Coloured Petri Net Model of a Simple Runway

Á. Kovács, E. Németh, K. M. Hangos

Research Report SCL-001/2004

Contents

List of Figures

Coloured Petri Net Model of a Simple Runway

Ákos Kovács[1] Erzsébet Németh[2,3]
E-mail: akos@rht.bme.hu E-mail: nemethe@scl.sztaki.hu

Katalin M. Hangos[2,3]
E-mail: hangos@scl.sztaki.hu

[1]Department of Aircraft and Ships, Budapest University of Technology and Economics
H-1521 Budapest, P.O. Box 91, Hungary

[2]Department of Computer Science, University of Veszprém
H-8201 Veszprém, P.O. Box 158, Hungary

[3]Process Control Research Group,
Systems and Control Research Laboratory, Computer and Automation Institute HAS
H-1518 Budapest, P.O. Box 63, Hungary

Abstract

A timed stochastic coloured Petri net (CPN) model of a single runway is presented in this paper that is capable of analyzing the effect of the availability of the taxiways on the capacity of the runway and on the timing of a given schedule. The developed CPN model is governed by elementary Air Traffic Control (ATC) principles (one aircraft at a time on runway, arrivals priority on departures), and takes the effect of the aircraft categories and their landing weights into account.

The model is realized by using the Design/CPN program package. The model has been verified and validated by using runway occupancy times generated by REDIM.

1 Introduction

The demand for air transport has been increasing rapidly over the years and this trend is expected to continue in the future. Therefore, it is of great practical importance to develop, verify and validate simple, yet powerful dynamic models that help in design and retrofit air traffic management systems (ATM). ATM is divided into two parts: air traffic flow management (ATFM), and air traffic services (ATS) as air traffic control (ATC) where the subject of this paper belongs to.

One of the recent problems in ATC is to minimize the delays of arriving and departing flights, where the dynamic modelling of runways and scheduling strategies based thereon are the critical issues. There are different approaches to this problem: a decision support system based on a stochastic analytic model for the estimation of the capacity envelope of an airports runway system is reported in [10].

Besides of the analytical and mathematical programming techniques [1] applied to ATM problems, combinatorial methods for scheduling various operations on airfields are also reported [2]. It has recently been anticipated [8] that "the relatively recent formalization of concepts of discrete event systems (DES) originated by Ho [4] may well lead to more powerful analysis techniques suited to ATC analysis".

Coloured Petri nets (CPNs) [7, 5] belong to the area of discrete event system methodology, where CPNs are well known for their capability in simulating and analyzing discrete event systems. There are useful extensions of the original CPN concept that includes its timed and stochastic versions.

The aim of our paper is to propose a simple dynamic model that describes the traffic flow of a single runway (RWY) due to schedule (i.e. estimated times of arrivals and departures). The model is governed by elementary ATC principles (one aircraft at a time on RWY, arrivals priority on departures), and it is a subject of random disturbances (such as weather conditions, landing weights etc.). Therefore, timed stochastic CPNs have been selected as modelling and simulation tools to develop and analyze a dynamic model of a simple runway.

The organization of the paper is as follows. First the runway model is described in the next section, then the CPN model of the simple runway is presented in section 3. The simulation results are given and discussed in section 4.

The list of the abbreviations is found in the Appendix A.

2 Description of the runway

The runway to be modelled is simple and yet has a potential containing most of the elements important in runway dynamics.

2.1 Assumptions

In order to obtain a relatively simple model for simulation and dynamic analysis purposes, the following modelling assumptions are made:

- Weather is not considered directly.
- The differences between A/C are described in terms of three threshold speed categories.
- No wake turbulence, therefore no wake turbulence separation is taken into account.
- It is assumed that the time value of reaching a definite position by an A/C can be precalculated.
- Only basic ATC laws are applied.
- Only visual flight rules (VFR) operations are considered.
- Pilots are assumed being familiar with airport configuration (i.e. with the location of rapid exit taxiways).

2.2 Model elements

In order to model operations around an airport, we need a couple of elements to work with. In this case the most important elements are:

- Runway (RWY)
- Taxiways (TWY)
- Aircraft (A/C)
- Rules that govern the interaction between A/C and use of the RWY

The characteristic properties of each of the model elements are as follows.

Runway (RWY)

A single 2500 m runway is considered with two 90° TWY on both end and two rapid exit taxiways (RETs) located at 1075 m and 1650 m from approach end threshold. Gradients are assumed to be 0° all over the runway (see Fig. 1).

This configuration was generated by the REDIM simulator [9] developed by Virginia Tech. Input data were runway length, weather conditions (wind, temperature, surface conditions) number and configuration of desired taxiways (two RETs of 30 degree Federal Aviation Administration (FAA) modified exits [11], and one 90 degree taxiway), entry speed of taxiways (30 m/s, 25 m/s and 5 m/s respectively) and a suitable traffic mix.

The selected *traffic mix* contains the following aircraft with percentage: C172 5%, C182 5%, PA28-236 5%, C421 10%, CRJ200 15%, Saab 340 15%, A300-600 15%, B737-800 20%, MD87 10%.

The output is a $2500m$ runway with minimum average weighted runway occupancy times for the given traffic mix.

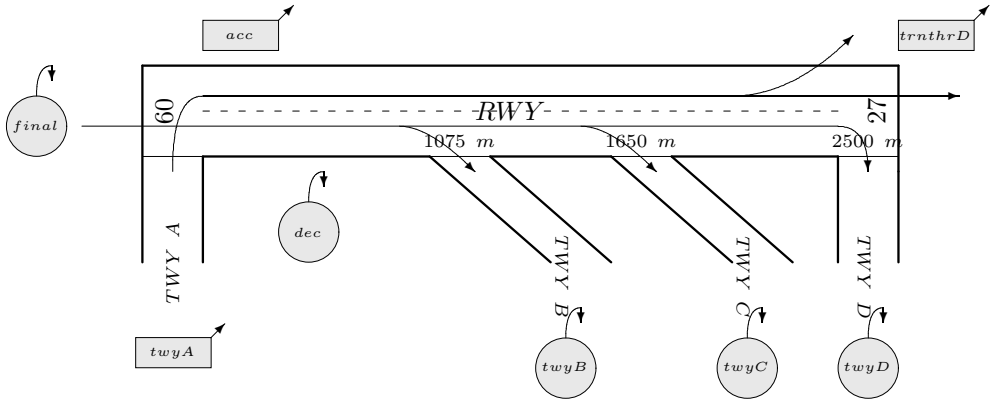

Figure 1: The runway

Aircraft (A/C)

The difference between aircraft is based on ICAO threshold speed categories (A to E). Only aircraft with categories A, B and C are considered.

Governing rules

The following rules are used to control the interactions between A/C and the use of the runway.

1. Arrivals have priority on departures.
2. A landing aircraft will not normally be permitted to cross the runway threshold on its final approach until the preceding departing A/C has crossed the end of the runway, or has started a turn, or until all preceding landing A/C are clear off the RWY.

Runway occupancy times (ROTs)

ROTDs (ROT for departures) for each category are calculated with use of REDIM averaging ROTDs of three representing A/C of the category for each taxiway possible. The ROTs can be found in Appendix-B. Calculation are made on sea level, no wind, $15°C$ and dry surface. A/C landing weight factor was assumed to be normally distributed with mean value of 0.5 and standard deviation 0.2. It is an important factor, as it greatly influences A/C threshold speed and ROTD.

ROTAs (ROT for arrivals) are less accurate, but still present for the categories containing line up and take-off roll times. They are considered as mean values with relatively large deviations.

3 Coloured Petri nets

In this section, we summarize the basic notations found in the literature about the coloured Petri nets and timed coloured Petri nets. For first, the general coloured case will be considered, while the second part concerns with the timed extension.

3.1 Coloured Petri nets

Coloured Petri nets (CPNs) [7] belong to the area of discrete event system methodology. CPN is well known for its capability in modelling discrete event systems.

The structure of a Petri net is a bipartite directed graph describing the structure of a discrete event system, while the dynamics of the system is described by the execution of the Petri net. A Petri net is coloured if the tokens are distinguishable. According to the formal definition of CPNs [5] a coloured Petri net model is a nine-tuple

$$CPN = (\Sigma, P, T, A, N, C, G, E, IN) \tag{1}$$

satisfying the following requirements:

(i) Σ is a finite set of non-empty types, called *colour sets*

(ii) P is a finite set of *places*

(iii) T is a finite set of *transitions*

(iv) A is a finite set of *arcs* such that $P \cap T = P \cap A = T \cap A = \emptyset$

(v) $N : A \rightarrow P \times T \cup T \times P$ is a *node function*

(vi) $C : P \rightarrow \Sigma$ is a *colour function*

(vii) G is a *guard function*. It is defined from T into expressions such that
$\forall t \in T : [Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \Sigma]$

(viii) E is an *arc function*. It is defined from A into expressions such that
$\forall a \in A : [Type(E(a)) = C(p(s))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$ where $p(a)$ is the place of $N(a)$ and C_{MS} denotes the set of all multi-sets over C

(ix) IN is an *initialization function*. It is defined from P into expressions such that
$\forall p \in P : [Type(IN(p)) = C(p(s))_{MS} \wedge Var(IN(p)) = \emptyset]$

where:
$Type(expr)$ denotes the type of an expression,
$Var(expr)$ denotes the set of variables in an expression,
$C(p)_{MS}$ denotes a multi-set over $C(p)$.

A *binding* of a transition t is a function b defined on $Var(t)$, such that:

(i) $\forall v \in Var(t) : b(v) \in Type(v)$,

(ii) $G(t)\langle b \rangle$ denotes the evaluation of the guard expression $G(t)$ in the binding b.

A *token element* is a pair (p, c) where $p \in P$ and $c \in C(p)$. A *binding element* is a pair (t, b) where $t \in T$ and $b \in B(t)$. By $B(t)$ denotes the set of all bindings for t. The set of all token elements is denoted by TE while the set of all binding elements is denoted by BE.

A *marking* is a multi-set over TE while a *step* is a non-empty and finite multi-set over BE. The *initial marking* M_0 is the marking which is obtained by evaluating the initialization expressions.

A *transition is enabled* if each of its input places contain the multi-set specified by the input arc inscription (possibly in conjunction with the guard), and the guard evaluates to true. When a transition is enabled it may *occur*, and this means that the tokens are removed from the input places and added to the output places of the occurring transitions. The number and colour of the tokens are determined by the arc expressions, evaluated for the occurring bindings.

A *finite occurrence sequence* is a sequence of markings and steps: $M_1[Y_1\rangle M_2[Y_2\rangle M_3 \ldots M_n[Y_n\rangle M_{n+1}$, such that $n \in \mathbb{N}$. A marking M'' is *reachable* from a marking M' if and only if exists a finite occurrence sequence having M' as start marking and M'' as end marking, i.e., if and only if for some $n \in \mathbb{N}$ there exists a sequence of steps $Y_1 Y_2 \ldots Y_n$ such that: $M'[Y_1 Y_2 \ldots Y_n\rangle M''$. We then also say that M'' is reachable from M' in n step. The set of markings which are reachable from M' is denoted by $[M'\rangle$.

3.2 Timed coloured Petri nets

Most applications of CPNs are used to investigate the logical correctness of a system. The CPN extended by time [6] gives a possibility to describe the dynamic properties of a system in the time space.

3.2.1 Introduction to timed coloured Petri nets

The time concept of CPNs is based on the introduction of a *global clock*. The clock values represent the *model time* and they be discrete (e.g. integers). Each token carry a *time value*, also called a *time stamp*. The time stamp describes the earliest model time at which the token can be used, i.e., removed by the occurrence of a binding element.

3.2.2 Timed multi-sets

Assume that we have a set of *time values* R, which is a subset of $\mathbb{Z}$ closed under $+$ and containing 0. Timed multi-sets are a modification of ordinary multi-sets.

A *timed multi-set* tm, over a non-empty set S, is a function $tm \in [S \times R \to \mathbb{N}]$ such that the sum:

$$tm(s) = \sum_{r \in R} tm(s, r)$$

is finite for all $s \in S$. The non-negative integer $tm(s)$ is the *number of appearances* of the element s in the timed multi-set tm. The list:

$$tm[s] = [r_1, r_2, \ldots, r_{tm(s)}]$$

is defined to contain the time values $r \in R$ for which $tm(s, r) \neq 0$. Each r appears $tm(s, r)$ times in the list, which sorted such that $r_i \leq r_{i+1}$ for all $i \in 1..tm(s) - 1$.

We usually represent the timed multi-set tm by a formal sum:

$$\sum_{s \in S} tm(s)`s \ @tm[s] \ .$$

The set of all timed multi-set over S is denoted by S_{TMS}.

Comparison of timed multi-sets is defined in the following way, for all $tm_1,\ tm_2 \in S_{TMS}$:

(i) $tm_1 \leq tm_2 \ = \ \forall s \in S : \ tm_1[s] \leq tm_2[s]$

(ii) When $tm_1 \leq tm_2$ we also define subtraction:
$tm_2 - tm_1 \ = \ \sum_{s \in S}(tm_2(s) - tm_1(s))`s \ @(tm_2[s] - tm_1[s])$.

3.2.3 Formal definition of timed coloured Petri nets

A timed non-hierarchical CPN is a tuple:

$$TCPN = (CPN, R, r_0)$$

such that:

(i) CPN satisfy the requirement of a non-hierarchical CPN in Eq. 1, when in (viii) and (ix) we allow the type of $E(a)$ and $I(p)$ to be a timed or an untimed multi-set over $C(p(a))$ and $C(p)$, respectively.

(ii) R is a set of *timed values*, also called *time stamps*. It is a subset of $\mathbb{R}$ closed under $+$ and containing 0.

(iii) r_0 is an element of R, called the *start time*.

Timed CPN models often contain one or more colour sets S which are *untimed*. This means the tokens of type S are required to be always available, independently of any time constraints.

The set of bindings $B(t)$, token elements TE, binding element BE and steps $\mathbb{Y}$ are defined in exactly the same way as for an untimed CPN.

A *marking* is a timed multi-set over TE. The *initial marking* M_0 is the marking obtained by evaluating the initialization expressions:

$$\forall p \in P : \ M_0(p) = I(p)_{r_0} \ .$$

A *state* is a pair (M, r) where M is a marking and r is a time value. The *initial state* is the pair (M_0, r_0).

The set of all markings and states are denoted by $\mathbb{M}$ and $\mathbb{S}$, respectively.

A step Y is *enabled* in a state (M_1, r_1) at time r_2 iff the following properties are satisfied:

(i) $\forall p \in P : \ \sum_{(t,b) \in Y} E(p,t)\langle b \rangle_{r_2} \leq M_1(p)$.

(ii) $r_1 \ \leq \ r_2$.

(iii) r_2 is the smallest element of R for which there exists a step satisfying (i) and (ii)

4 CPN model of the runway

Because of the stochastic and timed discrete event nature of the problem, timed stochastic coloured Petri nets (CPNs) [7, 5] are chosen as a modelling tool. The structure of a Petri net is a bipartite directed graph describing the structure of a discrete event system, while the dynamics of the system is described by the execution of the Petri net. A Petri net is coloured if the tokens are distinguishable.

For real systems it is often important to describe the temporal behaviour of the system, i.e. we need to model durations and delays. The CPN extended by time [6] gives a possibility to describe the dynamic properties of a system in the time space. The time concept of CPNs is based on the introduction of a *global clock*. The clock values represent the *model time* and were chosen to be discrete (e.g. integers). Each token carry a *time value*, also called a *time stamp*. The time stamp describes the earliest model time at which the token can be used, i.e., removed by the occurrence of a binding element.

4.1 The inputs, outputs and structure of the model

As described in section 2, the runway contains 2 rapid exit taxiways which can be possibly closed.

The *inputs* of the runway model are:

- the desired original scheduling,
- the occupancy time of runways,
- which rapid exit taxiways are open.

The *outputs* of the simulation are:

- occupancy graph,
- text file with simulation results.

Stochastic disturbances

The aim of the simulation includes to handle the effect of uncertainty of runway occupancy time (due to weather, landing weight, etc.) on the runway capacity. So the developed model puts some uncertainty into the desired original scheduling. In addition, there are possibilities to put a new aircraft into the queue, and change the status of any of rapid exit taxiways during the simulation .

CPN realization

The timed stochastic CPN model of the runway that obeys our modelling assumptions has been built in Design/CPN (its most recent name is CPN Tools) [3] program package. The model is seen in Fig. 2 as it appears in the program package.

The model is divided into 2 parts: the initialization and the scheduling part. The initial block is initializing the output chart, the output file and reading the input scheduling from the input file. The main part of the model realizes the simulation of the runway's operation (i.e. simulates aircraft moving on the runway as function of time).

The modelling time is handled by the CPN simulator. One unit in the model time is set to be one second.

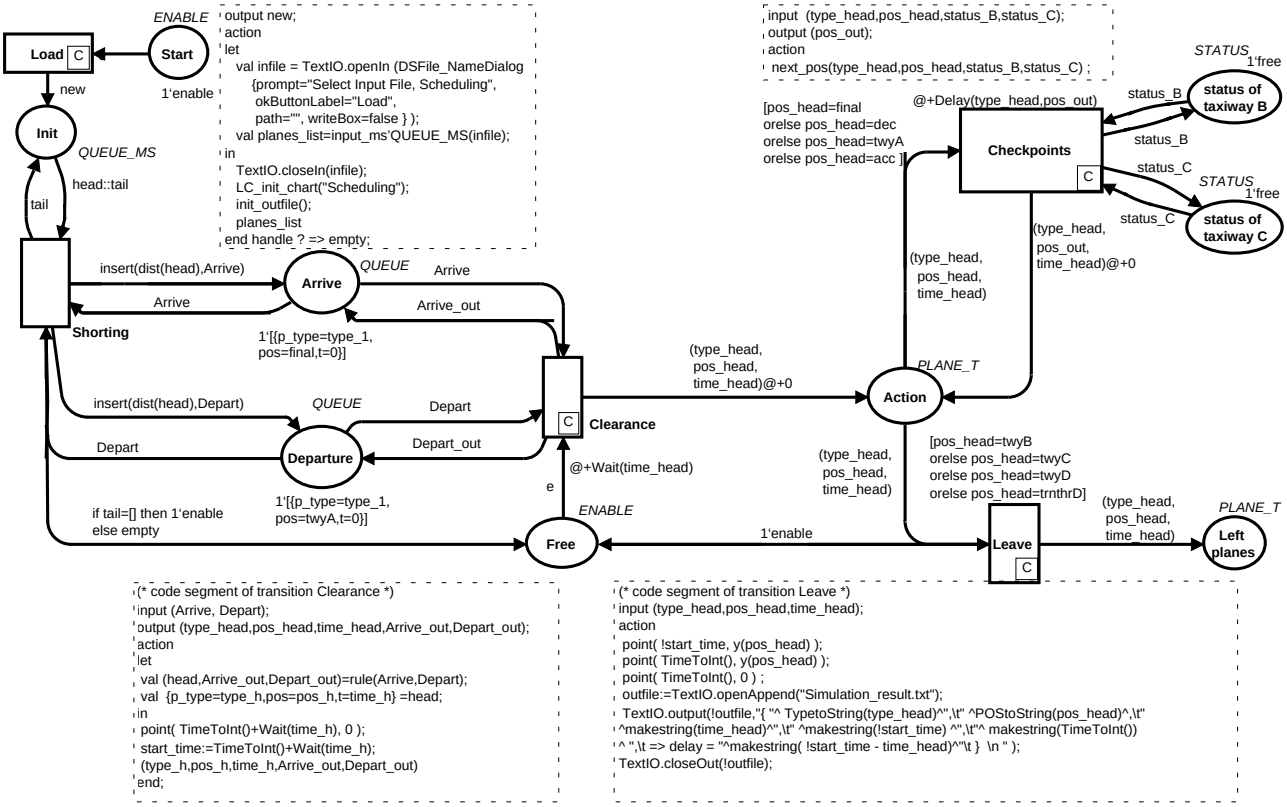

Figure 2: The CPN model of the runway

4.2 CPN model elements

To model the runway traffic as a discrete event system, we need to define events that are relevant, such as clearances (for landing, take-off) and passing some positions by the aircraft. Having considered the modelling assumptions and goals, a timed stochastic CPN model with the elements below have been constructed.

Tokens

Aircraft are described as 3-tuples `(type,pos,time)` by CPN colour with time stamp, where the `type` determines the type of the plane, `pos` defines the position of the aircraft on the runway, and `time` contains the initial starting time.

Aircraft are classified by their category as `type_1`, `type_2`, `type_3` that is stored in attribute `type`.

The position of an aircraft is indicated by its `pos` attribute, that takes different values for landing and departing aircraft.

The positions for arriving (see in Fig. 1) are:

- `final`: plane on its final approach until the threshold (THR)
- `dec`: plane after passing the THR and decelerating
- `twyB`: plane left the runway on taxiway B

- `twyC`: plane left the runway on taxiway C
- `twyD`: plane left the runway on taxiway D

The positions for departing are:

- `twyA`: plane on taxiway A before rolling out to runway
- `acc`: plane rolling onto runway and accelerating
- `trnthrD`: plane in air has started a turn or has flown over the opposite THR

4.3 Places and transitions in the CPN model

The main part of the model contains the following elements:

- places: `Arrive`, `Departure`, `Free`, `Action`, `status of taxiway B`, `status of taxiway C`, `Left planes`;
- transitions: `Clearance`, `Checkpoints`, `Leave`.

The remaining part of the model is the initial block, i.e. :

- places: `Start`, `Init`;
- transitions: `Load`, `Shorting`.

4.3.1 The initial block

The initial block contains 2 transitions. Explaining them is the best way to understand the model.

Transition `Load`

The role of the transition is to initialize the chart, read the schedule from the input file and initialize the output file.
1 input place for `Load`:

- `Start`: 1 token in this place indicates that the model is ready to start.

1 output place for `Load`:

- `Init`: after firing the transition it contains the plane queue from the input file without ordering, if the initialization is successful.

Transition `Shorting`

The role of the transition is to put a disturbance on the starting time, selecting and sorting the planes from place `Init` to either place `Arrive` or place `Departure` for the scheduling by using the following rules:

(i) Landing planes (i.e. planes with position `final`) are inserted into the landing plane queue contains the place `Arrive`.

(ii) Taking off planes (i.e. planes with position `twyA`) are inserted into the landing plane queue contains the place `Departure`.

(iii) The sorting is based on scheduled time of the A/C.

3 input places for `Shorting`:

- `Init`: token in this place denotes the remaining orderless plane queue.
- `Arrive`: token in this place containing the ordered landing plane queue.
- `Departure`: token in this place containing the ordered taking off plane queue.

4 output places for `Shorting`:

- `Arrive`: after firing transition it contains the ordered landing plane queue.
- `Departure`: after firing transition it contains the ordered taking off plane queue.
- `Init`: after firing transition it contains the remained orderless plane queue.
- `Free`: after firing transition if the orderless plane queue is empty (i.e. all planes in the ordered queue) this place is given a token `enable` which means the runway is free.

4.3.2 The main block

The main part of the model contains 3 transitions. Explaining them is also the best way to understand the model.

Transition `Clearance`

The role of the transition is to give a clearance to the selected element of the ordered plane queues from place `Arrive` and place `Departure` in the corresponding time if the runway is free (i.e. place `Free` has a token). For the selection the following rules are used:

(i) Priority of arrivals: landing planes (i.e. planes with position `final`) has priority over planes taking off (i.e. planes with position `twyA`).

(ii) A clearance for a landing plane is given before a clearance for a plane intends to take-off with earlier start time, if the plane taking-off would vacate the runway at the time the landing plane arrives over the threshold.

3 input places for `Clearance`:

- `Arrive`: token in this place containing the ordered landing plane queue.
- `Departure`: token in this place containing the ordered taking off plane queue.
- `Free`: token in this place denotes the runway is free.

3 output places for `Clearance`:

- `Arrive`: after firing transition it contains the remaining ordered landing plane queue.
- `Departure`: after firing transition it contains the remaining ordered taking off plane queue.
- `Action`: after firing transition it contains a plane which has clearance.

Transition `Checkpoints`

The role of the transition is to calculate and change the position of the plane on the runway with the corresponding runway occupancy time.
3 input places for `Checkpoints`:

- `Action`: token in this place containing a plane which is somewhere on the runway.
- `status of runway B`: token in this place denotes the status (free or nofree) of the rapid exit taxiway B.
- `status of runway C`: token in this place denotes the status (free or nofree) of the rapid exit taxiway C.

3 output places for `Checkpoints`:

- `Action`: after firing transition it contains the plane which is in the next checkpoint on the runway.
- `status of runway B`: after firing transition it contains the status (free or nofree) of the rapid exit taxiway B.
- `status of runway C`: after firing transition it contains the status (free or nofree) of the rapid exit taxiway C.

Transition `Leave`

The role of the transition is to select (i.e. filtering is solved by the guard of the transition) the plane from place `Action` which is left the runway and set the runway status in place `Free` to `free`. During the transition firing the output file is updated by the simulation results and the chart is also updated.
1 input place for `Leave`:

- `Action`: token in this place containing a plane which is somewhere on the runway.

2 output places for `Leave`:

- `Left planes`: after firing transition it contains the planes which are left the runway.
- `Free`: after firing transition it contains the status `free` of the runway.

4.4 Declaration and code segment of the CPN model

The code segment contains the declaration and the description of the functions used in the CPN model. The details of the code segment is found in Appendix C.

5 Simulation results

For the CPN model simulation Design/CPN program package is used.

Aim of the simulation was to calculate the effect on runway capacity of various events. These are closing of taxiways, displaced threshold and changing A/C sequence in a given schedule.

5.1 Scheduling strategies

Due to uncertainty of arriving exactly on estimated time to the threshold, the input schedule of the model is calculated from the original schedule by adding or subtracting a random number to the times of estimated arrival on the threshold.

The input schedule contains A/C with estimated time of reaching the threshold (for arrivals) and scheduled time of reaching the runway on taxiway A. The priority of arrivals law is applied with use of the philosophy *departures between arrivals not arrivals between departures.*

The model gives a clearance the latest time possible i.e. one clearance is valid at a time and a new clearance is given *after* the preceding A/C is off the runway.

Clearances

The scheduling method at a given model time is to select the next time an A/C arriving at the threshold and calculates, whether departing A/C scheduled to depart earlier are clear off the runway before this time. It is done by adding the time the A/C vacates the runway for departing (that depends on category) to the actual time or to the scheduled time of the aircraft reaching the runway on taxiway A, whichever is greater. When this value is smaller or equal to the estimated time the landing A/C reaching the threshold, the clearance is issued to the departing aircraft, otherwise not.

For example let us consider two aircraft: an A/C of category B scheduled to depart at time 140 and an arrival A/C estimated to reach the threshold at time 215. If the runway is free at time 140 then a clearance will be given to the departing category B aircraft because it vacates the runway for 70 seconds, and at time $140 + 70 = 210$ it will be off the runway, 5 seconds before the landing A/C expected to arrive at the threshold. However, if the preceding A/C leaves the runway at time 150 then the clearance will not be given to the departing A/C as it would vacate the runway until time $150 + 70 = 220$ and this would violate our simple *one at a time on runway* rule for five seconds.

5.2 The simulation output

Occupancy graph

One output of the simulation is a graph showing the runway occupancy vs time (see Fig. 3 - 6). It shows either low (i.e. runway is free in time interval on horizontal axis) or high level (i.e. runway is occupied). Runway occupancy is further divided into four sublevels indicating where the aircraft left the runway. $trnthrD$ is for departing aircraft, $twyB$, $twyC$ and $twyD$ indicates the exit used by landing aircraft.

A/C category	Arrive/Departure	Scheduled time
B	Arrive	60
B	Departure	110
A	Arrive	220
C	Departure	275
C	Arrive	405
A	Departure	490
B	Arrive	580

Table 1: The input data of the simulation

Output text file

The other output is a text file including the following data:

- type of the aircraft
- position the aircraft left the runway (e.g. taxiway B)
- scheduled operation time (e.g. scheduled time of departure)
- runway occupancy time (either ROTA or ROTD)
- time of clearance (e.g. clearance was given at time 160 though scheduled time was 140)
- delay (i.e. the difference between the scheduled and the real clearance time)

5.3 Simulation results

For illustrate the simulation some simulation result is following. The input data of the simulation can be seen in Table 1:

The corresponding CPN initial value is

1‘[p_type=type_2, pos=final, t= 60,
p_type=type_2, pos=twyA, t=110,
p_type=type_1, pos=final, t=220,
p_type=type_3, pos=twyA, t=275,
p_type=type_3, pos=final, t=405,
p_type=type_1, pos=twyA, t=490,
p_type=type_2, pos=final, t=580]

Four simulations outputs is shown in the following. The presented figures show the effect on runway capacity of *opening* or *closing* two rapid exit *taxiways* (RETs).

Case 1: Both RETs are opened When both RETs are opened the following output file is generated.

```
{ plane type, runway leaving position, scheduling time, real starting time,
  runway leaving time, => delay = difference from the scheduling }
 { type_2,  twyC    ,    55,     55,    112,     => delay = 0    }
 { type_2,  trnthrD ,   110,    112,    192,     => delay = 2    }
 { type_1,  twyB    ,   220,    220,    272,     => delay = 0    }
```

```
{ type_3,  trnthrD ,   275,    275,    370,     => delay = 0    }
{ type_3,  twyC    ,   419,    419,    473,     => delay = 0    }
{ type_1,  trnthrD ,   490,    490,    555,     => delay = 0    }
{ type_2,  twyB    ,   599,    599,    647,     => delay = 0    }
```

The corresponding occupancy graph can be seen in Fig. 3.

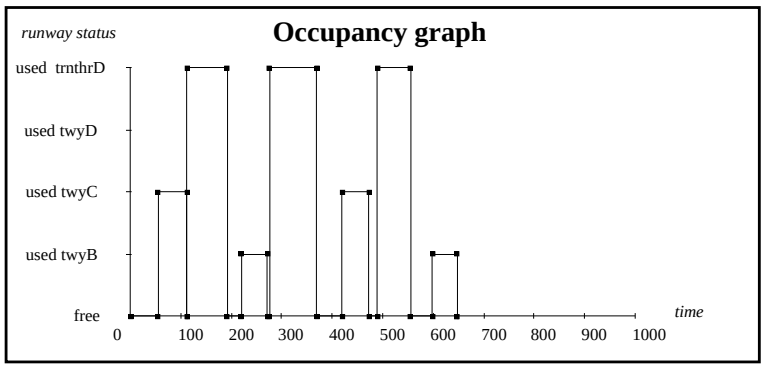

Figure 3: Case 1: Occupancy graph when all rapid exit taxiways are opened

Case 2: TWY B is opened, TWY C is closed When the TWY B is opened and TWY C is closed the following output file is resulted.

```
{ plane type, runway leaving position, scheduling time, real starting time,
  runway leaving time, => delay = difference from the scheduling }
 { type_2,  twyB    ,    61,     61,    109,     => delay = 0    }
 { type_2,  trnthrD ,   110,    110,    190,     => delay = 0    }
 { type_1,  twyB    ,   236,    236,    288,     => delay = 0    }
 { type_3,  trnthrD ,   275,    288,    383,     => delay = 13   }
 { type_3,  twyD    ,   378,    383,    475,     => delay = 5    }
 { type_1,  trnthrD ,   490,    490,    555,     => delay = 0    }
 { type_2,  twyD    ,   623,    623,    722,     => delay = 0    }
```

Fig. 4 shows the occupancy graph of case 2.

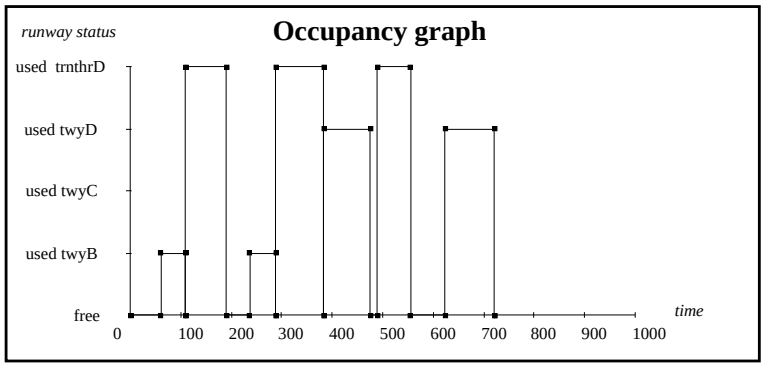

Figure 4: Case 2: Occupancy graph when TWY B is opened, TWY C is closed

Case 3: TWY B is closed, TWY C is opened When the TWY B is closed and TWY C is opened the following output file is generated.

```
{ plane type, runway leaving position, scheduling time, real starting time,
  runway leaving time, => delay = difference from the scheduling }
 { type_2,  twyC    ,    57,     57,    114,     => delay = 0    }
 { type_2,  trnthrD ,   110,    114,    194,     => delay = 4    }
 { type_1,  twyC    ,   227,    227,    288,     => delay = 0    }
 { type_3,  twyD    ,   371,    371,    463,     => delay = 0    }
 { type_3,  trnthrD ,   275,    463,    558,     => delay = 188  }
 { type_2,  twyC    ,   582,    582,    640,     => delay = 0    }
 { type_1,  trnthrD ,   490,    640,    705,     => delay = 150  }
```

In this case the corresponding occupancy graph (see in Fig. 5) is given.

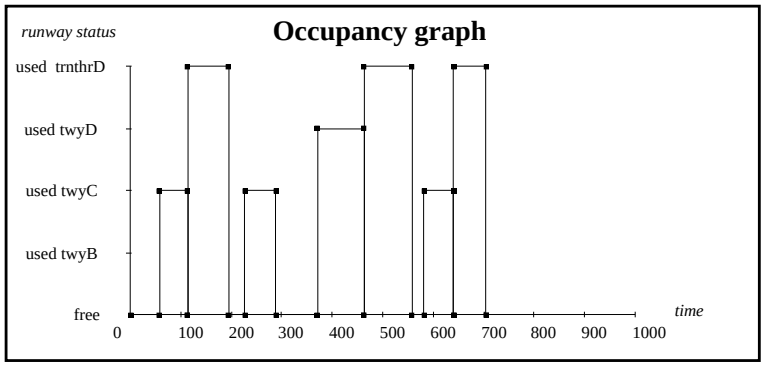

Figure 5: Case 3: Occupancy graph when TWY B is closed and TWY C is opened

Case 4: Both RETs are closed When both RETs are closed the following output file is generated.

```
{ plane type, runway leaving position, scheduling time, real starting time,
  runway leaving time, => delay = difference from the scheduling }
 { type_2,  twyD    ,    57,     57,    158,     => delay = 0    }
 { type_1,  twyD    ,   209,    209,    309,     => delay = 0    }
 { type_2,  trnthrD ,   110,    309,    389,     => delay = 199  }
 { type_3,  twyD    ,   444,    444,    535,     => delay = 0    }
 { type_2,  twyD    ,   523,    535,    633,     => delay = 12   }
 { type_3,  trnthrD ,   275,    633,    728,     => delay = 358  }
 { type_1,  trnthrD ,   490,    728,    793,     => delay = 238  }
```

The corresponding occupancy graph can be seen in Fig. 6.

5.4 Simulation experiments

The proposed CPN model was verified against qualitative engineering expectations by using simulation experiments. The effect of closing one of or both rapid exit taxiways displaced threshold is used for this purpose.

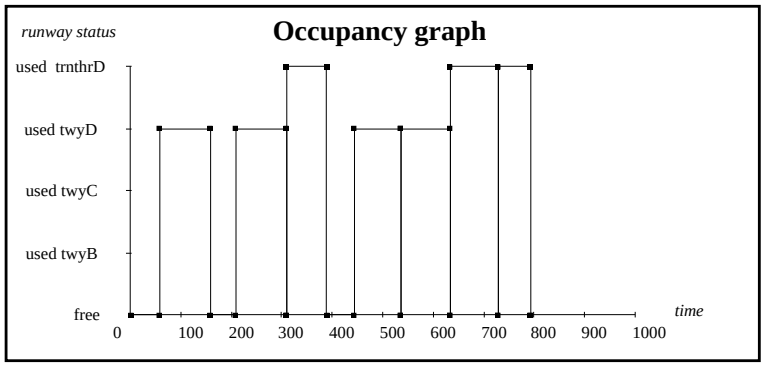

Figure 6: Case 4: Occupancy graph when all rapid exit taxiways are closed

Closing of rapid exit taxiways

The presented figures show the effect on runway capacity of *closing* two rapid exit *taxiways* (RETs). It can be seen that delays on departure A/C increased more dramatically then for arrivals. It is because the schedule was made with assuming full runway capacity, and the priority of arrivals law results in a great queue on taxiway A. This would not necessarily happen in a real situation.

If this situation can be foreseen then a new schedule might be set, and the model can then be used to determine whether the reduced capacity is enough to handle it.

If it was not foreseen then these results can be interpreted as realistic for short period of time i.e. for a couple of hours.

Displaced threshold

Studying the effect of *displaced threshold* is an other interesting issue. Expectations are not straightforward since the effect on ROTA is not simple to determine. Displaced threshold however has not much effect on ROTD.

The method of examination is to get new input ROTA data for each A/C category from REDIM with the new runway parameters, and run the model with these new times.

Results are also greatly influenced by the schedule and the type of aircraft using the runway.

Other effects of interests

Effect of other *weather conditions* can be investigated similar way as the effect of displaced threshold. In this case much more parameters can or need to be changed in REDIM: wind speed and direction, surface conditions, runway exit speeds (for new surface conditions), temperature etc.

The effect of *resequencing* a given schedule is not enough realistic since the model do not apply inter arrival and departure separations. However, the model might show some effect on different sequences based on ROTs. A further developed model containing separation conditions would be more sensitive for this case.

6 Conclusion

A timed stochastic coloured Petri net (CPN) model of a single runway is proposed in this paper that is capable of analyzing the effect of the runway status, i.e. the availability of the taxiways, the displaces threshold on the capacity of the taxiway and on the timing of a given schedule.

The model has been verified against qualitative engineering expectations by using simulation experiments and runway occupancy times generated by REDIM.

Work in progress includes the analysis of the effect of weather conditions and resequencing of a given schedule. Further work will be directed towards development of on-line re-scheduling based on the available actual runway configuration and weather conditions.

Acknowledgement

E. Németh, K. M. Hangos acknowledge support funding from the Hungarian National Research Fund through grant no. T042710 which is gratefully acknowledged.

References

[1] G. Andreatta, L. Brunetta, and G. Guastalla. The flow management problem: recent computational algorithms. *Control Engineering Practice*, 6:727–733, 1998.

[2] Y. Cheng. Solving push-out conflicts in apron taxiways of airports by a network-based simulation. *Computers Ind. Engng*, 34(2):351–369, 1998.

[3] Design/CPN v4.0.5. *Computer Tool for Coloured Petri Nets, 2002.* http://www.daimi.au.dk/designCPN/.

[4] Y. C. Ho. *Discrete event dynamic systems.* IEEE Press, 1992.

[5] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use Volume 1.* Springer-Verlag, 1992.

[6] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use Volume 2.* Springer-Verlag, 1995.

[7] K. Jensen and G. Rosenberg. *High-level Petri nets: Theory and Application.* Springer-Verlag, 1991.

[8] S. Kahne. Research issues in the transition to free flight. *Annual Reviews in Control*, 24:21–29, 2000.

[9] REDIM v2.1. *Runway Exit Design Interactive Model, 2001.* http://128.173.204.63/redim.html.

[10] M.A. Stamatopoulos, K.G. Zografos, and A.R. Odoni. A decision support system for airport strategic planning. *Transportation Research Part C*, in press, 2004.

[11] A. A. Trani. *Runway Occupancy Time Estimation and SIMMOD.* Virginia Tech, 2000. http://www.nasug.com/nasug_rot0900.pdf.

Appendix A: Abbreviations

A/C	-	aircraft
ATC	-	air traffic control
CPN	-	coloured Petri net
RET	-	rapid exit taxiway
ROT	-	runway occupancy time
ROTA	-	ROT for arrivals
ROTD	-	ROT for departures
RWY	-	runway
THR	-	threshold
TWY	-	taxiway

Appendix B: Runway occupancy times

15C 27005KT	**1. TWY**	**%**	**2. TWY**	**%**	**3. TWY**	**%**
A1	42.94	100	50.33	0	98.26	0
A2	42.83	100	50.31	0	92.59	0
A3	41.61	100	49.06	0	90.98	0
B1	37.66	100	46.01	0	83.3	0
B2	0	0	47.84	100	90.77	0
B3	40.75	59.2	47.99	40.8	91.75	0
C1	0	0	45.52	99.6	84.87	0.4
C2	0	0	44.93	85.2	83.27	14.8
C3	0	0	43.74	65.6	78.83	34.4

15C 27010KT	**1. TWY**	**%**	**2. TWY**	**%**	**3. TWY**	**%**
A1	42.82	100	49.85	0	95.31	0
A2	42.97	100	49.88	0	94.99	0
A3	42.07	100	49.25	0	92.97	0
B1	38.5	100	46.64	0	84.39	0
B2	0	0	48.76	100	92.89	0
B3	41.69	69.2	48.63	30.8	93.78	0
C1	0	0	46.98	99.2	84.22	0.8
C2	0	0	45.65	84.8	85.11	15.2
C3	0	0	44.2	76.8	76.62	23.2

15C 27015KT	**1. TWY**	**%**	**2. TWY**	**%**	**3. TWY**	%
A1	42	100	48.14	0	98.56	0
A2	42.03	100	48.3	0	98.21	0
A3	42.19	100	48.84	0	95.62	0
B1	39.16	100	47.22	0	86.2	0
B2	0	0	49.35	0	95.72	0
B3	42.34	90.4	49.23	9.6	96.11	0
C1	0	0	47.53	100	90.57	0
C2	0	0	46.6	90.4	86.93	9.6
C3	0	0	44.73	91.6	80.97	8.4

15C 27020KT	**1. TWY**	**%**	**2. TWY**	**%**	**3. TWY**	**%**
A1	39.92	100	43.95	0	102.29	0
A2	39.78	100	44.9	0	101.57	0
A3	41.88	100	47.98	0	98.42	0
B1	39.78	100	47.74	0	87.83	0
B2	44.53	3.6	50.11	96.4	98.37	0
B3	43.18	98.4	49.39	1.6	98.94	0
C1	0	0	48.23	100	92.7	0
C2	0	0	47.57	92.4	88.64	7.6
C3	0	0	45.54	97.6	81.73	2.4

15C 09005KT	**1. TWY**	**%**	**2. TWY**	**%**	**3. TWY**	**%**
A1	42.81	100	50.33	0	93.02	0
A2	42.78	100	50.38	0	92.45	0
A3	41.6	100	49.1	0	90.87	0
B1	37.75	100	46.05	0	83.45	0
B2	0	0	47.98	100	90.78	0
B3	40.85	56.6	48.01	44.4	91.86	0
C1	0	0	45.72	99.6	87.83	0
C2	0	0	44.78	80.8	83.57	19.2
C3	0	0	43.74	56	79.38	44

Appendix C: CPN code segment of the simple runway

```
globref outfile = TextIO.stdOut; val start_time = ref 0;

color ENABLE = with enable;
var e : ENABLE;

color INT = int;
var time_head : INT;

color STATUS = with free | nofree;
var status_B,status_C : STATUS;

color POS = with final | dec | twyA | twyB | twyC | twyD | acc | trnthrD;
var pos_head, pos_out : POS;

color PLANE_TYPES = with type_1 | type_2 | type_3;
var type_head : PLANE_TYPES;

color PLANE = record p_type : PLANE_TYPES * pos : POS * t : INT;
var head, A_head, D_head : PLANE;

color QUEUE = list PLANE;
var tail, Arrive, Depart, Arrive_out, Depart_out, A_tail, D_tail : QUEUE;

color PLANE_T = product PLANE_TYPES * POS * INT timed;

color QUEUE_MS = QUEUE declare input_ms;
var new : QUEUE ms;


(* determine the take-off time depending on the plane type *)
fun Time_take_off(type_head) =
case type_head of
    type_1 => 55 |
    type_2 => 70 |
    type_3 => 85;

(* determine the runway for landing using probabilities  *)
fun decision(type_head) =
case type_head of
    type_1 => 1 |
    type_2 => if rint(0,100)<=48 then 1 else 2 |
    type_3 => if rint(0,100)<=69 then 2 else 3 ;

(* convert the position into string *)
fun POStoString(pos) =
case pos of
    twyB => "twyB\t" |
    twyC => "twyC\t" |
```

```
    twyD => "twyD\t" |
    trnthrD => "trnthrD" ;

(* convert the plane type into string *)
fun TypetoString(p_type) =
case p_type of
    type_1 => "type_1" |
    type_2 => "type_2" |
    type_3 => "type_3" ;

(* convert the model time intointeger *)
fun TimeToInt() = IntInf.toInt (time());

(* convert the landing runway into chart coordinate y *)
fun y(pos) =
case pos of
    twyB => 1 |
    twyC => 2 |
    twyD => 3 |
    _    => 4;

(* put a point (x,y) into the chart *)
fun point(x,y) =
LC_upd_chart_int{lc = "Scheduling", values = [(1, x, y, true, true)]};

(* put a plane into the corresponding queue *)
fun insert (new,[]) = [new]
  | insert (new, l as head::tail) =
     let
      val {p_type=_,pos=pos_new,t=time_new} = new;
      val {p_type=type_head,pos=pos_head,t=time_head} = head;
     in
      if pos_new = pos_head
      then if time_head<>0
           then if time_new <= time_head
                then new::l
                else head::insert(new,tail)
           else [new]
      else l
     end;;

(* initialize the output file *)
fun init_outfile() =
let
in
 outfile := TextIO.openOut("Simulation_result.txt");
 TextIO.output(!outfile,"% The structure of output file: \n
 %{ plane type, runway leaving position, scheduling time, real starting time,
 runway leaving time, => delay = difference from the scheduling }\n ");
 TextIO.closeOut(!outfile)
```

```
end;

val bad={p_type=type_1,pos=acc,t=0};

fun empty_queue([]) = [bad]
  | empty_queue(li) = li;

(* Clearance rules *)
fun rule (Arrive as A_head::A_tail, Depart  as D_head::D_tail) =
     let
      val {p_type=A_type,pos=A_pos,t=A_time} = A_head;
      val {p_type=D_type,pos=D_pos,t=D_time} = D_head;
     in
      if Depart=[bad]
      then if Arrive=[bad]
           then (bad,Arrive,Depart)
           else (hd Arrive,empty_queue(tl Arrive),[bad])
      else if Arrive=[bad]
           then (hd Depart,[bad],empty_queue(tl Depart))
           else if A_time <= D_time orelse
                   A_time <= Int.max(TimeToInt(),D_time)+Time_take_off(D_type)
                then (A_head,empty_queue(A_tail),Depart)
                else (D_head,Arrive,empty_queue(D_tail))
     end;;

(* calculate the waiting time before the next clearance*)
fun Wait(time_head) =
if time_head > TimeToInt() orelse TimeToInt() = 0
then time_head-TimeToInt()
else 0;

(* disturbance on the scheduling time *)
fun dist(head) =
let
 val {p_type=type_head,pos=pos_head,t=time_head} = head;
 val value = 0.1*real(time_head);
 val newtime = time_head + rint(0, 2*round(value)) - round(value);
in
 if pos_head = twyA
 then head
 else {p_type=type_head,pos=pos_head,t=newtime}
end;

(* calculate the landing runway *)
fun runway(type_head,status_B,status_C) =
let
 val goal = decision(type_head);
in
 if goal=1 andalso status_B=free
 then twyB
```

```
 else if goal=1 andalso status_C=free
      then twyC
      else if goal=2 andalso status_C=free
           then twyC
           else twyD
end;

(* calculate the position of the plane *)
fun next_pos(type_head,pos_head,status_B,status_C) =
case pos_head of
    twyA  => acc     |
    acc   => trnthrD |
    final => dec     |
    dec   => runway(type_head,status_B,status_C);

(* normal distribution with mean, variance, min and max differences *)
fun distrib(mean,variance,min_diff,max_diff) =
let
 val value = round(normal( real(mean),real(variance) ));
in
 if value < mean + min_diff orelse value > mean + max_diff
 then  distrib(mean,variance,min_diff,max_diff)
 else value
end;

(* calculate the runway using time with a given distribution *)
fun Delay(type_head, aim) =
case (type_head,aim) of
    (type_1,twyB) => distrib(42,3,10,21) |
    (type_1,twyC) => distrib(50,3,10,25) |
    (type_1,twyD) => distrib(90,5,10,45) |
    (type_2,twyB) => distrib(38,2,10,19) |
    (type_2,twyC) => distrib(47,3,10,23) |
    (type_2,twyD) => distrib(87,5,10,43) |
    (type_3,twyB) => 0 |
    (type_3,twyC) => distrib(44,2,10,22) |
    (type_3,twyD) => distrib(81,4,10,41) |
    (_,trnthrD)   => distrib(Time_take_off(type_head),10,10,10) |
     _            => 0;
```