EUROSIM Comparison C2
"Flexible Assembly System"

Solutions and Results

F. Breitenecker, I. Husinsky
editors

ISBN 3-901608-08-7
FOREWORD

EUROSIM, the Federation of European Simulation Societies, started in 1990 the publication of the journal EUROSIM Simulation News Europe (SNE), a newsletter distributed to all members of the European simulation societies under EUROSIM's umbrella and to people and institutions interested in simulation. SNE is also part of Simulation Practice and Theory (SIMPRA), the scientific journal of EUROSIM.

The idea of the journal SNE (circulation 2500; edited by F. Breitenecker and I. Husinsky, ARGE Simulation News, Technical University of Vienna, Austria; three issues per year) is to disseminate information related to all aspects of modeling and simulation.

The contents of SNE are news in simulation, simulation society information, industry news, calendar of events, essays on new developments, conference announcements, simulation in the European Community, introduction of simulation centers and comparison of simulation software, simulators and (parallel) simulation techniques.

The series on comparisons of simulation software has been very successful. Based on simple, easily comprehensible models the software comparisons compare special features of modeling and experimentation within simulation languages:

- modeling technique
- event handling
- submodel features
- numerical integration
- steady-state calculation
- frequency domain
- plot features
- parameter sweep
- postprocessing
- statistical features
- statistical processors
- control strategies
- optimization
- random numbers
- complex strategies
- animation, etc.

Seven Software Comparisons, four continuous ones and three discrete have been set up. Furthermore, a second type of comparisons, the Parallel Comparison has been initiated.

The continuous comparisons are:

- Comparison 1 (C1; Lithium-Cluster Dynamics under Electron Bombardment, November 1990) deals with a stiff system;
- Comparison 3 (C3; Analysis of a Generalized Class-E Amplifier, July 1991) focusses on simulation of electronic circuits and eigenvalue analysis;
- Comparison 5 (C5; Two State Model, March 1992) requires very high accuracy computation;
- Comparison 7 (C7; Constrained Pendulum, March 1993) deals with state events.
The discrete comparisons are:
- Comparison 2 (C2; Flexible Assembly System, March 1991) gives insight into flexible structures of discrete simulators;
- Comparison 4 (C4; Dining Philosophers, November 1991) involves not only simulation but also different modeling techniques like Petri nets;
- Comparison 6 (C6; Emergency Department - Follow-up Treatment, November 1992) deals with complex control strategies;

SNE 10 introduced a new type of comparison dealing with the benefits of distributed and parallel computation for simulation tasks. Three test examples have been chosen to investigate the types of parallelization techniques best suited to particular types of simulation tasks.

Up to now, 100 solutions have been sent in. The table at the end of this ARGESIM report shows the number of solutions for the Software Comparisons as well as for the Parallel Comparison. The series will be continued.

This ARGESIM Report summarizes and discusses the solutions and results sent in for Comparison 2 (C2) **Flexible Assembly System**.

The report starts with a summary of Mr. Krauth (from BIBA Bremen, who defined the comparison) and Mr. Klussmann. This summary is a reprint from a contribution to the congress EUROSIM'95 entitled „Results and Experiences derived from a Comparison between Simulation Systems“ and refers to a „Preliminary Evaluation“ by Mr. Krauth, published in SNE 4, March 1992, which is the second contribution in this report.

The following two contributions, introducing the EUROSIM comparisons and discussing in some detail the Comparison 2, are reprints from papers written by the editors of SNE and published in Conference Proceedings or Reports, resp.

The presentation of the solutions sent in starts with the definition of this EUROSIM comparison (definition and definition with remarks, resp.)
In the following the solutions sent in up to now are printed in chronological order. Each solution is represented by the page printed in SNE and, if available, by the originals sent in by the originators. It is evident that early solutions are accompanied by more original paper work.

As reference a study with DESMO, by D. Martinssen and A. Häuslein (Universität Hamburg), discusses in detail many aspects of this comparison.

In 1993 / 1994 the solutions sent in were used to enrich a study on Efficiency and Availability of discrete simulation software. This study (in German, supported by a grant of the Austrian Ministry for Research) is reprinted at the end of the report.

As conclusion a Table of the EUROSIM Comparisons and the number of solutions sent in is given.

F. Breitenecker, I. Husinsky, Editors
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:

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

For information contact: ARGÉSIM, 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

Foreword ii  
About ARGESIM v  
Results and Experiences derived from a Comparison between Simulation Systems 1  
Preliminary Evaluation 7  
Comparison of Simulation Software 9  
EUROSIM Comparisons of Simulation Tools 19  
Comparison 2, Definition, SNE 1, March 1991 25  
Comparison 2, Definition (Remarks), SNE 2, July 1991 27  
Solution PS SIMDIS, SNE 2, July 1991 29  
Solution DOS MIS3, SNE 2, July 1991 45  
Solution SIMAN, SNE 2, July 1991 46  
Solution SLAM II, SNE 2, July 1991 51  
Solution MicroSaint, SNE 3, November 1991 56  
Solution SIMUL_R, SNE 3, November 1991 61  
Solution GPSS/H, SNE 3, Novembewr 1991 62  
Solution CASSANDRA, SNE 4, March 1992 66  
Solution DESMO, SNE 4, March 1992 67  
Solution TOMAS, SNE 4, March 1992 68  
Solution SIMPLE-mac, SNE 4, March 1992 69  
Solution WITNESS, SNE 4, March 1992 70  
Solution SIMFLEX2, SNE 6, November 1992 71  
Solution EXTEND, SNE 6, March 92 72  
Solution TAYLOR II, SNE 8, July 1993 75  
Solution PCModel, SNE 10, March 1994 76  
Solution MOSYS, SNE 10, March 1994 77  
Solution CASSANDRA 3.0, SNE 11, July 1994 78  
Solution DSIM, SNE 11, July 1994 79  
Solution DESMO, Full Version (Study in German) 81  
Studie „Leistungsfähigkeit und Verfügbarkeit diskreter Simulationssoftware” 107  
Table of EUROSIM Comparisons 143
Results and Experiences derived from a Comparison between Simulation Systems

J. Klußmann\(^1\), J. Krauth\(^1\) and R. Splanemann\(^2\)
Bremen Institute of Industrial Technology and Applied Work Science at the University of Bremen (BIBA)\(^1\), Degussa AG\(^2\)

0. Abstract

This paper investigates if different simulation tools produce the same results when applied to the same system. Two comparisons have been carried out using the same test model of an assembly system. First a number of researchers carried out simulation experiments independent from each other, only on the basis of a written model definition. The results varied considerably, partly due to unclear model definition. In a second step a smaller number of tools has been compared by the authors themselves, thus excluding misunderstandings of the model definition. In this comparison the tools produced identical results once the models had been made really identical. But it was difficult to produce exactly identical models using different tools. Each tool has its inherent assumptions on "normal" behaviour, and if these assumptions are not known to the modeller, he is likely to generate a model with slight errors.

1. Introduction

Nowadays a wide variety of simulation tools is available on the market, especially in the field of material flow simulation in manufacturing systems. They all claim to be precise, but they all claim to be different. Hence the question arises: If used to simulate the same system, will they produce the same results? This question actually implies two questions: First, is it possible to create identical models using different tools? Especially with the modern comfortable tools which require no more programming but offer ready-made building blocks, this seems to be a problem. Building blocks are certainly very comfortable for quick and easy modelling, but they limit flexibility. So it is not clear if different simulation tools allow to generate identical models at all. The second question then is: In case the models can be made identical, do the tools then produce the same simulation results? Apparently these questions are very critical for the credibility of simulation.

The answer to both questions is "Yes, but..." In principle we can trust simulation results, but we have to be careful. This result is not surprising, but we feel it is often ignored in practical applications.

The paper is structured as follows: The next section describes the test system which is a simplification of a real assembly system. Section 3 reports on the result of a "distributed" comparison carried out by a number of researchers who each had only the written definition of the test system. The last section 4 represents own experiences of modelling the test system with three, different simulation tools and draws some conclusions about it.
2. The test system

We published the following test system definition in the journal "Eurosim Simulation News" in 1992 [1] and asked all interested persons or institutes to send us their solutions.

The test system consists of 7 assembly stations and a load/unload station all linked by an automated flexible conveyor system. This system is sketched in figure 1. An inner rectangular conveyor circulates clockwise and transports pallets on which the products to be assembled are fixed. The inner conveyor connects 8 subsystems as shown in figure 2. Each subsystem comprises one of the eight stations (assembly or load/unload) Ax, a buffer conveyor B2 of variable length in front of the station, a one-place buffer behind it, a bypass conveyor B1, and two connecting elements Sx and Sy. Here B1 is part of the inner rectangle. A pallet coming from the left can either be shifted to B2 in Sx or move along on B1. It is shifted to B2 if the following two conditions are satisfied:

- the product on the pallet has not yet undergone the operation(s) carried out in station Ax,
- there is enough space on the conveyor B2 in front of station Ax.

![Figure 1: Flexible assembly line as test system](image)

Finished products are taken from pallets and replaced by unprocessed parts in station A1. The sequence of operations the products undergo is arbitrary with the only exception that A2 has to be the first or the last station. All three stations A2 perform the same operations, hence only one of them has to process each product. Station A6 functions as a substitute of stations A3, A4, and A5. It performs all of the missing operations of these three whenever a product is being processed in A6.

![Figure 2: Subsystem of the flexible line](image)
The processing times and the buffer sizes in front of all stations as well as the length of the respective bypass conveyors are given in table 1.

<table>
<thead>
<tr>
<th>Station</th>
<th>Operation time (seconds)</th>
<th>Length of buffer in front of station (m)</th>
<th>Length of bypass conveyor (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>15</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>A2</td>
<td>60</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A3</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A4</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A5</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A6</td>
<td>30</td>
<td>2.0</td>
<td>1.2</td>
</tr>
</tbody>
</table>

*Table 1: System parameters*

After publishing this model definition we received a letter pointing out some ambiguous details of the definition. So we published a second, more precise definition in the following issue of Eurosim Simulation News [2] which we do not repeat here.

3. First Comparison

In the first comparison many scientists were involved, and their communication was very limited. The result has already been documented in Eurosim Simulation News [3], so we only repeat a summary here:

Table 2 (on the next page) gives the results obtained with 19 different simulation tools for a simulation time of eight hours. All tools have been applied by different researchers. The table shows a surprising diversity of results.

From this table it is impossible to tell to what degree the diversity of results is due to different understanding of the model definition or to errors in either the model implementation or the simulation software itself. From some researchers we received a note afterwards that they had misunderstood the model definition.

One ambiguous point is the question whether the time needed to feed a pallet into a station is part of the processing time or not. In case it is the bottleneck stations are A2, A3, A4, and A5, and the maximum number of products that can be processed in eight hours is 1440 because in this case every 20 sec. a product can be finished. In case it is not the bottleneck are stations A2 with a processing time of 61.3 sec., hence every 20.4 sec. a product can be finished, and therefore the maximum number of products is 1411. Most of the results obtained are close to one of these numbers.
<table>
<thead>
<tr>
<th>Simulation system</th>
<th>Distributor</th>
<th>Author of test model</th>
<th>Number of assembled parts (with 20 pallets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSES</td>
<td>University of Chemnitz (D)</td>
<td>Ges. f. Prozeßautom. &amp; Consult., Chemnitz (D)</td>
<td>1462</td>
</tr>
<tr>
<td>TAYLOR</td>
<td>F&amp;H, Düsseldorf (D)</td>
<td>F&amp;H, Düsseldorf (D)</td>
<td>1441</td>
</tr>
<tr>
<td>EXTEND</td>
<td>Imagine That, San Jose (USA)</td>
<td>University of Rostock</td>
<td>1440</td>
</tr>
<tr>
<td>SLAM II</td>
<td>Schröder GmbH, Düsseldorf (D)</td>
<td>AIC, Turin (I)</td>
<td>1440</td>
</tr>
<tr>
<td>SIMPLE-mac</td>
<td>AESOP, Stuttgart (D)</td>
<td>Unseld &amp; Partner, Vienna (A)</td>
<td>1439</td>
</tr>
<tr>
<td>WITNESS</td>
<td>AT&amp;T Istel, Düsseldorf (D)</td>
<td>BIBA, Bremen (D)</td>
<td>1439</td>
</tr>
<tr>
<td>DSIM</td>
<td>University of Vienna (A)</td>
<td>University of Vienna (A)</td>
<td>1425</td>
</tr>
<tr>
<td>CASSANDRA</td>
<td>KFKI, Budapest (H)</td>
<td>KFKI, Budapest (H)</td>
<td>1415</td>
</tr>
<tr>
<td>MICRO SAINT</td>
<td>Rapid Data, Worthing (GB)</td>
<td>Micro Analysis and Design, Boulder (USA)</td>
<td>1411</td>
</tr>
<tr>
<td>GPSS/H</td>
<td>Dr. Staedler GmbH, Nürnberg (D)</td>
<td>University of Michigan (USA)</td>
<td>1409</td>
</tr>
<tr>
<td>SIMFLEX/2</td>
<td>University of Kassel (D)</td>
<td>University of Kassel (D)</td>
<td>1409</td>
</tr>
<tr>
<td>DESMO</td>
<td>University of Hamburg (D)</td>
<td>University of Hamburg (D)</td>
<td>1408</td>
</tr>
<tr>
<td>DOSIMIS-3</td>
<td>SDZ, Dortmund (D)</td>
<td>IML, Dortmund (D)</td>
<td>1408</td>
</tr>
<tr>
<td>SIMUL_R</td>
<td>Simutech, Vienna (A)</td>
<td>Simutech, Vienna (A)</td>
<td>1405</td>
</tr>
<tr>
<td>EXAM</td>
<td>Russian Academy of Science, Moscow</td>
<td>Russian Academy of Science, Moscow</td>
<td>1404</td>
</tr>
<tr>
<td>PC SIMDIS</td>
<td>University of Magdeburg (D)</td>
<td>University of Magdeburg (D)</td>
<td>1384</td>
</tr>
<tr>
<td>MOSYS</td>
<td>IPK, Berlin (D)</td>
<td>IPK, Berlin (D)</td>
<td>1346</td>
</tr>
<tr>
<td>SIMAN</td>
<td>Dornier-System GmbH, Friedrichshafen (D)</td>
<td>CI Multiphase Centre, Chippenham (GB)</td>
<td>919</td>
</tr>
<tr>
<td>TOMAS</td>
<td>DVZ, Neubrandenburg (D)</td>
<td>DVZ, Neubrandenburg (D)</td>
<td>884</td>
</tr>
</tbody>
</table>

Table 2: Simulation results with different simulation systems

What was particularly remarkable about this first comparison was the little feedback we received after having published the model definition the first time: Only one researcher asked for clarification of several ambiguous points. These are pointed out and clarified in [2]. All other researchers seemed to understand immediately what we meant. But the diversity of results proved that their understandings deviated from ours, and also from each other. When
we had published the clarification, nobody else asked for any more information, even though there were still many points unclear.

What does this mean? It is obviously very difficult to define a model in an unambiguous way. And it seems to be equally difficult to even notice where there are ambiguities. When one person defines something very precisely, and another person understands him perfectly well, it does not necessarily mean both have the same understanding. And it may take a very long time until they notice they have not. We believed our first - and even more our second - definition was clear enough to build a model, and the majority of researchers thought so, too. But what we defined and what they understood was not always the same. At least in some cases we definitely know differences in understanding the model definition.

With respect to simulation this implies there is always a risk of misunderstanding when a simulationist and an engineer cooperate and communicate about a model. This risk can of course be avoided when the engineer builds the model himself. But in order to enable him to do so, the tool must provide him with constructs he understands. Nowadays a considerable number of simulation tools provides such domain-specific building blocks as - in the case of manufacturing simulation - machines, buffers, conveyour, etc. Their dynamics are predefined, so the user does not have to define them any more, he simply selects and combines them. But as we shall see below, this creates a new source of misunderstanding.

4. Experiences derived from a second Comparison

In the next step our aim was to exclude all sources of misunderstanding. Therefore we built the models on our own, using three different simulation systems: Dosimis-3 [4], Simple++ [5], and Witness [6]. These tools are frequently used in German manufacturing industry. They are particularly suited to model manufacturing and assembly systems, they support graphical modelling, and they provide the user with pre-defined domain-specific building blocks.

First simulation runs showed small differences between the results of the different simulation systems. A very detailed validation process proved that the three models were not identical. With each of the tools we had made some mistakes in modelling, mainly based on misunderstandings of the functionality and the behaviour of the pre-defined building blocks or modules the simulation tools provide. The detailed problems in modelling with the three simulation systems is published in [7]. In the following we represent the results and conclusions of this comparison.

The same results have been achieved with all three simulation systems. As well with all systems small mistakes first showed a little impact on the result. The mistakes happened mostly by modelling the distributing and connecting elements Sx and Sy. The reasons for all the mentioned mistakes are misunderstandings of the detailed behaviour of simulation system building blocks.

We assume the same results could also have been obtained using any of the other tools involved in the first comparison - or at least with the majority of them. It may be easy with some of them, and more tricky with others. But this does not mean some are good and some are bad. It only means they have primarily been designed for different purposes, by designers who had different perceptions of what a "normal" manufacturing system does.

The problem is that the user is quite often not aware of these differences in details. He himself has his own understanding of "normal" behaviour, and he tends to assume these comfortable modern simulation tools provide him with precisely the building blocks he expects.
Why should he think a "conveyor" block e.g. does not behave the way the conveyors he knows behave in reality? Unfortunately this assumption is often wrong. And - even worse - the exact description of the dynamics is often not available in the manual.

One solution of this problem is of course to make users aware of the potential diversity of building block behaviour, and to document precisely the behaviour of all building blocks in the manual. Another solution might be to provide the user with techniques to define his own building blocks. These techniques however have to be very simple, otherwise we would be back at simulation languages or even programming languages. Moreover the verification of user-defined blocks must be supported because he is likely to make mistakes, he will probably not test and validate them with sufficient rigour, and he is likely to use his own blocks again and again. And finally, we expect that user-defined building blocks will be documented even less, and therefore they will only be useful for the author himself, and after some months maybe not even for him. Hence simulation tools which allow the user to define his own building blocks have to provide solutions for these three subsequent problems of comfort and simplicity, of correctness, and of documentation.

A prototype of such an advanced simulation tool, allowing for user-defined building blocks and providing some techniques for rigorous verification, has been described in [8]. Petri nets have been used to define or modify application oriented building blocks. The mathematical theory of Petri nets allows for some rigorous testing of user-defined blocks, thus supporting their verification and validation to some extent. To our knowledge not much has been done since then to investigate further possibilities of validation support. However, more recent work towards tools which enable the user to define his own building blocks can be found in [9] and [10].

References:

Comparison 2: Preliminary Evaluation

Issues 2, 3, and this one of EUROSIM - Simulation News Europe contain a number of reports on tests of simulation tools applied to a test example described in issues 1 and 2: Simulation of a Flexible Assembly System (Comparison 2). Some of the tools reported so far have produced results of remarkable conformity, whereas others are so different that we assume the model description has not been sufficiently precise. It is known to us that the colleagues who used Micro Saint had a different understanding of the operation time of ation A1 (Load/Unload) than we had: They assumed 15 sec - loading and 15 sec for unloading, whereas we meant 15 sec for both operations, i.e. 7.5 sec each (cf. issue 2, p. 26). Therefore their numbers are very different from the majority. Probably other ambiguities have led to other strongly deviating results (SIMAN, TOMAS, SLAM II, PS SIMDIS). It could be interesting to know what these ambiguities were, and it would also be interesting to know why no two tools have produced precisely the same results. Some of them are too close to each other. It seems that the authors have had the same understanding of the system, but that the software tools work a little bit different somehow. We will try to plumb some of these little differences in a later issue. The following table shows the findings for twenty pallets in the system:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Total throughput</th>
<th>Average throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMDIS</td>
<td>1384</td>
<td>-</td>
</tr>
<tr>
<td>QSMIS</td>
<td>1408</td>
<td>436.9</td>
</tr>
<tr>
<td>MAN</td>
<td>919</td>
<td>627.6</td>
</tr>
<tr>
<td>AM II</td>
<td>1082</td>
<td>400.0</td>
</tr>
<tr>
<td>Micro Saint</td>
<td>-</td>
<td>603.0</td>
</tr>
</tbody>
</table>

SIMUL R 1405 409.5
GPSS/H 1409 409.2
CASSANDRA 1415 410.7
DESMO 1408 408.0
TOMAS 884 623.8
SIMPLE-mac 1439 400.2
WITNESS 1439 409.3

This summary does not intend to close the comparison. Further contributions are still welcome. We would however encourage every colleague who intends to test another tool to contact us in case of any questions concerning the definition of the test model. One of the reasons why we chose this model was that it allows to check two features of discrete event simulation systems that we consider very important for the simulation of complex production systems:

- the possibility to define and combine submodels (the model consists of 8 slightly different submodels!)
- the method to describe complex control strategies

Unfortunately most of the contributions we received so far do not discuss these topics. So no evaluation of such properties of tools is possible at the moment. Maybe future contributions will include some remarks on these points, too.

We want to thank all the authors for their interesting reports and hope to receive more!

Contact: J. Krauth, BIBA (Bremer Institut für Betriebs- technik und angewandte Arbeitswissenschaft), Postfach 33 05 60, D-2800 Bremen 33, Tel: +49-421-22 009 51 Fax: +49-421-22 009 79.

---

Comparison 2: Flexible Assembly System from EUROSIM - Simulation News Europe (March '91)

System Clocks 29 minutes

Statistical for Last Departure
Time in System: 627 sec
Time Since Previous: 16 sec

Station 01 (Op'n 1)
Station 02 (Op'n 2)
Station 03 (Op'n 2)
Station 04 (Op'n 2)

Problem developed by J. Krauth, BIBA

Animated GPSS/H comparison model, available from Wolverine Software Corporation, 4115 Annandale Road, Annandale, Virginia 2203-2500 USA

The editors also received an animation diskette of the comparison 2 SIMAN model from The CIMulation Centre in England (see title page of this issue), comparison solution published in EUROSIM - Simulation News Europe, Number 2, July 1991, page 29.
Comparison of Simulation Software

I. Husinsky, F. Breitenecker

Computer Centre and Dept. Simulation Technique
Technical University of Vienna
Wiedner Hauptstr. 8-10, A - 1040 Vienna, Austria

Abstract

In the last years simulation languages have become quite numerous. The catalog of simulation software is quite comprehensive today. An attempt to obtain information about the properties of the different simulation languages has been started by Software Comparisons within the journal EUROSIM - Simulation News Europe. Before the idea and first results of these software comparisons are explained the question is discussed what the requirements for simulation languages should be and what features should be taken into account in a comparison.

Keywords: Simulation languages, comparison of features

FEATURES OF SIMULATION SOFTWARE

Features of simulation software may be seen as features for modelling, features for experimentation and general features. Modelling features may differ in the type of model description, in what complexity may be obtained, how events can be handled, if there is modularity, how submodels can be realized, how implicit models can be managed, etc. Experimentation features are type and efficiency of experimentation tools, like time domain analysis, calculation of steady states, linearization, frequency domain analysis, parameter studies, Monte Carlo studies, etc. General features deal with the 'handling' of the simulation software in general: modes, interfaces to other languages, etc.

Another classification may follow aspects regarding to computer science, software design and algorithms: input/output, state of implementation, documentation; program (model) structure (parallel or procedural structure, handling of events, submodels, hierarchical models, initialisation, macro description); efficiency of numerical algorithms (integration algorithms, algebraic loop solvers, steady state computation, special handling of linear systems, linearization, frequency domain methods, interpolation, error
control, error analysis, etc.; identification and validation (parameter identification, model comparison, sensitivity analysis, analysis of linearized models, etc.); program compilation and program execution; date storage and data access; statistical evaluation; table functions; etc.

A third classification may be based on analysis methods. In case of continuous simulation typical analysis methods (to be performed with a model) are deterministic analysis in the time domain, parameter studies, Monte Carlo Studies, computation of steady states, end game, analysis of linearized models, etc. In case of discrete simulation this point becomes very important, too: long term runs, iterated runs, output analysis of different runs, hypothesis testing, etc.

2. BENCHMARKS

First efforts in comparing simulation software were done after standardization in 1968 (CSSL-standard). Benchmark problems have been developed, they try to compare features of simulation languages within application models.

A few of the best known simulation benchmarks, that can be found in user manuals, are: control loop (testing macros for transfer functions), PHYSBE (physiological benchmark, testing 'literal' macros), pilot ejection study (state event, different model descriptions), discrete sample compensator (testing features for difference equations or/and time events), Joe's barbers shop, etc.

Solutions of these benchmark problems offer deep insight into the efficiency of a particular simulation software, concentrating on a few features, usually depending on the application area. Disadvantages of these benchmark problems are that the models are relatively large, that the user has to become familiar with a specific application area and that they may be difficult to reproduce.

3. EUROSIM COMPARISONS

As a consequence of the lack on concentrated information, software comparisons were developed, which on the one hand are concentrated enough to be overviewed within short time and on the other hand provide enough information for showing the implementation of the features, for showing how the features work and for reproducing the results within any language.

These comparisons started in 1990 within the journal EUROSIM - Simulation News Europe ([1]). EUROSIM - Simulation News Europe (in the following abbreviated with 'SNE') is the official newsletter of EUROSIM, the federation of European simulation societies. This newsletter is distributed to all members of these societies and to persons and institutions interested in. From the end of 1992 regular subscription will be available. The comparisons are based on simple, easily comprehensible models taken from different application areas. Up to four tasks (checking features mentioned above) have to be solved. The comparisons are selected and prepared by the editors of SNE.

People developing or using different simulation tools are asked to participate in this comparison by simulating the problem in a simulation language of their choice, to solve the given tasks, and to send a short report. The reports are published in SNE, one page
per solution. The reports contain a short description of the language used, the model
description (source code or diagram), the results of the tasks (commands, tables, plots),
and any comments.
Up to now five different comparisons have been introduced in SNE, continuous and
discrete problems alternatively, comparisons are numbered 1 to 5 (table 1). Preliminary
evaluations will be published from time to time, a comparison will be closed after three
years with a final evaluation. The idea has become quite successful. Many solutions have
been sent to the editors, demonstrating a broad spectrum of different simulation
software.

<table>
<thead>
<tr>
<th>No.</th>
<th>SNE No.</th>
<th>TITLE</th>
<th>TASKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. 1</td>
<td>SNE 0</td>
<td>Lithium-Cluster Dynamics</td>
<td>stiff system solution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>parameter sweep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>steady state calculation</td>
</tr>
<tr>
<td>No. 2</td>
<td>SNE 1/2</td>
<td>Flexible Assembly System</td>
<td>submodel features</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>average throughput</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>optimal throughput</td>
</tr>
<tr>
<td>No. 3</td>
<td>SNE 2</td>
<td>Generalized Class-E Amplifier</td>
<td>eigenvalue calculation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>stiff system solution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>table parameter sweep</td>
</tr>
<tr>
<td>No. 4</td>
<td>SNE 3</td>
<td>Dining Philosophers Problem</td>
<td>modelling technique</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>control strategies</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>deadlocks, reachability</td>
</tr>
<tr>
<td>No. 5</td>
<td>SNE 4/5</td>
<td>Two State Model</td>
<td>discontinuity modelling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>discontinuity location</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>high-accr. simulations</td>
</tr>
</tbody>
</table>

Table 1: EUROSIM
Comparisons 1990-1992

RESULTS OF TWO SOFTWARE COMPARISONS

This section summarizes preliminary results of the first continuous and the first discrete
comparison (both comparisons are still running).
Comparison 2 deals with a flexible assembly system testing submodel features and
complex control strategies and will be discussed in more detail (discrete simulation).
Comparison 1 is a problem taken from solid state physics, a nonlinear stiff third-order
model describes the concentration of certain aggregates (continuous simulation).  

---

11
EUROSIM Comparison 2: Flexible Assembly System

Up to now twelve simulation languages took the challenge to solve EUROSIM Comparison 2. This comparison checks two important features of discrete event simulation tools:

- features for defining and combining submodels
- features for describing complex control strategies.

The model formulated in [4] consists of a number of almost identical assembly stations Ax placed on two linked belts B1 and B2 (fig. 1). Parts to be processed and assembled are put into the system on pallets in station A1, where they leave the system, too - the pallets become free for new parts. The parts on pallets are processed in the stations A2 - A6 due to a complex strategy; some stations are identical (A2), some are 'intelligent' performing different operations (A6), etc.

![Diagram of Flexible Assembly System](image)

**Fig. 1: Comparison 2, model layout and animation (GPSS/H)**
The tasks are:

a) modelling the system by means of submodel features,
b) evaluation of the total throughput and the average throughput time of parts with 20, 40 and 60 pallets
c) determining the number of pallets with the maximal throughput and with a deadlock.

A preliminary evaluation ([5]) first showed, that because of the stochastic nature of the processes the results must differ. Some of the simulation tools have produced results of remarkable conformity, others are so different that different modelling techniques have to be assumed - based on different understandings of the problem.

<table>
<thead>
<tr>
<th>LANGUAGE</th>
<th>TOTAL THROUGHPUT</th>
<th>AVERAGE THROUGHPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS SIMDIS</td>
<td>1384</td>
<td></td>
</tr>
<tr>
<td>DOSIMIS</td>
<td>1408</td>
<td>436.9</td>
</tr>
<tr>
<td>SIMAN</td>
<td>919</td>
<td>627.6</td>
</tr>
<tr>
<td>SLAM II</td>
<td>1082</td>
<td>400.0</td>
</tr>
<tr>
<td>Micro Saint</td>
<td></td>
<td>603.0</td>
</tr>
<tr>
<td>SIMUL_R</td>
<td>1405</td>
<td>409.5</td>
</tr>
<tr>
<td>GPSS/H</td>
<td>1409</td>
<td>409.2</td>
</tr>
<tr>
<td>CASSANDRA</td>
<td>1415</td>
<td>410.7</td>
</tr>
<tr>
<td>DESMO</td>
<td>1408</td>
<td>408.0</td>
</tr>
<tr>
<td>TOMAS</td>
<td>884</td>
<td>623.8</td>
</tr>
<tr>
<td>SIMPLE-mac</td>
<td>1439</td>
<td>400.2</td>
</tr>
<tr>
<td>WITNESS</td>
<td>1439</td>
<td>409.3</td>
</tr>
</tbody>
</table>

Table 2: Comparison 2,
results task b) ([4])

Table 2 shows the results for twenty pallets in the system for the results sent in up to now. Some of the tools have produced results of remarkable conformity, whereas others are different because of misunderstandings of the model description. The summary will be updated with results of other simulation tools and corrected results, the comparison is running up to 1993.

The simulation languages compared in table 2 are of different nature. There are classical languages, like GPSS/H (results in table 3, animation in fig. 1, there are new (combined) languages with powerful postprocessing features (SIMUL_R, fig. 2), and there are the 'graphical' languages (MicroSaint, fig. 3).
<table>
<thead>
<tr>
<th>Number of Pallets</th>
<th>Jobs Completed in Final 8 Hours</th>
<th>Job Completion Time (Minutes)</th>
<th>Number of Jobs Needing 1 or 2 or 3 or More Laps to Finish</th>
<th>Number of Uses of Station A6</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1350</td>
<td>5.33 0.89</td>
<td>1350 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1350</td>
<td>5.69 0.89</td>
<td>1350 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>1371</td>
<td>5.96 0.82</td>
<td>1371 0 0 0</td>
<td>88</td>
</tr>
<tr>
<td>18</td>
<td>1408</td>
<td>6.13 0.79</td>
<td>1408 0 0 0</td>
<td>293</td>
</tr>
<tr>
<td>19</td>
<td>1409</td>
<td>6.47 0.53</td>
<td>1409 0 0 0</td>
<td>118</td>
</tr>
<tr>
<td>20</td>
<td>1409</td>
<td>6.82 0.50</td>
<td>1289 115 5 0</td>
<td>116</td>
</tr>
<tr>
<td>25</td>
<td>1408</td>
<td>8.52 2.75</td>
<td>682 300 180 246</td>
<td>252</td>
</tr>
</tbody>
</table>

Table 3: Results Comparison 2, GPSSH

Fig. 2: Results Comparison 2, SIMUL_R: pallet variation and animation
EUROSIM Comparison 1: 'Lithium-Cluster Dynamics' Model

The 'Lithium-Cluster Dynamics' Model describes the behaviour of defects under electron (and photon) bombardment of alkali halides. One of the important consequences of these electronic defects is the desorption of surface atoms ([2]). The dynamic model is based on the equations for the concentrations of different aggregates, resulting in three states governed by nonlinear stiff equations: at the beginning and after the end of the electronic bombardment the transients are very rapid, then they become relatively slow (solution with ESL, fig. 4).

The tasks to be performed are:

a) simulation of the stiff system (testing integration algorithms)

b) parameter study and plot (fig. 1)

c) steady state calculation.

Fig. 3: Results Comparison 2, MicroSaint: graphical modelling
Up to now 17 simulation languages took the challenge to solve EUROSIM Comparison 1.

First it has to be noted that all simulation languages fulfilled the tasks with sufficient accuracy. The main results of the comparison are: i) comparison of modeling techniques, ii) effectiveness of numerical algorithms, iii) comfortability of parameter sweep, iv) features for steady state calculation, and v) preventing problems by analytical transformations. A detailed preliminary evaluation can be found in [3].

The languages can be divided roughly into three groups: equation-oriented languages, (graphical) block-oriented languages, and application-oriented languages. Table 4 summarizes the modelling features of the languages in general, indicates the modelling technique used (marked with (*) ) and gives remarks.

It is relatively difficult to compare the results of tasks a). For comparison of the algorithms the relation between the different algorithms within one language may be of more importance than the absolute CPU-times. A comprehensive table on these results will be published in the next issue of SNE ([3]). The second task clearly answered which language offers runtime commands for parameter loops, where the loop can be programmed in the model description and where the parameter variation has to be done 'manually'. The third task checks which languages offer features for steady state calculation of the system.
<table>
<thead>
<tr>
<th>LANGUAGE</th>
<th>MODEL DESCRIPTION</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACSL</td>
<td>equations (ODE's)</td>
<td>General purpose simulator, event handling</td>
</tr>
<tr>
<td>DESIRE</td>
<td>equations (ODE's)</td>
<td>Combination with neural network simulation</td>
</tr>
<tr>
<td>DYNAST</td>
<td>equations (DAE's)</td>
<td>For linear systems semi-symbolic analysis</td>
</tr>
<tr>
<td></td>
<td>(*) graphical blocks (submodels) port diagrams (graphical)</td>
<td></td>
</tr>
<tr>
<td>ESACAP</td>
<td>equations (DAE's)</td>
<td>Based on numerical algorithms for circuit analysis</td>
</tr>
<tr>
<td></td>
<td>(*) nodes/branches arbitrary expressions</td>
<td></td>
</tr>
<tr>
<td>ESL</td>
<td>equations (ODE's)</td>
<td>Interpretative and Compile Mode, graphic postprocessor</td>
</tr>
<tr>
<td></td>
<td>(*) graphical blocks (submodels)</td>
<td></td>
</tr>
<tr>
<td>EXTEND</td>
<td>graphical blocks</td>
<td>Continuous and next event modelling</td>
</tr>
<tr>
<td>FSIMUL</td>
<td>graphical blocks (submodels)</td>
<td>&quot;Control-Engineering&quot; features, optimization features</td>
</tr>
<tr>
<td>HYBSYS</td>
<td>blocks (elementary)</td>
<td>Interpretative simulator, direct data base compilation</td>
</tr>
<tr>
<td></td>
<td>(*) equations</td>
<td></td>
</tr>
<tr>
<td>I Think</td>
<td>graphical blocks</td>
<td>Modelling based on System Dynamics</td>
</tr>
<tr>
<td>MATLAB</td>
<td>equations (MATLAB-functions)</td>
<td>Tool for mathematical and engineering calculations</td>
</tr>
<tr>
<td>NAP 2</td>
<td>blocks (electronic circuits)</td>
<td>Specialized for circuit simulation</td>
</tr>
<tr>
<td>PROSIGN</td>
<td>equations (ODE's)</td>
<td>Comb of modelling techniques, interfaces to C, etc.</td>
</tr>
<tr>
<td></td>
<td>(*) graphical blocks (submodels) application-oriented components</td>
<td></td>
</tr>
<tr>
<td>SIL</td>
<td>equations (ODE's, DAE's)</td>
<td>Simulation of continuous and discrete systems</td>
</tr>
<tr>
<td>SIMULAB</td>
<td>graphical blocks (submodels)</td>
<td>Based on MATLAB, analytical solution of linear parts</td>
</tr>
<tr>
<td></td>
<td>equations (MATLAB-function)</td>
<td></td>
</tr>
<tr>
<td>SIMUL_R</td>
<td>equations (ODE's)</td>
<td>Open system (C-based), combined simulation</td>
</tr>
<tr>
<td></td>
<td>(*) bond graphs (graphical preproc.) blocks (graphical preprocessor)</td>
<td></td>
</tr>
<tr>
<td>STEM</td>
<td>equations (ODE's)</td>
<td>Based on Turbo Pascal</td>
</tr>
<tr>
<td>XANALOG</td>
<td>graphical blocks (submodels)</td>
<td>Sophisticated linearization, realtime - features</td>
</tr>
</tbody>
</table>

Table 4: Comparison 1 - Modelling features of simulation languages

REFERENCES


EUROSIM COMPARISONS ON SIMULATION TOOLS

F. Breitenecker, I. Husinsky
Dept. Simulation Techniques / Computer Centre
Technical University Vienna
Wiedner Hauptstraße 8-10, A - 1040 Vienna
Email: sne@simserv.tuwien.ac.at

ABSTRACT

This contribution gives an overview on the comparisons of simulation tools and their results, published in EUROSIM's journal "EUROSIM Simulation News Europe". Among general developments and trends (also negative ones) special aspects as object oriented approaches, standards for simulation tools and the question of a "common denominator" for simulation languages, and parallel simulation are sketched briefly.

EUROSIM, the Federation of European Simulation Societies was set up in 1989. The purpose of EUROSIM is to provide a European forum for regional and national simulation societies to promote the advancement of modeling and simulation in industry, research, and development.

EUROSIM started in 1990 the journal "EUROSIM Simulation News Europe" (SNE), a newsletter distributed to all members of the EUROSIM member societies and to people and institutions interested in simulation. In 1993 the first issue of "Simulation Practice and Theory", the scientific journal of EUROSIM, was published.

1 THE EUROSIM COMPARISONS

The idea of the journal SNE is to promote simulation in Europe by dissemination of information related to all aspects of modeling and simulation. SNE is edited by I. Husinsky and F. Breitenecker, Technical University of Vienna, Austria. SNE has a circulation of 2500 copies. There are three issues per year (March, July, November). Furthermore SNE is also included in the scientific journal "Simulation Practice and Theory" in three issues per volume, 1000 copies each.

The contents of SNE are news in simulation, simulation society information, industry news, calendar of events, essays, conference announcements, simulation in the European Community, introduction of simulation centers, discussion forum, and comparison of simulation software and hardware of simulation tools. The series on comparisons of simulation software is very successful. Based on simple, easily comprehensible models special features of modeling and experimentation within simulation languages, also with respect to an application area, are compared:

- modeling technique
- event handling
- submodel features
- numerical integration
- steady-state calculation
- frequency domain
- plot features
- parameter sweep
- postprocessing
- output analysis
- optimization
- animation

2 DEFINITION OF COMPARISONS

Seven "Software Comparisons", four continuous ones and three discrete ones, have been defined up to now. This series will be continued.

Furthermore, another type of comparisons, a comparison on parallel simulation techniques ("Parallel Comparison") has been initiated.

2.1 Software Comparisons

The continuous comparisons are: Comparison 1 (C1; Lithium-Cluster Dynamics under Electron Bombardment, November 1990) addressed all kinds of simulation software. 22 solutions have been sent in, a summary can be found in SNE 6, November 1992.

Comparison 3 (C3; Analysis of a Generalized Class-E Amplifier, July 1991) focused on simulation of electronic circuits resulting in up to now 11 solutions.

Comparison 5 (C5; Two State Model, March 1992, revised July 1992) takes more into account a very high accuracy computation than state events.

Comparison 7 (C7; Constrained Pendulum, March 1993) is a continuous comparison which addresses all kinds of simulation software, with nine solutions up to now.
The discrete comparisons are: Comparison 2 (C2; Flexible Assembly System, March 1991, comments July 1991) resulted in 21 solutions. A preliminary evaluation can be found in SNE 4.

Comparison 4 (C4; Dining Philosophers, November 1991) is more general task involving not only simulation but also different modeling techniques like Petri nets. Up to now eight solutions have been sent in.

Comparison 6 (C6; Emergency Department - Follow-up Treatment, November 1992) deals with complex control strategies. Six solutions have been presented up to now.

2.2 Parallel Comparison

*SNE 10* introduced a new type of comparison dealing with the benefits of distributed and parallel computation for simulation tasks. Three test examples have been chosen to investigate the types of parallelization techniques best suited to particular types of simulation task.

Each test example should be first solved in a serial fashion to provide a reference for the investigation of speed-up factors. The examples should then be tested using the parallel facilities (software and hardware) available. Performance should be assessed in terms of a numerical value found by dividing the time for serial solution by the time for the parallel solution. Information must be provided about the method of parallelization or distribution of subtasks.

The objective is to make comparisons of different types of methods for the parallelization of simulation tasks, not to compare the hardware performance.

The first test example is a Monte-Carlo-study of the influence of the damping parameter in a damped second order mass-spring system. The second example is concerned with coupled predator-prey population models. Five predator-prey populations are interacting. The model is strongly coupled. The third example is based on the a second order partial differential equation describing a swinging rope. Discretization by the method of lines results in a set of weakly coupled differential equations.

Table 1 shows the number of solutions published in each issue of *SNE*.

### 3 RESULTS OF THE COMPARISONS

The results of the comparisons have the following form:
- short description of the language
- model description
- results of tasks with experimentation comments

In the following results of two comparisons and preliminary results of the Parallel Comparisons will be sketched briefly.

Comparison 1 is a problem taken from solid state physics, a nonlinear stiff third-order model describes the concentration of certain aggregates.

Comparison 2 deals with a flexible assembly system testing submodel features and complex control strategies and will be discussed in more detail.

#### 3.1 Results Comparison 1 (C1)

The 'Lithium-Cluster Dynamics' Model describes the behavior of defects under electron (and photon) bombardment of alkali halides. The dynamic model is based on the equations for the concentrations of different aggregates, resulting in three states governed by nonlinear stiff equations: at the beginning and after the end of the electronic bombardment the transients are very rapid, then they become relatively slow.

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNE 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Def</td>
</tr>
<tr>
<td>SNE 1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Def</td>
</tr>
<tr>
<td>SNE 2</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Def</td>
</tr>
<tr>
<td>SNE 3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Def</td>
</tr>
<tr>
<td>SNE 4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 5</td>
<td>4</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 6</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td>Def</td>
</tr>
<tr>
<td>SNE 7</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td></td>
<td>Def</td>
</tr>
<tr>
<td>SNE 8</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SNE 9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 10</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>Def/1</td>
</tr>
<tr>
<td>SNE 11</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>22</td>
<td>21</td>
<td>11</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>2</td>
</tr>
</tbody>
</table>

*Table 1: SNE - Comparisons, publication of solutions*
The tasks to be performed are:
1. simulation of the stiff system
2. parameter study and plot
3. steady state calculation.

Up to now 22 simulation languages took the challenge to solve Comparison 1. First it has to be noted that all simulation languages fulfilled the tasks with sufficient accuracy.

The languages can be divided roughly into three groups:
- equation-oriented languages,
- (graphical) block-oriented languages,
- and application-oriented languages.

Table 2 summarizes the modeling features of the languages in general, indicates the modeling technique used (marked with (*) ) and gives remarks.

<table>
<thead>
<tr>
<th>LANGUAGE</th>
<th>MODEL DESCRIPTION</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACSL</td>
<td>equations (ODE's)</td>
<td>General purpose simulator, event handling</td>
</tr>
<tr>
<td>DESIRE</td>
<td>equations (ODE's)</td>
<td>Combination with neural network simulation</td>
</tr>
<tr>
<td>DYNAST</td>
<td>equations (DAE's) (*)</td>
<td>For linear systems semi-symbolic analysis</td>
</tr>
<tr>
<td></td>
<td>graphical blocks (submodels)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>port diagrams</td>
<td></td>
</tr>
<tr>
<td>ESACAP</td>
<td>equations (DAE's) (*)</td>
<td>Based on numerical algorithms for circuit analysis</td>
</tr>
<tr>
<td></td>
<td>nodes/branches arbitrary expressions</td>
<td></td>
</tr>
<tr>
<td>ESL</td>
<td>equations (ODE's) (*)</td>
<td>Interpretative and Compile Mode, graphic postprocessor</td>
</tr>
<tr>
<td></td>
<td>graphical blocks (submodels)</td>
<td></td>
</tr>
<tr>
<td>EXTEND</td>
<td>graphical blocks</td>
<td>Continuous and next event modeling</td>
</tr>
<tr>
<td>FSIMUL</td>
<td>graphical blocks (submodels)</td>
<td>'Control-Engineering' - features, optimization features</td>
</tr>
<tr>
<td>HYBSYS</td>
<td>blocks (elementary) (*)</td>
<td>Interpretative simulator, direct data base compilation</td>
</tr>
<tr>
<td></td>
<td>equations</td>
<td></td>
</tr>
<tr>
<td>I Think</td>
<td>graphical blocks</td>
<td>Modeling based on System Dynamics</td>
</tr>
<tr>
<td>MATLAB</td>
<td>equations (MATLAB-functions)</td>
<td>Tool for mathematical and engineering calculations</td>
</tr>
<tr>
<td>NAP 2</td>
<td>blocks (electronic circuits)</td>
<td>Specialized for circuit simulation</td>
</tr>
<tr>
<td>PROSIGN</td>
<td>equations (ODE's) (*)</td>
<td>Combin. of modeling techniques, interfaces to C, etc.</td>
</tr>
<tr>
<td></td>
<td>graphical blocks (submodels)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>application-oriented components</td>
<td></td>
</tr>
<tr>
<td>SIL</td>
<td>equations (ODE's, DAE's) (*)</td>
<td>Simulation of continuous and discrete systems</td>
</tr>
<tr>
<td>SIMULINK</td>
<td>graphical blocks (submodels) (*)</td>
<td>Based on MATLAB, analytical solution of linear parts</td>
</tr>
<tr>
<td></td>
<td>equations (MATLAB-function)</td>
<td></td>
</tr>
<tr>
<td>SIMUL_R</td>
<td>equations (ODE's) (*)</td>
<td>Open system (C-based), combined simulation</td>
</tr>
<tr>
<td></td>
<td>bond graphs (graphical preproc.) blocks (graphical preprocessor)</td>
<td></td>
</tr>
<tr>
<td>STEM</td>
<td>equations (ODE's)</td>
<td>Based on Turbo Pascal</td>
</tr>
<tr>
<td>TUTSIM</td>
<td>equations (ODE's)</td>
<td>Bond graph preprocessor</td>
</tr>
<tr>
<td>XANALOG</td>
<td>graphical blocks (submodels) (*)</td>
<td>Sophisticated linearization, real-time - features</td>
</tr>
</tbody>
</table>

Table 2: Comparison 1 - modeling features of simulation languages
3.2 Results Comparison 2 (C2)

Up to now twelve simulation languages took the challenge to solve EUROSIM Comparison 2. This comparison checks two important features of discrete event simulation tools:

- features for defining and combining submodels
- features for describing complex control strategies.

The model consists of a number of almost identical assembly stations Ax placed on two linked belts B1 and B2 (fig. 1). Parts to be assembled are put into the system on pallets in station A1, where they leave the system, too - (free pallets for new parts). The parts on pallets are processed in the stations A2 - A6 due to a complex strategy; some stations are identical (A2), some are ‘intelligent’ performing different operations (A6), etc.

The tasks are:
1. modeling the system by means of submodel features,
2. evaluation of the total throughput and the average throughput time
3. determining the number of pallets with the maximal throughput and with a deadlock.

A preliminary evaluation first showed, that because of the stochastic nature of the processes the results must differ. Some of the simulation tools have produced results of remarkable conformity, others are so different that different modeling techniques have to be assumed - based on different understandings of the problem.

The simulation languages compared in table 3 are of different nature. There are classical languages like GPSS/H, there are new (combined) languages with powerful postprocessing features (SIMUL_R) and there are the „graphical” languages like MicroSaint.

<table>
<thead>
<tr>
<th>LANGUAGE</th>
<th>Throughput total</th>
<th>Throughput average</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS SIMDISH</td>
<td>1384</td>
<td></td>
</tr>
<tr>
<td>DOSMIS</td>
<td>1408</td>
<td>436.9</td>
</tr>
<tr>
<td>SIMAN</td>
<td>919</td>
<td>627.6</td>
</tr>
<tr>
<td>SLAM II</td>
<td>1082</td>
<td>400</td>
</tr>
<tr>
<td>MicroSaint</td>
<td>-</td>
<td>603.0</td>
</tr>
<tr>
<td>SIMUL_R</td>
<td>1405</td>
<td>409.5</td>
</tr>
<tr>
<td>GPSS/H</td>
<td>1409</td>
<td>409.2</td>
</tr>
<tr>
<td>CASSANDRA</td>
<td>1415</td>
<td>410.7</td>
</tr>
<tr>
<td>DESMO</td>
<td>1408</td>
<td>408.2</td>
</tr>
<tr>
<td>TOMAS</td>
<td>884</td>
<td>623.8</td>
</tr>
<tr>
<td>SIMPLE</td>
<td>1439</td>
<td>400.2</td>
</tr>
<tr>
<td>WITNESS</td>
<td>1439</td>
<td>409.3</td>
</tr>
</tbody>
</table>

Table 3: Comparison 2, results task 2

3.3 Results Parallel Comparison

Until now, three solutions (one with only one of three tasks implemented) have been received by editors. Two solutions come from the Technical University of Vienna, one from the University of Glasgow.

The first solution (published in SNE 10 as a sample solution) was programmed directly in the programming languages FORTRAN and „C”, using the message passing system „PVM”. The programs were developed on an IBM RS6000-cluster (8 workstations) connected by a Token-Ring network and using PVM version 3.2.6.

The second solution came from the University of Glasgow using the continuous system simulation tool SLIM developed there. The hardware used was a Parsytec Supercluster, consisting of several Transputer working with the PARIX operating system. By now, only the first comparison could be provided (Monte Carlo simulation, master-slave approach).

The third solution came from TU Vienna: The three tasks were implemented within the parallel simulation language MOSIS developed there. The hardware used was the 20-transputer system Cogent XTM with operating system QIX and communication system „Kernel Linda”. MOSIS itself provides the communication between the processors (simulation tasks) and can work with different operating communication systems (Linda, PVM, PC’s etc.).
The MOSIS models (CSSL-type) are compiled to "C" and linked to the run time system. Eight processors were used for comparability with the C+FORTRAN/PVM solution.

The first task, the Monte-Carlo simulation, is hierarchically structured (fig. 2). In two cases the solutions sent in, did not achieve a linear increase of time because of communication overhead: A speed-up factor of 4.4 (MOSIS - because of message polling within Linda) and 5.5 (direct programming in PVM). The PARIX solution achieved nearly a linear speed-up factor with different numbers of processors, but from 16 processors on the communication becomes a significant bottleneck with 7.9% of simulation time.

The second sample gave similar results at the first and the third solution (PARIX solution expected in SNE 12): the coupled predator-prey system became significantly slower with a parallel implementation (communication at each integration step resulted in solutions 20 times slower than the serial example). When communication was cut (only each n-th simulation step), the "speed-up factor" could reach 0.61 to 0.77. Parallelisation by means of submodels results in too small subtask with too high communication overhead (fig. 2).

The third sample (partial differential equation) gave different results with the first and the third comparison (no PARIX solution). While the PVM factor was only 0.72 with communication every integration step, the MOSIS solution proved a value of 4.33 with 8 processors. When communication was cut, this could be improved to 3.2 (PVM) and 5.6 (MOSIS).

5. DEVELOPMENTS AND TRENDS

The results of the comparisons give an interesting insight into the development of existing languages and tools. Although within the commercial tools the US market is the leading one, there are interesting new developments in Europe:

- In continuous simulation there are projects for normalizing the model description in order to use model descriptions of different languages in one simulation environment.

In more detail, the following trends (developments vs. problems) can be seen in continuous simulation:

- Implicit model descriptions - Loss of input-output relations
- Submodel features - Conflicts with macro features
- Graphical model descriptions - Loss of segment structure
- Graphical preprocessors - Too many generated equations
- Sophisticated integr. algorithms - Overhead for 80%
- State event handling - Depending on modeling
- New analysis methods - CSSL structure to weak
- Separation model-experiment - Powerful runtime system
- Windows - Implementations - Loss of speed, esp. on PC

In discrete simulation, each software is implemented with on an event mechanism. Developments and problems are similar to continuous simulation, but there are additional aspects:

- Petri net modeling - Implementation with events
- Object oriented approaches - Powerful hardware (no PCs)
- Predefined strategies - Description of complex strategies
- Interfaces for non-expert users - Validation of models
- Improved animation - Simulation = Video game?

Parallel techniques may offer advantages for certain kinds of problems. In case of hierarchically structured tasks the use of parallel processors promise the best results (1st task). But if a problem is relatively small, and the subproblems are connected to each other, parallelisation results in lower speed (2nd task). In case of bigger subtasks as in case of the 3rd task the parallelisation results in a speed up - the subproblems are relatively big, and they are weakly coupled.

Recent and future developments let hope for a model-independent features for automatic parallelisation in high level simulation languages.

---

**Fig. 2: Parallelisation for Monte-Carlo simulation**

```
Master
Calculation of average

Worker 1  Worker 2  Worker 3  ...  Worker 8
x(t,d)    x(t,d)    x(t,d)    ...    x(t,d)
```

---

23
Comparison 2: Flexible Assembly System

A new comparison in this issue deals with discrete simulation, a flexible assembly system. We invite all institutes and companies developing or distributing simulation software to participate in this comparison:

Please, simulate the model described and send a report to the editors in the following form:

- short description of the language
- model description (source code, diagram, ...)
- results of the tasks with experimentation comments
- approx. 1 page A4

Reports will be published in the next issues of EUROSIM - Simulation News Europe.

New comparisons will be prepared for the next issues, too. As it is difficult to find suitable "simple" models and relevant tasks we would like to ask you to contact the editors if you have an idea for a model to be compared in different simulation languages.

The following example of a flexible assembly system has been chosen because it checks two important features of discrete event simulation tools:

- the possibility to define and combine submodels,
- the method to describe complex control strategies.

The model consists of a number of almost identical submodels of the following structure (figure 1):

![Figure 1](image)

Two parallel conveyor belts, B1 and B2, are linked together at both ends. An assembly station Ax is placed at B2. Pallets are coming in on belt B1. If they are to be processed in Ax they are shifted in Sx to B2 and possibly enter a queue in front of Ax. If there is no more empty buffer space on B2 or the pallet is not to be processed in Ax it continues its way along B1. Parts that have been processed in Ax are shifted back to B1 in Sy, having priority over those coming from the left on B1.

The total system now consists of 8 of these subsystems, varying in length, operation and operation time (see figure 2). Between two subsequent subsystems there is a space of 0.4 m, whereas pallets from the third subsystem A2 can be shifted directly to A3, and from A6 directly to A1. The shifting parts, however, cannot function as buffers, i.e. a pallet can only enter an Sx if it can leave it immediately.

![Figure 2](image)

Table 1 shows the operation time of each station, the total length of B1 and the length of the buffer in front of the station.

<table>
<thead>
<tr>
<th>Station</th>
<th>Operation time (sec.)</th>
<th>Length of B1 (m)</th>
<th>Length of buffer in front of station (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
<td>15</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>A 2</td>
<td>60</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A 3</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A 4</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A 5</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A 6</td>
<td>30</td>
<td>2.0</td>
<td>1.2</td>
</tr>
</tbody>
</table>

There are three identical stations A2 in the system, because the operation in A2 takes much longer than the other operations.

Unprocessed parts are put on pallets in A1. They can either be processed in A2 first, and then in A3, A4, A5, or in A3, A4, A5 first, and then in A2. The sequence of operations among A3, A4, and A5 is arbitrary. Station A6 is a substitute for any of the stations A3, A4, A5, i.e. whenever one of these stations is down, or the buffer in front of it is free, the corresponding operation can be executed in A6. Finished parts are unloaded in A1, unfinished parts enter another circle.

All conveyors are running with a speed of 18 m/min., any shifting takes 2 sec., and pallet length is 0.36 m. Assuming that no station ever has a breakdown, the optimum number of pallets in the system is to be found. Therefore the total throughput and the average throughput time of the parts have to be evaluated, when 20, 40, and 60 pallets are circulating in the system.

To simplify comparison of results we suggest starting simulation experiments with empty pallets and collecting data from the 120th to the 600th minute (8 hours).
Comparison 2: Flexible Assembly System

The following example of a flexible assembly system has been chosen because it checks two important features of discrete event simulation tools:

- the possibility to define and combine submodels,
- the method to describe complex control strategies.

The model consists of a number of almost identical submodels of the following structure (figure 1):

Figure 1

Two parallel conveyor belts, B1 and B2, are linked together at both ends. An assembly station Ax is placed at B2. Pallets are coming in on belt B1. If they are to be processed in Ax they are shifted in Sx to B2 and possibly enter a queue in front of Ax. If there is no more empty buffer space on B2 or the pallet is not to be processed in Ax it continues its way along B1. Parts that have been processed in Ax are shifted back to B1 in Sy, having priority over those coming from the left on B1.

The total system now consists of 8 of these subsystems, varying in length, operation and operation time (see figure 2). Between two subsequent subsystems there is a space of 0.4 m, whereas pallets from the third subsystem A2 can be shifted directly to A3, and from A6 directly to A1. The shifting parts, however, cannot function as buffers, i.e. a pallet can only enter an Sx if it can leave it immediately.

Unprocessed parts are put on pallets in A1. They can either be processed in A2 first, and then in A3, A4, A5, or in A3, A4, A5 first, and then in A2. The sequence of operations among A3, A4, and A5 is arbitrary. Station A6 is a substitute for any of the stations A3, A4, A5, i.e. whenever one of these stations is down, or the buffer in front of it is free, the corresponding operation can be executed in A6. Finished parts are unloaded in A1, unfinished parts enter another circle.

All conveyors are running with a speed of 18 m/min., any shifting takes 2 sec., and pallet length is 0.36 m. Assuming that no station ever has a breakdown, the optimum number of pallets in the system is to be found. Therefore the total throughput and the average throughput time of the parts have to be evaluated, when 20, 40, and 60 pallets are circulating in the system.

To simplify comparison of results we suggest starting simulation experiments with empty pallets and collecting data from the 120th to the 600th minute (8 hours).

Table 1 shows the operation time of each station, the total length of B1 and the length of the buffer in front of the station.

<table>
<thead>
<tr>
<th>Station</th>
<th>Operation time (sec.)</th>
<th>Length of B1 (m)</th>
<th>Length of buffer in front station (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>15</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>A2</td>
<td>60</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A3</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A4</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A5</td>
<td>20</td>
<td>1.6</td>
<td>0.8</td>
</tr>
<tr>
<td>A6</td>
<td>30</td>
<td>2.0</td>
<td>1.2</td>
</tr>
</tbody>
</table>

There are three identical stations A2 in the system, because the operation in A2 takes much longer than the other operations.
Remarks

In number 1 of EUROSIM - Simulation News Europe (March 1991) we had proposed to test discrete event simulators using an example flexible assembly system. Some letters from readers however made it clear that the description of the system has been somewhat incomplete. We therefore try to answer the open questions and ask you not to hesitate to contact us if any other questions arise.

What follows is not a full definition of the model but only some details in addition to the description in EUROSIM - Simulation News Europe 1.

1. The subsystems contain two parallel conveyors B1 and B2. The total length of B2 between Sx and Sy is given in table 1. Sx and Sy themselves are 0.4 m wide. A pallet can either pass Sx or Sy without any delay with its normal speed along B1 or can be shifted to B2 in 2.0 sec. The lengths of B1 and B2 are the same. B2, however, is divided into three parts: the buffer in front of the station (its length being given in table 1), the station's positioning unit of length 0.4 cm, and the buffer behind the station (the remaining part of B2).

2. The conveyors themselves can function as buffers. Pallets can queue up in front of the stations or in front of Sx and Sy but the conveyor will move on with its normal speed. Also during the shifting of one pallet or while it is being processed on one of the positioning units in an Ax the other pallets are being transported without any delay. The capacity of each buffer can be easily calculated by dividing its length by the pallet length (0.36 cm). Of course, only integers are feasible results.

3. If the buffer in front of Ax is full, all pallets move on along B1 even if they require processing in Ax. They may either be processed when the pass Ax the next time, or they may be processed in A6 (if x = 3, 4, or 5).

4. The transportation time from Sx to Ax (i.e. its positioning unit) is not part of the operation time as given in table 1. The same holds for the transportation time from Ax to Sy.

5. In the beginning empty pallets are circulating in the system. Their positions on the conveyors B1 (not B2!) can be chosen randomly. Unprocessed pieces are put on them in A1 (operation time 7.5 sec), and finished parts are unloaded in A1, too (operation time 7.5 sec, hence total time for load/unload is 15.0 sec). A1 is only used for these load/unload operations.

6. Pallets are being brought to A6 if they have not undergone one or more of the operations of A3, A4, or A5. They can then undergo all the missing operations at a time.

We hope we have clarified the open questions now. Again, if any other questions come up during modelling, don't hesitate to contact us.

Finally we ask everybody who has tried or will try to model the system to send us a report on the experiences he/she has made even if no results have been achieved. We believe it is as important to learn why certain approaches or tools are not appropriate, as it is to learn how other colleagues have solved the problem. Unfortunately scientists do not communicate their unsolved problems and unsuccessful approaches as freely as they communicate their solutions. Please help us to change this and tell us if you have not been able to model this system with a simulation tool, and what the difficulties were. Thank you very much!

Contact:
J. Krauth, BIBA Bremer Institut für Betriebstechnik und angewandte Arbeitswissenschaft, Postfach 33 05 60, D-2800 Bremen 33, Tel.: +49 421 22009-51, Fax: +49 421 22009-79
Comparison 2 - PS SIMDIS

Description of PS SIMDIS OS/ES

PS SIMDIS is a block-oriented simulation language for discrete systems. It is a system of the GPSS family. Model elements are divided in static (storages, facilities, chains, queues...) and dynamic elements (transactions). Transactions can be generated and annihilated during the simulation process. A PC version (SIM-PC) of this simulation language was developed at the Department of Informatics at the Technical University Magdeburg. This PC-version consists of all features of the PS SIMDIS OS/ES and additional components for animation.

Model description

The model was programmed on a 80386 SX AT-type system. The parallel conveyor belts are modeled as static elements (storages). The capacities of these storages are equivalent with the length of the belts. The static element 'facility' represents an assembly station. The palettes are the dynamic elements in the system. The combination of the properties of the transactions, facilities and storages controls the whole system.

Results

The system was not able to work with more than 51 palettes. The optimum number of palettes is 40. The figures show the results of some simulation passes.

**SYN ADVANCE AVFVELT shifting the palette**

* Here begins the belt 3.

**ENLI ENTER SRH.AVPFL, L OK, entering this belt 3**

**RELEASE WAIT**

**ADVANCE SRH.AVPFL**

**GATE LR**

**SETZ WAIT**

**ENLI LEAVE SRH.AVPFL, L leaving the belt**

* This is the end of this belt.

**ENLI ADVANCE AVFVELT**

* This is subsystem C

**MODE TEST E**

**PB1, HCAI**

* We have to skip – no part loaded

**TEST E**

**PBD, HCAI**

* Can skip – nothing to do

* OK, we have something to do.

**TEST E**

**AVFVALU, HCAI**

* OK, let's do it.

**TEST GE**

**SRH2.EVPFL, HCAI**

**RELEASE WAIT**

* The buffer in front of station C is not full.

**SUC ADVANCE AVFVELT**

* Here begins the buffer in front of station C.

**BCH2 ENTER SRH.AVPFL, L OK, palette enters the buffer**

**ADVANCE SRH2**

* New palette to the station

* Station C is empty.

**SETZ SR**

* Entering the station C

**ENLI ADVANCE AVFVELT**

**BCH2 LEAVE SRH.AVPFL, L leaving the buffer**

* Here is the end of the buffer in front of station C.

Part of the model description

...
REALLOCATE BLO,300
REALLOCATE VAR,50

PAL_L VARIABLE 36 length of a palette (cm)
PAL_N VARIABLE 40 number of palettes in system
CON_S VARIABLE 30 speed of a conveyor (cm/sec)

* We have to define the shortest time in the system as a conclusion of the
* conveyors speed. For that reason we have to define one step of time as:
* 1 &t:=1/30 sec

DTIME VARIABLE AV#CON_S helps by computing
SHI_T VARIABLE AV#DTIME*2 shifting takes 2 sec.
WAY_L VARIABLE 40 distance between two subsystems

* A palette have to pass a distance of 200 cm means it have to wait 200 &t.

SIMTI VARIABLE AV#DTIME*60*60*10  \{ 10 hours \} simulation time (sec)

FIRST VARIABLE  INITIAL  AV#DTIME*60*60*2  first observation
INITIAL LR#COLLE

INITIAL SK#SAB1,120 length of belt B1 (cm)
INITIAL SK#SBB1,160
INITIAL SK#SCB1,160
INITIAL SK#SDB1,160
INITIAL SK#SEB1,160
INITIAL SK#SF21,160
INITIAL SK#SGB1,160
INITIAL SK#SHB1,200

INITIAL SK#SAB2,120 length of buffer in front of a station (cm)
INITIAL SK#SBB2,80 belt B2
INITIAL SK#SCB2,80
INITIAL SK#SDB2,80
INITIAL SK#SEB2,80
INITIAL SK#SF22,80
INITIAL SK#SGB2,80
INITIAL SK#SHB2,120
LBA VARIABLE SK#SAB1-SK#SAB2-AV#PAL_L
INITIAL SK#LBA,AV#LBA length of buffer in back of a station (cm)

LBB VARIABLE SK#SBB1-SK#SBB2-AV#PAL_L
INITIAL SK#LBB,AV#LBB

LBC VARIABLE SK#SCB1-SK#SCB2-AV#PAL_L
INITIAL SK#LBC,AV#LBC

LBD VARIABLE SK#SDB1-SK#SDB2-AV#PAL_L
INITIAL SK#LBD,AV#LBD

LBE VARIABLE SK#SEB1-SK#SEB2-AV#PAL_L
INITIAL SK#LBE,AV#LBE

LBF VARIABLE SK#SFB1-SK#SFB2-AV#PAL_L
INITIAL SK#LBF,AV#LBF

LBG VARIABLE SK#SGB1-SK#SGB2-AV#PAL_L
INITIAL SK#LGB,AV#LGB

LBH VARIABLE SK#SHB1-SK#SHB2-AV#PAL_L
INITIAL SK#LHH,AV#LHH

INITIAL LR#TORA
INITIAL LR#TORB
INITIAL LR#TORC
INITIAL LR#TORD
INITIAL LR#TORE
INITIAL LR#TORF
INITIAL LR#TORG
INITIAL LR#TORH

OPTIM FUNCTION PB9,D6 operation time of a station (sec)
1/15,2/60,3/20,4/20,5/20,6/30

OTIME VARIABLE FM#OPTIM*AV#DTIME gives the real operation time of a station

DONE VARIABLE PB1*PB2*PB3*PB4*PB5

SEQU VARIABLE PB8:4 Is there anything to do ?
Is the working sequence OK ?

INITIAL XF#AUS1,0
INITIAL XF#EIN1,0
INITIAL XF#CIC1,0
INITIAL XF#AUS2,0
INITIAL XF#EIN2,0
INITIAL XF#CIC2,0

Now we create the palettes.

GENERATE AV#PAL_L,0,0,AV#PAL_N,0,9PB

* Entering the system.

TRANSFER ,MMM1

MMM0 TEST G AV#DONE,0,MMM1 there is nothing to do
ASSIGN 1-8,0,,PB get assembled part from palette

CNTR SAVEVALUE AUS1+,1,XF count all assembled parts
GATE LS COLLE,=*2
SAVEVALUE AUS2+,1,XF count assembled parts after 2nd hour

* This is subsystem A

MM1 SEIZE OEBRZ
SAVEVALUE CYC1+,1,XF count all palettes
GATE LS COLLE,=*2
SAVEVALUE CYC2+,1,XF count palettes after 2nd hour
TEST E PBL,0,BAA1 we can skip this subsystem

* Oh, there is no part loaded. Try to load!

TEST GE SP#SAB2,AV#PAL_L,BAA1

* The buffer in front of station A is not full.

RELEASE OEBRZ
SXA ADVANCE AV#SHI_T shifting palette to belt B2

* Here begins the buffer in front of station A.

BAA2 ENTER SAB2,AV#PAL_L OK, palette enters the buffer
ADVANCE SK#SAB2 move palette to the station

* Station A is empty.

SEIZE STA entering the station A
ADVANCE AV#PAL_L
BAE2 LEAVE SAB2,AV#PAL_L leaving the buffer

* Here is the end of the buffer in front of station A.

* ASSIGN 9,1,,PB store number of current station type to get
STA ADVANCE AV#OTIME the operation time of this station
now we have to work, assembly parts
SAVEVALUE EIN1+,1,XF count loaded parts
GATE LS COLLE,=*2
SAVEVALUE EIN2+,1,XF count all loaded parts after 2nd hour
ASSIGN 1,1,,PB write the signum of a station of type 1

* Here begins the buffer at the back of station A.

LBA ENTER LBA,AV#PAL_L enter buffer after station
ADVANCE AV#PAL_L
RELEASE STA leaving the station
XLA VARIOUS AV#LBA-AV#PAL_L
ADVANCE AV#XLA
LOGIC S TORA close the second way
GATE FNU WAYA try to leave station
LEAVE LBA,AV#PAL_L leave buffer after station

* Here is the end of the buffer at the back of station A.

SYA ADVANCE AV#SHI_T shifting the palette

SEIZE WAYA between two subsystems
LOGIC R TORA open second way
TRANSFER ,AAAA

* Here begins the belt B1.

BAA1 RELEASE OFHRZ OK, entering this belt B1
ENTER SAB1,AV#PAL_L
ADVANCE SK#SAB1
GATE LR TORA
SEIZE WAYA

BAA1 LEAVE SAB1,AV#PAL_L leaving the belt

* This is the end of this belt.

AAAA ADVANCE AV#WAY_L between two subsystems

* This is subsystem B

MMH2 TEST E PB1,1,BBA1 we have to skip - no part loaded
TEST E PB2,0,BBA1 we can skip - nothing to do

* OK. We have something to do.

TEST E AV#SEQU,0,BBA1 sorry, not allowed to do anything

* OK. Let's do it.

TEST GE SP#SBB2,AV#PAL_L,BBA1
RELEASE WAYA

* The buffer in front of station B is not full.

SXB ADVANCE AV#SHI_T shifting palette to belt B2

* Here begins the buffer in front of station B.

BBA2 ENTER SBB2,AV#PAL_L OK, palette enters the buffer
ADVANCE SK#SBB2 move palette to the station
* Station B is empty.

SEIZE STB entering the station B
ADVANCE AV#PAL_L
BBE2 LEAVE SBB2,AV#PAL_L leaving the buffer

* Here is the end of the buffer in front of station B.

ASSIGN 9,2,,PB store number of current station type to get the operation time of this station

STB ADVANCE AV#OTIME now we have to work, assembly parts
ASSIGN 2,2,,PB write the signum of a station of type 2

LBB ENTER LBB,AV#PAL_L enter buffer after station
ADVANCE AV#PAL_L
RELEASE STB leaving the station

XLB VARIABLE AV#LBB-AV#PAL_L
ADVANCE AV#XLB

LOGIC S TORB close the second way
GATE FNU WAYB try to leave station
LEAVE LBB,AV#PAL_L leave buffer after station

* Here is the end of the buffer at the back of station B.

SYB ADVANCE AV#SHI_T shifting the palette

SEIZE WAYB
LOGIC R TORB open second way
TRANSFER ,BBB

* Here begins the belt B1.

BBA1 ENTER SBB1,AV#PAL_L OK, entering this belt B1
RELEASE WAYA
ADVANCE SK#SBB1
GATE LR TORB
SEIZE WAYB

BBE1 LEAVE SBB1,AV#PAL_L leaving the belt

* This is the end of this belt.

BBBB ADVANCE AV#WAY_L between two subsystems

* This is subsystem C

MM3 TEST E PB1,1,BCA1 we have to skip - no part loaded
TEST E   PB2,0,BCA1  we can skip - nothing to do
* OK. We have something to do.
  TEST E   AV#SEQU,0,BCA1  sorry, not allowed to do anything
* OK. Let's do it.
  TEST GE   SP#SCB2,AV#PAL_L,BCA1
  RELEASE   WAYB
* The buffer in front of station C is not full.
SXC ADVANCE   AV#SHI_T  shifting palette to belt B2
* Here begins the buffer in front of station C.
  *  
  BCA2 ENTER   SCB2,AV#PAL_L  OK, palette enters the buffer
  ADVANCE   SK#SCB2  move palette to the station
* Station C is empty.
  SEIZE   STC  entering the station C
  ADVANCE   AV#PAL_L
  BCE2 LEAVE   SCB2,AV#PAL_L  leaving the buffer
* Here is the end of the buffer in front of station C.
  *  
  ASSIGN   9,2,,PB  store number of current station type to get
  STC ADVANCE   AV#OTIME  the operation time of this station
  ASSIGN   2,2,,PB  now we have to work, assembly parts
  LBC ENTER   LBC,AV#PAL_L  write the signum of a station of type 2
  ADVANCE   AV#PAL_L
  RELEASE   STC  enter buffer after station
  XLC VARIABLE   AV#LBC-AV#PAL_L  leaving the station
  ADVANCE   AV#XLC
  LOGIC S   TORC  close the second way
  GATE FNU   WAYC  try to leave station
  LEAVE   LBC,AV#PAL_L  leave buffer after station
* Here is the end of the buffer at the back of station C.
*  
  SYC ADVANCE   AV#SHI_T  shifting the palette
  *  
  SEIZE   WAYC  between two subsystems
  LOGIC R   TORC  open second way
  TRANSFER   CCCC
* Here begins the belt B1.

BCA1 ENTER SCBI,AV#PAL_L OK, entering this belt B1
RELEASE WAYB
ADVANCE SK#SCB1
GATE LR TORC
SEIZE WAYC
BCE1 LEAVE SCBI,AV#PAL_L leaving the belt

* This is the end of this belt.

* CCCC ADVANCE AV#WAY_L between two subsystems

* This is subsystem D

MMD4 TEST E PB1,1,BDA1 we have to skip - no part loaded
TEST E PB2,0,BDA1 we can skip - nothing to do

* OK. We have something to do.

TEST E AV#SEQU,0,BDA1 sorry, not allowed to do anything

* OK. Let's do it.

TEST GE SP#SDB2,AV#PAL_L,BDA1
RELEASE WAYC
* The buffer in front of station D is not full.

SXD ADVANCE AV#SHI_T shifting palette to belt B2

* Here begins the buffer in front of station D.

BDA2 ENTER SDB2,AV#PAL_L OK, palette enters the buffer
ADVANCE SK#SDB2 move palette to the station

* GATE FNU STD

* Station D is empty.

SEIZE STD entering the station D
ADVANCE AV#PAL_L
BDE2 LEAVE SDB2,AV#PAL_L leaving the buffer

* Here is the end of the buffer in front of station D.

* ASSIGN 9.2,,PB store number of current station type to get the operation time of this station

STD ADVANCE AV#OTIME now we have to work, assembly parts
ASSIGN 2.2,,PB write the signum of a station of type 2
**EUROSIM COMPARISON 2 - SOLUTIONS AND RESULTS**

```
ENTER LBD,AV#PAL_L
ADVANCE AV#PAL_L
RELEASE STD
  enter the buffer in back of station
  leaving station

XLD VARIABLE AV#LBD-AV#PAL_L
ADVANCE AV#XLD
LOGIC S TORD
  close second way
SEIZE OEHRD
TEST E PB3,3,CHEE1
  which belt we can choose

* We must not enter the next station - only belt B1.

TEST GE SP#SEB1,AV#PAL_L
TRANSFER ,CHEE2

* We can choose between both belts.

SPE VARIABLE (SP#SEB2+SP#SEB1)/2
CHEE1 TEST GE AV#SPE,AV#PAL_L

CHEE2 RELEASE OEHRD
LEAVE LBD,AV#PAL_L
  leaving the station

* Here is the end of the buffer at the back of station D.

SYD ADVANCE AV#SHI_T
  shifting the palette

* Here begins the belt B1.

BDA1 ENTER SDB1,AV#PAL_L
  OK, entering this belt B1
RELEASE WAYC
ADVANCE SK#SDB1
GATE LR TORD
SEIZE OEHRD
TEST E PB3,3,CHEE3
  which belt we can choose

* We must not enter the next station - only belt B1.

TEST GE SP#SEB1,AV#PAL_L
TRANSFER ,BDE1

* We can choose between both belts.

CHEE3 TEST GE AV#SPE,AV#PAL_L
  waiting - next belt is not full
BDE1 RELEASE OEHRD
```
LEAVE SDB1,AV#PAL_L leaving the belt

* This is the end of this belt.

* DDDD CONTINUE

* This is subsystem E

** M#M5 TEST E PB1,1,BEA1 we have to skip - no part loaded
    TEST E PB3,0,BEA1 we can skip - nothing to do

* OK. We have something to do.

    TEST GE SP#SEB2,AV#PAL_L,BEA1

* The buffer in front of station E is not full.

    SXE ADVANCE AV#SHI_T shifting palette to belt B2

* Here begins the buffer in front of station E.

    BEA2 ENTER SEB2,AV#PAL_L OK, palette enters the buffer
    ADVANCE SK#SEB2 move palette to the station

* Station E is empty.

    SEIZE STE entering the station E
    ADVANCE AV#PAL_L leaving the buffer

* Here is the end of the buffer in front of station E.

    ASSIGN 9,3,,PB store number of current station type to get
    the operation time of this station

    STE ADVANCE AV#OTIME now we have to work, assembly parts
    ASSIGN 3,3,,PB write the signum of a station of type 3
    ASSIGN 8+,1,,PB I'm a part of three.

    LBE ENTER LBE,AV#PAL_L enter buffer after station
    ADVANCE AV#PAL_L leaving the station
    RELEASE STE

    XLE VARIABLE AV#LBE-AV#PAL_L
    ADVANCE AV#XLE

    LOGIC S TORE close the second way
    GATE FNU WAYE try to leave station
    LEAVE LBE,AV#PAL_L leave buffer after station

* Here is the end of the buffer at the back of station E.
SYE ADVANCE AV#SHI_T shifting the palette
SEIZE WAYE between two subsystems
LOGIC R TORE open second way
TRANSFER ,EEE

* Here begins the belt B1.

BEA1 ENTER SEB1,AV#PAL_L OK, entering this belt B1
ADVANCE SK#SEB1
GATE LR TORE
SEIZE WAYE
BEE1 LEAVE SEB1,AV#PAL_L leaving the belt

* This is the end of this belt.

EEE ADVANCE AV#WAY_L between two subsystems

* This is subsystem F

MMM6 TEST E PB1,1,BFA1 we have to skip - no part loaded
TEST E PB4,0,BFA1 we can skip - nothing to do

* OK. We have something to do.

TEST GE SP#SFB2,AV#PAL_L,BFA1
RELEASE WAYE
* The buffer in front of station F is not full.

SXF ADVANCE AV#SHI_T shifting palette to belt B2

* Here begins the buffer in front of station F.

BFA2 ENTER SFB2,AV#PAL_L OK, palette enters the buffer
ADVANCE SK#SFB2 move palette to the station

* Station F is empty.

SEIZE STF entering the station F
ADVANCE AV#PAL_L
BFE2 LEAVE SFB2,AV#PAL_L leaving the buffer

* Here is the end of the buffer in front of station F.

ASSIGN 9,4,PB store number of current station type to get
the operation time of this station
STF ADVANCE AV#OTIME now we have to work, assembly parts
ASSIGN 4,4,PB write the signum of a station of type 4
ASSIGN B+,1,,PB I'm a part of three.
ENTER LBF,AV#PAL_L enter buffer after station
ADVANCE AV#PAL_L
RELEASE STF leaving the station

XLF VARIABLE AV#LBF-AV#PAL_L
ADVANCE AV#XLF

LOGIC S TORF close the second way
GATE FNU WAYF try to leave station
LEAVE LBF,AV#PAL_L leave buffer after station

* Here is the end of the buffer at the back of station F.

SYF ADVANCE AV#SHL_T shifting the palette

SEIZE WAYF
LOGIC R TORF open second way
TRANSFER .FFFF

* Here begins the belt B1.

BFA1 ENTER SFB1,AV#PAL_L OK, entering this belt B1
RELEASE WAYE
ADVANCE SK#SFB1
GATE LR TORF
SEIZE WAYF

BFE1 LEAVE SFB1,AV#PAL_L leaving the belt

* This is the end of this belt.

FFFF ADVANCE AV#WAY_L between two subsystems

* This is subsystem G

MMM7 TEST E PB1,1,BGA1 we have to skip - no part loaded
TEST E PBS,0,BGA1 we can skip - nothing to do

* OK. We have something to do.

TEST GE SP#SGB2,AV#PAL_L,BGA1
RELEASE WAYF

* The buffer in front of station G is not full.

SXG ADVANCE AV#SHL_T shifting palette to belt B2

* Here begins the buffer in front of station G.

BGA2 ENTER SGB2,AV#PAL_L OK, palette enters the buffer
ADVANCE SK#SGB2 move palette to the station
* GATE FNU STG

* Station G is empty.

SEIZE STG entering the station G
ADVANCE AV#PAL_L
BGE2 LEAVE SGB2,AV#PAL_L leaving the buffer

* Here is the end of the buffer in front of station G.

ASSIGN 9,5,,PB store number of current station type to get the operation time of this station

* STG ADVANCE AV#OTIME now we have to work, assembly parts
ASSIGN 5,5,,PB write the signum of a station of type 5
ASSIGN 8+,1,,PB I'm a part of three.
LBG ENTER LBG,AV#PAL_L enter buffer after station
ADVANCE AV#PAL_L
RELEASE STG leaving the station

XLG VARIABLE AV#LBG-AV#PAL_L
ADVANCE AV#XLG

LOGIC S TONG close the second way
GATE FNU WAYG try to leave station
LEAVE LBG,AV#PAL_L leave buffer after station

* Here is the end of the buffer at the back of station G.

SYG ADVANCE AV#SHI_T shifting the palette

* SEIZE WAYG between two subsystems
LOGIC R TONG open second way
TRANSFER ,GGGG

* Here begins the belt B1.

BGAI ENTER SGB1,AV#PAL_L OK, entering this belt B1
RELEASE WAYF
ADVANCE SK#SGB1
GATE LR TONG
SEIZE WAYG
BGE1 LEAVE SGB1,AV#PAL_L leaving the belt

* This is the end of this belt.

GGGG ADVANCE AV#WAY_L between two subsystems

* This is subsystem II
* OK. We have something to do.

    TEST E    PB3,0,HHH1    nothing to do for type 3

* OK. I'll do your job 'type 3'

    ASSIGN  3,3,,PB
    TRANSFER ,HHH3

    HHH1 TEST E    PB4,0,HHH2    nothing to do for type 4

* OK. I'll do your job 'type 4'

    ASSIGN  4,4,,PB
    TRANSFER ,HHH3

* Not '3', not '4' ... OK. Let's do it like 'type 5'.

    HHH2 ASSIGN  5,5,,PB    write the signum of a station of type 5

    HHH3 TEST GE SP#SHB2,AV#PAL_L,BHA1
    RELEASE WAYG

* The buffer in front of station H is not full.

    SXH ADVANCE AV#SHI_T    shifting palette to belt B2

* Here begins the buffer in front of station H.

    BHA2 ENTER SHB2,AV#PAL_L
    ADVANCE SK#SHB2  OK, palette enters the buffer
                   move palette to the station

* Station H is empty.

    SEIZE STH
    ADVANCE AV#PAL_L    entering the station H

    BHE2 LEAVE SHB2,AV#PAL_L    leaving the buffer

* Here is the end of the buffer in front of station H.

*  ASSIGN  9,6,,PB  store number of current station type to get
                  the operation time of this station
  ADVANCE AV#OTTME
  ASSIGN  8+1,,PB  now we have to work, assembly parts
       I'm a part of three.

    LBH ENTER LBH,AV#PAL_L
    ADVANCE AV#PAL_L
    RELEASE STH

    XLH VARIABLE AV#LBH-AV#PAL_L
    ADVANCE AV#XLH
LOGIC S TORH
SEIZE OEHRI
TEST E PB1,1,CHEA1

* We must not enter next station.

TEST GE SP#SAB1,AV#PAL_L
TRANSFER ,CHEA2

* We can choose between both belts.

SPA VARIABLE (SP#SAB2+SP#SAB1)/2
CHEA1 TEST GE AV#SPA,AV#PAL_L

CHEA2 RELEASE OEHRI
LEAVE LNH,AV#PAL_L leaving the station

* Here is the end of the buffer at the back of station B.

*---------------------------------------------------------------------*
SYH ADVANCE AV#SHI_T shifting the palette

*---------------------------------------------------------------------*

LOGIC R TORH
TRANSFER ,BHHH

* Here begins the belt B1.

*---------------------------------------------------------------------*
BHA1 ENTER SHB1,AV#PAL_L OK, entering this belt B1
RELEASE WAYG
ADVANCE SK#SHB1
GATE LR TORH
SEIZE OEHRI
TEST E PB1,1,CHEA3 which belt can we choose

* We must not enter next station.

TEST GE SP#SAB1,AV#PAL_L
TRANSFER ,BHE1

* We can choose between both belts.

CHEA3 TEST GE AV#SPA,AV#PAL_L

BHE1 RELEASE OEHRI
LEAVE SHB1,AV#PAL_L leaving the belt

* This is the end of this belt.

*---------------------------------------------------------------------*

BHHH TRANSFER ,HHHH

*---------------------------------------------------------------------*

GENERATE ,,1,127,0PB,0PH,1PF We'll control the time of simulation.
SPLIT 1, OBSER
ADVANCE AV#SIMTI
TERMINATE 1

The time is over.

OBSER ADVANCE AV#FIRST
LOGIC S COLLE
TERMINATE

Now we are collecting data.

START 1

Run!
Comparison 2 - DOSIMIS3

1. Description of language

The simulator DOSIMIS3 used for solving the given task is an element oriented simulator. DOSIMIS3 is based upon chronological event lists and is particularly designed for analysing discrete Material Flow Systems (MFS). Being element oriented, DOSIMIS3 does not provide a simulator language as other simulation packages. Instead, it is possible to build a complete MFS by combining so called elements on the screen. The 20 standard elements are similar to those of real MFS and they can be placed via a menu oriented, graphic user interface. The characteristics of the elements are to be specified by parameters (length, speed, etc.) within a parameter mask, which can be popped up for each single element. Elements with more than one input or output can be equipped with a local intelligence by which strategies such as FIFO, priorities of input, etc. can be realized. Postrun animation and presentation of simulation results in tables and graphs allow for the evaluation of the simulation results. If control of the model via standard DOSIMIS3 strategies is insufficient, decision tables can be employed to model functional and informational interrelationships. These decision tables can be entered via the graphic user interface, such that no programming skills are required. In case of highly complex problems, the system's control can be improved by additionally using the DOSIMIS3 programming interface (PASCAL).

The given task was solved entirely by using decision tables. No extra programming was necessary.

2. Model Description

To solve this problem the following DOSIMIS3 elements were used

<table>
<thead>
<tr>
<th>elements</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 source *</td>
<td>create new parts</td>
</tr>
<tr>
<td>1 sink *</td>
<td>terminate completely processed parts</td>
</tr>
<tr>
<td>1 assembly element *</td>
<td>put unprocessed parts on pallets</td>
</tr>
<tr>
<td>1 disassembly element *</td>
<td>unload parts from pallets</td>
</tr>
<tr>
<td>7 workstations</td>
<td>assembly stations A2(3x), A3, A6</td>
</tr>
<tr>
<td>8 junctions</td>
<td>shiftplaces between conveyor belts B1-B2</td>
</tr>
<tr>
<td>8 discharging devices</td>
<td>sections of conveyor belts B1 and B2</td>
</tr>
<tr>
<td>30 buffer sections</td>
<td></td>
</tr>
</tbody>
</table>

* the station A1 consists of these elements

3. Parametrization of the elements

<table>
<thead>
<tr>
<th>length of a segment</th>
<th>0.4 m</th>
</tr>
</thead>
<tbody>
<tr>
<td>length of an object</td>
<td>0.36 m</td>
</tr>
<tr>
<td>conveying speed</td>
<td>0.3 m/s</td>
</tr>
</tbody>
</table>

Number of segments for each belt according to belt length in given task. The empty pallets were put in the buffer sections of all conveyor belts B1 and B2 before start.

4. Results and experimentation comments

<table>
<thead>
<tr>
<th>numbers of pallets</th>
<th>total throughput</th>
<th>average throughput time</th>
<th>min throughput time</th>
<th>max throughput time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>939</td>
<td>314.3</td>
<td>293.0</td>
<td>335.6</td>
</tr>
<tr>
<td>15</td>
<td>1350</td>
<td>327.5</td>
<td>225.3</td>
<td>413.0</td>
</tr>
<tr>
<td>20</td>
<td>1408</td>
<td>436.9</td>
<td>263.6</td>
<td>1222.3</td>
</tr>
<tr>
<td>30</td>
<td>1410</td>
<td>620.9</td>
<td>340.0</td>
<td>2436.3</td>
</tr>
<tr>
<td>35</td>
<td>1408</td>
<td>722.6</td>
<td>350.3</td>
<td>3160.7</td>
</tr>
<tr>
<td>40</td>
<td>1408</td>
<td>825.0</td>
<td>345.6</td>
<td>3145.3</td>
</tr>
</tbody>
</table>

Collecting data from 120 th to 600 th minute

- a) 20 is the optimum number of pallets in the system (see diagram below)
- b) with increasing number of pallets, the total throughput approaches the optimum value of 1440 parts. This figure is based on the following idea: The longest operation time of an assembly station is 20 s, (A2's is 60 s divided by the number of stations A2 equals 20 s again). Thus the best case will be: 8 hours divided by 20 s equals 1440 parts.
- c) With 60 pallets the system couldn't work because of a deadlock.

It seems to be possible to increase the number of pallets by changing the priority of the shift places Sy, thus, that pallets in B1 have priority over pallets on B2. The tradeoff would be longer throughput times.

For information and comments, please phone or fax or write to:

Ariane Beece or Michael Kluger, Fraunhofer Institute for Material Flow and Logistics, Emil Figgestr. 75, W - 4600 Dortmund, Germany, Tel: +49-(0)231 7549 171, Fax: +49-(0)231 7549 211
Comparison 2 - SIMAN

This report discusses the method used to simulate and optimize the Flexible Assembly/Manufacturing System using the Simulation Language SIMAN/CINEMA:

SIMAN is a general purpose SIMulation ANalysis program for modelling combined discrete continuous systems.

SIMAN is designed around a logical modelling framework that separates the model structure and experimental frame into two distinct elements. This allows different experimentation runs to be performed yet, keeping the control and flow logic unaltered in the model frame.

SIMAN runs on various types of computers and offers animation with the CINEMA system. This powerful tool allows not only a visual presentation of the system but is interactive with the debugger and allows rapid validation with the real world system.

A menu generator is available to allow the model builder to develop a series of menus so that the simulation can be run by an unskilled user. Output analysis is performed via the Output Processor.

Model description

For the purpose of this exercise the model was described as a Flexible Manufacturing (rather than Assembly) System. There is a 60 second drilling process at the A2 stations. Stations A3, A4 and A5 are mill, lathe and broach respectively. A6 is a multi-purpose (except drilling). The FMS model has some noteworthy features:

(a) The workstations are submodelled using the 'macro station' feature, allowing all 8 workstations to be modelled as one, as are all entrances to and exits from the conveyor belts.

The workstation belts are modelled as a single 'accumulating conveyor' SIMAN element. The loop conveyor is another.

(b) Two 'FINDJ' blocks determine the appropriate station with the lowest buffer level.

(c) Control for the system is via a 'BRANCH' block. In addition to the information from the FINDJ blocks, the branch statement determines whether the pallet has started on the mill/lathe/broach sequence. From this a decision is made to which process the pallet will visit next. Failing to find a station to visit, results in the pallet being conveyed to the next station and repeating the ‘FINDJ, BRANCH’ routine.

(d) All buffer levels before workstations, process times, number of pallets in system and conveyor dimensions/velocities can be adjusted in the experimental frame.

Experimentation results

The various pallet configurations were run in 'Batch Mode' and results of flowtime, cycle time and total throughput (number of pallets through system in 8 hours), recorded for 20, 40 and 60 pallets in the system. The SIMAN Output Processor was used to produce the graphs and associated confidence calculations and correlograms.

(a) Results from the first output show the system becomes congested at the 60 pallet level (the loop conveyor reaches its full capacity). The 20 pallet and 40 pallet runs yielded a 919 and 920 pallet throughput in 8 hours. The 20 pallet option is clearly more desirable as the work in progress and flowtimes are halved. More detailed experimentation was necessary around the 20 pallet figure to see if we could reduce these further.

(b) Figures 1 and 2 are plots of number of pallets in system against flowrate and throughput. It was decided 13 pallets are the optimum as it maintains the throughput of 919 pallets, yet has the lowest WIP figure and very low flowtime.

![Graph 1]

![Graph 2]

Remarks

Interpretation of whether the length of buffer in front of the work stations included the space for the pallet (while being processed) or not could produce different results. This would effect the congestion on the loop conveyor, throughput and flow times.

J.F.S. Heffeman, The CImulation Centre, Avon House, P.O. Box 46, Chippenham, Wiltshire SN15 1JH, England
A Flexible Assembly Model

This report discusses the method used to simulate and optimise the Flexible Assembly/Manufacturing System using the Simulation Language SIMAN/CINEMA.

SIMAN is a general-purpose Simulation Analysis program for modelling combined Discrete-Continuous systems.

(a) SIMAN is designed around a logical modelling framework that separates the model structure and experimental frame into two distinct elements. This allows different experimentation runs to be performed yet, keeping the Control & flow logic unaltered in the Model frame.

(b) SIMAN runs on a mainframe, mini & microcomputers. All versions are compatible. Models can be moved between computer systems without modifications.

(c) SIMAN can be used within the CINEMA system to generate real-time, high-resolution colour graphics animation of the system dynamics. This powerful tool allows not only a visual presentation of the system but, is interactive with the debugger and allows rapid validation with the real world system.

(d) SIMAN's debugger allows the user to monitor & control the execution of the modelled system without the need to recompile, relink and rerun the simulation.

(e) A menu generator is available to allow the model builder to develop a series of menus so that the simulation can be run by an unskilled user. These menus allow variables to be changed, the data collected & stored for subsequent analysis.

(f) Output analysis is via the Output Processor. This can represent graphically Plots, Tables, Histograms etc. Using this feature, the optimum Pallet number can be confirmed.

Experimental Results

The various Pallet configurations were run in 'Batch Mode' and results of Flowtime, Cycle time and total Throughput (Number of Pallets through system in 8 hrs), recorded for 20, 40 and 60 Pallets in the system. The SIMAN Output Processor was used to produce the graphs and associated confidence interval calculations and correlograms.

(a) Results from the first output show the system becomes congested at the 60 Pallet level (the Loop Conveyor reaches its full capacity). The 20 Pallet & 40 Pallet runs yielded a 919 and 920 Pallet throughput in 8 hrs. The 20 Pallet option is clearly more desirable as the Work in Progress and Flowtimes are halved. More detailed experimentation was necessary around the 20 Pallet figure to see if we could reduce these further.

(b) Figures 1 & 2 are plots of number of pallets in system against Flowrate & Throughput. It was decided 13 Pallets are the optimum as it maintains the throughput of 919 Pallets, yet has the lowest WIP figure and very low Flowtime.

Remarks:

Interpretation of whether the length of Buffer in front of the Work Station included the space for the Pallet (while being processed) or not could produce different results. This would effect the congestion on the Loop Conveyor, Throughput and Flowtimes.
**FLOWRATE**

**Pallet in System**

**Througput in 8hrs (K10^3)**

**Pallet in System**

---

**OBSERVATION INTERVALS:**

<table>
<thead>
<tr>
<th>No. of Pallets in System</th>
<th>Average Flow Rate</th>
<th>No. of Pallets Through System in 8hrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5.6289</td>
<td>853</td>
</tr>
<tr>
<td>12</td>
<td>6.2983</td>
<td>914</td>
</tr>
<tr>
<td>13</td>
<td>6.7884</td>
<td>919</td>
</tr>
<tr>
<td>14</td>
<td>7.3118</td>
<td>919</td>
</tr>
<tr>
<td>15</td>
<td>7.8307</td>
<td>919</td>
</tr>
<tr>
<td>20</td>
<td>10.462</td>
<td>919</td>
</tr>
<tr>
<td>25</td>
<td>13.054</td>
<td>919</td>
</tr>
<tr>
<td>30</td>
<td>15.748</td>
<td>919</td>
</tr>
<tr>
<td>40</td>
<td>20.927</td>
<td>920</td>
</tr>
<tr>
<td>50</td>
<td>26.047</td>
<td>917</td>
</tr>
<tr>
<td>60</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**CycTime**

- Average: 0.522
- Standard: 0.950 C.I.: 0.380E-04
- 95% Confidence: Min: 0.522, Max: 0.587

**FlowTime**

- Average: 6.79
- Standard: 1.29
- 95% C.I.: Min: 8.373E-02, Max: 4.24
- 95% Confidence: Min: 6.70, Max: 6.07

---

**INTERRVALS:**

- **CycTime**
  - Average: 0.522
  - Standard: 2.132E-03
  - 95% C.I.: Min: 0.522, Max: 0.587
  - Number of Observations: 9

- **FlowTime**
  - Average: 6.79
  - Standard: 1.29
  - 95% C.I.: Min: 8.373E-02, Max: 4.24
  - Number of Observations: 9
BEGIN.

EUROSIM EXPERIMENTAL FRAME WITH ACCUMULATING CONVEYORS

PROJECT: EUROSIM ACCUM CONVR.

ATTRIBUTES: icon = animation attribute

ARMS: icon = job, icon = job. D1Job = job process flag,
LastStat = previous arm stat.

ARRIVALS: 1, BLOCK: 30, 10, 10, 10, . . ., 10; [Send arriving entities to one Pallet],
100, 40, or 60 arriving pallets.

[Set All] to = .100,463] in LastStat = 10,
Iso this is set as an incoming pallet.

VARIABLES: Start(S), 0.25, 1, 1, 0.33333, 0.33333, 0.33333, 0.33333, 0.33333, 0.33333; Process delays.
Buffer(B), 0, 0; Current Buffer levels.
MaxBuff(B), 2, 2, 2, 2, 2, 2; Max Buffer Levels for Process Stations.
ProcessTime,P, 0; [Set to Process times.
NumCells, N, 0; Time to shift A & S x 2 Secs.
A2Chosen, 0; Variable for A2 Station to convey
A2xChosen, 0; 10.34 = Chosen 3, 4, or 5 Station.

STATIONS: IndB1: IndB2, IndB3, IndB4; Stations.
IndB5: IndB6, IndB7, IndB8; IndB9.
IndA: IndA1, IndA2; IndA3, IndA4.
Sta1: Sta2, Sta3, Sta4; Sta5.
Sta6: Sta7, Sta8; OutSta1, OutSta2, OutSta3, OutSta4.
OutSta5, OutSta6, OutSta7, OutSta8; OutSta9.
OutSta10, OutSta11, OutSta12; OutSta13, OutSta14.
OutSta15, OutSta16, OutSta17, OutSta18; Dummy.

QUEUES: IndB0: IndB20, IndB30, IndB40; Dummy.
IndA0: IndA20, IndA30, IndA40; Dummy.
Sta0: Sta20, Sta30, Sta40; Dummy.
Sta5: Sta60, Sta70, Sta80; Dummy.
OutSta10, OutSta20, OutSta30, OutSta40; Dummy.
OutSta50, OutSta60, OutSta70, OutSta80; Pallet in Ind-Dum.

RESOURCES: worker(B, 1); Process Stations.
Dumy; Dummy resource.

CONVEYORS: Belt, 1, 45.1, Active, 9, Accum, 9; Conveyor spec.

SEGMENTS: 1, IndSta, DumSta-21, Sta-9, OutSta-11,
IndB2, 19, Sta1-20, OutSta-11; [Station to Station distances]
IndA1-20, OutSta-11; For Conveyors.
IndB2, 19, Sta1-20, OutSta-11; [Belt Conveyors]
Comparison 2 - SLAM II

Short description of the simulation environment used

SLAM II is one of the most popular simulation languages available: it has been developed by Pritsker Corp. during the 70’s, and then constantly improved. SLAM II is a discrete-event oriented language, but it includes a network-oriented approach and continuous features as well. For discrete systems modeling with SLAM I, indeed, the network approach is the most direct one. A SLAM II graphic network is a graph with nodes and arrows; the nodes represent typical functions such as queuing points, and the arrows connecting them stand or delays and/or routings. In early versions of SLAM II the graphic network was described in a normal textual form, but in the last years, MS-Windows and OS/2 Presentation Manager versions became available, including a graphic network editor, and a powerful animation building system.

Work description

The model was described with the SLAM II language on an Olivetti PC (with 80386/387 20 MHz) with OS/2. An average simulation run of the model (8 hours) lasts approximately 9 minutes.

The model includes a network (the source is included), with all the conveyor lines modeled as resources. The dynamic entities (pallets) go through them during the simulation, by means of a main loop in the network n which the current resource is incremented: so, we need to define only once the basic submodel, as we can parameterize it. The next resource than an entity needs to go through is allocated by a user function ALLOC that lets wait if it is not possible: so the control strategy is implemented in a very simple, readable, and straightforward manner.

Results

We have conducted many experiments with the model, all of them for a time of eight hours, and collecting statistics for the last six, as suggested.

A table is included that summarizes the average throughput time (cycle time) and the total throughput, as well as the average time in system and the average number of laps for the pallets, for the most interesting experiments (the parameter that changes through experiments is the total number of pallets). One can see that we found a threshold number of pallets, beyond which the system gets blocked because there are some pallets that cannot go their way in the B1 even if there could be the theoretical alternative of taking it go through B2 without being worked by the machine, but the proposed model didn’t provide such a deadlock solving rule.

<table>
<thead>
<tr>
<th>Number of pallets</th>
<th>Average throughput time (cycle time)</th>
<th>Total throughput time (in 6 hours)</th>
<th>Average time in system</th>
<th>Average number of laps</th>
</tr>
</thead>
<tbody>
<tr>
<td>[pure number]</td>
<td>[s]</td>
<td>[s]</td>
<td>[s]</td>
<td>[pure number]</td>
</tr>
<tr>
<td>10</td>
<td>24.00</td>
<td>900.00</td>
<td>240.00</td>
<td>1.00</td>
</tr>
<tr>
<td>14</td>
<td>21.00</td>
<td>1028.00</td>
<td>294.00</td>
<td>1.38</td>
</tr>
<tr>
<td>15</td>
<td>20.30</td>
<td>1062.00</td>
<td>305.00</td>
<td>1.49</td>
</tr>
<tr>
<td>16</td>
<td>20.00</td>
<td>1080.00</td>
<td>320.00</td>
<td>1.72</td>
</tr>
<tr>
<td>17</td>
<td>20.00</td>
<td>1078.00</td>
<td>340.00</td>
<td>2.06</td>
</tr>
<tr>
<td>18</td>
<td>20.00</td>
<td>1080.00</td>
<td>360.00</td>
<td>2.45</td>
</tr>
<tr>
<td>19</td>
<td>20.00</td>
<td>1081.00</td>
<td>380.00</td>
<td>2.82</td>
</tr>
<tr>
<td>20</td>
<td>20.00</td>
<td>1082.00</td>
<td>400.00</td>
<td>3.37</td>
</tr>
<tr>
<td>30</td>
<td>20.00</td>
<td>1080.00</td>
<td>540.00</td>
<td>6.00</td>
</tr>
<tr>
<td>31</td>
<td>20.00</td>
<td>1080.00</td>
<td>800.00</td>
<td>10.70</td>
</tr>
<tr>
<td>50</td>
<td>Blocked within the first two hours</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Blocked within the first two hours</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(*) Different from the following in spite of the same cycle time, because the cycle times were rounded at two decimal digits.

Part of the model description

N. Amicucci, S. Gatteschi, C. Scibilia, AIC S.p.A. Divisione Sistemi, Via Oddino Mangini 31, 1-10112 Torino, Italy, Tel.: +39-(0)11 6690933, Fax: +39-(0)11 657207
CONTROL FILE (VARIABLES EQUIVALENCES AND INITIALIZATIONS)

GEN, SIMGROUP, EUROSIM, 16/05/1991, 1, Y, Y, Y, Y, Y, 1/132;

; TIMES IN SECONDS, LENGTHS IN CM;

LIMITS, 32, 12, 100;

; DYNAMIC ENTITIES (PALLETS) ATTRIBUTES
EQUIVALENCE/ATTRIB(1), MARK: CYCLE ENTERING TIME
EQUIVALENCE/ATTRIB(2), OP2: FLAG FOR OPER. 2 (1=ALREADY DONE, ELSE 0)
EQUIVALENCE/ATTRIB(3), OP3: FLAG FOR OPER. 3 (1=ALREADY DONE, ELSE 0)
EQUIVALENCE/ATTRIB(4), OP4: FLAG FOR OPER. 4 (1=ALREADY DONE, ELSE 0)
EQUIVALENCE/ATTRIB(5), OP5: FLAG FOR OPER. 5 (1=ALREADY DONE, ELSE 0)
EQUIVALENCE/ATTRIB(6), STATION: NUMBER OF CURRENT STATION (BASIC SUBMODEL)
EQUIVALENCE/ATTRIB(7), UP: TELLS IF IT SHOULD GO THROUGH THE MACHINE
EQUIVALENCE/ATTRIB(8), CURR: CURRENT RESOURCE (CONVOYER LINE) USED
EQUIVALENCE/ATTRIB(9), PREC: OLD RESOURCE (CONVOYER LINE) USED
EQUIVALENCE/ATTRIB(10), MACHINE: MACHINE RESOURCE NUMBER (SET ONLY IF UP=1)
EQUIVALENCE/ATTRIB(11), BUFFER: CONVOYER LINE AFTER MACHINE (SET ONLY IF UP=1)
EQUIVALENCE/ATTRIB(12), LAPS: NUMBER OF CIRCUIT LAPS CURRENTLY DONE

; WORK TIMES
ARRAY(1, 8)/15, 60, 60, 60, 60, 20, 20, 20, 30;
A-LINES LENGHTS (CONVOYERS BEFORE MACHINES)
ARRAY(2, 8)/120, 60, 60, 60, 60, 60, 60, 60, 120;
B-LINES LENGHTS (CONVOYERS THAT DOESN'T PASS THROUGH THE MACHINES)
C-LINES LENGHTS (CONVOYERS AFTER MACHINES)
ARRAY(4, 8)/80, 80, 80, 80, 80, 80, 80, 80;
SPACES BETWEEN STATIONS
ARRAY(5, 8)/0, 0, 40, 40, 40, 0, 0, 40, 40;
TIMES AND DISTANCES PARAMETRIZED BY THE STATION NUMBER
EQUIVALENCE/ARRAY(1, STATION), WTIME;
EQUIVALENCE/ARRAY(2, STATION), TRASF;
EQUIVALENCE/ARRAY(3, STATION), TRASFB;
EQUIVALENCE/ARRAY(4, STATION), TRASFC;
EQUIVALENCE/ARRAY(5, STATION), SPACE;
EQUIVALENCE/XX(1), SPEED/XX(2), SHIFTIME/XX(3), PALLETS;
INTL, SPEED=30, SHIFTIME=2;

; PRIORITIES FOR ENTITIES COMING OUT FROM THE MACHINE CONVOYER LINES,
; THAT IS, THOSE WHO HAVE ATRIB(7) (UP) SET TO 1
PRIORITY/1, HVF(7)/2, HVF(7)/3, HVF(7)/4, HVF(7);
PRIORITY/5, HVF(7)/6, HVF(7)/7, HVF(7)/8, HVF(7);

STAT, 4, NUMB. OF PALLETS;
NETWORK;

; EXPERIMENT DURATION (28800 s = 8 h)
; (THE STATISTICS RESET AFTER THE FIRST TWO IS DEFINED ELSEWHERE)
INITIALIZE, 28800, Y;
FIN;

; NETWORK DESCRIPTION

; RESOURCE (CONVOYER LINES AND MACHINES) DESCRIPTION

RESOURCE/ 1, BA1 (120), 1;
RESOURCE/ 2, BA21 (80), 2;
RESOURCE/ 3, BA22 (80), 3;
RESOURCE/ 4, BA23 (80), 4;
RESOURCE/ 5, BA3 (80), 5;
RESOURCE/ 6, BA4 (80), 6;
RESOURCE/ 7, BA5 (80), 7;
RESOURCE/ 8, BA6 (120), 8;
RESOURCE/ 9, BB1 (200), 9;
RESOURCE/ 10, BB21 (160), 10;
RESOURCE/ 11, BB22 (160), 11;
RESOURCE/ 12, BB23 (160), 12;
RESOURCE/ 13, BB3 (160), 13;
RESOURCE/ 14, BB4 (160), 14;
RESOURCE/ 15, BB5 (160), 15;
RESOURCE/ 16, BB6 (200), 8, 9:
EUROSIM COMPARISON 2 - SOLUTIONS AND RESULTS

RESOURCE/18, BC21 (80), 18;
RESOURCE/19, BC22 (80), 19;
RESOURCE/20, BC23 (80), 20;
RESOURCE/21, BC3 (80), 21;
RESOURCE/22, BC4 (80), 22;
RESOURCE/23, BC5 (80), 23;
RESOURCE/24, BC6 (80), 24;

; MACHINES
;
RESOURCE/25, A1 , 25;
RESOURCE/26, A21 , 26;
RESOURCE/27, A22 , 27;
RESOURCE/28, A23 , 28;
RESOURCE/29, A3 , 29;
RESOURCE/30, A4 , 30;
RESOURCE/31, A5 , 31;
RESOURCE/32, A6 , 32;

; MAIN NETWORK FLOW
;
; CREATION OF A PREDEFINED NUMBER OF PALLETS
ZAA0 CREATE,, 1, 1;
ACTIVITY:
ASSIGN,PALLETS=PALLETS-1, 2;
ACTIVITY:
ACTIVITY, 20, PALLETS.GT.0, ZAA0;

; ATTRIBUTES INITIALIZATION
ASS1 ASSIGN, OP2=0, OP3=0, OP4=0, OP5=0;
ACTIVITY:
ASS2 ASSIGN, STATION=1, PREC=16, UP=0;
ACTIVITY:

; OBTAIN THE FIRST RESOURCE FOR THE FIRST TIME
AWO WAIT(9), PREC/36;
ACTIVITY/9;

; MAIN LOOP
; OBTAIN THE NEXT RESOURCE, WAITING IN A QUEUE NUMBERED AS THE CURRENT STATION
; NUMBER, WITH PRIORITY (PREVIOUSLY DECLARED) FOR PALLETS COMING FROM 'UP',
; I. E. THOSE WITH ATRIB(7)=1
; (ALLOC IS THE FORTRAN-WRITTEN NEXT RESOURCE SELECTION RULE)
LOOP WAIT(STATION=1, 8), ALLOC(1);
ACTIVITY/10;
PREC FREE, PREC/36, 1;
SELECT WHERE TO GO, DEPENDING ON ATTRIBUTE UP
ACTIVITY/STATION=1, 8, SPACE/SPEED+SHIFTIME+TRASF+SPEED, UP.EQ.1;
ACTIVITY/11, SPACE/SPEED+TRASF+SPEED, UP.EQ.0, INCR;
; 'UP' SIDE: PASS THROUGH THE MACHINE, OBTAINING THE RELATED RESOURCES
AWMAC WAIT(MACHINE=25, 32), MACHINE,, 1;
ACTIVITY, WTIME, STATION.EQ.1;
ACTIVITY, WTIME,, AWBU;
; STATISTICS COLLECTION AT THE STATION 1 PASSAGE (OPERATION END FOR THE PALLER)
COCT(1), BET, CYCLE TIME;
ACTIVITY;
COCT(2), INT(1), TIME IN SYSTEM;
ACTIVITY;
COCT(3), LAPS, NUMBER OF LAPS, 10/0.0/1.0;
ACTIVITY;
; ATTRIBUTE RESET: THE ENTITY BECOMES A 'NEW' PALLETS
RST0 ASSIGN, MARK=TNOW, LAPS=0;
ACTIVITY;
RST1 ASSIGN, OP2=0, OP3=0, OP4=0, OP5=0;
ACTIVITY,, AWBU;
AWBUF WAIT(BUFFER=17, 24), BUFFER/36;
ACTIVITY;
CURR FREE, CURR/36;
ACTIVITY;
MACH FREE, MACHINE;
ACTIVITY;
ASSIGN, CURR=BUFFER;
ACTIVITY, TRASF/CYCLE+SHIFTIME, INCR;
; 'DOWN' SIDE: DO NOT PASS THROUGH THE MACHINE
; UPDATE STATION NUMBER (IF IT BECOMES 9 THEN IT'S RESET TO 1)
INCR ASSIGN, PREC=CURR, STATION=STATION+1, 1;
ACTIVITY,, STATION.EQ.9;
ACTIVITY,,,LOOP:
   ASSIGN, STATION=1, LAPS=LAPS+1;
   BACK TO THE LOOP
   ACTIVITY,,,LOOP:
END:

SUBROUTINE ALLOC(I,IFLAG)

$INCLUDE:'PARAM.INC'
$INCLUDE:'SOM1.COM'
$INCLUDE:'PARAM.USR'

LOGICAL GOMACHINE,A,B,G

GOMACHINE=.FALSE.

IF (STATION.EQ.1) THEN
   IF (OP2*OP3*OP4*OP5.EQ.1) GOMACHINE=.TRUE.

ELSE IF (STATION.EQ.2.OR.STATION.EQ.3.OR.STATION.EQ.4) THEN
   IF ((OP2.EQ.0).AND.
       ((OP3+OP4+OP5.EQ.0).OR.(OP3*OP4*OP5.EQ.0))) THEN
      OP2=-1
      GOMACHINE=.TRUE.
   ENDIF

ELSE IF (STATION.EQ.5) THEN
   IF (OP3.EQ.0) THEN
      OP3=-1
      GOMACHINE=.TRUE.
   ENDIF

ELSE IF (STATION.EQ.6) THEN
   IF (OP4.EQ.0) THEN
      OP4=-1
      GOMACHINE=.TRUE.
   ENDIF

ELSE IF (STATION.EQ.7) THEN
   IF (OP5.EQ.0) THEN
      OP5=-1
      GOMACHINE=.TRUE.
   ENDIF

ELSE IF (STATION.EQ.8) THEN
   IF (OP3*OP4*OP5.EQ.0) GOMACHINE=.TRUE.
   IF (OP3.EQ.0) THEN
      OP3=-1
   ELSE IF (OP4.EQ.0) THEN
      OP4=-1
   ELSE IF (OP5.EQ.0) THEN
      OP5=-1
   ENDIF

ELSE
   WRITE(*,*),'ERROR'
   STOP
ENDIF

ISTATION=STATION
A=NNRSC(IKURRENT).GE.36  ! CONVEYOR BEFORE THE MACHINE FREE
B=NRSC(8+ISTATION).GE.36  ! LOWER CONVEYOR FREE
G=GMACHINE  ! IT SHOULD GO THROUGH THE MACHINE

IF (A.AND.G) THEN
    UP=1
    CURR=STATION
    MACHINE=24+STATION
    BUFFER=16+STATION
    IF (OP2.LT.0) OP2=-OP2
    IF (OP3.LT.0) OP3=-OP3
    IF (OP4.LT.0) OP4=-OP4
    IF (OP5.LT.0) OP5=-OP5
ELSE IF (B) THEN
    UP=0
    CURR=8+STATION
    MACHINE=0
    BUFFER=0
    IF (OP2.LT.0) OP2=0
    IF (OP3.LT.0) OP3=0
    IF (OP4.LT.0) OP4=0
    IF (OP5.LT.0) OP5=0
ELSE
    IFLAG=0
    RETURN
ENDIF
ICURR=CURR
CALL SEIZE(ICURR,36)  ! ALLOCATE 36 CM OF CONVEYOR
IFLAG=1  ! KEEP ATRIB CHANGES

RETURN

C-----------------------------------------------------------------------------------

END
C-----------------------------------------------------------------------------------
Comparison 2 - Micro Saint

Micro Saint is a full-featured, discrete event simulation software tool that includes iconic animation. It is designed to simulate any type of process that consists primarily of discrete activities (e.g., manufacturing systems, service systems, human-machine systems), although it includes features for limited continuous modeling. In Micro Saint, users interact with menus or graphical interface elements such as windows and dialog boxes to create their models (which are compiled automatically during model execution).

The activity network for the FAS model includes 51 activities in all, 24 of which are in subnetworks. Figure 1 shows the Micro Saint user interface and diagrams for the main network and the A1 subnetwork. The queues at the reentry points in the main network are sorted to ensure that reentering parts have priority, while the subnetwork queues representing waiting lines for each station are designated as First In, First Out (FIFO).

The constraints, timing, effects, and routing logic for each activity (or "task") are defined by expressions and menu choices.

Assumptions made in building the model

- The buffers in front of each station function as FIFO queues, with buffer length limiting queue size according to the following formula: buffer length/pallet length = max queue size.
- A pallet moving along conveyor belt B1 can be paused momentarily if it reaches an S4 station at the same time as a pallet switching over from a B2 belt, which has priority.
- Station A6 emulates A3 if A3 is needed, otherwise A4 if A4 is needed, otherwise A5.
- The operation at station A1 is either loading or unloading, each of which takes 15 seconds.

The last assumption in the preceding list is not one we felt entirely comfortable with, since it makes A1 into a bottleneck station, but it seemed the most reasonable interpretation of the information provided (i.e., "Unprocessed parts are put on pallets in A1," "Finished parts are unloaded in A1," and the 15-second operation time for A1).

Results of running the model with 20, 40, and 60 pallets

- Throughput stayed the same (2 parts output per minute) in all cases because task A1 (the entry and exit point, according to our assumption) acts as a bottleneck. Task A1 takes 15 seconds, so 4 parts can go through A1 every minute. Because a new part comes in through A1 for every part that leaves through A1, 2 of the 4 parts are coming in and 2 are going out.
- Average throughput time per part between the 120th and 600th minute increases when more parts are added to the system, as follows: 10.05 minutes for 20 parts in the system; 20.97 minutes for 40 parts; and 30.38 minutes for 60 parts.

The graph in Figure 2, generated in Micro Saint, shows how average throughput time changed over the course of the simulation with 40 pallets in the system. The dip at time 120 occurs because the values used to calculate the average were zeroed at that point.

Development and execution times

- The model required 14 hours to design and develop and 11 hours to debug, generate graphs, and analyze data. We also spent 8 hours developing an iconic animation of the model for demonstration purposes (shown in Figure 3), but this animation was not necessary for data gathering or analysis.
- The model execution time varied depending upon the number of parts in the system. With 20 parts in the system, the model executed in 6 minutes on an i486 IBM PC compatible operating at 33 MHz. With 60 parts in the system the model executed in 19 minutes on the same computer.

For information or comments, please phone or fax or write to: Karen Ohlson or Ron Laughter, Micro Analysis and Design, 3300 Mitchell Lane, Suite 175, Boulder, CO 80301, USA. Tel: +1(303) 442-6947; Fax: +1(303) 442-8274

Micro Analysis and Design is represented in Europe by: Rapid Data Limited, Crescent House, Crescent Road, Worthing, West Sussex, BN11 8RJ. Tel: +44 903 202819; Fax: +44 903 820762
Flexible Assembly System Model Comparison—Micro Saint

Micro Saint is a full-featured, discrete event simulation software tool that includes iconic animation. It is designed to simulate any type of process that consists primarily of discrete activities, although it includes features for limited continuous modeling. Micro Saint has been successfully used to analyze a variety of processes including manufacturing systems, service systems (e.g., hospitals, banks), and human-machine systems.

Micro Saint has a built-in compiler that allows users to create complex code without the need for any external language. Instead, users interact with menus or graphical user interface (GUI) elements such as windows and dialog boxes, and compilation occurs automatically during model execution. Within this easy-to-use interface, users can develop activity networks, define how activities and queues operate, define and manipulate variables that represent system characteristics, develop optional iconic animations, run models, and generate statistical summaries and graphs from collected data. While running a model, users can select from menus or open multiple windows to view and manipulate different aspects of the model as it runs. They can also watch the activities illuminate on the network diagram as they occur or watch a graphical, iconic animation of the process if one was developed (see Figure 4).

The activity network for the FAS model includes 51 activities—27 in the main network (of which 8 are subnetworks, representing the processing stations), and 24 in the subnetworks (3 per subnetwork). Figure 1 shows the Micro Saint user interface and diagrams for the main network and the A1 subnetwork. The queues at the reentry points in the main network are sorted to ensure that reentering parts have priority, while the queues in the subnetworks (where parts wait to be processed by each station) are designated as First In, First Out (FIFO).
The constraints, timing, effects, and routing logic for each activity (or "task") are defined by expressions and menu choices, as shown in Figure 2 for task #11. This task represents movement along the section of conveyor belt B1 between stations A3 and A4. Because of the release condition for this task, no part can enter until enough time has elapsed to make space available on the conveyor belt. Note how the "Tactical" decision type and subsequent expressions determine whether a part should go into the A4 subnetwork or bypass it; the part will go to A4 only if it has not yet been processed at A4 and there is room for it in the queue in front of A4. Tag values and an array called "A4done" are used to identify the different parts and keep track of whether they have been processed by A4.

The sections that follow describe the assumptions we made in building our FAS model, the results of running the model with 20, 40, and 60 pallets, and the development and execution times for the model.

Assumptions made in building the model

- The buffers in front of each station function as FIFO queues, with the buffer length limiting the queue size according to the following formula:

\[(\text{length of buffer}) / (\text{length of pallet}) = \text{maximum queue size}\]

For example, \(8/36 = 2.2\), so the queues in front of A2, A3, A4, and A5 can contain only two pallets at a time.

- A pallet moving along conveyor belt B1 can be paused momentarily if it reaches an Sy station at the same time as a pallet switching over from a B2 belt, since parts shifting back to B1 are supposed to have priority.

- Station A6 takes in parts that have not been processed at one or more of the following stations: A3, A4, A5. Since A6 can be a substitute for any of these stations, we needed an algorithm to determine which station A6 emulates for any given part. We arbitrarily
made A3 be the first choice, A4 the second, and A5 the third; that is, station A6 performs A3 if A3 is needed, otherwise A4 if A4 is needed, otherwise A5.

- The operation performed at station A1 is either loading or unloading, each of which takes 15 seconds.

The last assumption in the preceding list is not one we felt entirely comfortable with, since it makes A1 into a bottleneck station, but it seemed the most reasonable interpretation of the information provided (i.e., "Unprocessed parts are put on pallets in A1," "Finished parts are unloaded in A1," and the 15-second operation time for A1).

Results of running the model with 20, 40, and 60 pallets

- Throughput stayed the same (2 parts output every minute) in all cases because of task A1 (the entry and exit point, according to our assumption) acting as a bottleneck. Because task A1 takes 15 seconds, 4 parts can go through A1 every minute. Because a new part comes in through A1 for every part that leaves through A1, 2 of the 4 parts are coming in and 2 are going out. Increasing the number of parts in the system from 20 to 40 or 60 can't increase the throughput, because the maximum output rate has already been achieved when there are 20 parts in the system.

- Average throughput time per part between the 120th and 600th minute increases when more parts are added to the system, as shown below:

  Average throughput time for 20 parts in the system: 10.05 minutes
  Average throughput time for 40 parts in the system: 20.97 minutes
  Average throughput time for 60 parts in the system: 30.38 minutes

The graph in Figure 3, generated in Micro Saint, shows how average throughput time changed over the course of the simulation with 40 pallets in the system. The dip at time 120 occurs because the values used to calculate the average were zeroed at that point, so that the overall average would cover only throughput times collected between time 120 and time 600.

![Average throughput time—40 pallets in system](image)

Figure 3
Development and execution times

- The model required 14 hours to design and develop and 11 hours to debug, generate graphs, and analyze data. We also spent 8 hours developing an iconic animation of the model for demonstration purposes (shown in Figure 4), but this animation was not necessary for data gathering or analysis.

- The model execution time varied depending upon the number of parts in the system. With 20 parts in the system, the model executed in 6 minutes on an i486 IBM PC compatible operating at 33 Mhz. With 60 parts in the system the model executed in 19 minutes on the same computer.

![Flexible Assembly System: ActionView](image)

Figure 4

For information or comments, please phone or fax or write to:

Karen Ohlson or Ron Laughery  
Micro Analysis and Design  
3300 Mitchell Lane, Suite 175  
Boulder, CO 80301  
U.S.A.  
Phone: (303) 442-6947  
Fax: (303) 442-8274

Micro Analysis and Design is represented in Europe by:

Rapid Data Limited  
Crescent House  
Crescent Road  
Worthing  
West Sussex  
BN11 5RW  
Phone: 44 903 202819  
Fax: 44 903 820762
Comparison 2 - SIMUL_R

1. The Language

SIMUL_R is a compiling simulation language for continuous and discrete systems. The discrete part is called PROSIMUL_R. The system offers graphical and textual modelling, using one or more models in one simulation program. Examinations are done by using menus and/or a strong runtime interpreter.

The interpreter allows the usage of loops, command files (recursive, too) and arbitrary expressions with assignments and displaying. A special feature are user defined functions which enable the user to add new commands to the system (commands for steady state, zero search, continuous and discrete optimization, statistical evaluations are available as well).

A huge graphical library supports among others moving plots, 3D-plots, niveau lines, cross plots, animation for both, continuous and discrete systems.

SIMUL_R is an open system as it allows data input and output from and to other systems, including user input during simulation (by keys or graphical) as well as hardware in the loop.

PROSIMUL_R only knows one resource: the station. Everything else, like conveyors, is implemented as macros (so it is easy to add new functional objects to the system by writing new macros).

2. The Model

The model consists of one macro for the submodels and the DYNAMIC-section, which contains eight callings of this macro, with different parameters. The conveyors of this example, which are used as buffers, too, are implemented by PROSIMUL_R’s TCONVEYOR_BUFFER macro.

Pallets are put into the system at the loading place for parts (Ax of A1). Old pallets coming to unload parts have priority over new pallets.

3. The Results

The table shows the results. SIMUL_R’s discrete optimization command DOPTCONPAR computes 21 as optimum number of pallets. With 60 pallets and more the system blocks, because pallets cannot enter the circulation conveyor and the pallets on this conveyor move on it endlessly.

<table>
<thead>
<tr>
<th>number of pallets</th>
<th>processed entities</th>
<th>mean of throughput time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>138</td>
<td>208.3</td>
</tr>
<tr>
<td>10</td>
<td>939</td>
<td>306.7</td>
</tr>
<tr>
<td>20</td>
<td>1405</td>
<td>409.5</td>
</tr>
<tr>
<td>21</td>
<td>1409</td>
<td>429.5</td>
</tr>
<tr>
<td>22</td>
<td>1409</td>
<td>450.3</td>
</tr>
<tr>
<td>30</td>
<td>1405</td>
<td>614.2</td>
</tr>
<tr>
<td>40</td>
<td>1403</td>
<td>825.0</td>
</tr>
</tbody>
</table>

The figure shows the animation screen and the plot using the MS-Windows version of PROSIMUL_R.

For information and comments, please phone, fax or write to R. Ruzicka, SIMUTECH, Hadikgasse 130, A-1140 Vienna, Austria. Tel: +43-(0)222-82 03 87 (new: 894 75 08); Fax: +43-(0)222-82 93 91 (new: 894 78 04).
Comparison 2 - GPSS/H

1. Description of GPSS/H

GPSS/H is described in an article on page 5. Please refer to it for an overview of GPSS/H.

2. Observations about the FAS System

The following aspects of the Flexible Assembly System (FAS) being modeled are worth noting:

1. The maximum production rate is about 1409 units every 8 hours. The A2 stations are the bottleneck. Each A2 station takes 61-1/3rd seconds to process one unit; 1-1/3rd seconds to move the unit onto the A2 station, then 60 seconds of operating time.

2. The minimum time needed to assemble a unit is 229-2/3rd seconds (about 3.8 minutes). This time consists of 135 seconds of operating time, 20 seconds of shifting time, and 74-2/3rd seconds to travel 22.4 meters in one system lap in which one Station A2 and Stations A3, A4 and A5 are used, with two A2 stations and the A6 station being bypassed.

3. Model Description

The Comparison 2 problem checks two features of discrete event simulation languages:

- the possibility to define and combine submodels;
- the method to describe complex control strategies.

GPSS/H macros and subroutines provide tools for defining and combining submodels; and such things as Boolean expressions combined with TEST blocks; GATE/Logic-Switch combinations; and the ease of implementing table-driven routings thanks to file I/O and matrices, give GPSS/H the ability to handle complex control strategies easily. (Those who want to obtain the GPSS/H comparison model(s) should see Section 5.)

4. Model Verification

The model(s) were verified using the interactive monitoring feature of GPSS/H. This was done by setting traps on transactions (pallets) and tracing their movement through the system. Pallet movement was consistent with the rules described in the problem statement. For example, it was verified that the minimum time required by the model to assemble a unit is 229-2/3rd seconds.

5. Model Size and CPU-Time Requirements

Two GPSS/H models were built. The larger model, using no macros or subroutines, consists of 264 Blocks. The smaller model, using macros and subroutines, consists of less than 100 Blocks (and can be executed under Student DOS GPSS/H!). Both models are included on the free animation disk (see Section 6). Also included on this disk is a GPSS/H model instrumented to produce the trace output file on which the animation is based.

Using the 264-block model, the 18-pallet simulation of Table 1 was run on a 33 MHz 80386 computer with a math co-processor and using GPSS/H 386 under MS-DOS 5.0. It took 20.4 CPU seconds to compile and execute the model. There were 263,507 block executions.

The same 18-pallet model was also run on the same hardware platform using Personal GPSS/H. Compilation and execution required 34.3 CPU seconds in this case. (GPSS/H 386 uses DOS extender technology not only to circumvent the 640K DOS barrier but also to process information in 32-bit chunks, whereas Personal GPSS/H works with information in 16-bit chunks. As a result, GPSS/H 386 is much faster than Personal GPSS/H.)

6. Animation of the Model

A GPSS/H comparison model has been animated using Wolverine's animation software, Proof Animation. The animation can be run on DOS 286 (or better) machines equipped with a math co-processor and DOS 3.0 or better. To obtain this animation on a disk at no cost, contact Wolverine Software Corporation, 4115 Annandale Road, Annandale, Virginia 22003-2500 USA; Tel: +1.703.750.3910; Fax: +1.703.642.9634. (The animation is self-contained in the sense that except for DOS, no other software is needed to view it.)

7. Experimental Results

Selected experimental results for the Comparison 2 are given in Table 1. These results were obtained by simulating for 2 hours, then reinitializing statistical aspects of the simulation, then simulating for another 8 hours. It was assumed that all pallets were empty initially and were positioned to be loaded at Station A1. Because all timings are deterministic, all correct models built under the same initial conditions and assumptions as the GPSS/H model(s) should produce the results in Table 1 (assuming the GPSS/H model(s) are correct), independent of the simulation software being used.

As shown in Table 1, the optimal feasible production rate (about 1409 units every 8 hours) is achieved (for all practical purposes) with 18 pallets.

Prof. Tom Schieber, Computer and Information Systems, The University of Michigan, Ann Arbor, MI 48109-1234 USA; Fax: +1.313.763.5688; Email: tjs@ub.cc.umich.edu or userk2bu@umichub.bitnet.

<table>
<thead>
<tr>
<th>Number of Pallets</th>
<th>Jobs Completed in Final 8 Hours</th>
<th>Job Completion Time (Minutes)</th>
<th>Number of Jobs Needing 1 or 2 or 3 or More Laps to Finish</th>
<th>Number of Uses of Station A6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Std. Dev.</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>150</td>
<td>5.33</td>
<td>0.89</td>
<td>1350</td>
</tr>
<tr>
<td>16</td>
<td>150</td>
<td>5.69</td>
<td>0.89</td>
<td>1350</td>
</tr>
<tr>
<td>17</td>
<td>1371</td>
<td>5.96</td>
<td>0.82</td>
<td>1371</td>
</tr>
<tr>
<td>18</td>
<td>1408</td>
<td>6.13</td>
<td>0.79</td>
<td>1408</td>
</tr>
<tr>
<td>19</td>
<td>1409</td>
<td>6.47</td>
<td>0.53</td>
<td>1409</td>
</tr>
<tr>
<td>20</td>
<td>1409</td>
<td>6.82</td>
<td>0.50</td>
<td>1289</td>
</tr>
<tr>
<td>25</td>
<td>1408</td>
<td>8.52</td>
<td>2.75</td>
<td>682</td>
</tr>
</tbody>
</table>

Table 1: Selected Experimental Results
**EUROSIM COMPARISON 2 - SOLUTIONS AND RESULTS**

*************** Comparison 2: FAS Model ***********************

* (Reference: EUROSIM Simulation News Europe, July 1991)
* Modeling Language: GPSS/H
* Modelers: Dan Brunner / Doug Smith
* Wolverine Software Corporation, Annandale, VA 22003

```
SIMULATE
TRUE SYN 1
FALSE SYN 0

B1 EQU 1(8),S
B2 EQU 9(8),S
B3 EQU 17(8),S
B4 EQU 25(8),S
SOURCE EQU 33,S
AX EQU 1(8),F
SX EQU 9(8),F
SY EQU 17(8),F
JUNCTION EQU 25(8),F
DONE1 EQU 1,PH
DONE2 EQU 2,PH
DONE3 EQU 3,PH
DONE4 EQU 4,PH
DONE5 EQU 5,PH

* STORAGE S1,5/S2-7,4/S8,5
STORAGE S9,3/S10-S15,2/S16,3
STORAGE S17-S24,1
STORAGE S25-S32,2
STORAGE S(SOURCE),60
WHICHOP FUNCTION PH(STN),D8
0.1/1.2/2.2/3.2/4,3/5,4/6,5/7,6
B1DIST FUNCTION PH(OPNUM),D6
1,2/1.6/3.1,6/4.1,6/5,1.6/6,2
B2DIST FUNCTION PH(OPNUM),D6
1.1,2/2.0,8/3.0,8/4.0,8/5,0.8/6.1,2
OPTIME FUNCTION PH(OPNUM),D6
1,15/2.60/3,20/4,20/5,20/6,30
* Variable Declarations
INTEGER &NPALLETS
REAL &SPEED
LET &SPEED=0.3
* Boolean Variables used to test for completed op'ns
1 BVARIABLE (PH(DONE1)=FALSE) OR
   ((V(DONE345) =3) AND (PH(DONE2) =TRUE))
2 BVARIABLE (PH(DONE2)=FALSE) AND (BV(IN345) =FALSE)
3 BVARIABLE (PH(DONE3)=FALSE)
4 BVARIABLE (PH(DONE4)=FALSE)
5 BVARIABLE (PH(DONE5)=FALSE)
6 BVARIABLE (V(DONE345)<3)
IN345 BVARIABLE (V(DONE345)>0) AND (V(DONE345)<3)
DONE345 VARIABLE PH(DONE3)+PH(DONE4)+PH(DONE5)
*
* Primary Macro to process parts through Stations
*
ASTAT STARTMACRO
ASSIGN STN,#A,PH
ASSIGN OPNUM, FN(WHICHOP),PH Op Num is func of Station Num
TEST E BV(PH(OPNUM)),TRUE,BYPASS#A Need Processing Here?
GATE SNF B2+PH(STN),BYPASS#A Is Buffer Free?
```
TEST E

PH(OPNUM),1,NO1#A Yes,Yes; Is this Operation 1?

ASSIGN
DONE1,TRUE,PH Operation 1 => Mark 1 as done
ASSIGN
DONE2,FALSE,PH New pallet => 2 not done yet
ASSIGN
DONE3,FALSE,PH New pallet => 3 not done yet
ASSIGN
DONE4,FALSE,PH New pallet => 4 not done yet
ASSIGN
DONE5,FALSE,PH New pallet => 5 not done yet
TRANSFER ,BEGIN#A

NO1#A TEST NE
ASSIGN
PH(OPNUM),6-BEGIN#A Is this Operation 6?
(DONE1-1)+PH(OPNUM),TRUE,PH Not 1 or 6; mark done

BEGIN#A SEIZE
ENTER SX+PH(STN) Seize the input switch
LEAVE B2+PH(STN) Movement into station begins here
ADVANCE 2.0 Transfer Switch time
RELEASE SX+PH(STN) Release the input switch
GATE SF B2+PH(STN),NOBLK#A
NOBLK#A ADVANCE
GATE FNU FN(B2DIST)/&SPEED Convey down input buffer
SEIZE AX+PH(STN) Wait for machine to be available
LEAVE B2+PH(STN) Relinquish space on input buffer
ADVANCE 0.4/&SPEED Move into machine
TEST E
PH(OPNUM),6,NO6#A If this is Operation 6...
LOOP6#A SELECT E
(OP)PH,DONE3,DONE5,FALSE,PH Choose Op 3, 4, or 5
TEST G
PH(OP),0,NO1#A Done if no op's left
ASSIGN
PH(OP),TRUE,PH Mark as done
ADVANCE FN(OPTIME) Use Operation 6
TRANSFER ,LOOP6#A Loop back for next op

NO6#A ADVANCE
TEST E
PH(OPNUM),1,NO1#A#A Branch unless this is Operation 1
DEPART SYSTEM Operation 1 => old job ends
QUEUE SYSTEM Operation 1 => new job begins

NO1#A#A ENTER
B3+PH(STN) Move into output buffer
PRIOR PR+1
SEIZE JUNCTION+PH(STN) Junction must be clear
PRIOR PR+1
SEIZE SY+PH(STN) Seize the output switch
RELEASE AX+PH(STN) Relinquish machine
ADVANCE 0.4/&SPEED Remove part from machine
LEAVE B3+PH(STN) Relinquish space on output buffer
ADVANCE 2.0-(0.4/&SPEED) Switching time less machine unload
ENTER B4+PH(STN) Move onto output conveyor
RELEASE SY+PH(STN) Release the output switch
MERGE#A ASSIGN
PREVIOUS,B4+PH(STN),PH Record current storage
TEST NE
PH(STN),3,SHFT1#A Branch if Station 3
TEST NE
PH(STN),7,SHFT7#A Branch if Station 7
RELEASE JUNCTION+PH(STN) Relinquish the junction
ADVANCE 0.8/&SPEED Move down connector
TRANSFER ,FIN#A End of processing

BYPASS#A ENTER
B1+PH(STN) Move onto bypass conveyor
LEAVE PH(PREVIOUS) Relinquish previous storage
ADVANCE (FIN(B1DIST)/&SPEED) Move down bypass
SEIZE JUNCTION+PH(STN) Seize the junction
LEAVE B1+PH(STN) Move off of bypass conveyor
ADVANCE 0.4/&SPEED Move to junction
ENTER B4+PH(STN) Enter output conveyor
* Transfer
 ,MERGE*A  Continue with logic above

* SHFT3*A SEIZE  SWITCH3  Seize Transfer Switch
ADVANCE  0.33  Clear the junction
RELEASE  JUNCTION+PH(STN)  Release the junction
ADVANCE  1.34  Clear transfer switch
RELEASE  SWITCH3  Release Transfer Switch
ADVANCE  0.33  End of transfer path
TRANSFER ,FIN*A

* SHFT7*A SEIZE  SWITCH7  Seize Transfer Switch
ADVANCE  0.33  Clear the junction
RELEASE  JUNCTION+PH(STN)  Release the junction
ADVANCE  1.34  Clear transfer switch
RELEASE  SWITCH7  Release Transfer Switch
ADVANCE  0.33  End of transfer path

* FIN*A ADVANCE
ENDMACRO

* The actual model begins here

* GENERATE 1,.,&NPALLETS,,10PH,1PL  Initialize the system
ASSIGN DONE1,TRUE,PH  Initialize pallet type

* QUEUE SYSTEM  Track entries
ENTER SOURCE  Dummy storage
ASSIGN PREVIOUS,SOURCE,PH  Initialize previous storage

REJCIRC ADVANCE
ASTAT MACRO 0  Visit Station 0
ASTAT MACRO 1  Visit Station 1
ASTAT MACRO 2  Visit Station 2
ASTAT MACRO 3  Visit Station 3
ASTAT MACRO 4  Visit Station 4
ASTAT MACRO 5  Visit Station 5
ASTAT MACRO 6  Visit Station 6
ASTAT MACRO 7  Visit Station 7
TRANSFER ,REJCIRC  Loop continuously

* GENERATE 60  Timer Transaction, 1 per minute
TERMINATE 1

* DO  &NPALLETS=20,60,20
START 120  120 Minutes warm-up time
RESET
START 480  8 Hours simulated time
PUTPIC LINES=4,(&NPALLETS,QC(SYSTEM)-&NPALLETS,QT(SYSTEM))

WITH * PALLETS...
NUMBER OF PARTS PROCESSED: *
AVERAGE TIME IN SYSTEM: *** SECONDS
======================================================================
PUTPIC LINES=4, FILE=SYSINDEX,_
(&NPALLETS,QC(SYSTEM)-&NPALLETS,QT(SYSTEM))

WITH * PALLETS...
NUMBER OF PARTS PROCESSED: *
AVERAGE TIME IN SYSTEM: *** SECONDS
======================================================================
CLEAR
ENDDO

* END

---
Comparison 2 - CASSANDRA

The Simulator

CASSANDRA (Cognizant Adaptive Simulation System for Applications in Numerous Different Relevant Areas) 2.1 - developed in the Institute for Measurement and Computing Techniques of the Hungarian Academy of Sciences - is a universal kernel system based internally on an object oriented structure utilizing primarily numerical Petri Net elements for its model representation. This approach ensures a realistic structural and non-procedural view of the systems investigated.

CASSANDRA 2.1 enhances the eectivity of simulation by automating the control of simulation experiments as well as the goal oriented reconstruction of the model structures using AI attributed demons.

The simulator provides for an easy extension into problem oriented, specialized and user friendly tools for various fields by means of extending it with application field specific higher level building elements and I/O communication layers.

In our case CASSANDRA 2.1 was extended with a set of macro elements (as robots, conveyor belts etc.) based on the internal PN elements for simulating FMS models. Beyond that an experimental user interface layer for the given field has been developed by TU Wien and the Simulation Project Center in Wiener Neustadt under the supervision of Prof. Dr. Felix Breitenecker. The extension of the kernel system by this specific I/O layer was given the codename IGENJA.

Model Description

The model was run on a 80386/387 AT type system under MICROSOFT WINDOWS 3.0. The model building blocks consisted of various robots with their respective conveyor belt segments and shifting parts according to the example that had to be modelled. The resources could examine the constraints corresponding to the order of operations to be performed on the workpieces which were checked. The figure illustrates the graphic description of the model in the IGENJA system. The models in the system could be assembled graphically from user level model elements of the system library that were transformed automatically into the internal PN representation.

Results

The final results of the simulation experiments are given in table 1.

<table>
<thead>
<tr>
<th>Number of pallets</th>
<th>total throughput</th>
<th>average throughput time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1039</td>
<td>277.9</td>
</tr>
<tr>
<td>15</td>
<td>1374</td>
<td>313.3</td>
</tr>
<tr>
<td>20</td>
<td>1415</td>
<td>410.7</td>
</tr>
<tr>
<td>25</td>
<td>1417</td>
<td>517.9</td>
</tr>
<tr>
<td>30</td>
<td>1417</td>
<td>629.1</td>
</tr>
<tr>
<td>35</td>
<td>1416</td>
<td>691.6</td>
</tr>
<tr>
<td>40</td>
<td>1415</td>
<td>838.5</td>
</tr>
<tr>
<td>45</td>
<td>1418</td>
<td>866.8</td>
</tr>
<tr>
<td>50</td>
<td>1379</td>
<td>1029.3</td>
</tr>
<tr>
<td>55</td>
<td>deadlock</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>deadlock</td>
<td></td>
</tr>
</tbody>
</table>

Remark: data collected from the 120th to the 600th minute.

Beyond the above results the IGENJA system provide the animation of the changes in the state of the model during simulation run enabling thereby a good overview of the operation of the system investigated.

For information and comments, please phone or fax to:

Prof. Dr. A. Jävor, KFKI Research Institute for Measurement and Computing Techniques of the Hungarian Academy of Sciences, H-1525 Budapest, P.O.Box 49, Hungary. Tel: +36-1 1699499, Fax: +36-1 1553894
Comparison 2 - DESMO

1. The DESMO Software

Background:

The simulation package DESMO, developed at the informatics department of Hamburg University, was inspired by the process style DEMOS system in Simula (G.M. Birtwistle) adopting the entity approach for simulation objects. The DESMO user can import a wide range of simulation functions into his model program, which is written in the base language Modula-2. Modula-2 with its comprehensive language kernel allows for well structured and readable, safe and efficient simulation programs on PCs offering features such as strong typing, modularization with separate compilation and interface checking, information hiding and access protection, respectively.

Package Functionality:

DESMO offers, next to event scheduling and process interaction functions, contracts for modelling on a higher level of abstraction (above processes) allowing for a more problem oriented and therefore more convenient implementation of simulation models. These synchronization mechanisms are: Resource competition with mutual exclusion (i.e. entity resource requests), producer/consumer relations of entities, direct co-operation of entities (master/slave relations), and conditional waiting of entities.

Semi-automatic statistics, collection and graphical display of simulation time series data, trace and debug facilities, consistency and deadlock checks as well as extensive reporting are available.

Technical Data:

Implemented in LOGITECH Modula/TopSpeed Modula on IBM PC (2/50 and 2/80 with 80287 coprocessor). 25 modules with 19000 lines of source code (430 KB) / 380 KB object code.

2. Model Implementation

The sample model is implemented in the process interaction style using also higher modelling constructs of DESMO. The model consists of three components: the subsystems, the conveyor systems linking these subsystems and the pallets as dynamic elements. The pallets are defined as processes. Each subsystem and each conveyor is realized as Modula-2 record with the related elements as record components. If feasible the elements are represented by higher modelling constructs; i.e. the station, one buffer behind the station and the conveyors between the systems are resources. The conveyors B1 and B2 are simply implemented as (free capacity) counter variables. To enter a system a pallet has to go through a conditional waiting object. There it is checked if the pallet can enter and on which way it will pass through the system. The pallet operation sequence is realized as a Modula set with station numbers as elements. The set contains a limiting condition (A2 being the first or last station) determine the station selection of a pallet. System control follows the pallet process description and the automatic synchronization mechanisms.

The simulation program has 400 lines of code (14 kB) and the run time on the PS 2/80 is 80 sec for the sample system (15 pallets).

3. Simulation Results

As starting condition pallets are not permitted on conveyor B2. Therefore the maximum number of pallets in the system is 40 (capacities of all B1 and the conveyors between the systems). The simulation experiments were executed with 10 pallets in the system up to 40 by steps of 5. The results show that 20 is a favourable number of pallets (see Tab.1). The throughput is too small if there are less pallets in the system. On the other hand more pallets are not increasing the throughput of the system because of congestion effects whereas the average throughput time of one pallet increases significantly.

<table>
<thead>
<tr>
<th>Nr. of pallets</th>
<th>throughput (pallets)</th>
<th>avg. throughput-time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>939</td>
<td>5.1</td>
</tr>
<tr>
<td>15</td>
<td>1351</td>
<td>5.3</td>
</tr>
<tr>
<td>20</td>
<td>1408</td>
<td>6.8</td>
</tr>
<tr>
<td>25</td>
<td>1407</td>
<td>8.5</td>
</tr>
<tr>
<td>30</td>
<td>1409</td>
<td>10.2</td>
</tr>
<tr>
<td>35</td>
<td>1408</td>
<td>12.0</td>
</tr>
<tr>
<td>40</td>
<td>1409</td>
<td>13.7</td>
</tr>
</tbody>
</table>

Tab. 1: Results
(data collection from 120th to 600th minute)

Documentation:


Contact:


67
Comparison 2 - TOMAS

Simulation System TOMAS/16

1. Description of the Language

The simulation system TOMAS (Technology Oriented Modelling And Simulation) supports simulation in the field of discrete technological processes. At the end of the 70ies it was designed by the Faculty of Informatics of the Technical University Dresden and realised for the first time in the beginning of the 80ies. In 1990 TOMAS/16 was implemented by DVZ Neubrandenburg GmbH for MS-DOS PCs. Since then it has been offered on the software market.

TOMAS is mainly used in two areas:
- during designing of manufacturing processes
- during planning and controlling of manufacturings

TOMAS - being a building element system - consists of 12 modules, by means of which the user - supported by the computer - can build models. As TOMAS is a simulation system oriented at special fields, the modules present universal manufacturing subprocesses, that means, they imitate typical processes of manufacturing.

These modules - operators and generators - will be passed by operands during simulation. Operands may be for example manufacturing jobs, lots, vehicles, parts.

Behaviour of the single operators and generators is specified by parametrization of the elements. Then the model is able to be processed.

The management of the simulation system TOMAS/16 is easy for the user, because it is menu-driven and many helps will be available in on-line status.

2. Model Description

To solve the problem 7 of the 12 modules were used, realising the following functions:

<table>
<thead>
<tr>
<th>Num.</th>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GENO</td>
<td>generator of operand</td>
</tr>
<tr>
<td>1</td>
<td>GENA</td>
<td>passive generator</td>
</tr>
<tr>
<td>2</td>
<td>VERE</td>
<td>joining operator</td>
</tr>
<tr>
<td>1</td>
<td>DISP</td>
<td>disposition operator</td>
</tr>
<tr>
<td>7</td>
<td>BEMM</td>
<td>processing operator</td>
</tr>
<tr>
<td>16</td>
<td>VERZ</td>
<td>branching operator</td>
</tr>
<tr>
<td>17</td>
<td>SPEI</td>
<td>storing operator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>creates first pallets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>terminates completely processed parts, put</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unprocessed parts on pallets (AI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>subelement to connect pallets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sink of model, statistics of pallets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>manipulate parts of pallets (A2-A6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>branching of pallets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sections of conveyor belts B1 and B2</td>
</tr>
</tbody>
</table>

3. Experimentation Results

In contrast to other solutions published in "Eurosim" in our model the pallets are created step by step. They are put on the conveyor belts if there is a place available (see model picture above). So we can show that never more than 40 pallets are on the belt. If more than 40 are created then a part of the pallets is waiting in front of the belt. If there were more pallets on the belt a deadlock would be determined.

The optimum number of pallets may be seen in the following table:

<table>
<thead>
<tr>
<th>Number of Pallets</th>
<th>Total Throughput</th>
<th>Average Throughput Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>848</td>
<td>5.655</td>
</tr>
<tr>
<td>13</td>
<td>964</td>
<td>6.461</td>
</tr>
<tr>
<td>14</td>
<td>1018</td>
<td>6.590</td>
</tr>
<tr>
<td>15</td>
<td>1014</td>
<td>7.091</td>
</tr>
<tr>
<td>16</td>
<td>1031</td>
<td>7.431</td>
</tr>
<tr>
<td>17</td>
<td>857</td>
<td>9.290</td>
</tr>
<tr>
<td>18</td>
<td>867</td>
<td>9.679</td>
</tr>
<tr>
<td>20</td>
<td>884</td>
<td>10.394</td>
</tr>
<tr>
<td>30</td>
<td>951</td>
<td>14.754</td>
</tr>
<tr>
<td>40</td>
<td>1005</td>
<td>18.832</td>
</tr>
</tbody>
</table>

The biggest throughput will be reached by a number of 16 pallets. If there are more pallets on the belt the average throughput time will be greater, that means some pallets must circulate over and over, before being processed.

Further increasing the number of pallets doesn’t yield the same numbers as with 16 pallets, because the third A2-station’s and the A6-station’s capacity aren’t fully used if 16 pallets are in the system.

For information and comments, please phone, fax or write to:

Beate Steinke, DVZ Neubrandenburg GmbH, Bereich Softwareentwicklung und Systemberatung, Woldegger Straße 12, 0-2000 Neubrandenburg, Germany, Tel: +37-90-587 443 Fax: +37-90-587 302
Comparison 2 - SIMPLE-mac

1. A description of SIMPLE-mac

The starting point for this object oriented simulator is a limited supply of parameterizable elements (modules) which allow a complete representation of any discrete system. Controller modules provide a flexible representation (in the form of decision tables) of the complete data flow. The individual modules can be grouped together to form macros, which can in turn serve as modules for the model development process. The creation of an animation layout results from the model development process and therefore requires no additional effort.

Because of SIMPLE's modular concept it is not necessary to create the simulation model via a programming language. Instead, the individual modules are arranged on the screen and connected to one another through material and data flow. In the event the fundamental attributes of a given element are insufficient, its attributes can be extended through the addition of local and/or global controllers.

An additional advantage of SIMPLE-mac is the possibility to view and modify all model parameters without having to leave the simulator.

SIMPLE-mac is operated via a menu driven, window oriented graphic interface. It also provides its own editor for the arrangement of modules and the creation of decision tables. The graphic interface was designed in the MAC-OS style.

2. Model description

The time elapsed for positioning a pallet in the work station A1 (moving from the end of the buffer zone to the work station - 1.33 seconds) is included as a part of the operation time. This method results in an optimal value for the total throughput of 1440 parts (see EUROSIM-91/2, age 28).

The basic submodule consists of seven "RUTSCHE" (Chute) elements and a local controller. This module was used to represent the work stations A2, A3, A4, A5 and A6. The model first work station (A1) the basic submodule was extended with a controller for the creation and destruction of entities. Further, it was attempted to improve the throughput in the throughput and throughput time by using a smaller number of pallets with the help of a supervisory controller.

3. Results of the simulation

The diagram shows the results of the individual simulation runs. By using the SIMPLE-mac features to create superimposed, complex controllers an improved strategy for null numbers of pallets could be found.

With the use of 50 pallets and more the system blocked due to the exhaustion of the capacity of the conveyor belt.

For information or comments, please phone, fax or write:

Gernot Kronreif, Fa. UNSELD & PARTNER, Lerchen-derstraße 44/V, A-1080 Vienna, Austria. Tel: +43-222-4030371; Fax: +43-(0)3332-65149.
Comparison 2 - WITNESS

1. Description of WITNESS

The simulating system WITNESS used to simulate the given problem offers global elements to build the model. Elements can describe discrete and continuous events. In this case only the discrete elements were needed. The elements are defined by name and displayed on the screen. Any display can be used by creating icons with an editor. The third step of describing the model is detailing the elements. Every element has got a parameter mask which has to be filled by the user. Parameters are cycle-time, capacity or breakdown details as well as material flow or information flow links between the elements and control strategies to run the model. Simulating the model produces an online animation. Results can be given by standard statistics or self-made functions and values.

The used WITNESS version runs on PC 386 with OS/2.

2. Building the model

To build a model that simulates the given problem, the following WITNESS standard elements have been used: machines for every Ax; conveyor for every Sx, Sy, B2, B3 (conveyor between Ax and Sy) and C (conveyor between the subsystems); buffer for every B1; part to define the pallet; attributes (of parts) to give every pallet its individual state of work; variables to define dynamic cycle-times.

Two assumptions were made to complete the model:

- empty pallets (and pallets with completely processed parts) that cannot get to station A1 because the buffer in front of A1 is full move on through the system

- the operation time in A6 is 30 sec for every execution of the work that should be done in A3, A4 or A5. A6 does all the remaining work of A3, A4, A5 as one complete operation.

Using WITNESS, it is at last impossible to build the model out of submodels. The given problem consists of eight slightly different areas but WITNESS does only give the opportunity to define slightly different elements. There is, as well, the possibility to build a system out of subsystems by creating a subsystem as an individual model and integrating these models into a new system. For this it is necessary for the user to make use of a text editor and to rename and control the identifier of all elements.

3. Results

To find the optimum number of pallets to run the system, various simulation runs were made. The results we are interested in were given by the standard statistic output of WITNESS by collecting data from the 120th to the 600th minute.

a.) the optimum number of pallets in the system is 17. By looking at the results it is interesting to see that it is possible to get an output of 1441 parts although there should be a maximum value of 1440 parts, given by the simulated time (8 hours) divided by the longest operation time (20 sec).

b.) the average throughput time by using 17 pallets is 350.6 sec.

c.) the system will get a deadlock situation by using 62 pallets.

d.) the optimum efficiencies of stations in the system will be given by using 17 pallets. In this case, station A6 won't be used at all (only perhaps at the reason of randomly chosen start positions of empty parts). The efficiencies of all other 'normal' stations is 100 %.

Ralph Meyer, Bremer Institut für Betriebswirtschaft und angewandte Arbeitswissenschaft BIBA Postfach 330560, W-2800 Bremen 33, Germany, Tel: +49 421 22009-43, Fax: +49 421 22009-79

<table>
<thead>
<tr>
<th>number of pallets</th>
<th>total throughput</th>
<th>average throughput time</th>
<th>A1</th>
<th>A2 (1)</th>
<th>A2 (2)</th>
<th>A2 (3)</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>999</td>
<td>311.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1398</td>
<td>320.35</td>
<td>72.81</td>
<td>100.00</td>
<td>100.00</td>
<td>99.97</td>
<td>0.00</td>
<td>66.66</td>
<td>66.55</td>
<td>66.67</td>
</tr>
<tr>
<td>16</td>
<td>1426</td>
<td>334.38</td>
<td>74.25</td>
<td>100.00</td>
<td>100.00</td>
<td>99.97</td>
<td>97.98</td>
<td>98.01</td>
<td>99.00</td>
<td>99.00</td>
</tr>
<tr>
<td>17</td>
<td>1440</td>
<td>350.62</td>
<td>75.00</td>
<td>100.00</td>
<td>100.00</td>
<td>99.97</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>18</td>
<td>1440</td>
<td>375.11</td>
<td>75.00</td>
<td>100.00</td>
<td>100.00</td>
<td>99.97</td>
<td>98.58</td>
<td>97.98</td>
<td>99.10</td>
<td>99.10</td>
</tr>
<tr>
<td>19</td>
<td>1441</td>
<td>385.33</td>
<td>75.04</td>
<td>100.00</td>
<td>100.00</td>
<td>99.97</td>
<td>99.84</td>
<td>94.58</td>
<td>96.27</td>
<td>99.23</td>
</tr>
<tr>
<td>20</td>
<td>1439</td>
<td>409.30</td>
<td>74.94</td>
<td>100.00</td>
<td>99.99</td>
<td>99.67</td>
<td>89.94</td>
<td>94.91</td>
<td>97.64</td>
<td>25.73</td>
</tr>
<tr>
<td>30</td>
<td>1438</td>
<td>602.85</td>
<td>74.93</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>97.76</td>
<td>92.20</td>
<td>95.34</td>
<td>37.39</td>
</tr>
<tr>
<td>35</td>
<td>1441</td>
<td>697.10</td>
<td>75.05</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>86.02</td>
<td>93.84</td>
<td>96.18</td>
<td>31.96</td>
</tr>
<tr>
<td>40</td>
<td>1441</td>
<td>791.92</td>
<td>75.06</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>91.57</td>
<td>95.36</td>
<td>96.69</td>
<td>24.76</td>
</tr>
<tr>
<td>60</td>
<td>1439</td>
<td>1166.45</td>
<td>74.94</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>90.84</td>
<td>94.80</td>
<td>96.52</td>
<td>25.15</td>
</tr>
<tr>
<td>61</td>
<td>1440</td>
<td>1183.70</td>
<td>75.02</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>93.55</td>
<td>96.30</td>
<td>97.27</td>
<td>18.82</td>
</tr>
</tbody>
</table>

Total Throughput

Average Throughput Time

number of pallets
EUROSIM COMPARISON 2 - SOLUTIONS AND RESULTS

Comparison 2 - SIMFLEX/2

Description of SIMFLEX/2

SIMFLEX/2 has been developed by the section Production Systems of the Department for Mechanical Engineering at the University of Kassel. It is an element orientated simulator for material flow systems. Out of a given set of standardized elements material flow systems are constructed via menu oriented, graphic user interface. The elements' graphic depiction can be taken from existing plant layouts with a CAD-interface. The elements' function is influenced by means of technical (e.g. speed, capacity) and logistic parameters (e.g. strategies). When required the user can modify the steering programs of the elements. In this way even plants with complex logistics can be modelled. Having started the model a graphic animation and a statistic registration system can be added. Thus the user is enabled to intervene in a running simulation. A special feature of SIMFLEX/2 is its real time interface for communication with Programmable Logic Controllers (PLC). Through this it is possible to use the simulator for controlling real plants.

Description of the model

The type of problem allows several sequences of operations. Therefore each pallet is assigned with a note which states the jobs already done. A pallet's note and the number of pallets waiting to be worked on by a specific sub-module decide whether a pallet is accepted for processing by some sub-module or whether it passes by. The decision is made by the controller at a sub-module's entrance. Figure 1 shows the complete structure of a submodule. At the construction of the model the CAD-interface has been used for the plant's graphic depiction (figure 2).

![Figure 1: Submodule](image)

![Figure 2: Plant Layout](image)

Results

In order to find out the optimal number of pallets, 10 experiments were run. We started with 5 pallets and increased their number up to 50 by steps of 5. We measured the pallets' average throughput time, the total throughput within 8 hours and the stations' efficiencies.

![Figure 3](image)

The optimal number of pallets in the system was found to be between 15 and 24, since then the throughput is already high while the throughput time is still low (Figure 3). With fewer pallets the throughput is considerably reduced and the throughput time only slightly reduced. An increase of pallets to more than 20 leads to only little more throughput but considerably prolongs throughput time. In addition, the jobs of station A3, A4 and A5 shift to the substitute station A6.

<table>
<thead>
<tr>
<th>number of pallets</th>
<th>total throughput</th>
<th>average throughput time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>470</td>
<td>306.0</td>
</tr>
<tr>
<td>10</td>
<td>941</td>
<td>306.0</td>
</tr>
<tr>
<td>15</td>
<td>1359</td>
<td>318.0</td>
</tr>
<tr>
<td>20</td>
<td>1409</td>
<td>408.6</td>
</tr>
<tr>
<td>25</td>
<td>1412</td>
<td>593.4</td>
</tr>
<tr>
<td>30</td>
<td>1410</td>
<td>612.6</td>
</tr>
<tr>
<td>35</td>
<td>1410</td>
<td>715.8</td>
</tr>
<tr>
<td>40</td>
<td>1414</td>
<td>816.0</td>
</tr>
<tr>
<td>45</td>
<td>1412</td>
<td>924.0</td>
</tr>
<tr>
<td>50</td>
<td>1412</td>
<td>1020.6</td>
</tr>
</tbody>
</table>

For more information and comments please contact: B. Kreuzer, G. Lührs, A. Reinhardt, S. Schröder, FG Produktionssysteme, Universität Gb Kassel, Mönchebergstr. 7, W-3500 Kassel, Germany, Tel: +49-(0)561-804-2693, Fax: +49-(0)561-804-2330.
Comparison 2 - EXTEND

Description of EXTEND

EXTEND is a general purpose simulation system supporting both continuous and next event modeling. It is library-based and uses a block diagram approach to modeling. You can use libraries of pre-built blocks to set up models with no programming (for example Manufacturing) or you can use MODL, a built-in modeling language, to modify existing blocks or create new ones. The Manufacturing library allows you to create complex factory simulations.

Since version 2.0 EXTEND supports hierarchical modeling. We worked with version 1.1.

EXTEND runs on Macintosh computer. EXTEND™ is a product of Imagine That Inc., 151 Bernal Road, Suite 5, San Jose, CA 95119 USA.

Model Description

Figure 1 shows a model description by default blocks of EXTEND's Sample Manufacturing Library (a freeware child of the Manufacturing library) and the general Discrete Event Library. The times to change the conveyors are added to the conveyor working times. It is difficult to describe flexible control strategies by the default blocks.

We developed a second model with modified and new created blocks (machine, conveyor switcher, control) to handle flexible control strategies. Figure 2 shows the general model layout for two machines. The "machine" block modifies item attributes, which are evaluated by the "control" block. The control strategies are described for each "control" block by a decision table (figure 3). The new features of version 2.0 allow a control description by logical equations and to model in hierarchical levels.

Simulation Results

The simulation experiments were executed with 10 pallets in the system up to 30 by steps of 5. The experiments showed that 20 is the favourable number of pallets. The simulation results are summarized for 20 pallets in table 1.

<table>
<thead>
<tr>
<th>20 pallets</th>
<th>finished items (throughput)</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine No</td>
<td>0...2 hours</td>
</tr>
<tr>
<td>2/1</td>
<td>120</td>
</tr>
<tr>
<td>2/2</td>
<td>120</td>
</tr>
<tr>
<td>2/3</td>
<td>118</td>
</tr>
<tr>
<td>3</td>
<td>347</td>
</tr>
<tr>
<td>4</td>
<td>345</td>
</tr>
<tr>
<td>5</td>
<td>344</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

| finished items | 346 | 1786 | 1440 |
| finished items | 346 | 1786 | 1440 |
| circulated pallets | 11 | 11 | 0 |

Table 1

Dr. Thorsten Pawletta, Student B. Strauch, Universität Rostock, FB Informatik, Albert-Einstein-Str. 21, PF 999, D-2500 Rostock 1, Germany; Tel: +49-(0)381-444424; Fax: +49-(0)381-446089; E-mail: pawel@informatik.uni-rostock.de
Comparison 2 - POSES

Short description of POSES V4.3

The simulation system POSES (Prädikat-Transition-Netz-Orientiertes-Simulations- und Entwurfs-System) developed by the Technical University Chemnitz and implemented in version 4.3 by GPC mbH Chemnitz allows modelling and simulation based on extended predicate transition nets. Extensions to predicate transition nets are (fix or tochaistic value dependent) time consuming transitions, ree matching expressions on arcs, additional boolean conditions for transition concession, special access mechanisms or predicates (ram, lifo, fifo, liforam, liforam), logical token generating interrupts and so on.

The models have to be specified using the POSES-language. In this language the user has to define data structures or tokens and predicate types like in the programming language PASCAL. Also the net structure with all necessary arc expressions has to be defined in this language.

The POSES-Editor, Compiler, Linker and Generator are a POSES-development shell including tools to create independent executable simulation programs. Nearly all parameters (consumption time, capacities, priorities, okens, states, lifeness, trace parameters, ...) of the modelled net elements can be defined or changed by the user during experiment sessions.

By using high level Petri nets the abstraction level for modelling and simulation depends only on the user's selection. Global and detailed aspects are possible in the same model. Models of ready-made net substructures are also useful.

POSES offers the inclusion of user-defined PASCAL or routines. In this way POSES is also a simulation environment to develop a test control software on a level chosen by the user. POSES applications are simulation services for plant and warehouse logistics, organisation, computer communication and control software development problems.

Model description

The full net model is segmented into 8 equally structured net modules. Buffer components like B1, B2, the rest buffer behind work stage B2', input shift Sx and output shift Sy are modelled by predicates. All work stations Ax, all transportation flows into B1, B2, B2' are modelled by time consuming transitions. The pallets flowing through the system are represented by data tokens containing a record structure like a work order paper. Depending on the data state of these records the control mechanism is implemented as conditions and matching masks in arc expressions.

Experimentation Results

The results of the simulation are given in the table:

<table>
<thead>
<tr>
<th>Number of pallets</th>
<th>total throughput</th>
<th>average throughput time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1454</td>
<td>282</td>
</tr>
<tr>
<td>16</td>
<td>1455</td>
<td>302</td>
</tr>
<tr>
<td>17</td>
<td>1454</td>
<td>322</td>
</tr>
<tr>
<td>18</td>
<td>1457</td>
<td>341</td>
</tr>
<tr>
<td>19</td>
<td>1459</td>
<td>360</td>
</tr>
<tr>
<td>20</td>
<td>1462</td>
<td>380</td>
</tr>
<tr>
<td>21</td>
<td>1457</td>
<td>420</td>
</tr>
<tr>
<td>22</td>
<td>1460</td>
<td>424</td>
</tr>
<tr>
<td>23</td>
<td>1462</td>
<td>438</td>
</tr>
<tr>
<td>24</td>
<td>1440</td>
<td>465</td>
</tr>
<tr>
<td>25</td>
<td>1439</td>
<td>485</td>
</tr>
<tr>
<td>40</td>
<td>1440</td>
<td>785</td>
</tr>
</tbody>
</table>

Dipl.-Ing. Bert Olshme, Gesellschaft für Prozeßautomation & Consulting mbH, Senefelder Str. 38, D-0-9022 Chemnitz, Tel: +49-(0)371 50593, Fax: +49-(0)371 50594.
Comparison 2 - EXAM

1. A description of EXAM

The general purpose system EXAM is intended to support all stages of simulation process: the model description, the experiment description and the simulation itself. In accordance with this EXAM contains two separate frames: Model Description Frame (MDF) and Experiment Description Frame (EDF). In order to apply EXAM to a definite area, one needs to put necessary modules (reflecting the main features of the process) to a special library. Because of the modular concept, the simulation model can be represented as a hierarchy of elements linked to each other. MDF makes it possible to view and modify both structure of the model and any parameters of elements (including procedural ones) without any reprogramming. EDF enables to include any standard or non-standard methods and combine them to design complex experiments which can be carried out even with several models.

The base language for EXAM is object-oriented Turbo Pascal. EXAM has an interactive shell, working under MS Windows 3.0 or its higher versions, which is intended to give the user the possibility to describe models and experiments without knowledge of any programming and mathematics.

2. Model description

In order to illustrate the possibilities of EXAM, we used two representations of the model. The general scheme of these models is the same in both cases and is shown in Figure 1, which is actually a representation appearing on the screen during the work in MDF. In both cases EXAM was extended with a set of necessary modules, based on the internal mathematical model (aggregative one). In the first representation each element shown in Figure 1 was built from a single module describing the dynamics of the station as a whole. Different elements are obtained from the module by specifying its parameters. In the second representation each element from Figure 1 is actually a subsystem consisting of several other elements, describing the dynamics of the station, see Figure 2. The links between the elements in both cases reflect both pallets flow and artificial information signals related to possible blocking of the belts.

3. Results

In order to illustrate the work of EDF, we organized an optimization of the system by Fibonacci’s method, using the objective function $Q = c_1T_1 + c_2T_2$, where $c_1 = 1$, $c_2 = -1/400$ (actually, they have been taken arbitrarily), $T_1 = \text{total throughput}$, $T_2 = \text{average throughput time}$. A general scheme of the experiment is shown in Figure 3. The steps of the optimization for the first model are collected in the table:

<table>
<thead>
<tr>
<th>Number of pallets</th>
<th>$T_1$</th>
<th>$T_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>1411</td>
<td>693.756</td>
</tr>
<tr>
<td>21</td>
<td>1409</td>
<td>429.674</td>
</tr>
<tr>
<td>13</td>
<td>1155</td>
<td>323.922</td>
</tr>
<tr>
<td>26</td>
<td>1410</td>
<td>532.318</td>
</tr>
<tr>
<td>18</td>
<td>1396</td>
<td>371.193</td>
</tr>
<tr>
<td>23</td>
<td>1411</td>
<td>470.103</td>
</tr>
<tr>
<td>20</td>
<td>1404</td>
<td>409.519</td>
</tr>
<tr>
<td>22</td>
<td>1410</td>
<td>449.886</td>
</tr>
<tr>
<td>21</td>
<td>1409</td>
<td>429.674</td>
</tr>
</tbody>
</table>

The optimal number of pallets is 21. The results for the second model are very close to those for the first one (the differences are only due to the randomness of the initial states). So, we omit them.

4. Technical Data

The above model was run on an IBM AT 386/387 compatible computer operating at 20 MHz. The total run time at each step of optimization took from 0.5 to 3 min (depending on the number of pallets and, therefore, on the number of events).

For information and comments, please contact Prof. Dr. Vladimir Kalashnikov, Institute of System Analysis, Russian Academy of Sci., 9, Prospect 60 let Oktyabrya, 117312 Moscow, Russia; Fax: +7-095-9382209; Telex: 411237 POISK. E-mail: person@vniis.uucp.free.msk
Comparison 2 - Taylor II

1. Description of Taylor II

Taylor is a Dutch product developed by F&H Logistics and Automation B.V. in Tilburg, the Netherlands, since 1986. In mid 1992 the package, now called Taylor II, received a complete new structure. Taylor II is developed for all kinds of discrete event simulation and offers a wide range of special functions for processes in production and logistics.

Modeling in Taylor II starts with building a layout of different element types like machine, buffer, conveyor, etc. The second step is to create one or more routings. Now the model runs immediately with the standard 2D-animation. This offers the possibility of a visual debug. The third step, detailing the model, is done by filling out parameter masks. Typical parameters are capacities, breakdown behaviour, cycle-times, etc.

At strategic points you often have to make complex decisions regarding where products are sent. For this purpose the fixed addresses in the routings can be replaced by TLI-statements. TLI (Taylor Language Interface) is an easy to use macro language that enables you to define rules for order picking, assembly, complex guiding and receiving strategies. Furthermore, the package includes many features for pre- and user-defined analysis, animation and presentation. Taylor II runs on PC with MS-DOS or compatible. An MS-Windows version will appear in 1993.

2. Model Description

For the given problem only two types of elements were necessary: machines for every $S_x$, $S_y$, and $A_x$; conveyors for every (part of) $B_x$ and the connection of the subsystems. The pallets (products in Taylor) were given attributes for individual information and user-defined variables for calculating the processing time at $A_6$ and measuring the throughput time. The attribute values of the pallets were stored in a matrix which has been used as a decision table. At every $S_x$ there is a query in the routing and depending on the table values (0 or 1) the pallet is sent to $B_2$ or $B_1$. $B_1$ is always possible just in case that there is not enough empty space on $B_2$.

Taylor II does not offer the possibility to build a model out of submodels. But, you only have to specify the parameters you need. This in combination with the easy creation of layout and routings, accounts for very fast modeling. The following figure shows the model in standard 2D-representation with an additional background drawing.

3. Results

When looking at the results given in the table below you find an optimum of 16 pallets with an average throughput time of 320 s. The maximum throughput varies between 1440 and 1441. In the model, the unloading and loading at $A_1$ is seen as one operation. Pallets are counted when the operation starts. When collecting data from the 120th to the 600th minute it happens that a pallet is counted but not finished when the simulation stops. The only way to avoid this is to build a more detailed model.

For comments, questions or info please contact Dirk Werner, F&H Simulationsoftware GmbH, Neubrückstraße 4, D-40213 Düsseldorf I. Tel: +49-211-322151, Fax: +49-211-322897.

<table>
<thead>
<tr>
<th>number of pallets</th>
<th>production per day</th>
<th>throughput time [s]</th>
<th>utilization $A_x$ [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>14</td>
<td>299.9</td>
<td>30.0</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>310.4</td>
<td>67.7</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>306.0</td>
<td>73.5</td>
</tr>
<tr>
<td>16</td>
<td>14</td>
<td>320.0</td>
<td>75.0</td>
</tr>
<tr>
<td>17</td>
<td>14</td>
<td>340.0</td>
<td>75.0</td>
</tr>
<tr>
<td>18</td>
<td>14</td>
<td>360.0</td>
<td>75.0</td>
</tr>
<tr>
<td>19</td>
<td>14</td>
<td>340.0</td>
<td>75.0</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>400.0</td>
<td>75.0</td>
</tr>
<tr>
<td>25</td>
<td>14</td>
<td>498.4</td>
<td>75.0</td>
</tr>
<tr>
<td>30</td>
<td>14</td>
<td>598.1</td>
<td>75.0</td>
</tr>
<tr>
<td>40</td>
<td>14</td>
<td>776.8</td>
<td>75.0</td>
</tr>
</tbody>
</table>
**Comparison 2 - PCModel**

PCModel is a full-function simulation language and graphic animation system with an interactive session control facility providing extremely responsive interaction with the simulation as it executes.

The object-based language allows to define 6 integer and 2 time variables for each object, in the simulation equal to a pallet. While the location dimensions of all A2 and the A3, A4, A5 stations are the same, it is easy to create submodels for each of the four subsystem types with relative locations. The conveyor B1 represents the main-path in the model. At each Sx-location in front of a submodel, the pallet is asked if the process of the station ahead is done or not, and if not, if the buffer in front of the operation location is able to absorb the pallet. After being processed in the subsystem, the appropriate object variable is decremented. If a desired location is blocked the pallet waits at place until the location is free again. With this PCModel feature it is not necessary to create a FIFO-buffer in front of the process locations.

**Results of running the model with 20, 40 and 60 pallets:** The number of throughputs stayed nearly the same with 20, 40 or 60 pallets in the system. During the animation the system seemed to be well balanced with 20 parts. Every station was well occupied, even A6 sometimes (1.7 hours of 8), this shows the result of processed pallets in A6. More parts in the system seem to make the subsystem A2 to a bottleneck. A change of logical control of A6 to a fourth station A2 shows that the throughput did not rise up, because of the new bottleneck stations A3, A4 and A5, but A6 was as well frequent with a full buffer in front of it, as the 'original' stations A2. This is a sign that the processing times of the system are not optimally synchronised. A partition of the processes shows that A6 substituted to 56% station A3, to 30% A4 and only to 14% A5 with 20 parts in system. The other runs show an equal substitution of A3, A4 and A5, (figure 1).

The more pallets are in the system the higher the average and maximum throughput time rises up. The longest throughput time of a pallet with 20 parts in the system was 21 minutes, with 60 pallets instead nearly 2 hours. The average throughput time stayed, except at the beginning, constant and variation was only little but on different levels: 6.6 minutes for 20 pallets, 13.3 minutes for 40 pallets and 20 minutes for 60 pallets in the system. (figure 2 and 3).

![Figure 2](image)

![Figure 3](image)

**Development and execution times:** The model required 8 hours to design and debug the logic and 3 hours for the overlay (figure 4). The DIPLAN Corp. developed analyzing software tools for PCModel report files for different needs. For this simulation the tools needed just a configuration file. This took about half an hour.

![Figure 4](image)

The model execution time varied depending upon the number of parts in the system. With 20 parts in the system, the model executed in 5 minutes on an i386 IBM PC compatible operating at 40 MHz. With 60 parts in the system the model executed in less than 15 minutes on the same computer.

For more information and comments please contact:
Comparison 2 - MOSYS

Description of MOSYS

MOSYS is a complex modularly structured simulation tool developed by the Fraunhofer Institute for Production Systems and Design Technology (IPK) Berlin. The tool enables the user to create and evaluate models of any discrete system with the desired degree of detail. The basic philosophy of the simulation tool is that any system can be generated by using five different types of elementary building blocks which can be composed in subsystems of the considered model on an arbitrary number of hierarchical levels. Thus the user is enabled to create models following either a topdown or a bottom up strategy what makes the work essentially easier. The software runs on different platforms as IBM/370, VAX-stations and Unix-PCs.

Model Description

The system's model was generated in a quite detailed way. The topological model shown below reflects the real distances and speeds of the transportation facilities of the system.

![Topological layout of the system](image)

Behind each assembly station Ax in the functional model there is a number of test elements determining the further route of parts through the system depending on their status (for instance in the picture below the three stations A2 are shown with the logical flow for a part which has already passed station A6 before it was processed on A2_x (x=1,2,3).

![Section of the functional operation plan](image)

Simulation Results

The main goal of the simulation was to find out the optimal number of pallets in the system and to determine the cycle time of the pallets (i.e. the time passing between fixing a part on a pallet and taking it apart). The results are given in the following diagram.

![Throughput within the last 8 hours](image)

The system has its maximum throughput for 25 - 27 pallets. The respective cycle times are given in the following diagram:

![Average cycle times](image)

Comparing the simulation runs and taking into account the pallet cycle times, too, it turns out that the optimum number of pallets in the system is about 25-26. Here the throughput is at the maximum and the pallet cycle times are not essentially larger than the overall processing time, i.e. waiting times in front of the stations and the time for additional turns around on the belt are very small. For a higher number of pallets in the system the throughput remains relatively stable up to 40, while the cycle time increases. For more than 40 pallets the throughput strongly declines due to blocking effects and, finally for 60 pallets in the system deadlocks occur.

For further information please contact: Markus Rabe or Norbert Deul: IPK Berlin, Pascalstraße 8-9, D-10587 Berlin, Tel: +49-(0)30 39006248, Fax: +49-(0)30.3911037.
Comparison 2 - CASSANDRA 3.0

Introduction

In SNE 4, March 1992, we already have reported the solution of the problem using an earlier version CASSANDRA 2.1. Here a qualitatively higher level solution using the entirely new version is presented. Beyond the new graphic I/O interface and animation two basic aspects should be mentioned: 1) The experiments can be controlled by intelligent demons finding the optimum determined by the user automatically [1] (see Fig. 1) 2) The models are represented internally by Knowledge Attributed Petri Nets (KAPN) [2] enabling the individual workpieces to carry the technological prescriptions and state of manufacturing with them in a naturally and easily describable way.

The experiment

![Diagram of Experimental Frame and Model](image1)

The graphic representation of model is shown in Fig. 2, together with the demon supervising the experiment. (Internally the models are represented by KAPN sub-networks.)

Two versions of the demon controlled experiment are presented. In the first experiment the demon was instructed to increase the number of pallets step by step and the throughput, transfer time, as well as the standard deviation of the product transfer time were recorded. The results can be seen in Fig. 3.

In the next experiment the starting point was that we have no knowledge in advance about the system parameters which can be obtained. The demon was instructed to find a complex measure regarded as an optimum as follows: "Find the maximum throughput and decrease it until the value of 90% in order to reduce the product transfer time!" In case we regard this as an optimum the value of 13 pallets has been obtained as can be seen from the results of the search procedure shown in Fig. 4. Obviously the demon(s) in the system can be instructed the obtain various optimums of the weighted values of different system parameters.

References


For information and comments, please phone or fax or write to: Prof. Dr. A. Jävor, KFKI Research Institute for Measurement and Computing Techniques of the Hungarian Academy of Sciences, H-1525 Budapest, P.O.Box 49, Hungary, Tel: +36 1 1699499, Fax: +36 1 1553894, E-mail: h7023jav@ella.hu
Comparison 2 - DSIM

The simulation system DSIM has been developed at the Dept. Simulation Techniques at Technical University Vienna, supported by a grant of the "Bundesministerium für Wissenschaft und Forschung" of Austria.

DSIM is a discrete simulation system working in an windows environment (PC) and consists of the three modules SIMSHELL, SIMPAINT, and SIMSTAT.

The module SIMSHELL offers graphical modelling of a process flow and menu-driven control of experiments and the storage of results. Icons represent subprocesses, connections between the icons show the flow of entities (workpieces, information, etc., fig. 2). The description of the subprocesses is stored in one or more model libraries; these submodels are defined in terms of coloured, time dependent Petri nets. An expert user may modify or define subprocesses and libraries within the module SIMPAINT. Control or routing strategies may be defined either by means of Work Tables at runtime in SIMSHELL (centralized control) or by means of decisions in the Petri net description in SIMPAINT.

The simulator itself scans the Petri net description and schedules immediately an action, if necessary (time event); two versions are available: one with and one without deadlock detection and deadlock handling. Analysis of the stored data may be performed within the module SIMSTAT, which offers basic display features and interfaces for postprocessing (e.g. EXCEL).

DSIM is free software and available from the server simserv.tuwien.ac.at. Further developments will include e.g. optimization with genetic algorithms.

Model description

In modelling the investigated process for the stations Ai submodels of a predefined library were modified slightly, for the load/unload station a new subprocess was defined. Fig. 1 shows the representing icon and parts of the Petri net description developed in SIMPAINT.

Within SIMSHELL the model was build up by subprocesses of the model library (fig.2, six simple stations, one intelligent station, and the load/unload station with a delay element). When opening the icons, specific parameters for the subprocesses and initial values, e.g. the number of pallets, can be chosen. During a simulation run (menu Simulate) relevant selected data are stored, additionally a simple animation is offered: bar charts with absolute values or average values show the status of the stations.

![Figure 2](image)

The table shows the results for different number of pallets:

<table>
<thead>
<tr>
<th>pallets</th>
<th>throughput [pallets]</th>
<th>av. cycle time [s]</th>
<th>av. time in system [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1344</td>
<td>21.43</td>
<td>278.58</td>
</tr>
<tr>
<td>20</td>
<td>1425</td>
<td>20.20</td>
<td>409.06</td>
</tr>
<tr>
<td>30</td>
<td>1416</td>
<td>20.34</td>
<td>549.36</td>
</tr>
<tr>
<td>40</td>
<td>1370</td>
<td>21.02</td>
<td>706.55</td>
</tr>
</tbody>
</table>

Postprocessing of the stored data can be performed within SIMSTAT by displaying the data (fig.3) or interfacing to programs like EXCEL.

![Figure 3](image)

Solution DESMO

Full Version

Lösung DESMO

Studie

D. Martinssen
A. Häuslein
Fachbereich Informatik
Universität Hamburg
1. Einleitung

Für den Vergleich von Simulationssoftware wurde in der Zeitschrift EuroSim - Simulation News Europe die Aufgabe gestellt, ein flexibles Fertigungssystem zu modellieren.

Die Aufgabenstellung befindet sich in der Ausgabe Number 1, March 91, p. 28. Weitere Spezifikationen sind in der Ausgabe Number 2, July 91, p. 26 gegeben.

Realisiert wurde das Modell mit DESMO, einem Simulationspaket auf Basis von Modula - 2, welches am Fachbereich Informatik der Universität Hamburg im Rahmen einer Diplomarbeit entwickelt wurde. ¹

Da im Modell viele parallele Handlungen abgebildet werden, wurde zur Realisierung der prozeßorientierte Ansatz gewählt. Zusätzlich werden noch höhere Synchronisationsmechanismen aus DESMO verwendet (Ressourcenwettbewerb, bedingtes Warten), die die Ablaufsteuerung in einigen Teilen wesentlich vereinfachen.

2. Modellbestandteile

Im folgenden werden die Modellbestandteile beschrieben, die sich nach der Interpretation der Aufgabenstellung ergaben.

Subsystem:

Die gesamte Fertigungsanlage besteht aus acht Subsystemen, im folgenden kurz System genannt, die von den Paletten entsprechend ihrer Bearbeitungsvorschrift durchlaufen müssen (s. Abb. 1, S. 12). Ein solches System besteht aus der Arbeitsstation, den Förderbändern vor und hinter der Station, die gleichzeitig als Puffer dienen, dem Hauptband auf dem Paletten an der Station vorbei können, sowie jeweils einem Schieber am Anfang und Ende des Systems, in dem Paletten vom Hauptband zur Station gelangen können und umgekehrt.

Die Bestandteile sind in einem Record zusammengefaßt, welcher im einzelnen folgende Komponenten besitzt:

```system_structure = RECORD
shift_place : CondQ.Object;
blocked : BOOLEAN;
buffer_cap, buffer_fill : CARDINAL;
station : Res.Object;
typ : [1..6];```


Der Puffer vor der Arbeitsstation sowie das Hauptband sind als einfache Zähler realisiert, die die aktuelle Füllung beinhalten. Daneben gibt es je einen Zähler für die maximal mögliche Füllung, die sich als Quotient aus der Länge des Bandes und der Palettenlänge ergibt, wobei nur ganzzahlige Lösungen erlaubt sind.

Die Arbeitsstation sowie der Puffer hinter der Station sind durch Ressourcen modelliert, wodurch sich hier die Ablaufsteuerung vereinfacht. Durch die interne Warteschlange einer Ressource kann ein möglicher Stau im Puffer einfach abgebildet werden. Die Palette wartet die Zeit ab, die sie für einen reibunglosen Durchlauf durch den Puffer benötigen würde und fordert dann die Ressource der Station an. In DESMO wird sie nun automatisch an das Ende der Warteschlange der Ressource eingereiht. Sollten sich noch andere Paletten vor ihr befinden, so stehen diese ebenfalls noch in der Warteschlange. Die Zeit, die eine Palette in der Warteschlange verbringt, ist somit identisch mit der Zeit, die sie im Puffer stehen würde, ohne sich fortzubewegen. Verlässt eine Palette die Station und gibt die Ressource frei, so wird von DESMO automatisch die erste wartende Palette in der Warteschlange (falls vorhanden) aktiviert. Sie setzt dann selbstständig ihre Handlungen fort. Ebenso kann eine Palette eine Arbeitsstation erst verlassen, wenn der Puffer hinter der Station frei ist. Falls dieser Puffer noch durch eine andere Palette besetzt ist, kann diejenige aus der Station nicht weiter und wird blockiert. Die Reaktivierung erfolgt auch hier automatisch in dem Moment, wo der hintere Puffer freigegeben wird.

Der zuerst eingeschlagene Weg, sowohl den Puffer als auch das Hauptband als Ressource zu modellieren, brachte ein Problem für die Ablaufsteuerung mit sich. Wenn eine Palette in einem System ankommt, in dem sie beide Wege durch das System nehmen kann, ist es möglich, daß die Kapazitäten des Puffers und des Förderbandes erschöpft sind und sie vorerst blockiert wird. Erst wenn eine Kapazität frei wird, kann sie ihren Weg durch das System fortsetzen. Bei der Modellierung mit Ressourcen und einer einstweiligen Blockierung müßte die Palette im voraus wissen, welche Ressource (Puffer oder Hauptband) als nächstes freigegeben wird, damit sie die "richtige" anfordert und bei
Freigabe derselben sofort weiterlaufen kann. Liegt dieses Wissen nicht vor, so kann es passieren, daß die Palette Resource A anfordert, Ressource B aber zuerst freigegeben wird. Dadurch würde die Palette unnötig blockiert werden, da sie in der internen Warteschlange der Ressource A solange blockiert wird und dort erst freikommt, wenn ihr eine Ressource A zugeteilt wird.

Es wurde daher der Weg gewählt, die Eingangsstelle durch ein Objekt für bedingtes Warten zu modellieren. Hier wird bei Ankunft der Palette überprüft, ob sie das System betreten kann. Ist dies nicht der Fall, wird sie vorerst blockiert. Bei einer für die Eintrittsbedingung relevanten Zustandsänderung, muß eine erneute Überprüfung dann explizit eingeleitet werden.

Die letzten Komponenten sind die Bearbeitungszeit einer Palette in einer Station und der Stationstyp. Letzterer beschreibt die Station des Systems, da es mehrere Systeme gibt, in denen derselbe Stationstyp realisiert ist.

Zwischentransportbänder:

Mit den Zwischentransportbändern sind die Verbindungen zwischen den einzelnen Systemen gemeint, wobei diese Verbindung zwischen den Systemen 4 und 5 sowie 8 und 1 entfällt. Falls keine Blockierung vorliegt, können Paletten direkt von einem System in das nächste überwechselt.

Eine solche Verbindung besteht aus einem Transportband sowie der entsprechenden Transportzeit. Auch diese beiden Komponenten sind in einem RECORD zusammengefaßt:

```pascal
connection_structure = RECORD
  c_belt : Res.Object;
  transport_time : SimTime;
END;
```


Bedingt durch das Fehlen dieser Transportbänder zwischen den Systemen 4 und 5 sowie 1 und 8 ergibt sich dort eine etwas andere Modellierung. Die Regelung wird hier von der Eingangsstelle des nächsten Systems übernommen (s. Ablaufsteuerung).

Die Zwischentransportbänder sind dem jeweiligen vorhergehenden System zugeordnet.
Paletten:

Mit den Paletten wird der wichtigste Bestandteil der Fertigungsanlage beschrieben. Da es die Paletten sind, die sich aktiv durch die Fertigungsanlage bewegen, erfolgt die Modellierung als Prozeß, d. h. jede Palette ist als eigenständiger Prozeß mit eigener Ablaufsteuerung realisiert. Für jede Palette wird ein solcher Prozeß erzeugt, der während der gesamten Simulationsdauer existiert, da sich die Paletten permanent im System befinden.

Für die Steuerung durch das System werden für jede Palette Attribute benötigt, auf die sie während der Simulation zugreifen kann. In DESMO wird ein jeweiliger Prozeß (hier: eine Palette) durch ein Entity repräsentiert, für welches Attribute in einem Record zusammengefaßt werden können. Bei der Erzeugung eines neuen Entity wird diesem ein Zeiger auf einen solchen Record mitgegeben, über den das Entity auf seine Attribute zugreifen kann. Im einzelnen hat der RECORD für die Paletten folgende Komponenten:

```plaintext
pallet_structure = RECORD
  load_time : SimTime;
  old_system, new_system : [1..8];
  stations : BITSET;
  station_elements : [0..4];
  in_station, from_station : BOOLEAN;
END;
```

Wenn die Palette in System 1 durch die Arbeitsstation A beladen wird, wird der Zeitpunkt dieser Aktion in der Variablen `load_time` festgehalten, der für die spätere Auswertung der Ergebnisse notwendig ist. Mit den beiden Variablen `old_system` und `new_system` kann sich die Palette im Gesamtsystem orientieren. Der Variable `stations` wird beim Beladen der Palette eine Menge zugewiesen, die die Arbeitsstationen enthält, in denen die Palette bearbeitet werden muß. Nach einer Bearbeitung wird die Menge genau um das Element reduziert, welches die aktuelle Arbeitsstation repräsentiert. Die Variable `station_elements` enthält die Anzahl der Elemente von `stations`. Dieser Zähler wird gebraucht, da in Modula-2 keine Funktion zur Ermittlung der Anzahl von Elementen einer Menge existiert. Der Zähler ist u. a. notwendig zur Entscheidung, ob eine Palette in einem System bearbeitet werden soll (s. Ablaufsteuerung). Ebenso für die Steuerung notwendig sind die boole'schen Variablen.
3. Modellannahmen und Modelldaten

Aus der Aufgabenstellung wurden die folgenden Annahmen abgeleitet:

- Als Zeitbasis wurden Minuten gewählt. Alle zeitbehaf teten Variablen im Modell beziehen sich auf diese Zeitbasis.

- Die Länge einer Palette wird mit 0.36m angegeben. Da die Länge der Puffer, der Hauptbänder und der Zwischentransportbänder jeweils ein Vielfaches von 0.4m ist, wird angenommen, daß eine Palette den Platz von 0.4m beansprucht, d. h. es können sich auf einem Transportmittel immer nur eine ganze Anzahl von Paletten befinden.

- Als Zeiteinheit (time_unit) wird die Zeit bezeichnet, die eine Palette benötigt, um 0.4m zurückzulegen. Sie berechnet sich aus der Geschwindigkeit der Förderbänder, die für jedes Band 18 m/min beträgt. (time_unit = 1/45 min.)

- Biegt eine Palette in den Puffer einer Arbeitsstation ab, so sind dazu 2 s = 1/30 min (shift_time) notwendig. Nachfolgende Paletten werden in ihrem Ablauf durch den Bandwechsel nicht beeinflußt und können, soweit keine Blockierung vorliegt, ihren Weg fortsetzen. Dies bedeutet z. B., daß auch zwei Paletten direkt hintereinander in den Puffer einer Station eintreten können, obwohl die erste Palette den Schieber noch nicht verlassen hat.

- Die Simulation soll mit der betrachteten Anzahl leerer Paletten im System beginnen, wobei sich die Paletten wahlweise auf den Hauptbändern befinden dürfen, jedoch nicht in den Puffern oder Arbeitsstationen. Letztere Annahme begrenzt die maximale Anzahl der Paletten im System auf 40. (Summe der Kapazitäten der Hauptbänder und Zwischentransportbänder.)

  Der Anfangszustand wird in diesem Modell erreicht, indem die Paletten über das Zwischentransportband hinter System 1 eingeschleust werden und sich gemäß ihrer Handlungsbeschreibung durch das Gesamtsystem bewegen. Die Simulation wird von dem Zeitpunkt an gerechnet, zu dem die letzte leere Palette das Gesamtsystem betritt.

- Auf Grund der Annahme, daß leere Paletten zu Anfang nicht in Puffern und Arbeitsstationen stehen dürfen, wird davon ausgegangen, das leere Paletten auch während der Simulation nicht in die Stationen abbiegen dürfen. Hiervon ausgenommen ist Station 1, in der leere Paletten beladen werden. Diese Annahme ist nur für die Steuerung in der Anfangsphase relevant, da sich danach keine leeren Paletten mehr im Gesamtsystem befinden.

Ebenso ist die umgekehrte Reihenfolge möglich, d. h. A2 wird als letzte Station durchlaufen.
Zusätzlich gibt es die flexible Station A6, die die Aufgaben von A3, A4 oder A5 übernehmen kann, wobei bei jedem Durchlauf einer Palette immer nur eine Maschine nachgebildet werden kann.

- Wird eine Palette durch Station A6 bearbeitet, so ist die Auswahl, ob Station A3, A4 oder A5 nachgebildet wurde, nicht festgelegt. Hier wird die erste der Stationen A3, A4, A5 als bearbeitet angesehen, die in der Menge der noch zu bearbeitenden Stationen gefunden wird.

- In Station A1 wird eine Palette ent- und beladen (jeweils 7,5 Sec). Als vereinfachende Annahme wird in der Anfangsphase eine leere Palette in A1 ebenfalls "entladen". Für die Ergebnisse ist dieses nicht relevant, da sich zum Reset-Zeitpunkt der Statistik (120. Min.) keine leeren Paletten mehr im System befinden.

4. Ablaufsteuerung

Im Hauptprogramm wird als erstes die Initialisierungsroutine aufgerufen, die die Systeme und Zwischentransportbänder einrichtet sowie den Filenamen für die Ausgabedateien und die Anzahl der Paletten im System (1..40) einliest.


Die Simulation beginnt dadurch, daß sich das Hauptprogramm für den Zeitraum der Simulation passiviert. Es wird zwischen-durch nur noch einmal aktiv, um nach der vorgegebenen Anlaufphase von 120 min die statistischen Zähler zurückzusetzen. Nach Ablauf der Simulationszeit wird vom Hauptprogramm aus die Ergebnisausgabe veranlasst.

Die Steuerung der Paletten ist in ihrer Prozeßbeschreibung enthalten.

Paletten-Prozeß:

Der Programmteil bis zur LOOP-Schleife dient zur Initialisierung der Palette und zum Einschleusen derselben in das System. Der eigentliche Ablauf wird innerhalb der LOOP-Schleife beschrieben.

Dies soll anhand einer möglichen Konstellation verdeutlicht werden: Beide Paletten kommen gleichzeitig am hinteren Schieber an und wollen über das Hauptband des nächsten Systems weiter. Die vom Hauptband kommende Palette muß in ihrem Ablauf jetzt solange blockiert werden, bis die von der Station kommende Palette den Schieber durchquert hat. Erst danach kann sich die vom Hauptband kommende Palette weiterbewegen, die nun unmittelbar an die andere Palette anstößt. Unterbleibt diese Blockierung, würde sich die Palette vom Hauptband auf die andere schieben, was ausgeschlossen sein soll.

Gesagtes gilt in gleicher Form für Konstellationen, bei denen Paletten innerhalb solcher kurzer Zeitabstände den Schieber erreichen, in denen eine Palette diesen noch nicht vollständig durchquert hat. Hier wird ebenfalls die später ankommende Palette bis zu dem Zeitpunkt blockiert, bis die erste das Ende des Schiebers erreicht hat.

Die Lösung dieses Problems erfolgt durch die Variable blocked, die bei dem Wunsch einer Palette ein System zu betreten u. a. abgefragt wird. Sie wird auf TRUE gesetzt, wenn eine Palette den Schieber betritt. Wenn sie das Ende des Schiebers erreicht hat, wird blocked wieder auf FALSE gesetzt und eine nachfolgende Palette kann ihren Weg fortsetzen, sofern auch die anderen Bedingungen erfüllt sind.

Für die Übergänge zwischen den anderen Systemen braucht diese Variable nicht verändert zu werden (Default = FALSE). Um hier ein System zu verlassen, wird die Ressource des nachfolgenden Zwischentransportbandes angefordert. Die Paletten werden automatisch in die Warteschlange der Ressource eingereiht und gemäß dieser Reihenfolge wird jeweils der ersten Wartenden die Ressource zugeteilt, d. h. es kann immer nur eine Palette weiter, während die anderen durch die nicht vorhandene Ressource automatisch blockiert werden.

Bei einer evtl. auftretenden Zeitgleichheit von Paletten steht die Palette von der Station aufgrund ihrer Priorität an erster Stelle der Warteschlange und bekommt die Ressource
zu erst zugeteilt. Die Ressource wird erst freigegeben, wenn die Palette, der sie zugeteilt wurde, über das Zwischentransportband gelaufen ist und in das nächste System eintreten kann. Eine nachfolgende Palette kann somit nicht mehr mit der ersten in Konflikt kommen.

Falls eine Palette nicht blockiert wurde, gibt es für sie zwei Wege durch ein System. Entweder biegt sie zur Station ab oder sie läuft geradeaus über das Hauptband. In beiden Fällen wird zuerst geprüft, welches System betreten wurde. Für die Systeme 5 und 1 ist weiterhin entscheidend, ob die Palette von der Station oder vom Hauptband des vorherigen Systems kommt. Im ersten Fall muß der Puffer hinter der Station freigegeben werden. Durch die Modellierung als Ressource erfolgt eine Aktivierung einer wartenden Palette automatisch. Wenn dagegen ein Platz auf dem Hauptband frei wird, muß dieses der Eingangsstelle explizit mitgeteilt werden, da eine evtl. blockierte Palette ihren Weg fortsetzen könnte.

Nach Eintritt der Palette in ein System besteht der Durchlauf im Wesentlichen aus dem Warten der Zeiteinheiten für den Durchlauf der Förderbänder sowie dem Belegen und Freigeben der Station und dem Puffer hinter der Station. Dabei wird angenommen, daß der Übergang vom Puffer zur Station ebenfalls eine Zeiteinheit benötigt. In System 1 mit Station A1 kommt noch zusätzlich nach Durchlauf der Palette durch die Station hinzu, daß die Statistik fortgeschrieben wird und die Attribute der Paletten für einen neuen Systemdurchlauf entsprechend gesetzt werden.

Beim Verlassen des Systems ist noch darauf zu achten, welches System als nächstes betreten werden soll. Bei System 5 oder 1 reiht sich die Palette gleich in die Eingangsstelle ein. Bei den anderen Systemen muß die Palette zuerst das Zwischentransportband anfordern und die Transportzeit warten, ehe sie die Eingangsstelle des nächsten Systems betreten kann.


Diese Regelungen gelten nur für volle Paletten, da leere Paletten ein System immer über das Hauptband durchlaufen. Ausnahme ist Station A1, in die leere Paletten zum Beladen abbiegen dürfen.
5. Simulationsergebnisse


<table>
<thead>
<tr>
<th>Anzahl Paletten</th>
<th>Durchsatz (Paletten)</th>
<th>mittl. Zeit einer Palette im System (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>939</td>
<td>5.1</td>
</tr>
<tr>
<td>15</td>
<td>1351</td>
<td>5.3</td>
</tr>
<tr>
<td>20</td>
<td>1408</td>
<td>6.8</td>
</tr>
<tr>
<td>25</td>
<td>1407</td>
<td>8.5</td>
</tr>
<tr>
<td>30</td>
<td>1409</td>
<td>10.2</td>
</tr>
<tr>
<td>35</td>
<td>1408</td>
<td>12.0</td>
</tr>
<tr>
<td>40</td>
<td>1409</td>
<td>13.7</td>
</tr>
</tbody>
</table>

Tab. 1: Ergebnisse
Abbildung 1: Aufbau der Fertigungsanlage
Anhang B: Programmlisting

(* Flexible Assembly System *)
(* see EUROSIM - Simulation News Europe, Number 1, Mar 1991, p. 28 *)
(* and EUROSIM - Simulation News Europe, Number 2, Jul 1991, p. 26 *)
(* Programmer : Dirk Martinissen *)
(* last change : 23.01.91 *)

MODULE EuroSim;

(* procedures of DESMO *)
FROM ProcessSimulation IMPORT Entity, New, Schedule, SetPriority,
       Attributes, Current, Hold, SimTime,
       Time, NOW, Reset, SetReportFile,
       SetErrorFile, CloseFiles;
FROM StringHdl IMPORT Copy, CardToString, Trim, Concat;
IMPORT Res, CondQ, Count, Tally;
FROM CondQ IMPORT WaitUntil, Signal;
FROM ReportIO IMPORT WriteString, WriteCard, WriteReal, WriteLn;

(* procedures of Modula-2 *)
FROM IO IMPORT RdStr, WrStr, WrLn, RdCard, WrCard, WrReal;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;

CONST shift_time = 1.0 / 30.0; (* time to shift from B1 to B2 or back *)
time_unit = 1.0 / 45.0; (* transport time for 0.4m on conveyor *)

TYPE (* attributes of a conveyor between two systems *)
   connection_structure = RECORD
      c_belt : Res.Object;
      transport_time : SimTime;
   END;

(* attributes of a system *)
system_structure = RECORD
   shift_place : CondQ.Object;
   buffer_cap, buffer_fill : CARDINAL;
   station : Res.Object;
   typ : [1..6];
   operation_time : SimTime;
   rear_buffer : Res.Object;
   conveyor_cap, conveyor_fill : CARDINAL;
   blocked : BOOLEAN;
END;

(* attributes of a pallet *)
pallet_structure = RECORD
   load_time : SimTime;
   old_system, new_system : [1..8];
   stations : BITSET;
   station_elements : [0..4];
VAR system : ARRAY [1..8] OF system_structure;
connection : ARRAY [1..8] OF connection_structure;

nr_of_tasks, i : CARDINAL;
pallet_attr : pallet_attributes;
throughput : Count.Object;
avg_throughput_time : Tally.Object;

PROCEDURE time_in_system () : REAL;
(* procedure for statistics *)
(* result: time of a pallet in the system *)
VAR attr : pallet_attributes;
BEGIN
  attr := Attributes (Current ());
  RETURN (Time () - attr ^. load_time);
END time_in_system;

PROCEDURE Init;

VAR c_cap : ARRAY [1..8] OF CARDINAL;
b_cap, kind
o_time : ARRAY [1..8] OF CARDINAL;
operation_time1, operation_time2,
operation_time3, operation_time4 : SimTime;
name, nr, name_with_nr,
name2, name2_with_nr,
name3, name3_with_nr : ARRAY [1..12] OF CHAR;
FileName
Extension
RF, EF,
i : ARRAY [1..8] OF CHAR;
: CARDINAL;

BEGIN

operation_time1 := 0.25;
(* operation time of A1 *)
operation_time2 := 1.0 / 3.0;
(* operation time of A3, A4, A5 *)
operation_time3 := 0.5;
(* operation time of A6 *)
operation_time4 := 1.0;
(* operation time of A2 *)

C_cap [1] := 5;  b_cap [1] := 3;  (* capacities of B1 (c_cap) *)
c_cap [8] := 5;  b_cap [8] := 3;
o_time[1] := operation_time1; kind[1] := 1; /* kind = stationtyp */

Copy("station_", name);
Trim(name);
Copy("shifting_", name2);
Trim(name2);
Copy("r_buffer_", name3);
Trim(name3);
(* initialization of the systems *)
FOR i := 1 TO 8 DO
  CardToString(i, 2, nr);
  Concat(name, nr, name_with_nr);
  Concat(name2, nr, name2_with_nr);
  Concat(name3, nr, name3_with_nr);
  WITH system[i] DO
    shift_place := CondQ.New(name2_with_nr, FALSE);
    buffer_cap := b_cap[i];
    buffer_fill := 0;
    station := Res.New(name_with_nr, 1);
    typ := kind[i];
    operation_time := o_time[i];
    rear_buffer := Res.New(name3_with_nr, 1);
    conveyor_cap := c_cap[i];
    conveyor_fill := 0;
    blocked := FALSE;
  END; /* WITH */
END; /* FOR */

Copy("connect_", name);
Trim(name);
(* initialization of the conveyors between the systems *)
FOR i := 1 TO 8 DO
  CardToString(i, 2, nr);
  Concat(name, nr, name_with_nr);
  WITH connection[i] DO
    c_belt := Res.New(name_with_nr, 1);
    transport_time := time_unit;
  END; /* WITH */
END;

(* initialization of statistic objects *)
throughput := Count.New("throughput");
avg_throughput_time := Tally.New("avg through", time_in_system);

WnStr("Please enter filename for output (<=8 characters) : ");
RdStr(FileName); WnLn;
Copy (FileName, RF);
Trim (RF);
Extension := ".RPT";
Concat (RF, Extension, RF);
Copy (FileName, EF);
Trim (EF);
Extension := ".ERR";
Concat (EF, Extension, EF);
SetReportFile (RF);
SetErrorFile (EF);

REPEAT
  Writeln("Please enter the maximal number of simultaneous pallets");
  WrLn;
  Writeln("in the system (1 <= x <= 40) -");
  nr_of_tasks := RdCard ();
UNTIL (1 <= nr_of_tasks) AND (nr_of_tasks <= 40);
END Init;

PROCEDURE Entry (pallet : Entity) : BOOLEAN;
(* procedure to check if a pallet can enter a system and if it *)
(* will enter the station of the system. Used in combination *)
(* with the object for conditional waiting of a system (shift_place) *)
VAR full_conveyor,
  full_buffer,
  work_in_station,
  turn_to_station : BOOLEAN;
attr : pallet_attributes;
BEGIN
  attr := Attributes (pallet);
  WITH system [attr^.new_system] DO
    full_conveyor := (conveyor_cap = conveyor_fill);
    full_buffer := (buffer_cap = buffer_fill);
  CASE attr^.new_system OF
    1 : work_in_station := (attr^.station_elements = 0); |
    2, 3, 4 : work_in_station := ((attr^.station_elements = 4)
      OR ((2 IN attr^.stations)
        AND (attr^.station_elements = 1)))); |
    5 : work_in_station := (3 IN attr^.stations); |
    6 : work_in_station := (4 IN attr^.stations); |
    7 : work_in_station := (5 IN attr^.stations); |
    8 : work_in_station := NOT (((2 IN attr^.stations)
        AND (attr^.station_elements = 1))
      OR (attr^.station_elements = 0)); |
END; (* CASE *)
turn_to_station := (NOT full_buffer
               AND work_in_station AND NOT blocked);
IF turn_to_station OR (NOT full_conveyor AND NOT blocked)
THEN IF turn_to_station
    THEN attr^.in_station := TRUE;
    ELSE attr^.in_station := FALSE;
END;
RETURN TRUE;
ELSE attr^.in_station := FALSE;
RETURN FALSE;
END;
END; (* WITH *)
END Entry;

PROCEDURE PalletProcess (pallet : Entity);
(* process description of a pallet *)
VAR attr       : pallet_attributes;
j             : CARDINAL;

PROCEDURE TickOffStation;
(* local procedure to remove a station *)
(* from the set of unmachined stations *)
VAR station_typ  : CARDINAL;
BEGIN
    WITH attr^ DO
    CASE system [new_system].typ OF
        2, 3, 4, 5 : EXCL (stations, system [new_system].typ);
        6 : station_typ := 3;
        WHILE (NOT (station_typ IN stations))
            DO INC (station_typ);
        END;
        EXCL (stations, station_typ);
    END; (* CASE *)
    DEC (station_elements);
END; (* WITH *)
END TickOffStation;

BEGIN
(* initialization of the pallet *)
attr := Attributes (pallet);
WITH attr^ DO
    load_time := Time ();
    old_system := 1;
    new_system := 2;
    stations := ();
    station_elements := 0;
    in_station := FALSE;
    from_station := FALSE;
END;
(* assignments to put a pallet in the system *)
SetPriority (pallet, 1);
Res.Acquire (connection [attr^.old_system].c_belt, 1);
SetPriority (pallet, 0);
Hold (connection [attr^.old_system].transport_time);

(* process description *)
LOOP
WITH attr^ DO
(* test to enter a system *)
WaitUntil (system [new_system].shift_place, Entry);
IF in_station
(* enter a station *)
THEN IF (new_system = 1) OR (new_system = 5)
THEN IF from_station
THEN (* come from buffer behind station *)
Res.Release (system [old_system].rear_buffer, 1);
SetPriority (pallet, 0);
system [new_system].blocked := TRUE;
INC (system [new_system].buffer_fill);
Hold (shift_time);
ELSE (* come from B1 *)
DEC (system [old_system].conveyor_fill);
Signal (system [old_system].shift_place);
system [new_system].blocked := TRUE;
INC (system [new_system].buffer_fill);
Hold (time_unit);
END;
system [new_system].blocked := FALSE;
Signal (system [new_system].shift_place);
ELSE Res.Release (connection [old_system].c_belt, 1);
INC (system [new_system].buffer_fill);
END;
(* wait time to change from B1 to B2 *)
Hold (shift_time);
(* wait time to go through the buffer *)
FOR j := 1 TO (system [new_system].buffer_cap) 
DO Hold (time_unit);
END;
(* acquire station *)
Res.Acquire (system [new_system].station, 1);
DEC (system [new_system].buffer_fill);
Signal (system [new_system].shift_place);
(* wait time to go in station *)
Hold (time_unit);
from_station := TRUE;
IF (new_system = 1)
THEN (* wait time to unload pallet *)
Hold (system [new_system].operation_time / 2.0);
(* statistics *)
Count.Update (throughput, 1);
Tally.Update (avg_throughput_time);
(* new initialization of the pallet *)
stations := (2, 3, 4, 5);
station_elements := 4;
load_time := Time ();
(* wait time to load pallet *)
ELSE (* wait operation time *)
    Hold (system [new_system].operation_time);
    TickOffStation;
END;
(* acquire buffer behind station *)
Res.Acquire (system [new_system].rear_buffer, 1);
(* release station *)
Res.Release (system [new_system].station, 1);
Hold (time_unit);
new_system := (new_system MOD 8) + 1;
old_system := (old_system MOD 8) + 1;
SetPriority (pallet, 1);
IF NOT ((new_system = 1) OR (new_system = 5))
    THEN (* leave system over conveyor between two systems *)
        Res.Acquire (connection [old_system].c_belt, 1);
        Res.Release (system [old_system].rear_buffer, 1);
        SetPriority (pallet, 0);
        Hold (shift_time);
        Hold (connection [old_system].transport_time);
END;
ELSE (* enter B1 *)
IF (new_system = 1) OR (new_system = 5)
    THEN IF from_station
        THEN (* come from buffer behind station *)
            Res.Release (system [old_system].rear_buffer, 1);
            SetPriority (pallet, 0);
            system [new_system].blocked := TRUE;
            INC (system [new_system].conveyor_fill);
            Hold (shift_time);
        ELSE (* come from B1 *)
            DEC (system [old_system].conveyor_fill);
            Signal (system [old_system].shift_place);
            system [new_system].blocked := TRUE;
            INC (system [new_system].conveyor_fill);
            Hold (time_unit);
        END;
        system [new_system].blocked := FALSE;
        Signal (system [new_system].shift_place);
    ELSE Res.Release (connection [old_system].c_belt, 1);
        INC (system [new_system].conveyor_fill);
    END;
(* wait time to go through B1 + shiftplace *)
FOR j := 1 TO (system [new_system].conveyor_cap) + 1
    DO Hold (time_unit);
END;
from_station := FALSE;
new_system := (new_system MOD 8) + 1;
old_system := (old_system MOD 8) + 1;
IF NOT ((new_system = 1) OR (new_system = 5))
THEN (* leave system over conveyor between two systems *)
    Res.Acquire (connection [old_system].c_belt, 1);
    DEC (system [old_system].conveyor_fill);
    Signal (system [old_system].shift_place);
    Hold (time_unit);
    Hold (connection [old_system].transport_time);
END;
END; (* WITH *)
END; (* LOOP *)
END PalletProcess;

PROCEDURE SelfReport;
(* report procedure for the results *)
BEGIN
    WriteString ("Flexible Assembly System");
    WriteLn;
    WriteString ("--------------------------");
    WriteLn; WriteLn;
    WriteString ("Number of pallets in the system : ");
    WriteCard (nr_of_tasks, 2);
    WriteLn; WriteLn; WriteLn;
    WriteString ("Clock Time = ");
    WriteReal (Time (), 3);
    WriteLn;
    WriteString ("***************************************************************************************
******")
    WriteLn;
    WriteString (**
    
    
    **")
    WriteLn;
    WriteString (**
    
    
    **")
    WriteLn;
    WriteString (**
    
    
    **")
    WriteLn;
    WriteString ("***************************************************************************************
******")
    WriteLn; WriteLn;

    Count.ReportAll;
    Tally.ReportAll;
END SelfReport;
BEGIN (* Main *)

Init:

(* mark buffer in front of A1 as full *)
(* => no pallet can enter A1 until all pallets are in the system *)
system [1].buffer_fill := system [1].buffer_cap;

(* generate and start pallet processes *)
FOR i := 1 TO (nr_of_tasks - 1) DO
    NEW (pallet_attr);
    Schedule (New ("Pallet ", PalletProcess, pallet_attr), NOW ());
END:

Res.Acquire (connection [1].c_belt, 1);
Res.Release (connection [1].c_belt, 1);

(* last pallet *)
NEW (pallet_attr);
Schedule (New ("Pallet ", PalletProcess, pallet_attr), NOW ());

(* clear buffer in front of station A1 => start simulation *)
system [1].buffer_fill := 0;
Signal (system [1].shift_place);

(* reset statistic after 120 min *)
Hold (120.0);
Reset;

(* wait simulation time (8 h = 480 min *)
Hold (480.0);
SelfReport;

CloseFiles;
END EuroSim.
Anhang C: Report - Ausgaben

Flexible Assembly System
---------------------------------

Number of pallets in the system: 10

Clock Time = 600.311

******************************************************************************
  *
  *
  REPORT
  *
  *
******************************************************************************

COUNTS
-------

Title  (Re)set  Obs
---------  -----  -----
throughput  120.311  939

TALLIES
-------

Title  (Re)set  Obs  Mean  Std.Dev  Min  Max
---------  -----  ----  ------  -----  ----  ----
avg through  120.311  939  5.111  0.357  4.755  5.467

Flexible Assembly System
---------------------------------

Number of pallets in the system: 15

Clock Time = 600.533

******************************************************************************
  *
  *
  REPORT
  *
  *
******************************************************************************

COUNTS
-------

Title  (Re)set  Obs
---------  -----  -----
throughput  120.533  1351
### TALLIES

<table>
<thead>
<tr>
<th>Title</th>
<th>(Re)set</th>
<th>Obs</th>
<th>Mean</th>
<th>Std.Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg through</td>
<td>120.533</td>
<td>1351</td>
<td>5.333</td>
<td>0.827</td>
<td>3.722</td>
<td>6.589</td>
</tr>
</tbody>
</table>

Flexible Assembly System

Number of pallets in the system : 20

Clock Time = 600.756

### COUNTS

<table>
<thead>
<tr>
<th>Title</th>
<th>(Re)set</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>120.756</td>
<td>1408</td>
</tr>
</tbody>
</table>

### TALLIES

<table>
<thead>
<tr>
<th>Title</th>
<th>(Re)set</th>
<th>Obs</th>
<th>Mean</th>
<th>Std.Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg through</td>
<td>120.756</td>
<td>1408</td>
<td>6.819</td>
<td>0.971</td>
<td>4.591</td>
<td>13.066</td>
</tr>
</tbody>
</table>
Flexible Assembly System

Number of pallets in the system : 25

Clock Time = 600.978

REPORT

COUNTS

Title   (Re)set  Obs
throughput  120.978  1407

TALLIES

Title   (Re)set  Obs  Mean  Std.Dev  Min  Max
avg through  120.978  1407  8.518  2.655  5.363  28.840

Flexible Assembly System

Number of pallets in the system : 30

Clock Time = 601.200

REPORT

COUNTS

Title   (Re)set  Obs
throughput  121.200  1409
### Totalies

<table>
<thead>
<tr>
<th>Title</th>
<th>(Re)set</th>
<th>Obs</th>
<th>Mean</th>
<th>Std.Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg through</td>
<td>121.200</td>
<td>1409</td>
<td>10.231</td>
<td>4.636</td>
<td>5.422</td>
<td>42.605</td>
</tr>
</tbody>
</table>

#### Flexible Assembly System

Number of pallets in the system: 35

Clock Time = 601.422

### Counts

<table>
<thead>
<tr>
<th>Title</th>
<th>(Re)set</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>121.422</td>
<td>1408</td>
</tr>
</tbody>
</table>

### Totalies

<table>
<thead>
<tr>
<th>Title</th>
<th>(Re)set</th>
<th>Obs</th>
<th>Mean</th>
<th>Std.Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg through</td>
<td>121.422</td>
<td>1408</td>
<td>11.953</td>
<td>5.839</td>
<td>5.847</td>
<td>60.510</td>
</tr>
</tbody>
</table>
Flexible Assembly System

Number of pallets in the system: 40

Clock Time = 601.644

**REPORT**

**COUNTS**

<table>
<thead>
<tr>
<th>Title</th>
<th>(Re)set</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>121.644</td>
<td>1409</td>
</tr>
</tbody>
</table>

**TALLIES**

<table>
<thead>
<tr>
<th>Title</th>
<th>(Re)set</th>
<th>Obs</th>
<th>Mean</th>
<th>Std.Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg through</td>
<td>121.644</td>
<td>1409</td>
<td>13.653</td>
<td>7.558</td>
<td>5.906</td>
<td>70.534</td>
</tr>
</tbody>
</table>
Studie

„Leistungsfähigkeit und Verfügbarkeit
diskreter Simulationsssoftware“

G. Pflug, Universität Wien

F. Breitenecker, Technische Universität Wien
3.2.1. GPSS/H

3.2.1.1. Allgemeines


3.2.1.2. Produktdaten

3.2.1.2.1. Hersteller:
Wolverine Software Corporation
4115 Annadale Road, Annadale VA 22003-2500
Tel: 703-750-3910 Fax: 703-642-9634

3.2.1.2.2. Distributor:
Scientific COMPUTERS GmbH
Franzstraße 107, Postfach 1865
D-5100 Aachen
Tel: (0241)26041/42 Fax: (0241)44983

3.2.1.2.3. Plattformen, Systemerfordernisse:
Es existieren GPSS/H-Versionen für PCs, Unix Workstations (Sun-3, SPARC, Hewlett-Packard oder Silicon Graphics), für VAXes unter VMS und für IBM Mainframe Computer.

3.2.1.2.4. Preis:
auf Anfrage, je nach Version (PC-Version etwa DM 5000.-)

3.2.1.2.5. Verfügbarkeit:
GPSS/H 2.0 Versionen, die unter DOS laufen: Student GPSS/H bietet alle Eigenschaften des kommerziellen GPSS/H und ist nur in der Modellgröße (höchstens 100 Blocks bzw. 200 Statements) begrenzt. Personal GPSS/H läuft auf 286er PCs, die Modellgröße ist auf 640K RAM begrenzt. GPSS/H Professional läuft auf 386er PCs, benützt den "DOS Extender" und macht es möglich mit 2 oder 4 Megabyte RAM große Modelle zu simulieren.

3.2.1.3. Evaluierungsresultate

3.2.1.3.1. User Interface
GPSS/H stellt keine eigene Editierumgebung zur Verfügung, jeder Texteditor, der Files in ASCII-Format speichert, kann verwendet werden.

3.2.1.3.2. Dokumentation
Banks J., Carson J.S.: "Getting Started with GPSS/H". Wolverine Software Corporation, Annadale, USA, 1989 (wird mit der Vollversion mitgeliefert, gutes Lehrbuch)

3.2.1.3.3. Grundprinzipien
GPSS/H arbeitet mit blockorientierter Beschreibung, die Abarbeitung erfolgt ereignisorientiert. Grundlage der Modellbildung sind Transactions, auch units of traffic genannt, die von Block
zu Block durch das System wandern. GPSS/H Model Files können in Batch Mode oder Test Mode (Simulation kann Schritt für Schritt verfolgt werden) ausgeführt werden. Basissprache ist Fortran.

3.2.1.3.4. Granularität
GPSS/H ist durch bis zu 80 verschiedene Blöcke (je nach Version) sehr fein strukturiert.

3.2.1.3.5. Grenzen
Da GPSS/H einfach strukturiert ist, besitzt es keine Softwaregrenzen.

3.2.1.3.6. Modellbeschreibung

3.2.1.3.7. Experimentbeschreibung
Da GPSS/H Modell-, und Experimentbeschreibung nicht trennt: siehe 3.2.1.3.6.

3.2.1.3.8. Output-Analysen

3.2.1.3.9. Import/Export von Datenformaten
ASCII Daten können von einem File oder vom Keyboard eingelesen werden, ebenso können Daten auf ein File oder auf den Bildschirm ausgegeben werden.

3.2.1.3.10. Animation
PROOF von Wolverine bietet eine unabhängige, objektorientierte und leicht erlernbare Postanimation. Das GPSS/H Programm wird um Animationsblöcke oder -macros erweitert. PROOF liest dann ein vom Anwender gestaltetes Layout File (graphischer Aufbau, Farben, Objekte, ...) ein und arbeitet das vom Simulator erzeugte Trace File zeitsynchronisiert ab.

3.2.1.4. Allgemeine Vorteile
Für GPSS/H spricht das weite Anwendungsfeld, langjährige Erprobung, hohe Verfügbarkeit, Kompatibilität und viele Implementierungen.

3.2.1.5. Allgemeine Nachteile
GPSS/H bietet (noch) keine graphische Modellbildung. Bei komplexeren Aufgaben muß eine relativ lange Einarbeitungszeit in Kauf genommen werden.

3.2.1.6. Schlußfolgerung
GPSS/H ist eine klassische, stabile "All-round"-Software; und stellt einen Quasistandard bei den klassischen Sprachen dar. GPSS/H ist allgemein einsetzbar, erfordert aber Einarbeitungszeit.

Beispiele für Modell- und Ergebnisbeschreibung:
3.2.2. PS SIMDIS

3.2.2.1. Allgemeines
PS SIMDIS ist eine diskrete Simulationssprache, die zur GPSS-Familie gehört.
PS SIMDIS kann als "bequemere" Umgebung von GPSS gesehen werden.

3.2.2.2. Produktdaten
3.2.2.2.1. Hersteller:
   Institut für Graphik und Simulation, TU Magdeburg
   Postfach 4120
   O- 3010 Magdeburg
   Tel.: +49-(0)391 5592

3.2.2.2.2. Distributor:
   (wie oben)

3.2.2.2.3. Plattformen, Systemerfordernisse:
   PS SIMDIS OS/ES: OS Version für PS-Systeme
   SIM-PC: allgemeine PC Version

3.2.2.2.4. Preis:
   auf Anfrage

3.2.2.2.5. Verfügbarkeit:
   PS SIMDIS: Version für OS/ES, SIM-PC: Version für PC

3.2.2.3. Evaluierungsresultate
3.2.2.3.1. User Interface
   Das Programm wird mit Hilfe eines Texteditors eingegeben.

3.2.2.3.2. Dokumentation
   Manual

3.2.2.3.3. Grundprinzipien
   PS SIMDIS ist eine blockorientierte Simulationssprache. Die Modellbildung erfolgt GPSS-
   ähnlich: Es gibt statische (storages, facilities,chains,...) und dynamische (transactions)
   Elemente. Transactions können im Laufe der Simulation erzeugt und vernichtet werden.

3.2.2.3.4. Granularität
   Wie GPSS/H ist PS SIMDIS fein strukturiert, entsprechend der GPSSH-Blockstruktur.

3.2.2.3.5. Grenzen
   Durch die Hardware beschränkt, auf PC durch DOS begrenzt.

3.2.2.3.6. Modellbeschreibung
   wie bei GPSS/H

3.2.2.3.7. Experimentbeschreibung
   wie bei GPSS/H
3.2.2.3.8. Output-Analysen
Daten können in Form von Tabellen, Graphiken (Histogrammen, Balkendiagrammen, Graphen,...) ausgegeben werden.

3.2.2.3.9. Import/Export von Datenformaten
Daten können mit ASCII-Files im-, und exportiert werden.

3.2.2.3.10. Animation
Nur SIM-PC, die PC-Version von PS SIMDIS verfügt über eine Animationsmöglichkeit auf einfacher direkter Basis.

3.2.2.4. Allgemeine Vorteile
Da PS SIMDIS im wesentlichen eine Erweiterung von GPSS/H darstellt, gelten auch diesen Vorteile.

3.2.2.5. Allgemeine Nachteile
wie bei GPSS/H.

3.2.2.6. Schlußfolgerung
PS SIMDIS ist eine Hochschulweiterentwicklung von GPSS, dementsprechend ist es für den Lehreinsatz geeignet.

Beispiele für Modell- und Ergebnisbeschreibung:

![Diagramm von interessanten Modell- und Ergebnisbeispielen für PS SIMDIS, einschließlich Visualisierung der Abläufe und der Ausgangsdaten.]
3.2.4. SIMAN

3.2.4.1. Allgemeines
Das 1983 eingeführte SIMAN ist ein "general purpose SIMulation ANalysis"-Programm zum Modellieren von diskreten und kontinuierlichen Systemen, wobei als Vorbild für die diskret/kontinuierlich kombinierten Elemente die Simulationssprache SLAM diente.

3.2.4.2. Produktdaten
3.2.4.2.1. Hersteller:
Systems Modeling Corporation
504 Beaver Street
Sewickley, Pennsylvania 15143
Tel: (412) 741-3727 Fax: (412) 741-5635

3.2.4.2.2. Distributor:
The CIMulation Centre Limited
Avon House, P.O. Box 46
Chippenham, Wiltshire
England, SN15 1JH
Tel: (0249) 650316 Fax: (0249) 443413

3.2.4.2.3. Plattformen, Systemerfordernisse:
Mainframes und PC’s unter DOS oder OS/2

3.2.4.2.4. Preis:
je nach Version; Grundpreis für PC etwa DM 5000.-

3.2.4.2.5. Verfügbarkeit:
neueste Version: SIMAN/CINEMA IV Version 4.0

3.2.4.3. Evaluierungsresultate
3.2.4.3.1. User Interface
Das Programm wird mit einem Texteditor eingegeben. Ein interaktives, graphisches Interface, BLOCKS genannt, wird als separates Unterstützungssystem angeboten.

3.2.4.3.2. Dokumentation
C. Dennis Pegden: "Introduction to SIMAN", Systems Modeling Corporation

3.2.4.3.3. Grundprinzipien
Der diskrete Teil von SIMAN IV arbeitet primär prozeßorientiert. (Es kann aber auch ein ereignisorientierter Ansatz verwendet werden.) Die 5 Prozessoren model (zum Erstellen des Blockdiagramms), experiment (beschreibt Experimente), link (verbindet model und experiment) run und output arbeiten unabhängig voneinander. In der neuesten Version sind sie in einer Shell bequem aufrufbar.
SIMAN IV basiert auf der Programmiersprache FORTRAN.
3.2.3.5. Grenzen
Die Modellgröße hängt nur vom verwendeten Betriebssystem ab.

3.2.3.6. Modellbeschreibung
Das Modell wird in Form von Blöcken erstellt, die z.B. das Netzwerk repräsentieren (textuell oder graphisch).

3.2.3.7. Experimentbeschreibung
Die Experimente werden textuell, bzw. über den graphischen Editor eingegeben und verändert.

3.2.3.8. Output-Analysen
Das Darstellen der Simulationsergebnisse unterstützt SLAM II mit statistischen Tabellen und Postprocessing.

3.2.3.9. Import/Export von Datenformaten
Mit ASCII-Files können Daten ein-, und ausgegeben werden.

3.2.3.10. Animation
SLAM II stellt auch ein Animationssystem (TESS genannt) zur Verfügung, das unter MS-Windows und OS/2 im Modell-Layout integriert werden kann.

3.2.3.4. Allgemeine Vorteile
SLAM II ist eine weitverbreitete, teilweise sehr beliebte Simulationssprache.

3.2.3.5. Allgemeine Nachteile
Die Programmierung gestaltet sich bei textueller Ebenen etwas mühsam.

3.2.3.6. Schlußfolgerung
SLAM II ist eine stabile, allgemein verwendbare Simulationssprache, die allerdings eine lange Einarbeitungszeit erfordert.

Beispiel für eine Modellbeschreibung:

```
; CONTROL FILE (VARIABLES EQUIVALENCE AND INITIALIZATIONS)
; -----------------------------------------------
GEN, SINGROUP, EUROSIM, 16/05/1991, 1, Y, Y, Y, Y, Y, Y, 1, 1322;
; TIMES IN SECONDS, LENGTHS IN CM
; LIMITS, 32, 12, 100;
; DYNAMIC ENTITIES (PALLETS) ATTRIBUTES
EQUIVALENCE/ATTRIB(3), OP2:  FLAG FOR OPER. 2 (1=ALREADY DONE, ELSE 0)
EQUIVALENCE/ATTRIB(3), OP3:  FLAG FOR OPER. 3 (1=ALREADY DONE, ELSE 0)
EQUIVALENCE/ATTRIB(4), OP4:  FLAG FOR OPER. 4 (1=ALREADY DONE, ELSE 0)
EQUIVALENCE/ATTRIB(5), OP5:  FLAG FOR OPER. 5 (1=ALREADY DONE, ELSE 0)
EQUIVALENCE/ATTRIB(6), STATION: NUMBER OF CURRENT STATION (BASIC SUBMODEL)
EQUIVALENCE/ATTRIB(7), UP: TELLS IF IT SHOULD GO THROUGH THE MACHINE
EQUIVALENCE/ATTRIB(8), CURR: CURRENT RESOURCE (CONVEYOR LINE) USED
EQUIVALENCE/ATTRIB(9), FNRC: OLD RESOURCE (CONVEYOR LINE) USED
EQUIVALENCE/ATTRIB(10), MACHINE: MACHINE RESOURCE NUMBER (SET ONLY IF UP=1)
EQUIVALENCE/ATTRIB(11), BUFFER: CONVEYOR LINE AFTER MACHINE (SET ONLY IF UP=1)
EQUIVALENCE/ATTRIB(12), LAPS: NUMBER OF CIRCUIT LAPS CURRENTLY DONE
;
; WORK TIMES
ARRAY(1, 8)/15, 60, 60, 60, 20, 20, 20, 30;
; A-LINES LENGTHS (CONVEYORS BEFORE MACHINES)
ARRAY(2, 8)/120, 80, 80, 80, 80, 80, 80, 120;
; B-LINES LENGTHS (CONVEYORS THAT DOESN'T PASS THROUGH THE MACHINES)
; C-LINES LENGTHS (CONVEYORS AFTER MACHINES)
ARRAY(4, 8)/80, 80, 80, 80, 80, 80, 80;
; SPACES BETWEEN STATIONS
ARRAY(5, 8)/0, 0, 40, 40, 40, 0, 0, 40, 40;
```

113
3.2.4. SIMAN

3.2.4.1. Allgemeines
Das 1983 eingeführte SIMAN ist ein "general purpose SIMulation ANalysis"- Programm zum Modellieren von diskreten und kontinuierlichen Systemen, wobei als Vorbild für die diskret/ kontinuierlich kombinierten Elemente die Simulationssprache SLAM diente.

3.2.4.2. Produktinformationen

3.2.4.2.1. Hersteller:
Systems Modeling Corporation
504 Beaver Street
Sewickley, Pennsylvania 15143
Tel: (412) 741-3727 Fax: (412) 741-5635

3.2.4.2.2. Distributor:
The CIMulation Centre Limited
Avon House, P.O. Box 46
Chippenham, Wiltshire
England, SN15 1JH
Tel: (0249) 650316 Fax: (0249) 443413

3.2.4.2.3. Plattformen, Systemerfordernisse:
Mainframes und PC's unter DOS oder OS/2

3.2.4.2.4. Preis:
je nach Version; Grundpreis für PC etwa DM 5000

3.2.4.2.5. Verfügbarkeit:
nächste Version: SIMAN/CINEMA IV Version 4.0

3.2.4.3. Evaluierungsresultate

3.2.4.3.1. User Interface
Das Programm wird mit einem Texteditor eingegeben. Ein interaktives, graphisches Interface, BLOCKS genannt, wird als separates Unterstützungssystem angeboten.

3.2.4.3.2. Dokumentation
C. Dennis Pegden: "Introduction to SIMAN", Systems Modeling Corporation

3.2.4.3.3. Grundprinzipien
Der diskrete Teil von SIMAN IV arbeitet primär prozeßorientiert. (Es kann aber auch ein ereignisorientierter Ansatz verwendet werden.) Die 5 Prozessoren model (zum Erstellen des Blockdiagramms), experiment (beschreibt Experimente), link (verbindet model und experiment) run und output arbeiten unabhängig voneinander. In der neuesten Version sind sie in einer Shell bequem aufforderbar.
SIMAN IV basiert auf der Programmiersprache FORTRAN.
3.2.4.3.4. Granularität
Die Struktur des Modells hängt von der verwendeten Ebene ab, da SIMAN Macros zuläßt. ("stations")

3.2.4.3.5. Grenzen
Die maximale Problemgröße ist durch die verwendete Plattform gegeben.

3.2.4.3.6. Modellbeschreibung
SIMAN IV teilt Modellstruktur und experimentellen Rahmen in zwei getrennte Bereiche. Die statischen und dynamischen Eigenschaften des Modells werden im Prozeß model beschrieben.

3.2.4.3.7. Experimentbeschreibung
Um verschiedene Simulationsläufe durchzuführen, muß nur der experimentelle Rahmen (Prozeß experiment) geändert werden, die Kontroll-, und Flußlogik im Modellrahmen (Prozeß model) bleibt unverändert.

3.2.4.3.8. Output-Analysen
Mit dem SIMAN output-Prozessor, der die Daten des output-Files bearbeitet, können Graphen, Tabellen, Histogramme, aber auch Konfidenz-, und Korrelationsberechnungen erzeugt werden.

3.2.4.3.9. Import/Export von Datenformaten
Daten können mit ASCII-Files im-, und exportiert werden. Auch LOTUS-kompatible Files können eingebunden werden.

3.2.4.3.10. Animation
SIMAN bietet Animation mit dem CINEMA-System.

3.2.4.4. Allgemeine Vorteile
Daten können mit dem Outputprozessor optimal aufbereitet werden.

3.2.4.5. Allgemeine Nachteile
Für komplexere Entscheidungsmechanismen ist die prozeßorientierte Beschreibung nicht geeignet.

3.2.4.6. Schlußfolgerung
Ähnlich wie ihr Vorbild SLAM ist SIMAN eine stabile, weit verbreitete Software, die auf vielen Plattformen läuft und für verschiedene Anwendungsbereiche geeignet ist. Nach einer soliden Einarbeitung können vor allem prozeßorientierte Modelle einfach gehandhabt werden.

Beispiele für Modell- und Experimentbeschreibung:

BEGIN,1,1,YES,test,NO;

initial Settings
CREATE,8;
ASSIGN:X(1)=X(1)+1;
ASSIGN:A(1)=X(1);
ASSIGN:A(2)=1;
REQUEST QUEUE,17;
WAIT:A(1);
BRANCH:
ALWAYS,REQUEST:
ALWAYS,PULLKIT;

Begin-Statement

8 Entities erzeugen.
die 8 verschiedenen
Platine markieren.
Entity erscheine!
in einen Puffer geben.
nimm richtige Platine
wenn ASSEMBLER
signalisiert.
Dupliciere sie.
gib sie weiter.
3.2.5. DESMO

3.2.5.1. Allgemeines
Das Simulationspaket DESMO (Discrete Event Simulation in Modula-2) unterstützt primär den prozeßorientierten Ansatz, erlaubt aber auch andere Modellierungsstile: transaktionsorientiert (wie GPSS), aktivitätsorientiert (wie ECSL) und ereignisorientiert. Als Vorbild für DESMO diente das von G.M. Birtwistle in Simula implementierte Paket DEMOS.

3.2.5.2. Produktdaten

3.2.5.2.1. Hersteller:
FB Informatik, Universität Hamburg
(Prof. Dr.-Ing. Bernd Page)
Vogt-Kölln-Str. 30
W-2000 Hamburg 54, Germany
Tel: 040/54715-426 Fax: 040/54715-246

3.2.5.2.2. Distributor:
( wie oben )

3.2.5.2.3. Plattformen, Systemerfordernisse:
VAX, PC unter MS-DOS

3.2.5.2.4. Preis:
auf Anfrage

3.2.5.2.5. Verfügbarkeit:

3.2.5.3. Evaluierungsresultate

3.2.5.3.1. User Interface
Das Modellprogramm wird mit Erweiterungen der Programmiersprache Modula-2 implementiert (Texteditor).

3.2.5.3.2. Dokumentation

3.2.5.3.3. Grundprinzipien

3.2.5.3.4. Granularität
Je nach verwendetener Ebene ist DESMO feiner oder gröber strukturiert.
3.2.5.3.5. Grenzen
Die Problemgröße ist vom verwendeten Rechner abhängig, unter DOS ist die Grenze demnach durch das Betriebssystem gegeben.

3.2.5.3.6. Modellbeschreibung
Das Modell wird in Module gegliedert und Modula-2 ähnlich beschrieben.

3.2.5.3.7. Experimentbeschreibung
Auch die Experimente haben modularen Aufbau, in getrennter Beschreibung.

3.2.5.3.8. Output-Analysen

3.2.5.3.9. Import/Export von Datenformaten
Daten lassen sich mit einem Modul (TimeSeries) auf eine Datei ausgeben, ein anderes Modul (DESgraph) ermöglicht eine graphische Ausgabe während des Simulationslaufes.

3.2.5.3.10. Animation
DESMO bietet keine Animationsmöglichkeiten.

3.2.5.4. Allgemeine Vorteile
Mit DESMO kann man verschiedene Modellierungsstile ("Weltbilder" der Simulation) verwenden und somit diese verschiedenen Ansätze vergleichen.

3.2.5.5. Allgemeine Nachteile
Schwachpunkte von DESMO sind die fehlende graphische Benutzeroberfläche und Animation.

3.2.5.6. Schlußfolgerung
Das Softwarepaket DESMO bietet sich vor allem für Lehrzwecke an, da es unterschiedliche Modellansätze erlaubt und vergleicht.

Beispiel für eine Modellbeschreibung:

```plaintext
/* Flexible Assembly System */
(* see EUROSIM - Simulation News Europe, Number 1, Mar 1991, p. 28 *)
(* and EUROSIM - Simulation News Europe, Number 2, Jul 1991, p. 26 *)
(* Programmer : Dirk Reinhardt *)
(* last change : 23.01.91 *)

MODULE EuroSim;

(* procedures of DESMO *)
FROM ProcessSimulation IMPORT Entity, New, Schedule, SetPriority,
Attributes, Current, Hold, SetTime,
Time, Wm, Reset, SetSchedule,
SetProfile, ClassFile;
FROM System IMPORT Copy, CardToBinary, Time, Conceal;
IMPORT Rev, Cond, Count, Tally;
FROM Cond IMPORT Multinomial, Signal;
FROM ReportTo IMPORT WriteString, WriteCard, WriteReal, WriteLine;

(* procedures of Modula-2 *)
FROM IO IMPORT Add, Write, Write, AddCard, AddCard, WriteReal,
AddReal, AddReal, AddReal, AddReal, AddReal,
FROM Storage IMPORT Allocate, Deallocate;

CONST shift_time = 1.0 / 30.0; (* time to shift from B to B1 or back *)
time_unit = 1.0 / 45.0; (* transport time for 0.4m on conveyor *)

TYPE (* attributes of a conveyor between two systems *)
connection_structure = RECORD
  c_belt : Res.Object;
transport_time : SetTime;
END;

(* attributes of a conveyor *)
system_structure = RECORD
  shift_place : CondEntity;
  buffer_cap, buffer_fill : CARDINAL;
  station : Res.Object;
  typ operation_time : SetTime;
  rear_buffer : Res.Object;
  conveyor_cap, conveyor_fill : CARDINAL;
END;
```
3.2.6. Micro Saint

3.2.6.1. Allgemeines


3.2.6.2. Produktdaten

3.2.6.2.1. Hersteller:
Micro Analysis and Design
3300 Mitchell Lane, Suite 175, Boulder
CO 80301, USA
Tel: +1(303) 442-6947 Fax: +1(303) 442-8274

3.2.6.2.2. Distributor:
Rapid Data Limited
Crescent House, Crescent Road
Worthing, West Sussex, BN115RW
Tel: +44 903 202819 Fax: +44 903 820762

3.2.6.2.3. Plattformen, Systemerfordernisse:
PC's unter DOS und Windows, UNIX; auch Apple Macintosh

3.2.6.2.4. Preis:
PC-Grundversion etwa DM 6000.-; günstige Hochschulversion

3.2.6.2.5. Verfügbarkeit:
letzte Version: Micro Saint 3 für Windows

3.2.6.3. Evaluierungsresultate

3.2.6.3.1. User Interface
Das User Interface von Micro Saint arbeitet mit Menüs und graphischen Elementen, wie etwa Fenster und Dialogboxen. Auch für Entscheidungsprozesse wird Menüsteuerung angeboten.

3.2.6.3.2. Dokumentation

3.2.6.3.3. Grundprinzipien

3.2.6.3.4. Granularität
Die Struktur des Modells hängt von der verwendeten Ebene ab, da Teilmodelle gebildet werden können.

3.2.6.3.5. Grenzen
Grenzen werden in DOS durch die DOS-Grenzen gesetzt; unter Windows keine Grenzen.

3.2.6.3.6. Modellbeschreibung
Das Modell wird in verschiedene Aktivitäten ("tasks") gegliedert, welche leicht modifiziert
werden können und die wiederum, abhängig von Bedingungen, Entscheidungstabellen, ... , andere "tasks" auslösen.

3.2.6.3.7. Experimentbeschreibung
Die Durchführung der Experimente wird durch ein Menü gesteuert.

3.2.6.3.8. Output-Analysen
Um die Resultate einer Simulation effizient auswerten zu können, stellt Micro Saint ein spezielles Analysepaket zur Verfügung, welches die Darstellung von Systemzuständen über die Zeit sowohl in graphischer (Balkendiagramme, Graphen), als auch in tabellarischer Form erlaubt. Durch sogenannte snapshots können, abhängig von gewissen Bedingungen, fast alle vorstellbaren Analysen durchgeführt werden.

3.2.6.3.9. Import/Export von Datenformaten
Import/Export von Datenformaten ist über ASCII-Files möglich.

3.2.6.3.10. Animation

3.2.6.4. Allgemeine Vorteile
Micro Saint erlaubt Simulation in einfacher, anwendungsfreundlicher Form für beliebige Anwendungsbereiche.

3.2.6.5. Allgemeine Nachteile
Bei großen Systemen kann, vor allem wegen der Menütiefe, aufgeblasenen Menüaufbau, die Übersichtlichkeit verloren gehen. Micro Saint ist ein relativ geschlossenes System.

3.2.6.6. Schlußfolgerung
Micro Saint ermöglicht eine rasche Modellerstellung, bei kurzer Einarbeitungszeit, und auch bei keinerlei Programmierkenntnissen seitens des Modellerstellers. Micro Saint ist benutzerfreundlich und besticht vor allem durch die Möglichkeit, komplexe Entscheidungskriterien einfach zu formulieren.

Beispiel für Modell- und Experimentbeschreibung:
3.2.7. SIMUL_R

3.2.7.1. Allgemeines

SIMUL_R ist eine allgemeine Simulationssprache für diskrete und kontinuierliche Systeme; der diskrete Teil wird PROSIMUL_R genannt und u.a. in den Bereichen Fertigungs-, und Verkehrstechnik angewandt. Es wird graphisches und textuelles Modellieren angeboten.

3.2.7.2. Produktdaten

3.2.7.2.1. Hersteller:
SIMUTECH
Hadikgasse 150
A-1140 Wien
Tel: +43-(0)222 894 75 08 Fax: +43-(0)222 894 78 04

3.2.7.2.2. Distributor:
( wie oben )

3.2.7.2.3. Plattformen, Systemerfordernisse:
PROSIMUL-R ist für PC's (MS-Windows), Transputernetze, DEC-Stations (X-Windows, UNIX) verfügbar.

3.2.7.2.4. Preis:
auf Anfrage, je nach Modell; der Grundpreis für PC-Version beläuft sich auf etwa DM 8000.-

3.2.7.2.5. Verfügbarkeit:
aktuelle Version (1992): SIMUL_R 2.3.

3.2.7.3. Evaluierungsresultate

3.2.7.3.1. User Interface
Über ein Menü-, und Mausgesteuertes Interface können Modelle graphisch (SIMDRAW) oder textuell eingegeben werden.

3.2.7.3.2. Dokumentation

3.2.7.3.3. Grundprinzipien
SIMUL_R basiert auf der Computersprache C.

3.2.7.3.4. Granularität

3.2.7.3.5. Grenzen
Grenzen werden nur durch die benützte Plattform gesetzt. Unter DOS wird die Grenze durch DOS selbst festgelegt.
3.2.7.3.6. Modellbeschreibung
SIMUL_R beschreibt Modelle mit einer großen Palette von Makrobefehlen, so können auch komplexe Systeme übersichtlich und leicht kontrollierbar formuliert und verändert werden.

3.2.7.3.7. Experimentbeschreibung

3.2.7.3.8. Output-Analysen
Zur Analyse und Präsentation stellt PROSIMUL_R u.a. Optimierung von diskreten sowie kontinuierlichen Vorgängen, diverse statistische Methoden (z.B. mittlere Bearbeitungszeit, Auslastungszeiten,...) und Darstellung in Zeitkurven und Histogrammen zur Verfügung. Besonderes Augenmerk wurde auf die graphische Darstellung der Daten gelegt: Eine umfangreiche graphische Bibliothek bietet u.a. 3D-Plots, Niveaulinien, selbstwählbare Skalen...

3.2.7.3.9. Import/Export von Datenformaten
Auf alle Daten kann mit ASCII Files für In-, und Output zugegriffen werden, während der Simulation kann der Anwender mit dem Keyboard oder graphisch Daten einbinden.

3.2.7.3.10. Animation
Die Animation mit SIMDRAW sieht die Verwendung von benutzererstellten Objekten vor, diese können z.B. unter Windows gezeichnet oder von vorhandenen Bildern eingescannt werden.

3.2.7.4. Allgemeine Vorteile
Vorteilhaft ist die Möglichkeit zur kombinierten oder parallelen Verwendung von diskreten und kontinuierlichen Systemen, die zahlreichen Analyse- und graphischen Methoden und die hohe Modularität. Außerdem ist SIMUL_R ein sehr offenes System, beliebige C-Programme können eingebunden werden.

3.2.7.5. Allgemeine Nachteile
Ein Nachteil von SIMUL_R ist die relativ lange Einarbeitungszeit.

3.2.7.6. Schlußfolgerung
PROSIMUL_R und Umgebung (SIMDRAW) ist ein für alle Probleme geeignetes Softwarepaket, das sich an informatischen Grundsätzen orientiert. Es ist eines der wenigen Simulationswerkzeuge, das kontinuierliche und diskrete Prozesse gleichgut unterstützt.

Beispiele für Modell- und Ergebnisbeschreibung:
3.2.8.3.7. Experimentbeschreibung
Die Beschreibung der Experimente erfolgt Menü-gesteuert.

3.2.8.3.8. Output-Analysen
Standardmäßig bietet DOSIMIS3 die Präsentation von Simulationsergebnissen in Form von Tabellen und Graphiken. Reicht dies nicht aus, so können Entscheidungstabellen (über das graphische User Interface einzugeben) verwendet werden, um komplexe Zusammenhänge darzustellen.

3.2.8.3.9. Import/Export von Datenformaten
Daten können mittels ASCII-Files im- und exportiert werden.

3.2.8.3.10. Animation
Die Animation erfolgt im Modell-Layout, synchron zur Simulation.

3.2.8.4. Allgemeine Vorteile
Standardelemente eines Materialflussystems kann man mit DOSIMIS3 leicht modellieren. DOSIMIS3 ist benutzerfreundlich und erfordert nur eine kurze Einarbeitungszeit.

3.2.8.5. Allgemeine Nachteile
Für komplexere Modelle muß man auf die Programmierbare (PASCAL) zurückgreifen. Die Elemente dieses Simulators sind nur auf Materialflussysteme zugeschnitten.

3.2.8.6. Schlußfolgerung
DOSIMIS3 bietet sich für nicht zu komplexe Materialflussprobleme an, es erfordert nur eine kurze Einarbeitungszeit.

Beispiele für Modell- und Ergebnisbeschreibung:
3.2.8. DOSIMIS3

3.2.8.1. Allgemeines
Der elementorientierte Simulator DOSIMIS3 arbeitet mit einer chronologischen Ereignisliste und wurde speziell zum Simulieren von diskreten Materialflußsystemen (MFS) entwickelt.

3.2.8.2. Produktdaten
3.2.8.2.1. Hersteller:
Fraunhofer Institute für Materialfluß und Logistik
Emil Figgestr. 75
W-6400 Dortmund
Tel: +49-(0)231 7549 171 Fax: +49-(0)231 7549 211

3.2.8.2.2. Distributor:
( wie oben )

3.2.8.2.3. Plattformen, Systemerfordernisse:
PC

3.2.8.2.4. Preis:
auf Anfrage

3.2.8.2.5. Verfügbarkeit:
---------

3.2.8.3. Evaluierungsresultate
3.2.8.3.1. User Interface
DOSIMIS3 besitzt keine textuelle Simulationssprache, sondern nur ein Menü-orientiertes, graphisches User Interface. Um komplexere Probleme zu modellieren, muß auf das Programmier-Interface (PASCAL) zurückgegriffen werden.

3.2.8.3.2. Dokumentation
Manual

3.2.8.3.3. Grundprinzipien
DOSIMIS3 basiert auf der Programmiersprache Pascal.

3.2.8.3.4. Granularität
DOSIMIS3 ist feinstrukturiert, die Blöcke haben einfachen Aufbau.

3.2.8.3.5. Grenzen
Die Modellgröße wird durch DOS Hardware beschränkt.

3.2.8.3.6. Modellbeschreibung
Das Modell wird mit prozeßorientierten Blöcken dargestellt, die sich speziell an Materialflußsystemen orientieren.
3.2.9. TOMAS

3.2.9.1. Allgemeines
Das Simulationssystem TOMAS (Technology Oriented Modeling And Simulation) wurde Ende der 70er Jahre von der Fakultät für Informatik an der TU Dresden entwickelt und 1990 als TOMAS/16 von der DVZ Neubrandenburg für MS-DOS PCs implementiert und auf den Softwaremarkt gebracht. TOMAS wird vor allem für das Simulieren von Fabrikationsabläufen eingesetzt.

3.2.9.2. Produktdaten
3.2.9.2.1. Hersteller:
DVZ Neubrandenburg GmbH
Bereich Softwareentwicklung und Systemberatung
Woldegker Straße 12
0-2000 Neubrandenburg, Germany
Tel: +37-90-587 443 Fax: +37-90-587 302

3.2.9.2.2. Distributor:
( wie oben )

3.2.9.2.3. Plattformen, Systemerfordernisse:
PCs unter MS-DOS

3.2.9.2.4. Preis:
auf Anfrage

3.2.9.2.5. Verfügbarkeit:
letzte Version: TOMAS/16 (1990):

3.2.9.3. Evaluierungsresultate
3.2.9.3.1. User Interface
Das User Interface von TOMAS ist Menü-gesteuert und bietet online zahlreiche Hilfsfunktionen.

3.2.9.3.2. Dokumentation
Manual

3.2.9.3.3. Grundprinzipien

3.2.9.3.4. Granularität
Verschiedene Ebenen ergeben unterschiedlich fein strukturierte Blöcke.

3.2.9.3.5. Grenzen
Die maximale Problemgröße hängt vom Betriebssystem DOS ab.
3.2.9.3.6. Modellbeschreibung
Das Modell wird in prozeßorientierten Modulen beschrieben, die speziell auf Fabrikationsabläufe ausgelegt sind.

3.2.9.3.7. Experimentbeschreibung
Der Simulationsablauf wird über ein Menü gesteuert.

3.2.9.3.8. Output-Analysen
TOMAS unterstützt die Aufbereitung der Simulationsergebnisse mit statistischen Tabellen und Postprocessing.

3.2.9.3.9. Import/Export von Datenformaten
ASCII-Files können zum Im- und Exportieren von Daten verwendet werden.

3.2.9.3.10. Animation
Die Animation erfolgt im Modell-Layout.

3.2.9.4. Allgemeine Vorteile
TOMAS ist einfach zu bedienen und bietet eine gute Online-Hilfe.

3.2.9.5. Allgemeine Nachteile
TOMAS ist speziell auf Fabrikationsabläufe ausgerichtet.

3.2.9.6. Schlußfolgerung
TOMAS erlaubt anwenderfreundliches Modellieren von Fabrikationsabläufen, die Einarbeitungszeit ist relativ kurz.

Beispiel für eine Modellbeschreibung:
3.2.11. SIMPLE

3.2.11.1. Allgemeines
SIMPLE ist ein echt objektorientierter Simulator nach dem Bausteinprinzip und wurde speziell für Fabrikationsabläufe (Produktion, Materialfluß, Logistik) konzipiert.

3.2.11.2. Produktdaten
3.2.11.2.1. Hersteller:
AESOP GmbH
Königstraße 82
Postfach 100121
D-7000 Stuttgart 10
Tel: (0711) 163 59-0 Fax: (0711) 163 59-99

3.2.11.2.2. Distributor:
UNSELD & PARTNER
Lerchenfelderstraße 44/V
A-1080 Wien
Tel: +43-(0)222-4030371 Fax: +43-(0)222-4030372

3.2.11.2.3. Plattformen, Systemerfordernisse:

3.2.11.2.4. Preis.
Hochschulversion: ungefähr 16.000.-
andere Versionen: auf Anfrage

3.2.11.2.5. Verfügbarkeit:
Neben der Vollversion bietet AESOP auch eine Hochschulversion an.

3.2.11.3. Evaluierungsresultate
3.2.11.3.1. User Interface
SIMPLE stellt ein menügesteuertes graphisches User Interface zur Verfügung und besitzt auch einen eigenen Editor zum Arrangieren der Module und zum Erstellen von Entscheidungstabellen.

3.2.11.3.2. Dokumentation
Anwendungsbericht: Simulationsystem SIMPLE. Beschreibung SIMPLE, Manual

3.2.11.3.3. Grundprinzipien
3.2.10. CASSANDRA

3.2.10.1. Allgemeines
Der Simulator CASSANDRA (Cognizant Adaptive Simulation System for Application in Numerous Different Relevant Areas) 2.1 basiert intern auf einer objektorientierten Struktur, die Elemente von Petri-Netzen nützt.

3.2.10.2. Produktdaten
3.2.10.2.1. Hersteller:
KFKI Research Institute of the Hungarian Academy of Sciences
P.O.Box 49
H-1525 Budapest, Hungary
Tel: +36-1 1699499 Fax: +36-1 1553894

3.2.10.2.2. Distributor;
( wie oben )

3.2.10.2.3. Plattformen, Systemerfordernisse:
PC'S unter MS-Windows 3.0

3.2.10.2.4. Preis:
auf Anfrage

3.2.10.2.5. Verfügbarkeit:
momentane Version: CASSANDRA 2.1.

3.2.10.3. Evaluierungsresultate
3.2.10.3.1. User Interface
Mit dem System IGENJA kann CASSANDRA um ein graphisches, Menü-, und Mausgesteuertes User Interface erweitert werden, sonst besitzt CASSANDRA nur ein textuelles Interface.

3.2.10.3.2. Dokumentation
Manual

3.2.10.3.3. Grundprinzipien

3.2.10.3.4. Granularität
CASSANDRA ist auf Petri-Ebene fein, auf Modul-Ebene grob strukturiert.

3.2.10.3.5. Grenzen
Die Problemgröße hängt vom verwendeten Rechner ab.

3.2.10.3.6. Modellbeschreibung
Der Aufbau des Modells erfolgt mit Grundbausteinen aus Petri-Teilnetzen.
3.2.10.3.7. Experimentbeschreibung
CASSANDRA erhöht die Effizienz der Simulation, indem die Rekonstruktion von Modellstrukturen und die Kontrolle über die Simulationsexperimente über sogenannte demons automatisiert werden.

3.2.10.3.8. Output-Analysen
Das Auswerten der Simulationsergebnisse wird mit statistischen Tabellen und Postprocessing unterstützt.

3.2.10.3.9. Import/Export von Datenformaten
Mit ASCII-Files können Daten eingebunden oder ausgegeben werden.

3.2.10.3.10. Animation
Ein Animationssystem für CASSANDRA besteht mit dem User-Interface IGENJA.

3.2.10.4. Allgemeine Vorteile
CASSANDRA arbeitet objektorientiert und stellt eine interessante Petri-Netz-Entwicklung dar.

3.2.10.5. Allgemeine Nachteile
Die Entwicklung von CASSANDRA ist noch nicht abgeschlossen.

3.2.10.6. Schlußfolgerung
Der Simulator CASSANDRA ist für die Forschungsaufgaben gut geeignet, da er die Verwendung von Petri-Netzen erlaubt.

Beispiele für die Modellbeschreibung:

SIMPLE basiert auf der Programmiersprache C++.

3.2.11.3.4. Granularität
SIMPLE ist grobstrukturiert und auf die Fertigung ausgerichtet. Feinstrukturierte Module können aber selbsttätig erzeugt werden.

3.2.11.3.5. Grenzen
Die Problemgröße der Hochschulversion wird vom Hersteller, die der Vollversion nur durch die Hardware begrenzt.

3.2.11.3.6. Modellbeschreibung
Durch die Strukturierung des Programmes in Objekte wird eine starke Modularisierung erreicht. Durch das Klassen- und Vererbungskonzept kann man Objekte leicht verändern.

3.2.11.3.7. Experimentbeschreibung
Die Experimentbeschreibung erfolgt Menü-gesteuert.

3.2.11.3.8. Output-Analysen
Statistische Ausgaben können für alle Objekte zu jeder Zeit in Form von Tabellen oder über Plotter-Funktionen abgerufen werden.

3.2.11.3.9. Import/Export von Datenformaten
SIMPLE kann Daten von ASCII-Files einlesen und auch exportieren.

3.2.11.3.10. Animation
Die Animation entsteht beim Entwickeln des Modells, erfolgt also im Layout des Modelles und erfordert prinzipiell keinen zusätzlichen Arbeitsaufwand. Nach Bedarf kann die Animation beliebig "verschönert" werden.

3.2.11.4. Allgemeine Vorteile
SIMPLE kann man auch ohne umfangreiche Programmierkenntnisse bedienen, und es entspricht dem modernen objektorientierten Aufbau.

3.2.11.5. Allgemeine Nachteile
SIMPLE ist derzeit nur auf Produktionsprozesse ausgerichtet.

3.2.11.6. Schlußfolgerung
SIMPLE gehört zu den neuen (objektorientierten) Simulationswerkzeugen, und ist momentan neben FACTOR/AIM das modernste Software-Tool für Fertigungsprozesse.

Beispiele für Modell- und Experimentbeschreibung:

Fig.: basic sub-module

Fig.: throughput time
3.2.12. WITNESS

3.2.12.1. Allgemeines

Das Simulationssystem WITNESS ist objektorientiert und kann diskrete sowie kontinuierliche Vorgänge beschreiben.

3.2.12.2. Produktdaten

3.2.12.2.1. Hersteller:
Bremer Institut für Betriebstechnik und angewandte Arbeitswissenschaft BIBA
Postfach 330560
W-2700 Bremen 33
Tel: +49 421 22009-43  Fax: +49 421 22009-79

3.2.12.2.2. Distributor:
( wie oben )

3.2.12.2.3. Plattformen, Systemerfordernisse:
PC's (386) unter OS/2

3.2.12.2.4. Preis:
auf Anfrage

3.2.12.2.5. Verfügbarkeit:

--------

3.2.12.3. Evaluierungsresultate

3.2.12.3.1. User Interface
WITNESS arbeitet mit einem graphischen User Interface, das auf verschiedenen Ebenen Menüs anbietet.

3.2.12.3.2. Dokumentation
Manual

3.2.12.3.3. Grundprinzipien
Die Elemente von WITNESS werden am Bildschirm graphisch erzeugt, dann werden über Parametermasken die notwendigen Eigenschaften zugewiesen: Kapazitäten, Durchlaufzeiten, aber auch Material-, und Informationsfluß zwischen den Elementen und Kontrollanweisungen zum Ablauf des Modells.

3.2.12.3.4. Granularität
WITNESS ist grobstrukturiert, in "Objekte".

3.2.12.3.5. Grenzen
Theoretisch werden WITNESS keine Grenzen gesetzt, da es unter OS/2 läuft.

3.2.12.3.6. Modellbeschreibung
Das Modell wird in ereignisorientierten Modulen, aber auch mit kontinuierlichen Elementen graphisch beschrieben.

3.2.12.3.7. Experimentbeschreibung
Die Experimente werden über ein Menü oder textuell eingeben und verändert.
3.2.12.3.8. Output-Analysen
   WITNESS bietet einige Standardstatistiken, Graphiken und Tabellen, es können aber auch
   vom Anwender selbst erstellte Funktionen und Werte verwendet werden um Resultate zu
   präsentieren.

3.2.12.3.9. Import/Export von Datenformaten
   Mit ASCII-Files können Daten eingebunden oder ausgegeben werden.

3.2.12.3.10. Animation
   Beim Simulieren des Modells wird eine "online"-Animation erzeugt.

3.2.12.4. Allgemeine Vorteile
   Vorteilhaft ist die rasche Modellierung und die leichte Erlernbarkeit von WITNESS.

3.2.12.5. Allgemeine Nachteile
   Leider steht WITNESS nur unter OS/2 zur Verfügung.

3.2.12.6. Schlußfolgerung
   Mit WITNESS können allgemein diskrete (aber auch kontinuierliche) Systeme rasch und nach
   kurzer Einarbeitsungszeit modelliert werden.

Beispiel für eine Ergebnisbeschreibung:
3.2.13. MODSIM

3.2.13.1. Allgemeines
MODSIM ist eine der neuen objektorientierten Simulationssprachen.

3.2.13.2. Produktdaten
3.2.13.2.1. Hersteller:
   CACI Products Company
   3344 North Torrey Pines Court
   La Jolla, California 92037
   Tel: (619) 457-9681   Fax: (619) 457-1184

3.2.13.2.2. Distributor:
   CACI Products Division
   Coliseum Business Centre
   Watchmoor Park, Riverside Way
   Camberley, Surrey GU15 3YL, UK
   Tel: 0276 671 671   Fax: 0276 670 677

3.2.13.2.3. Plattformen, Systemerfordernisse:
   MODSIM ist für die meisten Plattformen erhältlich.

3.2.13.2.4. Preis:
   von der Version abhängig; Einstiegspreis für PC ungefähr DM 4000.-

3.2.13.2.5. Verfügbarkeit:
   neueste Version: MODSIM II (1991)

3.2.13.3. Evaluierungsresultate
3.2.13.3.1. User Interface
   MODSIM stellt ein graphisches User Interface zur Verfügung.

3.2.13.3.2. Dokumentation
   Manual

3.2.13.3.3. Grundprinzipien
   MODSIM arbeitet objektorientiert.
   Basis ist die Programmiersprache C++.

3.2.13.3.4. Granularität
   MODSIM ist grobstrukturiert (in Objekte).

3.2.13.3.5. Grenzen
   Die maximale Problemgröße hängt von der verwendeten Plattform ab.

3.2.13.3.6. Modellbeschreibung
   Das Modell wird in Modulen beschrieben, die größtenteils prozeßorientiert sind.

3.2.13.3.7. Experimentbeschreibung
   Die Beschreibung der Experimente erfolgt Menü-gesteuert.
3.2.13.3.8. Output-Analysen
MODSIM sammelt während dem Simulationsablauf eine Reihe von statistischen Informationen und ermöglicht Postprocessing.

3.2.13.3.9. Import/Export von Datenformaten
MODSIM kann Daten mit ASCII-Files in das Modell einbinden und daraus exportieren.

3.2.13.3.10. Animation
MODSIM bietet vielfältige Animationsmöglichkeiten im Modell-Layout.

3.2.13.4. Allgemeine Vorteile
Ein Vorteil von MODSIM ist der modulare Aufbau.

3.2.13.5. Allgemeine Nachteile
Für komplexe Aufgaben erfordert es eine längere Einarbeitungszeit.

3.2.13.6. Schlußfolgerung
MODSIM kann als sehr moderne, allgemein für diskrete Prozesse einsetzbare Software verwendet werden.

Beispiele für die Modellbeschreibung:

```plaintext
FROM Animate IMPORT DynImageObj;
FROM GrpMod IMPORT QueueObj;

TYPE
  PlatformObj = OBJECT(DynImageObj)
  OrbitPosition : REAL;
  OrbitVelocity : REAL;
  OrbitRadius : REAL;
  Messages : QueueObj;
  Quadrant : INTEGER;

  TELL METHOD SendTo(IN platform : PlatformObj);
  TELL METHOD Receive From(IN platform : PlatformObj);
  ASK METHOD ComputePosition;

END OBJECT.
```
3.2.14. Pro Model PC

3.2.14.1. Allgemeines

3.2.14.2. Produktdaten
3.2.14.2.1. Hersteller:
Production Modeling Corporation International
1875 South State, Suite 3400
Orem, Utah 84058
Tel: (801) 226-6036 Fax: (801) 226-6046
3.2.14.2.2. Distributor:
(we oben)
3.2.14.2.3. Plattformen, Systemerfordernisse:
IBM PC XT, AT, PS/2 oder kompatibel;
Enhanced Graphics (EGA) oder Video Graphics (VGA)
3.2.14.2.4. Preis:
Einstiegspreis etwa DM 6000.-
3.2.14.2.5. Verfügbare:

3.2.14.3. Evaluierungsresultate
3.2.14.3.1. User Interface
Die Modelle werden mit einem eigenen Texteditor, mit Online-Hilfe erstellt.
3.2.14.3.2. Dokumentation
Manual
3.2.14.3.3. Grundprinzipien
3.2.14.3.4. Granularität
Je nach Ebene ist Pro Model PC feiner oder größer strukturiert.
3.2.14.3.5. Grenzen
Die maximale Problemgröße des Starter Package ist vom Hersteller beschränkt, die der Vollversion hängt vom verwendeten Rechner ab.
3.2.14.3.6. Modellbeschreibung
   Das Modell wird mit ereignis- oder prozeßorientierten Blöcken beschrieben.

3.2.14.3.7. Experimentbeschreibung
   Die Experimentbeschreibung und -änderung wird durch ein Menü gesteuert.

3.2.14.3.8. Output-Analysen
   Pro Model PC erstellt automatische statistische Reporte in Form von Tabellen und Graphiken
   (wie etwa Histo-, und Balkendiagramme, Torten, ...). Der Anwender kann für spezielle Zwecke
   aber auch eigene Reporte und Graphiken definieren.

3.2.14.3.9. Import/Export von Datenformaten
   Pro Model PC speichert die Daten im ASCII-Format, so kann das Output-File in andere
   Softwarepakete, Lotus. Excel, Quattro, ... eingebunden werden.

3.2.14.3.10. Animation
   Die Animation erfolgt im Modell-Layout.

3.2.14.4. Allgemeine Vorteile
   Pro Model PC ist relativ rasch erlernbar.

3.2.14.5. Allgemeine Nachteile
   Bei größeren Modellen können Speicherprobleme auftreten. (DOS)

3.2.14.6. Schlußfolgerung
   Mit Pro Model PC kann man kleinere Modelle rasch modellieren.

Beispiele für Modell- und Ergebnisbeschreibung:
3.2.15. EXAM

3.2.15.1. Allgemeines
Das general purpose system EXAM ist bestrebt alle Bereiche des Simulationsprozesses zu unterstützen: die Modellbeschreibung, die Experimentbeschreibung und die Simulation selbst.

3.2.15.2. Produktdaten
3.2.15.2.1. Hersteller:
Institute of System Analysis
Russian Academy of Science
9, Prospect 60 let Oktjabria
117312 Moscow, Russia
Fax: +7-095-9382209

3.2.15.2.2. Distributor:
(wie oben)

3.2.15.2.3. Plattformen, Systemerfordernisse:
IBM-kompatible PC’s unter MS-Windows

3.2.15.2.4. Preis:
auf Anfrage

3.2.15.2.5. Verfügbarkeit:
MS-Windows Version

3.2.15.3. Evaluierungsresultate
3.2.15.3.1. User Interface
EXAM arbeitet mit einem Menü-orientierten User-Interface, sodaß Modelle ohne besondere Programmierkenntnisse des Anwenders erstellt werden können.

3.2.15.3.2. Dokumentation
Manual

3.2.15.3.3. Grundprinzipien
EXAM basiert auf der objektorientierten Programmiersprache Turbo-Pascal.
EXAM trennt die Bereiche Modell-, und Experimentbeschreibung in zwei voneinander abgegrenzte Rahmen (frames): in den Model Description Frame (MDF) und in den Experiment Description Frame (EDF). Der Modellbilder erstellt die zur Modellerstellung nötigen Module, die den wichtigsten Features eines Prozesses entsprechen, verleiht ihnen die notwendigen Parameter und verbindet sie der Modell-Logik entsprechend.

3.2.15.3.4. Granularität
Je nach verwendeter Ebene, es können z.B. Subsysteme erstellt werden, feiner oder gröber strukturiert.

3.2.15.3.5. Grenzen
Grenzen werden nur durch die verwendete Hardware gesetzt.
3.2.15.3.6. Modellbeschreibung
   Das Modell wird, von der Experimentbeschreibung strikt abgegrenzt, im *Model Description Frame* (MDF) graphisch beschrieben.

3.2.15.3.7. Experimentbeschreibung
   Auch die Experimentbeschreibung erfolgt in einem eigenen Rahmen, dem *Experiment Description Frame* (EDF), in graphischer Form.

3.2.15.3.8. Output-Analysen
   Simulationsergebnisse können in Form von Tabellen präsentiert werden. Darüber hinaus werden Optimierungsmöglichkeiten angeboten.

3.2.15.3.9. Import/Export von Datenformaten
   Daten können mit ASCII-Files importiert und exportiert werden.

3.2.15.3.10. Animation
   Die Animation erfolgt im Modell-Layout.

3.2.15.4. Allgemeine Vorteile
   Der Modellbildner kann Systeme ohne Programmierkenntnisse mit Hilfe der benutzerfreundlichen Oberfläche von EXAM erstellen.

3.2.15.5. Allgemeine Nachteile
   Nachteil von EXAM ist die geringe Verfügbarkeit.

3.2.15.6. Schlussfolgerung
   Modelle können mit EXAM rasch, ohne lange Einarbeitungszeit und große Programmierkenntnisse erstellt werden.

Beispiel für eine Modellbeschreibung:
3.2.16. POSES

3.2.16.1. Allgemeines
Das Simulationssystem POSES (Prädikat-Transitionsnetz-Orientiertes-Simulations und Entwurfs-System), das an der Technischen Universität Chemnitz entwickelt wurde, basiert auf Petri-Netzen (Prädikat-Transitionsnetzen).

3.2.16.2. Produktdaten
3.2.16.2.1. Hersteller:
   Gesellschaft für Prozeßautomation & Consulting mbH
   Senefelder Str. 38
   D-O-9022 Chemnitz
   Tel.: +49-(0)371 50593       Fax: +49-(0)371 505 94

3.2.16.2.2. Distributor:
   (wie oben)

3.2.16.2.3. Plattformen, Systemerfordernisse:
   PC’s

3.2.16.2.4. Preis:
   auf Anfrage

3.2.16.2.5. Verfügbarkeit:
   aktuelle Version: POSES V4.3

3.2.16.3. Evaluierungsresultate
3.2.16.3.1. User Interface
   POSES besitzt eine eigene Shell mit Editor, Compiler und Linker, in der die Modelle textuell beschrieben und verarbeitet werden.

3.2.16.3.2. Dokumentation
   Manual

3.2.16.3.3. Grundprinzipien

3.2.16.3.4. Granularität
   ------

3.2.16.3.5. Grenzen
   Die Grenzen werden durch DOS gesetzt.

3.2.16.3.6. Modellbeschreibung
   In der POSES-Syntax wird das Modell aus Petri-Netz-Elementen (Places, Transitions) aufgebaut, dabei sind komplexe Verarbeitungs- und Routing-Daten formulierbar.
3.2.16.3.7. Experimentbeschreibung
Nahezu alle Parameter (Kapazitäten, Prioritäten,..) können während der Experimentierphase
definiert oder verändert werden.

3.2.16.3.8. Output-Analysen
Ergebnisse können in Form von Tabellen zusammengefaßt werden.

3.2.16.3.9. Import/Export von Datenformaten
Daten können mit ASCII-Files im-, und exportiert werden.

3.2.16.3.10. Animation
POSES bietet keine Animation.

3.2.16.4. Allgemeine Vorteile
POSES ist in vielen Bereichen einsetzbar.

3.2.16.5. Allgemeine Nachteile
POSES erfordert gute Programmierkenntnisse und somit eine lange Einarbeitungszeit. Ein
weiterer Schwachpunkt ist die fehlende Animation.

3.2.16.6. Schlußfolgerung
POSES ist geeignet für Anwender, die mit der Petri-Netz-Terminologie vertraut sind, und für
viele Bereiche verwendbar.

Beispiel für eine Modellbeschreibung:

```plaintext
system Philosoph:

const MAX = 5;
type Nr = 0..(MAX-1);
BufferType = ram [MAX] of Nr;
var x : Nr;
buffer Meditating : BufferType
(<<D>>+<<G>>+<<2>>+<<3>>+<<4>>):
Chopsticks : BufferType (<<D>>+<<1>>+<<2>>+<<3>>+<<4>>):
Eating : BufferType;

net GetSticks(in Meditating [<<D>>],
Chopsticks [<<D>>+<<x mod MAX + 1>>]
out Eating [<<D>>]);

PutSticks(in Eating [<<D>>],
out Meditating [<<D>>],
Chopsticks [<<D>>+<<x mod MAX + 1>>]);

end Philosoph
```
3.2.17. FACTOR / AIM

3.2.17.1. Allgemeines
FACTOR/AIM stellt in gewisser Weise eine Weiterentwicklung von SLAM dar, und ist auf Fertigung und Scheduling spezialisiert. FACTOR/AIM faßt Simulation (AIM) und Scheduling und Capacity Management (FACTOR) zusammen.

3.2.17.2. Produktdaten

3.2.17.2.1. Hersteller:
Pritsker & Associates, Inc.
P.O. BOX 2413
West Lafayette
Indiana 47906

3.2.17.2.2. Distributor:
  (wie oben)

3.2.17.2.3. Plattformen, Systemerfordernisse:
  PC's unter OS/2

3.2.17.2.4. Preis:
  Grundpreis etwa DM 10.000.-, günstige Hochschulangebote

3.2.17.2.5. Verfügbarkeit:
  FACTOR/AIM unter OS/2

3.2.17.3. Evaluierungsresultate

3.2.17.3.1. User Interface
  FACTOR/AIM ist auf allen Ebenen Menü-Gesteuert (mit OS/2 - Techniken). Eingabe und Resultatausgabe kann mit integrierten Oberflächen - spezifischen Programmen weiterverarbeitete werden.

3.2.17.3.2. Dokumentation
  Manual

3.2.17.3.3. Grundprinzipien
  FACTOR stellt als integriertes System Modellbausteine für Scheduling und Capacity Management zur Verfügung. AIM ist ein Simulationstool unter FACTOR, das die Modellbausteine von FACTOR verwendet (generelle Fertigungsabläufe) und um eigene dynamische Komponenten zur Zeitreihenanalyse erweitert.

3.2.17.3.4. Granularität
  Mithilfe der Modellbausteine kann auf beliebig hoher bzw. niedriger Ebene modelliert werden.

3.2.17.3.5. Grenzen
  Theoretisch keine Modellgrenzen, da das System unter OS/2 läuft.

3.2.17.3.6. Modellbeschreibung
  Modellbausteine erlauben einen graphischen Modellaufbau. Bausteine können auch von Datenbanken kommen (Scheduling-Tabellen, etc.). Die Bausteine können mit beliebig komplexen Routing-Folgen versehen werden.
3.2.17.3.7. Experimentbeschreibung
Die Simulation erfolgt über Menüs, Reports können online erzeugt und verändert werden. AIM ist das Simulationstool in FACTOR, weitere Experimente können mit anderen FACTOR-Anwendungen ausgeführt werden: SDM (Schedule Development) und SMM (Schedule Management) erlauben Detailstudien, Planung und Optimierung.

3.2.17.3.8. Output-Analysen
Innerhalb des integrierten Systems sind statistische (dynamische) Ausgaben jeder Art möglich: online zur Simulation, Vergleich von Simulationsvarianten (Postprocessing), etc.

3.2.17.3.9. Import/Export von Datenformaten
Entsprechend der OS/2-Philosophie können alle Datenformate der Umgebung verwendet werden.

3.2.17.3.10. Animation
Die Animation erfolgt im Modell-Layout, das beliebig graphisch verändert werden kann.

3.2.17.4. Allgemeine Vorteile
FACTOR/AIM entspricht dem modernen integrierten Aufbau eines Simulationspakettes; es kann benutzerfreundlich und ohne Programmierkenntnisse verwendet werden.

3.2.17.5. Allgemeine Nachteile
FACTOR/AIM ist nur auf Fertigungsprozesse ausgerichtet und läuft momentan nur unter OS/2.

3.2.17.6. Schlußfolgerung

Beispiele für die Modellbeschreibung:
EUROSIM Comparisons

Publication of Solutions

July 1995

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNE 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 1</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 2</td>
<td></td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 3</td>
<td></td>
<td>4</td>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 4</td>
<td></td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 5</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>SNE 6</td>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>SNE 7</td>
<td></td>
<td>2</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>SNE 8</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNE 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>SNE 10</td>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>SNE 11</td>
<td></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>SNE 12</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SNE 13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SNE 14</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>26</td>
<td>21</td>
<td>13</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>15</td>
<td>7</td>
</tr>
</tbody>
</table>
Unseld + Partner: Business- and Marketing Consulting Simulation!

What we can offer our customers!

• 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
Our discrete-event simulation and animation software handle even the most complex models

New users of GPSS/H™— in industry, education and government – quickly discover how GPSS/H’s superior flexibility makes simulation model development easier. Compared to GPSS/H, other packages and languages often fall short. GPSS/H won’t let you down whether you’re building a large, complex model or just learning about simulation.

Our Proof Animation™ software can bring your simulations to life.

It’s the next generation in PC simulation animation – faster, smoother, and more capable than the competition.

Call us today for more information or a free Proof Animation demo disk.