Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 60 (2015) 1299 – 1308

Procedia Computer Science

19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

Formal model for intelligent route planning

Maria Spichkova[a,*], Milan Simic[b], Heinz Schmidt[a]

[a]School of Computer Science and Information Technology, RMIT University, Melbourne, Australia
[b]School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University, Melbourne, Australia

Abstract

This paper presents an approach towards intelligent route planning in public transport systems. The approach focuses on formal modelling of the semi-dynamic intelligent route planning and optimisation. For these purposes, it is essential to have a well developed formal model covering real-time and space aspects. The proposed solution allows designers to extend a public transport system with additional routes, which are created dynamically based on the requests from passengers. The model can be applied within a sustainable Smart City both for (fully or partially) autonomous transport systems and for the decision support systems of a smart transport system.

Keywords: Route planning; Transport system; Smart City; Formal modelling; Autonomous systems

1. Introduction

In this paper, we introduce our approach towards an intelligent route planning for a public transport system within a sustainable Smart City. To make a transport system as a part of a Smart City[1,2], we have to provide the corresponding solutions and intelligent features of the system. In our model, we analyse transport system routes that are created dynamically based on the passengers request. We call the corresponding route planning and optimisation *semi-dynamic*, because the set of possible stops is predefined and finite, which allows partial pre-calculation of possible extensions and optimisations of the route and the corresponding timetables.

Our approach can be applied both to route planning for autonomous systems (AS), which perform complex tasks without human intervention, and to the optimisation of the public transport systems (PTS), operated by humans. In the case of AS, the route planning becomes one of the main behavioural functions of the system. In the second case, our model becomes a decision support systems (DSS).

DSSs have been designed for industrial and organisational users, but as web-based technologies have progressed, the applications area has become more broad, cf. the work of Bhargava et al.[3] Large parts of DSSs are now developed for casual users, e.g. for passengers interested in taking the fastest connection from some address to a desired

* Corresponding author. Tel.: +61-3-9925-0452; fax: +61-3-9925-0399.
E-mail address: maria.spichkova@rmit.edu.au

destination point. Moreover, from the development of traditional desktop applications based DSSs, they have evolved to the development of distributed, web-based applications, cf. e.g. approaches of Power[4], Mitra and Valente[5]. In our current work we propose a formal model of an intelligent system that can be applied as a DSS for the public transport driver support, where the requests for the route are collected from the passengers dynamically in realtime.

In comparison to the DSSs, autonomous systems not only provide an assistance for the users of the system, but need to determine their behaviour based only on the predefined behaviours patterns and the adaptivity algorithms. DDSs, however, also can be viewed as a special way of partially designing humans out of the main system actions, particularly if we assume that the human will follow the decision recommended by the support system (cf. e.g., the approaches of Marakas[6], Bonczek et al.[7], Fick and Sprague[8]). For a survey of modeling and engineering aspects of self-adapting and self-optimizing systems, we would like to refer to the work of Bauer et al.[9]

Driver assistance applications[10] are a step forward in that direction. The first steps are just informing messages, or warning messages in critical situations, like blind spot warning, line change detection and similar. The next step, which is, also, already there, is giving full control to the vehicle, like in autonomous parking application, available today with many modern vehicles.

In our previous work[11], we presented a formal framework for modelling and analysis of autonomous systems and their compositions, especially focusing on the adaptivity modelling aspects and reasoning about adaptive behaviour. In our current work we are focusing on the semi-dynamic intelligent route planning for such systems. For this purposes, it is essential to have a well developed formal model covering real-time and space aspects. The modelling language that we use in our approach is based on FocusST[12], which was inspired by Focus[13], a framework for formal specification and development of interactive systems.

The main focus of the existing related work on the formal modelling of transport systems, is on automatic operating/control systems for the vehicles with fixed predefined routes (such as train systems, cf. e.g., approaches of Scippacercola et al.[14], James and Roggenbach[15], Behm et al.[16]), traffic simulation system (cf. e.g., approaches of Yu et al.[17], Chen et al.[18]), and advanced parking management systems (cf. e.g. approach of Elbanhawai and Simic[19]). Our approach, in contrast, focuses on investigating dynamic aspects of the route planning also taking into account readability and usability aspects of formal models[20,21].

As per statistics presented by Dhillon[22], the human is responsible for 30% to 60% of the total errors which directly or indirectly lead to accidents, and in the case of aviation and traffic accidents, 80% to 90% of the errors were due to human errors. Thus, an intelligent route planing for an autonomous vehicle can make the transport system more reliable. By the integration of human factors engineering into the development process, we can improve the quality of software and system in general as well as deal with human errors in a systematic way[23]. The use of structured error information helps to understand the real problems in the requirements documents and eliminate faults in software artefacts, cf. the approach of Walia and Carver[24]. However, to embed a methodology for human error analysis into the software engineering process, we have to classify the errors. There are a number of approaches in this field. For example, the approach of Mioch et al.[25] introduces a method for selection of error types and error production mechanisms. A review on the strategies for the human factor taxonomies is presented by Baziuk et al.[26]

The approach we present in this paper is based on the experiences from several projects of software and systems engineering in mobility, modelling and testing, especially within existing transport systems and automotive fields in Europe and Australia[27,28,29]. Various route planning algorithms were investigated and their applications analysed. We also have analysed the scheduling algorithms used is the scenarios for hard disc addressing, as the scenario of r/w head carriage traveling have many similarities to the route planning for an intelligent transport system, even when the application areas are very diverse. A part of the route planning system presented in this paper can be seen as a modified procedure for a disk scheduling algorithm. In the rest of the paper, we will emphasise the development of a formal model for an intelligent route planning for autonomous systems.

Outline: The rest of the paper is organised as follows: In Section 2, our model for route planning and the corresponding optimisation methods is introduced. Section 3 presents an example of a smart route based on our model. Section overviews examples of real application of intelligent systems and DDS. Section 5 concludes the paper by highlighting the main contributions and introduces the future work directions.

2. Model of a smart route planner

In this section we introduce a model of an intelligent route planner for an autonomous vehicle. We assume that the set $S_0, \dots, S_n$ of possible stops of the vehicle is predefined, so for each run the autonomous vehicle obtains and optimised list of stops it needs to serve.

We specify a route from the stop S_0 to the destination S_n as a set of runs R. A single run $R^{(d,t)}$ is defined for a particular day d and starting time t. For each day the vehicle has a number of the route runs, where each of the runs can be identified by its starting time.

Each stop S_i ($0 \le i \le n$) is specified via the following attributes:

- $location_i$ – spatial information about the stop. This attribute value is the same for all the runs, as a stop cannot change its location;
- $atMin_i^{(d,t)}$ – the earliest time, when the bus could arrive at the stop during this run in the best case. At this time the passenger should be already at the stop. This attribute value is static, i.e. the same for all the runs;
- $atMax_i^{(d,t)}$ – the latest time, when the bus should arrive at the stop during the run in the worst case. Initially, its value is set to the value of $atMin_i^{(d,t)}$. This attribute value is also static. Thus, we can say that the transport system has predictable time behaviour, which allows us apply the ideas similar to the time-triggered paradigm within automotive domain[30,31,32];
- $at_i^{(d,t)}$ – the estimated time, when the bus should arrive at the stop during this run, based on the set of current requests from the passengers. This attribute value is dynamic, as it depends on the current requests. It should be taken into account (updated and made visible with the system) only if the vehicle should stop at S_i during this run;
- $req_i^{(d,t)}$ – Boolean marker whether the stop is requested to be served during a particular run $R^{(d,t)}$;
- $reqTime_i^{(d,t)}$ – the requests to stop at S_i within the run $R^{(d,t)}$ can be accepted only before this time. This attribute value is static and calculated based on the spatio-temporal properties of the system. The upper limit for this value is $atMin_i^{(d,t)}$.
- $option_i^{(d,t)}$ – Boolean marker whether the stop should be served during a particular run $R^{(d,t)}$; $option_i = true$ means that the vehicle should stop at S_i while the run $R^{(d,t)}$. In general, the vehicle comes trough more stops, then it was requested, due to spatial constraints of the route.

The vehicle should arrive at the stop S_i, $0 \le i \le n$, during the time interval $[atMin_i^{(d,t)}, atMax_i^{(d,t)}]$.

Initially, for each run and for each stop S_i, the values of $option_i^{(d,t)}$ and $req_i^{(d,t)}$ should be set to $false$. The values of these attribute are changed, when a potential passenger requests the vehicle to the stop.

For each particular run of the route R (for a particular day d and starting time t), we define $Stops^{R(d,t)}$ to be the sequence of stops, on which the vehicle should stop during this run, i.e. for all i, $0 \le i \le n$,

$$S_i \in Stops^{R(d,t)} \Leftrightarrow option_i = true$$

The corresponding sequence of the requested stops is then defined by $ReqStops^{R(d,t)}$:

$$S_i \in ReqStops^{R(d,t)} \Leftrightarrow req_i = true$$

We can say that $ReqStops \subseteq Stops$.

The passenger's request Req should include the following information: route R, the pick-up stop S_i, day d and time p, as well as the desired drop-off stop S_j ($0 \le i, j \le n$, $i \ne j$). This information is required for the update of the plan for the route run. Thus, first of all the system analyses the runs on the day d to find the runs $R(d, tx)$, which time attributes values are the nearest to the requested pick-up time. The request should be sent before the time be sent before the time $reqTime_i^{(d,t)}$. The following cases are possible:

(1) There is a run $R(d, tx)$ with $atMin_i^{(d,tx)} \leq p \leq atMax_i^{(d,tx)}$. No further action is required from the passenger, the sets $Stops^{R(d,tx)}$ and $ReqStops^{R(d,tx)}$ will be updated for this run according to the request automatically.
(2) There is no run $R(d, tx)$ with $atMin_i^{(d,tx)} \leq p \leq atMax_i^{(d,tx)}$. There is a run $R(d, tx_1)$ with $p < atMin_i^{(d,tx_1)}$, which is the first available run (i.e. there is no run $R(d, tx_0)$ with $atMax_i^{(d,tx_0)} < p$). The passenger is requested to confirm whether the pick-up time between $atMin_i^{(d,tx_1)}$ and $atMax_i^{(d,tx_1)}$ would be also acceptable. If the passenger agrees to this time correction, sets $Stops^{R(d,tx_1)}$ and $ReqStops^{R(d,tx_1)}$ will be updated for the run $R(d, tx_1)$. Otherwise, the request will be cancelled.
(3) There is no run $R(d, tx)$ with $atMin_i^{(d,tx)} \leq p \leq atMax_i^{(d,tx)}$. There is also no run $R(d, tx_1)$ with $p < atMin_i^{(d,tx_1)}$, but there is a run $R(d, tx_2)$ with $atMax_i^{(d,tx_2)}$, i.e. the last available run. The passenger is requested to confirm whether the pick-up time between $atMin_i^{(d,tx_2)}$ and $atMax_i^{(d,tx_2)}$ would be also acceptable. If the passenger agrees to this time correction, sets $Stops^{R(d,tx_2)}$ and $ReqStops^{R(d,tx_2)}$ will be updated for the run $R(d, tx_2)$. Otherwise, the request will be cancelled.
(4) There is no run $R(d, tx)$ with $atMin_i^{(d,tx)} \leq p \leq atMin_i^{(d,tx)}$, but there are two runs $R(d, tx_1)$ and $R(d, tx_2)$ with $atMax_i^{(d,tx_1)} < p < atMin_i^{(d,tx_2)}$. The system allows the passenger to choose between these two runs or to cancel the request. The sets $Stops^{R(d,tx_1)}$ and $ReqStops^{R(d,tx_1)}$ will be updated for the chosen run.

After that, the system advises the passenger to be at the stop at the corresponding time $atMin_i$ and notifies that the vehicle should arrive no later than at the corresponding time $atMax_i$.

A simplified view on the execution of a concrete run $R(d, t)$ is presented on Figure 1.

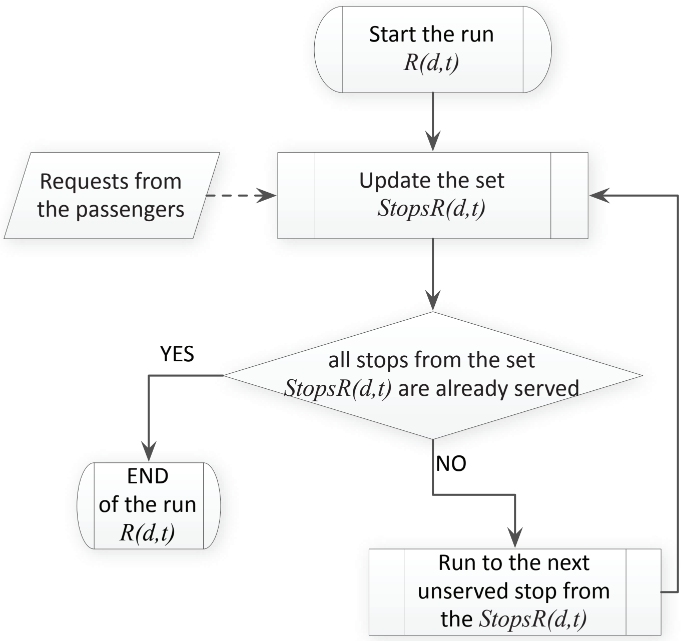

Start the run
R(d,t)
Requests from the passengers
Update the set StopsR(d,t)
YES
all stops from the set StopsR(d,t) are already served
NO
END of the run R(d,t)
Run to the next unserved stop from the StopsR(d,t)

Fig. 1. Execution of a concrete run $R(d, t)$

The core of our approach is the analysis of the spatio-temporal relations between the attributes of the stops with the run. Thus, the most interesting and the most complicated part of the algorithm is the dynamic update of the plan for the route. The update does not simply means “add the requested pick-up stop to the plan” but also the corresponding optimisation of the plan, according to the spatio-temporal information about the route. As we will discuss on an example later, a request for a single pick-up might mean that a number of stops get their *option*-parameter to be updated to be *true*.

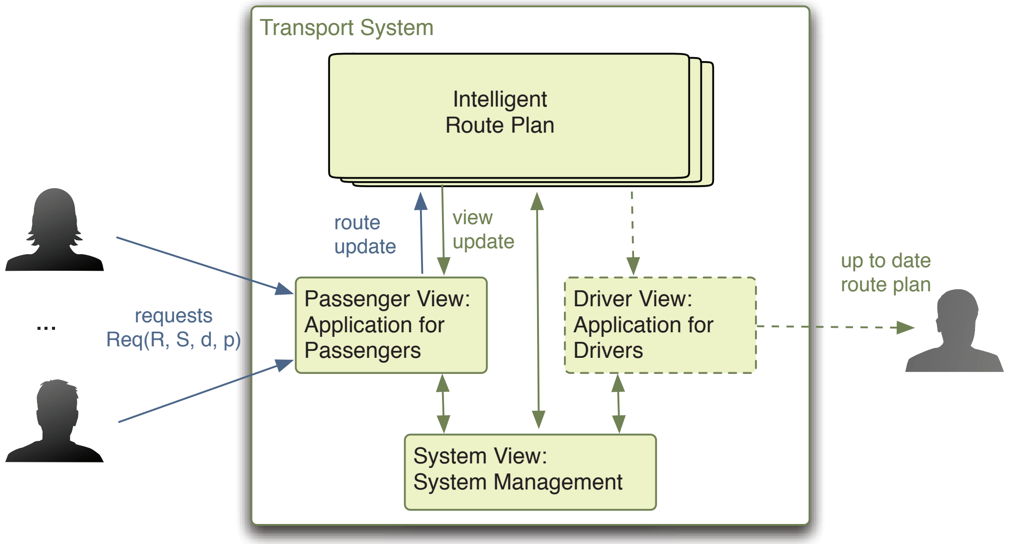


Fig. 2. General system architecture

To have a sustainable and human-oriented solution, we need three main architectural components of the transport system (cf. Figure 2 for the architecture of the system):

- *System Management*: The component is responsible for presentation of the complete information about the route and its runs, and also includes the statistics of the requests;
- *Application for Drivers*: For the autonomous vehicles, this component becomes a simple middleware for the route plan update. For the case the planing system is a DDS, this component is responsible for presenting to the driver the up to date route plan for this route run;
- *Application for Passengers*: The component is focused on the collection of the passengers' requesting to travel from/to a certain stop. It also provides to the passengers the information which stops are already marked as "requested" for the particular time. It would make no sense for a user to send a request for a vehicle to service a particular stop at the time t, if there were another earlier request for a vehicle to serve this same stop at this same time.

3. Example: Intelligent route planning

Assuming we have a route presented in Figure 3. The stops where the vehicle can turn back, are marked with darker colour. If the vehicle cannot turn back on its last stop to serve, it should proceed to the next stop which allows the shortest path back.

The set of stops is defined as $S_0, S_1, \dots, S_{15}$, and we assume that at the end of the journey the vehicle should return to its initial location on the stop S_0.

Initially for a particular run $R(dx, tx)$ starting at the time tx on the day dx, $option_i = false$ for all i, $0 \leq i \leq 15$. Thus, $Stops^{R(dx,tx)} = \langle\rangle$.

We denote by $p_0, \dots, p_{15}$ the pick-up times that corresponds to the run, starting at tx, where the following constraint holds for any i, $0 \leq i \leq 15$:

$$atMin_i^{(dx,tx)} \leq p_i \leq atMax_i^{(dx,tx)}.$$

Let us discuss examples of requests and their influences on the route plan, presented in Table 1. It is easy to see the difference between the sets $ReqStops$ and $Stops$ from this table.

For the case $Stops^{R(dx,tx)} = \langle\rangle$ (which also means $ReqStops^{R(dx,tx)} = \langle\rangle$), this set can become $\langle S_0, S_1, S_2, S_{15}, S_0\rangle$ after one of the following five requests:

$Req(R, S_0, dx, p_0, S_1)$,
$Req(R, S_0, dx, p_0, S_2)$,
$Req(R, S_1, dx, p_1, S_2)$,
$Req(R, S_1, dx, p_1, S_{15})$,
$Req(R, S_2, dx, p_2, S_{15})$.

However, $ReqStops^{R(dx,tx)}$ will be different for each of the above cases: $\langle S_0, S_1\rangle$, $\langle S_0, S_2\rangle$, $\langle S_1, S_2\rangle$, $\langle S_1, S_{15}\rangle$, $\langle S_2, S_{15}\rangle$, respectively.

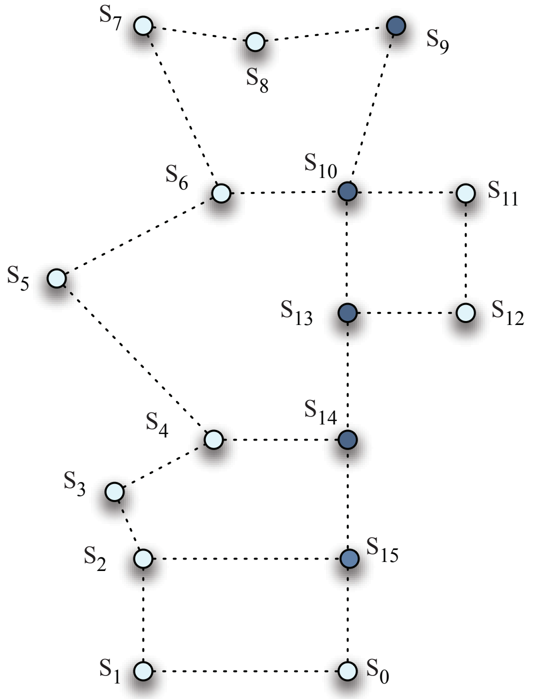


Fig. 3. Example of a smart route

In the case both pick-up and the destination stops are already in the $Stops$ sequence, the request will have no influence on the route plan, but the passengers will receive a confirmation, that these stops should be served for the chosen run. However, this might update the set $ReqStops$.

For example, if $Stops^{R(dx,tx)} = \langle S_0, S_1, S_2, S_{15}, S_0\rangle$, the following requests do not update the plan:
$Req(R, S_0, dx, p_0, S_1)$,
$Req(R, S_0, dx, p_0, S_2)$,
$Req(R, S_0, dx, p_0, S_{15})$,
$Req(R, S_1, dx, p_1, S_2)$,
$Req(R, S_1, dx, p_1, S_{15})$.

Table 1. Examples of the route plan updates for the case $Stops^{R(dx,tx)} = \langle\rangle$ before any request

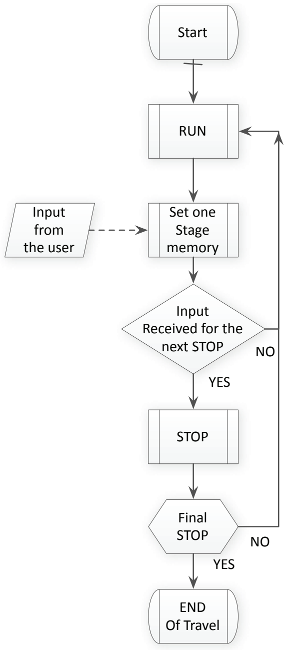
Request Options		$Stops$ after the request
$Req(R, S_0, dx, p_0, S_1)$	$Req(R, S_0, dx, p_0, S_2)$	$\langle S_0, S_1, S_2, S_{15}, S_0\rangle$
$Req(R, S_1, dx, p_1, S_2)$	$Req(R, S_1, dx, p_1, S_{15})$	
$Req(R, S_2, dx, p_2, S_{15})$		
$Req(R, S_0, dx, p_0, S_3)$	$Req(R, S_0, dx, p_0, S_4)$	$\langle S_0, S_1, S_2, S_3, S_4, S_{14}, S_{15}, S_0\rangle$
$Req(R, S_1, dx, p_1, S_3)$	$Req(R, S_1, dx, p_1, S_4)$	
$Req(R, S_1, dx, p_1, S_{14})$	$Req(R, S_2, dx, p_2, S_3)$	
$Req(R, S_2, dx, p_2, S_4)$	$Req(R, S_2, dx, p_2, S_{14})$	
$Req(R, S_2, dx, p_2, S_{15})$	$Req(R, S_3, dx, p_3, S_4)$	
$Req(R, S_3, dx, p_3, S_{14})$	$Req(R, S_3, dx, p_3, S_{15})$	
$Req(R, S_4, dx, p_4, S_{14})$	$Req(R, S_4, dx, p_4, S_{15})$	
$Req(R, S_0, dx, p_0, S_5)$	$Req(R, S_0, dx, p_0, S_6)$	$\langle S_0, S_4, S_5, S_6, S_{10}, S_{13}, S_{14}, S_{15}, S_0\rangle$
$Req(R, S_4, dx, p_4, S_5)$	$Req(R, S_4, dx, p_4, S_6)$	
$Req(R, S_4, dx, p_4, S_{10})$	$Req(R, S_4, dx, p_2, S_{13})$	
$Req(R, S_0, dx, p_0, S_7)$	$Req(R, S_0, dx, p_0, S_8)$	$\langle S_0, S_6, S_7, S_8, S_9, S_{10}, S_{13}, S_{14}, S_{15}, S_0\rangle$
$Req(R, S_6, dx, p_6, S_7)$	$Req(R, S_6, dx, p_6, S_8)$	
$Req(R, S_6, dx, p_6, S_9)$	$Req(R, S_6, dx, p_6, S_{10})$	
$Req(R, S_6, dx, p_6, S_{13})$	$Req(R, S_6, dx, p_6, S_{14})$	
$Req(R, S_6, dx, p_6, S_{15})$		
$Req(R, S_0, dx, p_0, S_9)$	$Req(R, S_9, dx, p_9, S_{10})$	$\langle S_0, S_9, S_{10}, S_{13}, S_{14}, S_{15}, S_0\rangle$
$Req(R, S_9, dx, p_9, S_{13})$	$Req(R, S_9, dx, p_9, S_{14})$	
$Req(R, S_9, dx, p_9, S_{15})$		
$Req(R, S_0, dx, p_0, S_{10})$	$Req(R, S_{10}, dx, p_{10}, S_{13})$	$\langle S_0, S_{10}, S_{13}, S_{14}, S_{15}, S_0\rangle$
$Req(R, S_{10}, dx, p_{10}, S_{14})$	$Req(R, S_{10}, dx, p_{10}, S_{15})$	
$Req(R, S_0, dx, p_0, S_{11})$	$Req(R, S_0, dx, p_0, S_{12})$	$\langle S_0, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_0\rangle$
$Req(R, S_{10}, dx, p_{10}, S_{11})$	$Req(R, S_{10}, dx, p_{10}, S_{12})$	
$Req(R, S_{11}, dx, p_{11}, S_{12})$	$Req(R, S_{11}, dx, p_{11}, S_{13})$	
$Req(R, S_{11}, dx, p_{11}, S_{14})$	$Req(R, S_{11}, dx, p_{11}, S_{15})$	
$Req(R, S_{12}, dx, p_{12}, S_{13})$	$Req(R, S_{12}, dx, p_{12}, S_{14})$	
$Req(R, S_{12}, dx, p_{12}, S_{15})$		
$Req(R, S_0, dx, p_0, S_{13})$	$Req(R, S_{13}, dx, p_{13}, S_{14})$	$\langle S_0, S_{13}, S_{14}, S_{15}, S_0\rangle$
$Req(R, S_{13}, dx, p_{13}, S_{15})$		
$Req(R, S_0, dx, p_0, S_{14})$	$Req(R, S_{14}, dx, p_{14}, S_{15})$	$\langle S_0, S_{14}, S_{15}, S_0\rangle$
$Req(R, S_0, dx, p_0, S_{15})$		$\langle S_0, S_{15}, S_0\rangle$

4. Applications of intelligent systems and DDS

In this section we give a short overview of a real application of intelligent systems and decision support systems. Intelligent route planning is already applied in various systems with less or more complexity. If we look at the train, shown on Figure 4(a), as one of the most popular and economical public transportation systems, we already have a rudimentary decision support system with the memory of one stage only. The basic flowchart capturing the system decision process is shown in Figure 4(b).

More advanced algorithms are used in operating systems, particularly by the disk management systems (cf.[33] for a deep introduction to this research area). The main objective here is to achieve the highest response time while serving large number of processes that request hard disk access. There are a number of disc scheduling algorithms. The first obvious and simple approach would be to use First Come First Serve (FCFS) algorithm. This algorithm has many application areas, e.g., in operating systems, in transaction management for databases, etc. This is not the best choice

(a)



(b)

Fig. 4. Example of a public transport systems using a rudimental DSS. (a) Trams in Melbourne, Australia. (b) A simplified representation of the DSS algorithm used in trams

since it leads to long paths when the successive requests are far apart. The criteria used to make decisions about the algorithms optimality are based on the shortest total travel while serving n requests, or the maximum throughput.

Analysing other various ideas, like Short Seek Time First (SSTF), SCAN, C-SCAN (Circular SCAN), LOOK and C-LOOK we end up with C-LOOK (Circular LOOK) approach (cf.[34] for a review on these techniques). The actor (e.g., in the case of disc scheduling this would be R/W head carriage system, in the case of an intelligent route this would be a vehicle) is moving in one direction, from the initial stage, that could be any, to the final stage, servicing all requests on the way. Since the requests could be placed any time, if they are placed for the address that was just left, or towards the any of the recently visited destinations, behind the direction of travel they will be serviced on the way back. Carriage system is going back to the request with the minimum address number, but not to the initial position if there are no requests. The same is with the maximum addresses. This method is used a lot in the disk scheduling, known as C-LOOK, but also in everyday life as lift management algorithm.

Algorithm presented in the Table 1 is modified C-LOOK procedure. It is also advanced since it involves time component, i.e. the memory of the management systems stretches along the time dimension as well.

5. Conclusion

In this paper, we introduced a formal model for an intelligent route planning and a scenario of its application within a public transport system.

The goal of this investigation is to make a transport system as a part of a Smart City, by providing smart planning of transport routes, both for autonomous and human driven systems. In the case of a human driven system our model becomes a decision support systems that can be applied within many existing transport routes over the world. In the case of an autonomous system, the route planning becomes one of the main behavioural functions of the system. This solution can allow for having more flexible/dense timetables and also possible longer routes. Moreover, the flexibility of the system would be especially beneficial for passengers with restricted physical abilities.

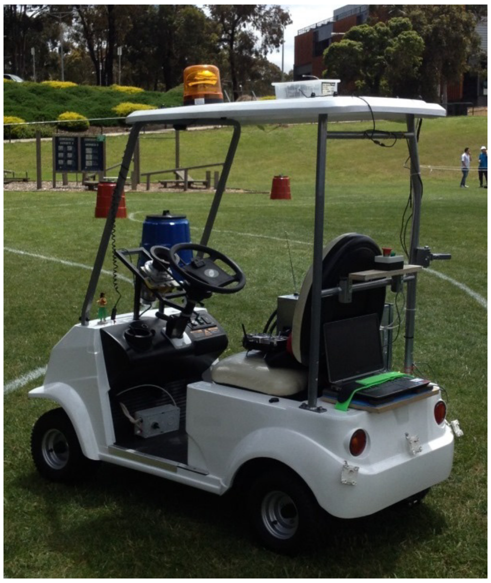


Fig. 5. Fully autonomous system with the control designed at the RMIT University

Future work: In the future we plan to apply the ideas presented in Sections 2 and 3 to the management of autonomous systems, as shown for instance in Figure 5. This particular system is designed for the golf courses, and suits an implementation of the formal model presented in this paper. After that, similar autonomous vehicles could be designed and put on the streets as part of the Intelligent Public Transport Systems.

Another possible direction of the future work is extension of our approach by model-based hazard and impact analysis[35]. Transport systems are safety critical, and for this kind of systems the hazard and impact analysis plays a crucial role.

Acknowledgements

We would like to thank the reviewers comments of the 19th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (Invited Session 14 – Intelligent Transport Systems) for their comments that helped us to improve the presentation of our work.

References

1. Ercoskun, O.. *Green and ecological technologies for urban planning: Creating smart cities*. IGI Global; 2011.
2. Nam, T., Pardo, T.. Conceptualizing smart city with dimensions of technology, people, and institutions. In: *12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*. ACM; 2002, p. 282–291.
3. Bhargava, H.K., Power, D.J., Sun, D.. Progress in web-based decision support technologies. *Decision Support Systems* 2007;**43**(1):1083–1095.
4. Power, D.. *Engineering Effective Decision Support Technologies: New Models and Applications*. Hershey, PA: IGI Global; 2013.
5. Mitra, G., Valente, P.. The evolution of Web-based optimisation from ASP to e-Services. *Decision Support Systems* 2007;**43**(1):1096–1116.
6. Marakas, G.. *Decision Support Systems, 2nd Edition*. Prentice-Hall; 2003.
7. Bonczek, R.H., Holsapple, C.W., Whinston, A.B.. *Foundations of decision support systems*. CA: Academic Press; 1981.
8. Fick, G., Sprague, R.. *Decision support systems: issues and challenges*. Pergamon Press; 1980.
9. Bauer, V., Broy, M., Irlbeck, M., Leuxner, C., Spichkova, M., Dahlweid, M., et al. Survey of modeling and engineering aspects of self-adapting & self-optimizing systems. Tech. Rep. TUM-I130307; TU München; 2013.
10. Simic, M.. Vehicle and Public Safety through Driver Assistance Applications. In: *Proceedings of the 2nd International Conference Sustainable Automotive Technologies (ICSAT 2010)*; vol. 490491. 2010, p. 281–288.
11. Spichkova, M., Simic, M.. Towards formal modelling of autonomous systems. In: *Intelligent Interactive Multimedia Systems and services: 2015*; KES-IIMSS. Springer; 2015 (to appear).

12. Spichkova, M., Blech, J.O., Herrmann, P., Schmidt, H.. Modeling spatial aspects of safety-critical systems with FocusST. *11th Workshop on Model Driven Engineering, Verification and Validation MoDeVVa*. 2014.
13. Broy, M., Stølen, K.. *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer; 2001.
14. Scippacercola, F., Pietrantuono, R., Russo, S., Zentai, A.. Model-driven engineering of a railway interlocking system. In: *International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. 2015, p. 509–519.
15. James, P., Roggenbach, M.. Encapsulating formal methods within domain specific languages: A solution for verifying railway scheme plans. *Mathematics in Computer Science* 2014;**8**(1):11–38.
16. Behm, P., Benoit, P., Faivre, A., Meynadier, J.M.. Meteor: A Successful Application of B in a Large Project. In: Wing, J.M., Woodcock, J., Davies, J., editors. *Formal Methods*; vol. 1708 of *LNCS*. Springer; 1999, p. 369–387.
17. Yu, Y., El Kamel, A., Gong, G.. Modeling intelligent vehicle agent in virtual reality traffic simulation system. In: *2nd International Conference on Systems and Computer Science (ICSCS)*. 2013, p. 274–279.
18. Chen, Q., Wilsher, A., Singh, D., Padgham, L.. Adding BDI Agents to MATSim Traffic Simulator. In: *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*; AAMAS '14. 2014, p. 1637–1638.
19. Elbanhawai, M., Simic, M.. Examining the use of B-splines in parking assist systems. *Applied Mechanics and Materials* 2014;**490491**.
20. Spichkova, M.. Design of formal languages and interfaces: "formal" does not mean "unreadable". In: Blashki, K., Isaias, P., editors. *Emerging Research and Trends in Interactivity and the Human-Computer Interface*. IGI Global; 2014.
21. Spichkova, M.. Human factors of formal methods. In: *IADIS Interfaces and Human Computer Interaction (IHCI 2012)*; 2012.
22. Dhillon, B.. *Engineering Usability: Fundamentals, Applications, Human Factors, and Human Error*. American Scientific Publishers; 2004.
23. Spichkova, M., Liu, H., Laali, M., Schmidt, H.. Human factors in software reliability engineering. In: *Workshop on Applications of Human Error Research to Improve Software Engineering*. WAHESE'15; 2015 (to appear).
24. Walia, G., Carver, J.. Using error information to improve software quality. In: *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 2013, p. 107–107.
25. Mioch, T., Osterloh, J.P., Javaux, D.. Selecting human error types for cognitive modelling and simulation. In: Cacciabue, P.C., Hjalmdahl, M., Luedtke, A., Riccioli, C., editors. *Human Modelling in Assisted Transportation*. Springer Milan; 2011, p. 129–138.
26. Baziuk, P.A., Rivera, S.S., Mc Leod, J.N.. Towards human factor taxonomy with cognitive generic terms. *Proceedings of the World Congress on Engineering* 2014;**2**.
27. Feilkas, M., Hölzl, F., Pfaller, C., Rittmann, S., Schätz, B., Schwitzer, W., et al. A Refined Top-Down Methodology for the Development of Automotive Software Systems –The KeylessEntry System Case Study. Tech. Rep.; TU München; 2011.
28. Spichkova, M., Hölzl, F., Trachtenherz, D.. Verified System Development with the AutoFocus Tool Chain. In: *2nd Workshop on Formal Methods in the Development of Software*; 2012.
29. Spichkova, M.. Architecture: Requirements + Decomposition + Refinement. *Softwaretechnik-Trends* 2011;**31:4**.
30. Kühnel, C., Spichkova, M.. Upcoming automotive standards for fault-tolerant communication: FlexRay and OSEKtime FTCom. *EFTS 2006 International Workshop on Engineering of Fault Tolerant Systems*; 2006.
31. Spichkova, M.. FlexRay: Verification of the FOCUS Specification in Isabelle/HOL. A Case Study. Tech. Rep. TUM-I0602; TU München; 2006.
32. Kühnel, C., Spichkova, M.. Fault-tolerant communication for distributed embedded systems. In: *Software Engineering of Fault Tolerance Systems (Series on Software Engineering and Knowledge Engineering)*; vol. 19. World Scientific Publishing; 2007, p. 175.
33. Silberschatz, A., Galvin, P.B., Gagne, G.. *Operating System Concepts*. Wiley Publishing; 8th ed.; 2008. ISBN 0470128720.
34. Celis, J.R., Gonzales, D., Lagda, E., Jr., L.R.. A comprehensive review for disk scheduling algorithms. *International Journal of Computer Science Issues* 2014;**11**(1):74–79.
35. Dobi, S., Gleirscher, M., Spichkova, M., Struss, P.. Model-based hazard and impact analysis. Tech. Rep. TUM-I1333; TU München; 2013.