Supervising Passenger Land-Transport Systems

Kiam Tian Seow, *Member, IEEE*, and Michel Pasquier

***Abstract*— In this paper, we propose a supervisory control approach, based on controlled automata concepts, to the planning for online service-operations control of a new class of Passenger Land-Transport Systems (PLanTS's). A PLanTS belongs to a class of dynamic demand-responsive transportation systems. Rapid advances in information and communication technologies are providing a new infrastructural and communications basis upon which higher levels of automation, flexibility and integration in the development of such transportation systems can be achieved. But to achieve these necessitates the use of more formal approaches for system planning and design. The supervisory control theory of Ramadge and Wonham offers one such methodology that presents the design for service-operations control as a formal synthesis of a modular supervisory controller. Importantly, the design solution is guaranteed to satisfy the given behavioral specifications in some optimal fashion, without blocking the completion of certain defined 'mandatory' tasks. The supervisory design methodology is presented and illustrated in detail via what in our opinion is a simplified but realistic PLanTS model. A structural property of the PLanTS model is used to analytically establish the *nonblocking* property of the modular supervisory controller designed. All automaton models for the PLanTS and the behavioral specifications considered are provided, together with the automata design of the corresponding modular supervisor.**

***Index Terms*— Automata, Discrete-Event Systems, Supervisory Control, Passenger Land-Transport, Service-Operations**

I. Introduction

Passenger land-transportation systems are concerned with transporting travellers from their source locations to their destination locations in a fleet of carrier vehicles, subject to various qualitative and quantitative constraints. These constraints characterize the environmental traffic conditions in which the services of transportation are carried out, as well as the operating conditions, limitations and preferences of the vehicle fleet operators and travellers. Taxi service management is an example of such a system. These systems are, however, *open loop* in that the logical feedback-control to react to and interleave the occurrences of incidents (eg. vehicle breakdown and admission of a travel request) in some desired manner, is apparently absent, implicit or at best done by way of *ad hoc* human intervention. Traditionally, the techniques available for these systems, such as those surveyed in [1], [2], do not *close the loop*, for they only determine the assignment of travellers to the fleet vehicles and construct the corresponding vehicles' service schedules or route plans. The automatic feed back of dynamically changing logical conditions needed to update the online information such as the availability of fleet-vehicles and the status of travel requests has never been formally characterized and explored. In other words, a basic research problem in passenger service-operations lies in the *open loop* nature of the information process flow in these transportation systems.

With rapid advances in information and communication technologies, such as Internet Technology [3], [4], Geographic Information Systems GIS [5], [6], Global Positioning Systems GPS [7], [8] and Intelligent Transportation Systems ITS [9], a new infrastructural and communications basis has emerged upon which *the information loop can be closed* to potentially achieve higher levels of automation, flexibility and integration towards the development of new transportation systems. But to achieve these necessitates the use of appropriate formal approaches for system planning and design. In particular, an alternative but complementary framework is needed that views a fleet of service vehicles and travellers uniformly as *behaviour-based* components, subject to various logical constraints to be met under close-loop supervision, or what we call *service-operations control*.

In this paper, we consider online service-operations control of a class of **P**assenger **Lan**d-**T**ransport **S**ystems (**PLanTS**'s). A **PLanTS** is a system that receives and services geographically-distributed travel requests, not known *a priori*, that demand immediate (i.e., 'as-soon-as-possible') service. It belongs to a class of dynamic demand-responsive transportation systems [10]. In an attempt to model and understand the dynamics of discrete information flow in the service-operations control of a **PLanTS**, we address the service-operations control problem using the controlled automata concepts and techniques of supervisory control for a class of logical discrete-event systems [11], [12], [13], [14], [15].

We model the service-operations in a **PLanTS** as a discrete-event system (DES) of interacting processes to be supervised or controlled. DES's represent dynamic systems that evolve in accordance to some abrupt and asynchronous occurrence of events. Such systems are encountered in a variety of many other fields, for example, in computer and communication networks [16], [17], manufacturing [18], [19] and task-level robotics [20].

To the best of our knowledge, our work represents a first effort to apply control-theoretic ideas of supervisory control to this class of transport service-operations problems. The approach is based on information feedback on the occurrence of events (see Fig. 1). Accordingly, the approach centres around three related elements, namely,

1) the models of the system (as discrete-event systems DES's) to be controlled,
2) the models of the control objectives (also called behavioral specifications) to be satisfied, and
3) a supervisory controller to be synthesized.

K.T. Seow is with the Division of Computing Systems, School of Computer Engineering, Nanyang Technological University, Republic of Singapore 639798. asktseow@ntu.edu.sg

M. Pasquier is with the Division of Computer Science, School of Computer Engineering, Nanyang Technological University, Republic of Singapore 639798. asmbpasquier@ntu.edu.sg

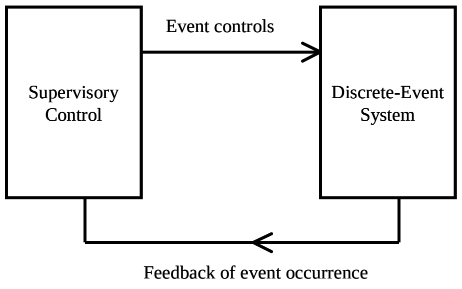


Fig. 1. Supervisory Control of DES

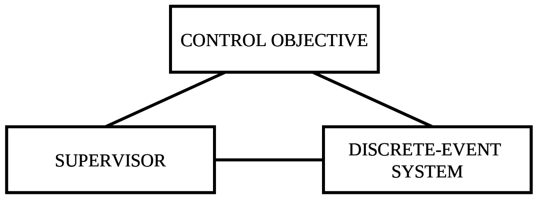


Fig. 2. The Logical Control Framework

These are graphically depicted in Fig. 2. The proposed methodology admits the design for service-operations control as an automata-based synthesis of a modular supervisory controller for the **PLanTS**. Other techniques such as Petri nets [21], [22] and communicating sequential processes [23] may be exploited, but the proposed methodology offers the following important advantages over these approaches:

1) the supervisory controller is correct by automatic construction, such that the resulting controlled system does not contradict the behavioral specifications, and is non-blocking; and
2) the controlled system is optimal (or *minimally restrictive*) within the behavioral specifications, such that all events whose occurrences do not eventually contradict the specifications are allowed to occur.

In other words, the design solution is guaranteed to satisfy given behavioral specifications (eg., vehicle seat-capacity must not be exceeded) in some optimal fashion, without blocking the completion of certain defined 'mandatory' tasks such as *emptying* the service-queue. A formal and conceptually rich control synthesis software **CTCT** [11] is now freely available[1] to support the automatic synthesis of supervisory controllers. Using **CTCT**, a DES model, behavioral specification and supervisory controller are represented by finite state automata that allow qualitative information such as the *admission of a travel request* and the *assignment of a request to an available vehicle* to be treated in a uniform way as events which are the state transitions in the automata.

In automating service-operations for passenger land-transportation, it is not exactly clear what constitutes a DES model for a **PLanTS**. Strictly speaking, no known prior and related work on service-operations control of a demand-responsive transportation system has been formally done from which a suitable DES model can be abstracted. However, a *service-oriented* model for a general transportation system exists [24] which conceptually specifies three submodels of *demand*, *supply* and *demand-supply* interactions. Our starting point is based on this conceptual model but seen from a supervisory control perspective. Together with a general understanding of the conventional but related problems of vehicle assignment and route planning (and the variations thereof) [1], [2], [10], we abstract and fix, in Section IV, what in our opinion is a simplified but realistic DES model for a **PLanTS**. The DES model incorporates the behavioral components of service demand by travellers and service supply by a fleet of vehicles. Importantly, the **PLanTS** model provides a basis on which various interesting behavioral specifications of interest representing the desired demand-supply interactions can be formulated, lending a unique opportunity to demonstrate the applicability of modular supervisory control theory to this problem. In our analysis, in Section IV-C.1, of a property that the **PLanTS** model has, we have also been able to infer some structural insights on a general DES model design which guarantees the nonblocking property in a supervisory controller that exists. These constitute the main contributions of this paper.

There has been some prior work on applying the supervisory control theory in different areas of intelligent transportation. For instance, Spathopoulos and de Ridder [25] consider the DES modelling and distributed supervisory control of a subway system. Yoo et al [26] design and verify a supervisory controller for a high-speed train. However, these past research has restricted itself to the supervision of a physical system such as a train or subway system modelled as a DES. As opposed to a physical oriented model, the research herein attempts to characterize, understand and supervise a service oriented model for a demand-responsive transportation system.

There has also been a lot of prior work done which is applicable to intelligent transportation. For instance, in the survey papers [1], [2], [10], [27] that include those cited earlier, algorithms based on heuristics, tabu search, constraint model and mathematical programming have reportedly been developed for the related problems of vehicle assignment and route planning. However, these algorithms are aimed at generating vehicle assignments and route plans that optimize (i.e., minimize or maximize) some *quantitative* performance specifications such as some cost or benefit functions. In contrast, the service-operations control problem addressed herein is aimed at regulating the flow of service-related events in passenger land-transportation in accordance to qualitative specifications, and is thus related but incomparable with these existing efforts.

The rest of the paper is organized as follows: Section II reviews the formulation and concepts of the supervisory control theory that are relevant to our research. Section III presents the supervisory design methodology to address the supervisory control problem in transport service-operations. Section IV illustrates the design methodology via a simplified but realistic **PLanTS**. All automaton models for the **PLanTS** are provided, together with the automata design of a modular supervisor that ensures proper service-operations according to a given set of behavioral specifications. Finally, Section

[1]CTCT design software can be downloaded from website http://www.control.toronto.edu/people/profs/wonham/wonham. html

V presents the conclusions and points to some future work. Preliminary versions of the research work appeared in [28], [29].

II. Review of Supervisory Control Theory

The control theory for discrete-event systems (DES) considered in our work is based on controlled automata concepts. The essential concepts and results reviewed are taken from [11] and can also be found in [13], [14], [15].

A. Discrete-Event Behaviour

1) Automaton Model: The behaviour of DES, such as (the service-operations of) a **PLanTS**, and behavioral specifications, can be modelled by finite state automata [30] at some appropriate level of abstraction. An automaton is a five-tuple

$$\mathbf{A} \overset{\text{def}}{=} (\Sigma, Q, \delta, Q_m, q_0)$$

in which

1) Σ denotes a finite set of transitions or event labels,
2) Q denotes a finite set of states,
3) $\delta : \Sigma \times Q \mapsto Q$ is a state transition function,
4) Q_m denotes a finite set of marked states (states indicating the completion of the tasks or sequences of tasks from a control perspective), and
5) $q_0 \in Q$ denotes the initial state.

Finite state automata are naturally described by directed-transition graphs. In order to represent the automaton $\mathbf{A}$, a state $q \in Q$ is identified by a node (represented by $\bullet$) of the graph whose edges are labelled by transition labels $\sigma \in \Sigma$ (represented by $\bullet \xrightarrow{\sigma} \bullet$. The initial state $q_0 \in Q$ is labelled with an entering arrow $\rightarrow\!\bullet$, while a marked state $q_m \in Q_m$ is labelled with an exiting arrow $\bullet\!\rightarrow$. When $q_0 \in Q$ is also a marked state, it is labelled with a double arrow $\leftrightarrow\!\bullet$.

2) DES as Composition of Automata: Consider an automaton $\mathbf{G}$ modelling (the behaviour of) a DES. A DES model $\mathbf{G}$ is usually modelled as a system of several interacting processes, each modelled by an automaton $\mathbf{G_i}$. To compose several automata $\mathbf{G_i}$ to obtain the global automaton $\mathbf{G}$, the idea of *synchronous* product of automata taken from [23], [11] is utilized.

$$\mathbf{G} = \mathbf{G_1} \parallel \mathbf{G_2} \parallel \cdots \parallel \mathbf{G_i},$$

where $\parallel$ is the composition operator. To elaborate, consider the case of two automata, i.e., i = 2. Then the synchronous product $\mathbf{G_1} \parallel \mathbf{G_2}$ models the behaviour $\mathbf{G_1}$ and $\mathbf{G_2}$ operating concurrently, by interleaving sequences generated by $\mathbf{G_1}$ and $\mathbf{G_2}$ such that

- events common to both the automata can occur only if each automata is in a state where such an event is defined; and
- events that are not common to both the automata may occur as long as they occur in the order defined by the respective transition functions of $\mathbf{G_1}$ and $\mathbf{G_2}$.

If the event sets of $\mathbf{G_1}$ and $\mathbf{G_2}$ are disjoint (i.e., no common event between the two), the synchronous product reduces to the shuffle product of $\mathbf{G_1}$ and $\mathbf{G_2}$. For a more formal definition, see Hoare [23].

3) Language Characterizations: The set Σ^* contains all possible finite sequences, or strings, over Σ, plus the null string ε. The definition of δ can be extended to Σ^* as follows:

$$\delta(\varepsilon, q) = q,$$
$$(\forall \sigma \in \Sigma)(\forall s \in \Sigma^*), \delta(s\sigma, q) = \delta(\sigma, \delta(s, q)).$$

The behaviour may then be described by two languages: $L(\mathbf{A})$, the prefix-closed language generated by automaton $\mathbf{A}$, and $L_m(\mathbf{A})$, the language marked by automaton $\mathbf{A}$. More formally,

$$L(\mathbf{A}) = \{s \in \Sigma^* : \delta(s, q_0) \text{ is defined } \}$$
$$L_m(\mathbf{A}) = \{s \in L(\mathbf{A}) : \delta(s, q_0) \in Q_m\}$$

By definition, $L_m(\mathbf{A}) \subseteq L(\mathbf{A})$ is the subset of strings in $L(\mathbf{A})$ which end in any of the states in Q_m and is a distinguished subset - if automaton $\mathbf{A}$ represents a DES, then Q_m is meant to represent completed 'tasks' (or sequences of tasks) carried out by the physical process that the model $\mathbf{A}$ is intended to represent [13]. If automaton $\mathbf{A}$ represents (or models) a behavioral specification K, then $K = L_m(\mathbf{A})$, the *behaviour of interest*.

An automaton $\mathbf{A}$ is said to be trim if it is accessible (i.e., every state $q \in Q$ is reachable in $\mathbf{A}$) and co-accessible (i.e., every state $q \in Q$ is co-reachable in $\mathbf{A}$). A state $q \in Q$ is reachable in $\mathbf{A}$ if there exists a string $w \in \Sigma^*$ such that $\delta(w, q_0) = q$; and co-reachable in $\mathbf{A}$ if there exists a string $w \in \Sigma^*$ such that $\delta(w, q) \in Q_m$. Note that if automaton $\mathbf{A}$ is trim, then $L(\mathbf{A}) = \overline{L_m(\mathbf{A})}$, i.e., every string in $L(\mathbf{A})$ can be completed to a string in $L_m(\mathbf{A})$.

B. Control Formulation and Concepts

As formally described in Section II-A, a DES (or plant) $\mathbf{G}$ can be modelled by an automaton:

$$\mathbf{G} \overset{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m).$$

To establish the control framework, the event set Σ is partitioned into disjoint sets of *controllable* events Σ_c and *uncontrollable* events Σ_u. Controllable events can be prevented (i.e., 'disabled') or allowed (i.e, 'enabled') by control, while *uncontrollable* events cannot be disabled by control and are deemed permanently enabled. The basic problem [13], [15] in supervisory control is to design a supervisory controller whose task is to enable or disable each of the controllable events during its observation of the event sequence generated by DES $\mathbf{G}$, such that the resultant closed-loop system generates only a subset of $L(\mathbf{G})$. Conceptually, a supervisory controller $\mathcal{S}$ consists of two components:

$$\mathcal{S} = (\mathbf{S}, \mathbf{V}) \tag{1}$$

where supervisor $\mathbf{S} = (X, \Sigma, \zeta, x_0, X_m)$ is an automaton called recognizer, and control law $\mathbf{V} : X \mapsto 2^{\Sigma}$ is the state feedback map. In a typical closed-loop configuration as shown in Fig. 1, the supervisor and the DES interact with each other via what is called 'event-feedback' through $\mathbf{V}(x)$. The automaton $\mathbf{S}$, as a language acceptor, is driven by the string of events generated by DES $\mathbf{G}$ and fed back to $\mathcal{S}$, which in turn, with $\mathbf{S}$ in state $x \in X$, the next set of events $\sigma \in \Sigma$ of DES $\mathbf{G}$ are subjected to the control law $\mathbf{V}(x)$ such that only events in $\mathbf{V}(x)$ are enabled. Note then that such a supervisor may be dynamic in the sense that not all strings of events of the DES $\mathbf{G}$ that lead to the same state will necessarily result in the same control action at that state. In a supervisor, the strings generated by the controlled system will always bring its automaton $\mathbf{S}$ to a defined state in X through the transition function ζ (see [13]). Consequently, the closed-loop system $\mathcal{S}/\mathbf{G}$, called the supervised or controlled DES, is another automaton which is defined as

$$\mathcal{S}/\mathbf{G} = Accessible(X \times Q, \Sigma, (\zeta \times \delta)^{\mathbf{V}}, (x_0, q_0), X_m \times Q_m)$$

where $(\zeta \times \delta)^{\mathbf{V}} : \Sigma \times X \times Q \mapsto X \times Q$ is defined as

$$(\zeta \times \delta)^{\mathbf{V}}(\sigma, x, q) = \begin{cases} (\zeta(\sigma, x), \delta(\sigma, q)) & \text{if } \sigma \in \mathbf{V}(x) \text{ and both transitions are defined} \\ undefined & \text{otherwise} \end{cases}$$

For a *proper* supervisory controller $\mathcal{S}$, at $(x, q) \in X \times Q$,

$$\sigma \in \mathbf{V}(x) \text{ iff } \zeta(\sigma, x) \text{ and } \delta(\sigma, q) \text{ are both defined,}$$

such that

$$\Sigma_u(q) \subseteq \mathbf{V}(x),$$

where $\Sigma_u(q) = \{\sigma \mid \delta(\sigma, q) \text{ is defined and } \sigma \in \Sigma_u\}$. Henceforth, unless otherwise stated, a supervisory controller is assumed to be proper.

Let $\Sigma(q) = \{\sigma \mid \delta(\sigma, q) \text{ is defined and } \sigma \in \Sigma\}$. Then, with $\mathbf{S}$ in state $x \in X$ and $\mathbf{G}$ in state $q \in Q$,

$$\mathbf{V}(x) = \Sigma(q) - \overline{\mathbf{V}(x)},$$

where $\overline{\mathbf{V}(x)}$ defines the subset of events $\sigma \in \Sigma_c$ disabled at $x \in X$.

In general, a supervisory controller $\mathcal{S}$ can be decomposed into two or more subsupervisors, giving rise to *modular* supervision. In the case of a modular supervisor $\mathcal{S}$ consisting of two supervisory controllers $\mathcal{S}_1$ and $\mathcal{S}_2$ given by

$$\mathcal{S}_1 = (\mathbf{S}_1, \mathbf{V}_1) \text{ and } \mathcal{S}_2 = (\mathbf{S}_2, \mathbf{V}_2),$$

we denote $\mathcal{S}$ by

$$\mathcal{S} = \mathcal{S}_1 \wedge \mathcal{S}_2,$$

such that

$$x = (x_1, x_2) \text{ and } \mathbf{V}(x) = \mathbf{V}_1(x_1) \cap \mathbf{V}_2(x_2).$$

Then, with $\mathbf{S}$ in state $x = (x_1, x_2) \in X$ and $\mathbf{G}$ in state $q \in Q$,

$$\mathbf{V}(x) = \Sigma(q) - \bigcup_{i=1}^{2} \overline{\mathbf{V}_i(x_i)},$$

and in this sense, we say that these two subsupervisors jointly enable (or disable) events. The above notion of modular supervision can be readily extended to more than two subsupervisors.

The behaviour of the supervised DES is described by the languages $L(\mathcal{S}/\mathbf{G}) = L(\mathbf{S}) \cap L(\mathbf{G})$ and $L_m(\mathcal{S}/\mathbf{G}) = L_m(\mathbf{S}) \cap L_m(G)$. Clearly $\overline{L_m(\mathcal{S}/\mathbf{G})} \subseteq L(\mathcal{S}/\mathbf{G})$. Supervisor $\mathcal{S}$ is said to be nonblocking if

$$\overline{L_m(\mathcal{S}/\mathbf{G})} = L(\mathcal{S}/\mathbf{G}),$$

i.e., every string in $L(\mathcal{S}/\mathbf{G})$ can be completed to a string in $L_m(\mathcal{S}/\mathbf{G})$.

C. The Nonblocking Supervisory Control Problem

The basic supervisory control problem considered [13], [15] is as follows:

Given a DES automaton $\mathbf{G}$ over an event set Σ, with the associated languages $L(\mathbf{G})$ and $L_m(\mathbf{G})$, and a behavioral specification (or control objective) $K \subseteq \Sigma^*$, the supervisory control problem is to find a (proper) nonblocking supervisory controller $\mathcal{S}$ such that $L_m(\mathcal{S}/\mathbf{G}) \subseteq K$ (or we say the DES under nonblocking control, $\mathcal{S}/\mathbf{G}$, satisfies specification K).

What this framework captures is a DES (finite-state machine) $\mathbf{G}$ and a behavioral specification describing a desired or legal behaviour K, with a supervisory controller being sought so that only desirable sequences of $L_m(\mathbf{G}) \cap K$ are generated.

1) A Centralized Solution: To provide a solution to the above problem, the notion of language controllability is introduced. A language $M \subseteq \Sigma^*$ is said to be controllable with respect to $\mathbf{G}$ if

$$\overline{M}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{M},$$

where $\overline{M}\Sigma_u = \{s\sigma \mid s \in \overline{M} \text{ and } \sigma \in \Sigma_u\}$. This controllability condition requires that if any $w \in \overline{M}$, i.e., any prefix of a string in M, followed by an uncontrollable event $\sigma \in \Sigma_u$ is in $L(\mathbf{G})$, then $w\sigma \in \overline{M}$, i.e., $w\sigma \in L(\mathbf{G})$ must also be a prefix of a string in M.

Now, suppose there is a $s \in \overline{K}$ such that $s\sigma \in L(\mathbf{G})$ is not in $\overline{K}$. Then, $s\sigma \notin K$. So, if $\sigma \in \Sigma_u$, no $\mathcal{S}$ exists that can exercise control to guarantee $L_m(\mathcal{S}/\mathbf{G}) = K$. In this case, we say that K is not controllable with respect to DES $\mathbf{G}$. But a largest or supremal controllable sublanguage (possibly $\emptyset$) of the marked language $K \subseteq L_m(\mathbf{G})$ with respect to $\mathbf{G}$ can always be found [14]. It is denoted by $\sup C(K, \mathbf{G}) \subseteq K$ and is a solution language $L(\mathbf{S})$ for the nonblocking supervisory controller $\mathcal{S}$ such that $L_m(\mathcal{S}/\mathbf{G}) \subseteq K$. To emphasize, the supervisory controller $\mathcal{S}$ is maximally permissive, i.e., it disables events in DES $\mathbf{G}$ only when absolutely necessary, as evident from the fact that the sublanguage $\sup C(K, \mathbf{G}) \subseteq K$ generated as a result is the largest.

2) A Modular Solution: Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be two automata. Then the prefix-closed languages $L(\mathbf{A}_1)$ and $L(\mathbf{A}_2)$ are said to be *nonconflicting* provided $\overline{L_m(\mathbf{A}_1) \cap L_m(\mathbf{A}_2)} = L(\mathbf{A}_1) \cap L(\mathbf{A}_2)$. If $\mathcal{A}_1 = (\mathbf{A}_1, \mathbf{V}_1)$ is a supervisory controller (of the form (1)) for $\mathbf{A}_2$, then $\mathcal{A}_1$ is a nonblocking supervisor for $\mathbf{A}_2$ provided $L(\mathbf{A}_1)$ and $L(\mathbf{A}_2)$ are *nonconflicting*.

With the above definitions, the result on modular supervision, readily extendible to more than two subsupervisors, may be given as follows:

If

1) $L(\mathbf{S}_1)$ and $L(\mathbf{S}_2)$ are each controllable with respect to DES $\mathbf{G}$,
2) $L(\mathbf{S}_1) \cap L(\mathbf{S}_2)$ and $L(\mathbf{G})$ are nonconflicting,

then $\mathcal{S}_1 \wedge \mathcal{S}_2$ is a nonblocking (modular) supervisory controller for DES $\mathbf{G}$.

A generic software **CTCT** [11] is now available to support the automatic synthesis of supervisory controllers. The DES and behavioral specification are input as automata to the software **CTCT**; operations supported by **CTCT** include composition of automata, supremal controllable sublanguage computation for a given behavioral specification with respect to the DES of interest, as well as a nonconflict test between two languages.

III. Design Methodology

To encompass the components of the supervisory control framework (see Fig. 2), the methodology to facilitate the planning for online supervision of a **PLanTS** consists of the following steps:

1) *Modelling the* **PLanTS** *and Behavioral Specifications*
   Both the service-operations process behaviour of a typical traveller and vehicle and the behavioral specifications are modelled as DES's translated into the form of automata. The modelling of the service-operations processes should also, through appropriate abstractions, take into account of the dynamic vehicle assignment (and routing) capabilities [27], [31] that an underlying planner is assumed to possess. More will be said about this planner later.
2) *Synthesizing the Supervisor and Control Law*
   Taking into account the supervisory control architecture adopted (for example centralized or modular), the automata representing the system of service-operations processes to be controlled and the automata representing the corresponding behavioral specifications are fed to the control synthesis program **CTCT** [11]. **CTCT** will tell us whether it is possible for the system to behave within the specifications and return the supervisor(s) and the corresponding control law(s) that ensure the controlled behaviour of the system is maximally permissive within the latter specifications.
3) *Simulating the Supervisor and Control Law*
   The supervisory control system is simulated to evaluate its effectiveness in that it takes appropriate actions in accordance to the supervisor(s) and corresponding control law(s). By default, all controllable events are assumed to be disabled. Let automaton $\mathbf{S}$ denote a supervisor, and $\mathbf{G}$, the **PLanTS** model. Then the simulation allows an enabled event as input (to simulate its occurrence), and control evaluation updates the current state x of $\mathbf{S}$ to, say x' and subsequently produces the corresponding control $\mathbf{V}(x')$ - the updated (online) permission set - only from which the next enabled event can be input. Only transitions in $\mathbf{V}(x)$ are events enabled or permitted to occur, and their occurrences never result in any eventual contradiction of the behavioral specifications; in this manner, $\mathbf{V}(x)$ keeps the system operations within the behavioral specifications.

Generally speaking, specifications should encompass the most desired[2] dynamic but orderly conditions under which a subset of vehicles in a given fleet is chosen, from which the admitted travel requests can be assigned to based on *quantitative* specifications asserting the desired *quality of service* to be achieved. These *quantitative* specifications are of course to be met by the underlying planner; the extent to which they would be met depends on various factors the planner considers, such as the vehicle assignment and route planning techniques used, the task-execution capabilities of the vehicle fleet and the accurate update of traffic information by the surveillance system. How the many existing algorithms - basic and applied - as reported in the literature (see [10], [27], [2], [32] and the references contained therein) might be exploited to address the related optimization problems of vehicle assignment and route-planning under close-loop supervision are beyond the scope of this paper. In the terminology of DES [11], the planner can be viewed as part of the underlying 'decision-making engine' of **PLanTS** that is capable of some 'choices' of spontaneous occurrences of events, including assignment and reassignment events denoted respectively by as^i_j and ras^i_j as precisely defined in Table II of Appendix I. These events are decision 'outputs' of the planner underlying our illustrative **PLanTS** model as described in the next section.

[2]What is meant by 'most desired' is decidedly a subjective opinion of the system analyst.

IV. A Simple Automated **PLanTS**

In this section, the design of a modular supervisory controller for a **PLanTS** using the methodology discussed in Section III is presented in detail.

A. Problem Description

In the scenario considered, travel requests are randomly initiated, geographically distributed, and require immediate or emergency attention. Each request is associated with only 1 person. The transport fleet is homogeneous. It consists of a small fleet of N vehicles and has a small seat capacity of C_s requests per vehicle and an assignment capacity of C_a requests per vehicle. The assignment-capacity C_a of a vehicle refers to the maximum number of requests that can be assigned to, but are yet to be fetched by the vehicle.

1) The **PLanT S***ystem Components:* The main behavioral components are described as follows.

1) **Initiator Behaviour:** This is a simple process that initiates the start and end of the transport service-operations.
2) **Vehicle Behaviour:** From an initial *shutdown* or *idling* state, each vehicle can be service-started or restarted. In the *service-ready* state, two possibilities are the vehicle ending its service-operations, or breaking down during operation. By the former occurrence, the vehicle returns to its *idling* state. By the latter, it falls into the *breakdown* state; repair and maintenance are then needed to get it to return to its initial state. In any state, it is possible that the vehicle gets trapped-in or out-of a traffic jam. The vehicle's task is considered completed once it ends its operations and is out of the traffic jam.
3) **Traveller Behaviour:** From the service-operations viewpoint, once a traveller is admitted for service, his request can possibly be cancelled either by the system or himself, or assigned to a particular vehicle by the underlying planner, after which the request cannot be cancelled unless the timeout set occurs before he boards the vehicle. While in a vehicle, the traveller has the options of making an urgent call (for another vehicle's service) or leaving the vehicle. The service-task is considered completed once the traveller exits the system.

2) Resource Limits N, C_s and C_a: In our current work, we consider the following limits: Number of Vehicles $N = 3$, Vehicle Seat-Capacity $C_s = 2$ and Vehicle Assignment-Capacity $C_a = 1$. These limits determine the upper bound on what we term *system processing limit*, as discussed next.

3) System Processing Limit $M \leq N(C_s + C_a)$: The system processing limit M is assumed to be the maximum number of travellers that can be concurrently serviced without degrading the performance of the underlying planner. Then, in general, the upper bound of the limit M, denoted M_{upp}, is $N(C_s + C_a)$ - the total of the maximal service-capacity $(C_s + C_a)$ of each vehicle that can possibly be concurrently utilized. By the resource limits in our current work, $M_{upp} = 9$, but we assume $M = 4 \leq 9$.

4) The Behavioral Specifications: The specifications to be conformed to via supervisory control are described below. In the context of **PLanTS**, these specifications are to be satisfied without 'blocking' or preventing the completion of any of the following mandatory 'tasks':

- the *emptying* of the service-queue and all service-vehicles,
- the *service-termination* of all service-vehicles in normal traffic conditions.

1) **Service Start-Up / Shutdown Operations**
   a) **Request Admission:** Once system operation-start has been initiated, all vehicles must be ready for service first before any travel request can be admitted.
   b) **Service Continuity:** During system operation, no vehicle is allowed to end its individual service-operations until the system operation-stop has been initiated, in which case no more travel requests will be admitted, and no vehicle will be service-restarted.
2) **Service Incident-Response Operations**
   a) **Vehicle Traffic Jam:** When a vehicle is caught in a traffic jam, no task (i.e., travel request) already assigned to another vehicle is allowed to be reassigned to the vehicle until it is out of the jam.
   b) **Limit on Service-Capacities:**
      i) **Vehicle Seat-Capacity:** The number of travellers (tasks-in-execution) in a vehicle must not exceed its seat-capacity of C_s.
      ii) **Vehicle Assignment-Capacity:** The number of (pending) assignments for a vehicle must not exceed C_a. Once assigned to a particular vehicle, a travel request must not be serviced by any other vehicle unless it is re-assigned or timeout occurs.

      A vehicle will not end its service-operations when its service-capacity (i.e., either seat or assignment capacity) is not empty.
   c) **Emergency Requests:** A traveller can make an emergency call to request service by another vehicle only when the vehicle servicing him has broken down.
   d) **Vehicle Breakdown:** When a vehicle breaks down, no further assignment or reassignment will be given to it, nor will any traveller be allowed to enter it unless it is service-restarted.
   e) **Fleet Service-Diligence:** Once system operation is ready, no vehicle is allowed to end its individual service-operations when the pending travel request 'queue' is not empty.

B. Modelling for **PLanTS** *and Behavioral Specifications*

Formalizing, the **PLanTS**'s component processes and the behavioral specifications introduced above are embodied in automata shown in Appendix I and listed in Table I therein. The event definitions are given in Table II of Appendix I. The trim automaton model $\mathbf{G}$ for **PLanTS** is a composition of its component processes via synchronous product [23], [11] as discussed in Section II-A.2. The completion of the mandatory tasks of the **PLanTS** is represented by a marked state which is formed by collecting together the marked state in each of the **PLanTS**'s component processes. The 'Number' column in Table I indicates the number of automata in each category for the **PLanTS** with $N = 3$ and $M = 4$.

C. Supervisor and Control Law Synthesis

In this section, we first discuss, in relation to the notion of language nonconflict, the special structures of the automata representing **PLanTS** model and all the behavioral specifications considered. These structures help to analytically establish the *nonblocking* property of our modular supervisor design.

1) Special Structures and Nonconflicting Languages: The trim structure of the automaton model $\mathbf{G}$ for **PLanTS** is such that there exists a string of uncontrollable events that leads any unmarked state in the structure to a marked state (which is the initial state) of the system. This property is formalized as follows.

Property 1: At any 'unmarked' state $q \in Q - Q_m$ of DES model $\mathbf{G} \overset{\text{def}}{=} (\Sigma, Q, \delta, Q_m, q_0)$, there exists a string $t \in \Sigma_u^*$ such that $\delta(t, q) \in Q_m$.

In the following, Property 1 and the notion of language nonconflict [11] (reviewed in Section II-C.2) are used to establish Theorem 1. As the subsequent section will show, this theorem allows us to analytically establish the second condition (as in Section II-C.2) of nonblocking modular control synthesis [11], [15] for **PLanTS**, *without* directly testing the property of language nonconflict which, for the whole set of specification automata considered (as shown in Appendix I-B), is found to be infeasible for the **CTCT** software to verify, due to the large state space complexity of intersecting these behavioral specifications.

Theorem 1: A *controllable* prefix-closed language $L(\mathbf{A})$, with automaton $\mathbf{A}$ having the property of $\mathbf{G}$-closure [13], i.e.,

$$\text{if} \quad s \in L(\mathbf{A}) \cap L_m(\mathbf{G}), \text{ then } \quad s \in L_m(\mathbf{A}),$$

is *nonconflicting* with (the prefix-closed language of) a DES model $\mathbf{G}$ that satisfies Property 1 .

Proof: Given that the prefix-closed language $L(\mathbf{A})$ is controllable with respect to $\mathbf{G}$. Then, suppose $L(\mathbf{A})$ conflicts with $L(\mathbf{G})$; this means that there exists a prefix $s \in L(\mathbf{A}) \cap L(\mathbf{G}) - \overline{L_m(\mathbf{A}) \cap L_m(\mathbf{G})}$; $\delta(s, q_0) \in Q - Q_m$ (i.e., prefix s is not marked by automaton $\mathbf{G}$) since automaton $\mathbf{A}$ is $\mathbf{G}$-closed. By Property 1, there exists a $t \in \Sigma_u^*$ such that $st \in L_m(\mathbf{G})$. But $st \notin L_m(\mathbf{A})$ (because prefix s *cannot be* completed to a marked string in $L_m(\mathbf{A}) \cap L_m(\mathbf{G})$). Clearly $st \notin L(\mathbf{A})$ since automaton $\mathbf{A}$ is $\mathbf{G}$-closed, thus contradicting the fact that $L(\mathbf{A})$ is controllable. ■

Remark 1: It is easy to re-designate some unmarked states as marked states to transform a given automaton into an automaton $\mathbf{A}$ that is $\mathbf{G}$-closed. Therefore, we can readily infer from Theorem 1 that an arbitrary DES model design that satisfies Property 1 will guarantee nonblocking in a supervisory controller ($\mathbf{A}$, $\mathbf{V}$) that exists.

2) Nonblocking Modular Synthesis: The trim automaton $\mathbf{G}$ obtained for **PLanTS** has 559,872 states and 11,197,440 transitions ! Fortunately, we could find a simpler trim model $\mathbf{G}'$ for **PLanTS**, having 110,592 states and 2,875,392 transitions, that renders the control computation using **CTCT** feasible. The control synthesis with respect to model $\mathbf{G}'$ is the same as that with respect to model $\mathbf{G}$, the elaboration of which is given in Appendix I-A. The automata representing the behavioral specifications are shown in Appendix I-B, and arbitrarily referenced herein as $\mathbf{S}_k$. Using **CTCT** on model $\mathbf{G}'$, each prefix-closed language $L(\mathbf{S}_k)$ is found to be controllable with respect to **PLanTS** model $\mathbf{G}$. Hence the overall prefix-closed language ($\bigcap_{all\ k} L(\mathbf{S}_k)$) is also controllable [11]. Let $L(\mathbf{A}) = \bigcap_{all\ k} L(\mathbf{S}_k)$, with automaton $\mathbf{A}$ being the (reachable) cartesian product of all $\mathbf{S}_k$; hence, $L_m(\mathbf{A}) = \bigcap_{all\ k} L_m(\mathbf{S}_k)$. By inspection, each simple automaton $\mathbf{S}_k$ (as in Appendix I-B) is $\mathbf{G}$-closed; it then follows that automaton $\mathbf{A}$ is also $\mathbf{G}$-closed and hence, by Theorem 1, $\bigcap_{all\ k} L(\mathbf{S}_k)$ is nonconflicting with $\mathbf{G}$. Thus, according to the modular control result [11], [15] (reviewed in Section II-C.2), $\bigwedge_{all\ k} \mathcal{S}_k$, where $\mathcal{S}_k$, in coded form, is shown in Appendix II, can serve as a modular supervisory controller which, when acting in synchrony with **PLanTS**, is nonblocking with respect to those mandatory 'tasks' defined at the beginning of Section IV-A.4, and therefore generates the largest ('marked') sublanguage of the **PLanTS** model that lies within the overall specification $L_m(\mathbf{A})$.

D. Supervisor and Control Law Simulation

To illustrate that the supervisory subcontrollers thus obtained jointly enable or disable events correctly, the following case is simulated.

Proper Start-Up/Shutdown: The following two sequences test the specification $\bigcap_{i=1}^{3} L_m(\mathbf{Vi\_SUDSP})$, (where automaton $\mathbf{Vi\_SUDSP}$ is shown in Fig. 6). Note that these test (prefix) sequences are feasible sequences of **PLanTS**, and violate only the specification under test.

By sequence 1: $tstart, st_1, st_2, ad_2, ...,$
the system operation-start is successfully initiated, but only vehicle 1 and vehicle 2 are ready for service when travel request 2 is admitted, thus violating the specification which requires all vehicles to be ready first after startup before any request can be admitted. Indeed, the simulation succeeds because only the events $tstart, st_1, st_2$ can be input in that order; event ad_2 cannot be input thereafter because it is disabled.

By sequence 2: $tstart, st_1, st_2, st_3, end_1, ...,$
the system operation-start is successfully initiated and completed with all vehicles ready for service when vehicle 1 ends its service-operations, thus violating the specification which requires an order of shutdown (event $tstop$) to be given first before any vehicle could end its individual operations. Again, the simulation succeeds this time because event end_1 cannot be input at that respective instance.

V. Conclusion and Future Work

Our initial study reported herein suggests that the theory of supervisory control [11] provides a useful framework for capturing the high-level structure of a dynamic service-operations manager for a **PLanTS**. The structure is realized in the form of a *permission-based* supervisory policy. The supervisory control theory offers a simple methodology for describing the event-based characteristics of a **PLanTS** and for determining the existence of nonblocking supervision. If nonblocking supervision exists, it is guaranteed that the **PLanTS** under such control will satisfy the behavioral specifications in the least restrictive manner without preventing (or 'blocking') any 'marked' service-operations task from completion. Besides, changing the desired behavior of the **PLanTS** is a matter of adding or removing a behavioral specification.

In the domain of planning for passenger transport-service, the supervisory control framework serves to provide a new basis to support the systematic development for online supervision of **PLanTS**. In this research, a modular approach using at least two subsupervisors to jointly track the behavior of **PLanTS** is considered. Importantly, the modular supervisor designed is structurally very simple and it issues permissions $\mathbf{V}(\mathbf{x})$ that form the necessary *logical* condition for the online validity of the vehicle assignment and route plans generated by the underlying planner.

The state space complexity which arises from intersection of all $L(\mathbf{S}_k)$, where $\mathbf{S}_k$ refers to the automaton of each behavioral specification given in Appendix I-B, prohibits the use of the **CTCT** software to verify the property of nonconflict between $\bigcap_{all\ k} L(\mathbf{S}_k)$ and $L(\mathbf{G})$. But, fortunately, by exploiting the special structures of the **PLanTS** model $\mathbf{G}$ and all $\mathbf{S}_k$, as formally discussed in Section IV-C, the property of *nonconflict*, and hence *nonblocking* for modular supervision, could be verified. The structural property of the **PLanTS** model provides a practical guide to general DES model design which guarantees the nonblocking property in a supervisory controller that exists.

It is hoped that in the future, progress on supervisory control will render the complexity problem more manageable, so that a more capable transport-service planner in terms of a larger N, C_s and C_a may be deployed using the proposed supervisory control approach. Note that in our illustrative supervisory design for the **PLanTS**, the system limit M is restricted to only 4 for a fleet of 3 vehicles ($N = 3$). Our conjecture about the design is that it is 'scalable' in the sense that the controllability and nonconflicting properties of the corresponding modular supervisor can be preserved for the same set of behavioral specifications and **PLanTS**, but extended accordingly to account for a larger $N > 3$ and $M > 4$.

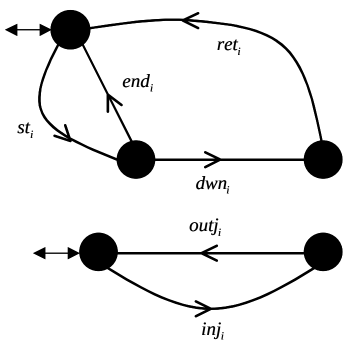
Finally, the hybrid architectural issues of developing the underlying planner to perform lower-level tasks such as distributed vehicle assignment and route planning under close-loop supervision would need to be defined and investigated.

APPENDIX I MODELS

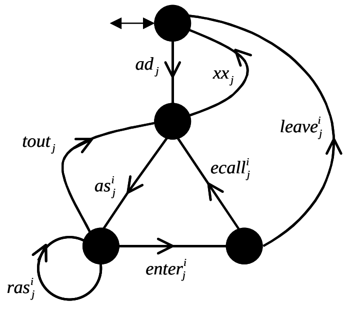
A. *For* **PLanTS**

The vehicle identification number (ID) is i, $1 \leq i \leq N$, and the admitted traveller ID is j, $1 \leq j \leq M$. In the **PLanTS** considered, $N = 3$ and $M = 4$. In the 'Code No.' column of Table II are the **CTCT** code representations of the various events in **PLanTS**. For instance, event as_1^3 - 'Traveller 1 assignment to Vehicle 3' - is arbitrarily represented in **CTCT** as 115, an odd number denoting that it is controllable, while event $leave_3^2$ - 'Traveller 3 leaves Vehicle 2' - is represented as 362, an even number denoting an uncontrollable event.

Composing (via synchronous product [11]) the trim automaton for Traveller j (as shown in Fig. 4(b)) and all the N automata (as shown in Fig. 5) yields another trim automaton (conveniently referenced as $\mathbf{TN_j}$) for Traveller j which includes the 'natural' constraint that Traveller j can only leave the vehicle he has entered (not any other vehicle !). We call these automata *entry-exit* laws when we refer to the kind of constraint they impose, via composition, on the automaton for Traveller j (as shown in Fig. 4(b)). Referring to our illustrative **PLanTS** with $N = 3$ and $M = 4$, incorporating the set of M such composed automata $\mathbf{TN_j}$ (each of 6 states and 18 transitions) instead via synchronous product results in an overall automaton model **G** for **PLanTS** having 559,872 states and 11,197,440 transitions ! For the behavioral specifications that we consider, the control synthesis with respect to this resultant **PLanTS** model **G** is beyond the computational capacity of **CTCT** running on a personal computer with a 200MHz CPU and 64MB memory. Fortunately, feasibility of control computation using **CTCT** can still be achieved by considering only the trim automaton that has a smaller state size (of 4 states and 18 transitions as shown in Fig. 4(b)) as the behavioral process of Traveller j. The N entry-exit laws consist only of events in this trim automaton for Traveller j, and composing all of them yields another trim automaton $\mathbf{TN_j}$ for Traveller j that *only excludes* the obviously impossible situations of Traveller j in one vehicle leaving another vehicle. Hence, these N laws effectively form a component of the **PLanTS**'s underlying engine for the behavioral process of Traveller j. It is thus practically inconsequential to supervision (that exists) whether or not these automata-theoretic laws are directly incorporated into the overall model for **PLanTS**, but by *not* doing so, a **PLanTS** model $\mathbf{G}'$ (of 110,592 states and 2,875,392 transitions) with a much smaller state size results, and this renders our control synthesis of individual specification automata using **CTCT** computationally feasible.



(a) **Vehicle** i



(b) **Traveller** j

$1 \leq i \leq N$, $1 \leq j \leq M$

Fig. 4. Basic Service-Operations Processes in a **PLanTS**

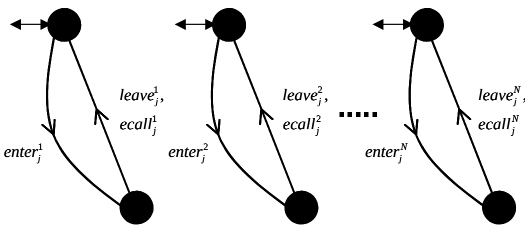


Fig. 5. Mechanism Underlying Traveller j, $1 \leq j \leq M$

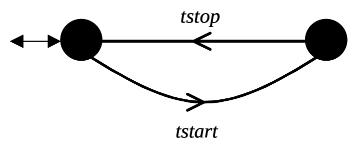


Fig. 3. **System Initiator**: Initiator Process in a **PLanTS**

B. *For Behavioral Specifications*

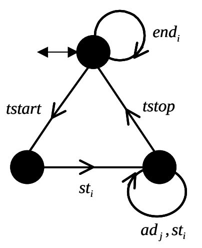
In each of the following specification automata (Figs. 6 through 12), self-loops (not shown) must be adjoined to account for all events which are irrelevant to the specification, but which may be executed in the **PLanTS** model.

TABLE I
AUTOMATA FOR **PLanTS** AND BEHAVIORAL SPECIFICATIONS

Process Name	Category	Automata		Number
PLanTS	**System Initiator**	-	Fig. 3	1
	Vehicle i	-	Fig. 4(a)	3×2
	Traveller j	-	Fig. 4(b)	4
	Entry-Exit laws in **Traveller** j	-	Fig. 5	4×3
Service Start-Up/Shutdown Operations	**Request Admission and Service Continuity**	**Vi_SUDSP**	Fig. 6	3
Service Incident-Response Operations	**Vehicle Traffic Jam**	**Vi_TJMSP**	Fig. 7	3
	Vehicle Seat-Capacity	**Vi_SEATSP**	Fig. 8	3
	Vehicle Assignment-Capacity	**Vi_AGNSP**	Fig. 9	3
	Emergency Request	**Vi_EMSP**	Fig. 10	3
	Vehicle Breakdown	**Vi_DWNSP**	Fig. 11	3
	Fleet Service-Diligence	**FLT_DILSP**	Fig. 12	1

TABLE II
EVENTS FOR **PLanTS** MODEL

Process	Events – **c**: controllable (odd no.); **u** : uncontrollable (even no.)			Code No.
System Initiator	$tstart$	Start of system operation	**c**	01
	$tstop$	End of system operation	**c**	03
Discrete Traffic Change for Vehicle i	inj_i	Vehicle i caught in traffic jam	**u**	$i4$
	$outj_i$	Vehicle i out of traffic jam	**u**	$i6$
Service 'Operational Status' Behaviour of Vehicle i	st_i	Vehicle i service-start	**c**	$i1$
	end_i	Vehicle i service-end	**c**	$i3$
	dwn_i	Vehicle i breakdown	**u**	$i0$
	ret_i	Vehicle i return	**u**	$i2$
Service 'Demand' Behaviour of Traveller j	ad_j	Request admission and label as Traveller j	**c**	$j01$
	xx_j	Traveller j request-cancellation	**u**	$j02$
	as_j^i	Traveller j assignment to Vehicle i	**c**	$j1(2i-1)$
	ras_j^i	Traveller j re-assignment to Vehicle i	**c**	$j2(2i-1)$
	$tout_j$	(Waiting) Time-out for Traveller j	**u**	$j30$
	$enter_j^i$	Traveller j enters Vehicle i	**c**	$j4(2i-1)$
	$ecall_j^i$	Traveller j leaves Vehicle i and makes emergency call	**c**	$j5(2i-1)$
	$leave_j^i$	Traveller j leaves Vehicle i	**u**	$j6(2i-2)$

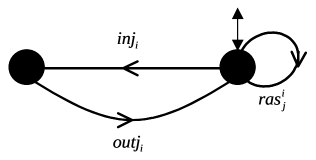


(Fixed i, Variable j)
Selfloop = $\Sigma - \{tstart, tstop, ad_j, st_i, end_i\}$

Fig. 6. **Vi_SUDSP: Service Start-Up/Shutdown**

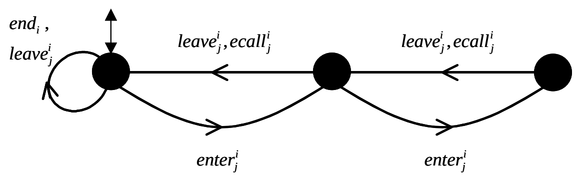
APPENDIX II
MODULAR SUPERVISORY DESIGN FOR **PLanTS**

The following (Figs. 13 through 19) are the (sub)supervisors and control laws that constitute the modular supervisory controller designed for **PLanTS**.



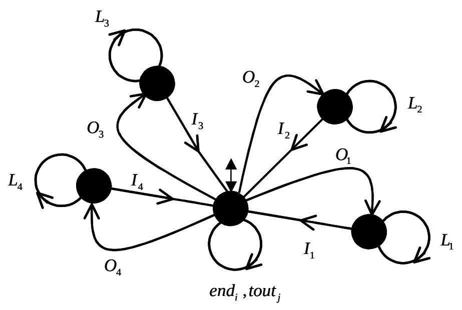
(Fixed i, Variable j)
Selfloop = $\Sigma - \{inj_i, outj_i, ras_j^i\}$

Fig. 7. **Vi_TJMSP: Vehicle Traffic Jam**



(Fixed i, Variable j)
Selfloop = $\Sigma - \{end_i, enter_j^i, leave_j^i, ecall_j^i\}$

Fig. 8. **Vi_SEATSP: Vehicle Seat-Capacity**

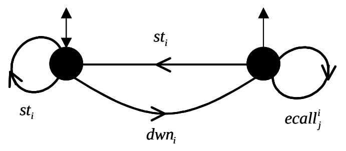


$$L_p = \{enter_n^i, enter_n^k, tout_n, ras_n^k \ | n \neq p\}$$
$$O_p = \{as_p^i, ras_p^i\}, I_p = \{enter_p^i, tout_p, ras_p^k\}$$

(Fixed i, Variable j and k, $k \neq i$)

Selfloop = $\Sigma - \{end_i, as_j^i, ras_j^i, ras_j^k, tout_j, enter_j^i, enter_j^k\}$

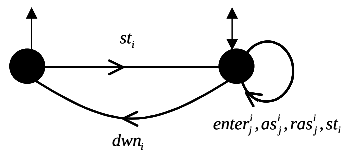
Fig. 9. **Vi_AGNSP: Vehicle Assignment-Capacity**



(Fixed i, Variable j)

Selfloop = $\Sigma - \{st_i, dwn_i, ecall_j^i\}$

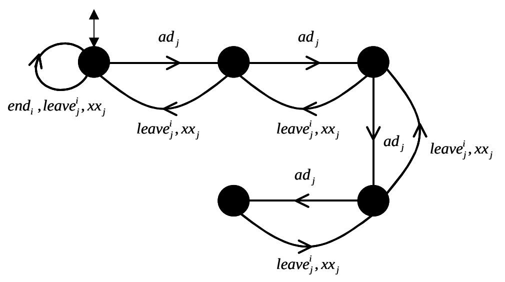
Fig. 10. **Vi_EMSP: Emergency Requests**



(Fixed i, Variable j)

Selfloop = $\Sigma - \{st_i, dwn_i, enter_j^i, as_j^i, ras_j^i\}$

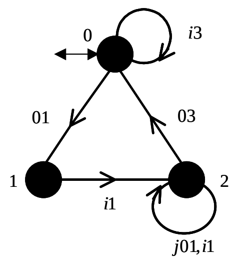
Fig. 11. **Vi_DWNSP: Vehicle Breakdown**



(Variable i and j)

Selfloop = $\Sigma - \{end_i, xx_j, ad_j, leave_j^i\}$

Fig. 12. **FLT_DILSP: Fleet Service-Diligence**

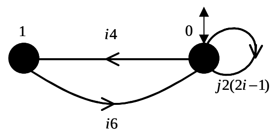


(Fixed i, Variable j)

Selfloop = $\Sigma - \{01, 03, j01, i1, i3\}$

State x	$\overline{\mathbf{V}(x)}$
0	$\{i1, j01\}$
1	$\{3, i3, j01\}$
2	$\{i3\}$

Fig. 13. **Vi_SUDCON: Service Start-Up/Shutdown Supervisors**

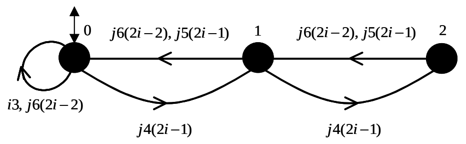


(Fixed i, Variable j)

Selfloop = $\Sigma - \{i4, i6, j2(2i-1)\}$

State x	$\overline{\mathbf{V}(x)}$
0	$\emptyset$
1	$\{j2(2i-1)\}$

Fig. 14. **Vi_TJMCON: Vehicle Traffic Jam Supervisors**

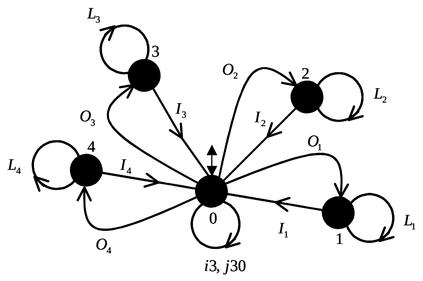


(Fixed i, Variable j)

Selfloop = $\Sigma - \{i3, j4(2i-1), j6(2i-2), j5(2i-1)\}$

State x	$\overline{\mathbf{V}(x)}$
0	$\{j5(2i-1)\}$
1	$\{i3\}$
2	$\{i3, j4(2i-1)\}$

Fig. 15. **Vi_SEATCON: Vehicle Seat-Capacity Supervisors**

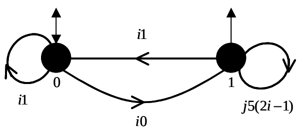


$$L_p = \{n4(2i-1), n4(2k-1), n30, n2(2k-1)\ |n \neq p\}$$
$$O_p = \{p1(2i-1), p2(2i-1)\}$$
$$I_p = \{p4(2i-1), p30, p2(2k-1)\}$$

(Fixed i, Variable j and k, $k \neq i$)
Selfloop = $\Sigma - \{i3, j1(2i-1), j2(2i-1), j2(2k-1), j30, j4(2i-1), j4(2k-1)\}$

State x	$\overline{\mathbf{V}(x)}$
0	$\{j2(2k-1), j4(2i-1), j4(2k-1)\}$
1	$\{i3, j1(2i-1), j2(2i-1)\}$
2	$\{i3, j1(2i-1), j2(2i-1)\}$
3	$\{i3, j1(2i-1), j2(2i-1)\}$
4	$\{i3, j1(2i-1), j2(2i-1)\}$

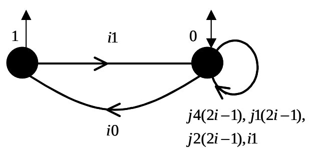
Fig. 16. **Vi_AGNCON: Vehicle Assignment-Capacity Supervisors**



(Fixed i, Variable j)
Selfloop = $\Sigma - \{i1, i0, j5(2i-1)\}$

State x	$\overline{\mathbf{V}(x)}$
0	$\{j5(2i-1)\}$
1	$\emptyset$

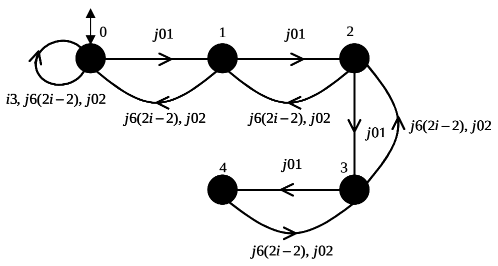
Fig. 17. **Vi_EMCON: Emergency Requests Supervisors**



(Fixed i, Variable j)
Selfloop = $\Sigma - \{i1, i0, j4(2i-1), j1(2i-1), j2(2i-1)\}$

State x	$\overline{\mathbf{V}(x)}$
0	$\emptyset$
1	$\{j1(2i-1), j2(2i-1), j4(2i-1)\}$

Fig. 18. **Vi_DWNCON: Vehicle Breakdown Supervisors**



(Variable i and j)
Selfloop = $\Sigma - \{i3, j02, j01, j6(2i-2)\}$

State x	$\overline{\mathbf{V}(x)}$
0	$\emptyset$
1	$\{i3\}$
2	$\{i3\}$
3	$\{i3\}$
4	$\{i3\}$

Fig. 19. **FLT_DILCON: Fleet Service-Diligence Supervisor**

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their critical but constructive comments on the review version of this manuscript. The responsibility for the integrity of the work, of course, remains solely with the authors.

REFERENCES

[1] M. W. P. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation Science*, vol. 29, no. 1, pp. 17–29, February 1995.

[2] D. J. Bertsimas and D. Simchi-Levi, "A new generation of vehicle routing research : Robust algorithm, addressing uncertainty," *Operations Research*, vol. 44, no. 2, pp. 286–304, March/April 1996.

[3] Shugo Katoh and Hirohiko Yanagawa, "Research and development on on-board server for Internet ITS," in *Proceedings of the IEEE Symposium on Applications and the Internet (SAINT'02 Workshops)*, 2002, pp. 35–36.

[4] Chi Hyun Park and Dong Ho Cho, "An adaptive logical link control for wireless internet service in ITS," in *Proceedings of the IEEE Vehicular Technology Conference Conference*, 1999, pp. 2213 –2217.

[5] Ness S. T. Lee, Hassan A. Karimi, and Edward J. Krakiwsky, "Road information systems : Impact of geographic information systems technology to automatic vehicle navigation and guidance," in *Proceedings of the Vehicle Navigation and Information Systems Conference*, 1989, pp. 347 –352.

[6] You Ning Wang, Russell G. Thompson, and Ian Bishop, "A GIS based information integration framework for dynamic vehicle routing and scheduling," in *Proceedings of the IEEE International Vehicle Electronics Conference*, 1999, pp. 474 –479.

[7] S. Bonora and D. Engels, "Guidelines for the use of GPS-based AVL systems in public transport fleets," in *Proceedings of the International Conference on Public Transport Electronic Systems (Conf. Publ. No. 425)*, 1996, pp. 16 –20.

[8] Hassan A. Karimi and J. Tom Lockhart, "GPS-based tracking systems for taxi cab fleet operations," in *Proceedings of the IEEE-IEE Vehicle Navigation and Information Systems Conference*, 1993, pp. 679 –682.

[9] Yilin Zhao, *Vehicle location and navigation systems*, Boston, Mass : Artech House, 1997.

[10] Norman M. Sadeh and Alexander Kott, "Models and techniques for dynamic demand-responsive transportation planning : A state-of-the-art assessment inspired by the aeromedical regulation and evacuation problem," Technical report CMU-RI-TR-96-09 , The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, Sept 1996.

[11] W. M. Wonham, *Notes on Control of Discrete-Event Systems ECE 1636F/1637S*, Systems Control Group, University of Toronto, updated 1st July 2003, http://www.control.utoronto.ca / DES.

[12] P. J. Ramadge and W. Murray Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, January 1989.

[13] P. J. Ramadge and W. Murray Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206–230, January 1987.

[14] W. Murray Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM Journal of Control and Optimization*, vol. 25, no. 3, pp. 637–659, May 1987.

[15] W. Murray Wonham and P. J. Ramadge, "Modular supervisory control of discrete-event systems," *Mathematics of Control, Signals and Systems*, vol. 1, no. 1, pp. 13–30, January 1988.

[16] Karen Rudie and W. Murray Wonham, "Supervisory control of communicating processes," in *Protocol Specification, Testing and Verification, X*, L. Logrippo, R. L. Probert, and H. Ural, Eds., pp. 243–257. Elsevier Science Publishers B. V. (North-Holland), 1990.

[17] Ratnesh Kumar, Sudhir Nelvagal, and Steven I. Marcus, "Design of protocol converters : A discrete event systems approach," in *International Workshop on Discrete Event Systems*, University of Edinburgh, UK, August 1996, pp. 7–12.

[18] B. A. Brandin, "The real-time supervisory control of an experimental manufacturing cell," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 1–14, February 1996.

[19] S. C. Lauzon, A. K. L. Ma, J. K. Mills, and B. Benhabib, "Application of discrete event system theory to flexible manufacturing," *IEEE Control Systems Magazine*, vol. 16, no. 1, pp. 41–48, February 1996.

[20] Jana Košecká and Ruzena Bajcsy, "Discrete event systems for autonomous mobile agents," *Robotics and Autonomous Systems*, vol. 12, no. 3-4, pp. 187–198, April 1994.

[21] James Lyle Peterson, *Petri Net Theory and the Modelling of Systems*, chapter 2 : Basic Definitions, Prentice Hall, Inc. Englewood Cliffs, New Jersey, 1981.

[22] Anthony Tzes, Seongho Kim, and William R. McShane, "Applications of Petri networks to transportation network modeling," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 2, pp. 391 –400, May 1996.

[23] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice Hall. International Series in Computer Science, 1985.

[24] Giulio Erberto Cantarella Ennio Cascetta, "Modelling dynamics in transportation networks: State of the art and future developments," *Simulation Practice and Theory*, vol. 1, no. 2, pp. 65–91, November 1993.

[25] M.P. Michael P. Spathopoulos and M. A. de Ridder, "Modelling and control of a transport system," in *Proceedings of the IEE International Conference on Control*, Warwick, U.K, March 1994, pp. 48 –53.

[26] Seung Pil Yoo, Doo Yong Lee, and Hyoung Il Son, "Design and verification of supervisory controller of high-speed train," in *Proceedings of the IEEE International Symposium on Industrial Electronics*, Pusan, Korea, June 2001, pp. 1290 –1295.

[27] Harilaos N. Psaraftis, "Dynamic vehicle routing : Status and prospects," *Annals of Operations Research*, vol. 61, pp. 143–164, 1995.

[28] Kiam Tian Seow, Michel Pasquier, and Mun Li Hong, "Supervisory control of transport-operations processes," in *Proceedings of The Third World Multiconference on Systemics, Cybernetics and Informatics (SCI'99) and the Fifth International Conference on Information Systems Analysis and Synthesis (ISAS'99)*, Orlando, Florida, U.S.A, 1999, pp. 185–192 (in Vol. VII).

[29] Kiam Tian Seow, Michel Pasquier, and Mun Li Hong, "A formal design methodology for land-transport operations," in *Proceedings of the IEEE/IEEJ/JSAI Conference on Intelligent Transportation Systems*, Tokyo, Japan, 1999, pp. 110–115.

[30] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Reading, MA : Addison-Wesley, 1979.

[31] Harilaos N. Psaraftis, "Dynamic vehicle routing problems," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds., pp. 223–248. Elsevier Science Publishers B.V. (North Holland), 1988.

[32] H. N. Djidjev, G. E. Pantziou, and C. D. Zaroliagis, "On-line and dynamic algorithms for shortest path problems," in *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science*. March 1995, pp. 193–204, Springer Verlag, Berlin.