3. THE SIMMEK SYSTEM

3.1 A summary introduction

SIMMEK is a tool for performing analysis of manufacturing systems. It was originally designed and implemented bearing in mind that it was to be used as a typical decision support tool in strategic matters concerning layout and machine capacities as well as automation in manufacturing systems. It was also intended to mainly be used in job shop type of manufacturing systems. Further development and certain facilities make it suitable for FMS as well as production line manufacturing. It handles modelling of "one of a kind" production as well as large series or batch production. The major change in the manufacturing management approach of SIMMEK has been that it is now more a decision support tool in production management than in factory planning.

SIMMEK is based on discrete event simulation, and the modelling environment is object oriented. The object oriented models are transformed by an object linker into data structures executable by a discrete event oriented simulation kernel.

The user friendly graphical modelling environment makes it possible for end users to build models in a quick and reliable way, using terms from manufacturing. Various tests and check of model logic are helpful functions when testing validity of the models. Integration with software packages with business graphics and statistical functions, is convenient in the result presentation phase.

The SIMMEK system also contains functions for performing investment and costs analysis. This means that it is possible to perform economical evaluation of different models.

SIMMEK consists of a number of modules described more in detail in later sections. Except from the main part of the result presentation module they are all developed at SINTEF Production Engineering as a part of the SIMMEK Research Programme. The reason why so much was developed from scratch, is also explained in later sections.

To summarise, there are some major points that make SIMMEK different from, and in certain areas superior to, other simulation systems for manufacturing.

*	A tool suitable for both decision support in strategic, tactical and operational production management
*	Includes economical analysis
*	Models and results are presented by spreadsheets and graphics, making it quick to model easy to read and interpret easy to post process the data easy to integrate
*	No programming or pseudo programming needed

Table 3.1 The SIMMEK system's major advantages

Regarding the last point, an ongoing R & D project at SINTEF has already shown that SIMMEK can be integrated with success with an MRP II based production management system.

SIMMEK has also been modified and improved to receive input from a set of Excel spreadsheets, and not only through the modelling units described later in this module. This specification and programming work have been performed by Eirik Borgen and myself, as well people from other companies, and are therefore not reported here.

3.2 Basic simulation approaches

3.2.1 The Simulation "world view" approach of SIMMEK

The Simulation Kernel in SIMMEK is based on process oriented discrete event simulation with continuous time measurements. It is discrete event oriented in the way that it has the traditional approach of using an event list to control what is happening during the simulation. The different steps of operations are executed by stepping to the next event scheduled, letting it happen, and updating the system parameters. This is described in more detail in Section 3.5. The modelling environment is object oriented as well as process oriented. This means that the life of an entity object is a process. An entity object is of course a type of product or part.

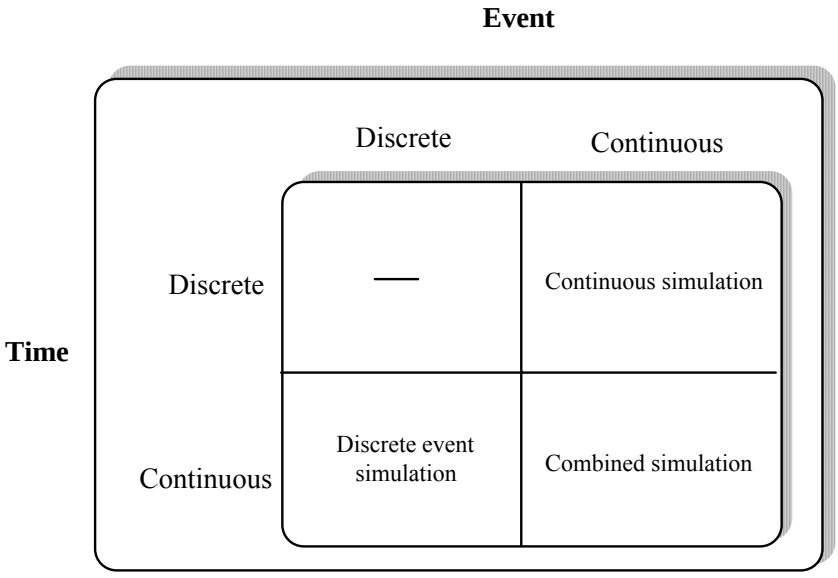

Figure 3.1 Time measurement in simulation

A discrete event approach has been chosen because of the nature of the manufacturing process. A discrete manufacturing system handles single parts or batches, batches which again consist of a discrete number of parts. The different states that such a system may be in are also discrete; a machine is idle or not, the number of jobs in a queue is 1,2,3,..n, the number of finished parts is 1,2,3,..n, etc.. What is not discrete in this type of manufacturing systems, is the time measurement. At a certain instant of time a job may be 71,3 % finished, there may be 3,45 minutes left of it. This is handled by the continuous time measurement of the simulation system.

For the sake of completeness it must be mentioned that time measurement in a discrete event simulation system is continuous only down to the maximum number of digits behind comma that the system allows. This limit has no practical influence; with three digits behind comma, and time units in seconds, the accuracy demands are taken well care of.

A manufacturing system may also be described as being made up of a number of objects of different types; machines, humans, products, components, etc. This is the reason why an object oriented modelling environment is chosen for SIMMEK.

Thirdly, the process orientation comes from the description of the manufacturing operations that creates the products. This information is found - and the operations are performed - as a linked list of operations. The operations must be performed on the same part/batch, and in the right sequence. This sequencing is reflected in the process oriented modelling environment of SIMMEK.

3.2.2 The SIMULA and Demos impact on SIMMEK

It must be pointed out that the Demos class of SIMULA was our first experience with discrete event simulation. Demos, Discrete Event Modelling on SIMULA, is a class in the object oriented, general purpose, programming language SIMULA, designed for doing simulation experiments [3]. SIMULA is now available on a large range of computers.

The brilliancy in SIMULA which is extensively used by the class DEMOS, is the class concept itself. This concept with its inheriting facilities allows the possibility of defining a class resource with a certain set of parameters and abilities. As subclasses of this resource may be distinguished between the consumable and non-consumable resources with additional, different abilities. These subclasses may again be divided into subclasses, for instance humans, machines, work stations and so on.

This is exactly what is done in class DEMOS, and this class may be used by a reference to this external definition, and every class may have extended and different subdivisions in every application. This class hierarchy is extremely useful in modelling for simulation. There is also, of course, a number of in built simulation functions available, functions like acquire, release, wait, etc.

In implementing the Simulation Kernel of SIMMEK we have used much of the ideas from this class Demos in SIMULA, the full concept of which is described in [3]. Kreutzer’s book [1] also gives a good description of different styles in simulation programming.

In fact plans existed at the early stages of the project to develop a SIMULA/Demos program code generator, see also next section. Through a modelling environment the user should specify the objects and entities of the different types, and a generator should transform this into SIMULA code. But this solution did not give the necessary flexibility in modelling. Changes in Demos concerning the reporting functions were also needed, and it is never a good solution to change program code in an existing system. Another reason was that the SIMULA version on Macintosh was not available at the time we started implementing the system. It was already decided at that time that Macintosh, for its unique user-friendly interface, was going to be the hardware platform for the SIMMEK tool.

But the concept of defining all resources in one "class", with "subclasses" of operators, machines, stores, etc., is certainly used in SIMMEK (see Section 3.4).

3.2.3 Data driven activity oriented simulation

In 1987-88 we became aware of some research work being performed at the Technical University of Denmark, Copenhagen. This was done by Kent Fisker as a part of his Ph.D. work. His work was very generously made available to us, and for some months we had a close co-operation.

We tried to interface the modelling facilities of SIMMEK to the activity based, data driven simulator called ASIST [14], developed by Fisker. Again we ran into problems with keeping the freedom in the modelling phase. After some tests with ASIST as the simulation kernel the integration work was stopped. The design and implementation work on this interface really made the distinctions between the world views clear to us. We had full access to the source code of ASIST and freedom to change it if needed. What stopped the work was the difference between a process oriented modeler and an activity oriented kernel.

The concept of ASIST as being data driven was no problem. As we understand this concept, SIMMEK is also data driven. In ASIST a model was made up from a number of activities with conditions for start and finish. If two different activities concern the same product, this connection is established in the conditions of the "second" of these activities. This is done by saying that the "first" activity must be complete before the "second" can start.

For our intended use of SIMMEK as a tool in production management the activity orientation of ASIST created some not solvable problems. Two important ones were the lacking reporting facilities in ASIST concerning production orders. In an activity oriented simulation system it is very difficult to track a production order through the simulation. Another problem was varying lot sizes through the manufacturing process.

A few more comments concerning the differences between the world views are given in the next section.

3.2.4 Process, event, and activity oriented simulation

The modelling environment in SIMMEK is object oriented as well as process oriented. The object orientation is more concerning the computer implementation of the modelling environment, or perhaps more correct, how the user is operating the computer program.

The Simulation Kernel can also be said to be event oriented though. We find, which may be a surprise, no contradictions between a process oriented modeler and an event oriented Simulation Kernel. A process is taken to be "what happens, in sequence, to one occurrence of one type of product". In a manufacturing context this is the sequence of operations of a part or a product. One occurrence may be one single item, a batch, or a full order, and is in either case an entity in the model. A model will consist of a number of processes, one for each type of product or part, i.e., one for each type of entity object.

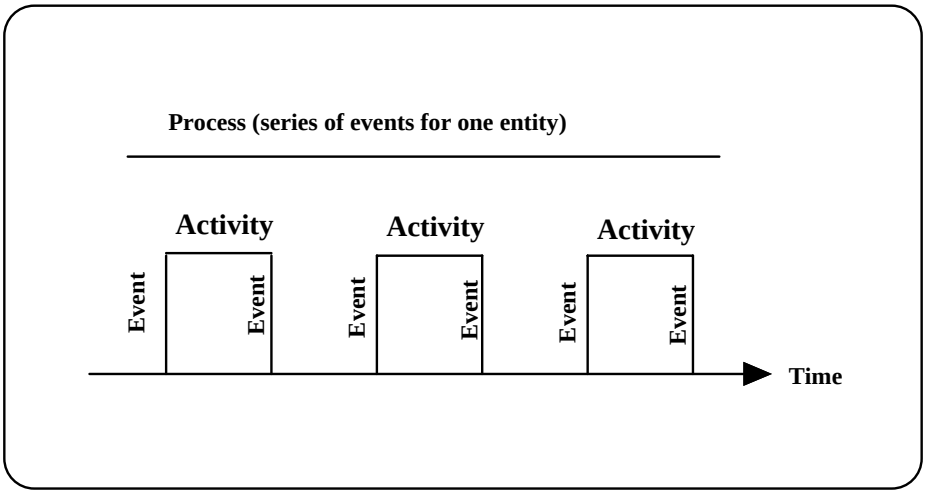

Figure 3.2 Process, activity and event oriented world views

The process oriented model is transformed, with no loss of generality, to the event oriented Simulation Kernel. The sequences of steps in the processes are transformed to sequences of events for each process. These sequences of events are nested into an event list. There is always one event from each process in the event list (see Section 3.5). This can be done since a process of course is a sequence of manufacturing operations, which again is a sequence of events like starting an operation, finishing an operation, asking for a resource, etc.

By stating this we also state that the real distinction in "world views" of simulation is between activity oriented on one hand, and process/event oriented on the other. To go from process orientation to event orientation is just a simple splitting and combining job. Different processes or events may refer to the same resources. The simulation kernel in process oriented model must link the events from one process together. This is easily done in a computer program by use of pointers. In a truly event oriented simulation this link is not present. The connection between events (events that concerns the same occurrence of a product), is given as conditions for performing the event.

In activity oriented modelling and simulation the sequence of the manufacturing operations of one occurrence of a product is given as conditions for start and finish of an activity. It is difficult to keep track of which activity belongs to which product, and the sequence between the activities.

Our conclusion is that the process oriented "world view" is to prefer when modelling and simulating discrete manufacturing systems. It is models based on this view that in the most direct way accept information from the Bill of Operations database. And in these models it is easier to extract results that concerns the products.

3.2.5 Object orientation

There is a true object orientation in the modelling environment of SIMMEK. The objects may be of two main categories; the products already mentioned, and the resources. The products routing, in SIMMEK called the process plans, together with the order data, are taken to be the simulation processes, see 3.2.1. Occurrences of products and parts are born and die during the simulation run. The resources, however, are permanent. The processes of the products are sequences of references to which resources they require, and for how long they need to possess them. The object orientation of the modelling environment is reflected in the internal structure of the Simulation Kernel, Section 3.5.

3.2.6 Use of an existing simulation language as a kernel in SIMMEK

It was also seriously considered to interface the modelling environment, which was developed first, to other existing simulation languages like SIMULA/DEMOS, SIMAN or SLAM II. The linker would then be more or less like a code generator of SIMAN/ SLAM II/SIMULA/DEMOS code. This idea was left for many of these same reasons that we left the "Demos-kernel" idea (Section 3.2.2). The cost of developing this code generator would be just as high as developing the kernel, and developing the kernel ourselves gave us access to source code of the kernel, and now makes it possible to improve and extend the modelling facilities.

3.3 The modelling features of the SIMMEK system

The SIMMEK system was originally designed to consist of seven main modules; two modelling modules, a linker, a simulation kernel, a result presentation module, and two modification modules. In addition to this there is a data store of models and results. In the current version the two modification modules are not present. Sometime in the future there will also be modules to perform the integration with other computer systems, see Sections 7 and 9.

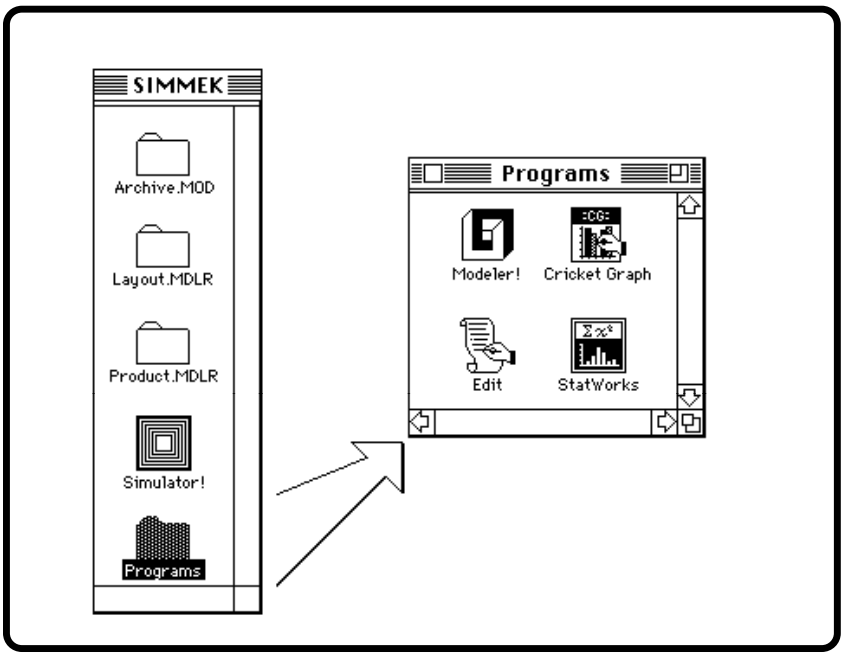

Figure 3.3 The main screen picture and the program modules of SIMMEK

It must be pointed out that this is a logical description of the different system modules. For the user some of these modules seem to be environments with model building functions and blocks. Others are more straightforward applications that the user explicitly runs. More than one module may be represented by one application. This is the case for the Linker and the Simulation Kernel. Figure 3.3 shows how the system is represented on the screen.

The "Modeler" application in the "Programs" archive needs some explanation. It is this application that controls much of the user interaction in the two modelers.

There are two separate modelling modules. They are called The Layout Modeler and The Product Flow Modeler. Both of them are operated through the Modeler application. The difference between them is that they operate on different archives and hence types of objects. This distinction is also reflected in the models. A simulation model consists of a layout model

and a product flow model, with resource objects in the layout model, and product and part objects in the product flow model.

Another way of describing the overall structure of the SIMMEK system is to show the hierarchy of archives that makes up the system and its models, Figure 3.4.

A third way is a figure showing how the different modules are linked, and how they work together on the models, here called the data store, Figure 3.5.

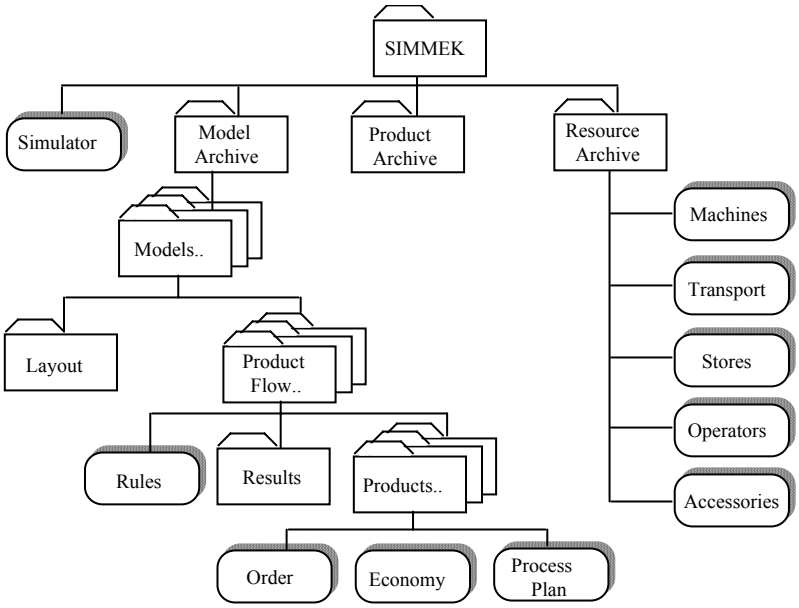

Figure 3.4 The hierarchy of archives of SIMMEK

3.3.1 The Layout Modeler

The Layout Modeler is the part of the modelling environment used to create and describe the resource objects. There are two main types of resources; those that may be consumable and those that may not. Most of the resource types; machines, transport units, stores and operators, are non-consumable. That does not mean that they are all-time present during a simulation run. They may be "down"/not available for some scheduled or unscheduled period. Such down time for a machine is typically when the machine breaks down, or when it is stopped for maintenance. For an operator this time is simply when he is not working.

Another type of resource objects is called requisites. Resources of this type may be defined to be consumable. If they are consumable, they may be refilled or supplied by certain ordering procedures.

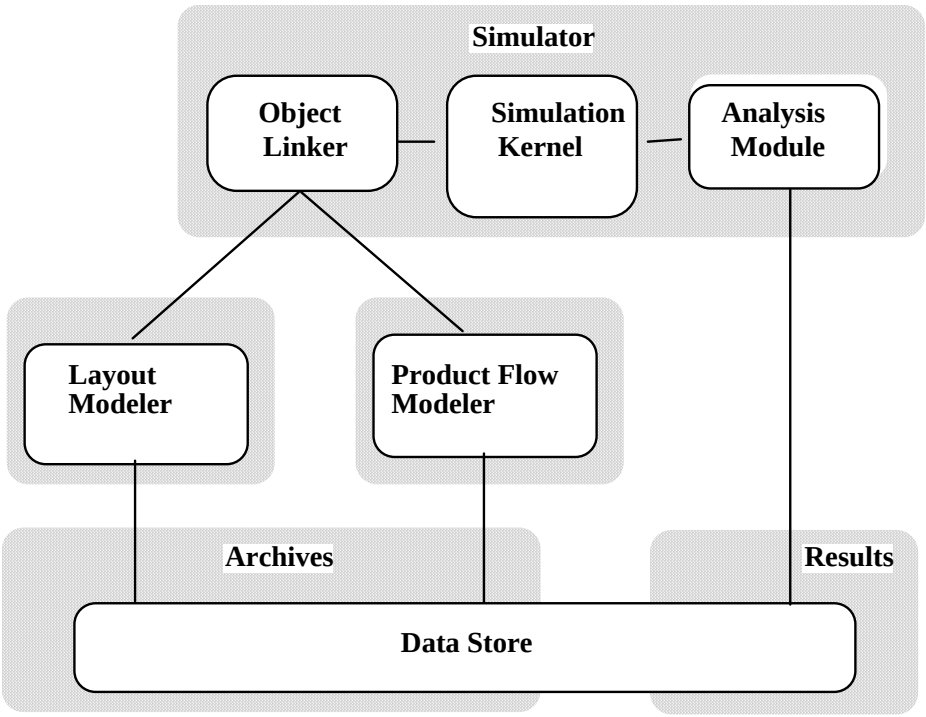

Figure 3.5 The linkage of the different modules in SIMMEK

By secondary we mean that an operation can not take place on such a resource alone. An operation must take place on either a machine/station, a transport unit or in a store. It follows from this that the operator resources are also secondary resources.

To show the features of the system it is necessary to go into the details of the modelling environment. In Figure 3.6 Resource type; Machine - page one, the input screens for modelling a machine resource are shown. In Table 3.2 Resource type; Machine - attributes, all the attributes available for a resource of type machine are explained.

For the other types of resources; requisites, stores, transport units and operators, the screen pictures are not shown, but Tables 3.3 to 3.6 explain all the attributes of these resource types.

3.3.1.1 The machine resource type

This section shows in Figure 3.6 a picture of the screen where input concerning a machine resource is given. This is an example of how the working environment is when the user is in the input phase. Similar screens are available for all types of resources and products.

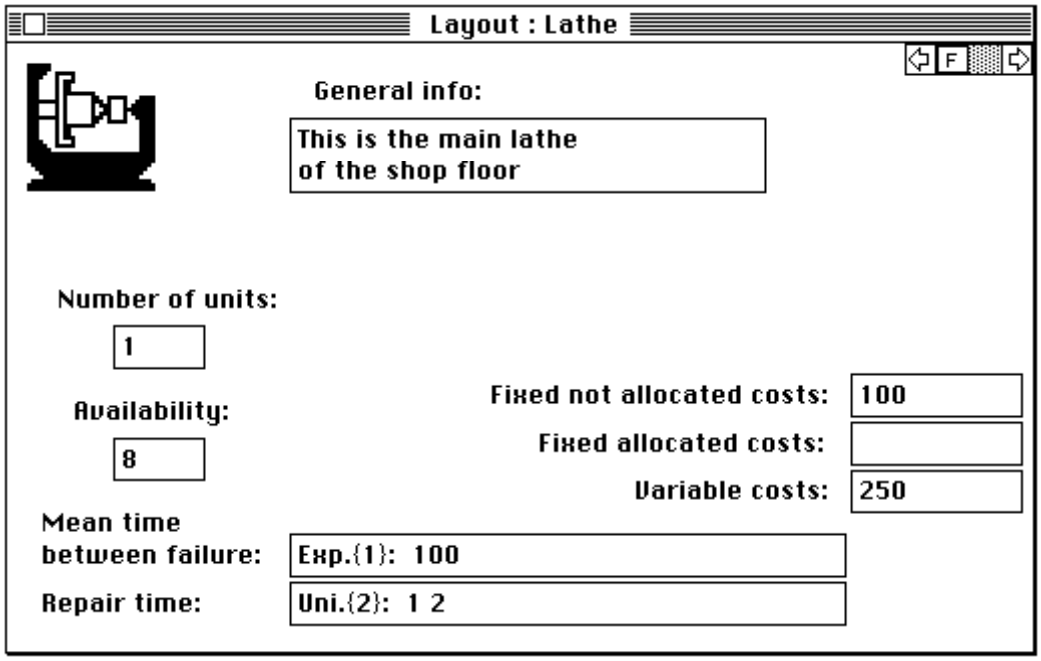

Figure 3.6 Resource type; Machine - page one

Table 3.2 lists and explains all the input parameters of the machine resource. The screen is shown in Figure 3.6.

Name	*Unit*	*Range*	*Description*
Number of units	#	integer > 1	Indicates how many similar machines there are of this type. These machines can be seen as a common resource
Availability	Time units	blank field or decimal number > 0	Indicates how many time segments the resource is available each day. The length of the day is given in "Rules". Maximum availability is obtained when the option is left blank or when the number given is bigger than the length of the day
Fixed, not allocated costs	Currency	blank field or decimal number > 0	Indicates a fixed hourly rate for the resource. This cost is not distributed at each product, but is charged to the system as a whole
Fixed allocated costs	Currency	blank field or decimal number > 0	Indicates a fixed hourly rate for the resource. This cost is distributed at each product that uses the resource. When the resource is idle, the cost is charged to the system as a whole
Variable costs	Currency	blank field or decimal number > 0	Indicates an hourly rate on use of the resource. This cost is distributed at each product that uses the resource, in a matter of the time each use the resource
Time to failure	Time units	blank field or decimal number > 0	Indicates the length in time between each time a failure occurs. This is given as a statistical distribution

Time to repair	Time units	blank field or decimal number > 0	Indicates the time required to repair the resource. This is given as a statistical distribution
Differentiated set-up times	Time units	blank field or decimal number between 0 and 100	Indicates the most probable times for set-up of the machine between different types of products for different families or within a family. NB! The times given here overrule any set-up times given in the process plans
Uncertainty	%	blank field or decimal number between 0 and 100	Indicates the distribution around the given set-up times as a percentage value of the times. NB! The set-up times are regarded as a triangle distribution where the times give the expected value and uncertainty in % gives the minimum and maximum values

Table 3.2 Resource type; Machine - attributes

For the other types of resources; requisites, stores, transport units and operators, the screen pictures are not shown, but Tables 3.3 to 3.6 explain all the attributes of these resource types.

3.3.1.2 The operator resource type

The Operator type resource is different from the other resource types in that it can only be used as a secondary resource.

Name	*Unit*	*Range*	*Description*
Total number of operators	#	integer > 0	Indicates how many operators this resource consists of
Cost per operator	Currency	blank field or decimal number > 0	Indicates the hourly wage for each operator
The number of operators per shift	#	blank field or integer > 1	Indicates how many operators this resource has per shift. This is given as a row of numbers. The first number is the number of operators at shift 1, the second number is the number of operators at shift 2 and so on
Working shift arrangement	Time units	blank field or integer > 0	Indicates the length of the different shifts in segments of time. This is given as a row of numbers. The first number is the duration of shift number 1, the second number is the duration of shift number 2 and so on

Table 3.3 Resource type; Operators - attributes

3.3.1.3 The requisites resource type

The parameters for the requisites resource type are slightly different from that of the machine resource type.

Name	*Unit*	*Range*	*Description*
Number of units	#	integer > 1	Indicates how many units of the requisite there is available
Consumption?		Yes or No	Indicates if the requisite can be reused, or if it is consumed. This decides which of the following parameters are used
Price per unit	Currency	blank field or decimal number > 0	Indicates how much each unit costs
Smallest inventory	#	blank field or integer > 0	Indicates smallest inventory. If the inventory goes below this level, a new order for the requisite is executed based on "Ordering time". If smallest inventory is given as zero, then the ordering time is used as a rate of arrival, independently of the smallest inventory
Number per order	#	blank field or integer > 0	Indicates the number of requisites to be delivered per order
Ordering time	Time units	blank field or decimal number > 0	Indicates the time needed from an order is given to the ordered requisites arrive. This is given as a statistical distribution
Fixed, not allocated costs	Currency	blank field or decimal number > 0	Indicates a fixed hourly rate for the resource. This cost is not distributed at each product, but is charged to the system as a whole
Fixed allocated costs	Currency	blank field of decimal number > 0	Indicates a fixed hourly rate for the resource. This cost is distributed at each product that uses the resource. When the resource is idle, the cost is charged to the system as a whole
Variable costs	Currency	blank field or decimal number > 0	Indicates an hourly rate on use of the resource. This cost is distributed at each product that uses the resource, in a matter of the time each uses the resource

Table 3.4 Resource type; Requisites - attributes

3.3.1.4 The store/buffer resource type

The third type of resource is the store/buffer resource, with the following parameters.

Name	*Unit*	*Range*	*Description*
Capacity	#	integer > 1	Indicates the size of the storage facility. Which means how many orders the storage facility can contain

Fixed, not allocated costs	Currency	blank field or decimal number > 0	Indicates a fixed hourly rate for the resource. This cost is not distributed at each product, but is charged to the system as a whole
Fixed allocated costs	Currency	blank field or decimal number > 0	Indicates a fixed hourly rate for the resource. This cost is distributed at each product that uses the resource. When the resource is idle, the cost is charged to the system as a whole
Variable costs	Currency	blank field or decimal number > 0	Indicates an hourly rate on use of the resource. This cost is distributed at each product that uses the resource, in a matter of the time each uses the resource
Availability	Time units	blank field or decimal number > 0	Indicates how many time segments the resource is available each day. The length of the day is given in "Rules". Maximum availability is obtained when the option is left blank or when the number given is bigger than the size of the day

Table 3.5 Resource type; Stores - attributes

3.3.1.5 The transport resource type

Different types of transport units may be modelled by using the common transport type resource with these parameters.

Name	*Unit*	*Range*	*Description*
Number of units	#	integer > 1	Indicates how many units the transport resource consists of. An assembly line would for instance have no limit (always have a unit spare), while an AGV-system would have a limited number
Availability	Time units	blank field or decimal number > 0	Indicates how many time segments the resource is available each day. The length of the day is given in "Rules". Maximum availability is obtained when the option is left blank or when the number given is bigger than the size of the day
Fixed, not allocated costs	Currency	blank field or decimal number > 0	Indicates a fixed hourly rate for the resource. This cost is not distributed at each product, but is charged to the system as a whole
Fixed allocated costs	Currency	blank field or decimal number > 0	Indicates a fixed hourly rate for the resource. This cost is distributed at each product that uses the resource. When the resource is idle, the cost is charged to the system as a whole
Variable costs	Currency	blank field or decimal number > 0	Indicates an hourly rate on use of the resource. This cost is distributed at each product that uses the resource, in a percentage of the time each uses the resource
Time to failure	Time units	blank field or decimal number > 0	Indicates the time between each time a failure occur. This is given as a statistical distribution

Time to repair	Time units	blank field or decimal number > 0	Indicates the time required to repair the resource. This is given as a statistical distribution

Table 3.6 Resource type; Transport units - attributes

This completes the presentation of the different resource types and their parameters.

3.3.2 The Product Flow Modeler

The Product Flow Modeler is used to model the parts and products, also called the entities of the system. Each type of part or product has its own symbol, which again consists of three main elements; order data, cost data, and a process plan.

The order data gives the number of work pieces in each order, the start-up times of orders of this type, the arrival rate of orders of this type, etc.

The process plan is really the key information, because it is used to describe the product's routing through the layout. This is a process plan in the manufacturing context, and must be distinguished from a simulation process. The description of one product's routing, i.e., its operations and operation times, *together* with the order data, is a process in the simulation context. The process plan contains references to all resources where operations on this product are going to take place. This includes all transport to be used, and all the buffers and stores the product is planned to occupy for some period. If operators are modelled as separate resources, i.e., not permanently attached to machines, they are referred to as secondary resources. This is also true for other resources like pallets, which may of course be reserved for several operations without being released between each operation.

The third part, the cost data, is a description of the different cost values connected to the part or product. The raw material costs may be given as well as purchasing costs. It is also possible to model the expected sale of the products along with the sales price.

The Product Flow Modeler is also used to specify which rules the model should be operated by. This includes which priority rules the jobs are to be selected by at the resources. Examples of selection rules are FIFO or LIFO, shortest or longest expected operation time, shortest time to scheduled delivery, scheduling by identification of critical resource, etc.

The input is given in three areas;

*	Plant operation rules	(3.3.2.1)
*	Product data (order, costs, process plans)	(3.3.2.2 - 4)
*	Experiment data	(3.3.2.5)

In Tables 3.7 to 3.12, the attributes available for modelling products and their cost data, order data and process plans are shown. In Table 3.13 the attributes for the entire experiment are shown.

3.3.2.1 Plant operation rules

For the plant it is necessary to identify the operation rules that are applied in the plant. These may of course be changed from experiment to experiment, but according to this input format.

Name	*Numbers*	*Description*
Rules of priority for the system	0 - 16	Indicates the conditions which decide where in the queue in front of the different resources the production orders are to be fitted in. This is stated by a rule-number
Rules of priority for certain resources	0 - 16	Indicates the conditions which decide where the production order is to be fitted into the queue in front of specific resources. This is stated by a rule-number and name from the layout

Table 3.7 Plant operation rules - part one

The optional priority rules that may be applied are;

0	highest priority
1	first in
2	last in
3	earliest arrival
4	highest raw material value
5	highest value of the manufactured product
6	highest sales value
7	highest real costs
8	longest expected operation's time in a resource
9	shortest expected operation's time in a resource
10	longest remaining operation's time
11	shortest remaining operation's time
12	most remaining operations
13	fewest remaining operations
14	least remaining time till delivery
15	least slack before delivery
16	least slack per remaining operation

Table 3.8 Priority rules

As examples two of these rules are explained. Rule 1 says the jobs are scheduled by the first in first out, FIFO, principle. Rule 6 means that the jobs are placed in the queue according to their sales value. The most expensive is

placed first, etc. When a job arrives, it may be placed any place in the queue, but will not affect a job that is already started.

The rest of the operation rules are the following;

Name	*Unit*	*Range*	*Description*
Method for calculating the value of the finished goods		A, B or C	This decides which one of three methods for calculating the value of finished goods is to be used. This it used for calculation of Work in production (WIP) A: Stated value of finished product B: Accumulated simulated costs C: Accumulated costs of raw materials only
Rate of interest	%	blank field or decimal number between 0 and 100	Indicates the interest per year on the capital tied up in WIP
Depreciation	Currency	blank field or decimal number > 0	Indicates the annual depreciation of the production equipment
The number of working days per year	Days	integer between 1 and 365	Indicates the number of working days per year. This is used to calculate the value of WIP and deprecations
A day converted into time segments	Time units	decimal number > 0	Indicates the number of time segments which represents a working day. This value is used to find the number of days which the interest is running in order to calculate the value of WIP and deprecations
An hour converted into time segments	Time units	decimal number > 0	Indicates the number of time segments which represents an hour. This value is used in the calculation of the resource cost which is to be divided at the different products

Table 3.9 Plant operation rules - part two

These rules, although named plant operation rules, are mainly concerning how costs should be calculated in the model, as well as some “calendar” information.

3.3.2.2 Order data

This is one of the three input formats concerning one type of product or part, namely the order data.

Name	*Unit*	*Range*	*Description*
Family		blank field or A, B, C, D or E	Indicates which family this type of product belongs to. This is used to choose set-up time in a machine when differentiated set-up times between families are given in the machine resource
Priority	#	integer between 0 and 99	Indicates the rank in comparison to other orders. Priority is given as a number, and highest number gives the highest rank
Time of start	Time units	decimal number ≥ 0	Indicates the time from the simulation starts to the first production order of this type arrives at the workshop. Given in time segments
Delivery time	Time units	blank field or integer ≥ 0	Indicates the time a production order has disposal of before it has to be finished
The number of production orders	#	integer ≥ 0	Indicates how many production orders of this type which are to be executed
The number of products in a production order	#	one or more integers ≥ 1	Indicates how many products of this type the production order contain. This can be given as a statistical distribution
The number of production orders per arrival	#	one ore more integers ≥ 1	Indicates how many production orders which arrive at the same time
Time between arrivals	Time units	blank field or one or more decimal numbers	Indicates how often production orders of this type arrive at the workshop. This is given as a statistical distribution
Conditions of arrival		blank field or Q: or I: followed by resource name and by one of the signs >/</= followed by integer > 0	Show the conditions which have to be fulfilled before the start of a production order can take place. The conditions can be queue size, Q, in front of resources or inventory, I, of products
Process plan	Time units	one or more integers ≥ 1 or statistical distribution	Can be used instead of rate of arrival. The plan is given as a row of numbers consisting of the times of arrival of the different production orders. The plan is repeated automatically

Table 3.10 Order data

3.3.2.3 Cost data

The second part concerns cost data for this product or part type.

Name	*Unit*	*Range*	*Description*
Initial cost	Currency	blank field or decimal number > 0	Indicates the price the firm has to pay for the raw materials of the products
Other costs	Currency	blank field or decimal number > 0	Indicates the costs outside the initial cost which the firm has to invest in order that the raw materials are available at the point in production where the simulation starts
Value of finished products	Currency	blank field or decimal number > 0	Indicates a fixed value which one can use instead of the calculated value of finished products. Calculated value of finished products is the sum of the initial cost, other costs and the increase in the value of the products in the production order due to the machining
Sales value	Currency	blank field or decimal number > 0	Indicates the value which the firm fix as the sales value for the product
Cost data per number of products	Currency	integer ≥ 1	Indicates the number of products which the given cost data concerns
Time between sales	Time units	blank field or one or more decimal numbers > 0	Indicates the average time between each sale. Products which are sold reduce the inventory of finished products. This is given as a statistical distribution
Sales volume	#	blank field or one or more decimal numbers > 0	Indicates how many products are sold each time a sale takes place
First sale	Time units	decimal number > 0	Indicates when the first sale is to take place. The later sales will be based on drawings from the distribution given in “Time between sales”
Start inventory	#	integer ≥ 1	Indicates how many finished products there are in stock when the simulation starts. These can be sold or used in assemblies. The inventory influence on the calculation of costs of inventory of finished products, WIP, sale and assembly

Table 3.11 Cost data

3.3.2.4 Process plans

A process plan consists of a sequential number of process steps for one product or part type. In Figure 3.7, the smallest window shows the first page of such a process plan. Each item in the list at the right hand represents one process step given by a sequential number and the name of a resource from the layout. Each process step refers to a machine, a store or a transport resource. The biggest window shows input parameters for one particular process step. Process step parameters are shown in Table 3.12. The process steps are set-up by commands from a Process Plan-menu.

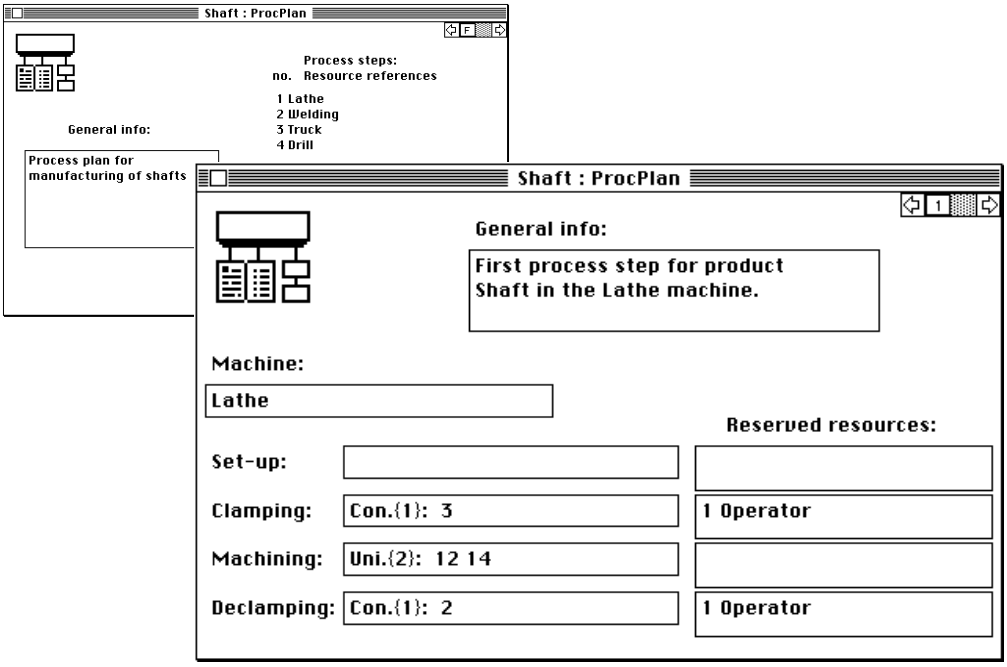

Figure 3.7 Process steps of a product type

Table 3.12 lists and explains all the input parameters of a process step. By unit; reference is meant that the value of the parameters is a reference by name to one of the other objects (products or resources) in the model.

Name	*Unit*	*Range*	*Description*
Machine	Reference	All machine resources in model	The identification of the machine this process step is to take place
General Info	Text	-	Any remarks about this step, the data is not processed
Set-up time	Time units	blank field or decimal number > 0	Indicates the time to set-up a machine before processing can start. This is given as a statistical distribution

Clamping time	Time units	blank field or decimal number > 0	Indicates the time to fix a work piece into a machine before machining can start. The time is specified for the whole production order or for every work piece. This can be given as a statistical distribution
Machining time	Time units	blank field or decimal number > 0	Indicates the time to machine a work piece in a machine. The time is specified for the whole production order or for every work piece. This can be given as a statistical distribution
De-clamping time	Time units	blank field or decimal number > 0	Indicates the time to detach the finished work piece from the machine. The time is specified for the whole production order or for every work piece. This can be given as a statistical distribution
Resource for Set-up	Reference	blank field or integer ≥ 1 followed by the name of the resource from the resource overview	Indicates the other resources that are needed to complete the readjustment. This is given as a number of units followed by the resource name. Several resources are tied together with "+" Example: 1 operator + 2 assembler
Resource for clamping	Reference	blank field or integer ≥ 1 followed by the name of the resource from the resource overview	Indicates the other resources that are needed to complete the fixing. This is given as a number of units followed by the resource name. Several resources are tied together with "+" Example: 1 operator + 2 assembler
Resource for machining	Reference	blank field or integer ≥ 1 followed by the name of the resource from the resource overview	Indicates the other resources that are needed to complete the machining. This is given as a number of units followed by the resource name. Several resources are tied together with "+" Example: 1 operator + 2 assembler
Resource for de-clamping	Reference	blank field or integer ≥ 1 followed by the name of the resource from the resource overview	Indicates the other resources that are needed to complete the detaching. This is given as a number of units followed by the resource name. Several resources are tied together with "+" Example: 1 operator + 2 assembler
Are the times applied for the whole production order?		Yes or No	Indicates if the times for fixing, machining and detaching are applied for every work piece (N) or for the whole production order (Y)
Line production?		Yes or No	Indicates the dependence between this process step and the next process step. If line production is set yes (Y), the main resource will not be released before the next process step resource is free

Sub-assemblies	Reference	blank field or integer ≥ 1 followed by the name of the product from the model	Indicates which parts from finished production orders shall be assembled together with the workplaces from the production order in this process step. The number of units is applied for assembly of one work piece from the production order in force. This is given as a number of units followed by the name of the product. Several products are tied together with "+". Example: 1 nut + 2 bolt
Resources that are reserved for longer periods of time	Reference	blank field or integer ≥ 1 followed by the name of the product from the model	Indicates the resources that will be reserved for use in this and in all the next process steps until released
Release reserved resource	Reference	blank field or integer ≥ 1 followed by the name of the product from the model	Indicates the resources that will be released after use in this process step. Is applied for resources reserved in an earlier process step. This is given as a number of units followed by the name of the product. Several products are tied together with "+"
Percentage of rejects	%	blank field or decimal number between 0 and 100	Indicates the percentage of the number of the work pieces that statistically has to be rejected in machining with this machine
Percentage of re machining	%	blank field or decimal number between 0 and 100	Indicates the percentage of the number of the work pieces that statistically has to be machined in this machine after inspection
New batch size	#	blank field or integer ≥ 1	This is used to split the original production order in several suborders. Regrouping indicates the number or work pieces in every suborder. The rest of the work pieces will be grouped to the last suborder

Table 3.12 Process plan for a machine step

3.3.2.5 Simulation experiment data

To set-up the actual experiment, a final set of data input must be specified. These data are given below.

Name	*Unit*	*Range*	*Description*
Simulation time period	Time units	integer > 0	This is the total time period for the simulation experiment. Simulation starts at time t = 0 and ends when the time given as a simulation time is reached
Warm up time period	Time units	integer > 0	This time is for "warming up" the model. When this amount of time units is simulated, all the registering variables are zeroed. In result reports only that what has happened after warm up time will be registered. That means that the time that will be set as basis for all registering = simulation time - warm up time
Number of replications	#	integer ≥ 1	Indicates the number of times the simulation experiment will be repeated. Every repetition will use different drawings in calculating machining times, transport times, etc. Running with several replications on the same model is done to get as reliable result as possible. A simulation result should be the average value of several replications. It is not enough to regard one running as a complete result

Table 3.13 The attributes for the entire experiment

These attributes are those which are merely simulation parameters. They concern the reliability of the results. It is important that they are considered together.

There are in principle two ways of getting reliable results;

* Replications
  Repeated runs of the same model with the same data, but different seed for the statistical drawings

* Long simulation runs
  Giving the model the chance to repeat all processes several times

The warm up time is used to reach a "steady state" before the results are collected.

3.3.3 The data store - The Model Archive

All the models are stored in the Model Archive. This archive may contain a number of other archives. Each of these archives contains one layout model and one or more product flow models. A simulation model is made up of the layout model and one of the product flow models. Figure 3.8 shows an example of the contents in an archive in the Model Archive.

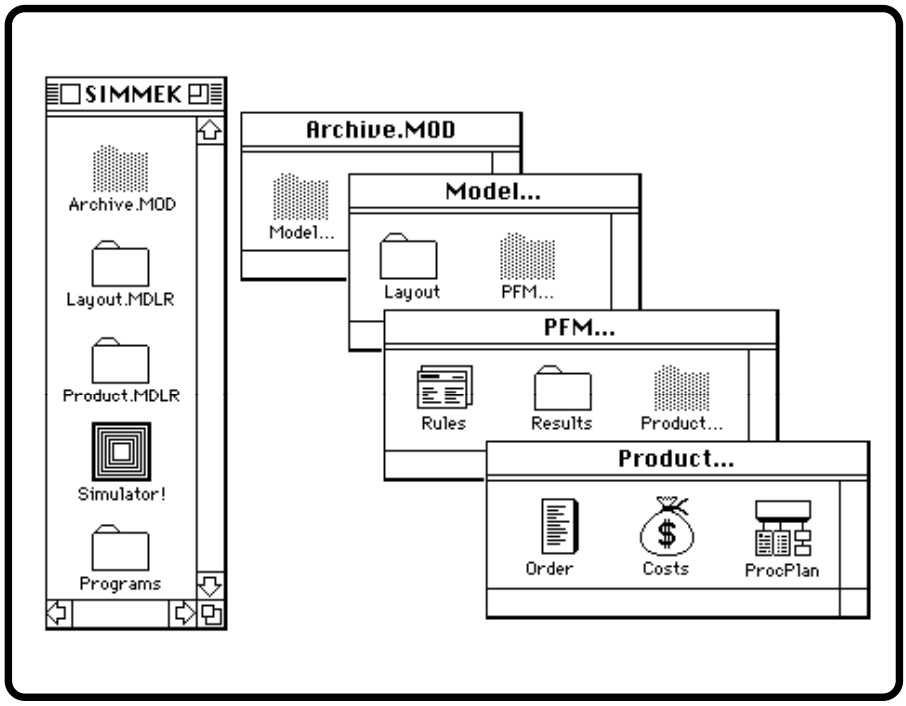

Figure 3.8 The contents of an archive in the Model Archive

3.3.4 The Object Linker

The Object Linker transforms all the information in the layout and one product flow model into information that can be processed by the Simulation Kernel, see 3.5.4 for more details. The Object Linker is a part of the "Simulator" Application. This module performs the tests required to be certain that the simulation can be run without complications. All these tests are automatically performed, and the user is not bothered with this unless any mistakes are revealed.

3.3.5 The Simulation Kernel

The Simulation Kernel is used to choose and perform the wanted number of replication runs of the model. The Simulation Kernel is also a part of the "Simulator" Application. This is the part of the system where the model "lives". After starting this module the user is asked to give the simulation experiment parameters, like warm up period, simulation time period, and number of replications. The Kernel may produce a full trace, which may be examined by Edit, and an aggregated summary report. This report, together with the trace, is used mainly for model validation.

An important part of the simulation module is the Monitor function. This function is used to display some important information during the simulation run. It is of special use in model validation (Section 4).

3.3.6 The result presentation module

The Analysis Module presents the more detailed results produced by the Simulation Kernel. The "menu of results" is found in Section 5. The amount and the degree of detail of the results are chosen from the menu. The results are presented in files by a format that is readable by two software packages; StatWorks and Cricket Graph. These packages can be used to perform statistical tests on the results, and to present the results in graphics. Cricket Graph and StatWorks are also used for presentation of the summary reports.

In the latest version of SIMMEK the format of the result files are that of Excel. Excel is the most used spreadsheet for Macintosh, and is also available on MS-DOS for IBM compatible PCs. This makes it possible to exchange the results between these two computers.

3.3.7 The two modification modules

The two Modification Modules are still not specified. They are meant to be a decision support tool giving advice about model changes in order to produce better overall results.

3.4 SIMMEK; How it is operated

In Section 1.3 we established what is considered to be the success factors for using simulation in manufacturing. Two out of five of them were dealing with the input and modelling facilities of simulation systems;

* The resemblance between the modelling facilities and the real world system being modelled

* The time an inexperienced user has to spend from the time he starts using the tool and till he has a model running

These two factors are closely related. A modelling tool where the building blocks are close to what you can see on shop floor is easy to learn to use. But also how the tool is technically operated is important when considering time needed for training.

The most important points in trying to satisfy the success factors are;

* *Symbols* that look like machines and other manufacturing resources

* Using *windows*, *pull down menus* and *graphics* in modelling

* *Adding* to and *deleting* from models by *moving* symbols

* *Referring* to symbols by *pointing* and *clicking* with mouse

* *On-line checking* of legal values

* On-line *help* from *pull down menu* and *message field*

All these are important to the two success factors mentioned, but also to model validation. In Section 4 on model validation some examples will be given.

3.4.1 Modelling

In any simulation system the modelling environment is the most crucial part. No matter how sophisticated and easy to use the rest of the system is, if it is too complex or time consuming to build models, the majority of interested users will never see the sophisticated rest.

Production planners and managers, floor managers and machine operators using this tool, should be able to recognise on the screen the layout with the machines, transports, operators, and the products and their Bill of Materials, Bill of Operations. Of course no programming should be needed. This is made possible by offering a number of symbols to represent the different manufacturing objects, each with different parameters according to the type of object. SIMMEK takes advantage of the Finder operating system on the Apple Macintosh with its windowing and graphical facilities. This means that all copying and duplication functions available on the Macintosh are available in the SIMMEK system.

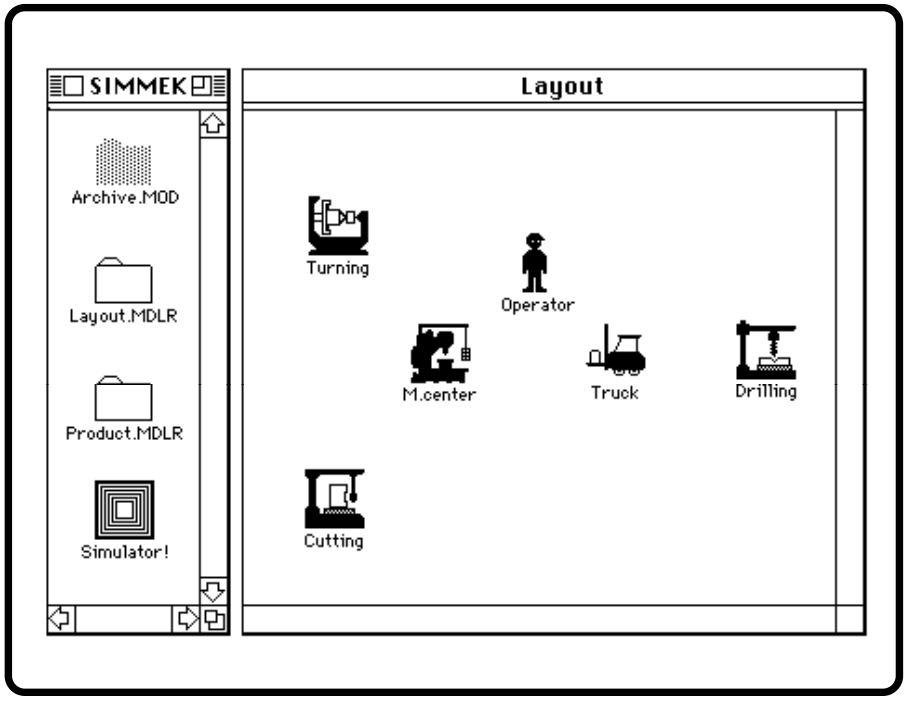

Figure 3.9 An example of a model layout

The uniformity of all applications for Macintosh, and the user-friendly interface, make the modelling very fast. Tests performed with the system have shown that after a few hours, inexperienced users are able to perform realistic experiments.

The first things to do in a simulation experiment are problem definition and establishing the objectives of the experiment. The system borders must be drawn and the real objects identified. These objects are either resources in the layout or products and components. When a resource element in the real layout is identified, and it is decided that it must be a part of the model, it must be put into the model layout. This is done very easy by pointing at a symbol of a resource element in the Layout Modeler, and copying it into the model layout. These symbols contain parameters which are typical for the chosen type of resource element. Machine objects have one set of parameters, operators have another, stores a third, etc. An example of a simple layout is shown in Figure 3.9.

We have put a lot of effort into designing the Layout and Product Flow Modelers in such a way that the modelling route is not set once and for all. The user is free to choose where to start and where to finish modelling, and is able to take a break no matter how incomplete the model is. Some simulators have the disadvantage that once started a modelling job it has to be finished without any break. This is not the case of SIMMEK.

This freedom in modelling is also important whenever you want to make changes in the model. New machines or products may be added just by duplication of the symbol of an existing object, and change the parameters that should be different.

Concerning the modelling route it is advisable though to start with modelling the layout. This is so because when modelling product flow, a lot of references to the resources in the layout is needed. If the layout is present, this is done simply by pointing and clicking the resource symbol. How this is done is shown in Figure 3.10. If not, the full name of the resource must be written from the keyboard. When identifying the resources to be used in one operation, the system automatically sets up the parameters for the operation. These parameters are of course different depending on whether it is a machining, transporting or storing operation.

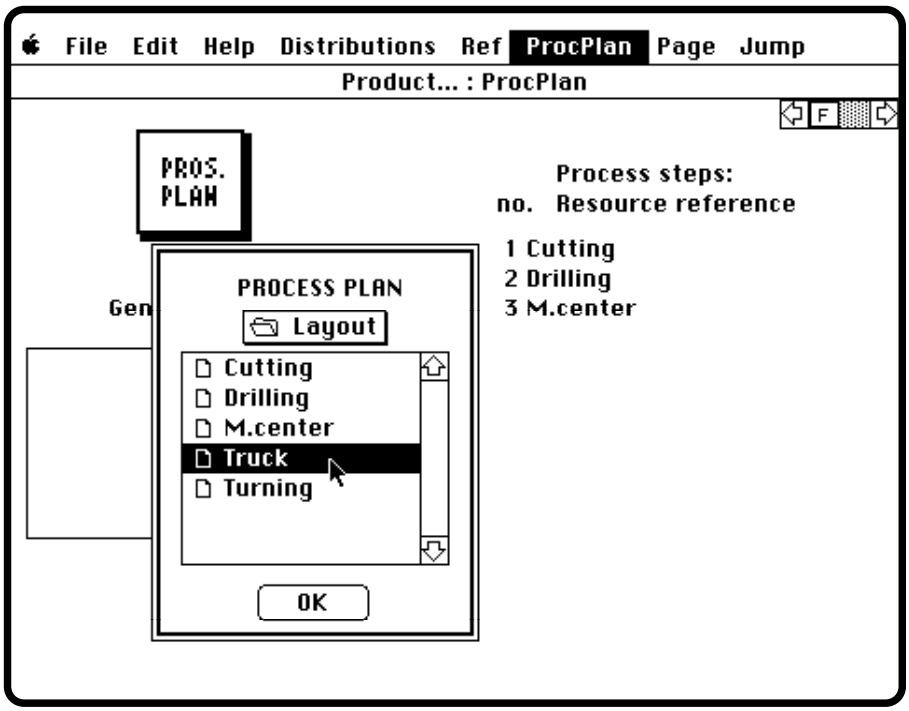

Figure 3.10 Creating the process route of one product type

3.4.2 Simulation

If the user gives values or attributes that are illegal or inconsistent, messages will be given. The parameter that is out of range is shaded, and must be corrected. Through the message field detailed information about the error is given. The entire syntax and logic of a completed model will be checked by the Linker before the simulation starts. This is automatically performed, and messages are given on the screen. The message contains information about which object, parameter and value that is incorrect.

It is possible to get aggregated schemes of the expected results. These schemes show what objects have been put into the model, the expected load of each resource, etc. This makes it easy to reveal modelling mistakes or overload of the model, and is extremely important in the validation aspect. A simple test showed that seven out of ten modelling mistakes were identified by examining these expected values.

3.4.3 Result presentation

The result reports are presented as files in Cricket Graph format, a format that is also readable by StatWorks, or in Excel format. This gives the user the possibility to manipulate and combine results from many experiments. This is done by standard copying and pasting procedures of the Macintosh. A number of mathematical and transforming functions is also available in Cricket Graph and StatWorks. In most cases these will cover all needs. It is also possible to compare the results directly with results from the real system, in order to convince everybody of the model's validity. The most important and significant results may be extracted and presented graphically, both on the screen and as hard copies, just the way the manager is used to see them.

With Excel it is possible to perform a wider range of mathematical operations. As with any other spreadsheet it is possible to put mathematical formulas into cells, columns and rows. Another possibility is the use of macros for automatic extraction of data, creating graphs and reports.

A library of company specific macros may be developed. These macros can be used for different models, and complete reports can be made by pushing one button.

There is a freedom to choose which result reports to be produced during simulation. Some of the reports will always be accessible after a simulation experiment, others may be generated on request. Section 5 will deal with the different reports already available in SIMMEK, and these can be post processed for hard copy presentation.

3.5 The internal structure of SIMMEK

3.5.1 Modelling specifications

During the specification phase, user requirements turned out to be very similar to the Macintosh concept of the user interface. Early we became quite sure that use of the windowing technique and the mouse device were the right way to go. Therefore we stated that the system should run on a Macintosh computer using all its facilities.

We wanted the different resources to be represented as graphical symbols. We specified that templates of resources should be stored in a palette, from where they could easily be picked up and placed in a window representing the job shop floor. Behind each symbol, numerical characteristics should be presented by clicking it up, modified and stored. Similar functions were specified for the product flow models.

First we started to create a new application to handle all these functions. After a while, we realised that this was a long way to go, both in time and costs. Then we looked at different database software for modelling and data storing, but we did not find any suitable for our needs.

In addition, the use of database software would introduce an extra cost element in our simulation package - a user had to buy the database software as well.

3.5.2 Use of the Macintosh concept

It stroke us that the Finder, the Macintosh operating system, itself could serve very well as a database. It had all the functions that we were looking for. The advantages of choosing the Finder as a storage medium were;

*	The Finder met our requirements of a storage medium
*	Every Macintosh has its Finder, this means no extra costs
*	New functions in the Finder could also be used on our model elements
*	Once a user became familiar with the Macintosh, he had also automatically learned the basic principles in our way to create a simulation model
*	It would reduce our time and cost expenses

Table 3.14 Reasons for choosing the Finder of Macintosh as media

In traditional computer systems, the normal way to run an application is to specify the name of the application and the name of the data file to be handled. However, the Macintosh, unlike most other computers, has a special connection between data files and application programs. A data file may be connected to a special application by specifying that the data file belongs to this application. When you click up such a data file, the Finder identifies which application the data file is connected to, and starts the application program. The application then takes appropriate actions based on information about the data file which is retrieved from the Finder. This facility is now copied by most operating systems.

This connection information is stored in a part of the data file called the resource part. The Macintosh files are divided in two; one resource part and one data part. The resource part of a file may contain different pre defined data structures and templates, the other part may contain traditional data for reading and writing by application programs.

Applications, which also are files, may contain in its resource part things like pre defined menus, alerts, dialogues, etc. The program code itself is also actually stored in the resource part.

In our modelling concept we defined the simulation resources (machines, transports, stores, etc.) as separate files. In addition we developed an application program to handle the parameter settings; i.e., the information that should be stored in the different files. All simulation resource files are connected to the application program in the earlier described Macintosh way. When the user clicks up one of our simulation resource files, the Finder identifies that the file belongs to our modelling application and then starts it. The application presents the data stored in the file on the screen, it handles data modifications and stores any changes made.

The use of pre defined structures and dialogue templates in the resource part of the simulation resource files, gave us access to all the functions defined in the Macintosh toolbox system [15]. These are among others, functions for text manipulating, pick, put, copy, paste, etc. The data stored in the simulation resource files is therefore located to the resource part of the file. Actually, the data part is not in use.

3.5.3 Model limitations

All the information is stored as text strings. When the simulator application accesses information from the files, it fetches a set of text strings from each file, converts the strings to specified formats and prepares them for input to the Simulation Kernel.

The Macintosh system software is based on the C programming language. A string defined in Macintosh C contains in the first byte its size. Therefore a C string is limited to 255 characters (the maximum size which can be expressed in one byte). Because we use standard toolbox functions [15], the content of one parameter field may not exceed 255 characters. This is normally not a problem (who specifies a number or a name greater than 255 characters?). It may cause problems in some of the parameter fields where a number of resources or product names is to be specified (i.e., in resource reservations), but we have never got into this problem yet.

As far as we know, there are not any practical limitations in the number of resources which can be placed in a layout (i.e., grouped in a folder). We have tried more than 256 without any problems (256 is a suspicious number in such tests), but there is of course limitations connected to disc space and memory size.

3.5.4 The basic input structure of the Simulation Kernel

Bringing model information from the Finder's storage medium and prepare it for input to the Simulation Kernel, is the task of the Linker. The Linker produces two separate linked lists, which we call the Resource List and the Product Type List (Figure 3.11). Both lists are single linked lists. The first one contains information about the different resources in the layout. The other one contains information about the different product types.

Linked lists, by nature, have inherent dynamic characteristics. There are no restrictions on the number of elements stored in a linked list. The chosen programming language; C, is well suited for linked list manipulation, and the lists are only limited by the amount of free memory.

Each element in the Resource List corresponds directly to a resource object (file) in the layout of the model. One list element contains data from a resource object specified by the user, run time data and a Q-list (queue-list) for holding products which request the resource during execution.

To gain maximum execution speed, a Q-list is a double linked list. This is done because of the ability to choose different rules for insertion of products in the lists. Sometimes products requesting a resource, will be placed at the end of the resources Q-list (FIFO), and sometimes the products will be placed in the start of the list (LIFO). Products can also be placed in other positions of the list depending on the chosen insertion rule. An example of such a rule is "The product with lowest machining time will be served first".

In addition to the elements corresponding to the resource objects in the model layout, each product type will also be represented as elements in the Resource List. That is, when a product is finished and "dies", it changes from being an entity (an object that moves around and requires resources) to a resource which may be required by an assembly. During the simulation, other not finished products (entities) may ask for work pieces as assembly parts from such a product resource.

Each element in the Product Type List corresponds directly to a product specification in the Product Flow Model. One list element contains data specified by the user in the Order data, Cost data and Process Plans. The last mentioned, however, is converted to an operation list. This is a single linked list where all the process steps in the Process Plan are split up in single operations in sequence. Every product will follow the execution of the operation list of its product type.

Operations will be; Hold a resource, Release a resource, Delay for a specific time, etc. All operations which refer to a resource has a pointer connection to the specific resource in the

Resource List (see Figure 3.11). The program can then quickly and easily identify the requested resources during run time.

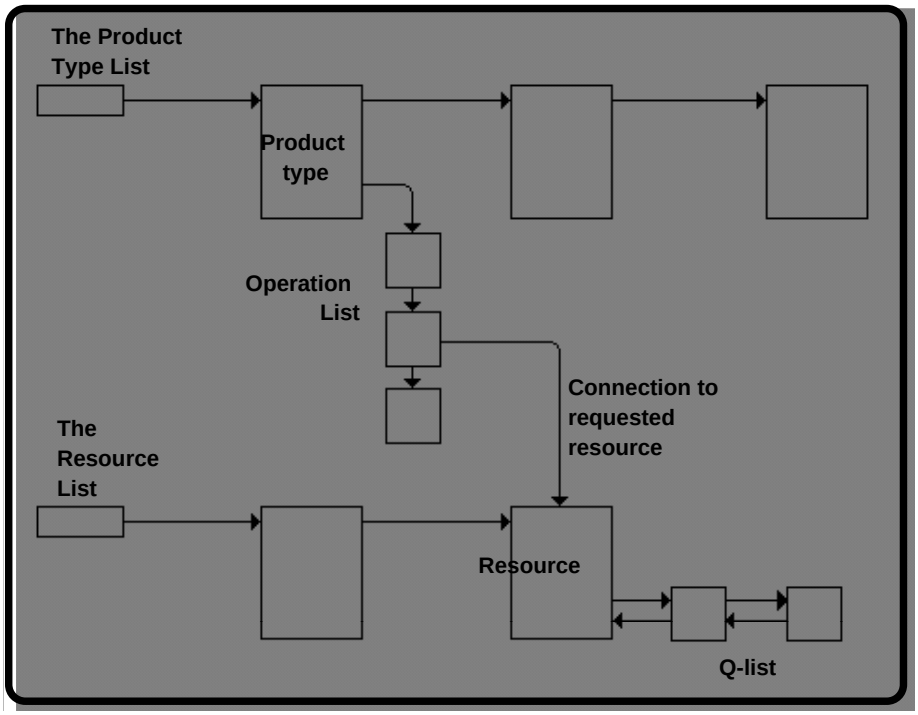

Figure 3.11 The lists created by the Linker

3.5.5 Special functions

Two special functions will internally be set-up as special entities and become elements in the Product Type List. These are the handling of the resources Mean Time Between Failure, MTBF, and the Sales Function of the products. The solution of these functions is transparent for the user.

To handle MTBF, the Linker produces a special entity which periodically requests the particular resource and holds it for the specified reparation time. This special entity has top priority and will be served first when it requests a resource. Each resource with MTBF specified, will generate such a special entity type. These entities will be represented as product type elements in the Product Type List.

The sales function acts in a similar way. The special entity for handling sales, requires work pieces from a finished product which has become a product resource. Its behaviour is very similar to products which request assembly parts. The sales entity has no special priority like the MTBF function, and will be served according to chosen priority rules. Each product type with sales function specified, will create a sales entity stored as a product type element in the Product Type List.

3.5.6 In the Simulation Kernel

Each occurrence of a product is represented by a product element. This element contains run time information connected to that specific product. In addition, the element contains a pointer to the operation list of its product type, pointing to the next operation that will be executed for this product (Figure 3.12). The element also contains a time variable which shows the point of

time for execution of the next operation, and a pointer mechanism which enables it to be put into a Q-list of a resource or into the Event List.

The Event List, like the Q-lists of the resources, is a double linked list. During execution of the Simulation Kernel, the list contains product elements which have, for the moment, got all their requested resources, and now consume time. The product elements in the list are sorted in ascending order after their point of time for next operation.

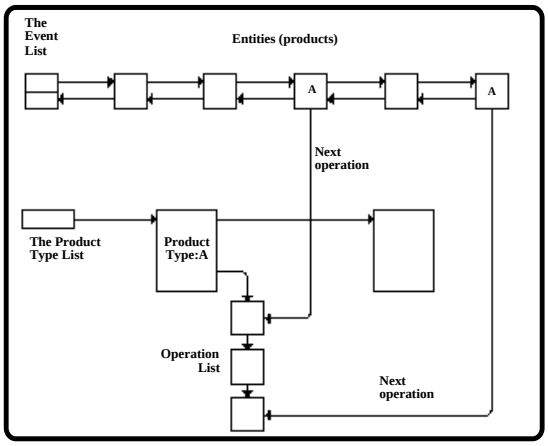

Figure 3.12 The entities' (products') connection to their next operation

It is always the first product element in the Event List which drives the Simulation Kernel. The first element is picked out of the list and becomes the current. The global simulation time is updated, and the current element's next operation is executed.

If the operation can be performed immediately, the element is still considered to be the current element, and the next following operation will then be executed. Otherwise, the element will be put back into the Event List with a new time schedule, or placed in a Q-list of a resource waiting for release of that resource (Figure 3.13).

Elements which are put back into the Event List, have executed a time consuming operation. Elements which are placed in a Q-list of a resource, have requested a resource which was not available at that moment.

New products are dynamically created during run time based on information stored in the elements of the Product Type List. These products are immediately put into the Event List. When a product has stepped through its Process Plan (i.e., the product types operation list), the product "dies". Summary information of the product type and the product resource is updated.

Allocating memory at run time for product bodies may slow down the execution speed of the system. Specially if the free memory available has been fragmented. To speed up the system, a pool (actually a stack) of product bodies are present. When a new product is to be created, a product body is popped off the stack and connected to that specific product type. When a product terminates, the product body is released from its type specification and pushed on the stack.

The stack is of limited size. If the stack is filled up, the body of terminated products is released as free memory. On the other hand, if the stack is empty, new product bodies must be allocated from the free memory space.

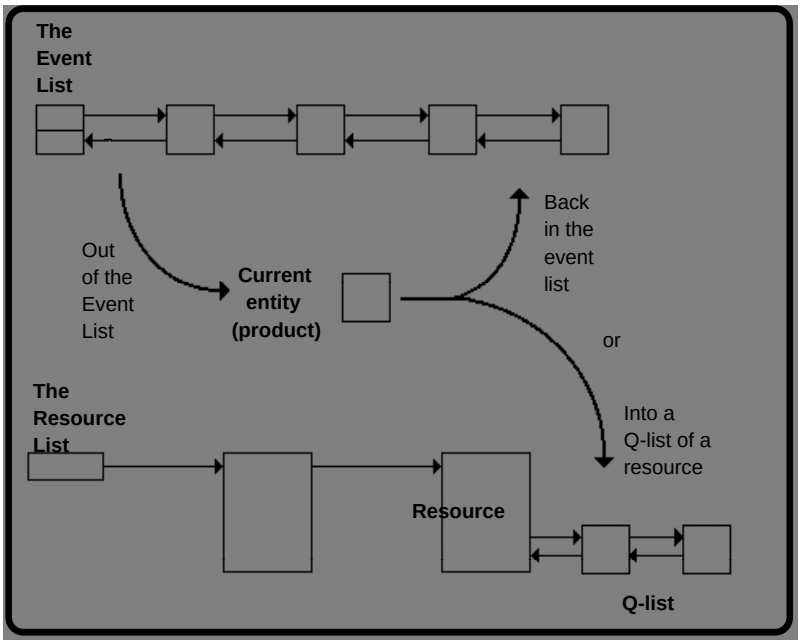

Figure 3.13 The current entity (product);
Back in the Event List or in a Q-list waiting for a resource

3.5.7 Result reports

The most detailed information about a simulation experiment is stored in the Event Log. The log contains a list of every event which has happened during the data registration period of the experiment; from the end of the warm up period to finished simulation time. Each event is recorded in one line and arranged after its point of happening time. The log is stored as alphanumerically characters and connected to the Edit program, but there is no problems found in looking at the log with other text processing systems (for instance Microsoft's Word).

All other result reports are stored in the general format supported by either Excel or the two programs Cricket Graph and StatWorks made by Cricket Software. In the next version of SIMMEK only the Excel format of result files will be available. The Excel format is chosen for many reasons. Excel is;

* A spreadsheet with macros and graphics

* The most used spreadsheet on Macintosh

* Available also on MS-DOS and other hardware platforms

A detailed description of all results produced by SIMMEK is given in Section 5.

3.6 Advantages and disadvantages of SIMMEK

As any other computer tool, SIMMEK has its advantages and disadvantages. On the positive side the following points are most essential;

* A tool suitable both for decision support in strategic and operational production management

* Includes economical analysis

* Models and results are presented by spreadsheets and graphics, making it;
  - quick to model
  - easy to read and interpret
  - easy to post process the data
  - easy to integrate

* No programming or pseudo programming needed

Table 3.15 The major advantages of SIMMEK

Most of these points are self explaining, and reference is given to this whole Section 3 to show how this is implemented in SIMMEK.

The general status of SIMMEK today may be described by that it is extremely fast to learn to use, and to create the first models in. But is still limited in what can be modelled, and how fast the models can be changed.

Concerning programming, it is still true that the so-called programming free tools like SIMAN and Witness, need pseudo programming to be able to model a plant, and thus more time is

required to learn to use. On the other hand, this makes them more flexible in what can be modelled.

On the negative side, SIMMEK has what could be called lacks of facilities compared to other existing packages. The most important of them are;

*	No animation
*	No programming facilities
*	Modelling limitations
*	Slow in changing large set of parameters
*	Available only on Macintosh
*	No integration possibilities

Table 3.16 The major disadvantages of SIMMEK

It has been a strategy all the way in the SIMMEK project that animation was not to be included in SIMMEK. Animation has its potential in “selling simulation or a simulation package”, as well as for validation of bottlenecks in a layout simulation. As layout simulation has not been identified as the most important application for SIMMEK, it was decided that there should be no animation facilities.

The lack of programming facilities is the most severe lack of a tool like SIMMEK. This limits what can be modelled. At least it limits what can be modelled without a lot of abstraction and “tricky modelling”. But some phenomena must be simplified and not modelled, and in several cases this is crucial for the validity of the model.

On the other hand, such programming facilities should be optional, so that the inexperienced user can create complete, but not so sophisticated models, without using these facilities.

The last three points, slow in changing large set of parameters, available only on Macintosh, and no integration possibilities, have certainly been solved with the new version of SIMMEK, made outside the work presented in this report.

Concerning the integration aspects, speaking of the new version of SIMMEK, as well as any other package, there is still a lot of research work to be done to be able to make a smart transformation of data.

A final remark on the future development of SIMMEK, concerning the common research areas identified in Tables 2.3 and 2.4, the most important ones are to improve the integration facilities for a smart transformation of data, as well as being able to import and model plants and products from a one-of-a-kind production environment.

3.7 Hardware and software specifications and requirements, SIMMEK-I

The prototype was implemented on a Macintosh computer. SIMMEK can be used on all types of Macintosh computers. A Macintosh with 4 or more MB, and a 68030 processor is recommended.

It uses the Finder for modelling purposes and storing of data. The parameters handling application, the Linker and the Simulation Kernel is implemented in Light speed C with extensive use of the Macintosh toolbox functions. The simulation results are presented in files of Cricket Software format or Excel format (see although Section 3.5.7).

3.8 The new versions of SIMMEK, SIMMEK-II

As mentioned earlier, a lot of improvements and modifications has been done with SIMMEK. This work has been done with some other partners, and the main part at SINTEF-NTH has been performed by Eirik Borgen. Therefore it is not reported as a part of this Dr.ing. work, but only the main achievements are referred to.

These achievements can be summarised in the three following points, see Borgen's [12] new report.

*	All input can be given in the Excel format
*	Data can be received from an MRP II computer based production management system, or any other system that can support the Excel format
*	Available on both PC - MS/DOS and Macintosh

Table 3.17 The new SIMMEK-II versions

The effects of these achievements are obvious. SIMMEK-II is now available for "all" who are in possession of a personal computer. It can be used for operational decisions in production management. And it is very quick and easy to make changes for a whole set of parameters at once.