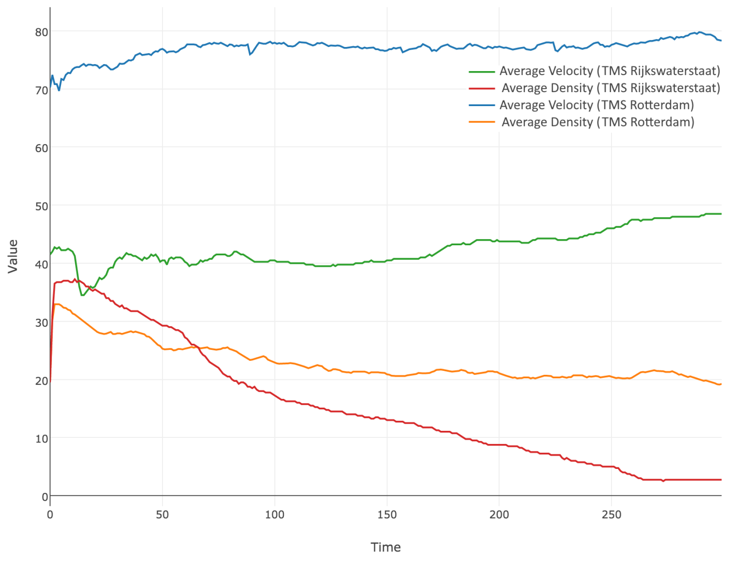

Fig. 4: Plotting speed and density with collaboration.

7 Concluding Remarks and Future Work

In this paper we have presented a novel way of dealing with conflicting interests between various TMSs, each responsible for guiding and steering traffic over (part of) a road network. We did this by specifying a VDM model that facilitates TMSs to negotiate about getting help from each other by applying traffic control measures that will improve overall performance, despite the fact that the measure may not have a positive effect for the TMS providing help itself. The hypothesis is that this results in an improvement of the overall performance of the network.

We built a traffic simulator that feeds the model with the (calculated) traffic situation on the network, and takes steering information from the model into account in each simulation step. The network, distribution of vehicle density and speed, and the status of traffic control measures, are visualised through a graphical user interface. This enables a nice demonstration of the working of the model, including the negotiation principles that have been included. Changes in density and speed for each TMS over time for the entire simulation run are visualised using new extensions to the Overture toolset.

A quantitative comparison between performance of a network with and without collaboration (based on negotiation) is the obvious first step for future work. Although we did build in some performance indicators, such as the number of vehicle kilometers produced by the network and the number of congestion kilometers that occur over the entire simulation run, these all turn out to be quite sensitive to the specific topology and configuration. This will probably require the involvement of experts in traffic modelling.

Ideas for future work include:

- Enhancement of the negotiation system. The negotiation strategy as described in this paper is relatively simple: the costs of offers made are compared to standard, predefined values. The offer is accepted if its costs are below that predefined value, and otherwise they are not. However, one can imagine a more sophisticated system where counter offers are made based on the initial offers that are being provided, possibly iterated over a number of steps. This could lead to more realistic values of what the offers are really worth. It would also be nice to be able to prepare the model for new types of negotiations needed in the future.
- Modelling of different types of TMSs. The two TMSs that are currently configured in the demonstrator (national road authority Rijkswaterstaat and municipality of Rotterdam) are "traditional" executing organisations of traffic management. It would be interesting to include new parties, such as providers of navigation equipment (introducing mobile sensors, and mobile control, in this case in-car dynamic trip planning/traffic information), cooperative systems, TMSs involved with other modalities (public transport, vessels, etc.) or even TMSs that do not have any control measures themselves (such as event organisers), but who can participate in the negotiation process.
- Exploring the possibility to use existing, more powerful traffic simulators (e.g. an open source simulator like SUMO[15]) instead of the one we built ourselves. Our main motivation for doing it this was that the budget for the project was limited, and we needed a simulator that could easily be adapted to the specific interface we are using for communication with the VDM model.
- Exploring the working of the model in a more physical setting, eg. using real traffic and using real traffic control measures. Worldwide there are several facilities where this can be done, varying of course in the level of sophistication they can offer to different experimenters.

Acknowledgments Our work was partially supported by the European Commission as a small experiment chosen for funding by the CPSE Labs innovation action (grant number 644400). Thanks also go to the anonymous reviewers of this paper and to Skip Balk, Pieter Buzing, Luis Couto, Kenneth Lausdahl, John Skovsgaard and Henrik Nymann Jensen for the various contributions they made to the TEMPO project.

References

1. Beam, C., Segev, A.: Automated Negotiations: A Survey of the State of the Art. Wirtschaftsinformatik 39(3), 263–268 (1997)
2. Couto, L.D., Lausdahl, K., Plat, N., Larsen, P.G., Pierce, K.: Extending Overture with Implicit Monitoring of Instance Variables. In: Larsen, P.G., Plat, N. (eds.) The 14th Overture Workshop: Analytical Tool Chains. Cyprus, Greece (November 2016)
3. Fitzgerald, J., Gamble, C., Larsen, P.G., Pierce, K., Woodcock, J.: Cyber-Physical Systems design: Formal Foundations, Methods and Integrated Tool Chains. In: FormaliSE: FME Workshop on Formal Methods in Software Engineering. ICSE 2015, Florence, Italy (May 2015)

[15] SUMO is an acronym for "Simulation of Urban MObility", see `http://sumo.dlr.de/wiki/Main_Page`.

4. Fitzgerald, J., Larsen, P.G., Verhoef, M. (eds.): Collaborative Design for Embedded Systems – Co-modelling and Co-simulation. Springer (2013)
5. Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra, C., Wooldridge, M.: Automated Negotiation: Prospects, Methods and Challenges. International Journal of Group Decision and Negotiation 10(2), 199–215 (2001), `http://eprints.soton.ac.uk/254231/`
6. Larsen, P.G., Battle, N., Ferreira, M., Fitzgerald, J., Lausdahl, K., Verhoef, M.: The Overture Initiative – Integrating Tools for VDM. SIGSOFT Softw. Eng. Notes 35(1), 1–6 (January 2010), `http://doi.acm.org/10.1145/1668862.1668864`
7. Larsen, P.G., Fitzgerald, J., Wolff, S.: Methods for the Development of Distributed Real-Time Embedded Systems using VDM. Intl. Journal of Software and Informatics 3(2-3) (October 2009)
8. Larsen, P.G., Lausdahl, K., Tran-Jørgensen, P.W.V., Ribeiro, A., Wolff, S., Battle, N.: Overture VDM-10 Tool Support: User Guide. Tech. Rep. TR-2010-02, The Overture Initiative, www.overturetool.org (May 2010)
9. Lee, E.A.: Cyber Physical Systems: Design Challenges. Tech. Rep. UCB/EECS-2008-8, EECS Department, University of California, Berkeley (Jan 2008), `http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html`
10. Maier, M.W.: Architecting principles for systems-of-systems. Systems Engineering 1(4), 267–284 (1998), `http://dx.doi.org/10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D`
11. Nielsen, C.B., Larsen, P.G., Fitzgerald, J., Woodcock, J., Peleska, J.: Model-based engineering of systems of systems. ACM Computing Surveys 48(2) (September 2015), `http://dl.acm.org/citation.cfm?id=2794381`
12. Nielsen, C.B., Lausdahl, K., Larsen, P.G.: Combining VDM with Executable Code. In: Derrick, J., Fitzgerald, J., Gnesi, S., Khurshid, S., Leuschel, M., Reeves, S., Riccobene, E. (eds.) Abstract State Machines, Alloy, B, VDM, and Z. Lecture Notes in Computer Science, vol. 7316, pp. 266–279. Springer-Verlag, Berlin, Heidelberg (2012), `http://dx.doi.org/10.1007/978-3-642-30885-7\_19`, ISBN 978-3-642-30884-0