ARGESIM Report no. 6

COMETT - Course "Object-Oriented Discrete Simulation"

Seminar Modellbildung und Simulation

N. Kraus, F. Breitenecker

ISBN 3-901608-06-0

ISBN 3-901608-06-0

ARGESIM Report No. 0

ARGE Simulation News (ARGESIM)
c/o Technical University of Vienna
Wiedner Hauptstr. 8-10
A-1040 Vienna, Austria
Tel: +43-1-58801 5386, 5374, 5484
Fax: +43-1-5874211
Email: argesim@simserv.tuwien.ac.at
WWW: <URL:http://eurosim.tuwien.ac.at/>

FOREWORD

At present there is a big need for automatisation of production processes, of manufacturing processes, of control systems, etc. For automatisation at each level different hardware and software tools are offered. It turns out that simulation is the most general tool offering not only certain software but also methodology and know-how for automatisation

Simulation is the process of designing a computerised model of a system and experimenting with this model in order to understand better the behaviour of the system, or to locate specific problems, or to evaluate different strategies for the operation of the system, or to plan a new process.

This course offers an introduction for engineers, scientists, and interested people from other fields to modern methods of computer simulation of and for automatisation.

Due to the nature of the investigated processes this course deals on the one side with discrete simulation and optimisation in discrete simulation - in order to handle the planning and scheduling problems in e.g. production systems.

On the other hand this course gives from the technical point of view insight into simulation tools for controlling and optimising machines like robots, in order to increase the efficiency of stations, robots, AGVs, etc. Especially the modern technique of fuzzy control and simulation will be considered.

The course consists of three independent units:

- Introduction into Discrete Simulation
- Simulation and Automatisation with Object-Oriented Tools (this report)
- Fuzzy Control for Automatisation

The first unit "Introduction into Discrete Simulation" is a basic unit introducing into the concepts of discrete simulation, and into applications for automatisation. Also mathematical background (statistic) will be sketched and modern methods for optimisation (genetic algorithms).
This unit adresses newcomers in simulation and automatisation. In this unit the classical simulation tool GPSS/H and the PROOF animation system will be used.

The second unit "Simulation and Automatisation with Object-Oriented Tools" adresses people from application areas giving an overview about modelling,

simulating and opimising discrete processes (manufacturing processes, etc.) by means of modern powerful object-oriented software tools.
Application and case studies presented in this unit work with the modern simulator SIMPLE++, which allows to formulate optimisation and automatisation strategies by means of methods. As this unit is based on applications, only little or no previous knowledge is necessary.

The third unit "Fuzzy Control for Automatisation" deals with the technical aspects of automatisation and adresses (control) engineers or people from related areas. After a short introduction into basic principles of (optimal) control for automatisation (of machines, robots, etc.) principles and applications of the new tools *Fuzzy Control* are considered.
Specific software tools and general ones (MATLAB) will be discussed. Although this unit deals with technical aspects and seems to be independent from the world of discrete processes, it completes the first and the second unit from the view of automatisation.

After the course a participant should be able to make decisions about the use of modelling and simulation methods in the area of automatisation, to work with simulation tools for automatisation purposes, and to decide on an efficient use of simulation for a specific automatisation

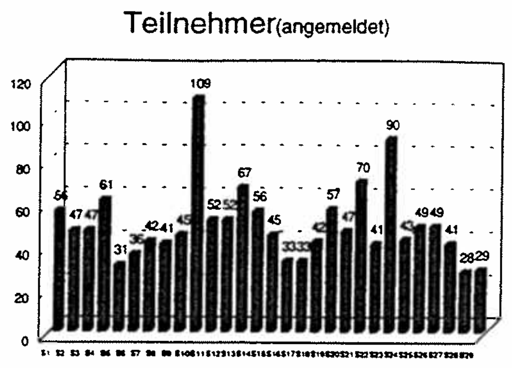
All three units of the COMETT Course are available as ARGESIM Reports.

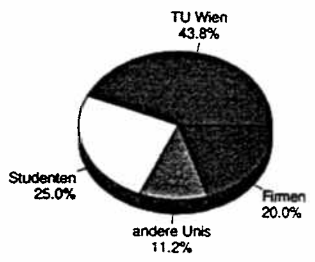
We wish to thank AESOP for making available the material for the second unit of this course.

Seminare über Modellbildung und Simulation

Seit dem Frühjahr 1991 veranstaltet das EDV-Zentrum gemeinsam mit der Abteilung Regelungsmathematik und Simulationstechnik des Instituts für Technische Mathematik und der ARGE Simulation News (ARGESIM) Vortragsveranstaltungen zum Thema Modellbildung und Simulation (Simulationsseminare). Organisatoren sind I. Husinsky und F. Breitenecker. Das Ziel ist, verschiedene Simulationswerkzeuge vorzustellen, über ihre Einsatzmöglichkeiten zu informieren und Erfahrungen auszutauschen. Ferner werden bekannte Simulationsfachleute eingeladen, Grundsatzvorträge zum Thema Simulation zu halten. Im allgemeinen werden die Seminare teilweise von Firmen gesponsert oder über Simulationsprojekte mitfinanziert. Sie dauern einen halben oder einen Tag, es gibt schriftliche Unterlagen zu den Vorträgen und Softwareprodukten. Ein Buffet fördert die Kommunikation zwischen den Seminarteilnehmern in den Pausen.
Bis jetzt haben folgende Seminare stattgefunden:

S1	23. 4. 1991	ACSL
S2	4. 6. 1991	CTRL_C, XANALOG
S3	22. 10. 1991	SIMUL_R
S4	5. 5. 1992	ACSL
S5	6. 5. 1992	MicroSaint
S6	17. 6. 1992	Objektorientierte Modellbeschreibung und qualitative Simulation (F. Cellier, University of Arizona)
S7	1. 7. 1992	Diskrete Simulation und Analyse (D. Kelton, University of Minnesota)
S8	23. 10. 1992	GPSS/H (T. Schriber, University of Michigan)
S9	10. 12. 1992	SIMPLE
S10	2. 2. 1993	MATLAB und SIMULINK
S11	25. 3. 1993	Modellbildung mit Bondgraphen (D. Karnopp, University of California)
S12	24. 5. 1993	MicroSaint
S13	22. 6. 1993	ACSL
S14	21.10.1993	XANALOG, SIMNON
S15	22.10.1993	GPSS/H (T. Schriber, University of Michigan)
S16	11.11.1993	IDAS
S17	7.12.1993	SIMPLE++
S18	14.12.1993	Petrinetze, D_SIM (R. Hohmann, Magdeburg)
S19	4.2.1994	Modellbildung und Simulation in der Lehre
S20	14.3.1994	GPSS/H und Proof (T. Schriber, University of Michigan)
S21	13.4.1994	ACSL
S22	10.5.1994	SIMUL_R, Partielle Differentialgleichungen
S23	22. 11. 1994	MATLAB/SIMULINK
S24	14.12.1994	SIMPLE++
S25	31.1.1995	Parallele Simulation, mosis
S26	28.3.1995	ACSL
S27	29.3.1995	MicroSaint
S28	13.6.1995	COMETT II, Part one, Discrete Simulation
S29	28.6.1995	COMETT II, Part two, Simulation and Automatisation





Die Teilnehmer, etwa 30 bis 110 je Seminar, kommen zum Großteil von der TU, aber auch von anderen Universitäten und aus der Industrie. Bei den bisherigen Seminaren waren etwa 20% der Teilnehmer aus der Industrie.

Das Programm eines Seminars setzt sich im allgemeinen aus einem oder zwei Grundlagenvorträgen, mehreren Anwendervorträgen, Produktpräsentationen, Vorführungen am Rechner und Diskussionen zusammen.

Die Teilnehmer werden um eine Anmeldung gebeten, daher können die Unterlagen (Seminarberichte), die zu Beginn des Seminars verteilt werden, schon eine Teilnehmerliste enthalten. Ab Herbst 1995 erscheinen die Unterlagen als ARGESIM Report. Alle, die bereits an einem Seminar teilgenommen haben, werden automatisch zu den weiteren Seminaren eingeladen.

Information:
I. Husinsky, EDV-Zentrum, Technische Universität Wien, Wiedner Hauptstr. 8-10, A-1040 Wien,
Tel: (0222) 58801 5484, Fax: (0222) 587 42 11,
E-Mail: husinsky@edvz.tuwien.ac.at

Prof.Dr. F. Breitenecker, Abt. Regelungsmathematik u. Simulationstechnik, Inst. 114, Technische Universität Wien, Wiedner Hauptstr. 8-10, A-1040 Wien,
Tel: (0222) 58801 5374, Fax: (0222) 587 42 11,
E-Mail: fbreiten@email.tuwien.ac.at

About ARGESIM

ARGE Simulation News (ARGESIM) is a non-profit working group providing the infra structure for the administration of **EUROSIM** activities and other activities in the area of modelling and simulation.

ARGESIM organizes and provides the infra structure for

- the production of the journal EUROSIM Simulation News Europe
- the comparison of simulation software (EUROSIM Comparisons)
- the organisation of seminars and courses on modelling and simulation
- COMETT Courses on Simulation
- "Seminare über Modellbildung und Simulation"
- development of simulation software, for instance: mosis - continuous parallel simulation, D_SIM - discrete simulation with Petri Nets, GOMA - optimization in ACSL
- running a WWW - server on EUROSIM activities and on activities of member societies of EUROSIM
- running a FTP-Server with software demos, for instance
  - * demos of continuous simulation software
  - * demos of discrete simulation software
  - * demos of engineering software tools
  - * full versions of tools developed within ARGESIM

At present ARGESIM consists mainly of staff members of the Dept. Simulation Technique and of the Computing Services of the Technical University Vienna.

In 1995 ARGESIM became also a publisher and started the series **ARGESIM Reports**. These reports will publish short monographs on new developments in modelling and simulation, course material for COMETT courses and other simulation courses, Proceedings for simulation conferences, summaries of the EUROSIM comparisons, etc.

Up to now the following reports have been published:

No.	Title	Authors / Editors	ISBN
# 1	Congress EUROSIM'95 - Late Paper Volume	F. Breitenecker, I. Husinsky	3-901608-01-X
# 2	Congress EUROSIM'95 - Session Software Products and Tools	F. Breitenecker, I. Husinsky	3-901608-02-8
# 3	EUROSIM'95 - Poster Book	F. Breitenecker, I. Husinsky	3-901608-03-6
# 4	Seminar Modellbildung und Simulation - Simulation in der Didaktik	F. Breitenecker, I. Husinsky, M. Salzmann	3-901608-04-4
# 5	Seminar Modellbildung und Simulation - COMETT - Course "Fuzzy Systems and Control"	D. Murray-Smith, D.P.F. Möller, F. Breitenecker	3-901608-05-2
# 6	Seminar Modellbildung und Simulation -COMETT - Course "Object-Oriented Discrete Simulation"	N. Kraus, F. Breitenecker	3-901608-06-0
# 7	EUROSIM Comparison 1 - Solutions and Results	F. Breitenecker, I. Husinsky	3-901608-07-9
# 8	EUROSIM Comparison 2 - Solutions and Results	F. Breitenecker, I. Husinsky	3-901608-08-7

For information contact: ARGESIM, c/o Dept. Simulation Techniques,
attn. F. Breitenecker, Technical University Vienna
Wiedner Hauptstraße 8-10, A - 1040 Vienna
Tel. +43-1-58801-5374, -5386, -5484, Fax: +43-1-5874211
Email: argesim@simserv.tuwien.ac.at

TABLE OF CONTENTS

The New Class of Simulation Software

SIMPLE++

- The standard software for object-oriented, graphical and integrated modeling, simulation and animation.

AESOP Partner in Austria :

AESOP

AESOP GmbH, Königstraße 82
70173 Stuttgart, Germany
Tel: +49-711-16 359 / 0
Fax: +49-711-16 359 / 99

Unseld + Partner, Lerchenfelderstr. 44/9
A-1080 Vienna, Austria
Tel: +43-1-40 30 371
Fax: +43 -1-40 30 371 / 90

SIMPLE++ Overview

1. Abstract

In the past, simulation software lacked user friendliness and functionality and therefore, the market penetration is still very low despite of the outstanding profitability of using simulation. SIMPLE++ has been developed to overcome these deficits in an innovative manner and its success in the market place shows that SIMPLE++ has broken down former barriers.

AESOP developed SIMPLE++ together with the Fraunhofer Institute for Production and Automation (IPA) and is based on more than 20 years of experience in developing and applying simulation software. SIMPLE++ was introduced to the market place in February 1992. Since its introduction, was installed over 200 times in the German speaking market in the first two years and became one of the market leaders. Since April 1993 the full English version of SIMPLE++ has been shipped world-wide.

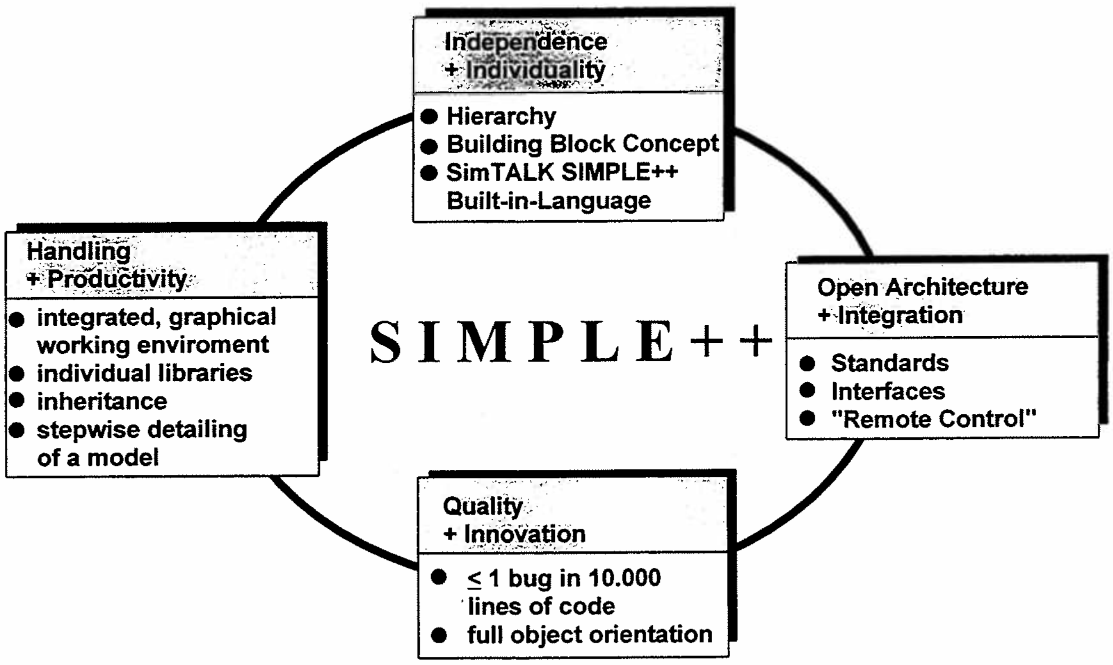
SIMPLE++ stands for **SIM**ulation in Production, Logistics and Engineering design and its implementation in C++. SIMPLE++ is the standard software for object oriented, graphical and integrated modelling, simulation and animation. SIMPLE++ is fully object oriented: the graphical user-interface, system architecture and implementation all conform to the demands of object orientation. Very different and complex systems and business processes can now be modelled accurately and true to life. Subsequently, SIMPLE++ is used in a wide range of applications in most segments of the economy. With SIMPLE++, you can simulate entire enterprises, technical systems and business processes. There is now no need to address each of these areas separately using different and incompatible systems.

A major strength of SIMPLE++ is the significant increase in productivity when building, changing and maintaining models. The most powerful features of building block, list and language concepts are provided in a single, powerful, integrated simulation environment. Features of object-orientation like inheritance, hierarchy, reusable objects or even models as part of models etc. make SIMPLE++ extremely efficient. These and more features are described in Section 2 and the intuitive, graphical and object oriented user interface of SIMPLE++ provides functionality and user-friendliness that overcomes former barriers.

2. SIMPLE++ the product

2.1 Design philosophy

Well engineered software should incorporate proven design fundamentals. SIMPLE++ has been developed according to the requirements in functionality shown below. Next to each are listed the key points of the implementation for which further description appears later in this chapter.



Graphical and intuitive model handling to increase **productivity** is a key requirement for the widespread adoption of simulation throughout the economy. In SIMPLE++ this is achieved with a unique integrated and incremental working environment, individual libraries (application templates), inheritance and a graphical object interface.

Users should be **independent,** and so are free to make an infinite range of tailored Application Objects which serve as templates for efficiently creating models. **Individuality** ensures that the simulation model has a lifelike animation and functionality. This functionality is facilitated by the use of hierarchy, the SIMPLE++ built-in language SimTALK and other features.

Quality and Innovation must be embedded by design. With SIMPLE++ this is achieved by the full object orientation and also by the use of industry standards.

Open Architecture and Integration requires the use of standards and the capability of exchanging data in real-time and communicating directly with other systems.

2.2 Innovative edge

SIMPLE++ is innovative in three key areas: Software technology, product features and its range of application.

SIMPLE++ is the software standard for <u>integrated, graphical and object-oriented modelling, simulation and animation</u>. SIMPLE++ is <u>fully object oriented</u>: Graphical user interface, system architecture and implementation all comply to the demands of object orientation. Very different and complex systems and enterprise processes can now be modelled accurately and true to life. The significant increase of productivity in <u>building, changing and maintaining models</u> is a major strength of SIMPLE++.

Object orientation is a modern and productive software technology. Technology per se has no value unless it has beneficial consequences for product development, the application and the user. The use of **object orientation** in developing SIMPLE++ has the following benefits:

- **High development speed and higher quality simultaneously**.
  Our experience with object orientation shows that our productivity is increased by factors ranging from 3 to 20. That means that the development team of SIMPLE++ is significantly more productive compared to teams using traditional techniques. The volume of the software code is reduced by several factors versus software with the same functionality but traditional technology. This also leads to a reduction of software bugs.

- **Longer product life cycle and better protection of your investment**.
  Software using traditional design and implementation methods can quickly reach a point where complexity is no longer controllable. In SIMPLE++ complexity is limited only by design.

- **Change management and enhancements in SIMPLE++** are easily effected by adding new objects. These communicate with existing objects by defined interfaces. Thus market requirements can be implemented quickly with a minimum of effort.

These are aspects of a software which cannot be seen easily by looking only at the functionality. Nevertheless, functionality has high priority. **In addition to the innovatory concept of the total product, SIMPLE++ offers some remarkable features:**

a) Integrated, graphical environment
b) Hierarchy
c) Inheritance
d) Object Concept
e) Changeability and maintenance
f) Integration and openness

These features are described in more details below:

a) <u>Integrated, graphical environment</u>

When working with traditional simulation software you have first to build your complete model. Next you are able to run simulation and finally, the simulation file is used to depict the process (animation). You cannot change the model during simulation/animation despite this stage being the one at which modelling bugs can most easily be identified. This means that you are forced by the system to work in a procedural manner.

The integrated, graphical environment in SIMPLE++ means that all functions are available at any time and all information about the model is graphically represented. You can simulate and animate parts of the model during modelling. You can even execute, test and debug controls without running simulations. This significantly increases modelling productivity and user acceptance.

b) <u>Hierarchy</u>

Hierarchy is needed to build lifelike and structured models which resemble the real world. Entire enterprises, complex distribution centres or even total national railway networks can be modelled realistically and in any accuracy. Managers and engineers can understand and examine the model at the appropriate hierarchy level without loosing the overall view and increases the efficiency of communication

Hierarchy is created and reduced dynamically by nesting or deleting objects. This means that models can be detailed or simplified during the planning process at any time without having had to consider it in advance. The user defines, without any constraints, the structure and the modularisation in SIMPLE++. <u>Thus you create hierarchy dynamically without pre-planning.</u>

c) <u>Inheritance</u>

<u>Inheritance</u> is a very important and productive feature of SIMPLE++. It allows the user to generate and update models very quickly and securely. **In SIMPLE++ inheritance can be controlled by the user up to the parameter level.** Definitions and changes at one point will be automatically installed at all other related points. Alternative models which you need for optimisation are effectively created and updated. Maximum efficiency and minimum faults are the result. Using inheritance with SIMPLE++ gives a productivity increase of a factor of 10 or more

Inherited instances of models and classes are not copies - a copy does not maintain a relation to its origin. In SIMPLE++ the new instance (child object) stays in a user controlled relationship to the class (parent object) and any aspect of this relationship can be updated at any time.

To illustrate the efficiency and security of using inheritance consider the change from traditional production to a just-in-time production, when the entrance buffer of dozens of machines and resources might become obsolete. In a SIMPLE++ model the entrance buffer is deleted in the parent unit and all child units are updated immediately and automatically.

d) <u>Object Concept</u>

The Object Concept of SIMPLE++ is unique. **Any application specific object can be graphically and interactively created from generic Basic Objects**. In this way specific libraries (templates) containing Basic and Application Objects are constructed. You can use and update the Application Objects as required and according your needs. As a result, you will always work productively and with well structured models.

e) Changeability and Maintenance

The ease and speed of building, modifying and maintaining models is very important. Since simulation often runs in parallel to the planning of systems, the necessary information and data required for modelling is frequently unavailable during initial stages. Having built a first draft of a model, you are invariably faced with having to make adaptations and changes. **Prototyping and the incremental detailing and adaptation to the final model are well supported by SIMPLE++.** Using traditional software, it's often the case that even a minor change demands a completely new implementation or at least, a great deal of effort and time consumption. This problem was frequently responsible for the lack of acceptance and justification of simulation projects in the past.

f) Integration and openness

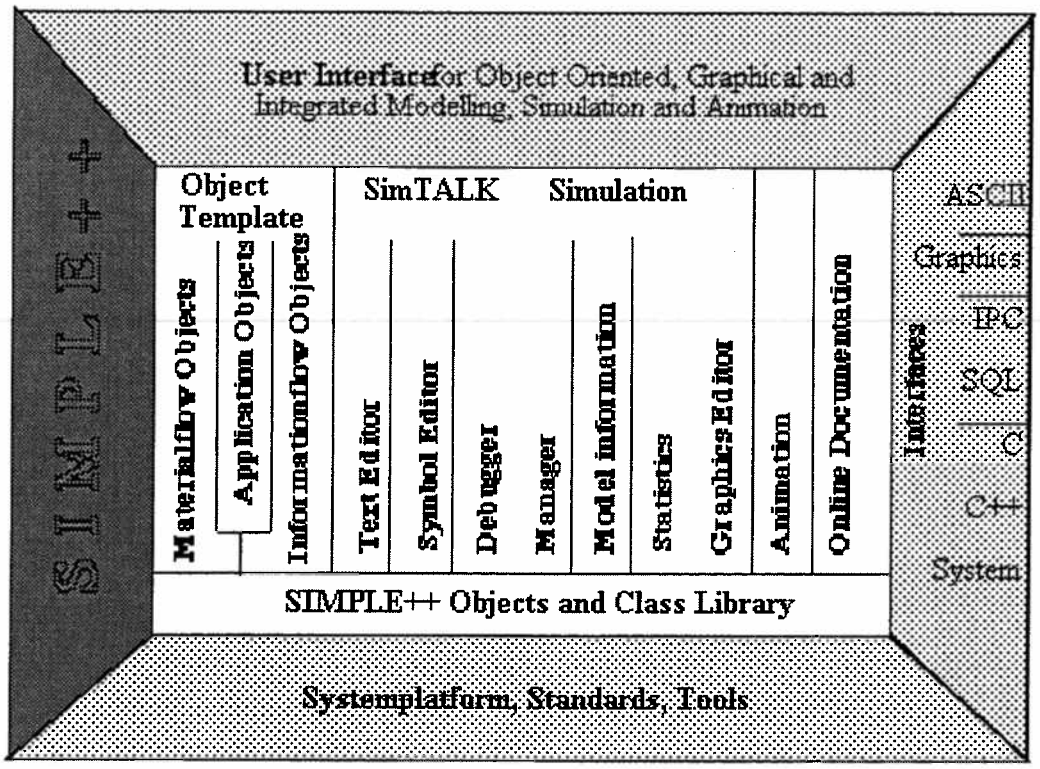
Standards and interfaces of SIMPLE++ make sure that integration and openness become a reality. **The following interfaces are available:**

- ASCII Files
- SQL Database
- Inter-Process-Communication
- Graphics
- C and C++
- Operating System

The real-time data exchange during simulation and the full external control of SIMPLE++ by other programs are important features for integrated solutions.

The full object oriented software technology, the remarkable features described above along with those described beneath '2.4 Specifications', summarise the functionality of SIMPLE++ which enables such a wide spectrum of application which are listed in section 3. Together, these remove the limitations of previously available software and characterise the innovative edge of SIMPLE++.

2.3 Description



The above picture shows the **architecture of SIMPLE++.** The elements of the architecture are described below.

The strength of SIMPLE++ lies in the ease and speed of building, modifying and maintaining models. High flexibility in modelling stems from the material and information flow being modelled independently. The required connections are implemented as a sensor-actor-concept. At the entrance and exit of an active material flow object is a sensor which activates a control where the desired actions are defined.

SIMPLE++ is software designed for use in technical application fields. Common **standards of technical application software** are used and supported. For example, SIMPLE++ is implemented in the language C++ under UNIX and X-Windows/Motif and is therefore easy to port to other **system platforms**. SIMPLE++ is currently released on all popular workstations and on personal computers with SCO-UNIX. The Windows NT version will be released in the first quarter of 1995.

SIMPLE++ offers a user friendly **Graphical User Interface** for interactive and integrated operation. All functions of modelling, simulation and animation are accessible at any time creating an 'integrated environment'. A 'non-procedural operation' means that you don't have to consider simulation flow when building SIMPLE++ models. Additionally, SIMPLE++ offers "incremental operation" which means that models or their parts can be detailed or simplified at any time. So if you have to implement changes during a simulation project, then you can do so immediately without having to change the entire model or build a new one. The productivity of modelling in this manner is usually greater by a factor of 3 or more. The integrated, non-procedural and incremental environment of SIMPLE++ allows you to work according to your own ideas and the requirements of real life and not according to procedures enforced by the simulation system. Last but not least, SIMPLE++ provides the capability to build individual, application specific dialogue masks for each object.

The **Interfaces** of SIMPLE++ allow you to exchange data in real-time, communicate with other programmes or call existing routines. The integration capabilities of SIMPLE++ are very important when using simulation to, for example, support the daily operation of an enterprise. In particular the exchange of data in real-time and the full control of SIMPLE++ by other software like MRPII can be highlighted. The following interfaces are available: ASCII file, Graphics, IPC (Interprocess Communication), SQL Database, C, C++ and Operating System. These interfaces make SIMPLE++ genuinely open and easy to integrate.

The **Object Template** contains both Basic Objects and user-defined Application Objects. All are visible as icons.

Application Objects are important for productive modelling. Any Application Object is graphical and interactively built by the user and stored in the Object Template for subsequent use. Because of the object orientation of SIMPLE++ you don't have to extensively modify an Application Object to use it for different models or applications. The user can create application specific Object Templates just by configuring Basic and Application Objects. An Application Object can be a network of Basic Objects (material and information flow objects), an entire model or sub-model and any combination of Basic Objects and models.

In the standard configuration, SIMPLE++ provides some powerful sets of Application Objects. An extraordinary feature of these Application Objects is their "user openness". The user can modify the Application Objects because they are open SIMPLE++ models built from Basic Objects. So even if the Application Objects provided do not match your requirements, they can be adapted or used as reference objects. These help you to make a most productive start with SIMPLE++.

The following sets of Application Objects are included in the SIMPLE++ standard configuration:

SIMPLE++_controls

SIMPLE++_controls is a set of Application Objects which model standard control strategies without programming. The following Application Objects are provided in the application template for interactive use: OR, RANDOM, PERCENT, PRIORITY, SEQUENCE, ALTERNATING, SET, IF, PARAMETER and VARIO. The Basic Objects of SIMPLE++ possess a defined standard behaviour when a part enters and leaves them. If a different behaviour is required, you can use the Application Objects of SIMPLE++_controls. These are represented by graphical symbols in the template as usual, and can be used in any model. By using the connector, you define its relationship to the material flow. The Application Objects can be combined with each other in order to get additional functionality depending on the chosen combination.

SIMPLE++_SFS

SIMPLE++_SFS is a set of Application Objects which represent the most frequently used material flow components of a real material flow system and are more complex than the SIMPLE++ Basic Objects. The following Application Objects are provided in the application template: Straight, bend, branches, confluence (local and global).

SIMPLE++_staff

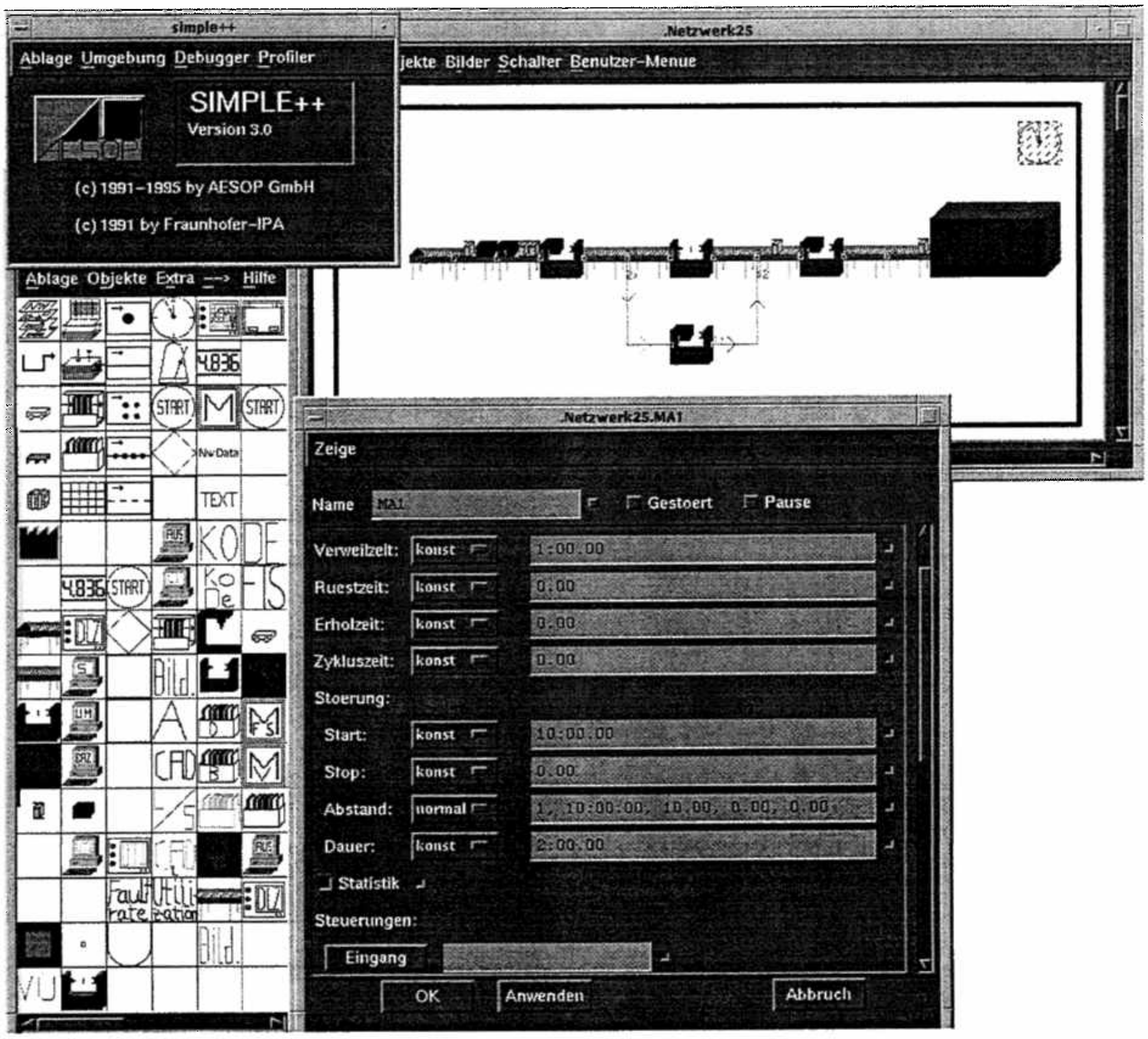
SIMPLE++_staff is a set of Application Objects to model the dispatching of production staff in an effective and structured manner. The following Application Objects and functionalities are provided in the application template for interactive use: Staff pool, qualification profile and priorities, working time model (shifts, pauses etc.), a matrix of distances and times, description of the working place, staff statistics.

SIMPLE++_AGV

SIMPLE++_AGV is a set of Application Objects to quickly and easily model automated guided vehicle systems (AGV). The following Application Building Blocks are provided in the application template for interactive use: Vehicle, path, curve, distribution, collection, intersection, central station, load station, central control and local controls.

Special sets of Application Objects like SIMPLE++_process (chemical industry), SIMPLE++_shop (Shop Control) etc. are available as optional products (refer to 2.5 Optional products).

Below you can see as a screen shot, an application template to the left-hand side. At the bottom, you can see the dialogue window in which you can type the parameters of an Object and in the upper right-hand corner, a new Object is being built.


simple++
Ablage Umgebung Debugger Profiler
SIMPLE++
Version 3.0
(c) 1991-1995 by AESOP GmbH
(c) 1991 by Fraunhofer-IPA
Ablage Objekte Extra --> Hilfe
.Netzwerk25
jekte Bilder Schalter Benutzer-Menue
.Netzwerk25.MA1
Zeige
OK
Anwenden
Abbruch

Name	MA1	Gestoert Pause
Verweilzeit:	konst	1:00.00
Ruestzeit:	konst	0.00
Erholzeit:	konst	0.00
Zykluszeit:	konst	0.00
Stoerung:		
Start:	konst	10:00.00
Stop:	konst	0.00
Abstand:	normal	1, 10:00:00, 10.00, 0.00, 0.00
Dauer:	konst	2:00.00
Statistik		
Steuerungen:		
Eingang		

You can position, parameterise and connect individually designed Application Objects and Basic Objects and by doing so, you can create a realistic simulation model. You determine the number, design, function and behaviour of the objects. The objects are represented by icons in the animation layout .

SIMPLE++ Basic Objects							
material flow				information flow			
movable		immovable		movable		immovable	
active	passive	active	passive	active	passive	active	passive
• vehicle	• part • container	• processor • line • conveyor	• storage • path		• data	• control • generator	• lists • displays

With the **Basic Objects for material and information flow**, the user is able to design any Application Object. The **Material flow Objects** can be divided firstly into moveable and unmoveable and secondly, into active and passive line, conveyor, storage, path, vehicle, container and part. The **Information flow Objects** are: Control, generator, lists, displays and data. The comprehensive capabilities of SIMPLE++ for information processing stem from the features of the information flow objects, the powerful built-in language SimTALK and a large number of functions, data types and operators.

Material and information flow are linked together by a sensor actor concept. A sensor e.g. at the entrance of a machine (material flow) activates a control (information flow) in order to control the related process.

A set of standard controls is provided with the SIMPLE++ standard configuration. Special controls are defined by the built-in Language **SimTALK** by the **Text Editor** or in a guided and graphical manner, without typing the syntax, by the **Symbol Editor**. Local controls for Application Objects or global controls for the total model ensure that the information processing in your simulation model is modular and well structured. Controls can be executed and tested without running the simulation. The capacity to trace and debug a model in the integrated environment is one of the most powerful features of the **debugger** built into SIMPLE++. Modelling errors can be effectively identified and rectified.

The **Simulation** can run without animation for systematic experiments which the user can easily manage. For experimentation, SIMPLE++ models can be pre-loaded with any

information and status. A model status can be stored during simulation interactively by the user or as a programmed event, which may result from a change of status. The status can be subsequently re-activated at any time. The simulation speed can be adjusted by the user or the user can 'step' through each event.
The simulation **Manager** in SIMPLE++ controls the simulation. It can be pre-set by the user or programmed during simulation.

During simulation, **Model information** can be displayed. Single parameters are shown via digital displays. Multiple parameters are shown concurrently over the time via the analogue display. SIMPLE++ offers various **Statistics over various periods of time** - interval, current and total simulation. These include utilisation, preparation time, wait time, down time etc..

The **Animation** runs on-line to the simulation. Selection of the different animation pictures for each object can be through a change of state or event driven. In this way changes can be visualised dynamically during simulation. These features are important for both, presentation purposes and verification. The pictures are defined by the user through the **Graphics Editor** or simply copied from a picture template.

By pressing the help-function the **On-line Documentation** appears on the screen. This does not render the reference manual obsolete but provides helpful information immediately and in a relevant context.

2.4 Specifications

Clear specifications will help you make the right decisions. Below are listed the specifications of SIMPLE++ V.2.3 which was released in March 1994 . You may not fully appreciate SIMPLE++ until you have seen it but a comparison of features will highlight the differences between SIMPLE++ and other products.

User interface

- object oriented, graphical user interface with window, menu and mouse technique.
- all model information is graphically represented and accessible.
- application oriented dialogue and plausibility check
- user definable dialogue masks
- Screen control by 'Zoom' and 'Scroll'
  "integrated working environment": All functions for modelling, simulation and animation are available all the time
- 'Incremental operation': Detailing or simplifying of the model step by step as requested
  'non-procedural operation' modelling according to the user's preference and independent of the simulation flow
- on-line documentation

Modelling

- model size not limited
- detailing or simplifying of a model at any time
- Object Concept with Basic Objects and user definable Application Objects
  template with Basic Objects for material and information flow:
  Processor, conveyor, line, stock, vehicle, path, container, part,
  lists (2D, stack, queue, random), generator, controls, display and
  information objects
- templates with Application Objects for very productive modelling:
  Standard Controls, Staff Pool, Automatic Guided Vehicles etc.
  picture template containing various pictures to represent objects
  in the animation layout
- graphics editor to define and change individual animation pictures per object which are changed event or state driven during animation
- inheritance controlled by the user at parameter level
- unlimited model hierarchy and process structures
- fixed and free attributes for each object. All attributes can be processed
- Material- and information-flow can be modelled independently

- disruption generator per active object
- unlimited independent Random Generators
- stochastic distribution of parameters
- flexible and powerful information management. More than 50 mathematical, textual and logical functions.
- SimTALK object oriented built-in language for individual controls
- interactive test and correction of controls without running the simulation
- Debugger to debug controls and support model verification
- Information support

Simulation / Animation

- animation on-line to simulation
- animation switchable on/off
- animation speed selectable: step or variable speed
- any preloading at the beginning and storing of model status during simulation
- control by external systems like MRPII
- data exchange during simulation
- statistics for total simulation time, intervals and actual runtime:working time, preparation
- time, break down time, blocked, throughput time, capacity load (min/max)
- change of model and parameters during simulation/animation
- step mode to follow the simulation/animation events
- individual collection and presentation of model and simulation values
- free definition of the animation layout
- event driven and interactive change of the animation pictures
- visualisation of hierarchy levels by opening objects

Others:

- SIMPLE++ Reference Manual
- SIMPLE++ Tutorial
- file interface (ASCII)
- system interface
- graphics interface (PPM and TIF)
-

Optional products:		
	SIMPLE++_C	"C" interface
	SIMPLE++_C++	"C++" interface
	SIMPLE++_IPC	Inter process comm.
	SIMPLE++_SQL	SQL database
	SIMPLE++_shop	Shop Control Model
	SIMPLE++_gantt	Gantt Charts
	SIMPLE++_process	Chem. Industry
	SIMPLE++_GA	Genetic Algorithm

System requirements
Workstations or PC with UNIX and X-Windows/MOTIF, > 16MB memory, 100MB free disc capacity, floppy, 256 colour graphics and a resolution of > 1024 x 768, mouse and keyboard. Serial interface for SIMPLE++ security device.

Released Platforms
DECstation5000 / ULTRIX
DECalpha AXP / OSF1
HP9000-4xx and -7xx / HP-UX
IBM RS6000 / AIX
SGI indigo / IRIS
SUN SPARCII / SUN OS and SOLARIS
PC / SCO-ODT
PC / WIN-NT

Important Features of SIMPLE++ which should be benchmarked:

We know of no other simulation software which compares to SIMPLE++. If you have knowledge of or are seriously considering purchasing a different product, then please use the form below to conduct a personal evaluation.

● Features	● Benefits	SIMPLE++	Others
hierarchy	reflect reality structuring	☑	☐
Inheritance	efficency less faults	☑	☐
Integrated, graphical and object oriented enviroment	user acceptance productivity	☑	☐
Specific user-defined application templates	reuseability modularity fast modelling	☑	☐
User-open application objects	adaptable reference objects	☑	☐
Control language with debugger	universal and flexible application	☑	☐
incremental modelling	fast, actual	☑	☐
Efficient modification and maintenance of models	actual models low costs	☑	☐
Interfaces open architecture	integration real time data exchange	☑	☐
Full object oriented technology and standards	innovation speed, security for the future	☑	☐

2.5 Optional products

The standard configuration of SIMPLE++ contains some sets of Application Objects. They are SIMPLE++_controls, SIMPLE++_staff, SIMPLE++_SFS and SIMPLE++_AGV. Details are described in "2.3 Description". In addition and at extra costs the following options for SIMPLE++ are available:

SIMPLE++_C
"C" Programming Interface for SIMPLE++.
The user is able to define and integrate special C-routines. These user defined routines can be called up like internal functions (sin, round,...). All basic data types available in 'C' and SIMPLE++ can be used. It is not possible to activate the internal methods of the SIMPLE++ basic building blocks.
Prerequisite: C-compiler, SIMPLE++

SIMPLE++_IPC
Interprocess communication with SIMPLE++.
SIMPLE++_IPC allows you to communicate directly with other software systems. The appropriate SIMPLE++ functions (e.g. start, stop, delete create, data exchange etc.) can be activated by Remote Procedure Calls (RPC's). Therefore SIMPLE++ can be controlled externally. This means that SIMPLE++ can be used as a simulation module in a complex application system.
Prerequisite: SIMPLE++, TCP/IP connection

SIMPLE++_SQL
SQL-Database Interface for SIMPLE++.
SIMPLE++_SQL allows the direct, bi-directional data exchange with SQL-Databases during the simulation. For example orders or work schedules can be transferred from an MRP-system to a SIMPLE++ model and the results of simulation can be transferred back. The SIMPLE++ library provides an SQL Object block which is used to establish the communication to an SQL-server on the network. To establish more than one connection at the same time you can duplicate the SQL Object block. SQL statements and data are sent from SIMPLE++ to the database. Data from the database are available in SIMPLE++ in automatically generated lists.
Prerequisite: SIMPLE++, Network.

SIMPLE++_shop

SIMPLE++_shop is a model structure which enables you to quickly and easily build a model which corresponds to shop floor organisation. The following Application Objects are provided in the application template for interactive use: Source, Workstation, Shop Control, Part. The user puts Workstations in the model and defines the parameters. Lists of orders, operation plans and resources are prepared for application. An order list or the Source can be used to load Parts into the system. The data of the order list can be imported from an MRP system. The Shop Control Object block controls the processing of the Parts according to the operation plans. All Application Objects are built from Basic Objects and are open for the user to make individual adaptations.

Prerequisite: SIMPLE++

SIMPLE++_gantt

SIMPLE++_gantt is the software for the graphical presentation and interactive manipulation of the order operation sequence or the occupation of the resources by using bar charts and an interactive user interface. The bars can be moved and by double clicking each one, a window opens to show related data which can be changed. The updated data is immediately depicted as bar charts and stored in the corresponding file. Input and output data of the Gantt charts are well structured ASCII files.

Prerequisite: SIMPLE++

SIMPLE++_process

SIMPLE++_process is a set of Application Objects to model continuous processes in the chemical industry. The following Application Objects are provided in the application template for interactive use: Source and Sink, Pipe and Tank, Mixture and Distribution, Batch Reactor, Continuous Batch Reactor, Container Stock, Resource Management, and Control. All Application Objects are built from Basic Objects and are open for the user to make individual adaptations.

Prerequisite: SIMPLE++

SIMPLE++_GA

SIMPLE++_GA is a software that uses "genetic algorithms" for finding optimised solutions considering different and competing goal functions. For example the optimal order sequence when the competing goal function of minimum throughput time, minimum stock capital and maximum resource utilisation are defined. The quality of the calculated solutions is depicted graphically. The user can stop the calculation process at any time. The best optimisation result found to a given point in time is available. The goal functions and constraints of the optimisation are defined by the user dependent on the type of problem

that is to be solved. Input and output data of SIMPLE++_GA are well structured ASCII files.
Prerequisite: SIMPLE++

2.6 History

SIMPLE++ has been developed by AESOP together with the Fraunhofer Institute for Production and Automation (IPA) and is based on more than 20 years of experience in developing and applying simulation software.

SIMPLE++ version 1.0 was released in February 1992. From that date until the release of SIMPLE++ Version 2.0 in February 1993, the software was installed over 100 times in Germany. In July 1993 when shipping SIMPLE++ Version V.2.1, AESOP registered more than 200 installations world-wide. In December 1993, SIMPLE++ version 2.2 was shipped and this was followed by SIMPLE++ Version V.2.3 in March 1994. You can expect a much higher development speed in the future. SIMPLE++ V.3.0 is planned to be released in the 4th quarter of 1994. Software, reference manual, tutorial and training material for SIMPLE++ are available both in English and German.

3. SIMPLE++ the range of application and some examples

SIMPLE++ is used in the planning of production, logistics and engineering as well as in the daily operation of enterprises to plan and control processes.

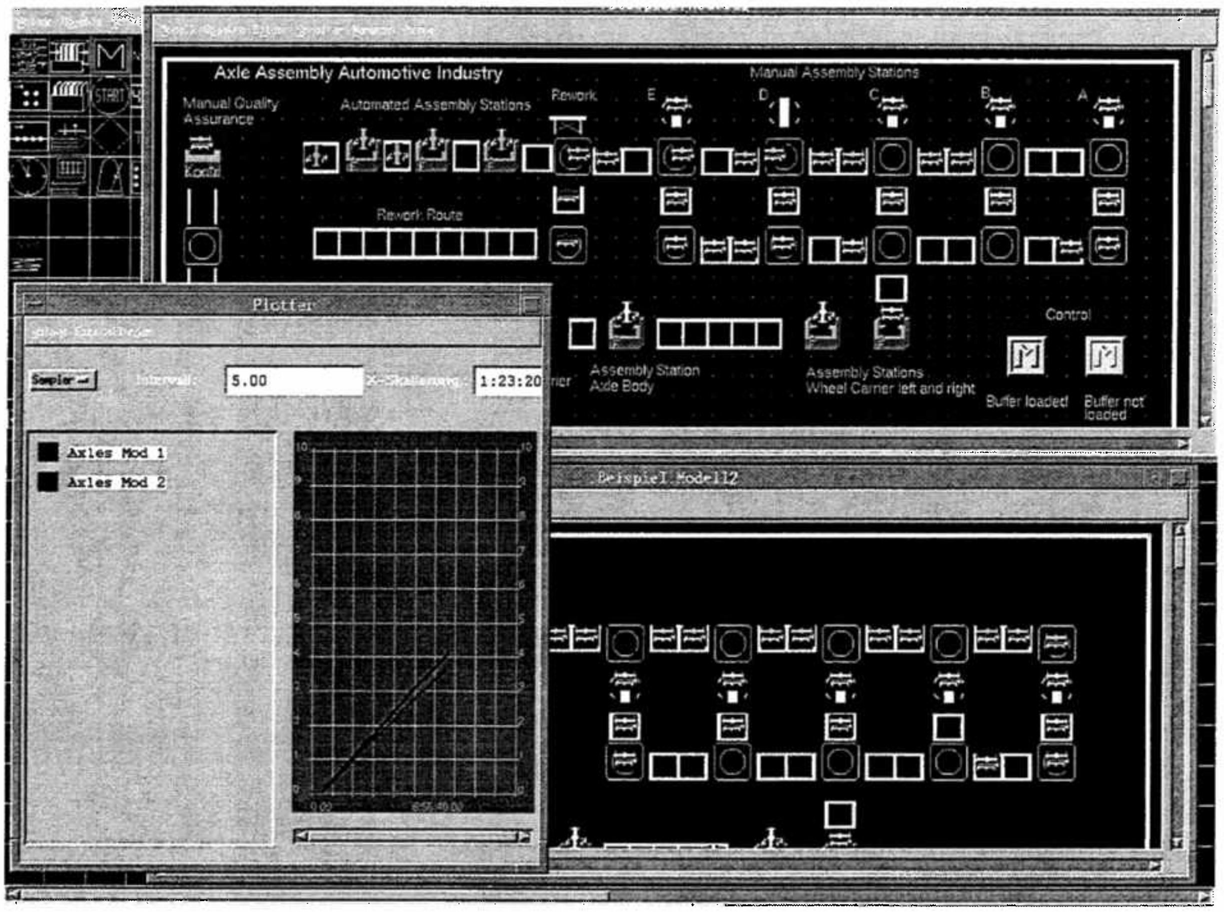
In the **planning process**, SIMPLE++ is used to dynamically plan systems in production, logistics and engineering in most segments of the economy. In this application area the structure, the dimension and the control of systems is optimised by SIMPLE++. Manufacturing systems, assembly lines, transportation systems, warehouses, picking of goods, packaging lines, hospitals, railway networks, traffic control, entire factories and distribution centres etc. are examples where SIMPLE++ is used successfully.

SIMPLE++ is also used for **Shop Control and Monitoring** in the manufacturing industry. This is performed by the same model that is built during the planning phase. The focus of this application is to look ahead at the operations (lifelike forecasting) for next time period (e.g. next 24 hours or next week) based on plans (e.g. customer orders and worksheets from a MRP-system) and data available at the time of simulation. The output of the simulation is a fine tuned operation plan (order schedules, lot sizing, staff dispatching, throughput times, stock and operation costs etc.) to prepare the staff with better knowledge with which to tackle the real situation.

The SIMPLE++ simulation model that performs the look ahead and forecasting is also used as a monitoring system to monitor the events of the operations in the lifelike SIMPLE++ animation layout. Inter-Process-Communication and real-time data exchange of SIMPLE++ are prerequisite features for that sort of application.

The following examples of projects carried out by us should show the range of different applications of SIMPLE++.

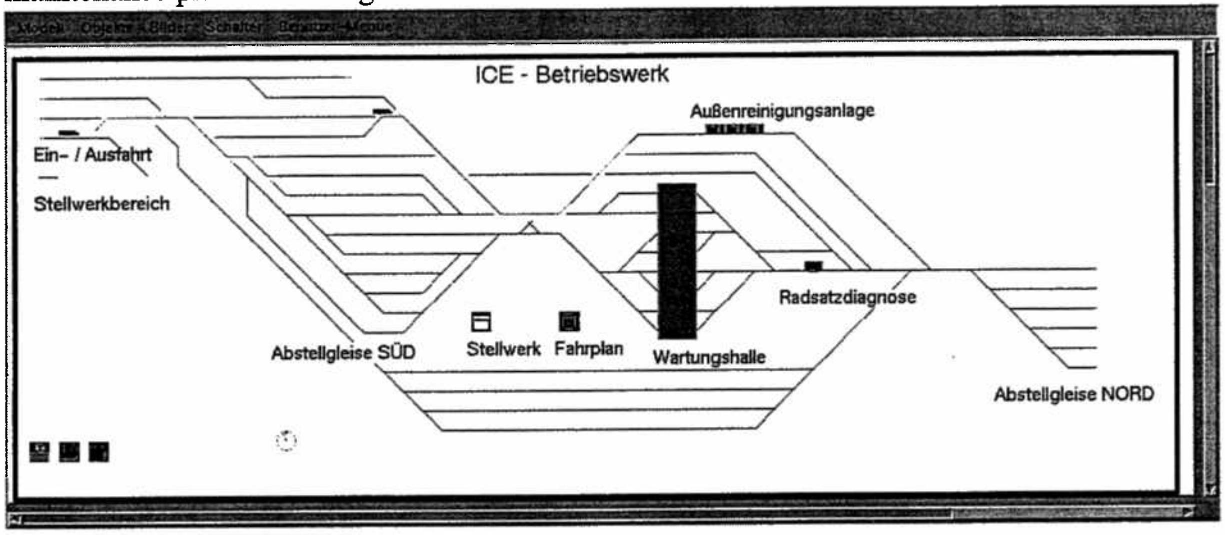
Axle Assembly in the Automotive Industry
This example shows the optimisation by simulation by varying layout, capacity and control strategies of a back axle assembly line at **Mercedes Benz** in Stuttgart.
The results of changes are shown graphically. Particularly the throughput of two competing alternatives that ran against each other is monitored. Any other parameter of the model could be monitored dynamically. The picture shows two alternative layouts for the manual assembly stations - the throughput is displayed dynamically and on-line to the simulation and animation.



ICE train maintenance

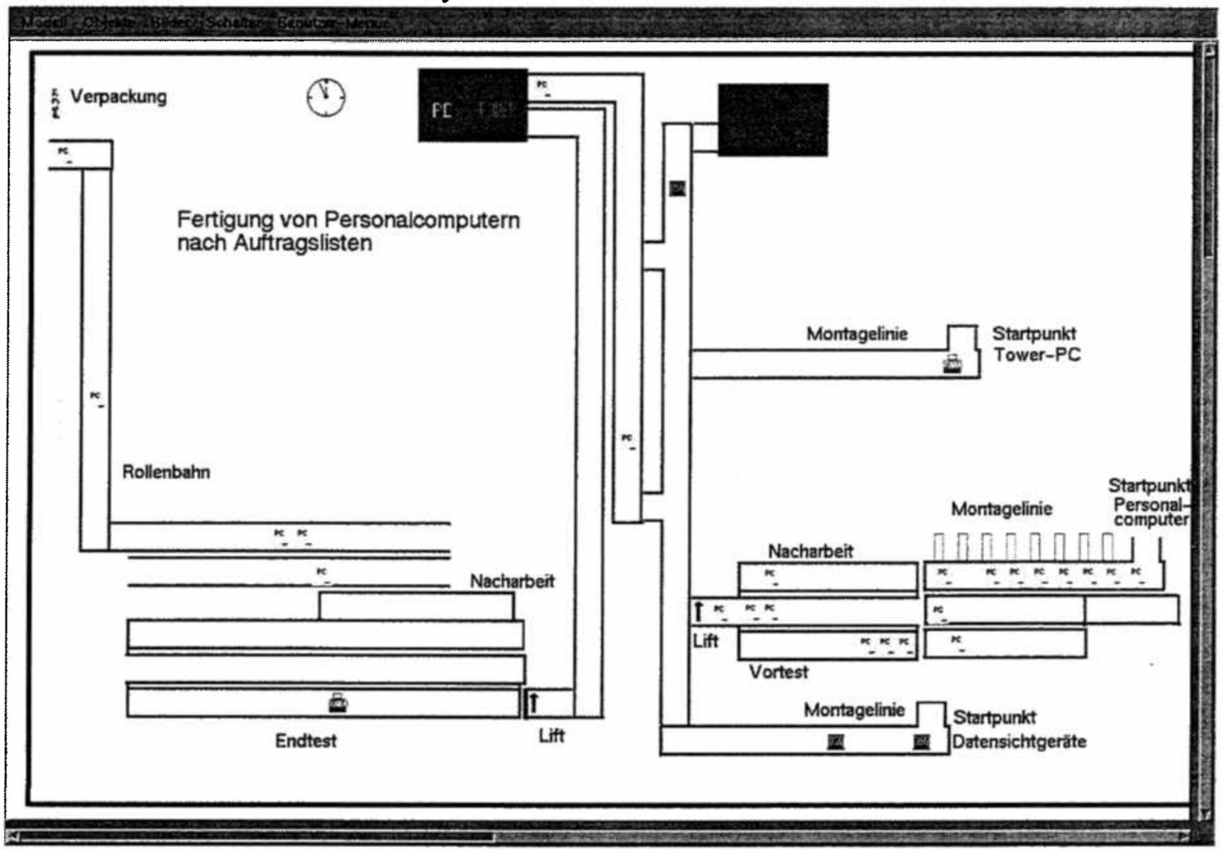
This example shows the maintenance plant for the ICE (Intercity Express) of **German Railways**. SIMPLE++ is used to optimise the scheduling of trains for all required maintenance resources. Each train costs about £21 million and thus the time for repair and maintenance should be kept to a minimum. Gantt charts are available to visualise the maintenance schedules. They can be changed interactively by the user in order to augment the scheduling and optimisation algorithms with their invaluable experience. The refined data of the Gantt Charts are used for the next simulation run to check the result of the changes. The access to their database by using the SQL-Interface of SIMPLE++ allows to exchange data in real-time.

Based on the current circumstances a realistic look-ahead is given by simulation in order to tackle the situation better. The outputs of the simulation are fine tuned operation plans and staff allocation which ensure that all maintenance is done and that the train leaves the maintenance plant according the time table.



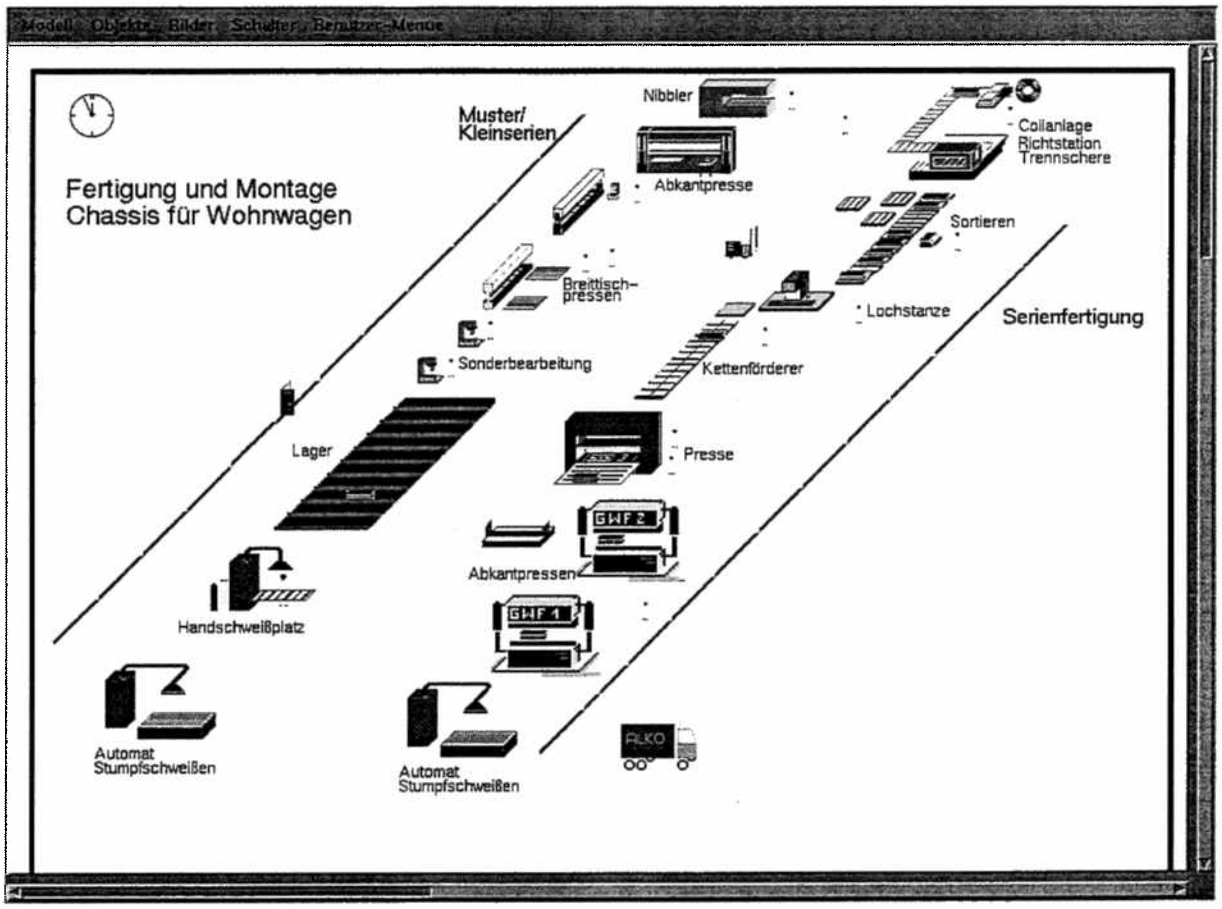
PC Assembly

The total PC assembly (labour, material flow, assembly) at **Siemens** in Ausburg is simulated in order to check the feasibility of the production plan generated by an MRP system. The daily operation is thus supported by using simulation. One result of the simulation is the list of all produced PC's during one shift with throughput times and other information. Different actions can be performed and tested by simulation before the real shift begins. This is a unique tool to get realistic forecasts and schedules with the consequence of reliable delivery dates. If quality and costs of products are at comparable level to competitors, then reliable delivery becomes a critical success factor. This critical success factor can be controlled by simulation.



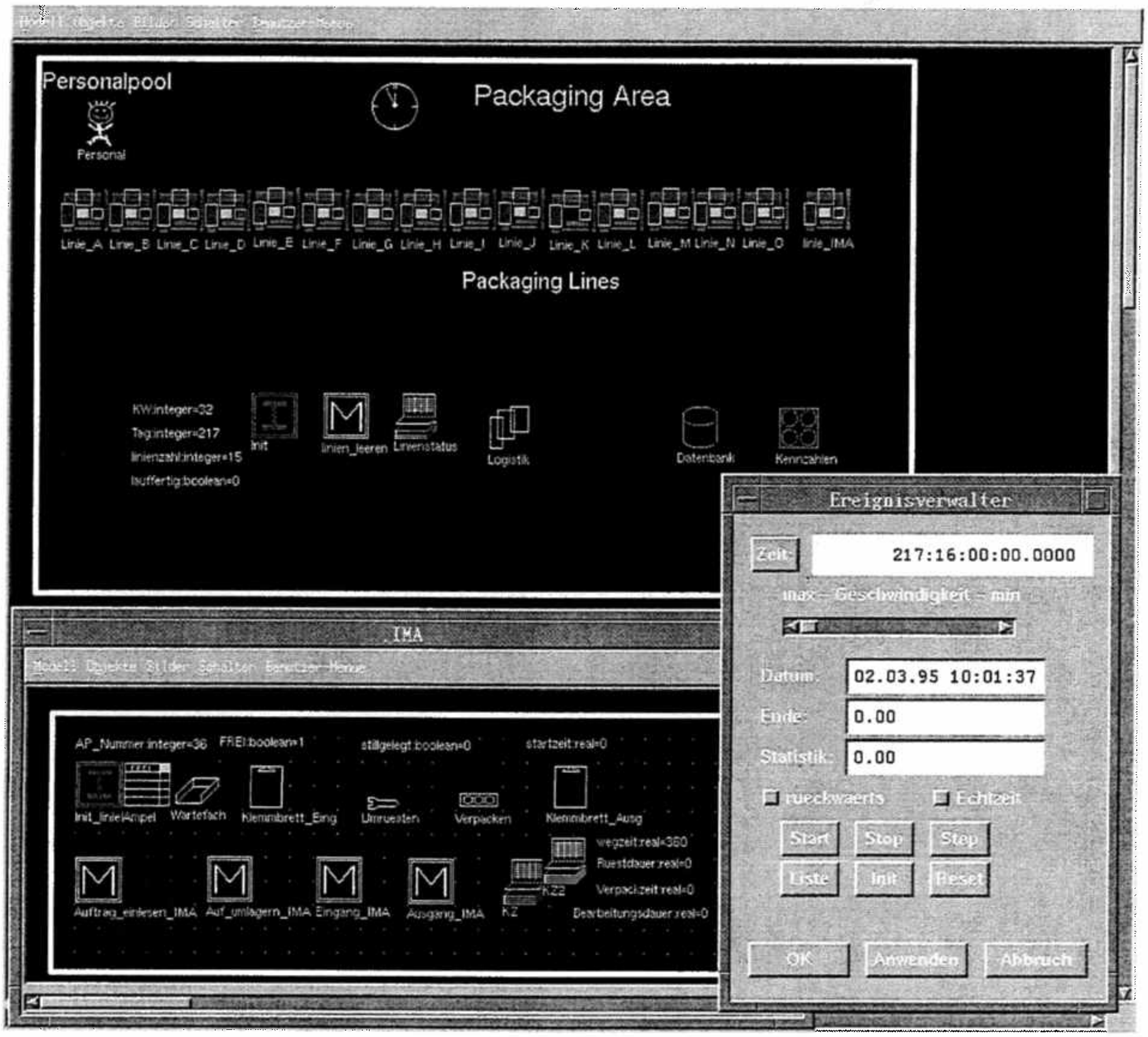
Production Planning in the Manufacturing Industry

The animation layout represents the production of **Alco** chassis for caravans. The production plan of one week is simulated very quickly and animated in the attractive and realistic layout of the plant. The purpose of this simulation is to get the optimal order sequence, minimum set-up time, optimised costs and the minimum throughput time for the production programme for one week. The throughput times are decreased significantly by using SIMPLE++. The picture below shows the animation layout and the parameter sheet of one machine.



Packaging in the Pharmaceutical Industry

The packaging area with automated packaging lines, labour, material and information flow is modelled and simulated to test different production and organisation scenarios. To achieve "Lean production". All scenarios are developed and tested using SIMPLE++. Simulation is performed in the phases of strategic planning (world-wide evaluation of locations) and of optimisation of the current facilities. Different qualification profiles of labour and varying shift models are tested to find the optimal production technique of the future. Even the local costs and specialities are considered to identify the best location for production world-wide.



KW:integer=32
Tag:integer=217
linienzahl:integer=15
lsuffertig:boolean=0

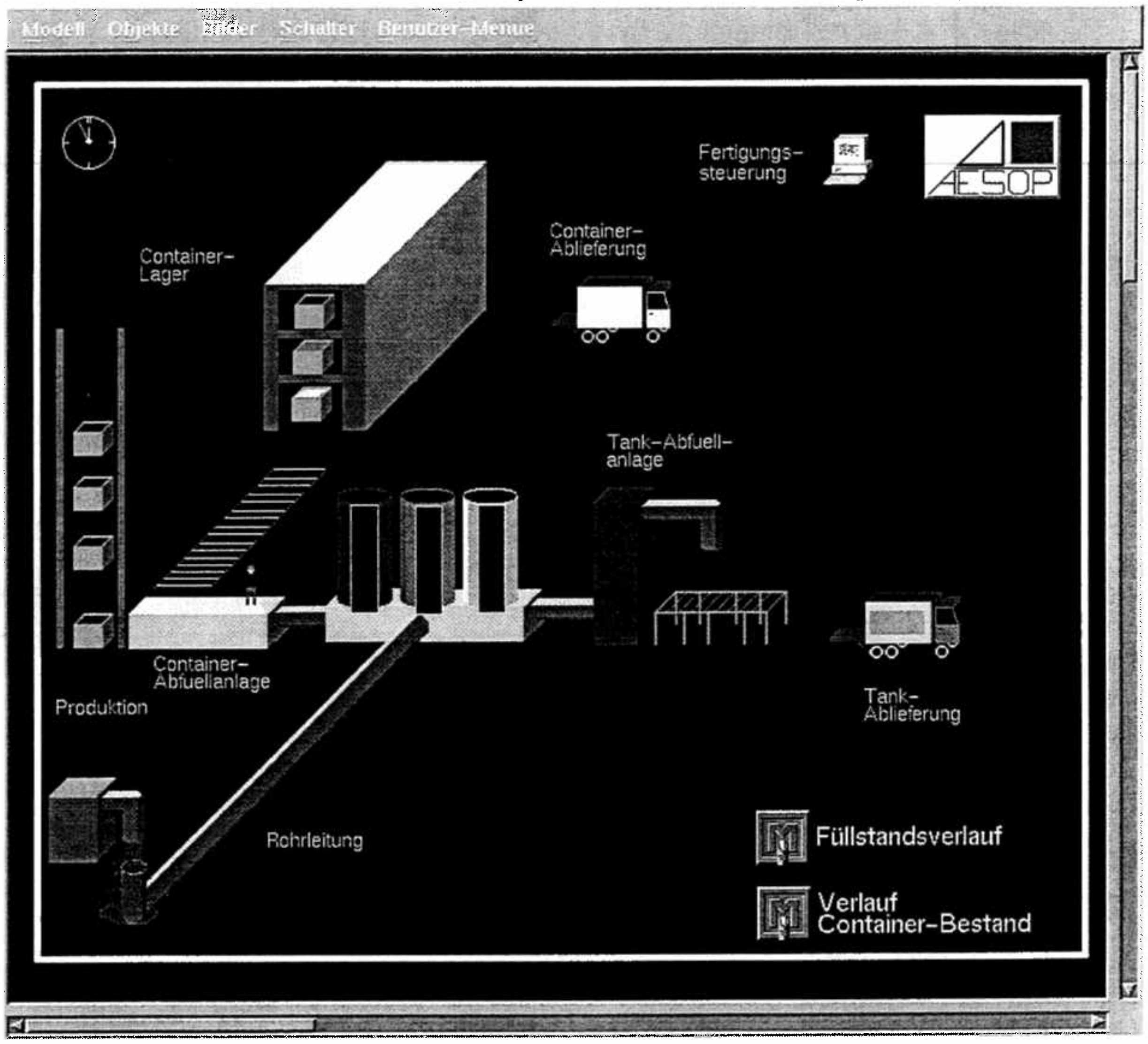
AP_Nummer:integer=36
FREI:boolean=1
stillgelegt:boolean=0
startzeit:real=0
wegzeit:real=360
Ruestdauer:real=0
Verpackzeit:real=0
Bearbeitungsdauer:real=0

Zeit: 217:16:00:00.0000
Datum: 02.03.95 10:01:37
Ende: 0.00
Statistik: 0.00

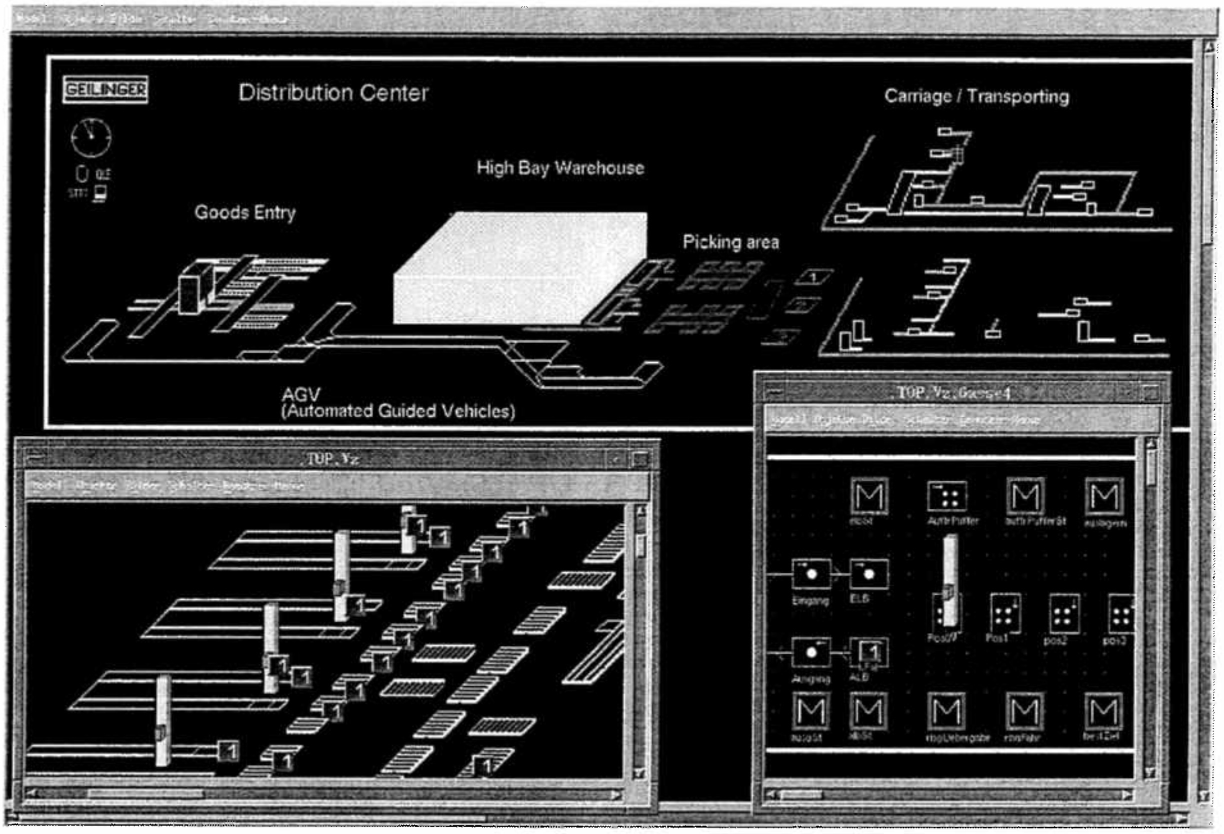
Production and Logistics in the Chemical Industry

Production, logistics and particularly the control strategies are developed, optimised and tested by using the below SIMPLE++ model at **BASF** in Ludwigshafen.

The advantages of SIMPLE++ in this application are lifelike animation, the transparent and modular structure of the controls and the dynamic visualisation of any process parameters.

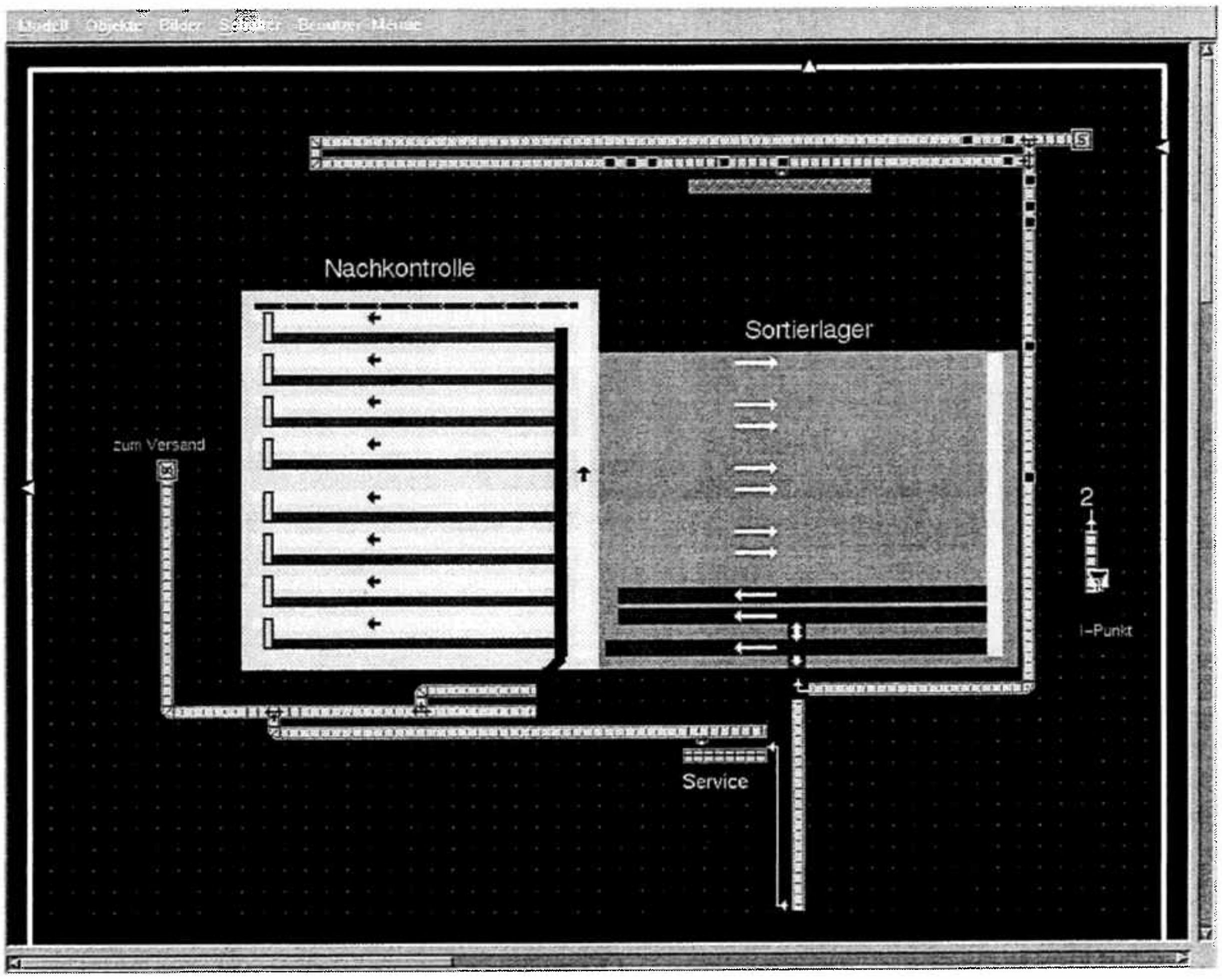


Distribution Centre
A large distribution centre is modelled in a very detailed way with some levels of hierarchy by **GEILINGER engineering,** Switzerland. Simulation was used during the whole planning and implementation process in order to optimise the system, process and communication on all technical and organisational levels. SIMPLE++ supported the decision making process.
To achieve the shortest modelling time for such a complex project, several people worked in parallel developing different sub-models. Each was tested independently before incorporation into the overall model. Hierarchy and inheritance had been invaluable throughout this project.



Planning of a new Sorting Storage

The objektive was to simulate the commissioning process to the point of dispatch for a finished parts warehouse holding 13 commissioning areas on two floors.
The commissioning areas, a sorting area including 105 sorting lines, the combinated final inspection and packaging area, the service area, dispatch area and a extensive driven transportation system make up the main components of the model. Another task assigned to simulation was to examine the functionality of the overall system renouncing 35 sorting lines. Further on, several strategies for the work flow management have been investigated. A testing was done on the employement of flexibly assignable comissioneers. Their assignment had to be considered according to the state of the overall system and queues in the particular commissioning areas.

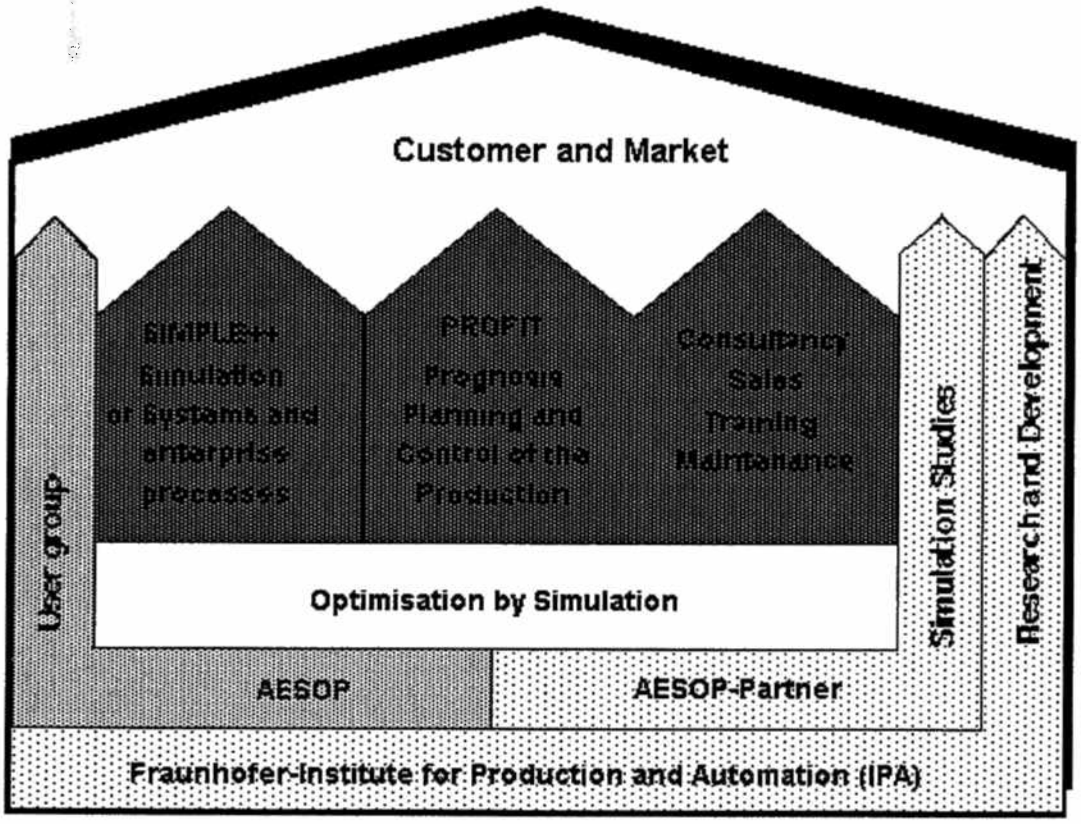


4. AESOP - the vendor of SIMPLE++

AESOP is a privately owned company and was founded in May 1990 in order to develop, produce, market and support simulation software for optimisation. Since that time AESOP has been growing continuously each year. All investments and growth are financed by the stock capital and the operating profit without any loan from a bank. Most of the profit is being reinvested in the company.

The management has many years of experience in developing software - particularly simulation software - applying simulation software and marketing of information systems. The staff is highly educated and motivated to keep SIMPLE++ the most attractive simulation software at the market and to provide first class service to our customers. All employees participate in the financial success of AESOP.

We tried to draw our activities into one solid building. At the top are the customers and the market. They are most important and drive our activities in the outlined frame.


Customer and Market
SIMPLE++ Simulation of Systems and enterprise processes
PROFIT Prognosis Planning and Control of the Production
Consultancy Sales Training Maintenance
Optimisation by Simulation
User group
Simulation Studies
Research and Development
AESOP
AESOP-Partner
Fraunhofer-Institute for Production and Automation (IPA)

At the foundation level is the Fraunhofer-Institute for Production and Automation (IPA). SIMPLE++ has been developed by AESOP together with the IPA and the mutual co-operation with the IPA is now even closer. The results found in research projects by IPA can be transferred to commercial products immediately by AESOP. In addition to the formal co-operation, we are happy to have Professor Dr. Ing. H.J. Warnecke, the president of all Fraunhofer-Institutes and the former president of IPA, in our advisory board.

Today AESOP and about 20 AESOP-Partners provide consultancy services based on SIMPLE++ and market the software SIMPLE++ world-wide.

SIMPLE++ is the only product line of AESOP. All resources are focused on it to provide solutions for "Optimisation by Simulation". We offer SIMPLE++ as a standard simulation package with application oriented options for the optimisation of systems and business processes. PROFIT is a modular software system based on SIMPLE++ for the prognosis, planning and control of production to optimise daily operations. Our solutions should help our customers to dynamically analyse systems and processes for better and safer decisions and increased profit. For all the products and solutions we provide first class consultancy, training and maintenance. "Optimisation by Simulation" is our mission.

5. Unseld + Partner

Since 1991 Unseld + Partner is a Partner of AESOP focusing on Simple++ as simulation software. Unseld + Partner has become the service company with the highest turnover of simulationsoftware in Austria.

The occupation of Unseld + Partner include planing and controlling of production lines and depots.
Together with FhG-IPA innovative project are worked out, for example refined production planing in steel construction. Furthermore, the themes metal-working and commissioning are in the centre of interest. Co-operative software developing companies and institutes work in the area of order volume optimising in der Fensterproduktion und Einlastungssteuerung in der Schleifmittelindustrie.

In 1995 following goals will be realised:

1. Construction of a net for demonstration centres for SIMPLE++ and PROFIT
2. Development of specific SIMPLE++ software modules in order to determine the optimal strategy for production planing and control bei complex Fertigungsabläufen
3. Installation of PROFIT at customers in Austria.

In addition to simulation on the industrial sector Unseld + Partner is engaged in simulation of freight centres and the logistics thereof within the ITF-Umbrellaproject „Logistics and Transportation“

Profit from PROFIT

I.A Walls

AESOP Ltd., PTMC, Marsh Lane, Preston, UK
Tel. (44) 01772 200380
fax (44) 01772 200404

Abstract: PROFIT is the commercial realisation of two ESPRIT supported projects. It is an integrated simulation based system for planning, controlling and monitoring production. The core of any application is an accurate, company specific model of the production process, constructed using the object oriented simulation tool, SIMPLE++.
T&D Automotive have recently commissioned a new factory, to Supply In the correct Line Sequence (SILS) car bumpers to the adjacent Vauxhall Motors factory. PROFIT was chosen to provide the complete solution for all information processing between Vauxhall, the group-wide MRP and the shop floor PLCs. It will schedule, monitor, control and trace the production flow and processes at T&D Automotive. A special focus was put on the integration of shop floor personnel in the planning and control of the factory. A key concept of PROFIT is that of "user openness" through which it is possible to involve people and to integrate T&D Automotive in the design and construction of their own solution. This paper provides an insight into this first implementation of PROFIT in the UK.

1. Background

PROFIT is the commercial realisation of the system initially developed during two projects receiving funding from the EC under the ESPRIT programme. It is an integrated simulation based system for planning, controlling and monitoring production. The core of any PROFIT application is an accurate, company specific model of the production process, constructed using the object oriented simulation tool, SIMPLE++.
This paper introduces PROFIT and the research from which it was developed. It then goes on to discuss the first implementation of PROFIT in the UK, at the factory of T&D Automotive.

1.1 ESPRIT - Shop Control / Real-I-CIM

Shop Control (ESPRIT project 5478) had as its objective the development of a suite of standard modules that would allow companies to quickly install and configure complete manufacturing and control systems. Its success was such that a follow-on project called Real-time Interactive CIM (Real-I-CIM) was commissioned (ESPRIT project 8865).

1.2 Partners and Pilots

The PROFIT product arising from these ESPRIT projects is a modular system with individual components developed by AESOP GmbH, DiCONSULT GmbH, Fraunhofer-Institut für Produktionstechnik und Automatisierung (FhG-IPA), Instituto de Engenharia de Sistemas e Computadores (INESC) and TECHNOTRON.
Pilot sites for the initial implementations were established at ARJAL, an automotive component supplier, and GROWELA a shoe factory. In both cases the results were positive. The current Real-I-CIM project features a pilot application at the BMW factory in Lohhof.

2. PROFIT

SIMPLE++

The discrete event simulation system SIMPLE++ forms the heart of any PROFIT installation. SIMPLE++ is to our knowledge unique in being the only fully object oriented, graphical and interactive simulation system commercially available. An accurate customised model of the factory is used for on-line monitoring and scheduling and off-line capacity planning. The object oriented nature of SIMPLE++ means that very complex systems can be modelled in an incremental and hierarchical way. A library of re-usable application specific objects ensures that the model can be built rapidly and, just as importantly, it can be easily modified and maintained.

PROFIT Characteristics

Figure 1 illustrates the modular architecture of PROFIT. The key features of PROFIT are:

- "user openness", which for a CIM system means the maximum involvement of the customer in the design phase and flexibility for the customer to make future changes and adaptations to the system themselves;
- excellent system ergonomics through the use of object oriented modelling techniques, benefiting users both on the shop floor and in the planning department;
- modularity based on a comprehensive set of CIM functions such as management information tools, scheduling, exception handling, monitoring, maintenance, DCE software administration, personnel management, quality management, traceability, shop floor data acquisition, shop floor control etc.;
- connectivity to legacy systems, especially MRP, PPC, CAP, SCADA etc.;

- fast design and implementation of a customer specific solution through highly productive object oriented modelling techniques, client server data integration and open architectures.

3. T&D Automotive - Requirements for the PPC system

3.1 Scenario

The T&D Automotive factory makes plastic car bumpers, having the capability to produce up to 2500 bumpers per day. Its main customer is the Vauxhall Motors plant situated next to the factory. T&D Automotive are contracted to supply bumpers in the correct line sequence (SILS) in a just in time manner. An order for a bumper pair of a particular colour and build-level arrives approximately once per minute through an electronic data link. These orders are received some three hours before they have to be delivered to the Vauxhall Motors line-side.
There is a multitude of variants possible. Each bumper can be one of 13 basic colours (with special order colours also possible); in addition each bumper can have head lamp washer jets, fog-lamps, two grille options, and can be for a saloon or estate car variant. It would be impractical to keep this number of variants in stock, yet three hours is insufficient time in which to make a bumper from scratch. The solution is to produce intermediate stocks of unpainted bumpers and painted bumpers, and only to proceed to the final assembly on receipt of an order from Vauxhall. A central requirement of the Production Planning and Control (PPC) System is to minimise the size of these intermediate stocks while ensuring that parts are always available to fulfil orders.

3.2 First make a bumper

The manufacturing process is as follows: the bumpers are injection moulded, loaded onto pallets and transported by overhead monorail to the unpainted bumper store. From there they are called-off under the direction of the planning and control system to be painted. The painting process consists of a plasma surface treatment followed by a two stage paint application, which is performed by robots. The painted bumpers are then held in store until the corresponding order appears on the call-off list, following which they are brought to the assembly areas (one each for front and rear bumpers). Once assembled to the appropriate level they are loaded on trolleys and shipped to the Vauxhall Motors line-side by electric vehicle.

3.3 Simulation

In the pre-production planning phase the designers of the factory and its various systems required simulation in order to prove that the different sub-systems would work together as planned and to investigate the operation of the factory in various contingency scenarios in which the production capability is impaired in some way.
Once the factory is operational, the simulation model continues to be of importance. At any time the model can be automatically initialised with the current work in progress and plant status, and a simulation run performed to investigate the future behaviour of the plant. In addition, the simulation model developed is re-used to provide the graphical front end for monitoring and, in an adapted form, forms the basis of the scheduling function.

3.4 Monitoring

A crucial aspect of operating such a factory is to have a detailed and up-to-the-minute knowledge of the work in progress and the status of various resources. This plant monitoring role also had to be integrated with the other management functions such as scheduling. Access to this monitoring function was required at various strategic locations around the factory.
An additional requirement was to construct an archive containing a record for each bumper produced in which the conditions at each stage of the process are recorded. This database formed part of the quality assurance system, giving complete traceability on individual parts.

3.5 Scheduling

The scheduling of bumpers through the plant centres on the minimisation of set-up (at injection moulding and at paint) while ensuring that sufficient intermediate stocks are available to fulfil orders. Should a bumper be scrapped at the final assembly stage where it has been allocated to a particular customer order, the scheduling system must expedite its replacement so that the fundamental SILS requirement can be met.

3.6 Flexibility

The entire planning and control system required sufficient flexibility to be re-configured easily to accommodate production route changes and product additions and changes. Examples of modifications that are likely in the future include the addition of further moulding machines to produce bumpers for different car types and the supply of unfinished bumpers and spare parts to other manufacturers and resellers.

4. The Team

The team formed to address the requirements of T&D Automotive, outlined above, consisted of the Fraunhofer IPA, DiCONSULT GmbH and AESOP, with AESOP (UK) Ltd. as the prime contractor.

4.1 Fraunhofer IPA

The Fraunhofer IPA have established themselves as one of the leading consultancy and applied research organisations in the area of factory planning, control and optimisation. Their experience of simulation and its application in on-line systems, is extensive. It was here that the predecessors of SIMPLE++ were devised and developed to the point where their commercial potential was evident, leading to the establishment of AESOP GmbH.

4.2 AESOP

AESOP GmbH was formed in 1989 to develop and exploit the simulation activity of FhG-IPA. AESOP (UK) Ltd was formed in 1994 to market and support SIMPLE++ and integrated simulation based systems in the UK and Ireland.

4.3 DiCONSULT

DiCONSULT, who are based near Munich, specialise in Shop Floor Production Planning and Control Systems. Their expertise lies in the acquisition and communication of production information from and to the shop floor. They use their own rugged industrial terminal system to act as the interface between people and machines on the shop floor and the higher level planning and control system.

5. The Solution

5.1 General Overview

The principal objective of the Manufacturing Planning and Control system at T&D Automotive is to control and enable the production of the bumpers. To achieve this a bumper pair (front and rear) meeting stringent quality requirements must be available for assembly at Vauxhall Motors just as the car to which they are to be fitted reaches the attachment station on the assembly line. To ensure that this objective was met, the entire T&D Automotive factory was first simulated using SIMPLE++, to identify potential bottlenecks and appropriate operating regimes.

The implementation consisted of the following modules:

- scheduling and forecasting system
- monitoring system
- traceability archive system
- shop floor control system
- interface to the machines, shop floor and planning personnel.

5.2 Off-line simulation

The first stage in implementing the production control system was to simulate the entire T&D Automotive factory so that its various operating modes could be validated. The simulation model, built using SIMPLE++, showed that part of the automatic transport system within the factory was incapable of achieving the required throughput with the parameters as specified. It is worth noting that due to the complex interaction of carriers competing for access to track sections and the rules governing how transport orders are allocated to carriers, it is impossible to predict the detailed behaviour of such a system using static design calculations. By experimenting with the model it was possible to show the implications of various suggested modifications to the system, helping to ensure that the final configuration met the requirements of T&D Automotive.

5.3 Scheduling and forecasting system

The use of simulation as the basis of a planning and scheduling system is rapidly gaining acceptance as the approach of choice in many manufacturing situations (see for example [1]). The scheduling and forecasting system plans the production of the required bumpers. It provides the information required to oversee production and to make decisions in the case of contingencies. In the normal working mode the primary route of material flow from the delivery at Vauxhall Motors back through the system to the injection moulding section is automatically scheduled and controlled. Orders are received through EDI link direct from the Vauxhall motors plant. Three different time horizons are used in planning the production at the T&D plant. These are: a three-week rolling forecast of anticipated production; firm orders for three shifts ahead (whose sequence is only about 80% correct); a 3 hour call-off signal, updated every 50 seconds, of actual bumper pairs required at line-side.
Scheduling optimises batch sizes at the injection moulding and paint plant while meeting the various competing system constraints (such as limited storage space for partially complete bumpers). Additionally, the SIMPLE++ system will incorporate human decisions for certain well defined contingencies and exceptions. The planning system also forecasts the use of material, future bottlenecks and implications of human decisions.

5.4 Monitoring

The monitoring sub-system presents to the customer information about the exact status of his plant and of the orders flowing through it. This information was presented through a graphical display derived from the simulation model of the factory. Colour encoded icons represented the production status of the various resources in the system. Each resource icon could in turn be opened to present further information about the production history for that resource and the work currently in progress there. A third, more detailed, level of information was available to the user by directly accessing the databases associated with the machines, orders and produced parts. Access to this data was made available through a GUI employing data filters for selectivity.

5.5 Low Level Control & Data Acquisition

Information about the status of each machine in the factory and information about the bumpers themselves as they move through their production process, were captured by dedicated industrial terminals from DiCONSULT. Data was acquired direct from the machine PLCs, through bar-code readers, through manual input and through direct digital signal inputs. In this way the high level production planning and control system can access up-to-date information about the exact status of the factory, presenting this to the operators and management through a graphical display. The planning system uses the information about resource availability and inventory at various production stages to produce a viable schedule. This schedule information and the consequent direct control of the various machines and transport systems were relayed to the shop-floor through the same DiCONSULT terminals.
The system has been designed in such a way that it is fault tolerant. Should the planning computer or the connecting network go off-line, the factory can continue to operate under the direct control of the terminals until the fault is rectified.

5.6 Operating Modes

The normal operating mode for the T&D Automotive factory is where all the resources are available and customer orders are coming into the system by EDI. In this situation the entire planning process is fully automatic, and the factory can deliver all bumpers required in the correct sequence. In addition various contingency scenarios were identified in which the ability of the factory to produce bumpers was impaired. Typical contingencies are due to machine failure (e.g. where one of the two moulding machines breaks down), or where raw materials are in short supply. In the former case the planning system was able to predict the time at which orders would fail to be fulfilled allowing the customer to be forewarned of potential problems. In the latter case the planning system recognised when such a raw material shortfall was likely and suggested to the production manager the best product mix in order to maximise the time before orders could not be fulfilled.

6. Implementation

The project to design and successfully implement the production Planning and Control System for the T&D Automotive factory was characterised by a number of key elements.

6.1 Time Scales

The time scales for the project were extremely compressed, with only some seven months from the inaugural project meeting to having a fully functional planning and control system. It is the author's contention that this was only made possible through the use of the modular object oriented software that forms PROFIT.

6.2 Customer Participation

An essential success factor in the implementation of the planning and control system was the close working relationship established with the customer. The key person responsible for the system implementation, the computer services manager, was seconded to AESOP for the duration of the contract. This enabled him to be involved in the day-to-day decision making process and to learn the details of the system's inner workings. This last point means that not only do T&D have the skills necessary to run their plant, they also have the ability to react to changing market situations by updating and modifying the system themselves - a rare feature in a planning and control system.
In addition a series of regular (approximately monthly) workshops where held at the factory site, which key personnel from T&D Automotive (including the Operations Director, Technical Director and the Technical Manager) and the AESOP consortium attended. These workshops were invaluable in reaching a rapid consensus on the various technical issues that arose during the project.

6.3 Functional Design Specification

The Functional Design Specification (FDS) was a detailed document defined and agreed during the early workshop sessions. It set out the functionality of the planning and control system - what it would do, what inputs were expected, what contingencies were considered, what the outputs were to be and the specification of the interfaces to the machinery supplied by T&D and other third party suppliers.

Although a mutually agreed version of the FDS was used as the formal basis of the implementation, the document itself continued to evolve as more detail was added. In this way the FDS also served as the basis for the technical documentation of the system.

6.4 Liaison with 3rd Party Suppliers

The equipment specified for the individual production steps can all, to some extent, stand alone. For an integrated factory the planning and control system acts as the bridge between these islands of automation. The interface to all the machinery therefore is a crucial aspect of the planning and control system. In an ideal world these would all be specified at the procurement phase; unfortunately for various reasons much of the equipment had already been purchased (without too much consideration of interface capabilities). Consequently, the need to have timely and effective communication with these suppliers was critical. The experience of this project was that this best achieved through face-to-face meetings at which the technical issues were quickly resolved.

6.5 Maintenance & Upgrading

AESOP's philosophy in implementing a production planning and control system is to give the customer the necessary skills to allow him to implement changes to the system himself (with recourse to AESOP should this be needed).
Routine system upgrades are performed remotely through ISDN routing giving access to the T&D Automotive computers from team member sites is Germany and the UK.

7. Where Do We Go From Here ?

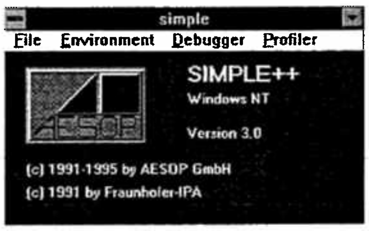
The Production Planning and Control System implemented at T&D Automotive is a success in at least two ways. T&D have a flexible solution to their operational requirement, which because of the nature of PROFIT was implemented within tight time scales, while at the same time being of lower cost than any of the "conventional" solutions proposed for the factory. Secondly, the project has demonstrated how the results of ESPRIT funded research projects can be effectively commercialised.
In terms of the implementation of the system some important points can be discerned. In particular it is difficult to understate the importance of developing a Functional Design Specification, of the type described above, in a project of this nature.
The future looks secure for simulation based Production Planning and Control Systems and in particular PROFIT. The final word should go to T&D Automotive's computer services manager:

"PROFIT has provided T&D Automotive with a manufacturing system that gives us three main advantages in a highly competitive automotive market.
Firstly, it is a complete manufacturing solution, with all parts of a traditionally disjointed system seamlessly integrated. The hierarchical nature of the installation, gives flexibility allowing changes to be accommodated easily and very quickly. However of almost greater importance it gives us easy access to the critical data being collected and processed on site.
Secondly, the ability of the whole manufacturing operation to run in a truly lean manner, with minimal stocks of all components held, directly reduces our costs to the customer ensuring that we maintain our competitiveness within the marketplace.
Thirdly, it provides a greater understanding of the major benefits that simulation can provide. This will allow us to plan the expansion of the current site , or the development of any new sites for new customers, with the confidence that we have gained from this installation. Using SIMPLE++ as the simulation tool has one major advantage over other conventional simulation systems in that the final simulation is not discarded but provides a core element of the implemented PROFIT system. This of course means that people are more inclined to invest the time in simulation knowing that it will directly benefit the end result.
I believe that the future of modern manufacturing systems is in complete integrated solutions, with simulation providing core functionality. No other system that I reviewed before finding PROFIT came close to the functionality that it has provided us.".

8. References

[1] "New Ways of Planning Production in the Process Industries", B-D Becker and S Nestler,FhG-IPA, 1995

SIMPLE ++

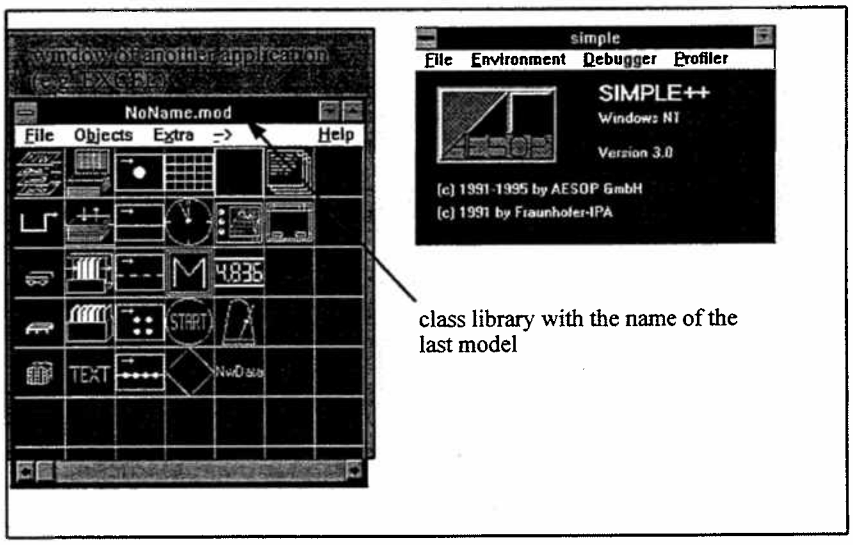


SIMPLE++
Windows NT
Version 3.0
(c) 1991-1995 by AESOP GmbH
(c) 1991 by Fraunhofer-IPA

EUROSIM'95

The Graphic User Interface of SIMPLE++

After starting SIMPLE++, a class library containing the basic objects will be displaied (unless otherwise specified) and, as an example, the following screen appears:

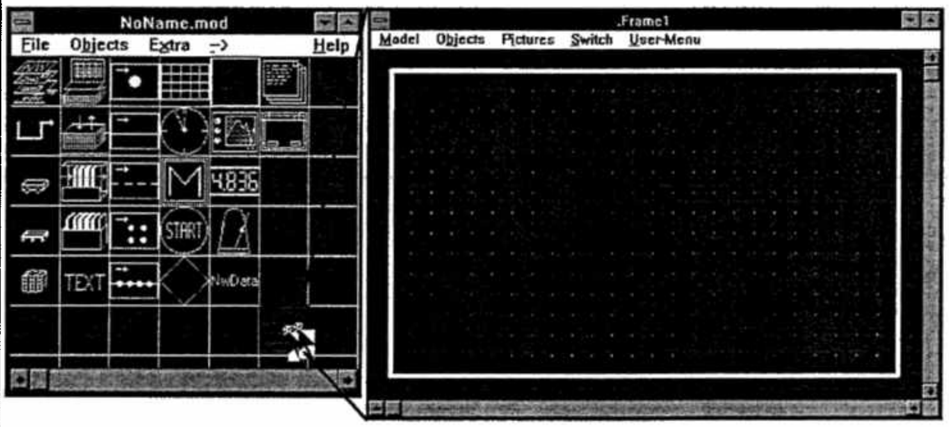

window of another application (e.g. EXCEL)
NoName.mod
File Objects Extra -> Help
TEXT
START
4.835
simple
File Environment Debugger Profiler
class library with the name of the last model

SIMPLE++
Windows NT
Version 3.0
(c) 1991-1995 by AESOP GmbH
(c) 1991 by Fraunhofer-IPA

EUROSIM'95

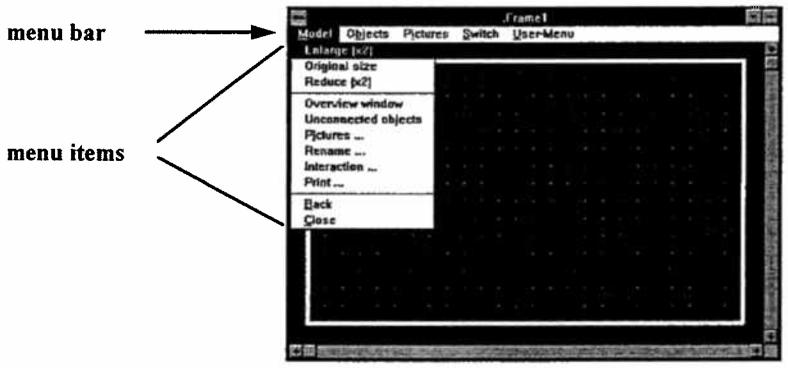
Opening Objects or Models

A model is opened by double clicking on the desired object in the library with the left mouse button.


NoName.mod
File Objects Extra -> Help
TEXT
START
4.836
NwData
.Frame1
Model Objects Pictures Switch User-Menu

EUROSIM'95

Using Menus

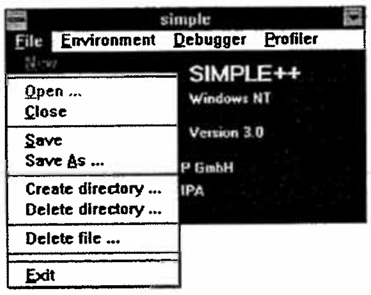


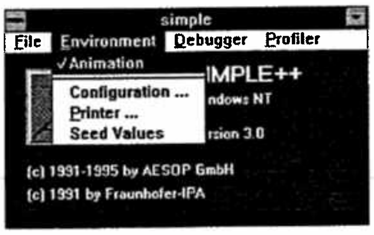
To activate a menu click on the desired menu title with the left mouse button. A list of menu items will appear. To select one of these items also click on it with the left mouse button.

Alternatively you could press the "Alt" - key and the underlined letter of the menu title simultaneously.This will also open the menu. One of these items can be activated by entering the underlined letter. Please note that these letters are casesensitive.

EUROSIM'95

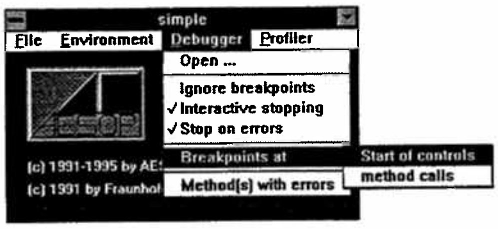
The File and the Environment Menus

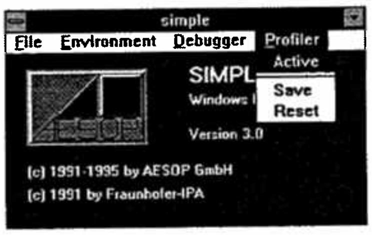



simple
File Environment Debugger Profiler
✓Animation
Configuration ...
Printer ...
Seed Values
IMPLE++
ndows NT
rsion 3.0
(c) 1991-1995 by AESOP GmbH
(c) 1991 by Fraunhofer-IPA

EUROSIM'95

The Debugger and Profiler Menus



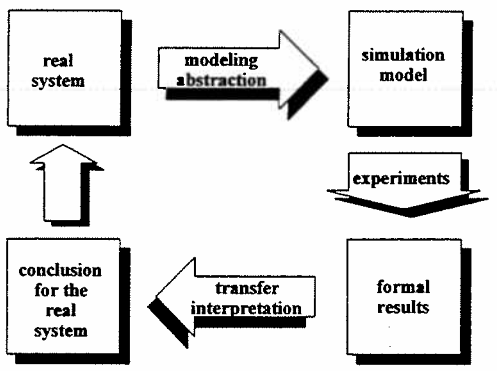

simple
File Environment Debugger Profiler
Active
Save
Reset
SIMPL
Windows
Version 3.0
(c) 1991-1995 by AESOP GmbH
(c) 1991 by Fraunhofer-IPA

EUROSIM'95

What is Simulation?

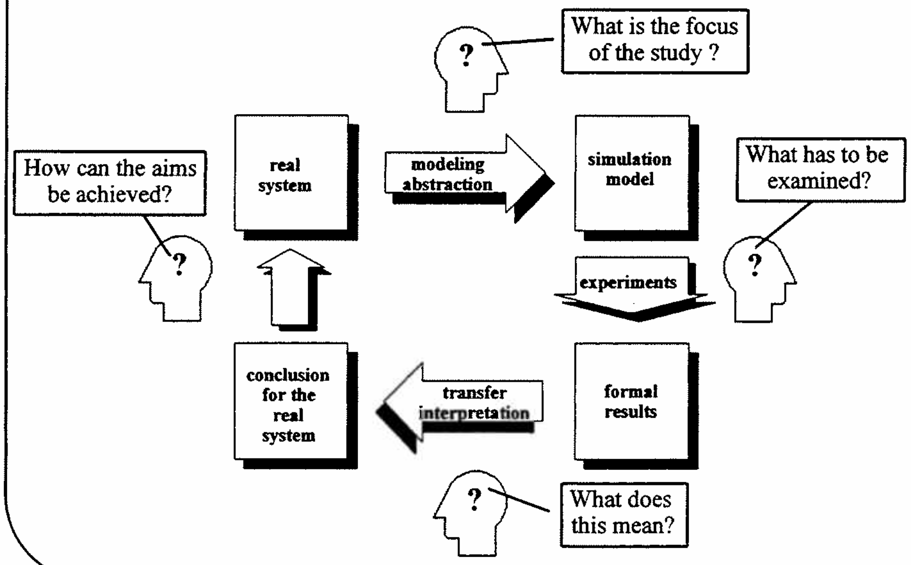
Simulation is defined by the German Institute of Engineers (Directive 3633) as:

Simulation is the depiction of a dynamic process in a model to gain knowledge that can be transferred to the real system.

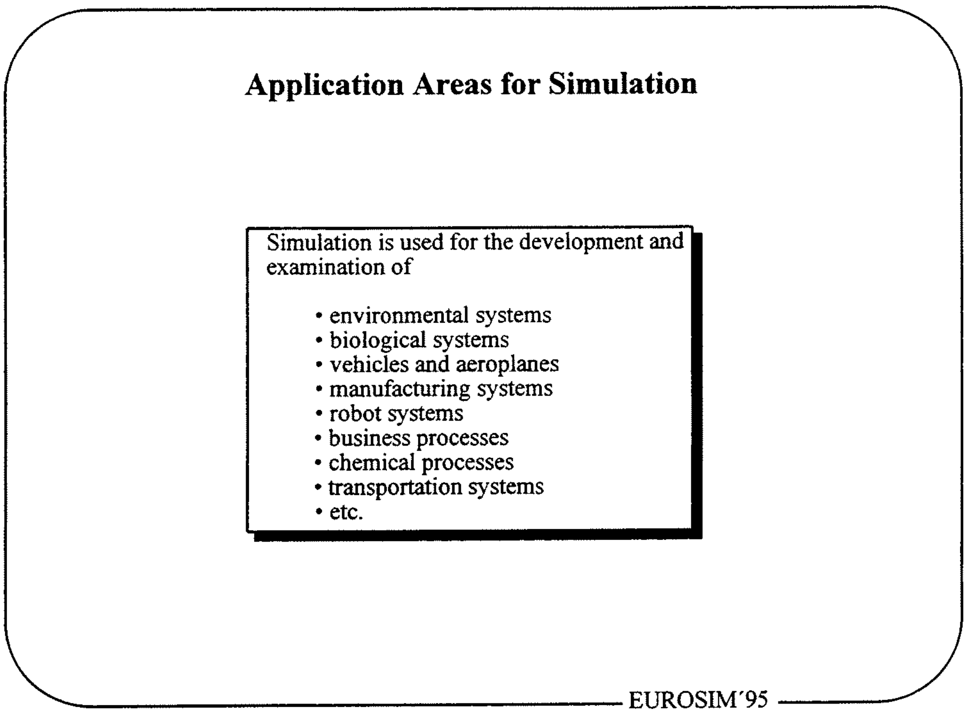


EUROSIM'95

Simulation and the User



EUROSIM'95

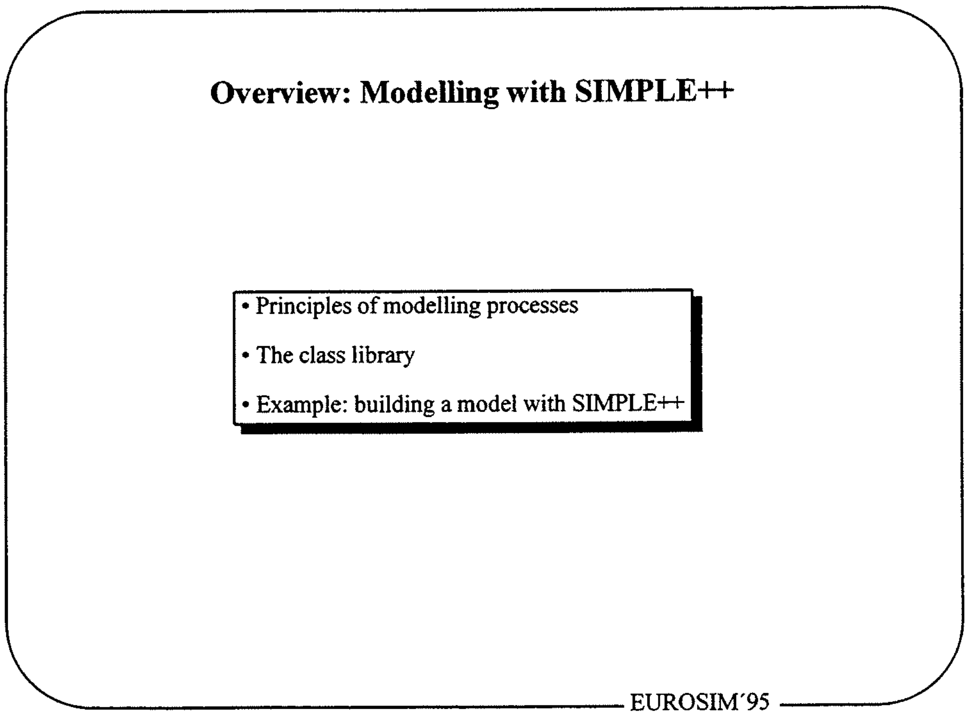


Application Areas for Simulation

Simulation is used for the development and examination of

- environmental systems
- biological systems
- vehicles and aeroplanes
- manufacturing systems
- robot systems
- business processes
- chemical processes
- transportation systems
- etc.

EUROSIM'95

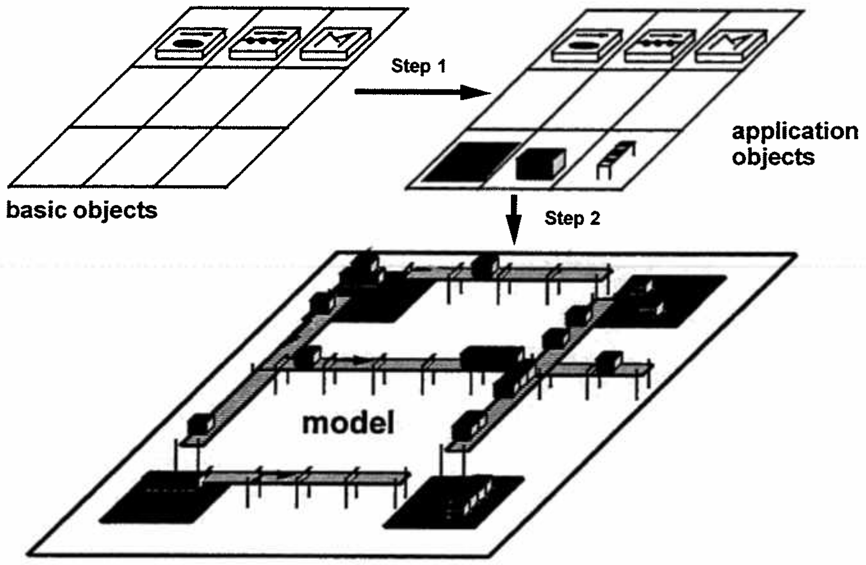


Overview: Modelling with SIMPLE++

- Principles of modelling processes
- The class library
- Example: building a model with SIMPLE++

EUROSIM'95

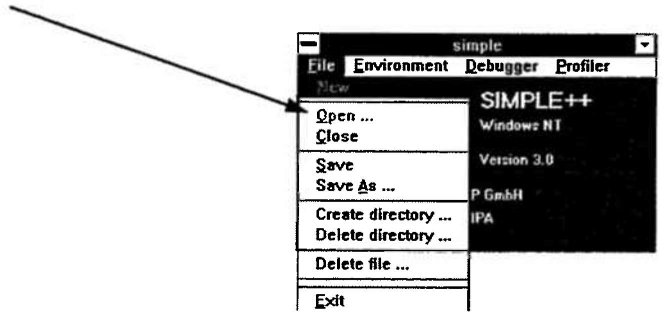
Modelling Steps



EUROSIM'95

Opening an Existing Model

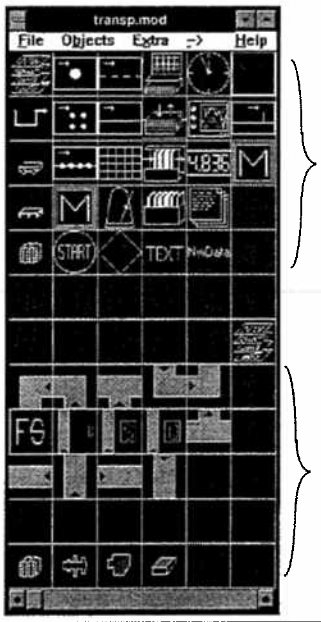
Clicking on the *Open...* menu item in the SIMPLE++ main window an existing model can be opened.
Note: if a class library is already opened, this must be closed first.



+ **Step1:**
Open the model named *Transp.mod* in your training directory.

EUROSIM'95

The Class Library

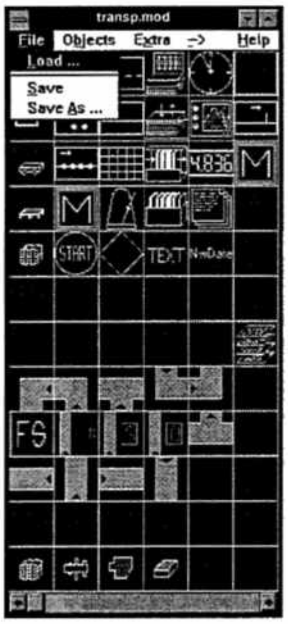


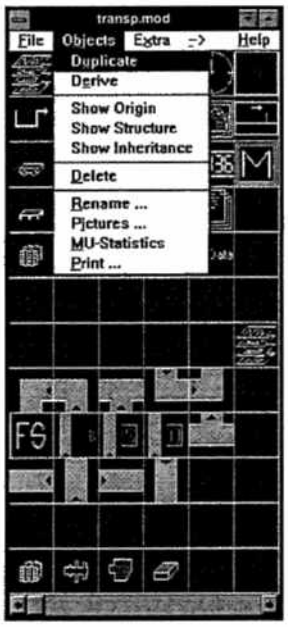
basic objects

specific application objects
built from basic objects

EUROSIM'95

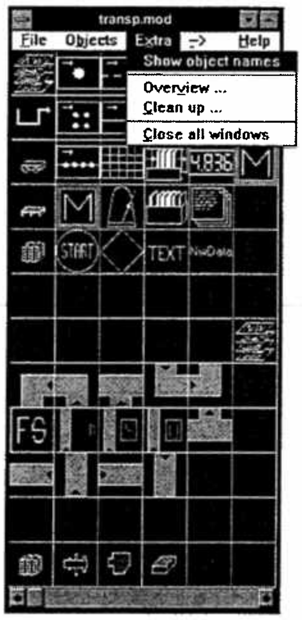
Menus of the Class Library (1)

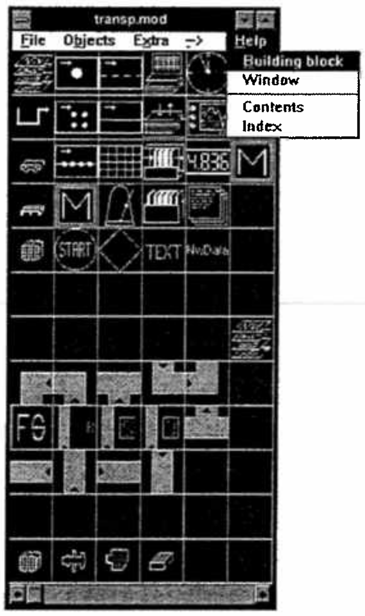




EUROSIM'95

Menus of the Class Library (2)

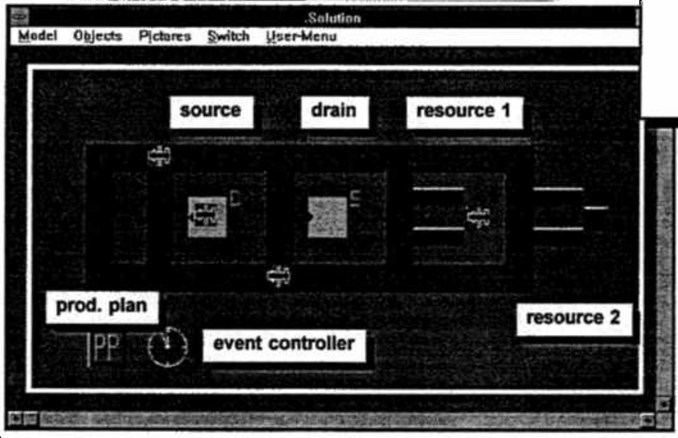


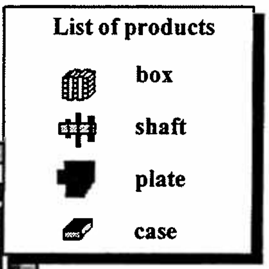


EUROSIM'95

Modelling with SIMPLE++

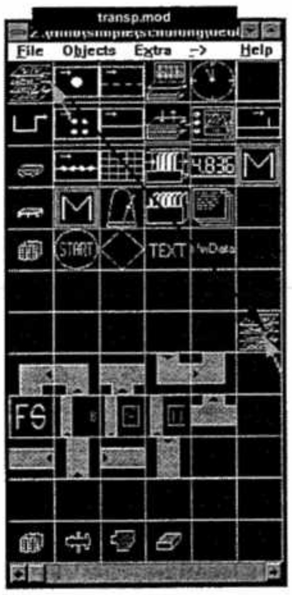
+ Creating a modell using pre-defined application objects





EUROSIM'95

Copying an Empty Network



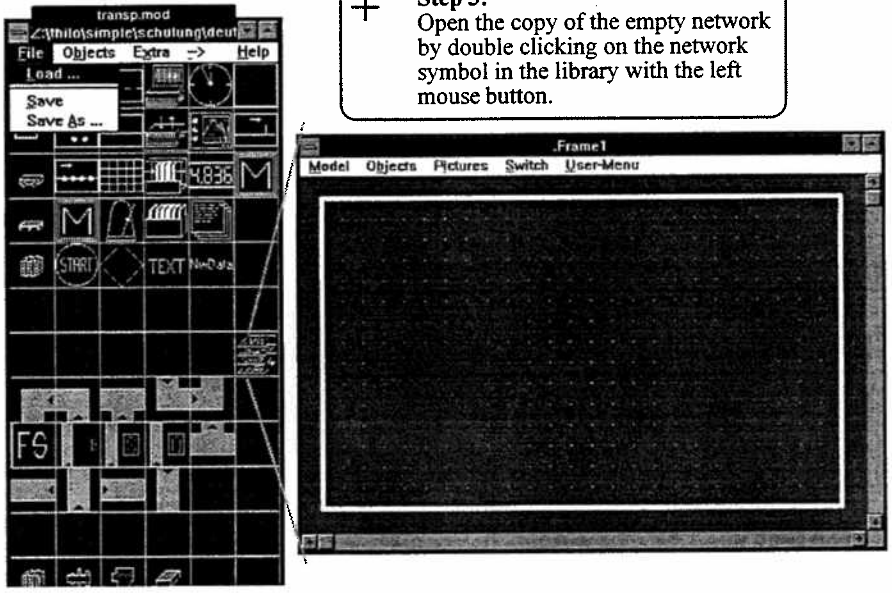
+ **Step2:**
Copy an empty network into an empty area of the library by clicking on the frame object then pressing the "shift" key and the left mouse button simultaneously and dragging the network to an empty field.

Remark:
The creation of an empty network is always the first step in creating a new model. The empty frame offered in the class library cannot be opened.

EUROSIM'95

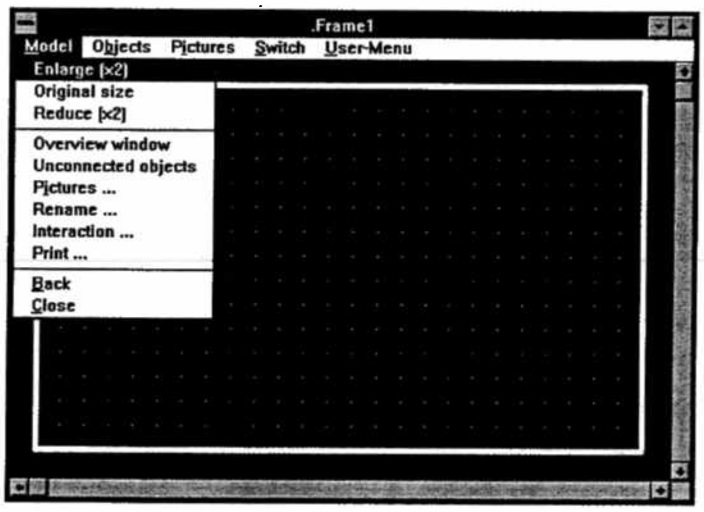
Opening the Network

+ **Step 3:**
Open the copy of the empty network by double clicking on the network symbol in the library with the left mouse button.



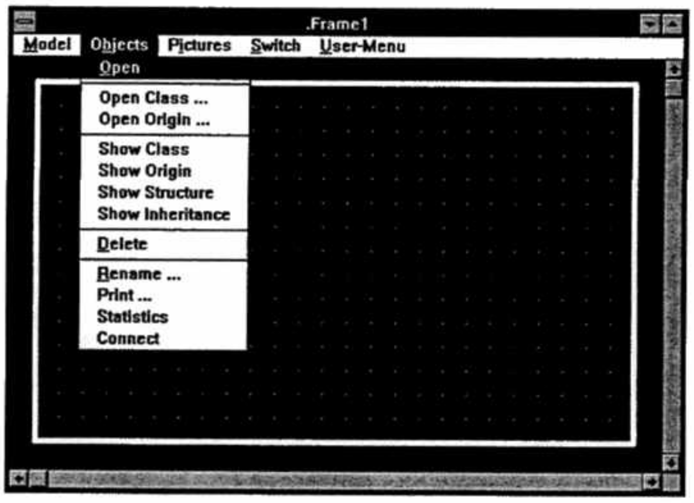
EUROSIM'95

The Frame Menu (1)



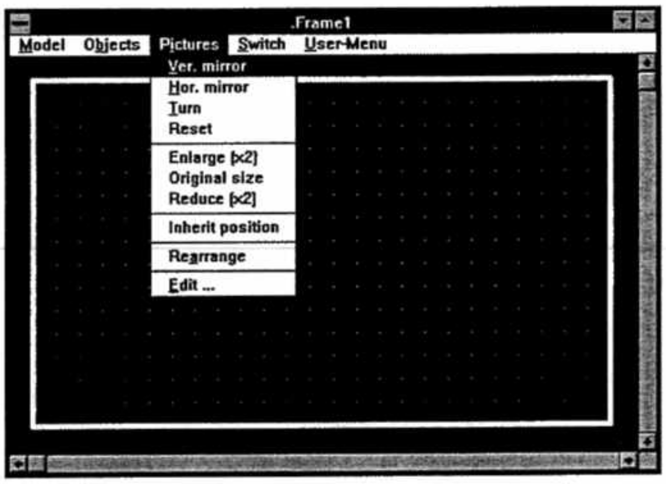
EUROSIM'95

The Frame Menu (2)



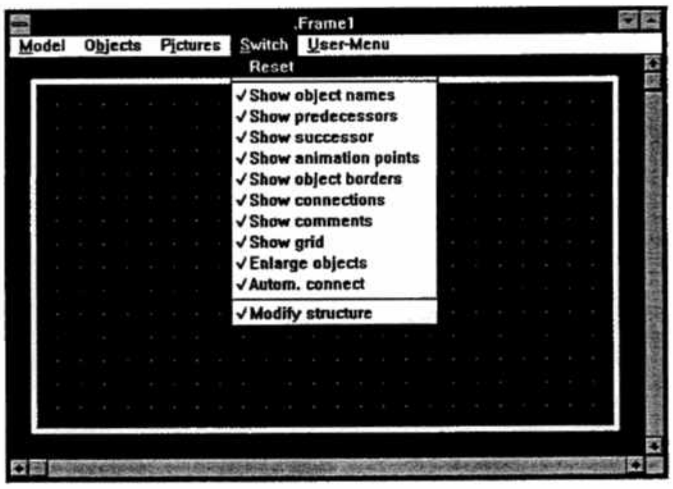
EUROSIM'95

The Frame Menu (3)



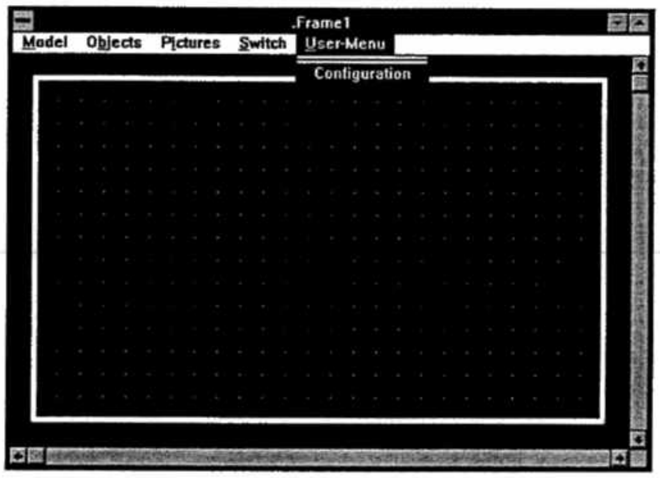
EUROSIM'95

The Frame Menu (4)



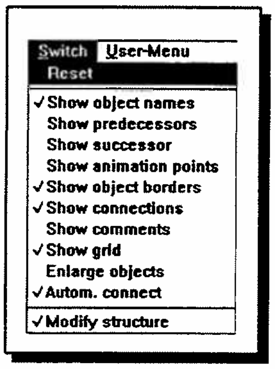
EUROSIM'95

The Frame Menu (5)



EUROSIM'95

Configuration


Switch User-Menu
Reset
✓Show object names
Show predecessors
Show successor
Show animation points
✓Show object borders
✓Show connections
Show comments
✓Show grid
Enlarge objects
✓Autom. connect
✓Modify structure

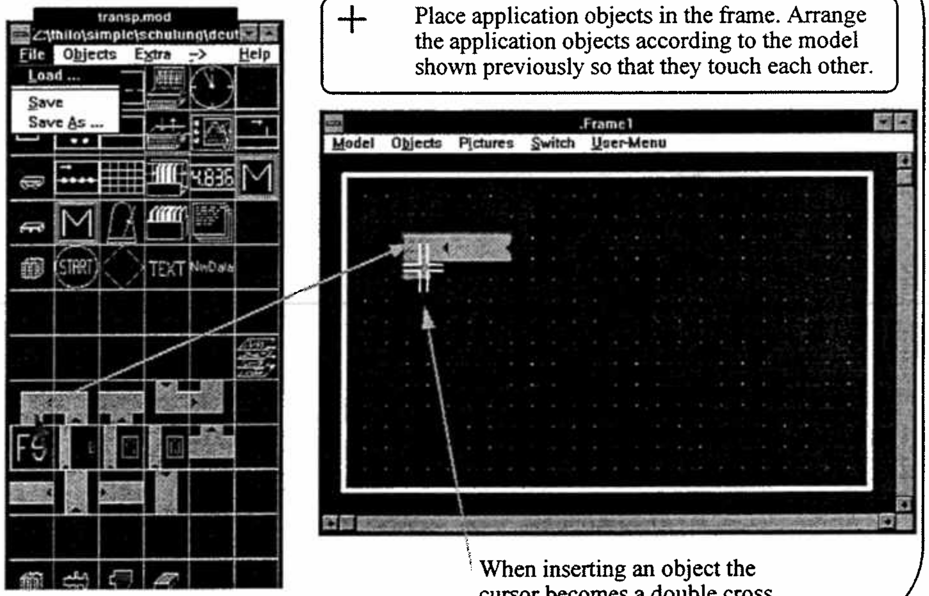
Step 4:
Before starting to model, toggle the options below the menu *Switch* as shown in the picture.

Note:
The automatic connection of building blocks is possible if there is an obvious connection between those building blocks.

EUROSIM'95

Inserting Objects into the Frame

+ Place application objects in the frame. Arrange the application objects according to the model shown previously so that they touch each other.


transp.mod
File Objects Extra -> Help
Load ...
Save
Save As ...
.Frame1
Model Objects Pictures Switch User-Menu

When inserting an object the cursor becomes a double cross.

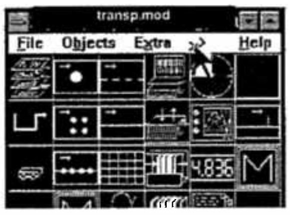
EUROSIM'95

To leave the INSERT Mode

As long as an object in the library is, an instance of this object will be placed in the frame each time you click the left mouse button within the model window.

You can deselect INSERT mode by:

- Clicking the right mouse button in the frame
- Clicking on an empty field in the class library (left mouse button)
- Clicking on the arrow in the menu bar of the class library (left mouse button)



EUROSIM'95

Connecting Objects

1. possibility

graphical, interactive with the connector tool

2. possibility

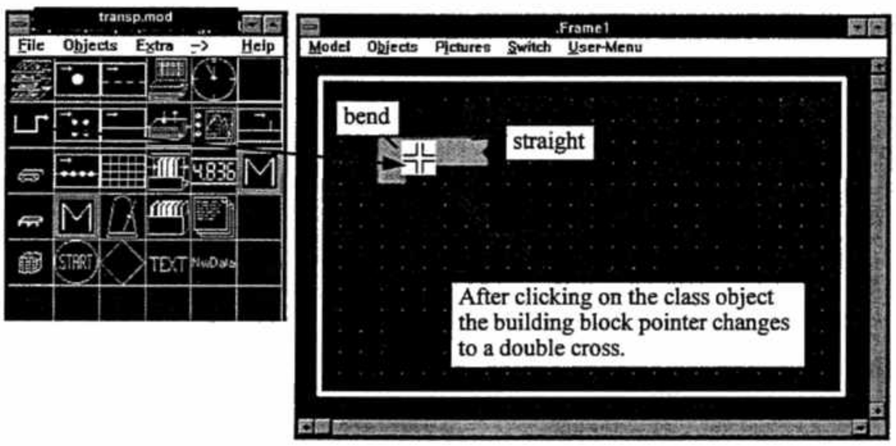
automatically while inserting building blocks
(if the Switch - Autom. connect is on)

3. possibility

automatically after inserting the building blocks
frame-menu: Objects - connect

EUROSIM'95

Interactive Connection of Objects(1)


transp.mod
File Objects Extra -> Help
4.836
START
TEXT
Frame1
Model Objects Pictures Switch User-Menu
bend
straight

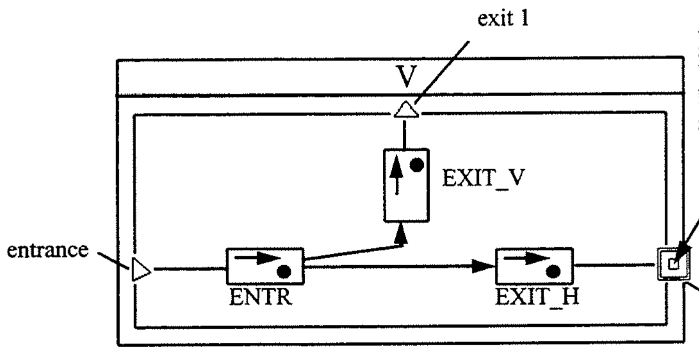
After clicking on the class object the building block pointer changes to a double cross.

Application objects are connected manually by first selecting the *Connector* tool in the class library, then clicking on the starting building block and then on the goal building block of the desired connection.

EUROSIM'95

Interactive Connection of Objects(2)

After selecting the starting object in the model, the object is opened and, in the example below, the following picture appears, displaying the entrances and exits and the subelements of the building block. This picture only appears if the building block has more than one unconnected exit. If there is only one unconnected exit the connection is unambigous and will be automatically closed.

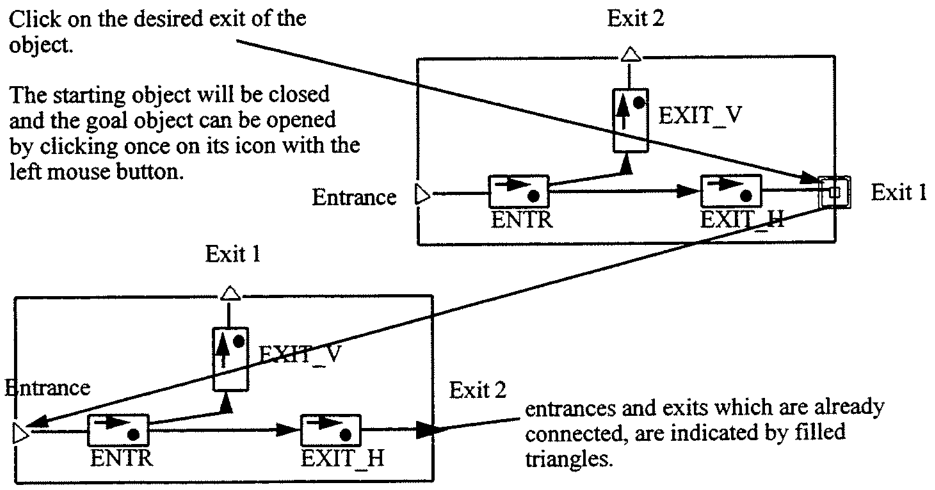


After opening the starting building block the cursor becomes a square symbol.

Interactive Connection of Objects(3)

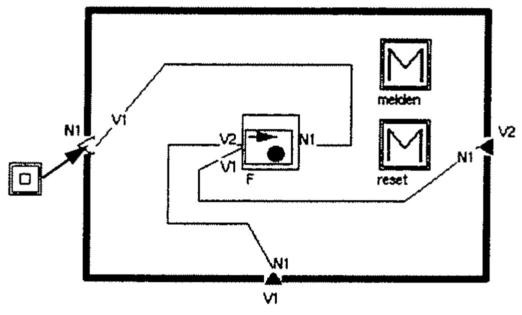
Click on the desired exit of the object.

The starting object will be closed and the goal object can be opened by clicking once on its icon with the left mouse button.



entrances and exits which are already connected, are indicated by filled triangles.

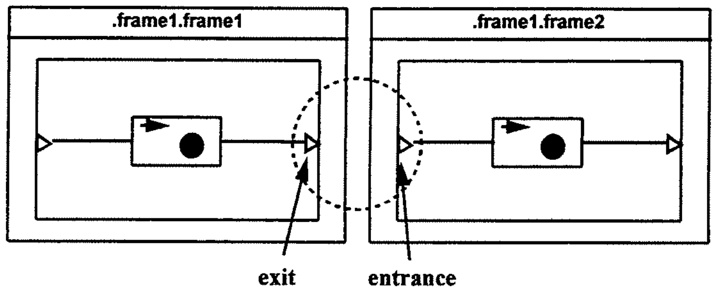
Interactive Connection of Objects (4)



The connection is established by clicking on an entrance of the goal building block. The goal building block will be closed automatically. This process may be repeated until all the entrances and exits of the model are connected.

EUROSIM'95

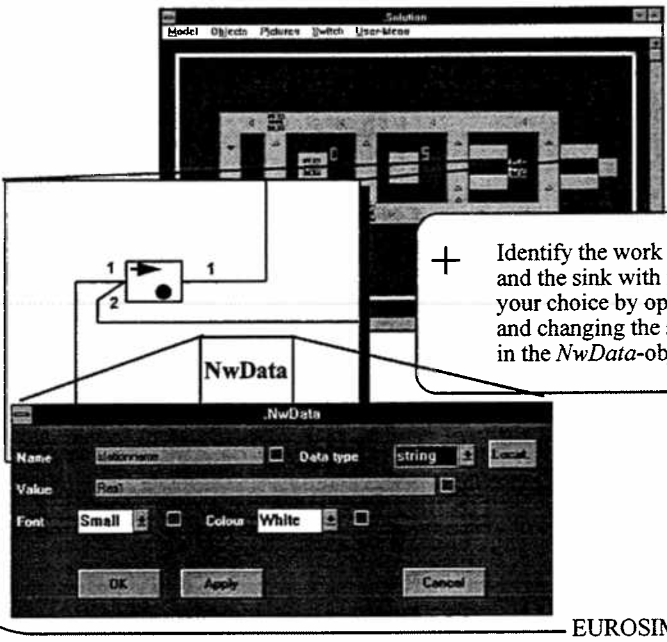
Automatic Connection of Objects



The automatic connection of building blocks is done, when the entrance and the exit of the building blocks being inserted are close together. The circular catch window has a 3 pixel radius.

EUROSIM'95

Changing the Names of the Stations



Name [illegible] Data type string

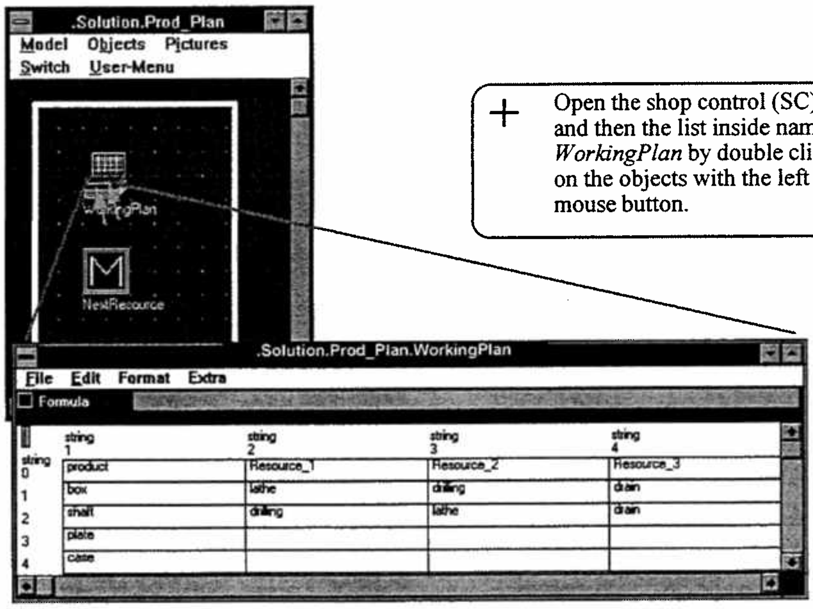
Value Real

Font Small Colour White

+ Identify the work stations and the sink with names of your choice by opening them and changing the station name in the *NwData*-object.

EUROSIM'95

Building the Working Plan (1)



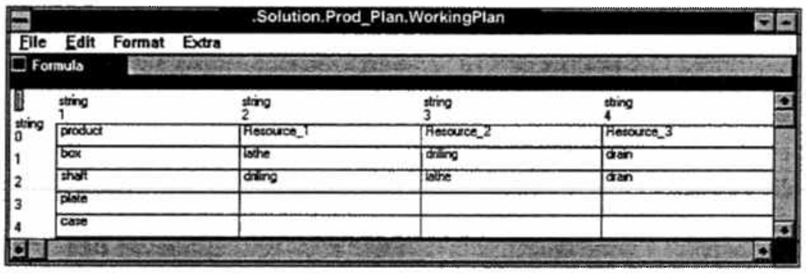
	string 1	string 2	string 3	string 4
string 0	product	Resource_1	Resource_2	Resource_3
1	box	lathe	drilling	drain
2	shaft	drilling	lathe	drain
3	plate			
4	case			

+ Open the shop control (SC) and then the list inside named *WorkingPlan* by double clicking on the objects with the left mouse button.

EUROSIM'95

Building the Working Plan (2)

+ Build the working plan using the machine names you have chosen.



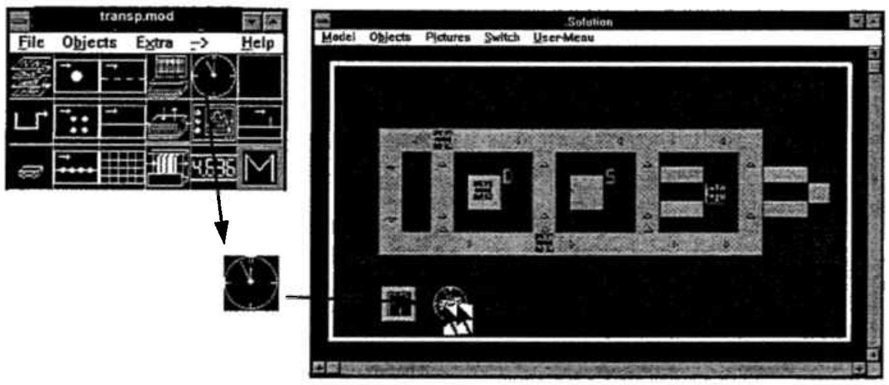
string	string 1	string 2	string 3	string 4
0	product	Resource_1	Resource_2	Resource_3
1	box	lathe	drilling	drain
2	shaft	drilling	lathe	drain
3	plate			
4	case			

$\longrightarrow$
manufacturing sequence

Be sure that the last work station is a drain. Otherwise the parts will not be deleted!

EUROSIM'95

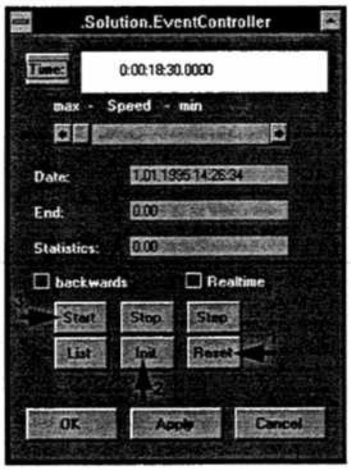
Placing the *EventController* in the Model



+ Place an *EventController* in the model and open it by double clicking on it with the left mouse button.

EUROSIM'95

Starting the Simulation with the *EventController*



.Solution.EventController

Time: 0:00:18:30.0000

Date: 1.01.1995 14:26:34

End: 0.00

Statistics: 0.00

Step 6:

1. Click on the *Reset* button to bring the model into a defined state.
2. Click on the *Init* button to initialize the model.
3. Start the simulation by clicking the *Start* button

EUROSIM'95

Inheritance (1)

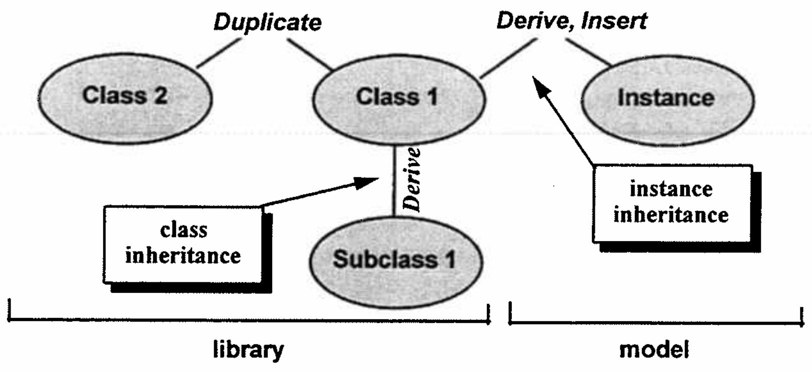
The difference between *class* and *instance*

class:	each model/object in the class library. A class passes all its characteristics to its instance.
instance:	specimen (child) of the class (parents), e.g. the inserted object *Bend* is the instance of the object *Bend* in the class library. Building block *Bend* in library: *class* Inserted building block *Bend*: *instance*
Important:	if changes of properties are to effect all specimens they must be changed for the *class*.

EUROSIM'95

Inheritance (2)

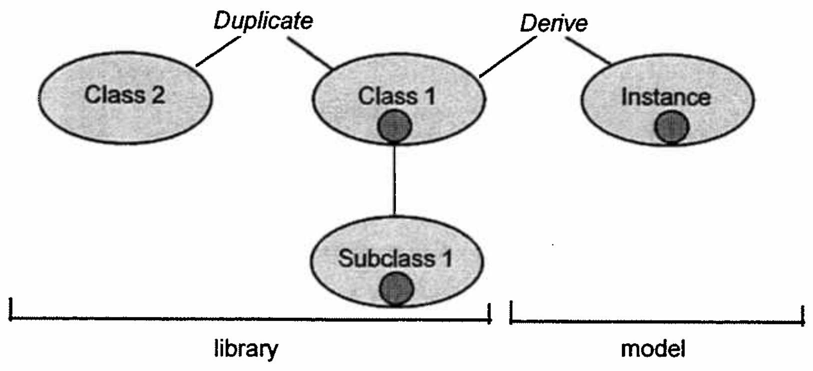
Difference between *Duplicate(=Copy)* and *Derive*



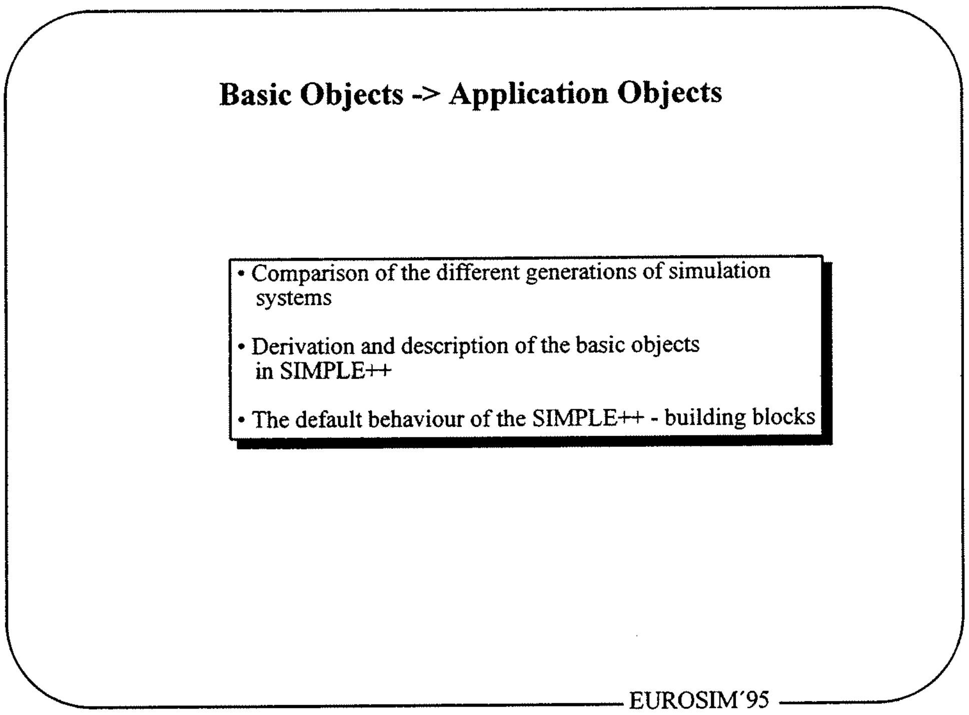
EUROSIM'95

Inheritance (3)

Situation after the change of a property



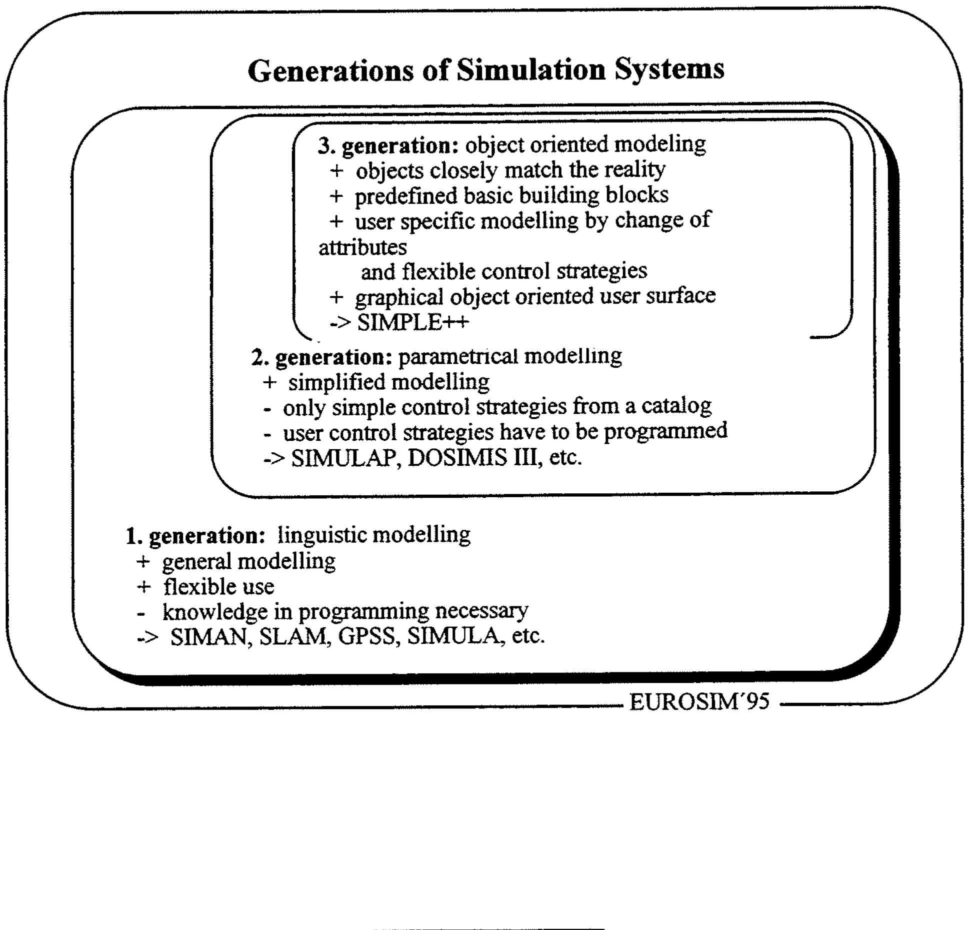
EUROSIM'95



Basic Objects -> Application Objects

- Comparison of the different generations of simulation systems
- Derivation and description of the basic objects in SIMPLE++
- The default behaviour of the SIMPLE++ - building blocks

EUROSIM'95



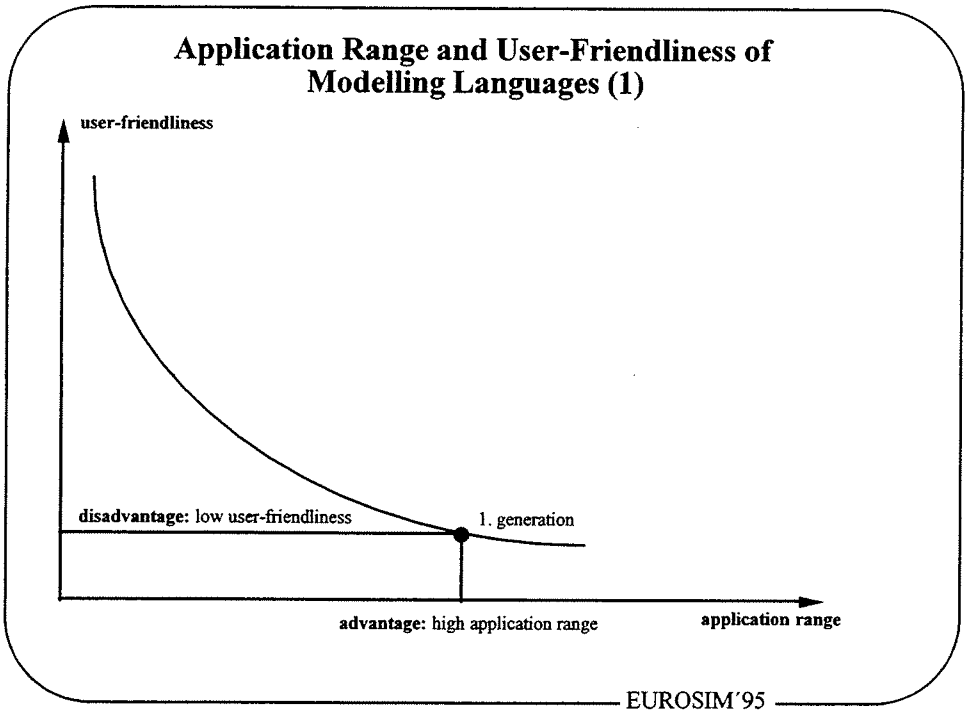
Generations of Simulation Systems

3. generation: object oriented modeling
+ objects closely match the reality
+ predefined basic building blocks
+ user specific modelling by change of attributes
and flexible control strategies
+ graphical object oriented user surface
-> SIMPLE++

2. generation: parametrical modelling
+ simplified modelling
- only simple control strategies from a catalog
- user control strategies have to be programmed
-> SIMULAP, DOSIMIS III, etc.

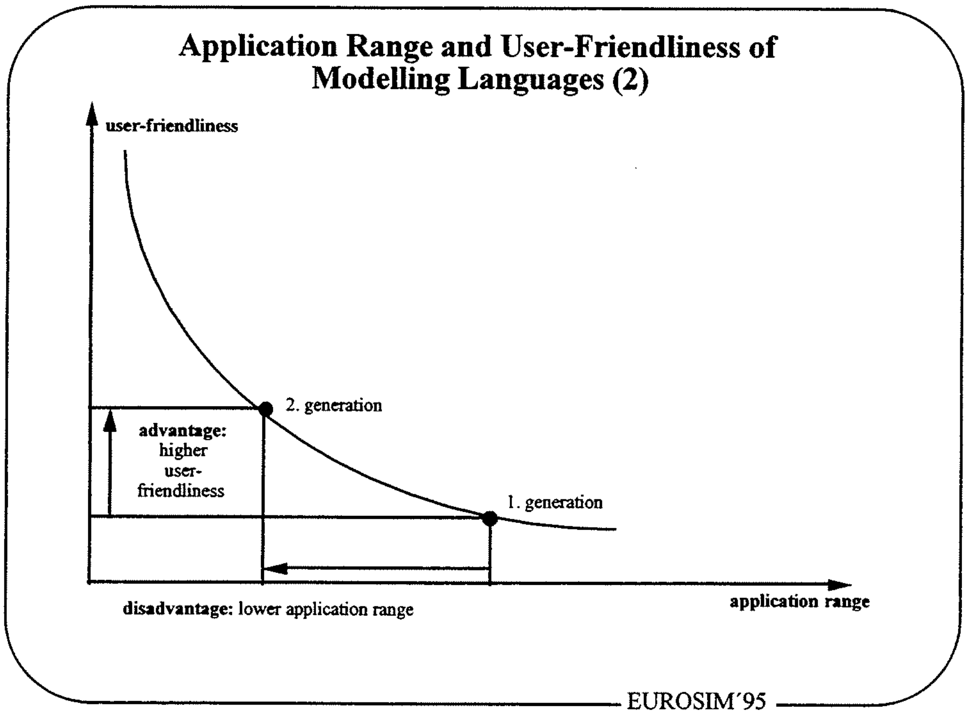
1. generation: linguistic modelling
+ general modelling
+ flexible use
- knowledge in programming necessary
-> SIMAN, SLAM, GPSS, SIMULA, etc.

EUROSIM'95



Application Range and User-Friendliness of Modelling Languages (1)

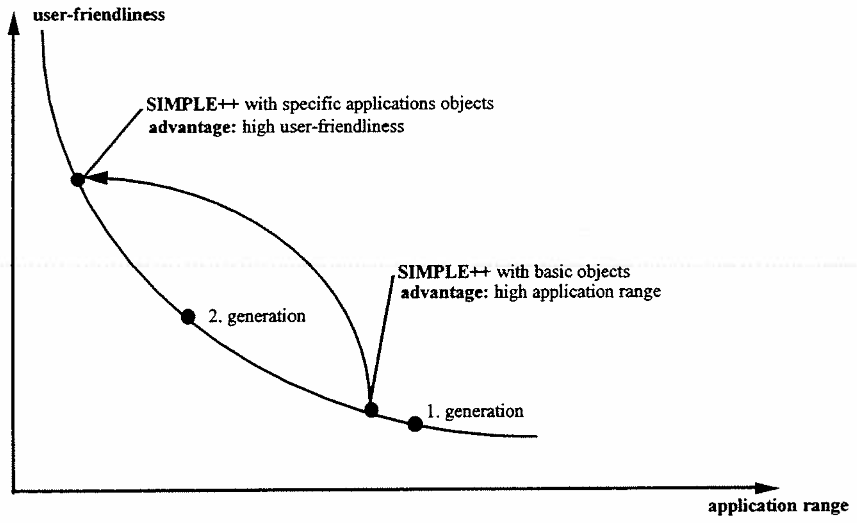
EUROSIM'95



Application Range and User-Friendliness of Modelling Languages (2)

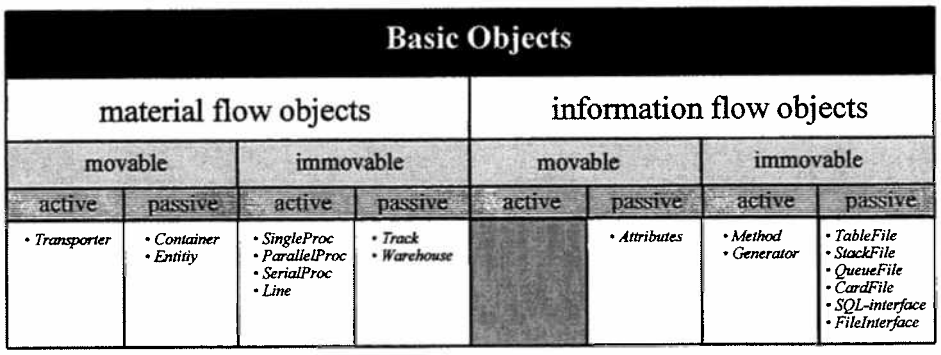
EUROSIM'95

Application Range and User-Friendliness of Modelling Languages (3)



EUROSIM'95

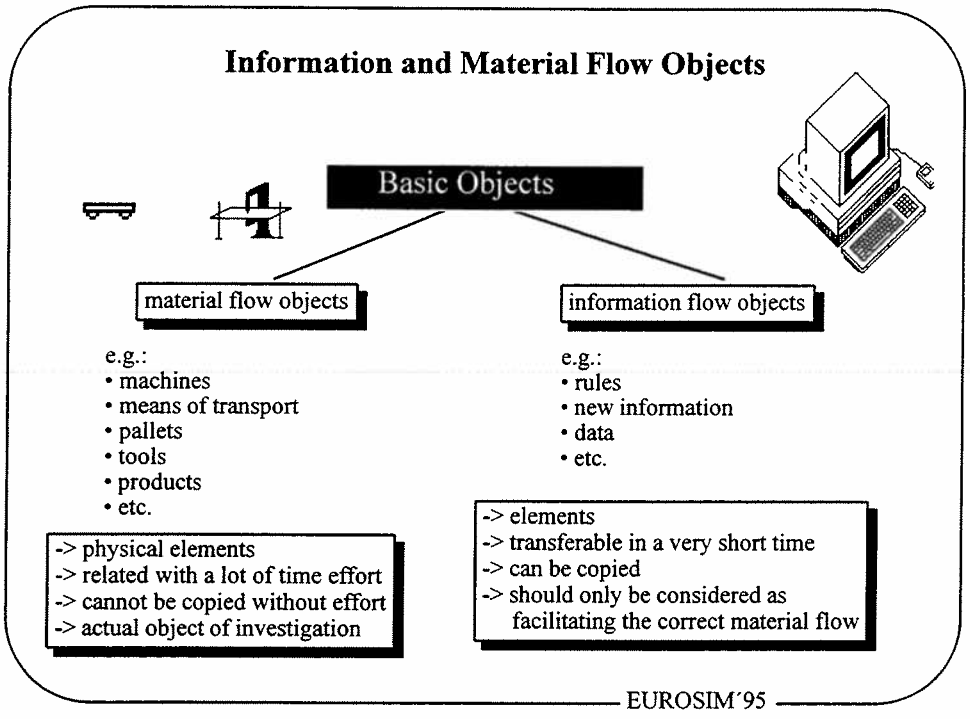
Structure of the Basic Objects



Basic Objects							
material flow objects				information flow objects			
movable		immovable		movable		immovable	
active	passive	active	passive	active	passive	active	passive
• *Transporter*	• *Container* • *Entitiy*	• *SingleProc* • *ParallelProc* • *SerialProc* • *Line*	• *Track* • *Warehouse*		• *Attributes*	• *Method* • *Generator*	• *TableFile* • *StackFile* • *QueueFile* • *CardFile* • *SQL-interface* • *FileInterface*

Additional Service Objects: *Plotter, Gauge, Dialog, Text, Connector, EventController*

EUROSIM'95



Information and Material Flow Objects

Basic Objects

material flow objects

e.g.:
• machines
• means of transport
• pallets
• tools
• products
• etc.

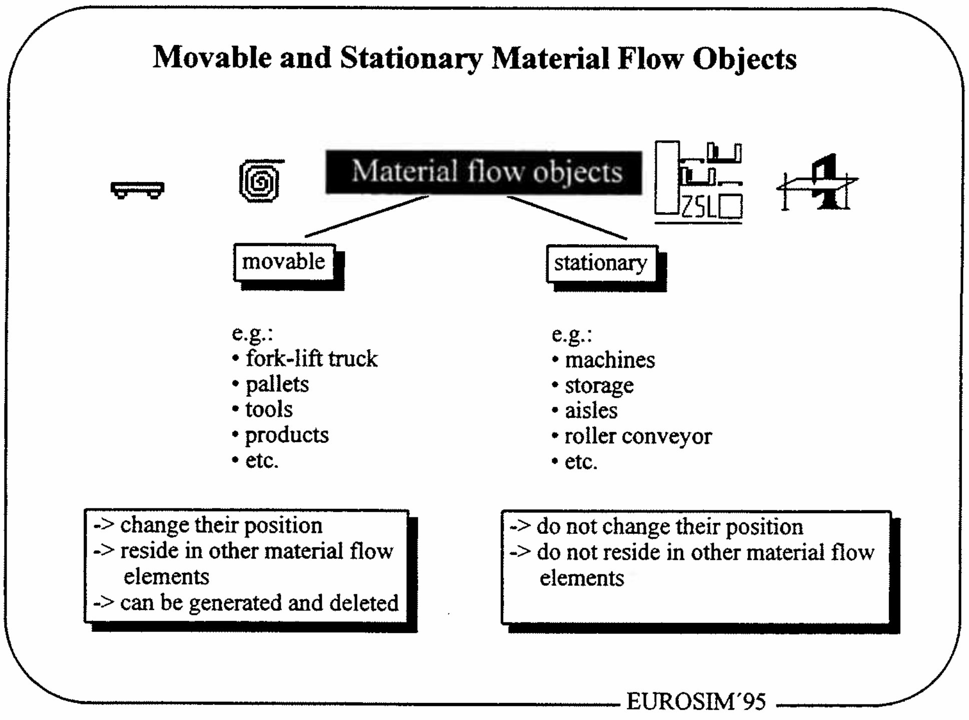
-> physical elements
-> related with a lot of time effort
-> cannot be copied without effort
-> actual object of investigation

information flow objects

e.g.:
• rules
• new information
• data
• etc.

-> elements
-> transferable in a very short time
-> can be copied
-> should only be considered as facilitating the correct material flow

EUROSIM'95



Movable and Stationary Material Flow Objects

Material flow objects

movable

e.g.:
• fork-lift truck
• pallets
• tools
• products
• etc.

-> change their position
-> reside in other material flow elements
-> can be generated and deleted

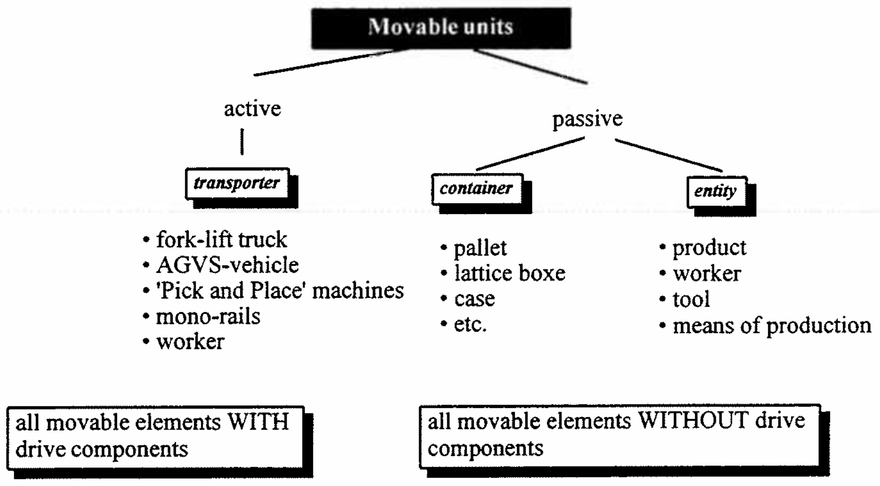
stationary

e.g.:
• machines
• storage
• aisles
• roller conveyor
• etc.

-> do not change their position
-> do not reside in other material flow elements

EUROSIM'95

Movable Units (MU)



- fork-lift truck
- AGVS-vehicle
- 'Pick and Place' machines
- mono-rails
- worker

- pallet
- lattice boxe
- case
- etc.

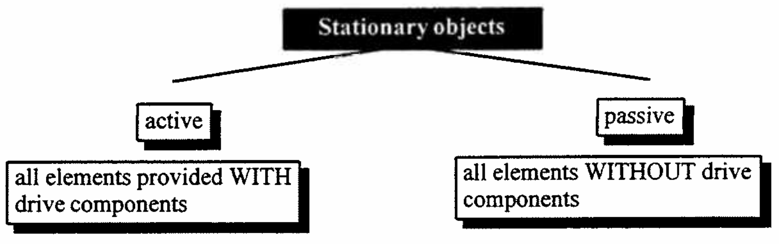
- product
- worker
- tool
- means of production

all movable elements WITH drive components

all movable elements WITHOUT drive components

EUROSIM'95

Stationary Material Flow Objects



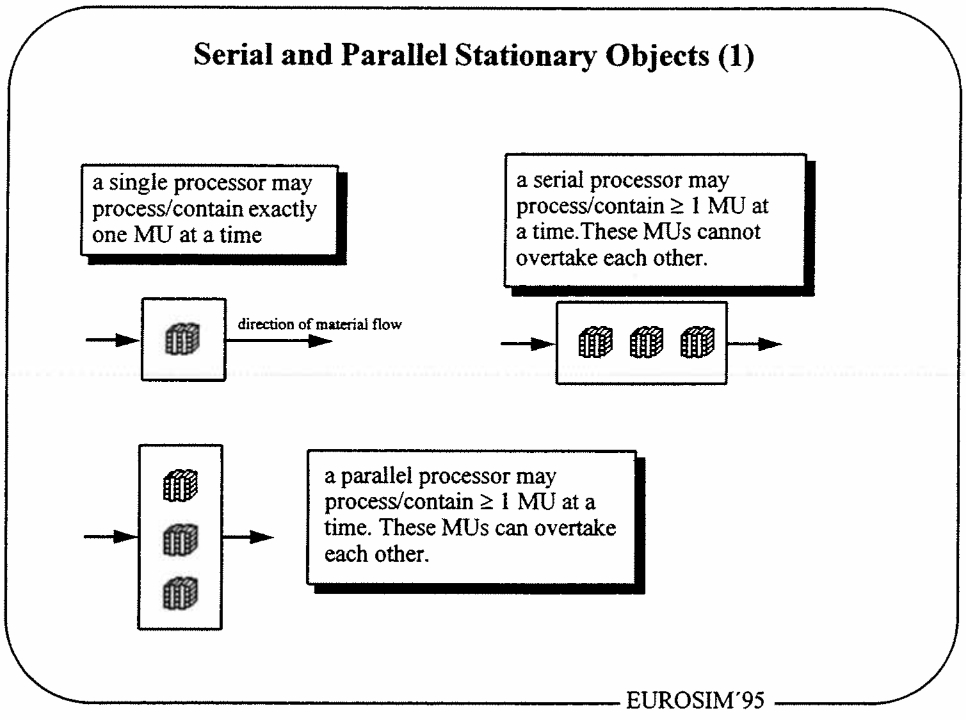
all elements provided WITH drive components

- roller conveyor
- belt conveyor
- lathe
- assembly work station
- etc.

all elements WITHOUT drive components

- aisles
- storage rack
- halls
- etc.

EUROSIM'95



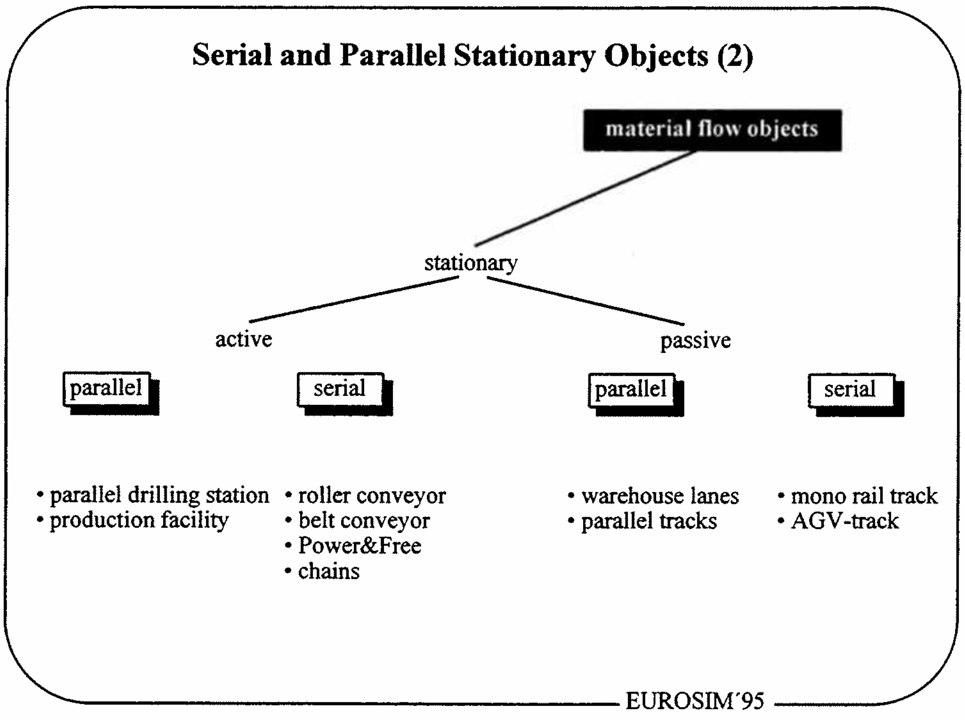
Serial and Parallel Stationary Objects (1)

a single processor may process/contain exactly one MU at a time

a serial processor may process/contain ≥ 1 MU at a time. These MUs cannot overtake each other.

a parallel processor may process/contain ≥ 1 MU at a time. These MUs can overtake each other.

EUROSIM'95



Serial and Parallel Stationary Objects (2)

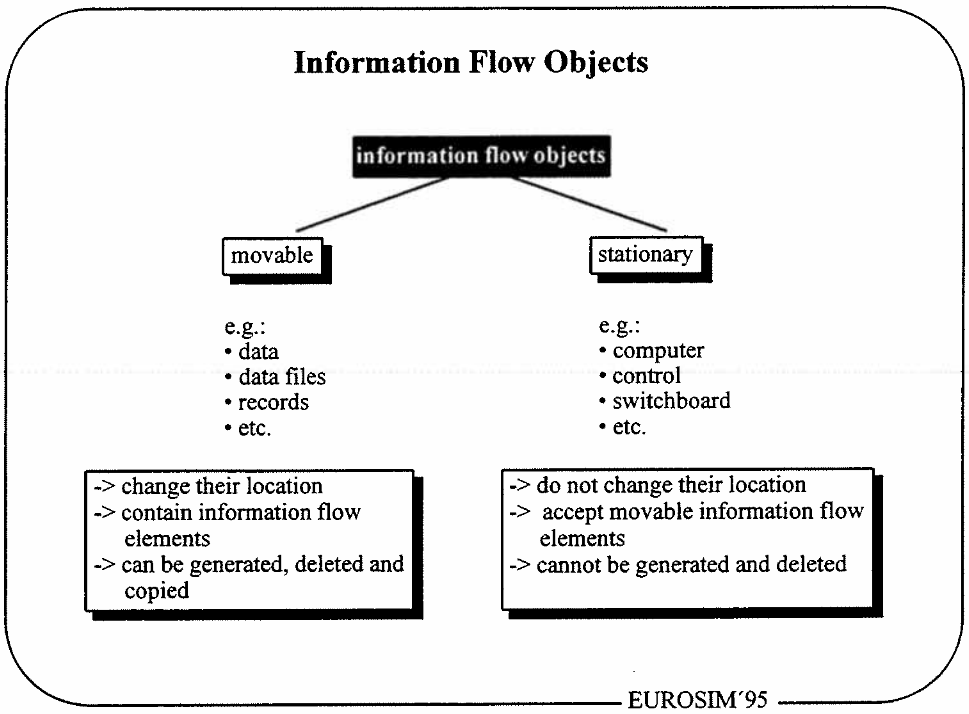
- parallel drilling station
- production facility

- roller conveyor
- belt conveyor
- Power&Free
- chains

- warehouse lanes
- parallel tracks

- mono rail track
- AGV-track

EUROSIM'95



Information Flow Objects

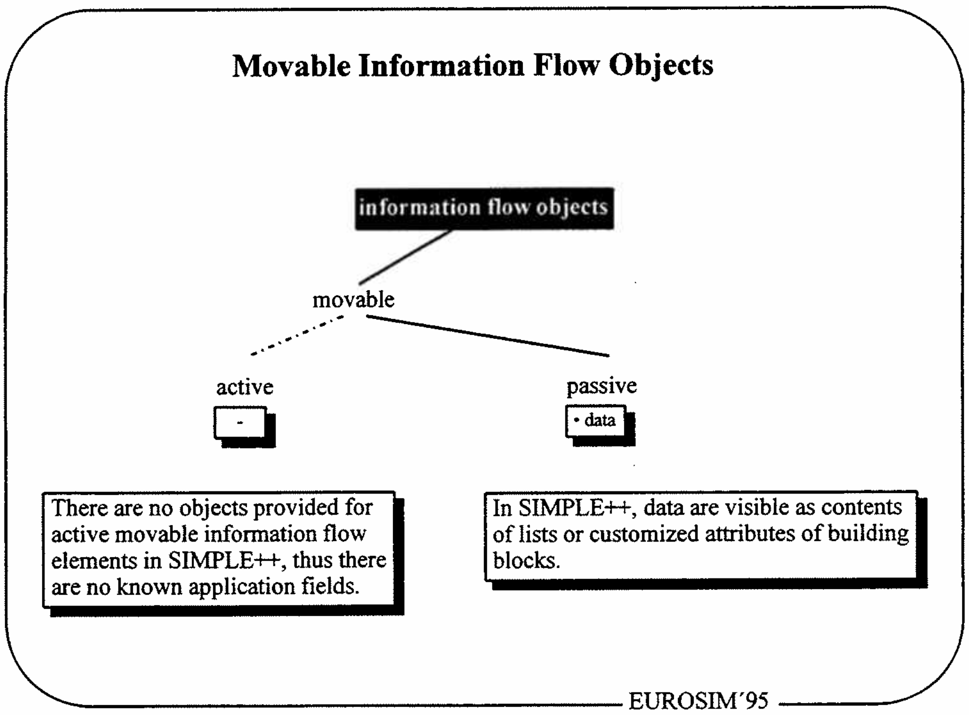
e.g.:
• data
• data files
• records
• etc.

e.g.:
• computer
• control
• switchboard
• etc.

-> change their location
-> contain information flow elements
-> can be generated, deleted and copied

-> do not change their location
-> accept movable information flow elements
-> cannot be generated and deleted

EUROSIM'95



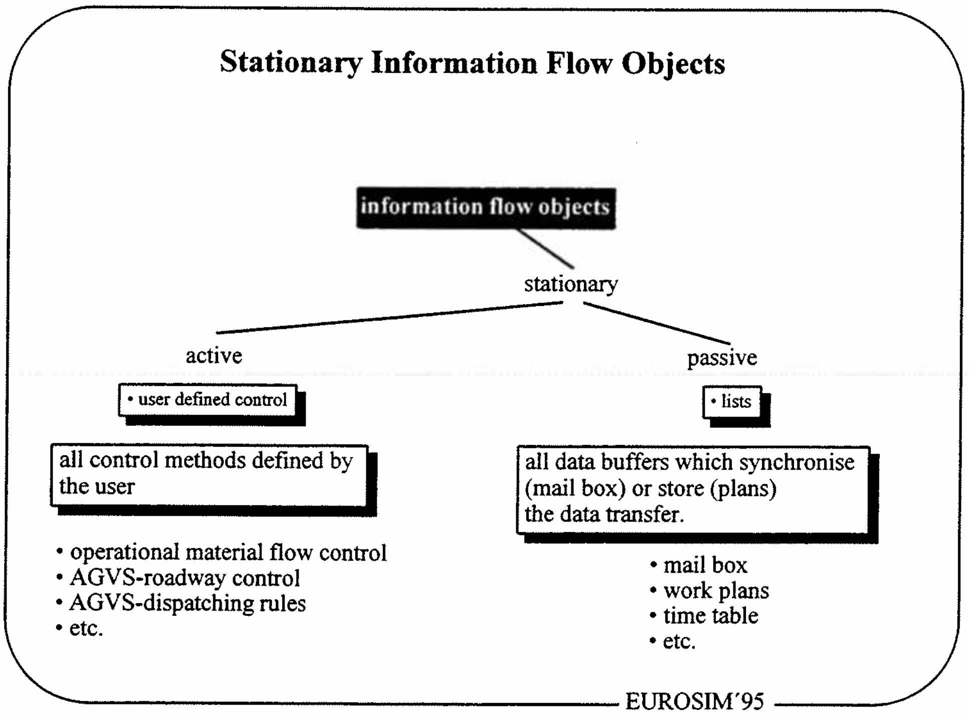
Movable Information Flow Objects

There are no objects provided for active movable information flow elements in SIMPLE++, thus there are no known application fields.

In SIMPLE++, data are visible as contents of lists or customized attributes of building blocks.

EUROSIM'95

Stationary Information Flow Objects



all control methods defined by the user

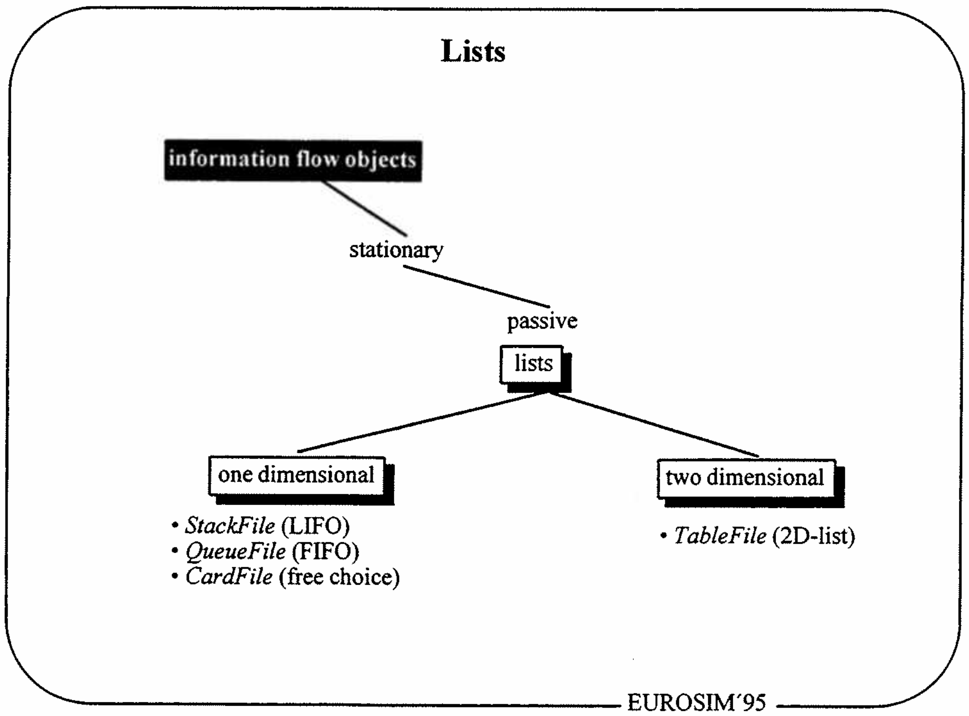
- operational material flow control
- AGVS-roadway control
- AGVS-dispatching rules
- etc.

all data buffers which synchronise (mail box) or store (plans) the data transfer.

- mail box
- work plans
- time table
- etc.

EUROSIM'95

Lists



- *StackFile* (LIFO)
- *QueueFile* (FIFO)
- *CardFile* (free choice)

- *TableFile* (2D-list)

EUROSIM'95

Names of the Basic Objects (1)

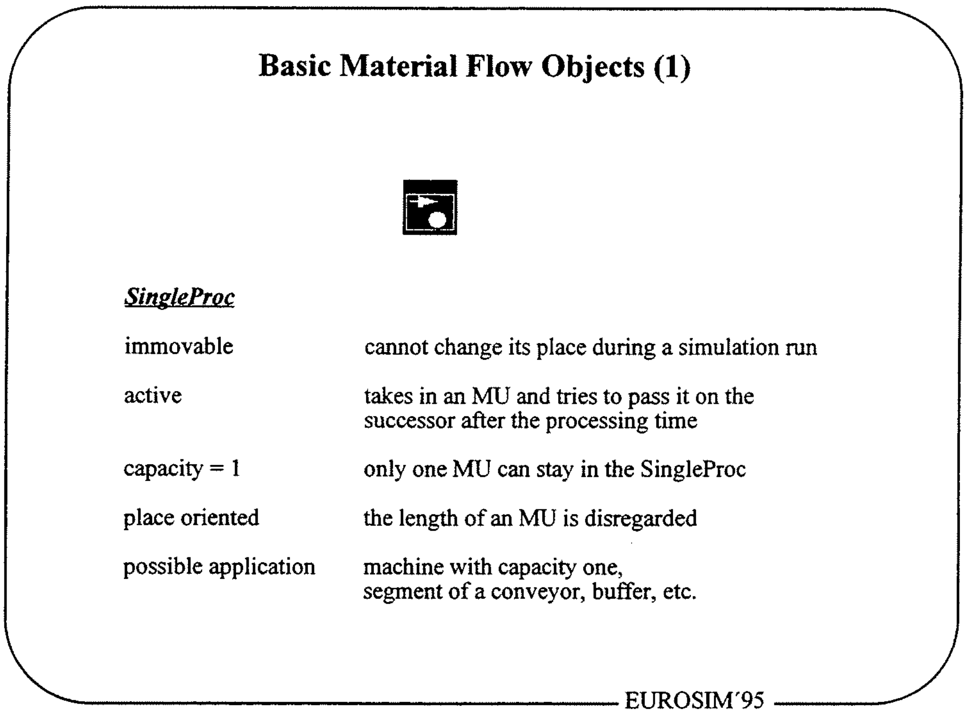
Frame	*Warehouse*
SingleProc	*Entity*
ParallelProc	*Container*
SerialProc	*Transporter*
Line	*Connector*
Track	*EventController*

EUROSIM'95

Names of the Basic Objects (2)

StackFile (LIFO)	*Generator*
QueueFile (FIFO)	*Method*
CardFile	*Comment*
TableFile	*Display*
NwData	*Plotter*
FileInterface	*Dialog*

EUROSIM'95

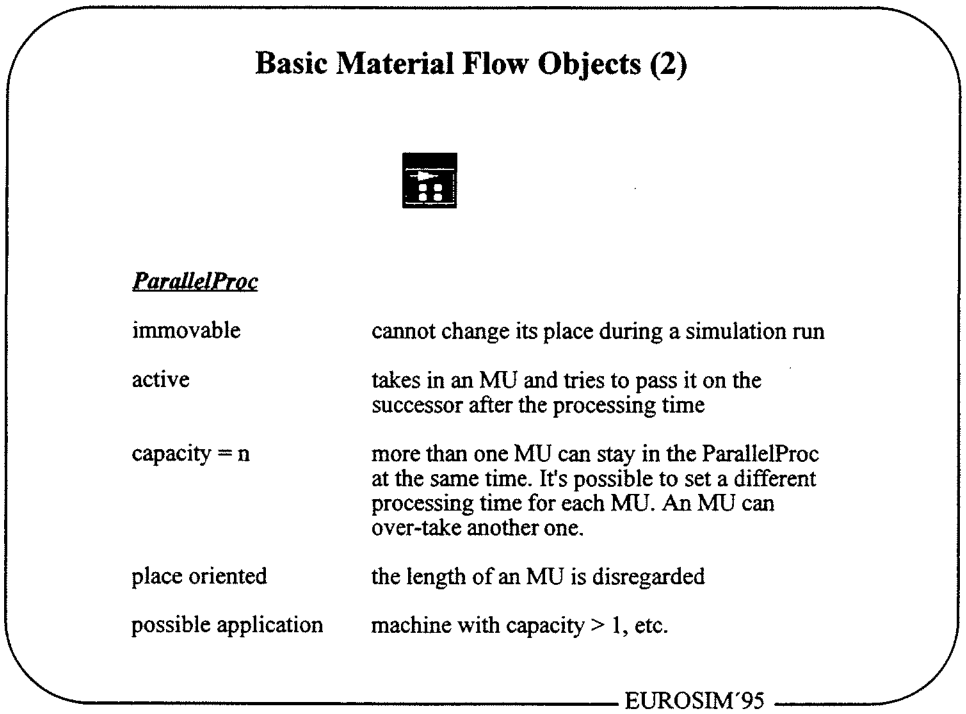


Basic Material Flow Objects (1)

SinglePróc

immovable	cannot change its place during a simulation run
active	takes in an MU and tries to pass it on the successor after the processing time
capacity = 1	only one MU can stay in the SingleProc
place oriented	the length of an MU is disregarded
possible application	machine with capacity one, segment of a conveyor, buffer, etc.

EUROSIM'95



Basic Material Flow Objects (2)

ParallelProc

immovable	cannot change its place during a simulation run
active	takes in an MU and tries to pass it on the successor after the processing time
capacity = n	more than one MU can stay in the ParallelProc at the same time. It's possible to set a different processing time for each MU. An MU can over-take another one.
place oriented	the length of an MU is disregarded
possible application	machine with capacity > 1, etc.

EUROSIM'95

Basic Material Flow Objects (3)

SerialProc

immovable active	cannot change its place during a simulation run
capacity = n	the number of parallel lines can be defined. An MU cannot over-take another one in the same line.
place oriented	the length of an MU is disregarded
jamable/un-	user defined setting
possible application	conveyor, buffer, etc.

EUROSIM'95

Basic Material Flow Objects (4)

Line

immovable active	cannot change its place during a simulation run
capacity	the length of the line can be freely defined. An MU cannot over-take another one.
length oriented	the length of an MU is regarded, i.e. the max. number of MUs on a line depends on the length of the line and the length of the MUs.
accumulating	user defined setting
possible application	conveyor bend with segments of different lengths, etc.

EUROSIM'95

Basic Material Flow Objects (5)

Track

immovable	
passive	used to realise a track of an active MU (e.g. a transporter).
capacity	the length of the track can be freely defined. An active MU cannot over-take another one.
length oriented	
possible application	AGV-track, etc.

EUROSIM'95

Basic Material Flow Objects (6)

Warehouse

immovable	
passive	stores passive MUs (e.g. a container)
capacity	the storage capacity could be freely defined in a x/y-matrix
place oriented	
possible application	aisle, shelf, etc.

EUROSIM'95

Basic Material Flow Objects (7)

Entity

movable	moves during a simulation run. An entity's location can be an immovable basic building block,a transporter or a container. The length of an entity can be freely defined.
passive	an entity has no own drive
capacity=0	cannot hold another part
possible application	parts to be produced or transported, etc.

Basic Material Flow Objects (8)

Container

movable	
passive	
capacity=n	the capacity of a container can be freely defined as a x/y-matrix.
place oriented	
possible application	pallet, box, etc.

Basic Material Flow Objects (9)

Transporter

movable	
active	the transporter has its own drive. The velocity is freely definable
capacity=n	the capacity of a transporter can be freely defined as a x/y-matrix.
place oriented	
possible application	AGV, EOM, fork lifting truck, etc.

EUROSIM'95

The Application Objects *SingleProc*

material flow objects			
movable		immovable	
active	passive	active	passive
• *transporter*	• *container* • *entitiy*	• ***SingleProc*** • *ParallelProc* • *SerialProc* • *Line*	• *track* • *warehouse*

Characteristics:

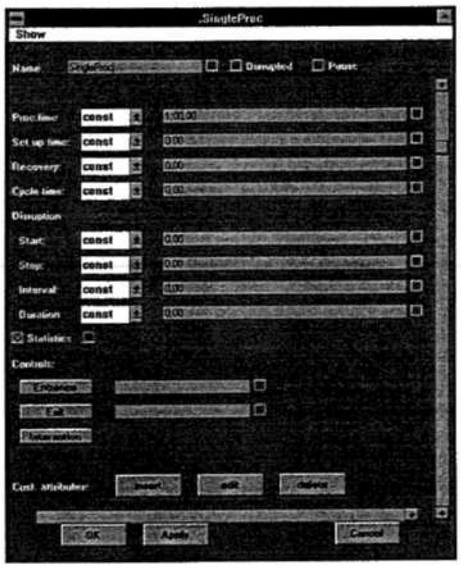
• active material flow object

• capacity: 1

• icon:

EUROSIM'95

The Dialog Window of the *SingleProc*



.SingleProc

Show

Name	SingleProc	Disrupted	Pause
Proc time	const	1:00.00	
Set up time	const	0.00	
Recovery	const	0.00	
Cycle time	const	0.00	
Disruption			
Start	const	0.00	
Stop	const	0.00	
Interval	const	0.00	
Duration	const	0.00	

Statistics

Controls:

Entrance

Exit

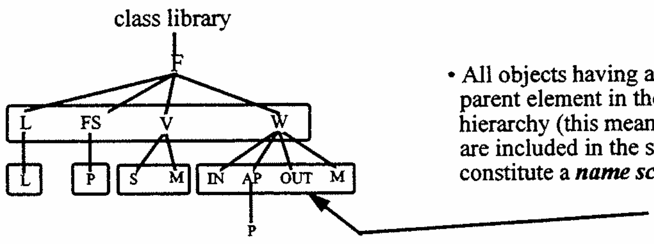
Interpolation

Cust. attributes: insert edit delete

OK Apply Cancel

EUROSIM'95

Name Scope



• All objects having a common parent element in the model hierarchy (this means that they are included in the same object) constitute a ***name scope.***

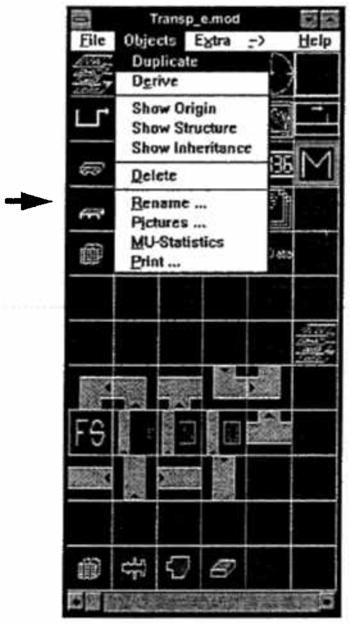
• Within a name scope, the names have to be unique.

• But is possible to use the same names in different name scopes.

EUROSIM'95

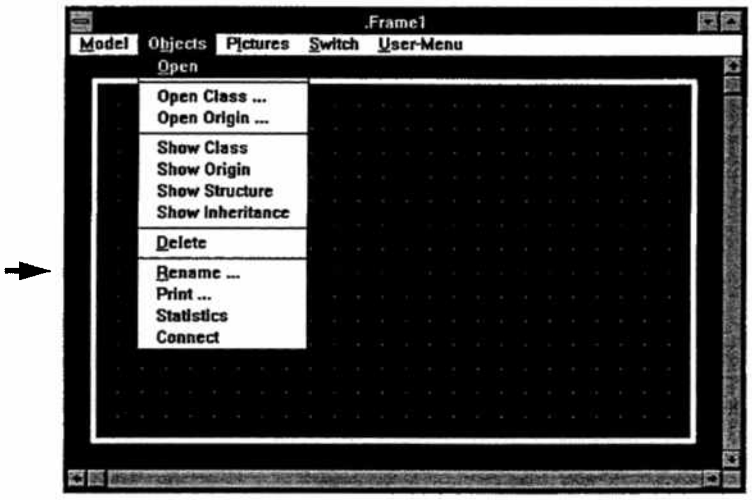
Renaming Objects in the Library

Objects placed in the library can be renamed by selecting the object and then clicking on the menu item *Rename*.



EUROSIM'95

Renaming already Inserted Objects

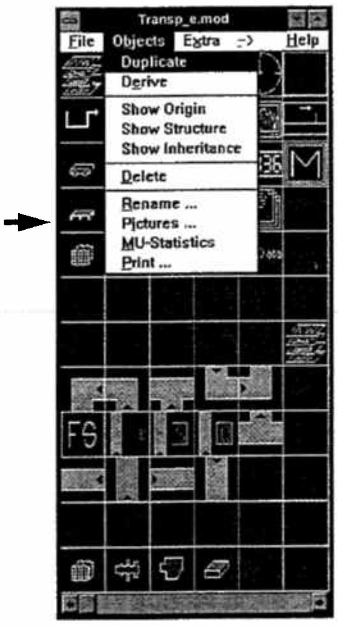


Already inserted objects can be renamed by selecting the object and clicking on the menu item *Rename* in the model *Objects* menu.

EUROSIM'95

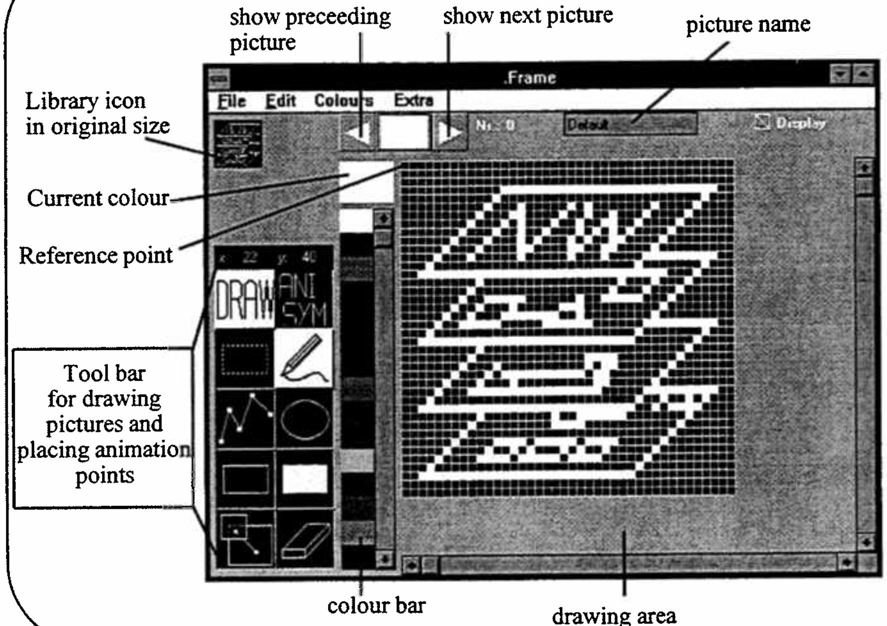
Opening the Picture Editor

The picture editor is opened by first clicking on the desired object and then selecting the menu item *Pictures....*



EUROSIM'95

The Picture Editor



EUROSIM'95

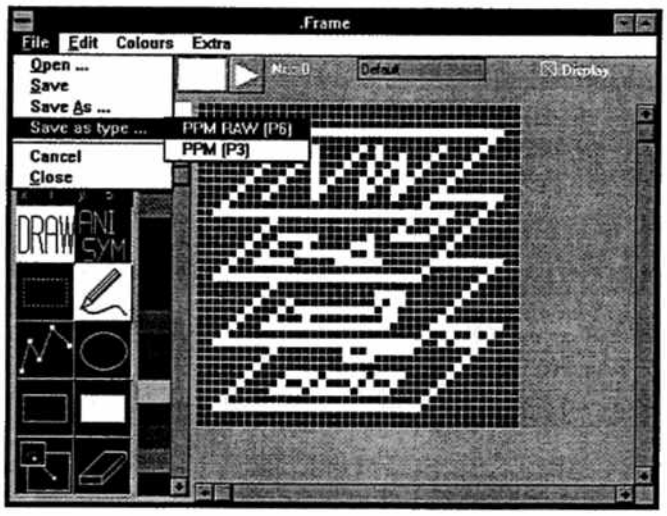
Pictures

- One ore more icons for each* object in the class library can be defined.
- Each icon gets a number from the system and can be named by the user. The icon names for one object must be unique.
- The number of icons is unlimited.
- The maximum size of an icon is 999x999 pixels.
- Each object has an icon numbered with 0 and named *Default* . This is the icon the object is shown in the library. Therefore it has a maximum size of 40x40 pixels.
- Icons require a lot of memory. Therefore they are associated with the class and not with the instances of the class. Consequently, if the icons of an instance of a class are changed, the icons of all instances of the class, including the parent class, are changed too.

* with the exception of the objects with fixed icons - *EventController*, *Comment*, *Connector* and *NwData*

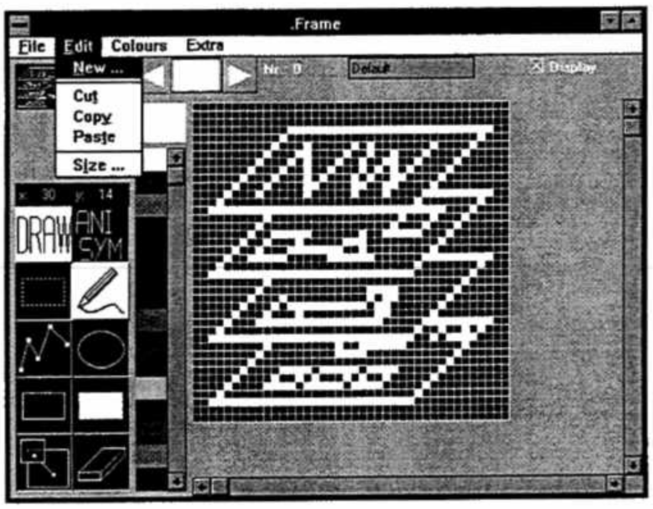
EUROSIM'95

The Menus of the Picture Editor (1)



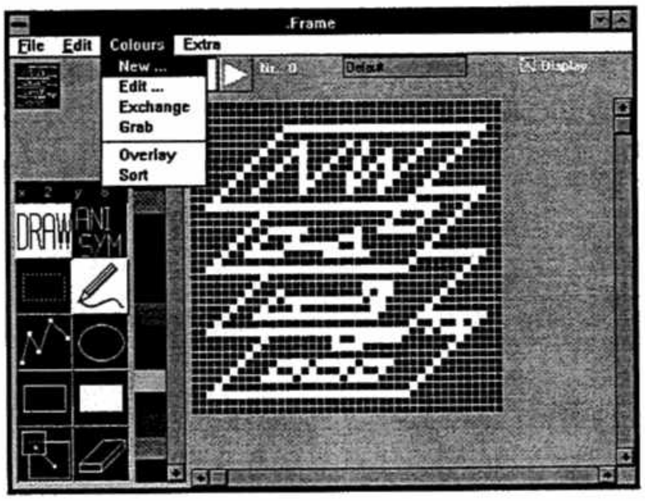
EUROSIM'95

The Menus of the Picture Editor (2)



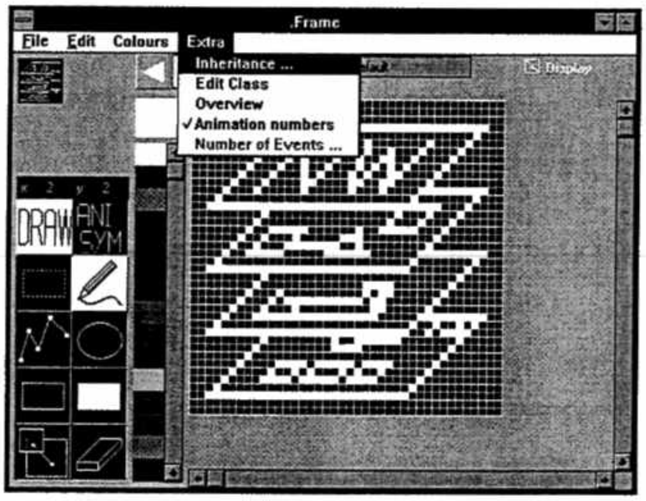
EUROSIM'95

The Menus of the Picture Editor (3)



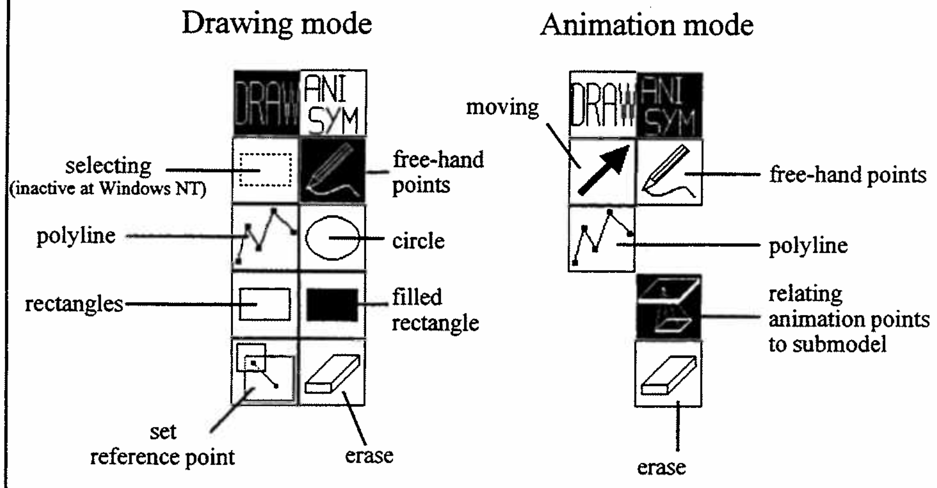
EUROSIM'95

The Menus of the Picture Editor (4)



EUROSIM'95

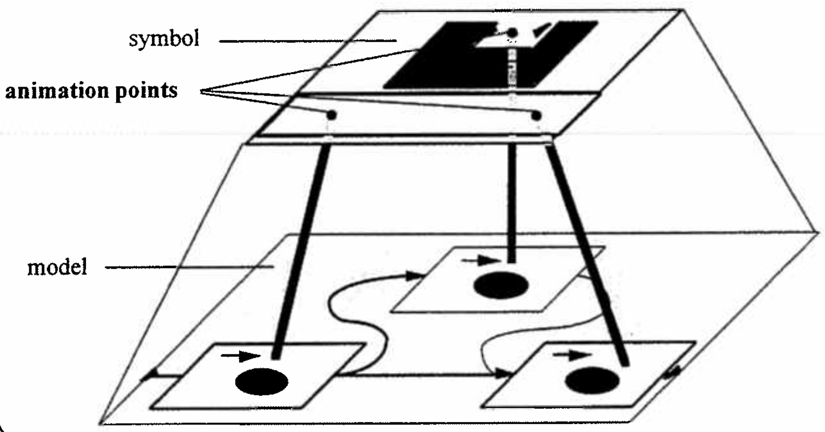
The Drawing Tool Bar



EUROSIM'95

Symbols and Animation Points

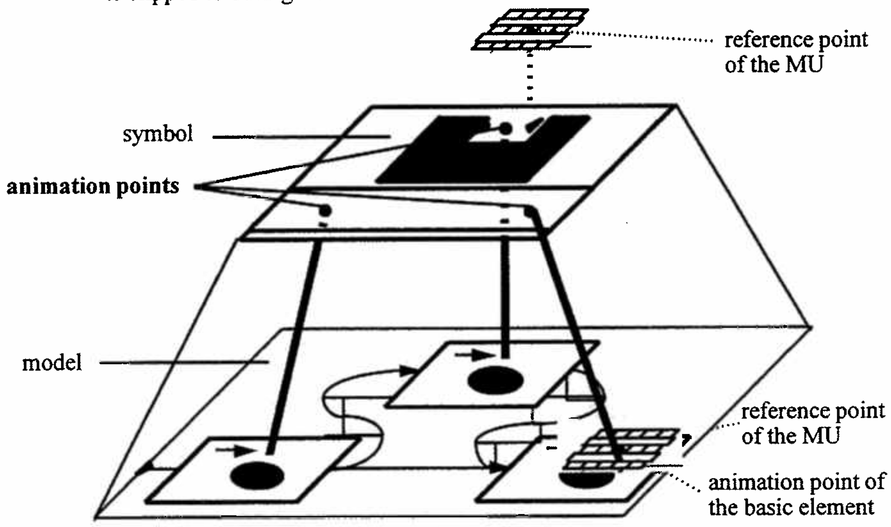
To visualise the MUs, which are moving through the logical components of the model, **animation points** have to be placed in the picture of the symbolic level. Correct animation requires an assignment of these animation points to the logical level.



EUROSIM'95

Animation Points

Animation points are 'attachment points', where the picture of a MU is clipped to during a simulation run.

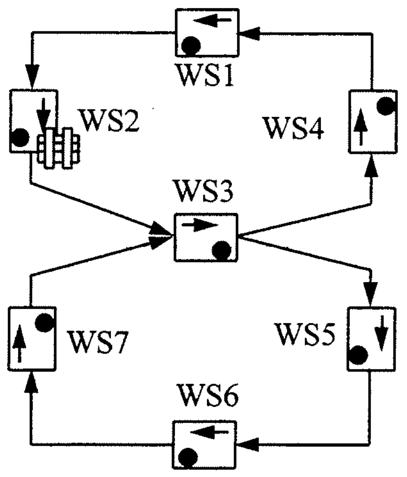

reference point of the MU
symbol
animation points
model
reference point of the MU
animation point of the basic element

EUROSIM'95

Default Material Flow Behaviour at a Divergence Point (1)

+ Create a model according to the following picture:

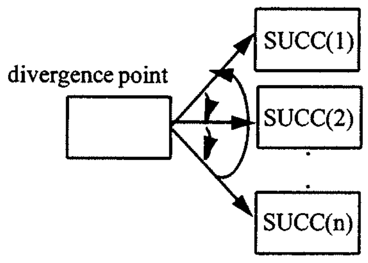
+ Start the simulation and observe the transfer behaviour at the divergence point.



EUROSIM'95

Default Material Flow Behaviour at a Divergence Point (2)

If no user control is specified, the departing entities are transferred to the succeeding building blocks in the sequence in which these were connected to the divergence point

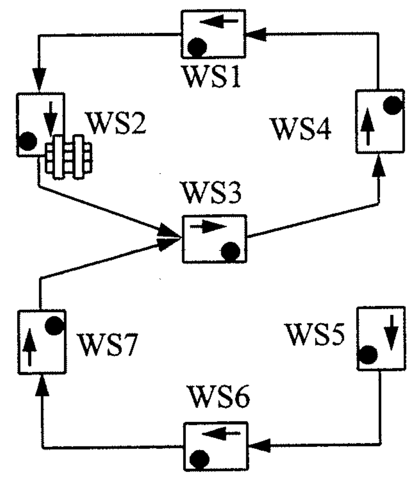


Thereby Simple models can be constructed without developing special control rules.

EUROSIM'95

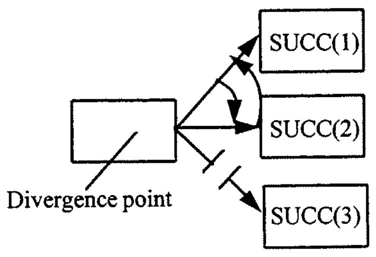
Exercise

+ Delete one connector, start the simulation again and describe the transfer behaviour.



EUROSIM'95

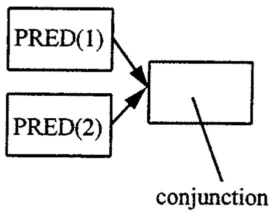
Default Material Flow Behaviour at a Divergence Point (3)



In SIMPLE ++ the departing entities are transferred alternately to the successors which still exist.

EUROSIM'95

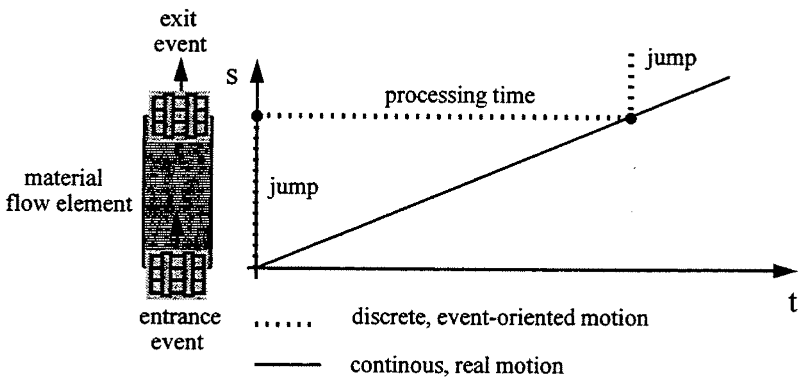
Default Material Flow Behaviour at a Convergence Point



In SIMPLE ++ entities are transferred to the successor following the FIFO-rule (first come, first served).

EUROSIM'95

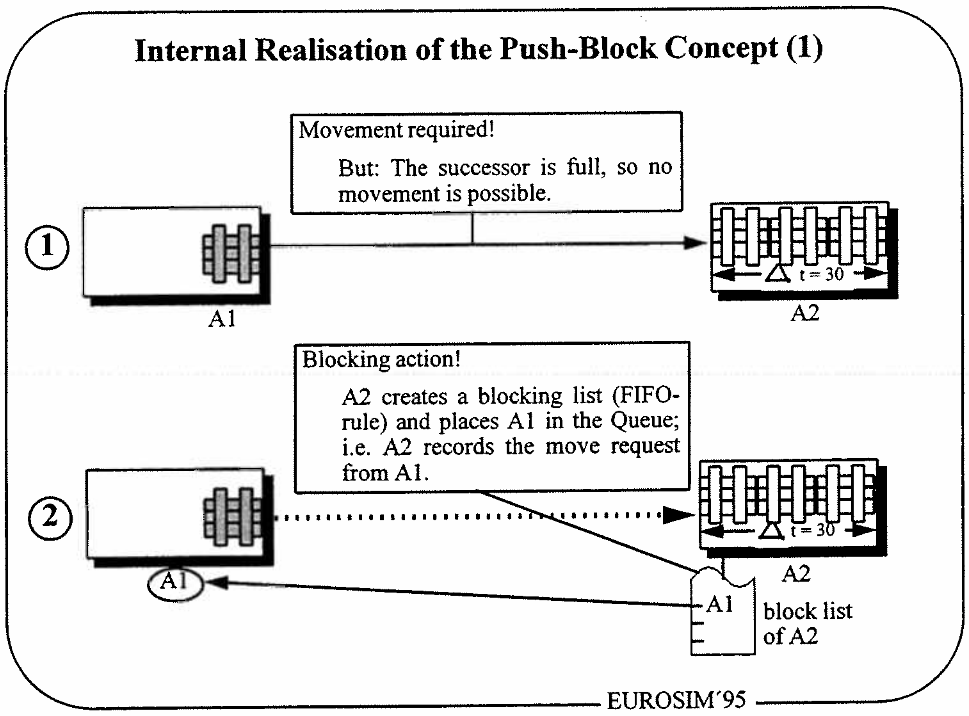
Continuous vs. Discrete Movement



Note: On current digital computers, only discretised simulation is possible. The larger the jump in time, the more calculation time is available for simulation.

EUROSIM'95

Internal Realisation of the Push-Block Concept (1)



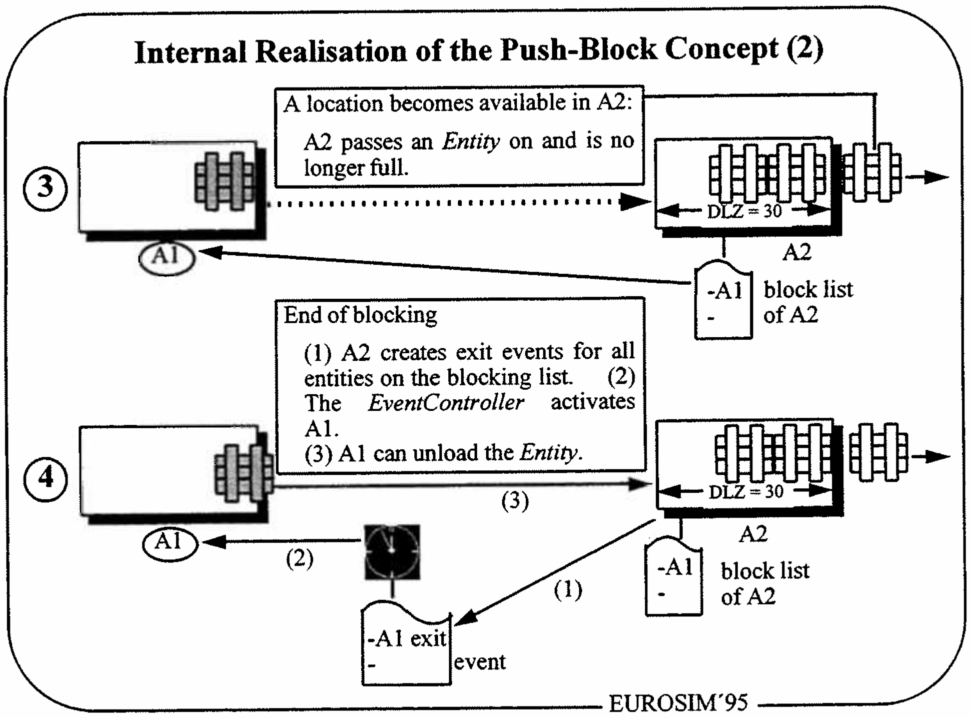
Movement required!

But: The successor is full, so no movement is possible.

Blocking action!

A2 creates a blocking list (FIFO-rule) and places A1 in the Queue; i.e. A2 records the move request from A1.

Internal Realisation of the Push-Block Concept (2)



A location becomes available in A2:

A2 passes an *Entity* on and is no longer full.

End of blocking

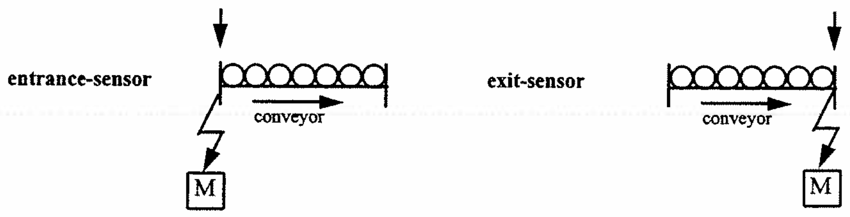
(1) A2 creates exit events for all entities on the blocking list. (2) The *EventController* activates A1.

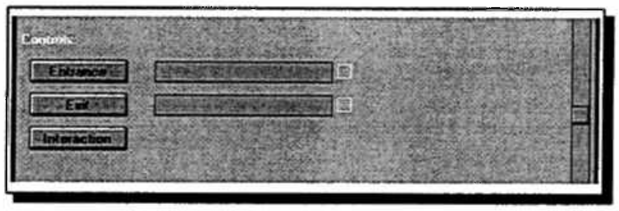
(3) A1 can unload the *Entity*.

The Sensor-Actor Concept (part 1)

The basic material flow elements (*SingleProc, ParallelProc, ...*) have an *Entrance-* and an *Exit-Control.*

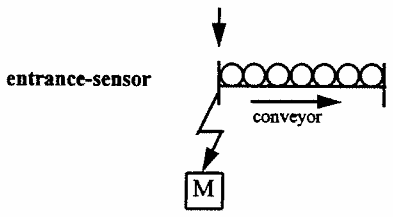
A method specified as an entrance or exit control will be activated, whenever a MU wants to enter or leave that material flow element.





EUROSIM'95

The Sensor-Actor Concept The Entrance Control



Activation:

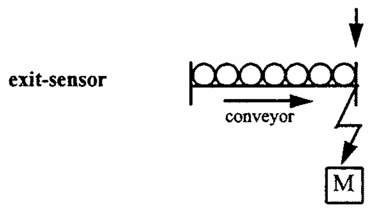
• the entrance control is activated when the MU succesfully enters the building block

Important:

• when the entrance control is activated, the MU has already moved onto that building block, i.e. the change of the processing time within the entrance control does not affect the present MU.
• the entrance control does overwrite the default behaviour of the building block
• the entrance control can be activated only once for each entering MU

EUROSIM'95

The Sensor-Actor Concept
The Exit Control



Activation:

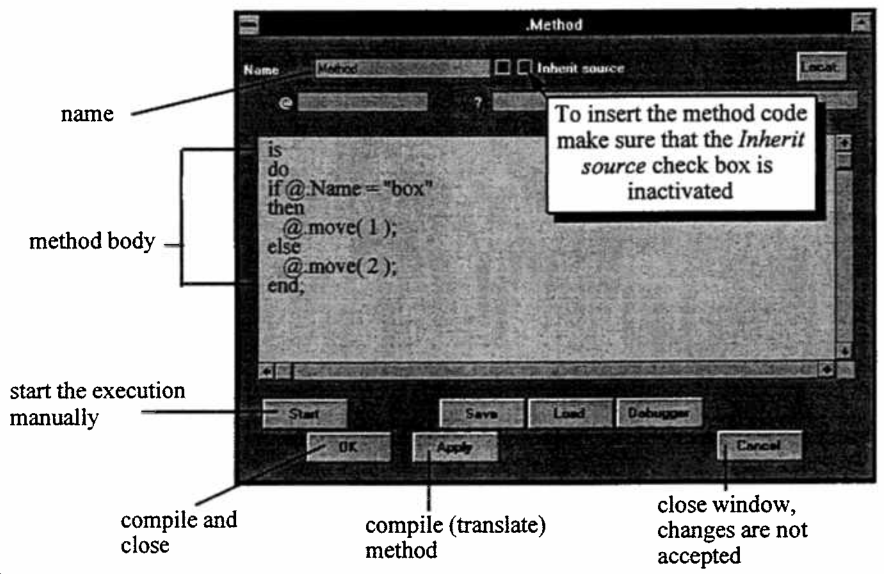
• the exit control is activated whenever an MU wants to leave its location

Important:

• the exit control overwrites the dafault behaviour of the building block, i.e. the MU will **not be moved** to the successor **automatically**
• the exit control may be activated more than once by a single MU.
Reason: If the MU is blocked and cannot leave its location, the exit control will be activated a second time when the MU becomes operational again.

EUROSIM'95

Construction of a Method (Control)



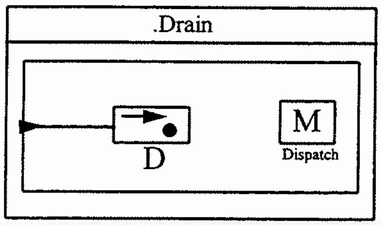
```
is
do
if @.Name = "box"
then
  @.move( 1 );
else
  @.move( 2 );
end;
```

To insert the method code make sure that the *Inherit source* check box is inactivated

EUROSIM'95

The Application Object *Drain* (1)

model:



symbol:

Exercise:

familiar

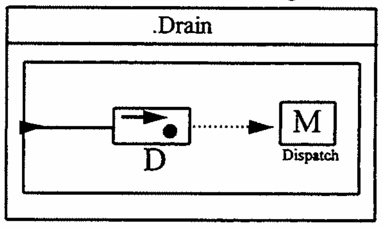
1. open the pre-defined building block *Term*
2. insert the basic object *SingleProc*
3. insert the basic object *Method*
4. connect the model entrance with the *SingleProc* (manually)
5. rename the the building block
6. draw a separate picture for the model
7. define an animation point
8. assign the animation point to the *SingleProc*

EUROSIM'95

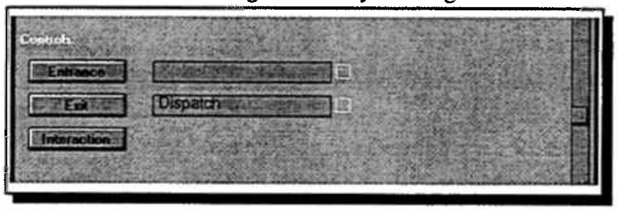
The Application Object *Drain* (2)

new

9. define the control method *Dispatch* as the exit control of the user-defined building block *D*.



a. open the dialog window of the *SingleProc*
b. enter the name of the exit control method
c. close the dialog window by clicking on the *OK*-button



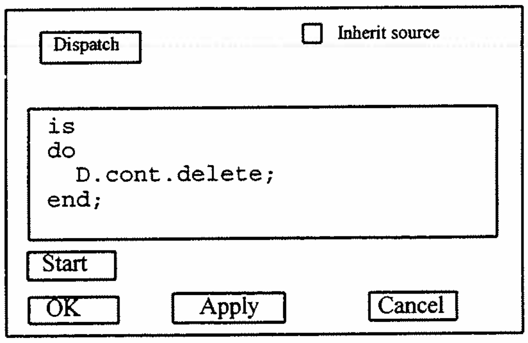
Exit: Dispatch

EUROSIM'95

The Application Object *Drain* (3)

new

10. define the exit control method
    a. open the window of the *Dispatch* method with a double click.
    b. switch off the *Inherit source*-button
    c. enter the method the code shown below
    d. close the window with the *OK*-button

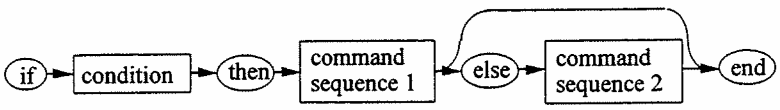


```
is
do
  D.cont.delete;
end;
```

EUROSIM'95

Flow Control Structures - Conditional Branch

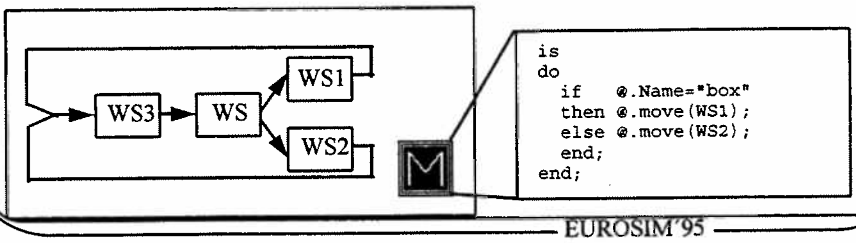
The conditional branch is used to make the execution of command sequences depending on a certain condition.



If the condition is fulfilled (TRUE), then command sequence 1 is executed, otherwise command sequence 2 is executed, if an else branch is specified.

The *else* -branch is optional.

+ Build a model with a method using the conditional check, e.g.:

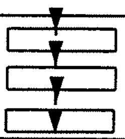
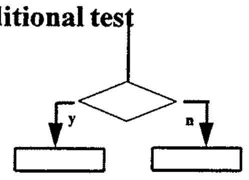
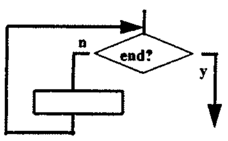


```
is
do
  if    @.Name="box"
  then @.move(WS1);
  else @.move(WS2);
  end;
end;
```

EUROSIM'95

Flow Control Structures

Flow control structures are used to control the sequence of commands executed in the information flow language *SimTalk*. *SimTalk* offers three different flow control structures:

sequence		ONE or SEVERAL commands which are placed and executed in sequence.
conditional test		Different sequences of commands are executed depending on the result of the conditional test.
loop		If the termination condition evaluates to FALSE, the command sequence is executed and the termination condition is evaluated again.

EUROSIM'95

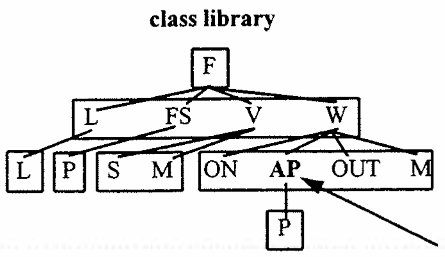
How to create Addresses?

Object addresses - overview

- relative address
- absolute address
- anonymous names
- navigating
  - through the model hierarchy
  - through the logical network

EUROSIM'95

Relative Address



A relative address is always relative to the name scope, i.e. the hierarchical level of the model where it is used (as opposed to an absolute address stating with the class library). In the example below, SIMPLE++ searches the name scope of method *M* for an object with the name *AP*.

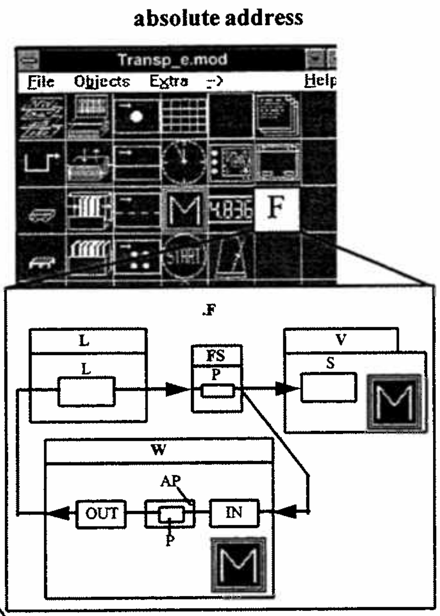
+ Insert the following lines into the method Method *.F.W.M*:

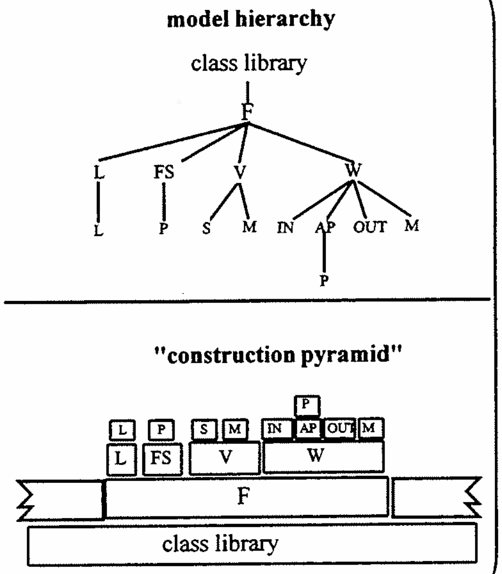
```
is
do
  print AP;
end;
```

+ Click *Apply* and *Start* and watch the result in the standard output.

EUROSIM'95

Absolute Address



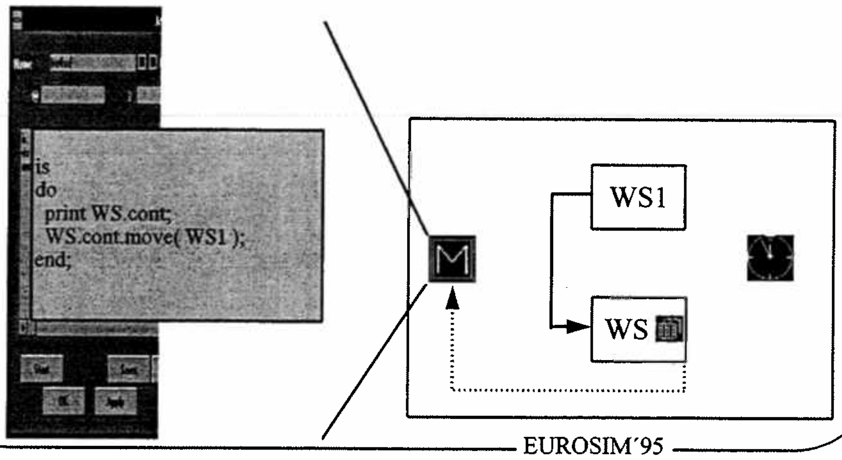


EUROSIM'95

The Anonymous Name *cont*

Cont returns the address of the MU being loaded on an object.

+ • Build the model shown below (WS and WS1 have to be *SingleProcs*).
• Insert M as the exit control of WS.
• Start the simulation and watch the animation and the standard output.



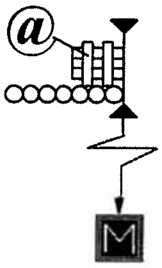
```
is
do
  print WS.cont;
  WS.cont.move( WS1 );
end;
```

EUROSIM'95

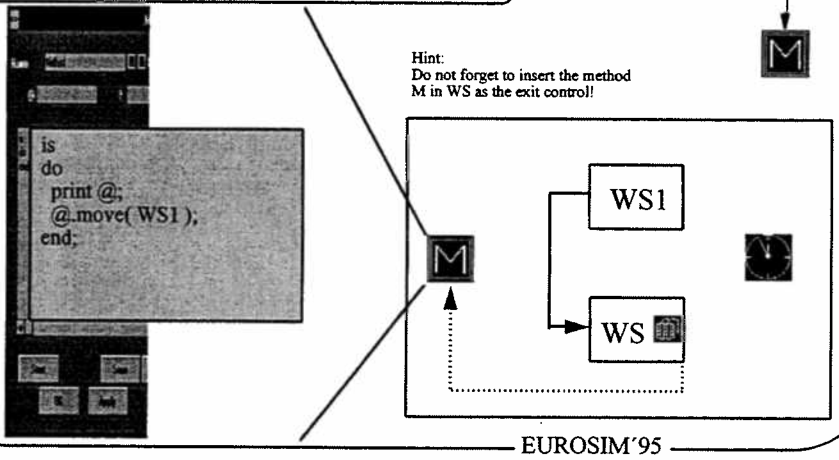
The Anonymous Name @

@ is the address of the movable unit which triggered the current event.

+ • Build the model described below.
• Start the simulation, watch the standard output and the animation.



Hint:
Do not forget to insert the method M in WS as the exit control!



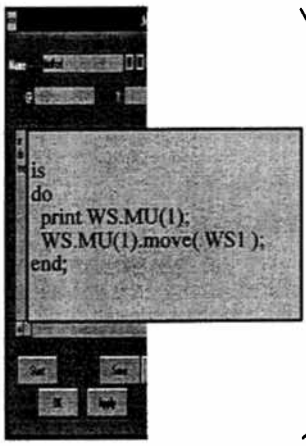
```
is
do
  print @;
  @.move( WS1 );
end;
```

EUROSIM'95

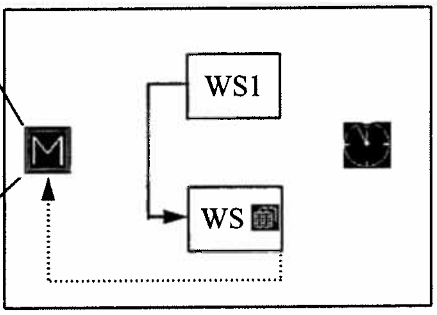
The Anonymous Name *MU(n)* (1)

MU(n) returns the address of the n-th MU in a stationary material flow object (e.g. *SingleProc, ParallelProc,...*). (The MUs are arranged in the departure order).

- Change the latest model to the model shown below.
- Start the simulation and watch the animation and the standard output.



```
is
do
  print WS.MU(1);
  WS.MU(1).move( WS1 );
end;
```



EUROSIM'95

The Anonymous Name *current*

current is a pointer to the location of the current method.

Insert the following lines in a method, click *Apply* and *Start* and watch the stardand output:

```
is
do
  print current;
end;
```

EUROSIM'95

Further Anonymous Names

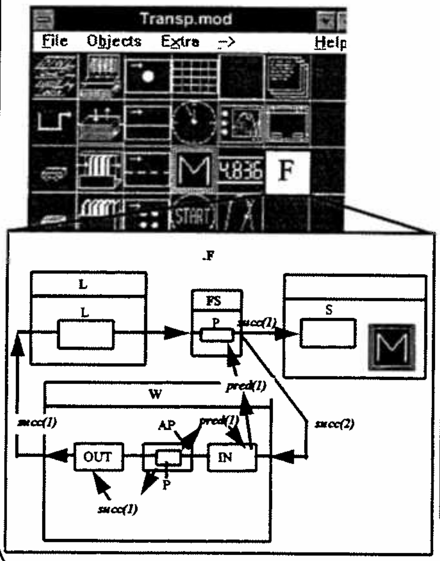
In the navigating chapter you've got to know some more anonymous names:

- *location*
- *pred(n)*
- *succ(n)*
- *root*

+ Think of an example for each construct. Make sure that you are able to use them in the right way.

EUROSIM'95

Navigating through the Logical Networks



+ • Insert the following lines in the method building block *F.W.M.*

```
is
do
  print AP.succ(1);
  print AP.succ(1).succ(1);
  print AP.pred(1);
  print OUT.pred(1).pred(1);
  print OUT.succ(1).succ(1).succ(2);
end;
```

• Click on *Apply* and *Start* and watch the result in the standard output.

From a given object you can move through the logical network. With ... *succ(n)* you access the n-th successor and with ... *pred(n)* the n-th predecessor.

EUROSIM'95

Customised Attributes

Customised attributes can be defined for all material flow objects and lists. The customised attributes can be used to store information like part type, part routing, order number, etc.

To insert a customised attribute, open the data dialog box of the basic objects and move the scroll bar to the bottom. The field *Cust. attributes* will appear. After clicking the *insert*-button a new window is displayed in which you can insert the name, type and value of the customised attribute.

The number of attributes is not limited.

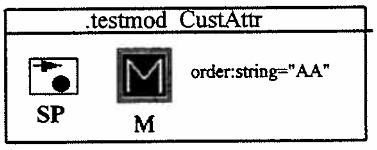
A customised attribute consists of a **name**, a **data type** and a **value**.

EUROSIM'95

Accessing the Value of a Customised Attribute (1)

+ Create the following test model

layout

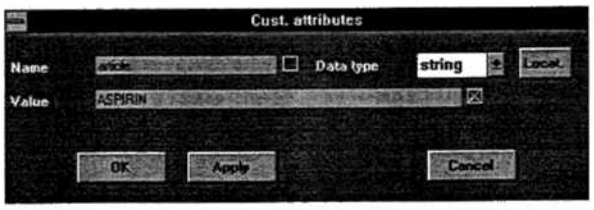


Exercise:

familiar

1. copy an empty model in the library
2. open the model
3. insert a basic object *Method*
4. insert a basic object *SingleProc*
5. rename the basic object to *SP*
6. insert a customised attribute for *SP*

new


Cust. attributes

Name: article ☐ Data type: string [Local]

Value: ASPIRIN ☒

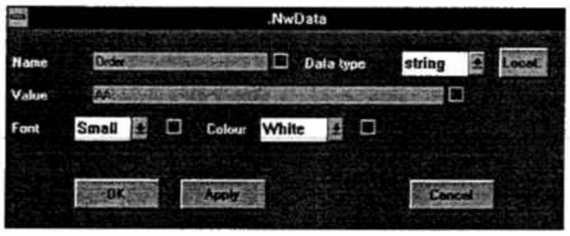
7. close the dialog window with the *OK*-button

EUROSIM'95

Accessing the Value of a Customised Attribute (2)

familiar

8. insert an instance of the building block *NwData*
9. open the dialog window with a double click and fill it out



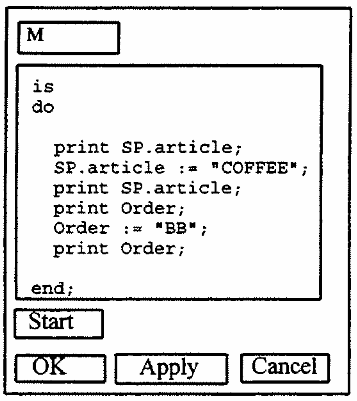
Name: Order
Data type: string
Value: AA
Font: Small
Colour: White

10. close the window with the *OK*-button

Accessing the Value of a Customised Attribute (3)

familiar

11. enter the following method code and click on the *Apply*-button



```
is
do

   print SP.article;
   SP.article := "COFFEE";
   print SP.article;
   print Order;
   Order := "BB";
   print Order;

end;
```

12. click on the *Start*-button and watch the standard output (the window from which Simple++ was started)
13. open the *SP*-dialog window and look for the value of the customised attribute
14. close the model

Data Types

Customised attributes consist of a **name**, a **data type** and a **value**.

SIMPLE++ offers a range of different data types:

• *boolean*:	truth value; can be "TRUE" or "FALSE"
• *integer*:	whole number, e.g. 1, 127, -3566
• *real:*	floating point number, e.g. 3.141, -382.777
• *string*:	string of characters or numbers; e.g. "This is text", "123-ABC"
• *object:*	address of a model element (absolute address), e.g. .F.W.AP

...

EUROSIM'95

Boolean Operators

The following operators can be used with the data type *boolean*:

operator	example
• AND	TRUE AND FALSE. Always FALSE if one of the operands is FALSE.
• OR	TRUE OR FALSE. Always TRUE if one of the operands is TRUE.
• NOT	NOT TRUE is FALSE. Reverses the value of the boolean operator.
• =	FALSE = FALSE. TRUE if the operands are equal.
• /=	FALSE /= TRUE. TRUE if the operands are not equal.
• bool_to_num:	conversion operator, e.g. bool_to_num (FALSE) returns 0.
• bool_to_str:	conversion operator, e.g. bool_to_str (FALSE) returns "FALSE".

EUROSIM'95

Integer Operators

+	addition	
-	subtraction	integer
*	multiplication	
//	integer division	
\\	modulo operator	integer
/	division	real
=	equal	
/=	not equal	
>	greater than	
<	less than	boolean
>=	greater than or equal	
<=	less than or equal	

num_to_bool, num_to_string: type conversion operator

+ Perform different experiments with the conversion operators.

EUROSIM'95

REAL Operators

+	addition
-	subtraction
*	multiplication
/	division
=	equal
/=	not equal
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal

num_to_bool, num_to_string: type conversion operators

+ Perform different experiments with the REAL operators.

EUROSIM'95

STRING Operators

+	concatenation	string
=	equal	boolean
/=	not equal	boolean
	toLower	string
	toUpper	string
	copy	string
	incl	string
	omit	string
	strlen	integer
	pos	integer
	sprint	object

num_to_bool, num_to_string: type conversion operators

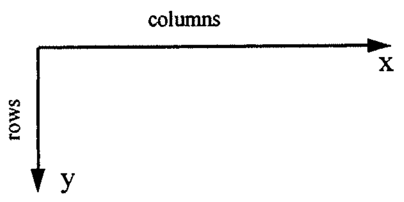
+ Perform different experiments with the STRING operators.

EUROSIM'95

The Basic Object *TableFile* 2D list (1)

WorkingPlan:

	1	2	3
1	a1	a2	...
2	b1	b2	
3	c1	*c2*	drain
4	d1	d2	drain



Specific fields are accessed via an index, e.g.:

read entry: *WorkingPlan[2, 3];* -> c2

column | row

write entry: *WorkingPlan[3, 4] := "drain";*

assignment

EUROSIM'95

The Basic Object *TableFile* 2D list (2)

In addition to the system defined index (integer), the user can define indices for both the columns and the rows, e.g.:

WorkingPlan:

		1	2	
	▨	**station1**	**station2**	**column index**
1	**box**	source	drill	
2	**shaft**	source	lathe	
	row index			

example: *print WorkingPlan["station2", "shaft"]* -> lathe

EUROSIM'95

The Basic Object *CardFile, StackFile, QueueFile*

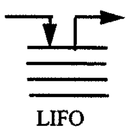
- *QueueFile*

QF.push("lathe");

QF.pop; ↷

QF.top; ↷

- *StackFile*

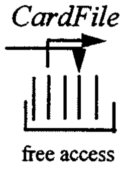


SF.push("lathe");

SF.pop; ↷

SF.top; ↷

- *CardFile*



CF.insert(2, "lathe");

! CF.cutRow(4); ↷ !

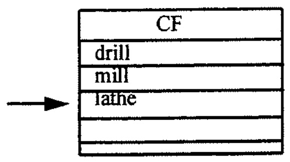
CF.read(4); ↷

EUROSIM'95

Searching a 1D-list

- each list has an internal cursor which points to the current field of the list.
- the list can be searched automatically. The result of the search can be used in boolean expression to determin whether a certain value was found or not.
- the search starts at the current cursor position and continues until the value is found or the end of the list is reached.
- if the correct value is found, the internal cursor points to the field containing this value and the boolean result *TRUE* is returned. If a search is unsuccessful, the cursor position is unchanged.
- if you want to search the complete list, the cursor must be set to the first field of the list prior to searching.
- **commands (e.g.):**

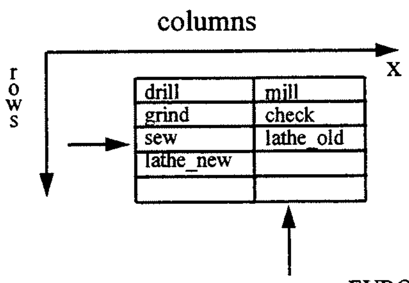
```
CF.Cursor := 1;
print CF.find( "lathe" );        TRUE
print CF. Cursor;                3
```

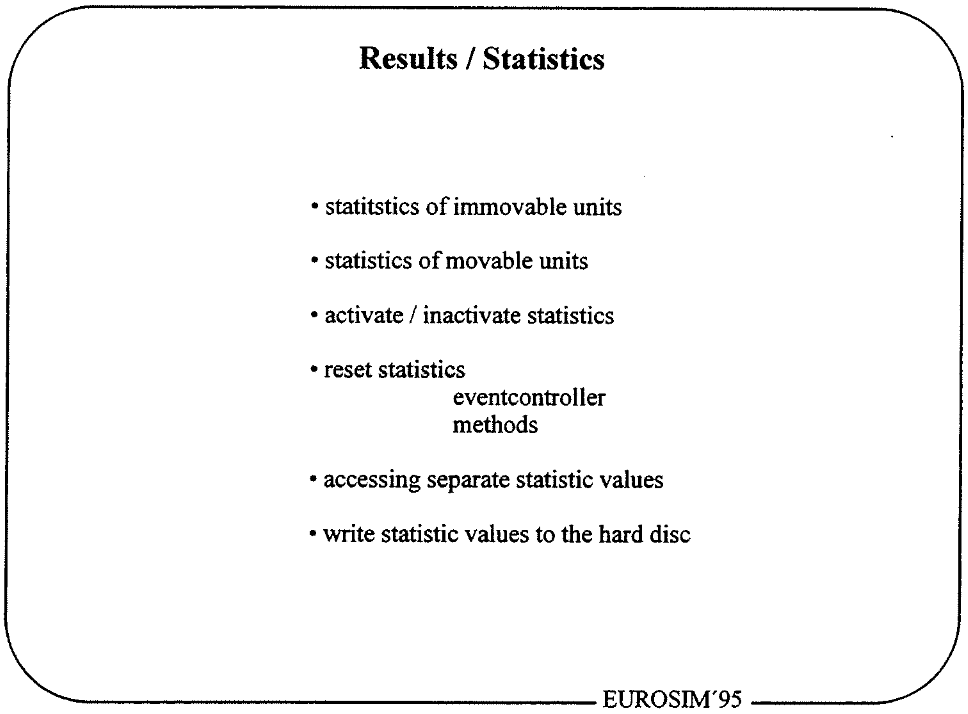


Searching a 2D-list

- **example**:

```
TF.CursorX := 1;
TF.CursorY := 1;
print TF.find( "lathe_old" );        TRUE
print TF.CursorY;                    3
print TF.CursorX;                    2
```





Results / Statistics

• statitstics of immovable units

• statistics of movable units

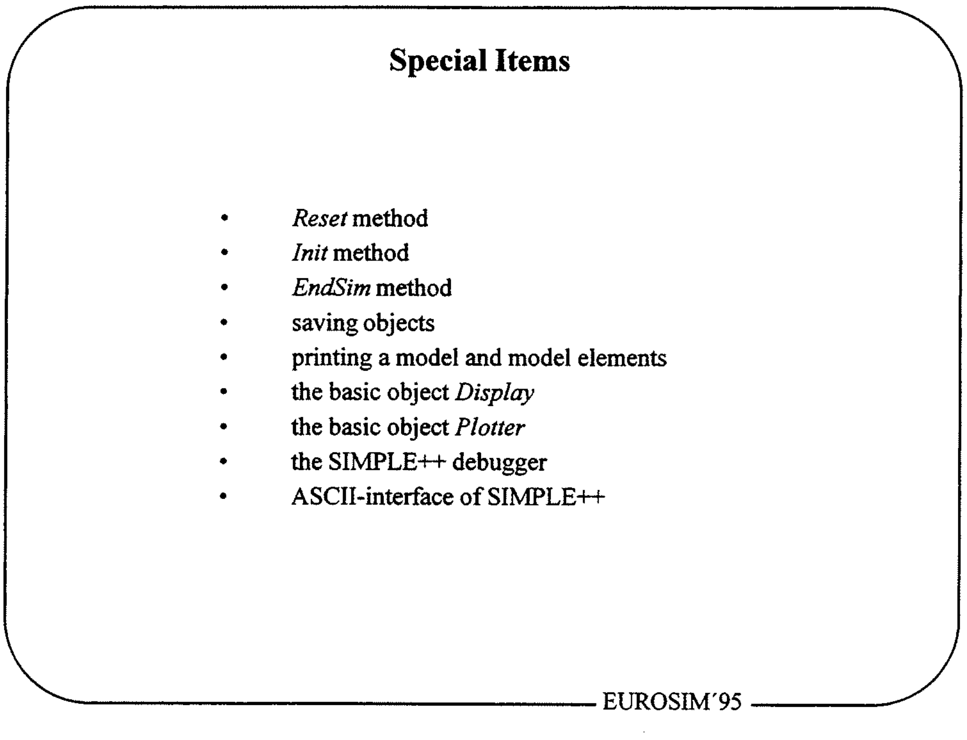
• activate / inactivate statistics

• reset statistics
  eventcontroller
  methods

• accessing separate statistic values

• write statistic values to the hard disc

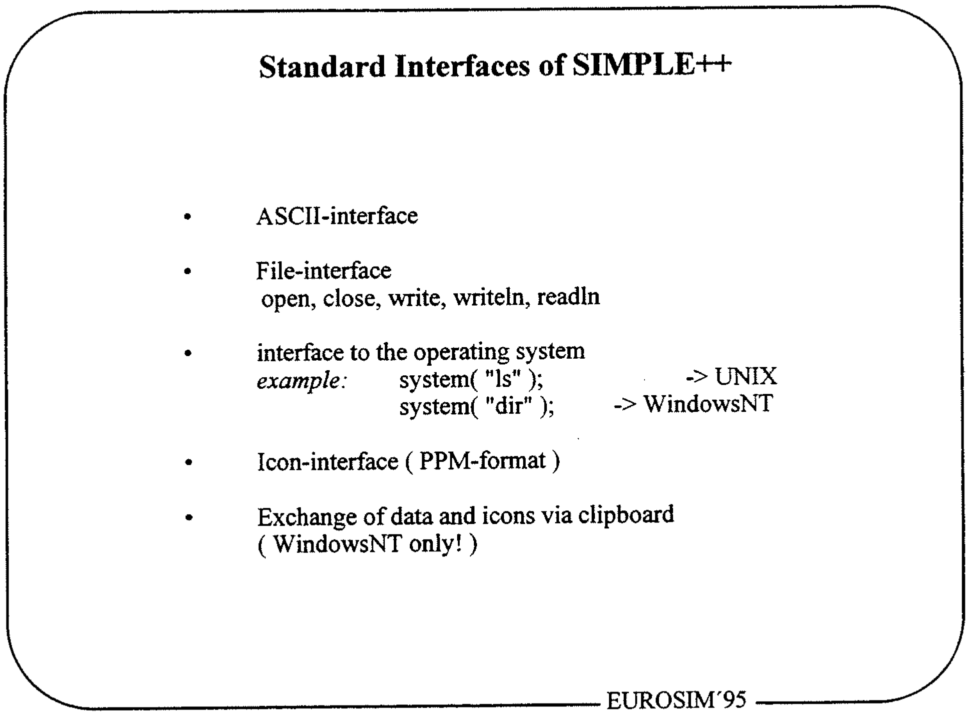
EUROSIM'95



Special Items

- *Reset* method
- *Init* method
- *EndSim* method
- saving objects
- printing a model and model elements
- the basic object *Display*
- the basic object *Plotter*
- the SIMPLE++ debugger
- ASCII-interface of SIMPLE++

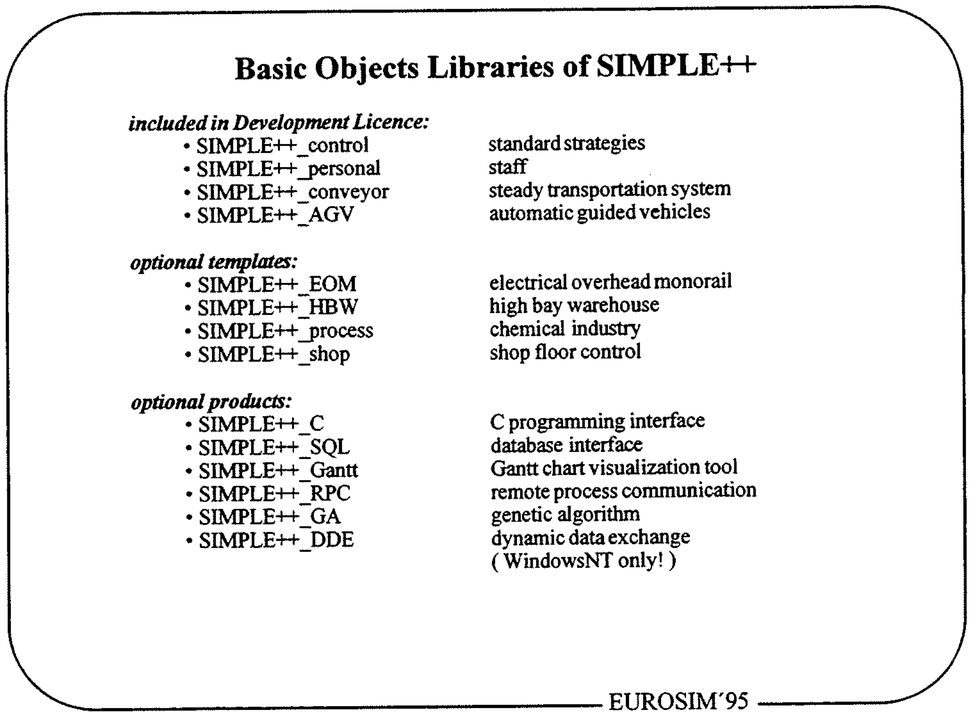
EUROSIM'95



Standard Interfaces of SIMPLE++

- ASCII-interface
- File-interface
  open, close, write, writeln, readln
- interface to the operating system
  example: system("ls"); -> UNIX
  system("dir"); -> WindowsNT
- Icon-interface (PPM-format)
- Exchange of data and icons via clipboard
  (WindowsNT only!)

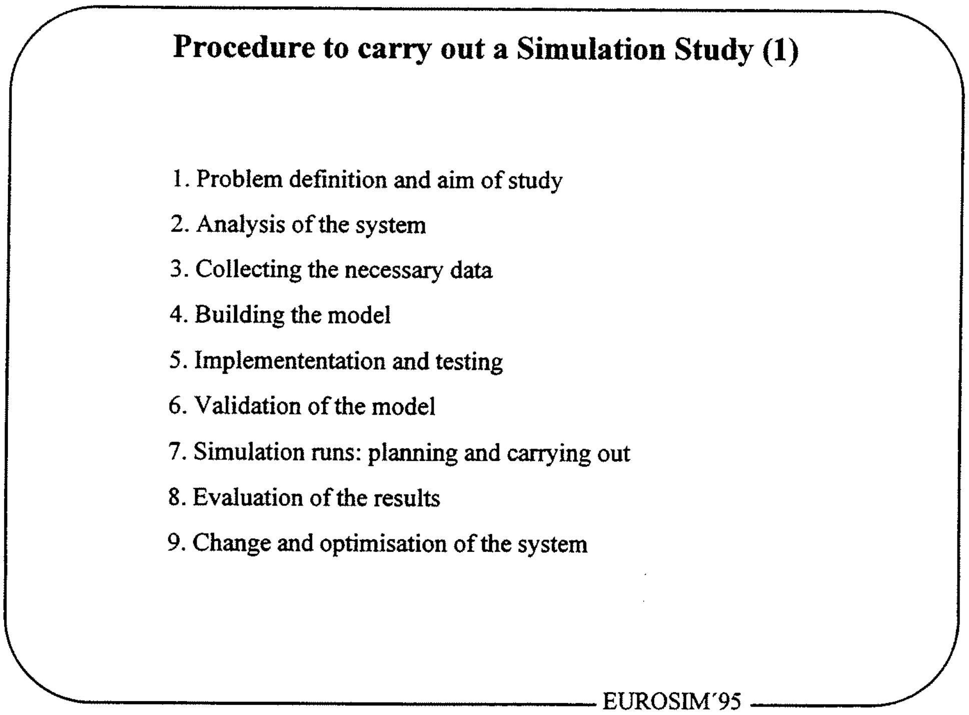
EUROSIM'95



Basic Objects Libraries of SIMPLE++

included in Development Licence:	
• SIMPLE++_control	standard strategies
• SIMPLE++_personal	staff
• SIMPLE++_conveyor	steady transportation system
• SIMPLE++_AGV	automatic guided vehicles
optional templates:	
• SIMPLE++_EOM	electrical overhead monorail
• SIMPLE++_HBW	high bay warehouse
• SIMPLE++_process	chemical industry
• SIMPLE++_shop	shop floor control
optional products:	
• SIMPLE++_C	C programming interface
• SIMPLE++_SQL	database interface
• SIMPLE++_Gantt	Gantt chart visualization tool
• SIMPLE++_RPC	remote process communication
• SIMPLE++_GA	genetic algorithm
• SIMPLE++_DDE	dynamic data exchange (WindowsNT only!)

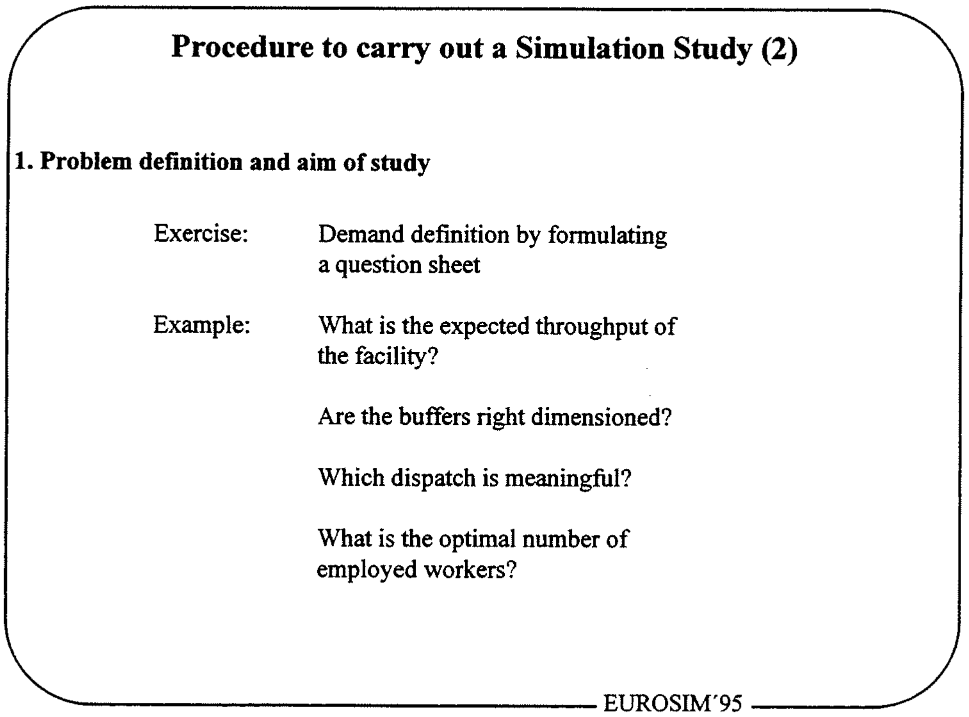
EUROSIM'95



Procedure to carry out a Simulation Study (1)

1. Problem definition and aim of study
2. Analysis of the system
3. Collecting the necessary data
4. Building the model
5. Implementententation and testing
6. Validation of the model
7. Simulation runs: planning and carrying out
8. Evaluation of the results
9. Change and optimisation of the system

EUROSIM'95



Procedure to carry out a Simulation Study (2)

1. Problem definition and aim of study

Exercise:	Demand definition by formulating a question sheet
Example:	What is the expected throughput of the facility?
	Are the buffers right dimensioned?
	Which dispatch is meaningful?
	What is the optimal number of employed workers?

EUROSIM'95

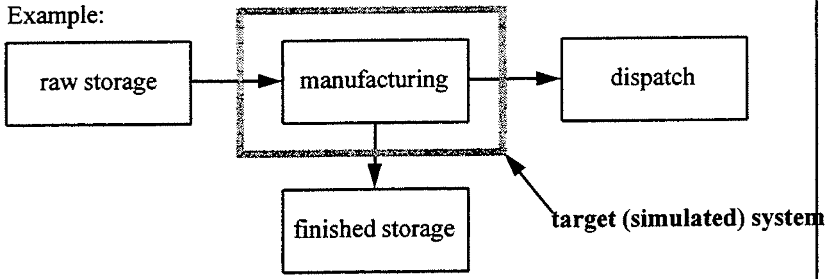
Procedure to carry out a Simulation Study (3)

2. Analysis of the system (1)

Exercise: Describe the system to be simulated

Outline of the system vs. environment

Example:



EUROSIM´95

Procedure to carry out a Simulation Study (4)

2. Analysis of the system (2)

Exercise: Analysing the elements of the system

Determine the attributes of the elements

Example:

- Processing time
- Capacity
- Availability
- etc.

EUROSIM´95

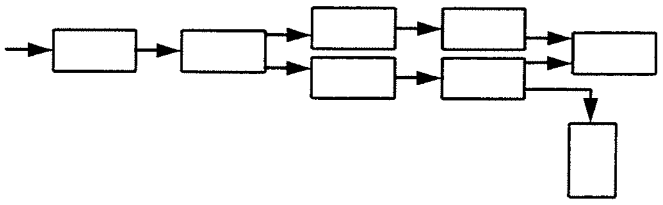
Procedure to carry out a Simulation Study (5)

2. Analysis of the system (3)

Exercise: Determine the structure of the system

Mutual interface specification of the system elements

Example:



EUROSIM'95

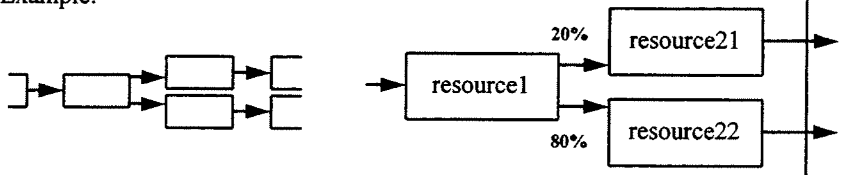
Procedure to carry out a Simulation Study (6)

2. Analysis of the system (4)

Exercise: Analysing the dispatch and process rules

Determine this rules affecting the entities and the facility

Example:



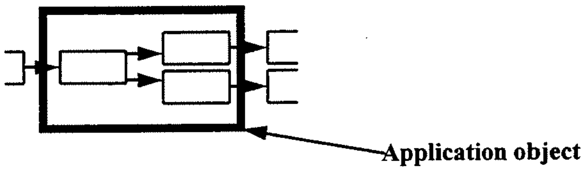
EUROSIM'95

Procedure to carry out a Simulation Study (7)

2. Analysis of the system (5)

Exercise:	Defining the required application objects
	Is it possible to use pre-defined application objects?

Example:



EUROSIM'95

Tuning Tips for Simulation Runs

- Switch off the animation
- Switch off the interactive stopping
- Close the *EventController*
- Activate the fast index access (*TableFile*)
- Animation activated:
  - hide grid
  - hide connectors
  - hide object names
  - use pictures in original size
  - as few animation points as possible
  - as few animation events as possible (animation line)

EUROSIM'95

How to model with SIMPLE++

- Identify the system elements
- Encapsulate these elements to application objects
- Is it possible to use pre-defined application objects?
- Design the application objects **on paper**
  - define the interfaces
  - determine the structure of the building blocks
  - formulate the controls in spoken language
- Design the application objects
- Test the objects one by one
- Build up the complete model and test it

EUROSIM'95

Doing Simulation Studies (1)

1. System definition/goals

Compile a description of the real system and fix the borders of it.
Keep asking questions until the functionality of single elements and the general behaviour of the overal system are clear.
Formulate the goals of the simulation study.
This is a very important step as the goals determine the necessary level of detail for modelling and thereby the effort which is necessary.

2. Design

Make a list of all elements of the system to be modelled. Evaluate which elements have a similar or identical functionality.
From that a list of building blocks to be modelled can be created.
Try to think of already existing objects which may be used.
Specify and design the remaining objects on paper. Define and describe explicitely the interfaces for material and information flow.
If necessary specify *Reset* and *Init* methods.

EUROSIM'95

Doing Simulation Studies (2)

3. Data

Make sure that the data necessary for the simulation of the real system are available or are captured if necessary.
This often causes considerable effort and may be a longer process.
A competent and skilled contact person who is responsible for organising the data is important.

4. Implementation

According to the design, the building blocks are created in Simple++.
The model is then built using these building blocks. However, the model may also be structured into several hierarchical levels.
Do not forget the documentation for the model.

EUROSIM'95

Doing Simulation Studies (3)

5. Verification

This step is to make sure that the single building blocks have the correct functionality and behave according to the specification.
The single objects of the model usually are tested on their own, in conjunction with further building blocks and in the entire model.
Verify that all parameters are set to correct values.

6. Validation

Check that the model corresponds to the planned or to the existing system. Develop estimates for important results and check if the simulation results are in a corresponding range. Discuss the model, the behaviour of the model and the results with an expert.

7. Plan for simulation experiments

Make a plan for the number of simulation runs, the variation of parameters, variation of random distributions, etc.

EUROSIM'95

Doing Simulation Studies (4)

8. Simulation experiments

Perform the necessary simulation runs and collect the results.

9. Evaluation of results

The results gained by the experiments are evaluated in this step and are documented.

10. Variation, Optimisation

Check if the goals as specified in step 1 are reached by the experiments. If necessary, adjust the model and perform further simulation runs until you reach an optimum result.

Simulation studies usually are an evolution process. A first design will often be revised a few times where new findings or results will lead to modifications in earlier steps of the entire process. Repeated loops will thereby lead from a fairly rough first draft to a final detailed model.

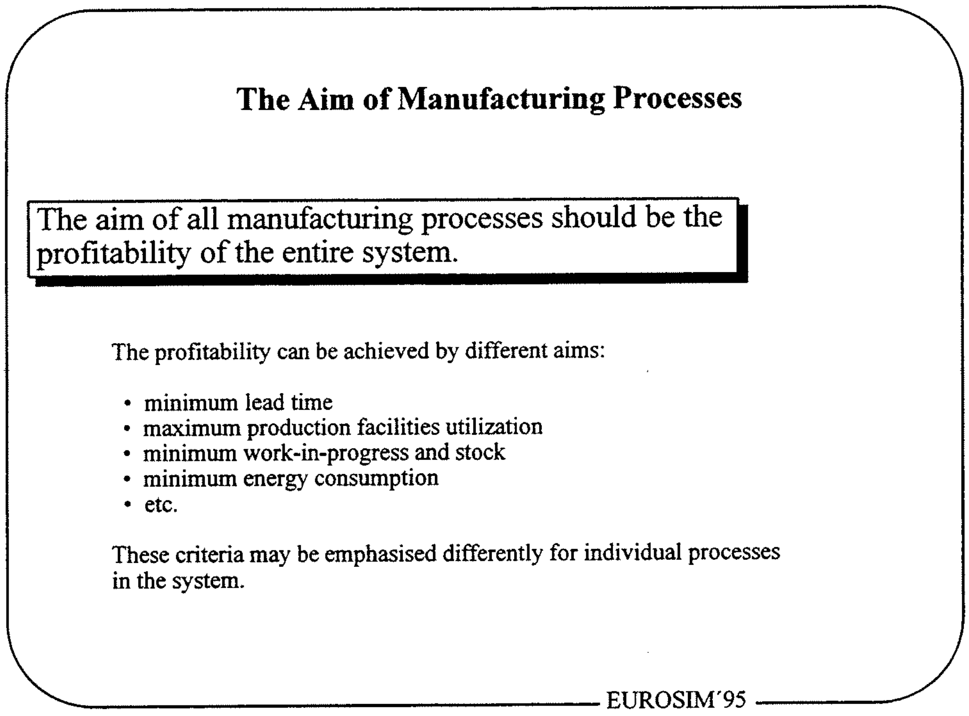
EUROSIM'95

Why Simulation of Manufacturing Systems?

A manufacturing system consists of a variable number of interacting elements. The interactions consist of parallel and serial processes which exchange information and material. The complex nature of modern production systems means that with only a few elements the number of factors which need to be considered is so vast that rational and exact planning with conventional planning tools is impossible.

To obtain measures for optimisation of the facility during the planning phase, some techniques are necessary to model the interaction of the physical and logical elements of each system precisely, i.e. in its full complexity.

EUROSIM'95



The Aim of Manufacturing Processes

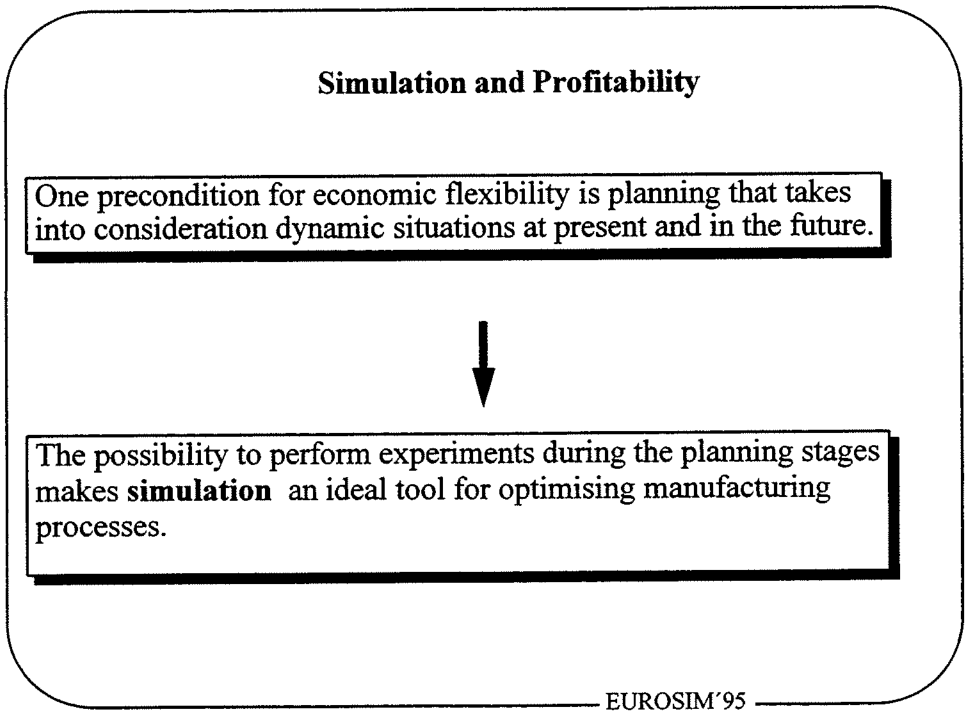
The aim of all manufacturing processes should be the profitability of the entire system.

The profitability can be achieved by different aims:

- minimum lead time
- maximum production facilities utilization
- minimum work-in-progress and stock
- minimum energy consumption
- etc.

These criteria may be emphasised differently for individual processes in the system.

EUROSIM'95



Simulation and Profitability

One precondition for economic flexibility is planning that takes into consideration dynamic situations at present and in the future.

$\downarrow$

The possibility to perform experiments during the planning stages makes **simulation** an ideal tool for optimising manufacturing processes.

EUROSIM'95

The Necessity of Adapting a Manufacturing System to Changing Conditions

A manufacturing system has to be adapted to changing conditions which are permanent, short-term and a long-term.

New conditions arise through, e.g.:

- increasing variety of products
- shorter cycles of innovation
- personnel costs
- changes of
  - working time
  - energy costs
  - environmental regulations
  - rates of exchange
  - etc.

The effects of these changing conditions on the system can only be assessed completely by **simulation**.

EUROSIM'95

Significant Changes V 2.3 -> V 3.0

- **new interpreter -> increase of speed and performance**
- **better memory management (lists)**
- **event debugger**
- **profiler**
- **modelling with SimTALK commands**
- **battery-powered vehicles with charge-monitoring**
- **front/rear-control of material flow elements**

EUROSIM'95

Unseld + Partner

Business- and Marketing Consulting Simulation!

What we can offer our customers !

Unseld + Partner :

- focuses on simulation topics only and provides efficient industrial simulation experience, gained in many industrial projects
- consists of a group of highly educated simulation experts representing the highest concentration of such know-how in Austria
- offers simulation expertise in industrial projects, while being receptive for sophisticated new strategies in freight centres and intercompany logistics, tackling especially multimodal aspects.
- provides advice on EU-programs (concerning railways, IT and Telematics) to leading members of Austrian industrial organisations and government bodies
- is a completely independent Austrian company
- has co-operation contracts and contacts with many prominent international research and university institutes

Of course state-of-the-art simulation software technology for instance SIMPLE++ and VISIO are used and professional project management skills are provided.

➔ Without Simulation no Innovation ←

Unseld + Partner Business- and Marketing Consulting Simulation!
A-1080 Vienna Lerchenfelderstraße 44/9 phone +43-1-4030371-0* fax +43-1-4030371-90