Transformation of CIMOSA-Models into Petrinet-Models with PACE

Bernd Eichenauer, IBE Simulation Engineering (Eichenauer@ibepace.com)
Martin Zelm, CIMOSA Association e.V. (Zelm@cimosa.de)

Abstract : Enterprise modelling provides the means to capture, structure and manage the enterprise system and to identify the needed and produced information. CIMOSA (CIM Open System Architecture) provides a process oriented modelling concept to capture both process functionality and process behaviour. Petrinet-models are very powerful to model process scenarios with the objective to achieve computer executable models, which enable quantitative, exact results of the simulated processes.

CIMOSA models created with user oriented constructs can serve various purposes in simulation and operation control, providing primarily qualitative results. Petrinet models, yield mathematically exact quantitative simulation results. The models are generally less transparent due to the more formal representation needed in mathematical simulation.

This paper describes general rules to transform CIMOSA models into Petrinet models, created with the modelling tool PACE. The method is deduced on a basis of modelling constructs and illustrated with an example of a simple process modelling scenario. More work is required to extend and validate more complex transformation rules .

Keywords : CIMOSA, Petrinets, dynamic process modelling, simulation, PACE tool

Introduction

CIMOSA (CIM Open System Architecture) provides a process oriented modelling concept to capture both process functionality and process behaviour. [CIMOSA-93,-96,-99, Vernadat-96] . CIMOSA supplies requirements models of discrete processes -especially for business models- in a user oriented mode, in order to make the process scenario and the interaction of its components transparent and easy to understand. Petrinet-models created with the modelling tool PACE [PACE-99] can model the same process scenario with the objective to achieve computer executable models, which enable quantitative, exact simulation results. Both modelling methodologies address the same reality with different objectives in mind. CIMOSA models for instance attempt to support decisions by demonstrating the qualitative results. PACE-Models yield exact quantitative simulation results, the models are generally less transparent due to the more formal representation needed in mathematical simulation with Petrinets.

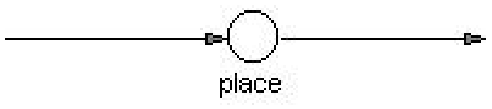
In the following, general rules for the transformation and implementation of CIMOSA models into Petrinet-models with the PACE tool are proposed and verified in a simple application example. These rules have to be further validated in more complex examples and must be extended with more complex rules, which for instance include re-usable sets of Events and sets of Enterprise Activities.

General transformation rules

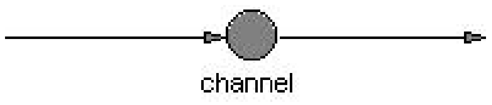
We define the following conversion methods:

1. Enterprise Activities are depicted in PACE-modules. For certain standardised Enterprise Activities, a PACE-Module library provides standard-modules.
2. Events are depicted in PACE-modules. For certain standardised Events (e.g. time-Events) another PACE-Module library provides standard-modules.

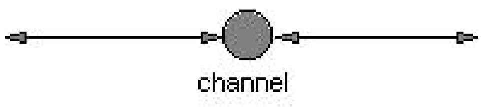
3. Simple connectors between CIMOSA building blocks (Events, Enterprise Activities) are converted into the PACE-construct ‘*place*’:



4. Multiple connections between CIMOSA building blocks pointing in one direction are represented by the PACE construct ‘*channel*’:



.
If connections in both directions occur, then the construct:



has to be used. In special cases the construct ‘*channel*‘ may degenerate into the construct ’*place*‘ .

Example:
We apply the transformation rules in the following the CIMOSA model of example of a car-wash station, shown in Fig 1a

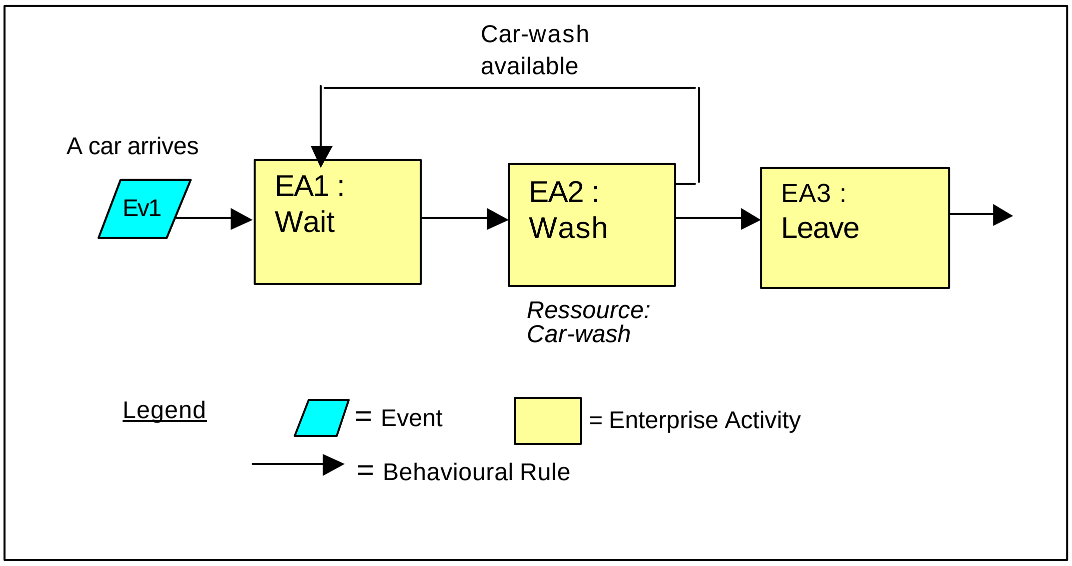


Figure 1a: Car-wash station modelled with CIMOSA

and receive the following main net of the related PACE-model, shown in Fig. 1b:

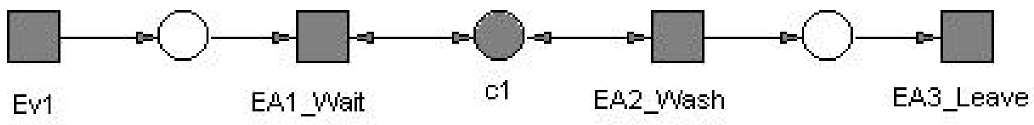


Figure 1b: Same car wash station modelled in a Petri net with PACE

Modeling of Events

PACE offers numerous methods to model Events. Which method is used in a particular case, depends on the modelling goal of the model designer as well as on the type of event. A final specification to transform CIMOSA models into PACE models has to provide a classification of the event types in CIMOSA, has to describe their implementation and provide standard-modules in a PACE-library.

We look at the implementation of an Event in the above mentioned example of a car-wash station :

Event

Event	
Type	Material (Arrival)
Identifier	EV1
Name	**Car arrives**
Description	A car arrives at the car-wash
Triggers	Enterprise Activity ‘Wait‘
Resource	Driver
Object	Car
Frequency	Exponential distribution, Mean = 12

It concerns a sequence of exponentially distributed points in time, at which cars arrive at the car-wash. This is expressed in the CIMOSA declaration:

WHEN (START WITH Event1)
Car arrives DO (EA1) Wait

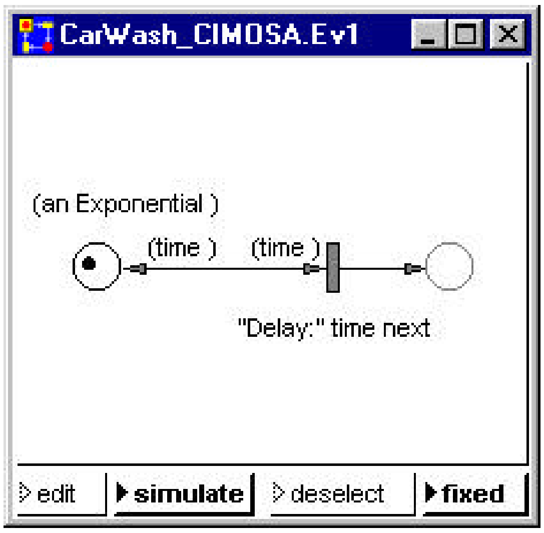
From the above table we need only the object name and the frequency quotation. The other information contained in the table, as far as it is required in this example, has already be considered in the main net. The statement „Triggers“ has already been taken into account at in the connection of Ev1 with EA1_‘Wait‘.

The name of the „Object“ can be used later in the process to replace the standard icon used for tokens with icons of vehicles and thereby visualise the PACE model with respect to the modelled reality.[1]

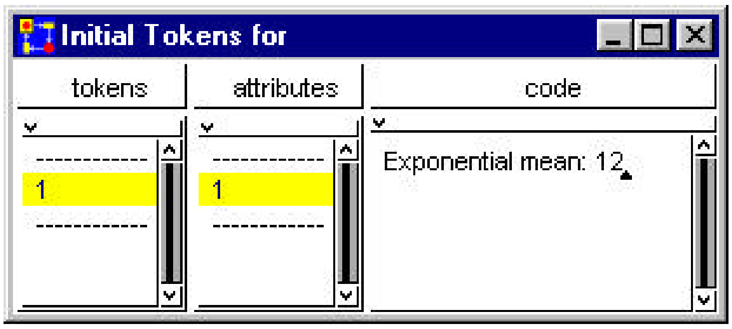
In the following, two modules needed for the creation of events which occur at an exponentially distributed point in time are given :

[1] Measures to design ‚beautiful‘ PACE-Models are not considered here. They are described in detail in the PACE documentation .

Fixed Mean
An exponential distribution is completely defined by stating its mean value. If the mean value is predetermined, the following module may be selected:



A so-called 'initial token' (representing a car) is put into the left place, which contains a method to calculate the values of an exponential distribution. When running the model, this calculation method is carried to the transition on the right hand place, where the next value of the distribution „time next" will be determined. This value provides the waiting time until the next car arrives. The calculation rule is pre-determined in the menu of the initial tokens :



tokens	attributes	code
1	1	Exponential mean: 12

After this time has elapsed, the token with the calculation rule runs from the left place to the transition where it is validated and then brought back to the left place. At the same time a token (a car) is brought to the right side place. This place is shown with lower intensity, because it is not defined in this module but in the main net, located at a higher level.

Variable Mean
The module described above has the disadvantage that the rule inserted manually to calculate the next waiting time is fixed. This rule can only be changed if a new distribution respective a new mean value is inserted by hand.

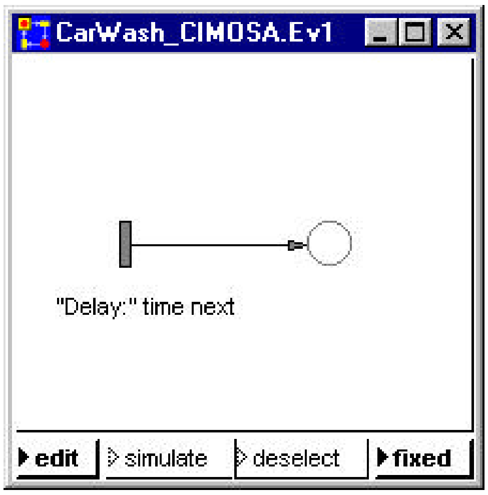
Occasionally by external events or with time the exponential distribution changes and has also to be changed in the model during simulation.

In this case it is appropriate to store the calculation rule during the initialisation phase of a simulation in a net or global variable. When the distribution has to be changed the new distribution is stored in that variable.

If we insert for example the following assignment into the PACE initialisation code:

Distribut := Exponential mean: 12.

where „Distribut“ is a global variable, the module „Ev1“ would look as follows:

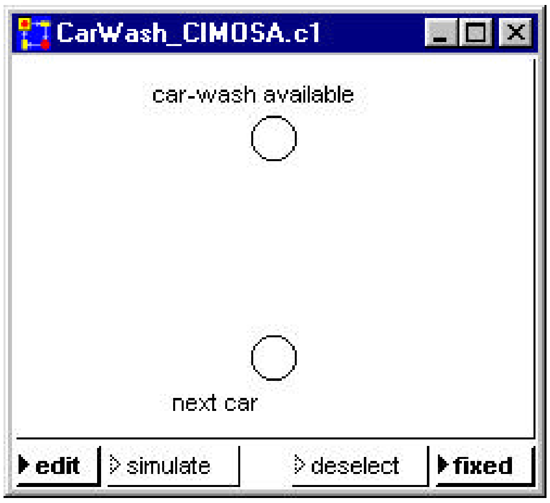


The two described modules could be inserted as standard modules in a PACE module library to be used and adapted appropriately if time delays have to be modelled.

Modelling channels

Channels as interfaces between modules can only contain passive net elements, i.e. places and channels. A place has to be used for every single connection between two modules. If several (sub)connections have to be combined the channel can contain further (sub)channels.

In our example we open the module „c1“ in the main net module and according to the CIMOSA description (see page 2) we insert two places. One of the places is to trigger the washing process the other is used for the feedback “car-wash available”.

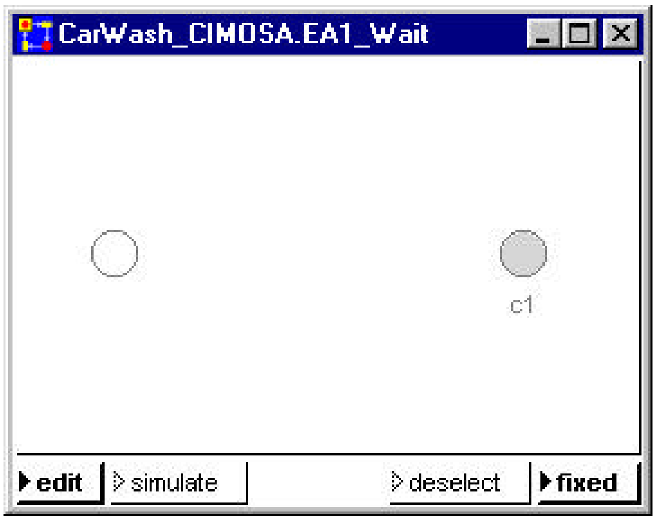


Modelling Enterprise Activities

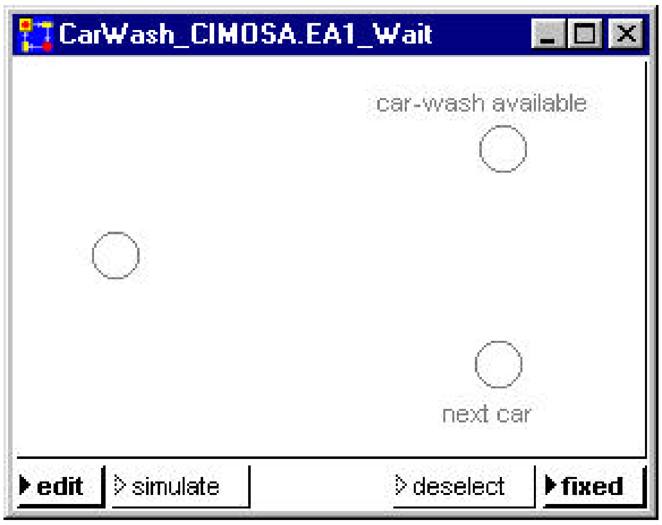
Our example contains three CIMOSA activity types, namely “store”, “transform” and “move”.[2] In the following we model these three Enterprise Activities as defined in the main net module.

Wait (activity type: store)

We open the module EA1 to see the following picture:



On the left side we see the interface to the event module Ev1, on the right side the interface to the Enterprise Activity EA2. To establish the connection in detail we first insert the refinement of the channel and receive the following module:

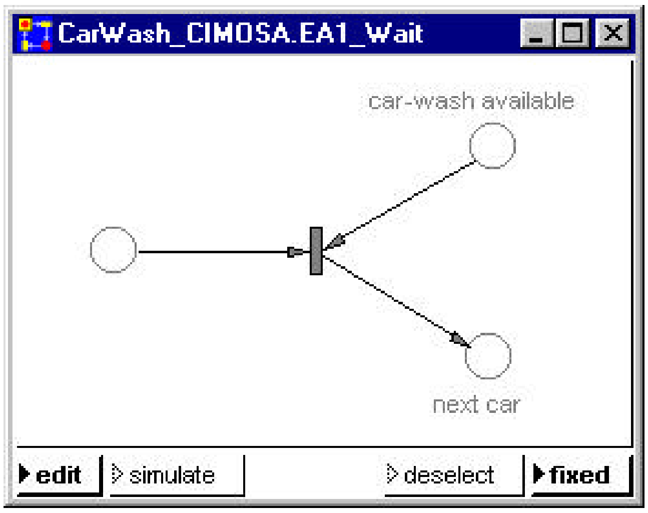


[2] The problem oriented differentiation of activity types is not meaningful for the lower description level of a PACE. There are no activity types in PACE.

From the main net module and from the CIMOSA description of the module ‘Wait’:

Enterprise Activity	
Type	Store : Material or information
Identifier	*EA1*
Name	**Wait**
Description	A car waits in the queue
Function Input	Car
Control Input	Car-wash status
Resource Input	Driver
Function Output	Car
Control Output	
Resource Output	Driver Waiting time
Duration Min / Max / Average	Avg. waiting time
Ending Status -	Car in place

we conclude, that a car has to wait until the waiting queue is empty. We fulfil this condition by connecting the three places via a transition:

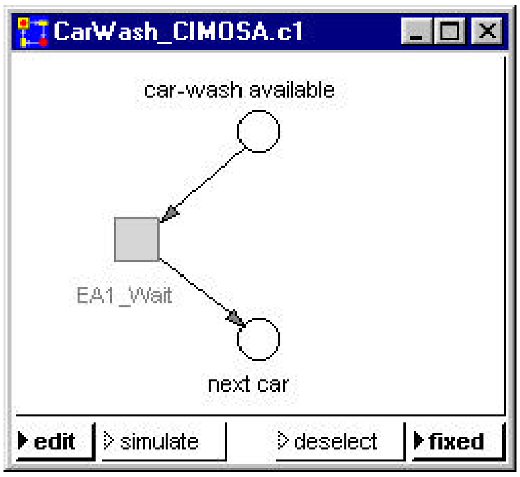


The transition can only fire when the left place and the place „car-wash available“ contains a token at the same time. In this case a token is inserted in the place “next car” which represents the car to be washed. In order for the car-wash to be empty when the simulation starts we have to insert an initial token in the place “car-wash available”. Hence, the CIMOSA statement:

WHEN Ending Status Car-wash available
AND Next car in place DO (EA2) Wash

is fulfilled and we have developed by this a template of a standard module which delivers an object if two conditions hold.

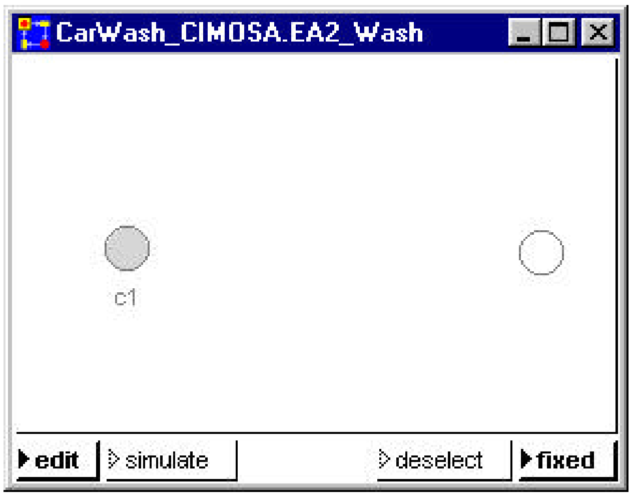
It is worth mentioning that the picture of the channel c1 has been automatically completed after modelling of the module „EA1_Wait“:



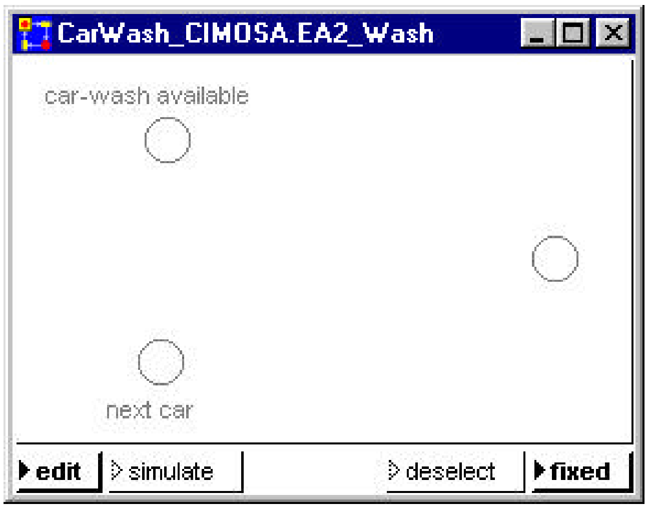
The linkage of the channel elements with the Enterprise Activity in the main net module is now visible.

Wash (activity type: transform)

As with EA1 we at first open the module EA2 and see the following picture:



It corresponds with the picture which we have found when we started with the modelling of the module “EA1_Wait”. We again refine the channel and receive:



From the description

Enterprise Activity	
Type	Transform : Material or information
Identifier	*EA3*
Name	**Wash**
Description	A car enters the car-wash and is washed, while the car-wash is not available for other cars
Function Input	Car
Control Input	-
Resource Input	Car-wash Driver
Function Output	Car
Control Output	-
Resource Output	Car-wash operation time Driver operation time
Duration	Mean washing time = 10 minutes
Ending Status	Car-wash available

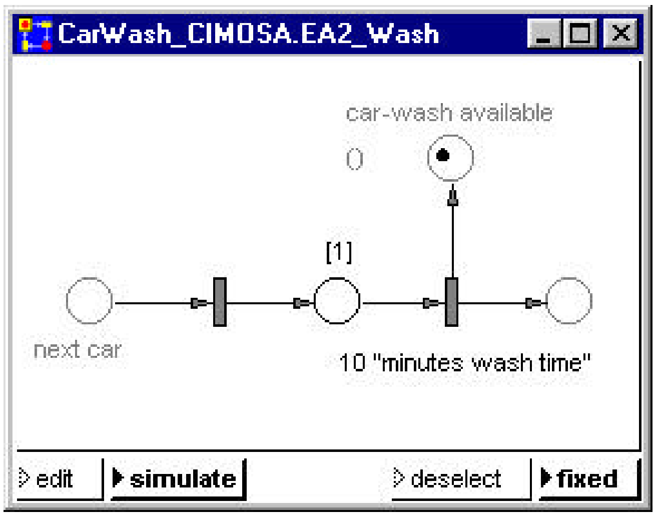
we learn two facts essential for the next steps:

- The module “car-wash” can contain only one token (one car) at a time.
- The mean washing time for a car is 10 minutes.

The further proceeding is defined in the CIMOSA statement:

WHEN Ending Status Car washing complete
AND Car-wash available DO (EA3) Leave

A subnet which fulfils this requirement is contained in the next window:

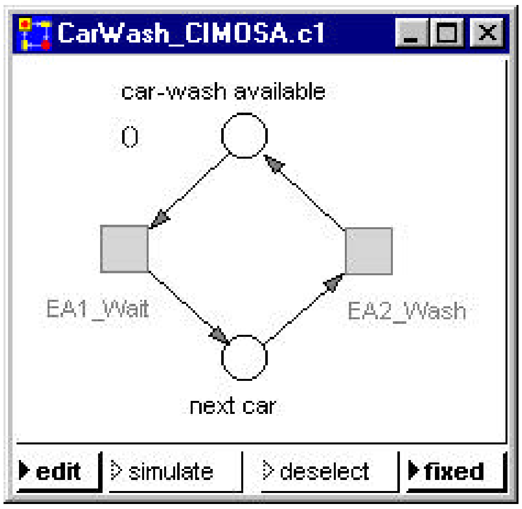


Apart from position changes of the places which act as interfaces to other subnets (painted with lower intensity) three further net elements have been inserted. The left transition transfers tokens (cars) to the place in the middle (car-wash) which is marked with “[1]”. This means that the place can only store one token. The right transition guaranties that the token leaves the middle place after 10 simulation time units (1 unit = 1 minute in our case).

After 10 time units this transition fires and transfers a token to each of the two output places. The token in the place “car-wash available” is a “control token” and is used as described earlier in the wait module. The token in the right place models the car which just left the car-wash.

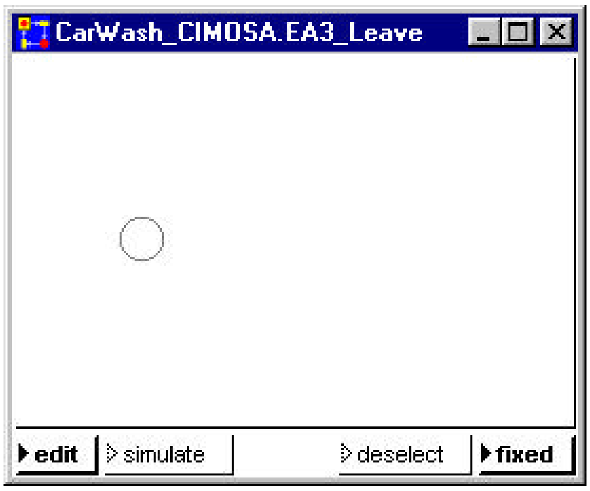
This module can serve as a prototype of a working module with one server.

It should be mentioned that with the modelling of EA2 , the interface to EA1 has been completed automatically:

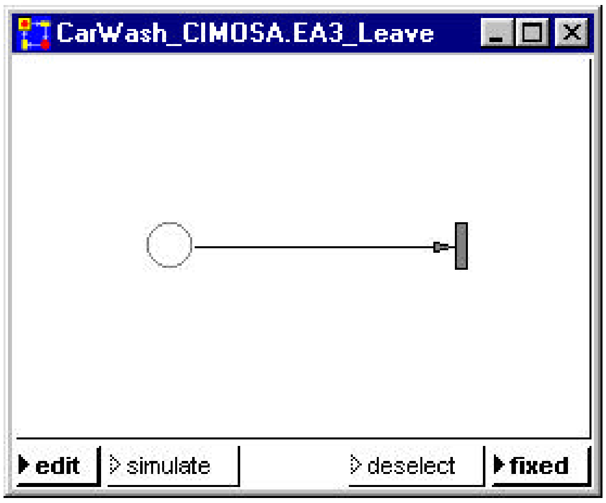


Leave (activity type: move)

The last of the three modules is also very simple. When we open module EA3, we see the following window:



It only contains the interface to module EA2. We insert a transition and connect it to the place:

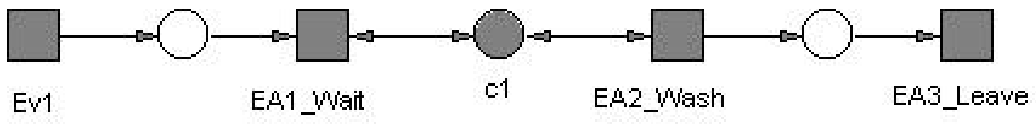


The transition fires if the place contains a token and removes the token (the car) from the model.

Results and discussion

In Fig. 2 we show all developed PACE components on two levels in a summary:

Level 1:



Level 2:

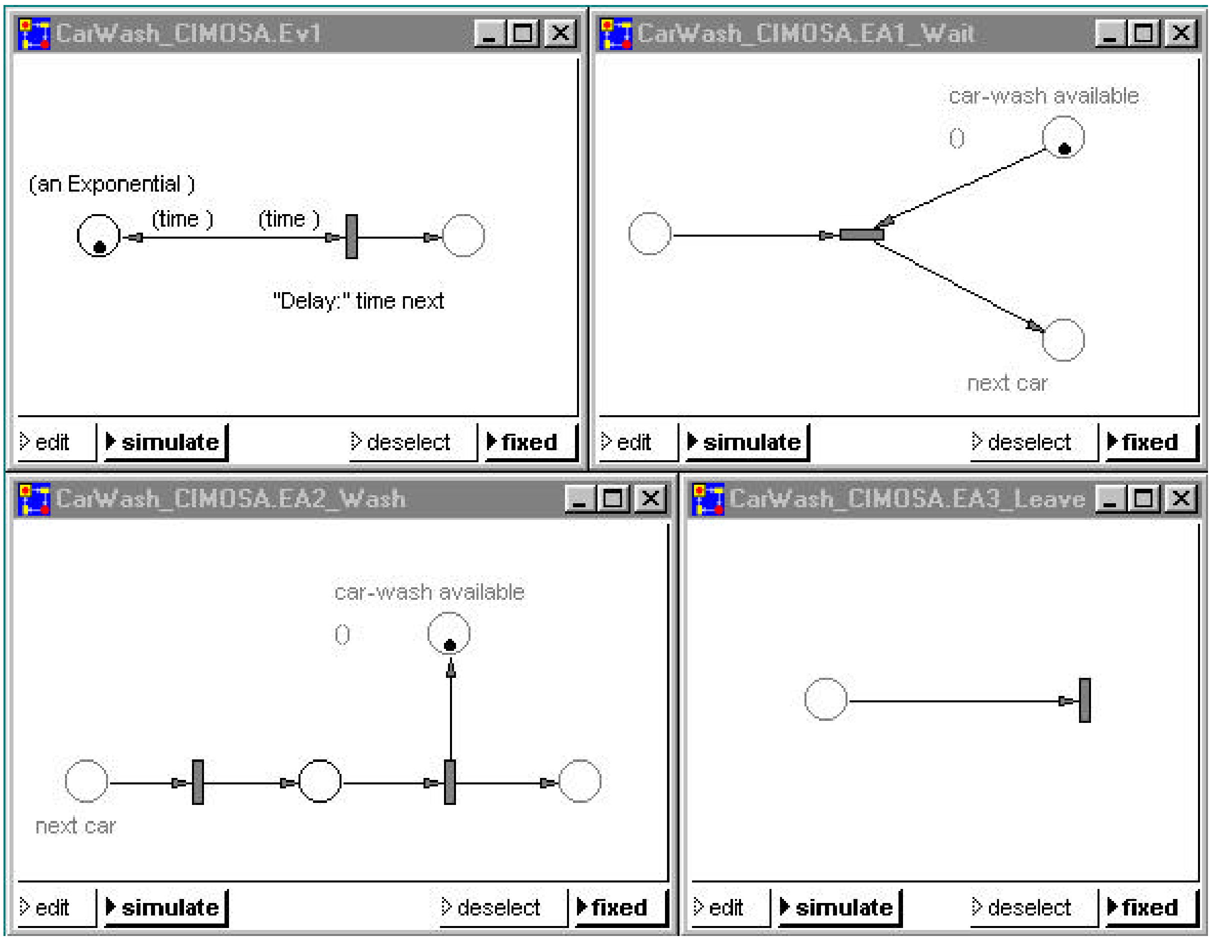


Figure 2: Example of the car wash station modelled in two Petri net levels with PACE

To see the benefit of a transformation of CIMOSA models to PACE models and to demonstrate the possible results with respect to the dynamics of a model, we investigate, what the waiting queue in front of the car-wash looks like.

If more than one car appears, the tokens which represent the cars are collected in the most left place in the level 1 picture above. This place is also painted as the right respectively left place in window “CarWash_CIMOSA.Ev1” respectively “CarWash_CIMOSA.EA1_Wait”.

PACE offers the possibility to connect a histogram or a line diagram to a place. If we connect these diagrams to the place mentioned before and then run the simulation we receive:

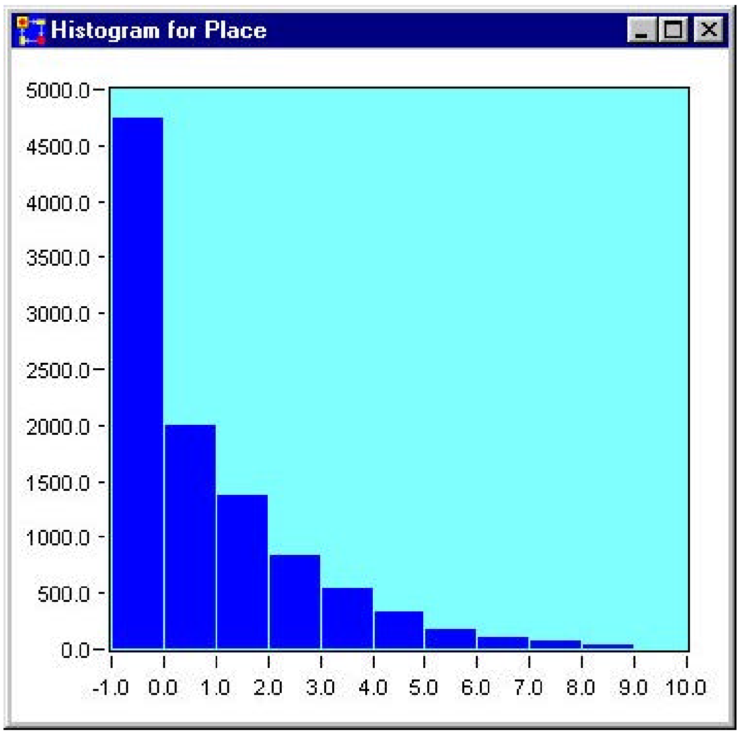


Figure 3: Frequency distribution of the number of waiting cars

The histogram of figure 3 shows that normally no more than 6 cars are waiting in front of the car-wash and that in approximately 40% of all cases no waiting times occur (the car can enter the car-wash immediately).

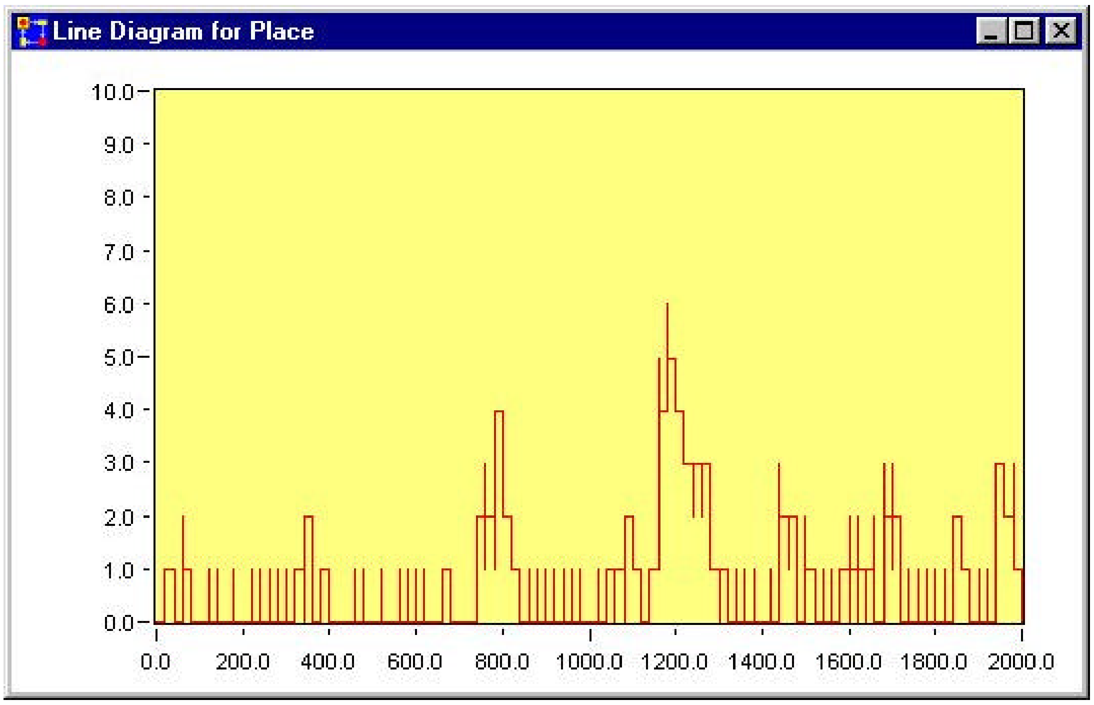


Figure 4: Occupancy of the car-wash.

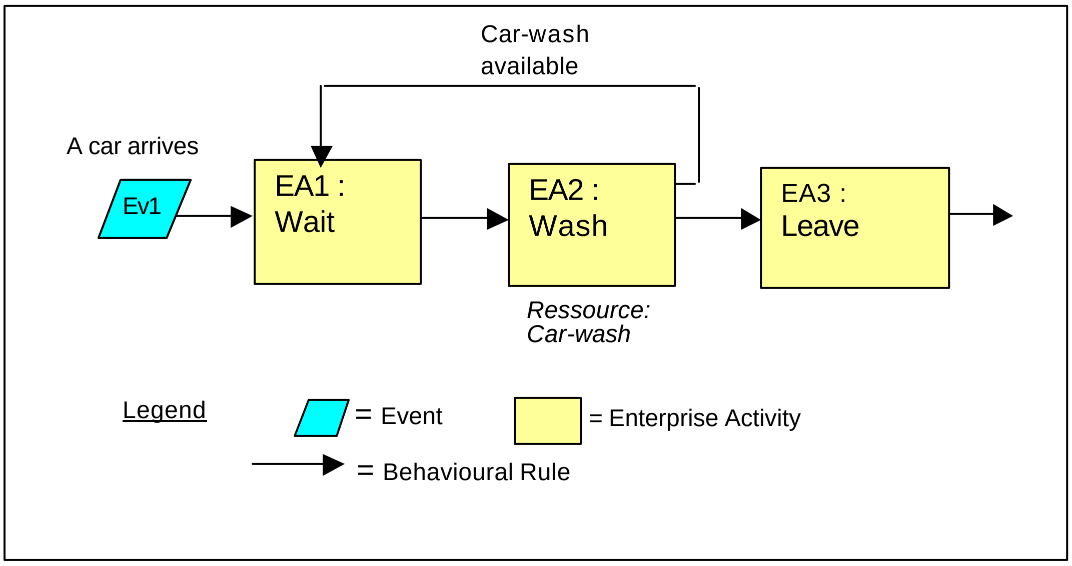
The line diagram in Fig 4 shows a random occupancy of the car-wash. The maximum occupancy is six cars in front of the station, typically only one or two cars are waiting.

This example demonstrates the potential of high level Petrinets to predict general process scenarios or business scenarios. The advantage of the two step approach is that the requirements can be modelled in a clear user oriented mode. The following simulation yields mathematically exact powerful results.

More work is needed on the described methodology to simulate any complex process requirements models made of event driven process chains. On the one hand side the general transformation rules must be extended to cope with more complex scenarios. Further, re-usable sets of Events, sets of Enterprise Activities and libraries of partial models have to be specified and the corresponding PACE constructs have to be provided in PACE module libraries. PACE provides the mechanisms to assemble such model parts in order to make this methodology to create and maintain enterprise models an attractive, economically justified task. By this, an engineering approach to transform CIMOSA models into PACE models can be established.

Annex: Modelling example car-wash

In this annex, the CIMOSA model of the car-wash is summarised with the templates of all modelling constructs used in the example.



CIMOSA modell of a car-wash station

Templates of Activities and Events

Aktiviy type 'Store'

Example: Car waiting

Enterprise Activity	
Type	Store : Material or information
Identifier	*EA1*
Name	**Wait**
Description	A car waits in the queue
Function Input	Car
Control Input	Car-wash status

Resource Input	Driver
Function Output	Car
Control Output	
Resource Output	Driver Waiting time
Duration Min / Max / Average	Avg. waiting time
Ending Status -	Car in place

Aktiviy type 'Transform'

Example : Wash car

Enterprise Activity	
Type	Transform : Material or information
Identifier	*EA3*
Name	**Wash**
Description	A car enters the car-wash and is washed, while the car-wash is not available for other cars
Function Input	Car
Control Input	-
Resource Input	Car-wash Driver
Function Output	Car washed
Control Output	-
Resource Output	Car-wash operation time Driver operation time
Duration	Mean washing time = 10 minutes
Ending Status	Car-wash available

Activity Type ‘Move‘

Example: Move car away

Enterprise Activity	
Type	Move : Material or information
Identifier	*EA3*
Name	**Leave**
Description	Car leaves after the wash operation is finished, the car-wash becomes available for other cars
Function Input	Car
Control Input	Car washed
Resource Input	Driver
Function Output	Car
Control Output	
Resource Output	Driver driving time
Duration Min / Max / Average	-

Ending Status	End of process

Event

Event	
Type	Material (arrival)
Identifier	*EV1*
Name	**Car arrives**
Description	A car arrives at the car-wash station
Triggers	Enterprise Activity ‘Wait‘
Resource	Driver
Object	Car
Frequency	Exponential distribution, Mean =12

Behavioural Rule Set
The behavioural rule set to dynamically link Activities, follows the Event/ES-Condition-Action principle

```
WHEN (START WITH Event1)
                         A car arrives        DO (EA1)     Wait

WHEN Ending Status  Car-wash available
            AND        Car in place           DO (EA2)     Wash

WHEN Ending Status  Car washing complete
            AND        Car-wash available     DO (EA3)     Leave
```

Literature

[CIMOSA-93] ‘*CIMOSA - Open System Architecture for CIM*; ESPRIT Consortium AMICE, Springer-Verlag 1993, (ISBN3-540-56256-7), (ISBN 0-387-56256-7).

[CIMOSA-96] ‘*CIMOSA - Open System Architecture for CIM’, Technical Baseline*; Version 3.2, CIMOSA Association e.V. private publication (1996), CIMOSA Association, http://www.cimosa.de

[CIMOSA-99] ‘*CIMOSA- Evaluation and Applications in Enterprise Engineering and Integration’*, Eds.: K. Kosanke, F. Vernadat, M. Zelm, Computers in Industry, Vol. 40 Numbers 2-3 (1999)

[Vernadat-96] F.B. Vernadat, *Enterprise Modelling and Integration, Principles and Applications;* Chapman and Hall, 1996, ISBN 0-412-60550-3

[PACE-99] PACE 3.1, User Manual, IBE Simulation Engineering, D-85823 Glonn, Mai 1999, http://www.ibepace.com