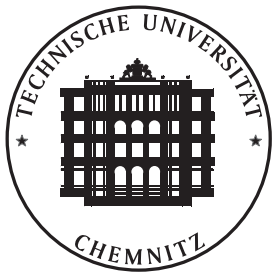

Technische Universität Chemnitz

Fakultät für Informatik

CSR-01-03

KASIMIR - object-oriented KAnban SIMulation Imaging Reality

P. Köchel · U. Nieländer · M. Sturm

März 2001

Chemnitzer Informatik-Berichte

KaSimIR – object-oriented Kanban Simulation Imaging Reality

Kanban systems have been invented by the Japanese Toyota Motor Corporation in the 1950s. Meanwhile, this implementation of the Just-in-Time-principle gained world-wide popularity, reputation and success. At Chemnitz University of Technology, the professorship "Modelling and Simulation" investigates such systems and analyses their perfomance according to costs and all the classical performance measures. The aim is to design Kanban systems in an optimal way. For that reason we apply simulation optimization, and that is why the KaSimIR–simulator has been developed. This software completely imitates the working process using the same structures that can be found in a real multi-stage, Kanban-controlled system.

This report starts with some introductory comments and a literature overview of this subject. Afterwards, a short review of simulation-based research of Kanban systems is given. Furthermore, some general requirements for a simulator of multi-stage Kanban-controlled systems will be defined. The following sections contain in detail the concepts of the KaSimIR–simulator, its main structures and the selection rules within. The basic classes of that software will be described as well as its organization and the statistical monitoring. A few words will remark on the usage of KaSimIR too. Finally, the last section finishes this report with a short resume and outlook to future work. Furthermore, a comprehensive list of references completes this report.

Prof. Dr. Peter Köchel Ulf Nieländer Martin Sturm

Contents

1 Introduction and Literature Overview

New circumstances require new arrangements. To meet the increasing requirements for e. g. efficiency and responsiveness, a set of manufacturing and operations management philosophies has been developed during the last two decades. One of them is Just-in-Time (JIT), first introduced at Toyota Motor Corporation (see SUGIMORI et al. 1977). Appropriate planning and control mechanisms are a necessary basis for the effective implementation of the JIT idea in an organization. In that context, Kanban – a Japanese word meaning "card" – is one of the extensively investigated principles to control the flow of material in a multi-stage production. For simplicity we start with the description of the Kanban control mechanism for a single-item, multi-stage, serial production system as shown in figure 1. It can be seen that the item flow goes from stage 1 towards the warehouse, which is the storage for finished products, whereas the flow of Kanbans runs towards the opposite direction. To cover the case of items being produced and moved through the system in lots with corresponding lot sizes, we replace items by containers with a given capacity from now on.

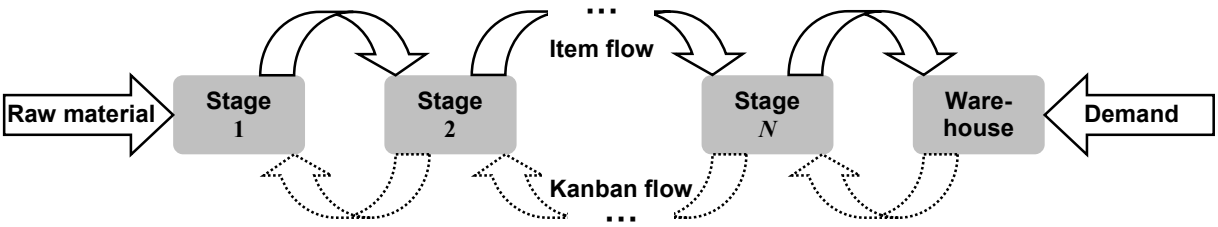

Figure 1: Scheme for a single-item, multi-stage, serial production system

To understand the mechanism behind the two flows, we consider the movement of containers and Kanbans between a stage and its neighbour stages (see figure 2). The representation of stage n ($1 < n < N$) in figure 2 includes the following parts:

1. A processing unit (workstation, cell, etc.) containing a server and a buffer;
2. An output hopper for containers finishing the service in the present stage n and waiting for withdrawal by the downstream stage n+1;
3. A Kanban box for requests of containers from the upstream stage.

Stage n owns a finite number k_n of Kanbans. When a container enters the stage, then exactly one Kanban of that stage will be attached. This Kanban will then be removed again at the moment the container leaves the stage. More exactly, if at least one Kanban is in the Kanban box of stage n, and if the output hopper of stage n–1 is not empty, then the following actions are executed:

(i) The pair (container, Kanban of stage n–1) in the output hopper of stage n–1 is separated. The Kanban is returned to its box in stage n–1, and the container is moved to stage n.
(ii) In stage n, one Kanban from its Kanban box is attached to that container.
(iii) Now the pair (container, Kanban of stage n) is ready for processing in stage n, i. e. it moves to the buffer. If the server is busy, the container with its Kanban has to wait in the buffer queue, otherwise processing starts immediately.
(iv) After finishing the processing in the server of stage n, the pair (container, Kanban of stage n) enters the output hopper. That is the queue of pairs with containers waiting for transhipment to the next stage n+1.

We point out that the container sizes before and after processing in a stage may be different. It is further remarked that in modern information systems a Kanban may be represented by a corresponding signal.

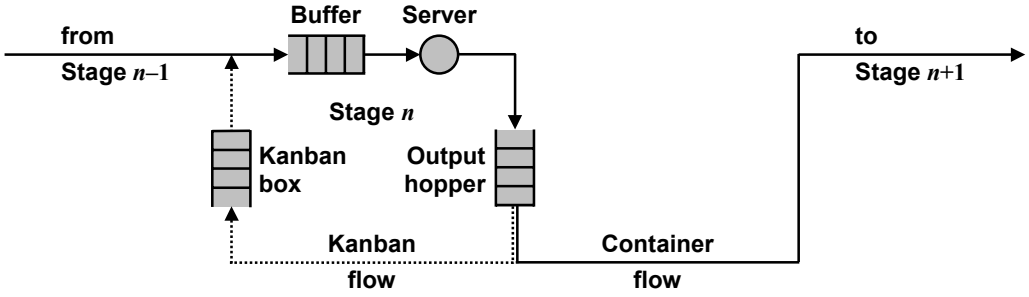

Figure 2: Flow of Kanbans and containers in a stage

The Kanban control mechanism just described has two main consequences. First, the total Work-in-Process (WIP) for a given stage is limited by the corresponding number of Kanbans (or, equivalently, containers). Second, the entire system is decomposed into a sequence of locally coordinated and controlled processing units. We remark that a one-card system has just been described. In so-called dual-card systems (cp. GROENEVELT 1993), there exist two types of Kanbans – production Kanbans and withdrawal/transportation Kanbans. Production Kanbans circulate within a stage whereas withdrawal Kanbans circulate between two successive stages. Since there do not exist any differences in principle between one-card and dual-card systems, we restrict ourselves to one-card systems.

The existing literature to Kanban systems is very voluminous, see e. g. ALTIOK (1997) or BUZACOTT and SHANTHIKUMAR (1993) for a comprehensive treatise. The papers differ from one another in regard to the approaches chosen and in regard to the goals investigated. Several analytical techniques as well as simulation have been used as approaches. However, simulation is the favourable one. Analytical models (Markovian processes, queueing networks) are applied almost exclusively for serial systems with unrealistic assumptions and deterministic or exponentially distributed times, see e. g. BUZACOTT (1989), DELEER-SNYDER et al. (1989), MITRA and MITRANI (1990, 1991), BADINELLI (1992), SPEARMAN (1992), TAYUR (1993). More complex systems are investigated by simulation, e. g. YAVUZ and SATIR (1995), SAVSAR (1996, 1997), AYTUG and DOGAN (1998), HUM and LEE (1998). Simulation has the advantage that it can handle the dynamics and the stochastic factors of a real system. Furthermore, a simulation model is capable of providing experimental statistics.

With respect to the investigation goals, Kanban research can be divided into two directions. The majority of papers treat the *analysis* problem. Papers dealing with both *performance analysis* problems and the derivation of *structural properties* belong to this group. For a given Kanban system they investigate important steady-state performance measures such as throughput, average WIP and average flow time of items. Closely connected to the first investigation goal, there is a second one – the *synthesis* problem. That is the problem to design a Kanban system that fulfills some predefined conditions. However, only a little is done in regard to the synthesis problem. The *Kanban allocation problem* (KAP) is most widely investigated, and it can be verbally formulated as "the problem to allocate a given total number of Kanbans among the stages of a multi-stage system such that a given criterion will be optimized". Again, analytical solutions for KAP are derived for sufficiently simple systems only, e. g. BITRAN and CHANG (1987), MITRA and MITRANI (1990, 1991), WANG and WANG (1991). A multi-criterion approach

for a single-item, multi-stage, serial system is considered by ANDIJANI (1998). The KAP is investigated for multi-product systems e. g. by BARD and GOLANY (1991) and ASKIN et al. (1993). However, for more complex systems only simulation remains as working tool, but optimal control policies or optimal design solutions for Kanban systems cannot be found using simulation exclusively. An appropriate combination of simulation and optimization is needed, i. e. *simulation optimization* should be applied. See FU (1994) for a review of simulation optimization.

Another aspect is that today's planning and control problems usually lead to complex optimization problems where the criterion function possesses no analytically tractable form and owns many local optima. In such cases, the estimation of the values of the criterion function by simulating the corresponding system and the search for an optimal solution by *genetic algorithms* or *evolution strategies* has been proved to be a powerful approach. At Chemnitz University of Technology that approach has been successfully applied for example to multi-location inventory systems with lateral transhipments (ARNOLD and KÖCHEL 1996, HADER 1998) and to fleet sizing and allocation problems (KÖCHEL et al. 2002). The first application to Kanban systems is reported by NIELÄNDER (1999) and KÖCHEL and NIELÄNDER (1999, 2000). Furthermore, KÖCHEL and NIELÄNDER (2002b) showed that their approach outperforms the heuristic methods of MITRA and MITRANI (1990, 1991) and of WANG and WANG (1991). HADER (2001) applies a hybrid approach for solving arbitrary parametric optimization problems to the KAP for a system with non-serial structure. He presents a software tool, that combines – among others – genetic algorithms with simulation. It should be noted that another simulation optimization approach, i. e. the combination of simulation and neural networks, is realized by SAVSAR and CHOUEIKI (2000). They solve the KAP for two-card systems by training a neural network using the simulation results of specially chosen decisions. After that, they get the estimation of the value of the criterion function for all admissible decisions from the trained neural network. The criterion function is a convex linear combination of the cost for WIP and demand delay, and the approach of SAVSAR and CHOUEIKI needs a new neural network with new training for another situation, whereas the approach of KÖCHEL and NIELÄNDER is applicable to arbitrary systems and general criterion functions.

In spite of the fact that the Kanban principle was developed as a simple control mechanism for assembly lines in the automotive industry, we are convinced that it is a generally applicable mechanism to control not only production systems with non-linear structure but also complex logistic or supply chain systems (KÖCHEL and NIELÄNDER 2002a). It should be clear that both for the design and for the performance analysis of such Kanban systems an appropriate simulator is needed. In the following section 2 of this report we define the requirements that such a simulator for multi-stage Kanban-controlled systems has to fulfill. Starting point is a short review of the simulation approach to such Kanban systems.

According to the aim of the present report, sections 3 to 8 contain the description of the simulator KaSimIR (Kanban Simulation Imaging Reality), developed within the professorship "Modelling and Simulation" at Chemnitz University of Technology. The concepts of KaSimIR, its main structures and the selection rules within are explained. The basic classes of that simulator will be described as well as its organization and the statistical monitoring. A few words will remark on the usage of KaSimIR too.

The last section 9 concludes our report with a summary and a short outlook to further research.

2 Kanban Systems and Simulation

The aim of the present section is to give a brief literature review of simulation-based research on Kanban systems. This will be the starting point to define some general requirements for a simulator of multi-stage Kanban-controlled systems. However, it is a hard task to classify the papers that apply simulation to Kanban systems. BERKLEY (1991) remarks that "several factors, including the wide variety of Kanban systems, have made the interpretation of research results difficult". A comprehensive review of the JIT-literature including Kanban research is given by GUNASEKARAN et al. (1993). Simulation is named as one of the most powerful investigation methods.

Despite the fact that simulation is the main approach in almost all Kanban-related papers, only a few papers exist containing an overview of simulation application to Kanban systems. From our literature review we can conclude that simulation of Kanban systems was mainly used for four purposes in the past:

1. To estimate the quality of approximation procedures for the evaluation of performance measures (see e. g. BERKLEY 1991, MITRA and MITRANI 1991);
2. To derive structural properties for performance measures from simulation results as done by YAVUZ and SATIR (1995);
3. To compare different scheduling rules or control policies within a given Kanban system, e. g. BERKLEY and KIRAN (1991), BERKLEY (1993a, 1993b, 1996), SAVSAR (1996, 1997), HUM and LEE (1998);
4. To compare the Kanban control mechanism with others, e. g. the traditional push system (SARKER and FITZSIMMONS 1989), a CONWIP production line (LEU 2000), base stock systems (DURI et al. 2000, KARAESMEN and DALLERY 2000), generalized Kanban systems (GUPTA et al. 1999) or hybrid systems (BEAMON and BERMUDO 2000).

However, the research done here at Chemnitz University of Technology provides a fifth purpose – simulation optimization (e. g. NIELÄNDER 1999, KÖCHEL and NIELÄNDER 1999, 2000, 2002b).

A first comprehensive review of papers related to simulation analysis of Kanban systems can be found in CHU and SHIH (1992). Their review is focused on how the major steps of a simulation study (cp. LAW and KELTON 1991) have been addressed. Among others the purpose of simulation, the used simulation models (model structures, Kanban types, model assumptions), distributions and random variables, used simulation languages (GPSS, Q-GERT, SIMAN, SLAM), questions of the experiment design (experimental factors, performance measures, steady state conditions, number and length of simulation runs, statistical output analysis) are considered. CHU and SHIH state that although "most of the models in use are relatively small in scale", both "many simulation-related statistical issues are ignored or neglected" and the overall behaviour of some experimental factors has not yet been well explained. Furthermore the interaction effects of more than two factors at a time have not yet been examined until now.

Later YAVUZ and SATIR (1995) as well as AYTUG and DOGAN (1998) overcame some of the mentioned deficiencies. In a very general manner YAVUZ and SATIR classified simulation studies of Kanban systems firstly as explorative analysis of pull systems, and secondly as comparative analysis of push and pull systems. Their brief literature review of simulation-oriented research of Kanban systems is very informative. Comparing the research results of different papers they found several contradictory statements. That is a natural consequence of the different assumptions made and the different production

scenarios investigated. Furthermore, YAVUZ and SATIR point out that most investigations are realized under several simplistic assumptions, as e. g.

- Infinite delivery rate for raw materials, i. e. supplier issues are ignored;
- Infinite demand rate, i. e. finished goods inventory issues are ignored;
- Flow line structure of the system;
- No transportation delays;
- Stochastics is restricted to demand and processing time distributions only.

Using SLAM II they investigate a rather general manufacturing system, the base model. In seven experiments, at most three out of nine model parameters are varied. The aim of the experiments is to investigate the main and interactive effects of parameters influencing seven considered performance measures.

To overcome the problem that none of the existing simulation languages provides appropriate constructs to build simulation models for Kanban systems, AYTUG and DOGAN (1998) introduce a simulation generator for very general dual-card systems. The simulation generator itself is written in C++ but generates SIMAN code. Each work center may have a certain number of input items and a certain number of output items. Furthermore, it is capable of performing several (manufacturing) processes. Parallel processing is possible with corresponding capacities. For the item supply from outside there exist specific reorder points, reorder quantities and delivery lead times. On the other end, demand orders are queued in different queues by item type with own sequencing rules. The list of selection rules (for source, process, Kanban, customer order and customer order source selection) includes e. g. RAN (random priority), CYC (cyclic priority), LNQ (largest number in queue), SNQ (smallest number in queue), HUT (highest utilization) and LUT (lowest utilization). However, the simulation generator of AYTUG and DOGAN has several limitations such as

- It requires a large amount of data.
- It still requires basic statistical skills.
- Set-up times and transportation times can only be deterministic.
- Formal modelling concepts do not exist for material handling devices.
- Several operating rules have limited options.

The paper of HUM and LEE (1998) compares the four scheduling rules first come first served (FCFS), shortest processing time (SPT), number of Kanbans (NKB) and ratio of Kanbans (RKB) with respect to several performance measures. The rules FCFS and SPT are well known and commonly used in practice. NKB and RKB give priority to the type of items that has the greatest number of waiting Kanbans resp. the greatest number of waiting Kanbans relatively to the total number of waiting Kanbans. Using GPSS/H a simple six-station line is simulated for five scenarios. Some suggestions are:

- Do not apply an arbitrary rule.
- The use of FCFS does not appear to be justified.
- The choice of rules should be made with reference to the nature of the processing times.
- A careful choice of rules should be made above all for systems with small utilization.

HUM and LEE give also a brief review on former research to the performance of scheduling rules.

To summarize our review of Kanban (and JIT) literature we state six main drawbacks with respect to Kanban modelling and simulation:

First, the investigated Kanban systems are rather simple and far from reality.

Second, different simulation experiments lead to different conclusions because of different simulation models and experimental assumptions.

Third, up to now the work force inclusion and the consideration of finite transportation capacities are open problems.

Fourth, the application of the Kanban control mechanism is restricted to manufacturing systems only. We are sure that Kanban-like control policies are suitable for other situations too, e. g. multi-location inventory systems, supply chain management or fleet sizing and allocation problems.

Fifth, all simulation models just allow to investigate the influence of system parameters on a single performance measure only. No model of any kind considers a generalized performance criterion, which for instance compares cost and gain of a Kanban system.

Sixth, the user support to design simulation models of Kanban systems, to plan simulation experiments and to analyse simulation results is not sufficient.

We hope that our simulator KaSimIR eliminates some of the above drawbacks. As mentioned in AYTUG and DOGAN, the incomplete description of Kanban systems is one of the major reasons for the still existing confusion surrounding Kanban research. Therefore KÖCHEL (1999) proposes that the following building elements should be committed for the definition of a Kanban system:

1. Number of products: Single-item or multi-item
2. Structure of the system: Serial, assembly-tree, arborescent, arbitrary
3. Order policy of raw materials: Common policies of inventory theory
4. Delivery of raw materials:
   a) Delivery rate: Infinite or finite (Poisson or arbitrary process)
   b) Lead time: Zero, constant, random
5. Stage description:
   a) Number of machines: Single or multi
   b) Buffer selection rules: FCFS, random, others
   c) Service times: Constant or random
   d) Reliability of machines: With and without failures
   e) Repair times: Constant or random
6. Container characteristics:
   a) Volume: One item or an arbitrary number of items
   b) When triggered: As soon as it gets empty or as a given number of items is processed
   c) Transportation time: Zero, constant, random, equal or different within a stage or between stages, equal or different for full and empty containers; transport as soon as the container is empty or as a given number of items is used
7. Warehouse: Zero, finite or infinite backlogging queue
8. Demand characteristics:
   a) Rate: Infinite or finite (Poisson or arbitrary process)
   b) Size: Single, multiple (constant or random)
9. Operating regime:
   a) Time: Continuous or discrete, finite or infinite horizon
   b) Breaks: With and without breaks

Taking into account on the one hand the drawbacks of today's existing simulators and on the other hand the requirements for a simulator of realistic Kanban systems mentioned above, we developed the first version of KaSimIR, which will be described in the next section.

3 The KaSimIR–Simulator and its Main Structures

The following sections give an overview of the structures and the possibilities of KaSimIR. Whereas the task of an optimizer is to find an optimal solution to a given problem, a simulator is just a tool to evaluate a special solution, considering all specified conditions. The quality of a simulator can be measured in different categories:

1. Universality: The simulator meets almost all usual requirements.
2. Expandability: It is easy to add new features.
3. Performance: The simulator prevents overhead and uses only that source code really necessary to gather all the required data to compile the statistics.

KaSimIR is programmed in C++ (using OOP). It completely imitates the working process using the same structures that can be found in a real multi-stage, Kanban-controlled system. This opens the possibility to add new types of work centers without complicated changes in the source code.

The essential part of the Kanban process is the transport of intermediate products between two work centers. The following scheme illustrates the implementation of this transport in KaSimIR:

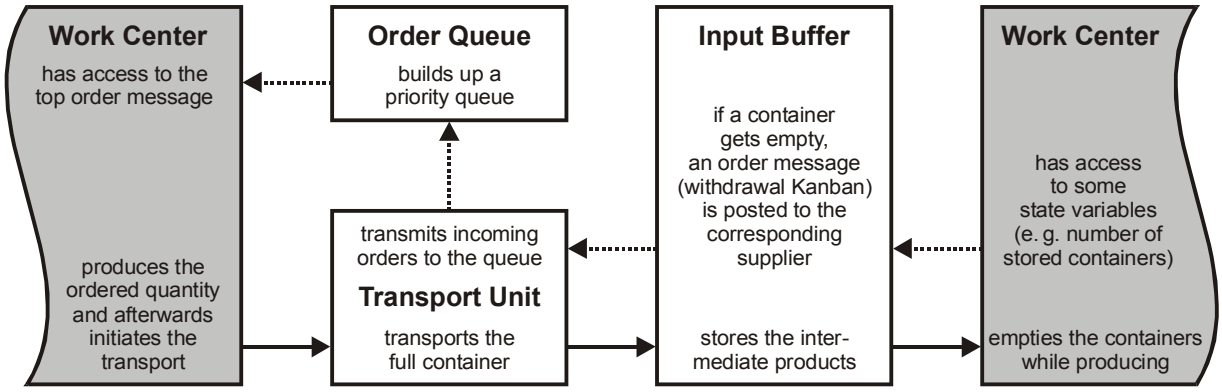

Figure 3: Transport process

This figure should be read from the right hand side to the left and then back. While the right work center is producing, the containers carrying intermediate products and stored in the corresponding input buffer are successively becoming empty. If one container is empty, a withdrawal Kanban message (specifying the type and the quantity of the intermediate product to be ordered as well as the address of the input buffer itself) is sent to the transport unit (in reality the empty container is transported with a Kanban on its top), which transmits it to the order queue of the supplying work center. Here all the Kanbans demanding this intermediate product are sorted according to some Withdrawal Kanban Selection Rule (WKSR). The left work center calls for the top order message and produces the intermediate product, so that the container successively gets refilled again. Finally the transport unit carries it back to the input buffer of the consuming work center.

In addition to these four object types, arrival stores (which represent the depots of raw material and externally supplied items) and customer queues (that simulate and manage incoming customer demand orders) exist. These objects are necessary to build up a model of a Kanban-controlled manufacturing system with KaSimIR (see the following simple example).

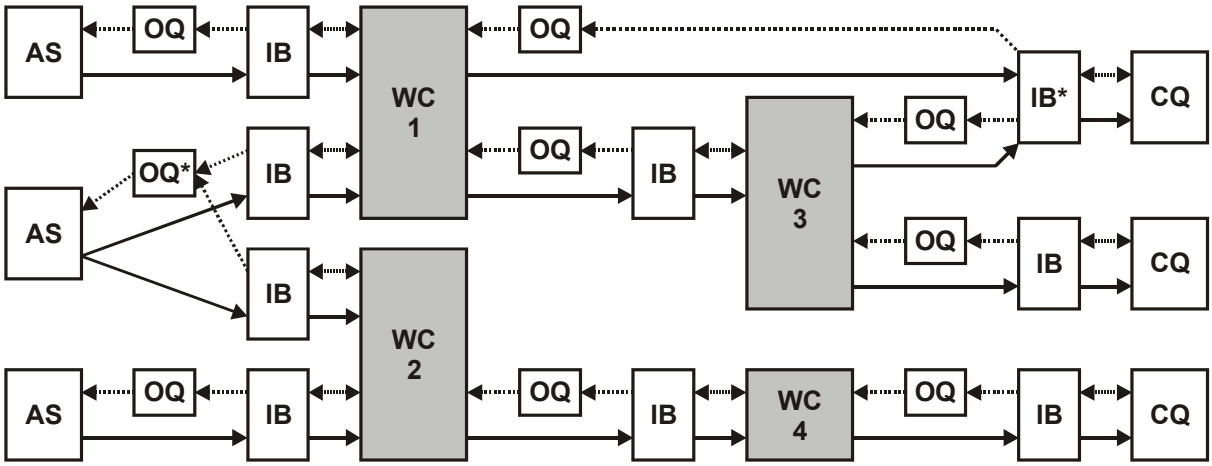

Figure 4: Exemplary manufacturing system

In this figure the transport units TU are left out for a better clarity. On the left hand side there are the arrival stores AS whereas on the right hand side there are the customer queues CQ. In the space between them, there are several types of work centers WC with corresponding input buffers IB and order queues OQ in different varieties. The work centers shown in figure 4 can be classified into four types:

1. Multiple input multiple output (MIMO), see WC1
2. Multiple input single output (MISO), see WC2
3. Single input multiple output (SIMO), see WC3
4. Single input single output (SISO), see WC4

Obviously, the SISO type is the simplest one. It starts producing, when there are orders in its order queue and enough raw material in its input buffer too. Other types of work centers are more complex, because some selection rules are required. These decision rules are used if there is more than one possibility to continue processing; they are explained in the next section.

4 The Selection Rules within KaSimIR

Consider for example the SIMO work center WC3. It is able to produce two different kinds of items, but certainly it can only produce one item anytime. If there is enough raw material stored in the input buffer, and if there are orders for both item types, then the work center could produce both types theoretically. Therefore it needs to decide which one to produce actually.

A Kanban-controlled manufacturing system requires several types of selection rules:

- WKSR (Withdrawal Kanban Selection Rule applied to order queues): This rule defines the order of the incoming withdrawal Kanbans to be sorted into a priority queue (necessary for example in the order queue OQ* of figure 4).
- PSR (Process Selection Rule applied to work centers): This is an algorithm to decide which possible process is chosen (by WC3 just described above).
- SSR (Source Selection Rule applied to input buffers): If an input buffer has more than one supplier, it has to decide which of them shall get the withdrawal Kanban (for example IB* in figure 4 needs such a selection rule).
- Possible extensions: OQSR and IBSR (Order Queue Selection Rule resp. Input Buffer Selection Rule; both applied to work centers): These rules are used only if there are more than one order queue resp. more than one input buffer containing the same kind

of items (it is possible to simulate these kinds of selection rules using the PSR; for example if there are two similar processes defined, one is using the first input buffer and the other one is using the second).
- COSR (Customer Order Sequencing Rule applied to customer queues): This rule specifies in which sequence the incoming customer demand orders will be served.
- CQSSR (Customer Queue Source Selection Rule): This rule is comparable to SSR or IBSR.

Obviously, only transport units do not need any selection rule. However – this is an important aspect – these rules are not necessary for the other object types, if there is only one single way of proceeding. Thus it makes sense to define different objects for each type of work centers, input buffers, order queues, etc. If a selection rule is necessary, one can build in a corresponding object, whereas if not, one can take a more simple object to save computing time.

According to the above enumeration, KaSimIR supports the following types of decision rules:

- POR (Preferred Order; applicable to WKSR and PSR): A predefined order exists specifying which Kanban / process to take (if there is no Kanban of the preferred client or if the preferred process cannot be started, because it does not have enough raw material, then the next possible Kanban will be set on top resp. the next process will be taken).
- CYC (Cyclic Priority; applicable to WKSR, PSR and SSR): The different possibilities alternate in a predefined sequence.
- RAN and PROB (Random Priority resp. Probabilistic Priority; applicable to all rules): Uniform random and weighted random choice between the possible candidates.
- LNQ and SNQ (Largest Number in Queue resp. Smallest Number in Queue; applicable to WKSR and PSR): To select the Kanban / process that is most or least demanded.
- SOQF and LOQF (Smallest Order Quantity First resp. Largest Order Quantity First; applicable to WKSR and COSR): These sequencing rules sort incoming orders and prefer orders with the smallest or largest quantity.
- FIFO and LIFO (First In First Out resp. Last In First Out; applicable to WKSR und COSR): These are the well known decision rules.

We remark that it is possible to add other rules too.

5 The Basic Classes of KaSimIR

Each object type, i. e. work center, input buffer, transport unit, order queue, arrival store and customer queue, is represented by its own basic class. These classes contain the main functions that will be described in this section. There are already some specialized sub-classes. However, if it is necessary to add new features it is very simple to do so, because if one class is expanded, all other objects will continue performing without any problems or changes.

5.1 *WorkCenter*–Class (WC)

We already know about the different types of work centers, some of them having more than one possible process. All these processes are given by the user and are represented in a table. Each process has its own row whereas the columns stand for the different input buffers. A number within this table specifies how many intermediate products from which input buffer are required for what process. There are additional columns for dates determining the time a process takes and the outcome of that process. We remark that it is

possible to define a process with no outcome (so it is possible to simulate waste, e. g. applying PROB to PSR).

The *state*–variable contains information about the active process (or that the work center is idle and doing nothing at the moment). The most important method – *evaluate* – is called if an input buffer has got new material, if an order queue has got a new Kanban or if a process has finished. If the work center is active and the process still goes on, *evaluate* just returns. Otherwise, if a process has finished, a message is sent to the input buffer the goods were produced for. Afterwards (or if the work center was not active when calling), the method *evaluate* checks all input buffers and all order queues to find out, which process / processes could be started. If there are two or more possibilities, it uses PSR to determine the one actually to start. All the necessary raw materials are taken from the input buffers whereas a Kanban is taken from the order queue. Finally, the *state*–variable is set to its new value and a message is sent to the work center itself to call *evaluate* again when the process has finished.

5.2 *ArrivalStore*–Class (AS) and *CustomerQueue*–Class (CQ)

These classes are children of the class WorkCenter. They stand at the beginning resp. at the end of a production. For the corresponding order queue and input buffer they appear just like work centers.

Arrival stores simulate an infinite store. That is why a container is sent back with goods immediately as soon as a withdrawal Kanban comes in. Customer queues manage all waiting customer demand orders and simulate their arrival. The queue length as well as the arrival rate and other parameters can be specified.

5.3 *InputBuffer*–Class (IB)

Input buffers contain some data structures to memorize how much material is stored. They have methods that return information about it, a method to refill the buffer and a method to take material out of the buffer. If this *empty*–method is called, it checks whether a container got empty or not. If so, a message is sent to the transport unit that belongs to the order queue as selected by SSR.

5.4 *OrderQueue*–Class (OQ)

Order queues are sorting the Kanbans according to WKSR. There are methods to enqueue a Kanban, to return data about the queue and to dequeue the top Kanban.

5.5 *TransportUnit*–Class (TU)

These little objects are very simple. They just connect one input buffer with one order queue. A transport unit contains data about the transport of Kanbans and containers (e. g. transportation time and cost) and has two methods. The first one is for the transport of Kanbans, the other one is for the transport of containers.

We remark that it is possible – in general – to give some parameters random variables. Furthermore, time periods and costs for production and transport can be specified by a probability distribution with free parameters to use.

6 Organization within KaSimIR

In the last section we defined the basic classes *WorkCenter*, *InputBuffer*, *OrderQueue*, *TransportUnit*, *CustomerQueue* and *ArrivalStore*. Every variant described in section 3 is a child of one of these six (respectively four, because *ArrivalStore* and *CustomerQueue* both are children of *WorkCenter*) basic classes. That opens the possibility to handle them with basic class pointers. In fact each type of basic class objects has its own container class. In the following figure, these are represented as CWC (container class for *WorkCenter*, *ArrivalStore* and *CustomerQueue*), COQ (container class for *OrderQueue*) and CIB (container class for *InputBuffer*). Again, this figure does not display the *TransportUnit*–objects and their container class CTU for a better clarity.

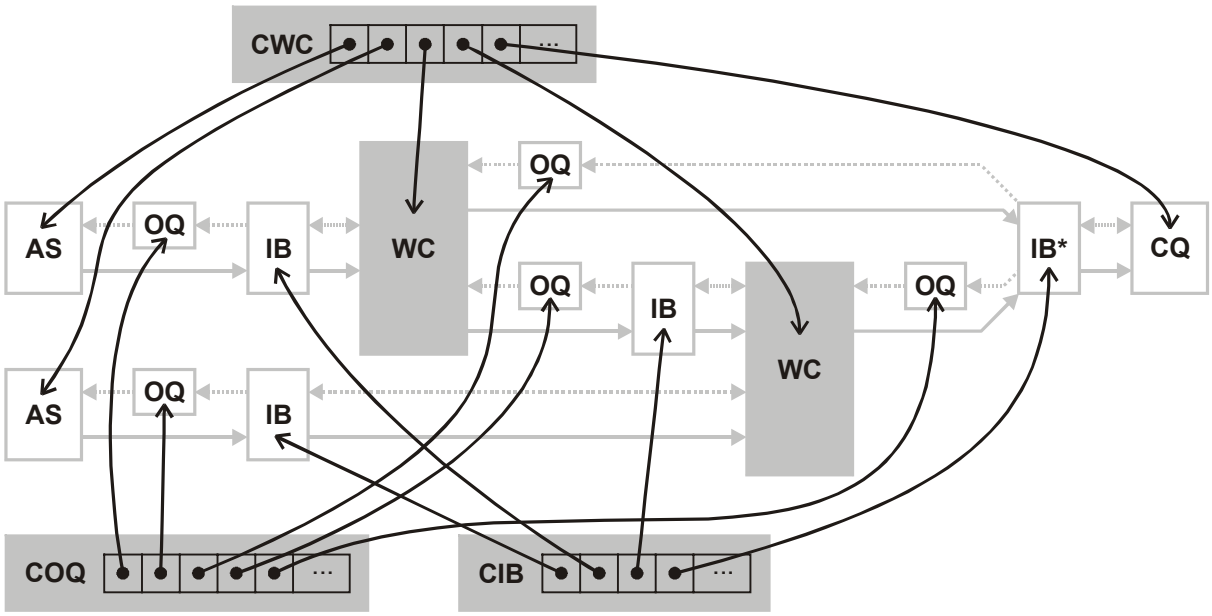

Figure 5: Another exemplary manufacturing system (with container classes)

At the beginning of a simulation run, the simulated multi-stage, Kanban-controlled manufacturing system is internally set up by a short source code to arrange the objects and to define the structure. The CWC–, COQ–, CIB– and CTU–containers are filled with pointers to the corresponding objects.

Because KaSimIR is a data-driven software, an input file is read out in the next step. In order to keep the user interface as simple as possible, this is just a text file to specify all the actual parameters for the system to simulate. These parameters can be the number of Kanbans and the size of the containers, the special type of selection rule, … (all the many other features mentioned in the last sections). When all this is put together and the internal model of the system to simulate has been built up, the data structures are initialized. For example, input buffers have to be empty whereas order queues have to be filled with Kanbans to replenish the buffers.

Then the simulation of the corresponding multi-stage, Kanban-controlled manufacturing system is ready to start. The simulation itself is an event-oriented process. There are about ten different types of events, all of them synchronized by an event heap. This is a special binary tree data structure which is a lot faster that a linear event list commonly used.

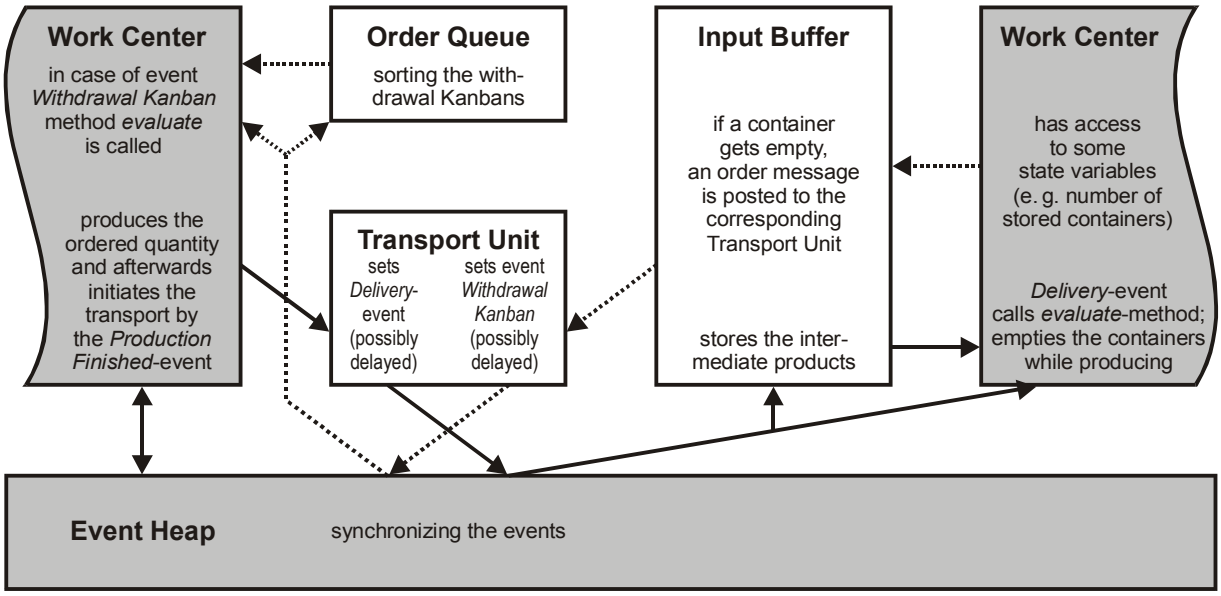

Figure 6: Function of the event heap

This figure can be read like figure 3. Every action that takes place immediately within the system, can be dealt with direct calls to the corresponding object (for example, if the right work center starts production, it empties the input buffer directly by calling the *empty*–method). All actions that take place after a certain period of time (for example, the arrival of a container in the input buffer after starting the transport) are represented by events. When the action starts (i. e. the transport begins) the object (here: transport unit) determines the point of time the action will be finished (here: arrival time of that container) using given parameters (here: probability distribution with parameters, random number generator). Then it sets an event (here: *Delivery*–event) containing the point of time when it takes place as well as some additional information (here: the transport destination, the container size, …). The event heap synchronizes all events and executes the next one on top. Thus, step by step the global *time*–variable increases and the simulation proceeds.

However, figure 6 above does not refer to the following important events:

- The *NewCustomer*–event is set by the customer queues. This is a "self-reproducing" event: While executing this event, such a new event is set for the future (i. e. the next customer demand arrival is scheduled). Therefore new customers are coming successively all over the time and the event heap does not get empty.
- The *SimulationStarts*–event: After the initialization of the objects, a very unnatural situation exists within the system, because all buffers are empty. That is why KaSimIR waits for a certain induction period of time (this transient phase depends on the length and the oscillation of the customer queues as well as the order queues) before the statistical monitoring starts. This event resets all statistical monitoring variables and sets the following event too.
- The *SimulationStops*–event: After a second period of time (this actual length of the simulation run is given by a parameter) the simulation process stops. Therefore, this event is the last one executed. Then an output file is written containing all statistical data (see the next section), and the memory is deallocated.

All events have the same internal data structure. If an event is executed and subsequently deleted from the event heap, the allocated memory is returned to an *event pool*. If a new event has to be created afterwards, the concerning object takes it from here and does not need to allocate new memory. This saves a lot of computing time and makes KaSimIR faster.

7 Statistical Monitoring within KaSimIR

Regarding simulation optimization, the main function of a simulation software is to collect data that helps the optimizing software to evaluate its solutions. In general, KaSimIR offers a large variety collecting data. One possible way is to create an own child of a special object – for example a work center – that writes its own output file containing the data required. But that is not the most elegant way, and it should only be used, if the following standard way is not sufficient.

If KaSimIR is used in combination with some optimizing software for the optimal design of Kanban systems, it is only necessary to determine the total costs arising within the system and within a certain period of time. To do this, each object calculates its own costs. All types of costs that are listed further below may consist of three components:

1. Constant / fixed cost factors (for each item passing through or every executed action, a certain amount of costs is calculated, for example starting a container transport);
2. Time-dependent / variable cost factors (here an amount of costs arises per item and per time period, for example transporting the container);
3. Random parameters (already described at the end of section 5).

According to the basic classes, the following costs are possible:

- *WorkCenter*–class: For each type of process, different costs can be defined. If a work center cannot produce, because there is not enough material in the input buffer or there are no Kanbans in the order queue, failure costs can be defined as well specifically for these two cases and the work center concerned.
- *InputBuffer*– and *OrderQueue*–class: Costs can be defined that are invoiced for every item stored in the buffer resp. for every Kanban hold in the queue.
- *ArrivalStore*– and *CustomerQueue*–class: Costs are handled in a similar way to the *InputBuffer*– and *OrderQueue*–class.
- *TransportUnit*–class: Different amounts of costs can be defined for transporting a withdrawal Kanban resp. transporting a full container.

Each object transmits its costs to a special monitoring module. Within the monitoring module, all costs coming in between the *SimulationStarts*– and the *SimulationStops*–event can be added up in a self-defined way. For a more comprehensive analysis, additional (non-cost) variables can be defined in the monitoring module. For example, considering a work center, such variables may count for all the costs, for the time that the work center could not produce due to insufficient raw material, for the total throughput of goods as well as WIP, etc. This gives extensive possibilities for analysing the simulated multi-stage, Kanban-controlled manufacturing system.

8 Usage of KaSimIR

According to the last sections, KaSimIR is a very general and complex simulation software, that can be adapted and parameterized in many different ways.

To make KaSimIR easily usable, a comfortable graphical user interface is required. This is a separate software that allows, for instance, to define the multi-stage, Kanban-controlled manufacturing system to be simulated in a graphical way and to enter its parameters instantly.

However, the structure behind the basic classes is easy to understand, and this opens up the possibility of further extensions as well as “creative misuse” in a positive manner to meet lots of additional requirements.

9 Resume and Outlook

The present report summarized some research on the simulation of Kanban systems done at the professorship "Modelling and Simulation" within the Computer Science Faculty of Chemnitz University of Technology. We considered the Kanban principle as generally applicable to control multi-stage systems. The primary aim of this report was not to give a comprehensive overview of simulation studies of such systems but to describe the design and the implementation of a simulator for very general multi-stage systems (e. g. Kanban systems, multi-echelon inventory systems). Some final goals for the present version of the KaSimIR–simulator are:

1. Empirical investigation of several performance measures of multi-stage systems;
2. Using it for simulation-based optimization of more complex systems;
3. Proving or rejecting findings of earlier studies in the scientific literature.

Although KaSimIR possesses many degrees of freedom to model multi-stage systems of that kind, there exist some fields for future work.

First, we can address the problem to develop optimal *flexible Kanban systems*. In such systems the number of Kanbans may vary depending on the demand intensity at any given moment. A simple control policy for the creation and the destruction of Kanbans can be defined by corresponding capture and release thresholds. Such thresholds can use several information sources, e. g. the (average) filling of buffer, output hopper and Kanban box or the utilization of the server in a given stage. Furthermore, it is possible to include information on future demand provided by suitable forecasting tools. These thresholds may be given locally for each stage or centrally for the whole system.

Second, up to now, no Kanban simulator includes the available *work force*, finite *transporting capacities* or *unreliable work* of the stages. *Non-linear cost functions* should be considered too.

Next, tools for the simulation of multi-stage systems controlled by *other principles* (e. g. CONWIP) or by *hybrid principles* should be developed and implemented.

Finally, with the increasing complexity of the systems to be investigated, the development of parallel and distributed simulation software will become an acute problem.

References

ALTIOK, T. (1997): *Performance Analysis of Manufacturing Systems*. Springer, New York.

ANDIJANI, A. A. (1998): A multi-criterion approach for Kanban allocations. *Omega: International Journal on Management Science*, **26**, 483–493.

ARNOLD, J. / KÖCHEL, P. (1996): Evolutionary optimization of a multi-location inventory model with lateral transhipments. *9th International Working Seminar on Production Economics*, Igls/Innsbruck, Pre-Prints, **2**, 401–412.

ASKIN, R. G. / MITWASI, M. G. / GOLDBERG, J. B. (1993): Determining the number of Kanbans in multi-item Just-in-Time systems. *IIE Transactions: Design and Manufacturing*, **25**, 89–98.

AYTUG, H. / DOGAN, C. A. (1998): A framework and a simulation generator for Kanban-controlled manufacturing systems. *Computers and Industrial Engineering*, **34**, 337–350.

BADINELLI, R. D. (1992): A model for continuous review pull policies in serial inventory systems. *Operations Research*, **40**, 142–156.

BARD, J. F. / GOLANY, B. (1991): Determining the number of Kanbans in a multiproduct, multistage production system. *International Journal of Production Research*, **29**, 881–895.

BEAMON, B. M. / BERMUDO, J. M. (2000): A hybrid push/pull control algorithm for multi-stage, multi-line production systems. *Production Planning and Control*, **11**, 349–356.

BERKLEY, B. J. (1991): Tandem queues and Kanban-controlled lines. *International Journal of Production Research*, **29**, 2057–2081.

BERKLEY, B. J. (1993a): Effect of buffer capacity and sequencing rules on single-card Kanban system performance. *International Journal of Production Research*, **31**, 2875–2893.

BERKLEY, B. J. (1993b): Simulation tests of FCFS and SPT sequencing in Kanban systems. *Decision Sciences*, **24**, 218–227.

BERKLEY, B. J. (1996): A simulation study of container size in two-card Kanban systems. *International Journal of Production Research*, **34**, 3417–3445.

BERKLEY, B. J. / KIRAN, A. S. (1991): A simulation study of sequencing rules in a Kanban-controlled flow shop. *Decision Sciences*, **22**, 559–582.

BITRAN, G. R. / CHANG, L. (1987): A mathematical programming approach to a deterministic Kanban system. *Management Science*, **33**, 427–441.

BUZACOTT, J. A. (1989): Queueing models of Kanban and MRP controlled production systems. *Engineering Costs and Production Economics*, **17**, 3–20.

BUZACOTT, J. A. / SHANTHIKUMAR, J. G. (1993): *Stochastic Models of Manufacturing Systems*. Prentice Hall, Englewood Cliffs.

CHU, C. H. / SHIH, W. L. (1992): Simulation studies in JIT production. *International Journal of Production Research*, **30**, 2573–2586.

DELEERSNYDER, J. L. / HODGSON, T. J. / MULLER(-MALEK), H. / O'GRADY, P. J. (1989): Kanban controlled pull systems: An analytic approach. *Management Science*, **35**, 1079–1091.

DURI, C. / FREIN, Y. / DI MASCOLO, M. (2000): Performance Evaluation and Design of Base Stock Systems. *European Journal of Operational Research*, **127**, 172–188.

FU, M. C. (1994): Optimization via simulation: A review. *Annals of Operations Research*, **42**, 199–247.

GROENEVELT, H. (1993): The Just-in-Time system. In Graves, S. C. / Zipkin, P. H. / Rinnooy Kan, A. H. G. (eds): *Handbook in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory*. Elsevier, Amsterdam, 629–670.

GUNASEKARAN, A. / GOYAL, S. K. / MARTIKAINEN, T. / YLI-OLLI, P. (1993): Modelling and analysis of Just-in-Time manufacturing systems. *International Journal of Production Economics*, **32**, 23–37.

GUPTA, S. M. / AL-TURKI, Y. A. Y. / PERRY, R. F. (1999): Flexible Kanban system. *International Journal of Operations and Production Management*, **19**, 1065–1093.

HADER, S. (1998): A multiagent simulation optimization system. In Bargiela, A. / Kerckhoffs, E. (eds): *Simulation Technology: Science and Art. 10th European Simulation Symposium*. Society for Computer Simulation International, Proceedings, 124–128.

HADER, S. (2001): *Ein hybrider Ansatz zur Optimierung technischer Systeme* [*A hybrid approach for the optimization of technical systems*]. Shaker, Aachen.

HUM, S. H. / LEE, C. K. (1998): JIT scheduling rules: A simulation evaluation. *Omega: International Journal on Management Science*, **26**, 381–395.

KARAESMEN, F. / DALLERY, Y. (2000): A performance comparison of pull type control mechanisms for multi-stage production control. *International Journal of Production Economics*, **68**, 59–71.

KÖCHEL, P. (1999): About the optimization of Kanban systems. *4th ISIR Research Summer School on Inventory Modelling*. School of Mathematical Sciences, Exeter, 127–136.

KÖCHEL, P. / KUNZE, S. / NIELÄNDER, U. (2002): Optimal control of a distributed service system with moving resources: Application to the fleet sizing and allocation problem. Forthcoming in *International Journal of Production Economics*.

KÖCHEL, P. / NIELÄNDER, U. (1999): Optimierung von Kanban-Systemen mittels Simulation und Genetischen Algorithmen [Optimization of Kanban systems by simulation and genetic algorithms]. In KÖCHEL, P. (ed): *KI-Methoden in der simulationsbasierten Optimierung: 13. Workshop der ASIM-Fachgruppe Simulation und Künstliche Intelligenz*. Chemnitzer Informatik-Berichte, **CSR-99-03**, 7–29.

KÖCHEL, P. / NIELÄNDER, U. (2000): Evolutionary optimization of Kanbans. *INFORMS& KORMS–Conference: Information and Knowledge Management in the 21st Century*, Seoul, Proceedings, 150–155.

KÖCHEL, P. / NIELÄNDER, U. (2002a): Defining optimal policies in multi-echelon inventory systems: The simulation optimization approach. *12th International Working Seminar on Production Economics*, Igls/Innsbruck, Pre-Prints, **1**, 165–173.

KÖCHEL, P. / NIELÄNDER, U. (2002b): Kanban optimization by simulation and evolution. Accepted at *Production Planning and Control*.

LAW, A. M. / KELTON, W. D. (1991): *Simulation Modeling and Analysis*. McGraw Hill, New York, 2nd ed.

LEU, B. Y. (2000): Generating a backlog list for a CONWIP production line: A simulation study. *Production Planning and Control*, **11**, 409–418.

MITRA, D. / MITRANI, I. (1990): Analysis of a Kanban discipline for cell coordination in production lines I. *Management Science*, **36**, 1548–1566.

MITRA, D. / MITRANI, I. (1991): Analysis of a Kanban discipline for cell coordination in production lines II: Stochastic demands. *Operations Research*, **35**, 807–823.

NIELÄNDER, U. (1999): Simulation optimization of Kanban systems using a non-standard genetic algorithm. *4th ISIR Research Summer School on Inventory Modelling*. School of Mathematical Sciences, Exeter, 137–146.

SARKER, B. R. / FITZSIMMONS, J. A. (1989): The performance of push and pull systems: A simulation and comparative study, *International Journal of Production Research*, **27**, 1715–1731.

SAVSAR, M. (1996): Effects of Kanban withdrawal policies and other factors on the performance of JIT systems: A simulation study. *International Journal of Production Research*, **34**, 2879–2899.

SAVSAR, M. (1997): Simulation analysis of maintenance policies in Just-in-Time production systems. *International Journal of Operations and Production Management*, **17**, 256–266.

SAVSAR, M. / CHOUEIKI, H. (2000): A neural network procedure for Kanban allocation in JIT production control systems. *International Journal of Production Research*, **38**, 3247–3265.

SPEARMAN, M. L. (1992): Customer service in pull production systems. *Operations Research*, **40**, 948–958.

SUGIMORI, Y. / KUSUNOKI, K. / CHO, F. / UCHIKAWA, S. (1977): Toyota production system and Kanban system – Materialization of Just-in-Time and Respect-for-Human system. *International Journal of Production Research*, **15**, 553–564.

TAYUR, S. (1993): Structural results and a heuristic for Kanban controlled serial lines. *Management Science*, **39**, 1347–1368.

WANG, H. / WANG, H. P. (1990): Determining the number of Kanbans: A step toward non-stock production. *International Journal of Production Research*, **28**, 2101–2115.

WANG, H. / WANG, H. P. (1991): Optimum number of Kanbans between two adjacent workstations in a JIT system. *International Journal of Production Economics*, **22**, 179–188.

YAVUZ, I. H. / SATIR, A. (1995): A Kanban-based simulation study of a mixed model Just-in-Time manufacturing line. *International Journal of Production Research*, **33**, 1027–1048.

Chemnitzer Informatik-Berichte

In der Reihe der Chemnitzer Informatik-Berichte sind folgende Berichte erschienen:

CSR-96-01 W.Benn, Y.Chen, I.Gringer, A Rule-based Strategy for Schema Integration in a Heterogeneous Information Environment, Januar 1996

CSR-96-02 W.Benn, I.Gringer, Datenbank-Anwendungen über das Internet, Februar 1996

CSR-96-03 W.Kalfa, Dynamische Adaption in Betriebssystemen – Das CHEOPS-Projekt, März 1996

CSR-96-04 Jahresbericht der Fakultät für Informatik 1995, Januar 1996

CSR-96-05 D.Monjau (Hrsg.), Custom Computing - GI/ITG Workshop, Juni 1996, Schloß Dagstuhl

CSR-96-06 W.Dilger, M.Schlosser, J.Zeidler, A.Ittner, Beiträge zum 9. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3

CSR-97-01 Jahresbericht der Fakultät für Informatik 1996, Januar 1997

CSR-97-02 D.Monjau (Hrsg.), Hardwarebeschreibungssprachen und Modellierungsparadigmen, 3. ITG/GI/GMM–Workshop, 26.–28. Februar 1997, Holzhau

CSR-97-03 Y.Chen, W.Benn, A Systematic Method for Query Evaluation in Federated Relational Databases, Juni 1997

CSR-97-04 A.Goerdt, The Giant Component Threshold for Random Regular Graphs with Edge Faults, Juni 1997

CSR-97-05 W.Rehm (Ed.), Cluster – Computing, Tagungsband zum 1. Workshop, 6./7. November 1997, Chemnitz

CSR-98-01 A.Goerdt (Ed.), Workshop über Komplexitätstheorie, Datenstrukturen und effiziente Algorithmen, Tagungsband zum 34. Workshop, 10. März 1998, Chemnitz

CSR-98-02 Ch. Giraud-Carrier, M. Hilario (Ed.), Upgrading Learning to the Meta-Level: Model Selection and Data Transformation, ECML'98 Workshop Notes, 24. April 1998, Chemnitz

CSR-98-03 D. Canamero, M. van Someren (Ed.), Learning in Humans and Machines, ECML'98 Workshop Notes, 24. April 1998, Chemnitz

CSR-98-04 R. Basili, M. T. Pazienza (Ed.), TANLPS Towards adaptive NLP-driven systems: linguistic information, learning methods and applications, ECML'98 Workshop Notes, 24. April 1998, Chemnitz

CSR-98-05 Y. Kodratoff (Ed.), Text Mining, ECML'98 Workshop Notes, 24. April 1998, Chemnitz

Chemnitzer Informatik-Berichte

CSR-98-06 G. Nakhaeizadeh, E. Steurer (Ed.), Application of Machine Learning and Data Mining in Finance, ECML'98 Workshop Notes,
24. April 1998, Chemnitz

CSR-98-07 C. Nédellec, C. Rouveirol, Demonstration and Poster Papers, ECML'98, 21.-24. April 1998, Chemnitz

CSR-99-01 D. Thie, J. Flohrer, Charakterisierung des Datenverkehrs von HTTP-Clienten, Januar 1999

CSR-99-02 Wolfgang Rehm, Theo Ungerer (Ed.), Cluster-Computing, Tagungsband zum 2. Workshop, 25./26. März 1999, Universität Karlsruhe

CSR-99-03 Peter Köchel (Hrsg.), KI-Methoden in der simulationsbasierten Optimierung, 13. Workshop der ASIM-Fachgruppe „Simulation und Künstliche Intelligenz", 12./13. April 1999, Chemnitz

CSR-99-04 Lutz Wohlrab, Frank Schubert, Francisco Ballesteros, Henning Schmidt, Ashish Singhai (Editors), Proceedings of the 2nd ECOOP Workshop on Object-Orientation and Operating Systems (ECOOP-OOOSWS'99),
June 14th, 1999, Lisbon, Portugal

CSR-99-05 Martin Matamala, Klaus Meer, On the computational structure of the connected components of a hard problem, Dezember 1999, Chemnitz

CSR-99-06 Klaus Meer, Burkhard J. Schmitt, Harold Schreiber, Dimensional Synthesis of Planar Stephenson-Mechanisms for Motion Generation by Circlepoint Search and Homotopy Methods, Dezember 1999, Chemnitz

CSR-00-01 J.A. Makowsky und K. Meer, Polynomials of bounded tree-width, Januar 2000

CSR-00-02 Andreas Goerdt, Efficient interpolation for the intuitionistic sequent calculus, Januar 2000, Chemnitz

CSR-01-01 Werner Dilger, Evelyne Keitel, Kultur und Stil in der Informatik?, Januar 2001, Chemnitz

CSR-01-02 Guido Brunnett, Thomas Schaedlich, Marek Vanco, Extending Laszlo's Algorithm to 3D, März 2001, Chemnitz

CSR-01-03 M.Köchel, U.Nieländer, M.Sturm, KASIMIR - object-oriented KAnban SIMulation Imaging Reality, März 2001, Chemnitz

Chemnitzer Informatik-Berichte

ISSN 0947-5125

Herausgeber: Fakultät für Informatik, TU Chemnitz
Straße der Nationen 62, D-09111 Chemnitz