Marco Morawec, BSc

A simulation-based approach to detect and prevent deadlock situations in flexible production systems

MASTER'S THESIS

to achieve the university degree of
Master of Science

Master's degree programme:
Production Science and Management

submitted to
Graz University of Technology

Supervisor: Ass.Prof. Dipl.-Ing. Dr.techn. Nikolaus Furian

Institute of Engineering and Business Informatics

Head of Institute: Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner

Graz, May 2020

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date, Signature

Abstract

In the last few decades, changes have taken place in the production sector. Increasing desire for personalization and uncertainty in demand of customers led to a perspective change and the use of Flexible Manufacturing Systems. These systems offer a wide range of flexibility that can cope with the demand but also increase complexity.

An often-occurring problem that has gained significant interest are deadlocks. A deadlock is defined as a production system or a section that is blocking itself, so that no parts can move anymore without intervention. Most of the time workers are used to re-establish the flow of products again by moving products out of the deadlocked section and returning them later. The unpredictability and the decreased efficiency of the system as well as the increased cost make deadlocks something to be avoided.

This master thesis presents a new flexible solution of handling deadlocks with the additional goal to keep the productivity decrease low. This is done with the use of a simulation and an encapsulated prediction simulation. Both are built on a Hierarchical Control Conceptual Model and are the basis for deadlock control methods. Through this subdivision of deadlock prediction and deadlock control method a high flexibility is achieved. For these methods already existing strategies can be used or completely new ones that are highly adjusted to the production system on hand can be developed. Also, combinations of methods are possible.

This is tested in a case study that compares a simple deadlock strategy with three strategies that are built on top of the presented solution. The results show that the new method delivers better deadlock handling results while maintaining efficiency to a large extend. They also indicate that even better results can be achieved if more complex and better to the production system adjusted methods are developed.

Kurzfassung

In den letzten Jahrzehnten gab es Veränderung im Produktionssektor. Der zunehmende Wunsch nach Personalisierung und Unsicherheiten in der Nachfrage von Kunden führte zu einem Perspektivenwechsel und dem Einsatz von flexiblen Fertigungssystemen. Diese Systeme bieten ein breites Spektrum an Flexibilität die Anforderungen gerecht werden, aber auch die Komplexität erhöht.

Ein oft auftretendes Problem, dem besondere Aufmerksamkeit gegeben wurde sind Deadlocks. Ein Deadlock ist definiert als ein Fertigungssystem oder ein Abschnitt davon, der sich selbst blockiert, sodass sich keine Teile mehr ohne Eingriff bewegen können. Meistens werden Arbeiter eingesetzt um den Produktfluss durch herausnehmen von Produkten und späterem wieder einführen wiederherzustellen. Die Unvorhersehbarkeit und der Verlust der Effizienz des Systems, sowie die erhöhten Kosten machen Deadlocks zu etwas, das vermieden werden sollte.

Diese Masterarbeit präsentiert eine neue flexible Lösung für den Umgang mit Deadlocks, mit dem zusätzlichen Ziel, den Abfall an Produktivität gering zu halten. Dies wird mit einer Simulation und mit einer verschachtelten Prognosesimulation gemacht. Beide basieren auf dem Hierarchical Control Conceptual Model und sind die Basis für Dreadlocks-Steuerungsmethoden. Durch diese Unterteilung der Deadlock-Vorhersage und der Deadlock-Steuerungsmethode wird eine hohe Flexibilität erreicht. Für diese Methoden können bereits bestehende verwendet werden oder völlig neue, die stark an das jeweilige Fertigungssystem angepasst sind, entwickelt werden. Auch das kombinieren von Methoden ist möglich.

Dies wird in einer Fallstudie getestet, die eine einfache Deadlock-Strategie mit drei Deadlock-Strategien, die auf der präsentierten Lösung aufbauen vergleicht. Die Ergebnisse zeigen, dass durch die neue Methode besser Deadlock Handhabungsresultate erreicht werden können, während die Effizienz zum weitgehend beibehalten wird. Sie deuten auch an, dass noch besser Ergebnisse erreicht werden können, wenn komplexere und besser and Fertigungssystem angepasste Methoden entwickelt werden.

Contents

List of Abbreviations

AGV Automated Guided Vehicle

DES Discrete Event Simulation

DML Dedicated Manufacturing Lines

FMS Flexible Manufacturing Systems

HCCM Hierarchical Control Conceptual Model

NC Numerical Control

RAEL Requested Activities and Events Lists

RMS Reconfigurable Manufacturing Systems

SEL Scheduled Event List

1 Introduction

Looking into industries, changes have taken place over the last few decades. Demand for personalization of customers, uncertainty through customer demand and the competitive factor of offering customization has brought high variation with low batch sizes into companies' product mixes. Another change that can be observed is that technological steps are getting bigger and therefore market changes are faster. This leads to shorter product life cycles. To keep up with these demands, production systems must be able to respond quickly to the variable market requirements (Li 2011: 1).
These challenging times for companies are leading to a perspective change on manufacturing planning and the emergence of flexible manufacturing systems (FMS). FMS have a strong focus on flexibility which generates new possibilities. There are many degrees for flexibility, for example variable product routing, where products have many possibilities to move through the system until they are finished or multipurpose machines that can process different products in various ways.

FMS are not considered as a new evolution of manufacturing system. They are rather an extension to the existing ones. They can be compared to the likes of dedicated manufacturing systems which, in this case, excel in opposite characteristic (ElMaraghy 2005: 262).

The main advantage FMS are known for, is the high degree of customization enabled by their flexibility. The drawbacks to this are less efficiency and an increase in complexity. This is the result of more degrees of freedom leading to more effort in production planning and control.

A commonly known problem are deadlocks, where a production system or a section of it is blocking itself, so that no parts can move anymore without intervention. Such deadlocks need intervention from outside, most of the time workers who re-established the flow of products again by moving products out of the deadlocked section and return them later. The unpredictability and the decreased efficiency of the system as well as the increased cost make deadlocks something to be avoided. This has led to the works of deadlock control strategies, which focus on methods to handle such situations. These deadlock control strategies can be assigned to three categories. Deadlock detection and recovery, which are algorithms that try to find where a deadlock has occurred with the intention to re-established to flow in the most efficient way. Deadlock avoidance algorithms try to detect deadlocks before they appear and consequently change the systems behaviour so that the deadlock can be avoided. The last category, deadlock prevention, which focuses on a system design and planning so that deadlock can never occur.

In this thesis a deadlock control strategy is presented that predicts deadlocks in the future to allow pre-emptive actions to avoid the occurrence of these. An advantage of this method is that the parts of deadlock prediction and avoidance policies are separated. This makes the strategy a general solution which can be adapted and optimized for specific systems.

In the end a case study is presented that compares an existing strategy with three variations of the predictive method. The results will then be discussed. It is also shown how these strategies can be implemented into a simulation to enable future work.

2 Theory

At first, an overview and definitions of FMS, Deadlocks and Simulations will be given. This includes the discussion of the current state of the art and problems that occur.

2.1 FMS Definition

It is well over 30 years ago since the discussions about flexibility in manufacturing started. Upton wrote 1995 that ten or 15 years before the same happened with the term quality. It was vague and difficult to improve but everyone knew of its competitive value. The same phenomenon is happening with the term flexibility. It is only at the beginning of being explored and it still has different meanings to different people (Upton 1995: 3).

2.1.1 Flexibility

Taking a closer look at the concept of flexibility, literature still shows that it is complex and hard to describe. There are more than 50 different types of flexibility, not all of which are in agreement with each other and are often vague defined (Sethi and Sethi 1990: 289).
For example, it can be seen from an adaptive or from a proactive point of view. While the adaptive view sees the ability to change, the proactive view uses it as a tool to increase customer expectations and therefore increase uncertainty for competition. (Jain 2013: 5947)

One of the earliest definitions is from Ropohl (1967: 644) where he defines manufacturing flexibility as property of linked system elements with the ability to adapt to various production tasks. A later definition by (Cox Jr 1989: 68) includes the aspect of the market as he defines manufacturing flexibility as “the quickness and ease with which plants can respond to changes in market condition”. Upton (1994) has a more comprehensive definition in “the ability to change or react with little penalty in time effort, cost or performance”. Another definition is from Garrett (1986) in which he sees manufacturing flexibility as the ability to handle environmental uncertainties which can be divided into internal uncertainties and external forces. While internal uncertainties or disturbances can be breakdowns, rejects, rework or queueing delays, external forces include demand, price, product mix or availability of resources.

Trying to find a standardized understanding of the topic, a few authors used taxonomies to categorise various types of flexibility. The following list of 11 types of flexibility by Sethi and Sethi (1990: 296-313) is a good comprehensive basis and commonly used and referred to.

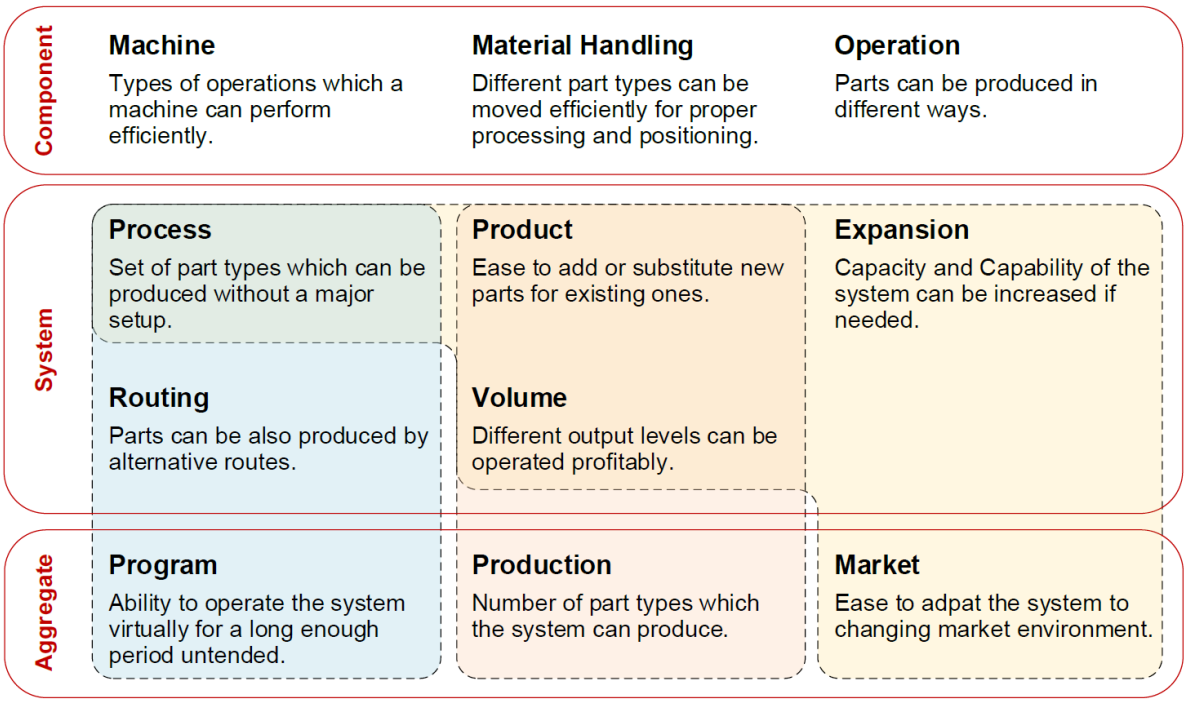

Figure 1: Flexibility Dimensions (Sethi and Sethi 1990: 297)

Not all of the types have to be present to define an FMS, they rather give an overview on how many dimensions of flexibility there are, while each dimension can be implemented to a different degree.

2.1.2 Measurement of Flexibility

There are two approaches with the goal to describe the degree of flexibility. The qualitative approach uses linguistic assessment while the quantitative approach tries to use mathematical models to define numerical values.

Due to the different perspectives and the vague definitions of flexibility the measurement of flexibility is still a subject that has to be worked on. Narain (2000: 205) identifies the following problems:

- Flexibility measures potential rather than performance
- Flexibility is multidimensional
- There is no coherent classification of flexibility
- It is difficult to determine the domain of flexibility
- It is hard to define universal measures since flexibility is strongly situation dependent

2.1.3 Elements of Flexible Manufacturing Systems

As the meaning of flexibility and flexibility of manufacturing systems has been discussed, the question arises what elements have to be present to be classified as an FMS. Stecker (1983: 273) defines FMS as a system consisting of an integrated, computer-controlled complex of automated material handling devices and numerically controlled (NC) machine tools that can simultaneously process medium-sized volumes of a variety of part types. In many cases, buffers are added at various places.

While these systems, which have automation as their key conceptual requirements, can be found often, a system with the right components does not have to be flexible as long as the system is not designed with one or more elements of any types of flexibility to a certain degree.

Browne (1984: 116) gives a classification of four types of FMS.

- Flexible Machining Cell: It is the simplest FMS since it consists only of one general purpose CNC machine. Semi-finished parts that were stored in an input buffer are brought to the machine by an automated material handling device and afterwards taken away to an output buffer. It has exactly all the components needed to classify as an FMS.
- Flexible Machining System: Consists out of multiple flexible machining cells with different general purposes, real-time part production control and multi routing, specialized on a small volume production. The flexible machining system can already fulfil a lot of flexibility dimensions. The dimension of flexibility can be extended depending on which material handling concept is implemented.
- Flexible Transfer Line: In this system there is always only one process step assigned to only one machine for every part. This is called fixed routing. The layout is ordered and most of the time the material handling system is arranged in a circle with buffers in between the machines. This set up is less flexible and rerouting has to be done manually, in addition breakdowns cannot be handled easily but once set up it is producing efficiently.
- Flexible Transfer Multi-Line: This type is an interconnection of one or more flexible transfer lines. This system I still very similar to the flexible transfer line but has more possibilities and therefore advantages in breakdown handling trying to combine the best of type 2 and 3.

Figure 2 Part of a Flexible Machining System (Unisig 2018)

2.1.4 Dedicated vs. Reconfigurable vs. Flexible Manufacturing Systems

In search for a production system that fulfils all requirements other approaches can be mentioned. The most know and traditional might be the dedicated manufacturing lines (DMLs). Its focus does not lie on being flexible but rather being optimized and pre-planned for one specific part type at high volume. Characterized through cost-efficiency makes it a good fit for mass production. The goal of mass production is to offer a high number of standardized products at a low price. As the name of DML suggests the route of products is often strictly in one line, whereas the other systems are able to change routes to increase resource efficiency (ElMaraghy 2005: 265).

The FMS on the other hand specializes in exactly the opposite, focusing on being able to process several types of products, with minimal or none changeover cost on the same system. In return the output level of products is lower. These properties make FMS ideal for a high degree of customization. Customization offers customers adjustments of products to their personal needs in exchange for a higher price. (ElMaraghy 2005: 265).

The reconfigurable manufacturing systems (RMS) lies somewhere in between the DML and FMS. It attempts to combine both advantages, depending on the market requirements, through the ability to change its manufacturing system in order to adjust production capacity and functionality within a given range of parts. The objective is to have an optimized production system which can be changed in certain time intervals. The output level of products lies in between the DML and the RMS. RMS enables mass customization which offers little customization options for only a small increase in price. (ElMaraghy 2005: 265).

Figure 3 shows a graphical explanation. FMS can produce a high variety of parts with a limitation of volume leading to a customization strategy. Dedicated Manufacturing Systems have a very low scope of variety but excel the others in output volume, ideal for mass production. In between these two options are RMS balancing variety and volume properties, customization, and mass production to mass customization.

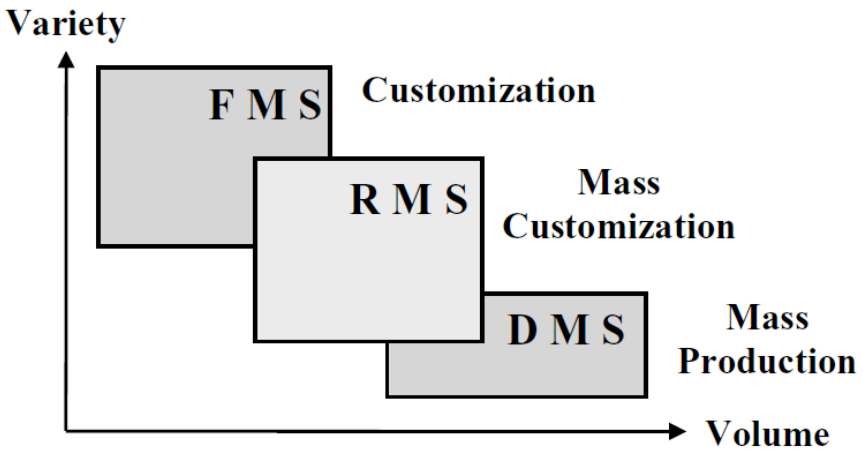

Figure 3: Categorization of Manufacturing Systems (Hu 2005)

2.1.5 Recap

FMSs are not always the best way, since “more flexibility does not always mean a more economical solution” (Lenz 1992: 22). High investment costs and the inability to take advantage of “economies of scale” limit the cause for FMS. Also, not all branches may profit from customization. Without a clear reason and strategy, the change to or the implementation of an FMS is a risk (Lenz 1992: 22).

If a decision for an FMS is made, a concept can be developed. Especially during the concept phase, the high degree of flexibility and degrees of freedom of FMS have to be carefully considered. They generate a lot of new chances but increase complexity. Multi routing of products, multiple different products, the ability to choose between machines while the system should still be kept expandable, are only a few examples what must be considered. The number of variables and their relation to each other makes this not an easy task, especially since some variables can have a dramatic impact on the systems. For all these options decisions have to be made, some in advance during the planning phase and some decisions even live during production. This increases the effort for production planning and to control such a system. A lot of work has been done about FMS design for pre-production planning to find principles on which elements and what arrangements are most useful (some have already been mentioned in chapter “Elements of Flexible Manufacturing Systems“). There has also been worked on how to find ways to optimize such complex systems to make them more efficient. A peculiar characteristic that arises from the degrees of freedom of such systems are deadlocks. Deadlocks a whole system or a section of it, that is blocking itself so that no parts can move anymore without intervention. The works trying to solve this problem can be found under the topic of deadlock control.

2.2 Deadlock Definition

Work on deadlocks had first been done in general computing, where problems with resource sharing happened more often and before the first ones occurred in manufacturing. Later, when flexibility in manufacturing rose, the existing knowledge about deadlocks could be applied on manufacturing, as can be seen by the Coffman conditions, which if true verify a deadlock and will be explained later on. While computing system resources are elements like processor and disks, manufacturing system resources are machines, buffers, transport units and more.

There are various definitions what a deadlock in an FMS is, or how a deadlock state occurs.

Zajac (2004: 367) describes: “Deadlocks occur through the arbitrary routing of various parts which intersect with each other and the limitation of buffering”.

Li explains the cause and state of deadlock as follows:

> The existence of resource sharing may lead to circular wait conditions, which is the real cause of deadlocks in which each of two or more jobs in a set keeps waiting indefinitely for the other jobs in the set to relinquish resources that they hold. In such a system, once deadlocks occur, they persist and would not be resolved without the intervention from human beings or other external agency[sic]. (Li 2011: 437)

And a mathematical definition which is in agreement with the definition before from Seidl and Günther states:

> An FMS state is called a deadlock, if a set of finite resources R_{DL} is completely allocated by a set of jobs W_{DL}, where no $r \in R_{DL}$ can be released without allocation of at least another $r' \in R_{DL}$. (Seidl and Günther 2000: 149)

In such a system, four conditions by Coffman et al. (1971: 70) can be mentioned that lead to deadlocks.

- Mutual exclusion: process require the exclusive use of resources.
- Hold and wait: a process is holding a resource while waiting to acquire additional resources.
- No pre-emption: processes holding resources cannot be forcibly removed
- Circular wait: a closed chain of processes exists in which each process is waiting for a resource held by the next process in the chain.

The first three are present in nearly all FMS. Closed chains are often needed to get the full potential out of the FMS. Nevertheless, an FMS without circular routing would prevent deadlocks from happening.

2.2.1 Deadlock examples

A simple deadlock example can occur in traffic management at a roundabout or like layouts as shown in Figure 4. There are 4 crossings with green arrows showing the entries and red arrows showing the exits. Each rectangle represents a car with the arrow pointing in its moving direction which is also pointing in its desired exiting location. It can be observed that the system is filled and no capacity is left, so no car can enter the system, since all exits are also blocked, the system is in a deadlock state and unresolvable until a car is forcefully removed from the system (Coffman et al. 1971: 69).

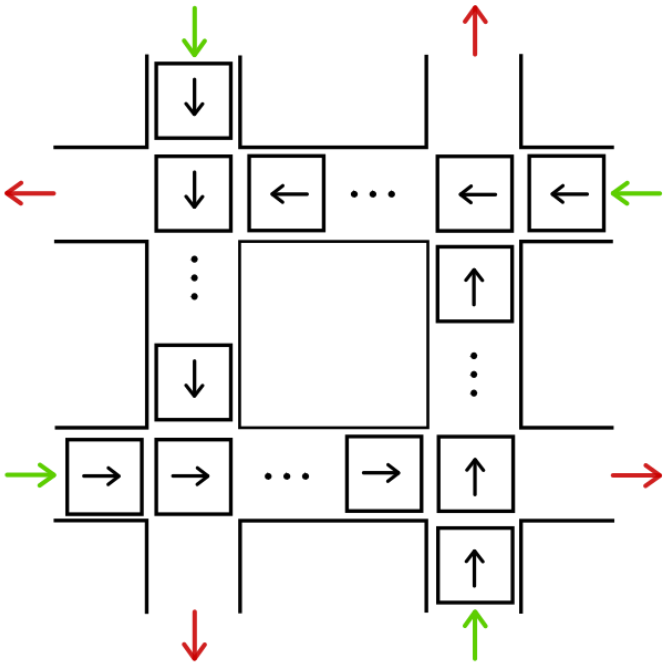

Figure 4: Crossroad Deadlock Example (Coffman et al. 1971: 69)

Another two examples are shown in Figure 5 Circles represent Machines which can be occupied by a product, represented by dots. Products then again have desired directions they want to move along indicated by arrows. For the purpose of simplification, no transport units are needed, and no buffers are given.

Example a) is the smallest possible circle and therefore the smallest possible deadlock. A Product from machine A wants to go to machine B and vice versa. Since there are no buffers or transport units, the machines are blocking each other, making it impossible to move products.

In example b) four machines A, B, C and I, which can be seen as input, are given. The desired rout of product 1 is I → A → B → C → A→ B→ and then leaving the system. The rout of product 2 is I → A → B → and then leaving the system. The upper state is critical. There are two possible movements. Either product 1 moves from machine C to machine A and therefore resolving the critical state by not getting into a deadlock state, or product 2 moves from Input to machine A which creates a deadlock.

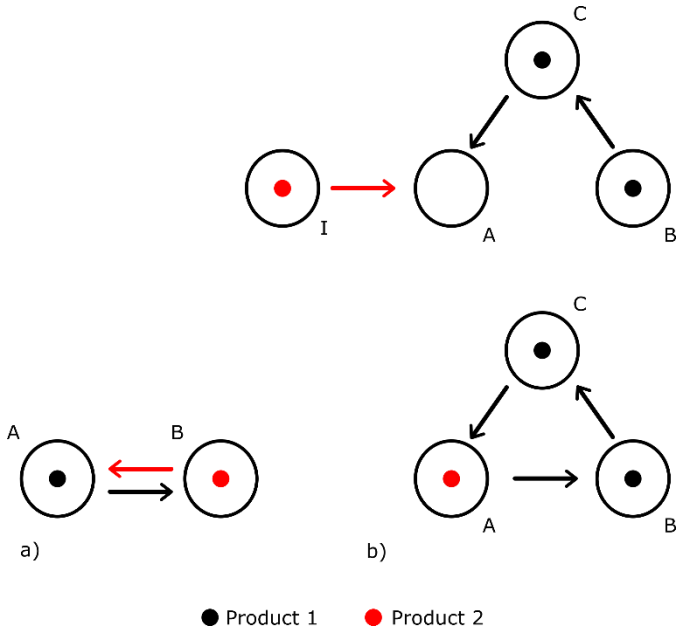

Figure 5: Simple Deadlock Scenarios

2.3 Methods for Deadlock control/handling

There are different easy strategies to solve deadlock scenarios with guarantee, so that even if the production system is completely blocked, steps can be followed to resolve the deadlock and re-established the flow of products. But they are most of the time to conservative, meaning that performance suffers too much because resources are used inefficiently, or additional costs arise. A simple method would be to remove a product from the wait circle and put it back in after the flow of products has been re-established. Methods like this often need employees to intervene, are time consuming and lower performance. It also creates the question where and when to reinsert the product. There are also solutions which do not lower the performance but have the disadvantage to not guarantee a resolution of the deadlock

situation. The decision has to be made according to the concept and the needs for manufacturing.

2.3.1 Modeling tools for Deadlocks

To make deadlocks easier to understand or help communicating graphical tools are of good use. They are not only used for showing a static state but can also help understanding dynamic steps of the system. Currently three graphical tools are commonly used. Diagraphs, automata and petri nets.

Diagraph or graph theory is part of discrete mathematics and an easy and intuitive tool to show, communicate and analyse interactions between resources and their relationships. As seen in Figure 5 deadlock situations can be presented easily, and control policies can be derived (Li 2011: 438).

Automata is a graphical tool that represents transitions of system states. It is often used in computer science with the intend of running through predefined states the system can get into. At each state a string is received, and a transition function determines the next steps.

Figure 6 shows two states. The first one is the initial state. If the string “a” is received it transitions to the next state, state 2. State 2 is an ending state, indicated by the double circles, in which it will stay as long as it receives “b” as input. If the string “c” is sent it will transition back into the initial state 1. (Wainer 2009: 17)

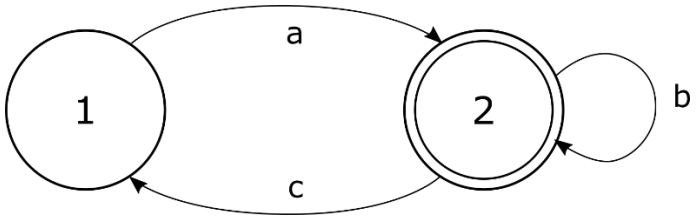

Figure 6: Automata Example

More commonly used are petri nets. They consist of places, tokens, transitions and arcs. An example, shown in Figure 7, would be, a token/product is currently placed in a place/machine (p1) where it wants to transition (t) along the arc/rout to another place/machine (p2). A Transition is a logical controller that only lets tokens move if the logic is fulfilled, if that happens the token “gets fired”, disappears in place 1 and appears in place 2. Between two arcs there is always a transition and no place is connected with another through an arc without transition. Petri nets do have, in comparison to digraphs, more information about the transition of parts. As can be seen in, they also carry information about the path and how many tokens will be fired. Petri nets are also well suited for deadlock detection due to the formulation of liveness. A transition is live if it is able to fire. If the transition is unable to fire it is called dead. Therefore, a circle of dead transitions is a deadlock (Cardoso and Heloisa 1998: 14).

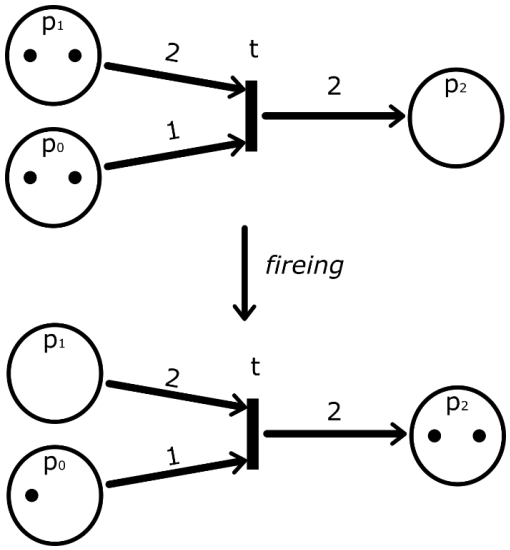

Figure 7: Petri-Net Example

There are currently three major approaches for deadlock handing called, deadlock detection and recovery, avoidance and prevention (Li 2011: 438). In some cases, methods can be combined.

2.3.2 Prevention

This method is off-line, meaning that the deadlock handling is done in advance before the system starts running. The idea behind it is to design a control policy and the system in a way that it is not possible that deadlocks happen. This could be done for example, by creating a product schedule, which defines which product enters at which time. In other words, it must be assured, that at no point in time all four of the Coffman conditions hold true. Stochastic events like machine breakdowns are typically not considered and can still lead to deadlocks (Li 2011: 439).

An advantage of deadlock prevention is, that since the policies are developed off-line and in advance, no runtime is needed. The biggest disadvantage is that most of the policies are too conservative.

Two examples will be given. The first one is called block an recirculate (Figure 8), where a conveyer runs in a circle and connects all machines. If the machine, that is required from the product, is blocked the product stays on the conveyer until the machine is ready to process. This approach is wildly accepted but the drawbacks are clear. Waiting time for products to rotate, limited space on the conveyer and restrictions in layout limit the flexibility and have to be accepted. The second example uses the same principle of offering enough space by haven a high buffer space in front of each machine. With this method products never have to wait in a machine and can always move forward (Kim 1997: 1549).

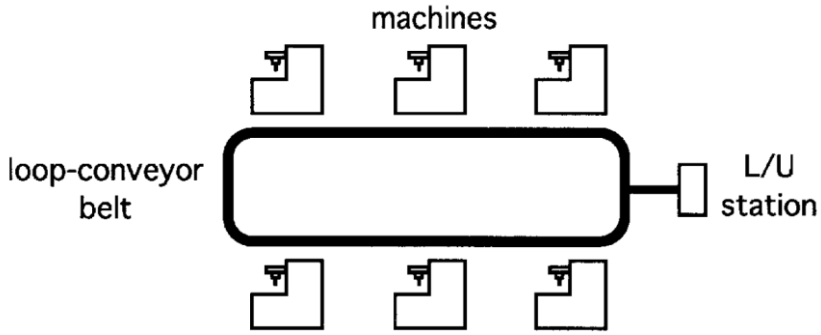

Figure 8: Loop-Conveyor System (Kim 1997: 1548)

2.3.3 Detection and recovery

The detection and recovery methods allow deadlocks to occur. The task of detecting is to locate where the deadlock happened, and report back the right information that is needed to identify if the system is in a deadlock state. This means that this approach operates on-line and therefore is working while the system is running. Such algorithms are usually executed over certain time intervals. If a deadlock is found, the system then recovers by aborting one or more tasks which are involved in the deadlock by removing all resources currently needed by the task and through this, bypassing the pre-emption condition. This condition is one of the four Coffman that needs to be true so that a deadlock can occur. Another job of the recovery process is to find an efficient way to re-insert the aborted task. The recovery process often requests human operators and thus, can be very expensive depending on how often deadlocks occur in the system (Li 2011: 439).

Wysk, et al. (1991: 855-857) presented a deadlock detection based on a string multiplication algorithm to identify circuits. A symbol matrix S is defined that shows all connections between machines. For example, if a wait relation between machine 1 and 2 exists, it is signalled by writing 12 into the matrix, symbolising the machines and not numerical values. If no wait relation exists 0 is written. Two arc circuits are in S^2, three arc circuits are in S^3 and so on. After the string multiplication all circuits are defined. Furthermore, circuits have to be validated and intersecting circuits have to be investigated.

Fanti et al. (1996: 237-239) detects deadlock with the help of a depth search in a diagraph and the definition of a Maximal-weight, Zero outdegree Strong Component (MZSC). All blocked jobs are written into a list. Starting from the first blocked job, if all vertices reachable from there are busy, the diagraph contains a MZSC. Furthermore, if the resource of the blocked job is element of the MZSC a deadlock is detected, the recovery procedure starts.

This procedure solves the deadlock in four steps by first selecting a deadlocked cycle. After the selection it removes a job into a buffer. The system is now running again and while this happens a restriction policy is set that inhibits new resources to get into the system that would reach the circle and also inhibits jobs that would transition corresponding to edges that would lead into the circle. At a certain point the restriction is removed. The last step is to move the job from the buffer back to the resource (Fanti et al. 1996).

2.3.4 Avoidance

Deadlock avoidance is also an on-line method, looking one or multiple steps ahead in time to check if the resources are safe to be released from their current location or can be used by other resources, without causing a deadlock. For this method to work a good communication in the system has to be installed to provide the data needed for policy change algorithms. This method is the least conservative but with the drawback of not always guaranteeing deadlock freeness (Li 2011: 439).

For example, there is the possibility that a one-step look-ahead is not enough, and a deadlock cannot be avoided because all scenarios in the next step lead to an unavoidable deadlock. Therefore, multi-step look-ahead deadlock avoidance policies have been developed in recent years. Since these algorithms have to calculate all possible outcomes with respect to the next system change and in the case of multi-step look-ahead also all possibilities in the next few steps, especially big manufacturing systems take a multiple of runtime and processing effort in comparison to other policies. Even though, computational performance has increased over the years and it is expected to further increase, the efficiency of these algorithms is key. If the on-line policies cannot be calculated in time, the calculation would lag behind the real manufacturing system and therefore be without use (Li 2011: 439).

The most common method is the use of the Bankers Algorithm. At the entry of a product/process its demand for each resource is written down in a 2-dimensional array. And next the number of resources of each type that is currently allocated to each product/process. The two arrays will be subtracted to get the need-array. The safest option now is to allocate the free resources to the product/process with the lowest need value. If there are enough resources to allocate, the product/process for this step is done and its resources are set free for the other products/processes. This step of finding the next lowest need value, allocating free resources to it and setting them free again is repeated until there are no more products/processes. This way, every step the safest way to allocate resources for products is found. Problems with this algorithm are for example no robustness to resource failure or that the principle only works at certain points in time where many resources have to be considered at once. There are many principles building on this algorithm to extend the algorithm in different directions (Lawley and Sulistyono 2002: 351).

One of the early works on deadlock avoidance with petri-nets was done by Viswanadham et al. (1990: 718-720). Using a one-step look-ahead and a reachability graph to find out if the next transition is safe or if it would lead to a deadlock. This is done to find a decision what transition to fire. A simple example shown in Figure 9, consisting of an input station, an automated guided vehicle (AGV) and a NC machine. In Figure 10 the petri-net to the example and the transition graph can be seen. Starting from marking P_1 two transition t_1 to Place P_3 and t_2 to place P_4 are possible. P_3 represents the AGV carrying raw material and P_4 carrying a finished part. If t_2 gets fired and the AGV carries a finished part but there is also already a raw material in the station a deadlock would happen, therefore t_1 has to be fired before t_2.

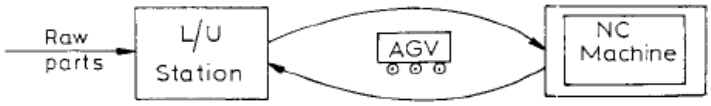

Figure 9: Simple Manufacturing System, AGV and NC Machine (Viswanadham et al. 1990: 713)

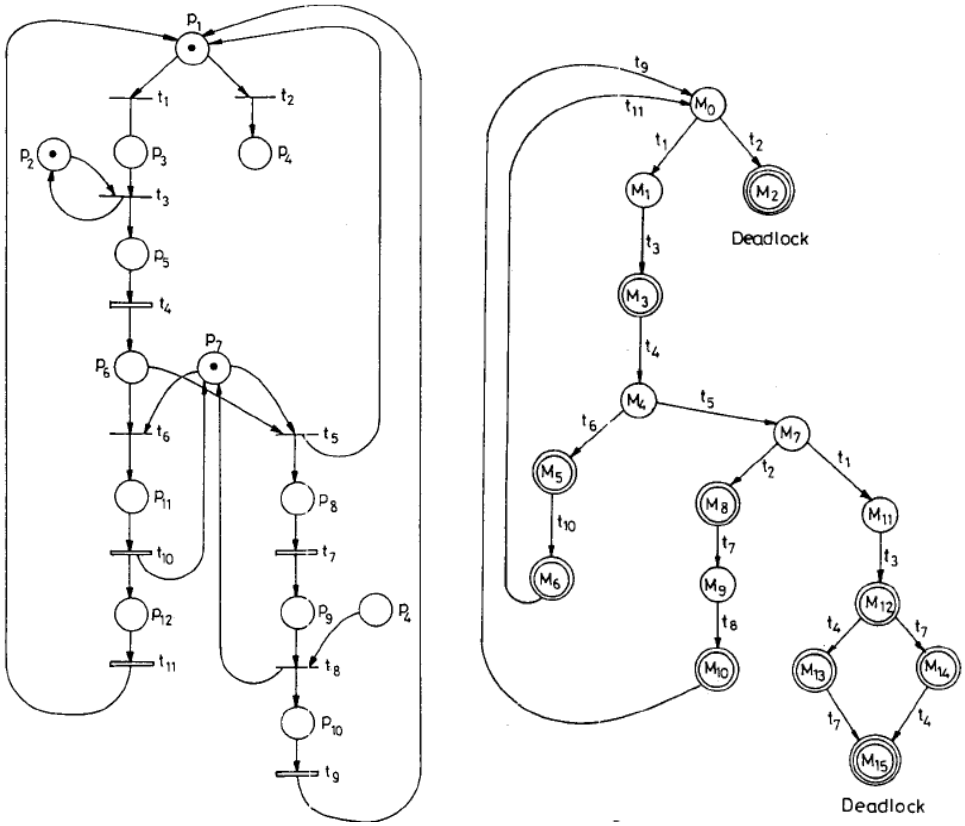

Figure 10: Petri-Net Model and Reachability Graph of Simple Manufacturing System (Viswanadham et al. 1990: 718)

Places :

1 : AGV available

2 : Raw parts available

3 : AGV available to carry a raw part

4 : AGV available to carry a finished part

5 : AGV carrying a raw part to the NC machine

6 : AGV, with raw part, waiting for the NC machine

7 : NC machine available

8 : NC machine processing part; AGV released

9 : NC machine waiting for AGV, after finishing processing

10 : AGV unloading the finished part

11 : NC machine processing a part; AGV not released

12 : AGV, not released during processing by machine, unloading a finished part

Immediate Transition:

1 : AGV assigned to raw part

2 : AGV assigned to finished part

3 : AGV starts transporting a raw part

5 : AGV released after finding a machine free

6 : AGV not released after finding a machine free

8 : AGV starts unloading a finished part

Timed Transition:

4 : AGV carrying a raw part to the NC machine

7 : Machine processing a part; AGV released

9 : AGV carrying a finished part to L/U station

10 : Machine processing a part; AGV not released

11 : AGV, not released during processing by machine, carrying a finished part to L/U station

Table 1: Description of Petri-Net Model

2.3.5 Current Research

Current deadlock control researches focus on specific cases like railway systems by Fanti et al. (2006: 1231) or semiconductor fabrication by Zhu (2014: 117). Others need to change or manipulate the model in order to apply the deadlock control (Xing et al. 2011: 608). Due to the complexity of the systems and the inability to find the perfect solution in a practical time

heuristic searches are used. The extension of petri nets to time colored petri nets makes deadlock control concepts easier to formulate, where color is used to differentiate between different tokens and the concept of a global time is added. (Baruwa et al. 2014: 833). Current deadlock avoidance strategies using multi-step look-ahead are relying on strictly defined models and model analysis to find the right avoidance policy. Part of the avoidance strategy is reducing and parameterizing (Gu et al. 2018).

Even though not all works on deadlock control have been analyzed, they seem to be focusing on specific problems and are therefore standalone solutions that are not easily combinable. The practically and applicability to real manufacturing systems that can have a high variability in their set up and workflow still has to be proven.

2.4 Discrete Event Simulation (DES)

In this section a brief introduction on simulation is given. At first, definitions about modeling and simulations are stated and classified. Based on this, DES and the three-phase-approach can be explained. Finally, a new view of DES in the form of the Hierarchical Control Conceptual Modeling (HCCM) is explained which will be used later on.

2.4.1 Modeling definition

When confronted with problems it can be observe in life that it is difficult to find exact solutions. The complexity of problems is high due to the high degree of detail. It is difficult to describe and consider every element, influence and dynamic that can observe in reality. A way of dealing with this is to find an abstraction of the system by using a model. Assumptions and simplification can be applied to a model in order to make the system more predictable or to reduce its complexity if only a part of the system is of interest for in the problem. It has to be taken care of not simplifying the model so much, that the results are not comparable with the real system anymore. The implementation of a model for this use is called simulation (Wainer 2009: 4).

2.4.2 Simulation definition

A definition for simulation by Robinson considers the four aspects of operations systems, purpose, simplification and experimentation.

> Experimentation with a simplified imitation (on a computer) of an operations system as it progresses through time, for the purpose of better understanding and/or improving that system. (Robinson 2004: 4)

Or in other words. A simulation is an abstraction of a real system used for experiments with the intention to get information about its behaviour and how it can be influenced with a concept of how it progresses to solve a problem.

Simulations can be done physically through experiments but at present time most of the simulations are done only virtually with computers. This is because experimentation is often not a feasible solution due to ethics, risks or cost. Another advantage of virtual simulations is the repeatability and the ease to change parameters to observe their influence quickly (Wainer 2009: 3).

Simulations can be classified on how variables change over time. In discrete simulations the state variables will only change at countable points in time for example a traffic light. Continuous simulations have their state variables changed constantly over time like the temperature in a room (Robinson 2004: 24-25).

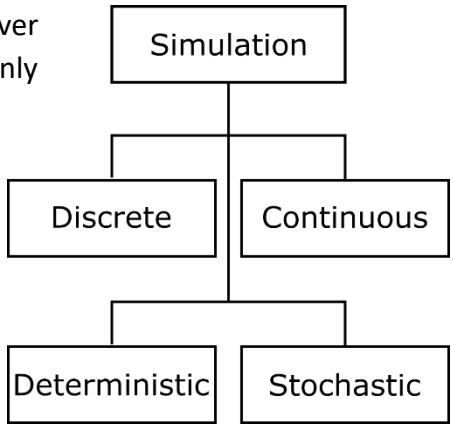

Figure 11: Classifications of Simulation

Another differentiation is if a Simulation is deterministic or stochastic. If deterministic, all variables are know with certainty. In stochastic simulations at least one variable is probability distributed (Robinson 2004: 138-139).

2.4.3 Discrete-Event Simulation

Discret-event simulations are a special kind of simulation that evolve around a series of events, which are state changes that happen at certain instants called event time. The series of events is also called event calender and is orderd descending by the event time. Other points in time are not relevent to the system and can be skipped. In a state change at least one element in the system has to change. Examples of events can be seen in Table 2 where a customer arrives, a operator starts service, completes service. All these events occur in an discrete instant (Robinson 2004: 15-18).

Time	Event
3	Customer arrives
	Operator 1 starts service
6	Customer arrives
	Operator 2 starts service
8	Operator 1 completes service
9	Customer arrives
	Operator 1 starts service
11	Operator 2 completes service
12	Customer arrives
	Operator 2 starts service
14	Operator 1 completes service
15	Customer arrives
	Operator 1 starts service
17	Operator 2 completes service
18	Customer arrives
	Operator 2 starts service
20	Operator 1 completes service
21	Customer arrives
	Operator 1 starts service
23	Operator 2 completes service
24	Customer arrives
	Operator 2 starts service

Table 2: Simple Telephone Call Centre Simulation (Robinson 2004: 16)

Other important parts of DES besides events and the eventcalender are, a simulation clock that represents the time during the simulation and activities that last for a specific time and are started and ended by an event. An activiety could be getting served at a counter, which starts with the event “start service”, lasting for a specific time, while getting served, that

afterwards ends with the end event “service finished”. Futhermore entities have to be mentioned. Entities are the moving or interacting parts in the system that are adressed during state changes. Entities have characteristics called attributes which define what they are. For example a customer has as an attribute the time of arrival and a reason for coming into the system, maybe an injury. While different classes of entities can have different attributes, entities of the same kind can have differen values of their attributes. Entities compete for entities and use them, like a counter or an operator. If an entitie is used by another it can be called resourece. A resource has a capacity for how many entities can use them. If the limit is reached no entity can access the resource anymore. In this case the entities have to wait in queues, which are places where the entities wait. Also queues can have a capacity limit (Robinson 2004: 2,11,18).

To enable a DES, policies have to be defined which set guidelines on how entities have to be treated. If a counter is free again and a customer from the queue can be served the policies have to define which customer is the next one in line (Robinson 2004: 7).

2.4.4 Three-Phase Approach

In the three phase approach events are divided into two categories:

- Bound or Booked events are state changes at a given point in time. A customer arrives every 5 minutes, for example.
- Conditional events are dependent on changes in the system, and therefore the state changes happen. An operator can only start a service if the customer has already arrived.

At first the simulation is initialized. All key elements are set up to the initial state and the first events are scheduled. In phase A the time of the first event is picked up and the simulation advances to that time. In-between this time no state changes happen. Now all booked events that are scheduled at this time are executed. Some of these booked events might create conditional events that will be executed right after the booked ones in phase C or at a later point in time. Since the execution of a conditional event can also create another conditional event or even a chain of events, which are called sequential events, that have to be executed all at the current time, a loop has to be carried out to check if there are any new conditional events that have to be triggered. After all events at that certain time have been carried out, a logical question decides if the simulation has to run further, searching for the next time in the event calendar or if the simulation is stopped. This could be for example if the simulation has reached a specific time or if there are no more events in the event calendar (Robinson 2004: 17-19).

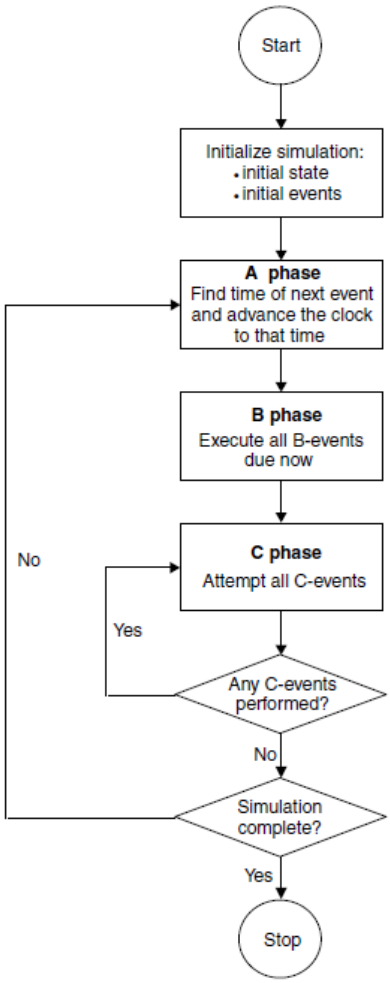

Figure 12: The Three-Phase Simulation Approach (Robinson 2004: 19)

2.5 Hierarchical Control Conceptual Modeling (HCCM)

The trigger of conditional activities in the three-phase approach can be summarized as: if a condition evaluates to true an entity is chosen from the queue and with it an activity is triggered. This approach needs a rather rigid connection of resources to queues and the autonomy of queues to each other. With rising complexity this approach reaches its limits soon and customized workarounds are often designed in practice (Furian et al. 2014: 207).

2.5.1 Extended Activity Classification

While researching DES in health care systems, which are often very flexible, Furian et al. (2014: 208) found other problems with the standard approach. It could also be seen that that the conditional activities trigger could be classified into two groups and therefore needed some extension. In some scenarios, health care personal had to move through the triggered condition of patients. Patients requested personal. Hence, these activities are called requested activities and are not much different as in the standard approach.

On the other hand, other motivations for activities could be seen that were not conditional to other elements in the system but rather conditional to the system. An example would be the anticipation of future requests or in general optimization. Since these activities are

determined and triggered by the control policies of the system they are referred to as controlled activities (2014: 208).

2.5.2 Hierarchical Simulation Control

Another problem occurred that in some scenarios an element could be a resource or an entity depending on the perspective. The following hierarchical world-view by Furian et a. (2014: 209), replaces entity queues through activity requests combined with control units and makes the differentiation between entity and resource unnecessary. The control units are hierarchical structured to a tree of control units.

The whole system is divided into, by its task distinguishable sub systems or organizational areas. In some cases, it is useful to implement further sub systems into an existing sub system. Each of them, has its own control unit that is responsible for the control policies. In addition to the control policies they are also controlling a list of pooled requested activities and events. These lists are called Requested Activities and Events Lists (RAELs) and replace the standard mechanism of separating activities into different types and corresponding queues. The RAEL only hold requests that could be performed in the current state of the system (Furian et al. 2014: 209).

Through this, by organizational areas classified, hierarchical structure and the use of assigned RAELs, an organized structure is built that establishes a clear overview even in complex system. Provided, the depth and design of the tree is modeled with care.

A core element of control units are rules. The five categories for rules that can be distinguished are assessment, dispatching, control, replace and custom rules. Assessment rules check which activities could be carried out, depending if entities are available, and adds these activities requests to the RAEL. Dispatch rules decide which request of the RAEL is next in line. Control rules are responsible for triggering the behaviour that is the result of the dispatch rules or behaviour to influence the performance of the system. Activity replacement decides when and which requests are removed from the RAEL. In the end custom rules are for any control functionality not covered in the categories above (Furian et al. 2014: 209).

In some cases, actions happen that a control unit is not responsible and designed for. In this case delegates, which are responsible for communication between control units transport the information either up-, down- or sideways in the tree.

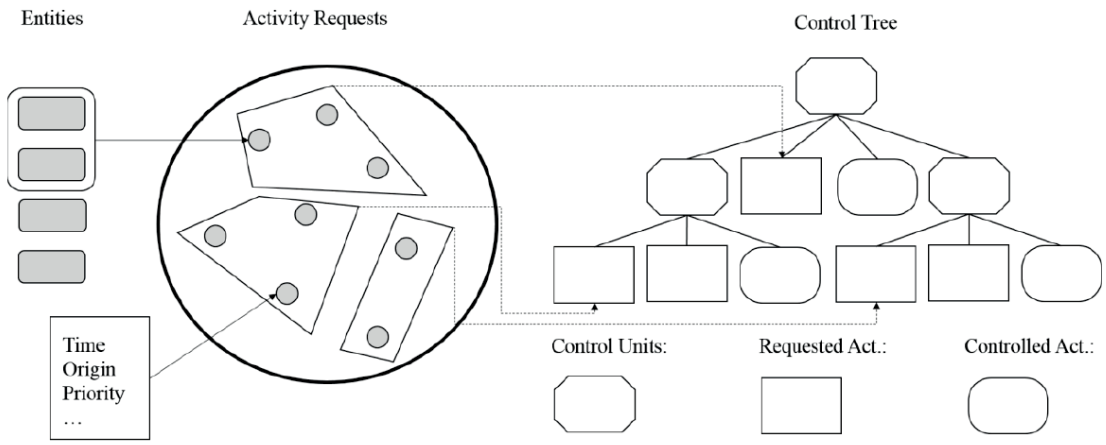

Figure 13: Concept of Hierarchical Control World View (Furian et al. 2014: 210)

2.5.3 Time Advancement

As in the three-phase model time advancement happens through scheduled behavior. The event calendar or Scheduled Event List (SEL) hold all events, sorted by time. At the start and after each finished execution of an event, the simulation searches for the next time in the SEL and advances to that point. All events at that time are started. This happens from the top of the control tree downwards. All rules are executed in the following sequence: assessment, dispatch, control replacement. As soon as a request or an event is triggered the simulation jumps back to the first rule, the assessment rule. If no further action were launched, the delegates are sent. This procedure will be repeated until to actions are executed and no changes are happening at this specific time anymore. After that the next time of the SEL is looked up and things repeat until either there are no more events in the SEL or another stop-condition is met (Furian et al. 2014: 210).

The exact algorithm of execution can be seen in Figure 14. It starts at “Select Next Item from SEL” and selects the time of the next item. It continuous by updating the time. Afterwards the stop-conditions are checked, for which examples were already mentioned. Next, the scheduled items are executed, and rules are performed. It is checked afterwards if any new behaviors are triggered and a conditional loop is started. In this conditional loop another stop-condition is implemented, checking each round if the simulation has to be stopped. If no additional behavior is found the simulation advances in the hierarchical tree by sending delegates if needed. Should the sending of delegates be necessary, the conditional loop is executed again until no new behavior occurs. If there are no more delegates left, all actions at that time are executed and the simulation can select the next point in time. This again, is repeated until the stop-condition is met (Furian et al. 2014: 210).

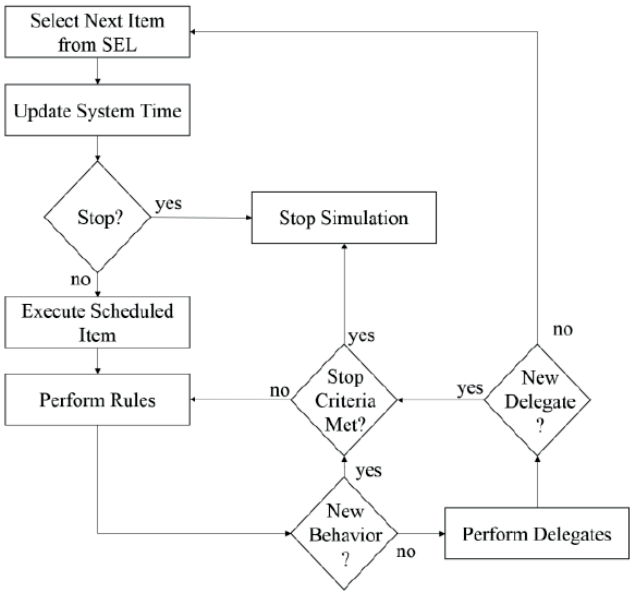

Figure 14: Time Management Algorithm (Furian et al. 2014: 210)

2.5.4 Structure of the HCCM framework

Figure 15 shows the underlaying structure trough four phases, that if followed can help to set up the model step by step. Phase one is about understanding the problem with the intended result of an informal, textual description of the problem situation. Assumption made during this phase should be documented and also included in the problem description (Furian et al. 2015: 89).

In the “Identifying the Goals” phase two categories of objectives have to be defined. General objectives are requirements needed to be fulfilled by the simulation model. For example, calculation time, visualization or changeability for other cases. Modeling objectives define what can be achieved from the development and us of the model (Furian et al. 2015: 89).

Phase three defines input and output factors. Input or also called experimental factors are values that can changed over different experiments to achieve a different outcome. Output factors represent the outcome of a model and should have some relation to the problem situation (Furian et al. 2015: 89).

In the last phase the model structure, individual behavior and system behavior is defined. The model structure is set by the entity structure and the relation of them to each other. It is recommended to capture the system in an informal graphical way. This way it is easy to determine the flow and interaction of entities. The individual behavior of each entity is identified in the next step. Using the queueing example before, a customer arrives, he will than wait in a queue until a server is ready, receives serves and after that leaves. The server for example will wait until a customer comes to it, starts the service and after finishing will wait again. Also, other behavioral information needs to be defined. Attributes of entities like the arrival time or how many customers a server can accept at the same time, participating entities, state changes and information about the requests made. The last step is the system

behavior that defines the tree structure of the control units and their rules, which were both explained before (Furian et al. 2015: 90-92).

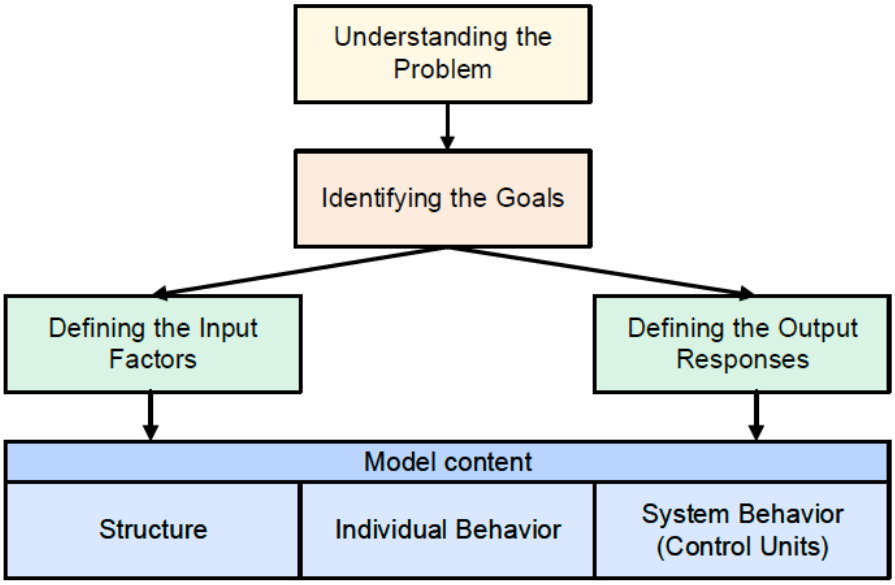

Figure 15: Structure of the HCCM Framework (Furian et al. 2015: 89)

3 A Simulation Approach to Deadlock Control

Deadlock control is currently an important topic in industries. But as already mentioned, literature delivers either a very theoretical approach or case studies with a detailed system and a specific solution, both of them not easy to use as general solution.

In this chapter a solution approach is presented, with the intend to be general applicable or easy customizable. Its advantage lies in splitting the detection from the recovery or avoidance part. This happens through a deadlock prediction by simulating into future time steps, by which information about the later system state is generated. This could be for example, knowledge about the predicted deadlock like deadlock time or involved entities. But also, information about the system not directly connected to the deadlock could be used, like a possible drop off in throughput before the deadlock or number of products in the system compared to maximum capacity of the system. Through this, more information than the current state and the layout, that most existing methods use, can be utilized to build avoidance or recovery policies on it and to adjust them precisely to either influence the flow of the system beforehand or to set the right measures for an imminent deadlock.

The principle is applicable to many different scenarios where entities need resources and are routed through a system that is able to fulfil the deadlock conditions. If the model gets more complex and additional information is added, more sophisticated polices can be made. Because of this the method can be easily adjusted to any complexity of a model.

3.1 System Definition

Key elements in the system are products, machines and buffers. Products need a specific set of process steps in a defined order called process route or routing. After the product has received all processings, it is finished and can move out of the system. Machines are able to perform a specific process step or a set of process steps and can treat one product at a time. In front of each machine is a buffer that can hold one product. If the machine is done processing the product, but the product cannot move onward because the next machine is currently processing and its buffer is also currently occupied, the product waits in the machine. If the next machine is empty the product will be sent. The system operates in a push principle, since the information flow is in the same direction as the product flow (Bonney et al. 1999: 55).

Products arrive in certain small deviating time intervals. This implies that the arrival flow of products is not dependent on the current state of the system or specific, how many products there are already in the system. That's why there is an overflow buffer at the input of the system, storing all products until the system is able to take in another product.

Due to simplification reasons no transport units are included, meaning that products move by changing the location where they are currently at to where they will be in an instant. This does not change the outcome significantly since transport units influence the system mainly in two points. First, they can be seen as additional buffer while holding a product during transportation. And second, as bottleneck, making products stay in the machine while waiting for transportation. Both raise complexity of the system but do not add a new mechanism/dynamic to it.

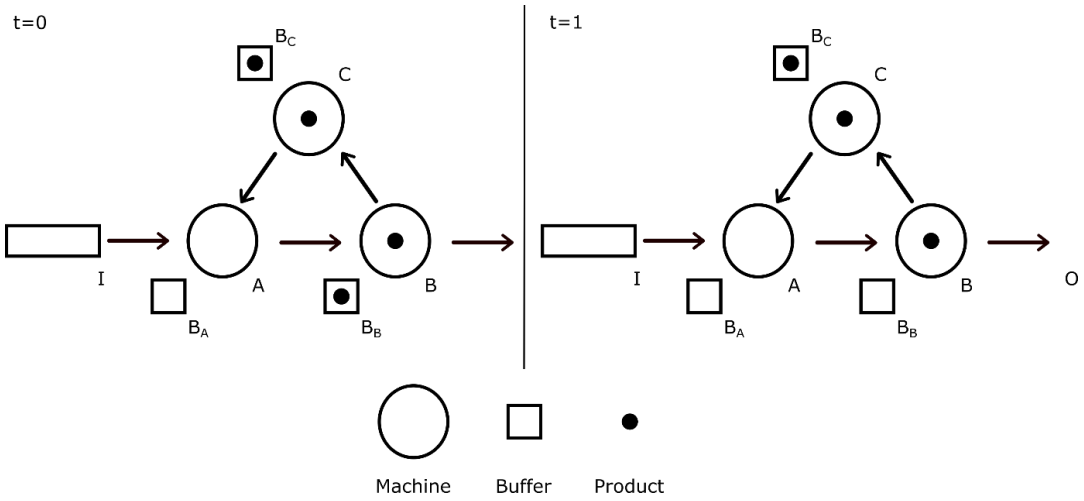

Figure 16: Elements of the Production System

3.2 General Idea

An overview on how the new method, that will be presented, has already been given before. Furthermore, a basic deadlock recovery method that will be called “Locked Buffer” and functions by disabling a buffer temporarily to artificially reduce the maximum system capacity until a deadlock occurs, will be tested. Next, the prediction method will be explained in detail and after that, combined with the locked buffer strategy. An additional combination of the prediction method with an early policy change to avoid predicted deadlock in early stages is also tested to show the versatility of the method. During both combinations other extension possibilities are given to give an overview of the full potential.

3.3 Simulation Procedure

The procedure is centered around products and starts with the arrival of a product that gets placed into the input buffer. As soon as a product arrives, a new product arrival is scheduled, providing a continuous operation of the simulation. The arriving product request to get machined according to its process route. If the desired machine is not idle the product requests to get placed into the buffer in front of the machine. If the buffer is occupied the product waits in the current machine and blocks it. Otherwise the product is placed in the buffer and requests to get into the machine until the machine is idle. As soon as the product gets placed into the desired machine it gets processed. The process time is predefined and depending on which machine it is. Upon completion of processing the product is either finished and moves out of the system or requests for the next machine until it is finished.

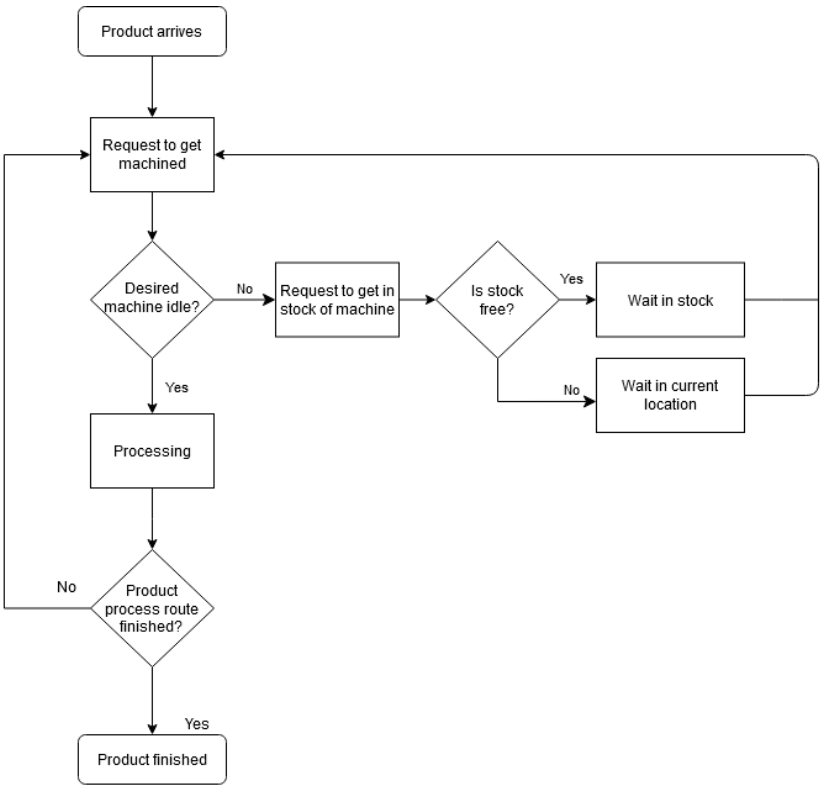

Figure 17: Flowchart Simulation Procedure

3.4 Deadlock Detection

This approach focuses mainly on machines as resources since they will be the limiting factor for deadlocks to occur. Two important states of machines can be distinguished for deadlocks, blocked machines $m_{blocked}$ and blocking machines $m_{blocking}$. Blocked machines are occupied by a product that cannot advance to the next step and is therefore waiting. In front of every blocked machine is a blocking machine that is either currently processing or itself blocked by another machine. In the first case the machine is only blocking, in the second case it is blocked and blocking.

Defining:

If $M_{blocking} \subseteq M_{blocked}$, the circular wait condition is met, and a set of machines is in deadlock.

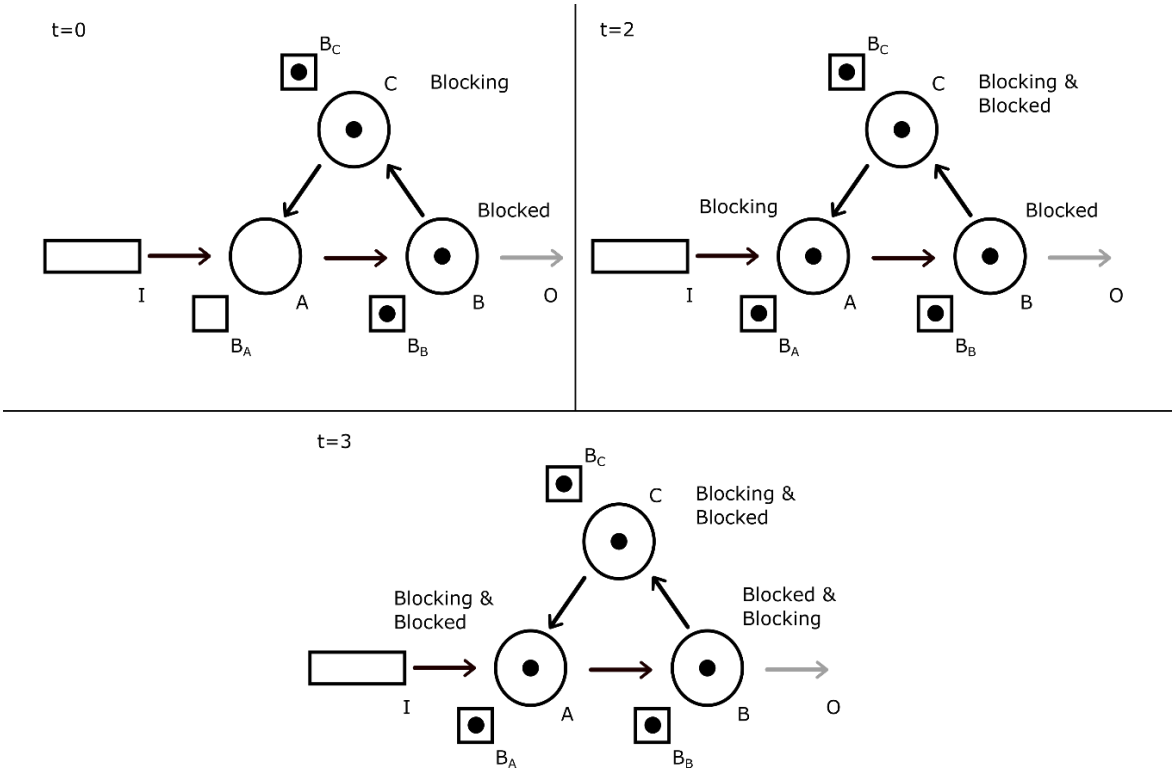

Figure 18: Blocked and Blocking Machines in a Production System

Figure 18 shows three time steps. The first state represents three machines in which two machines (B, C) and their stocks are occupied. Machine B is blocked and therefore finished processing. Because there is currently no product in machine A that wants to move over to machine B, machine B is not blocking. Machine C is currently processing and therefore blocking Machine B. At t=1 and t =2, two products get into the system, filling machine A and its stock. Also, machine C finished processing and cannot move forward to machine A, leading to being blocked and blocking because machine A is blocking. At this point the deadlock can already be seen but is not detected until machine A is done processing and tries to move the product at t=3. As soon as this happens also machine A is blocked and the deadlock condition, that has been defined before, is true.

3.5 Deadlock Controls

The following methods are a combination of deadlock recovery and avoidance, which can be seen as the basis for more complex strategies. This thesis will focus on more simple solutions to make the principles clear and more generally applicable.

3.5.1 Locked Buffer

The following deadlock recovery method is a simplistic approach and therefore easy to apply generally. Since all places in a circle have to be occupied by products for a deadlock to happen, a solution is to guarantee that there is always one product less than the maximal possible number of products in a circle. This can be achieved by locking one buffer in front of a machine (in the case of no buffers, a machine could be locked instead). A locked buffer or machine is

temporarily disabled and cannot be used. As soon as a deadlock is detected the locked buffer is set free. This guarantees at least another step with the chance of lowering the critical state in the system and re-establishing the flow of products. As soon as the product leaves the buffer it is locked again until the next deadlock occurs. It has to be noted that a locked buffer has to be in every independent circle. The drawback of locking buffers for most of the time is a lowering in performance (Xing et al. 2011).

Depending on how many products there are in the system the likelihood rises that right after the buffer is set free another deadlock emerges with no ability to intervene. To add more security more buffers could be locked with the drawback of lowering performance even more.

3.5.2 Deadlock Prediction

The previously explained method is conservative due to the fact that it is not able to get any information about the system state until a deadlock happens and therefore has always to be in an alert state by blocking a buffer. This could be changed by analysing the system state and getting information about how critical it is. Depending on the information the system can change into an alert state and can be prepared for possible deadlocks.

There might be a lot of possibilities for variables or combination of variables that return information about the critically of the systems state or how imminent a deadlock is. But these variables and the evaluation can change with how the system is designed, making it hard to find a general approach.

This method tries to evaluate the chance of a deadlock occurrence directly by reporting if a deadlock is happening in any future steps. It can be categorized as a pre-emptive deadlock detection strategy, with the advantage that different strategies for the recovery/avoidance part can be applied. Only the deadlock definition and the algorithm to detect the deadlock has to be defined.

The procedure is as follows: at every time step in the main simulation, a different simulation with the exact same state is started and runs for a specified time. How long it is run, mainly depends on what is needed for further policies and the computational limitation, since a longer calculation means more processing effort. This simulation is called “prediction simulation”. If the prediction simulation ever runs into a deadlock it reports back at which time it occurred. Since there are stochastic elements involved, a single simulation is most likely not representative and wrong, therefore this step is done multiple times to get a deadlock probability distribution. This distribution returns how likely a deadlock will occur within a specific time window. How often the predictive simulation has to be carried out depends who detailed and preceise the distribution hast to be which is also influenced by the stochstic elements, and again on the computational limitations.

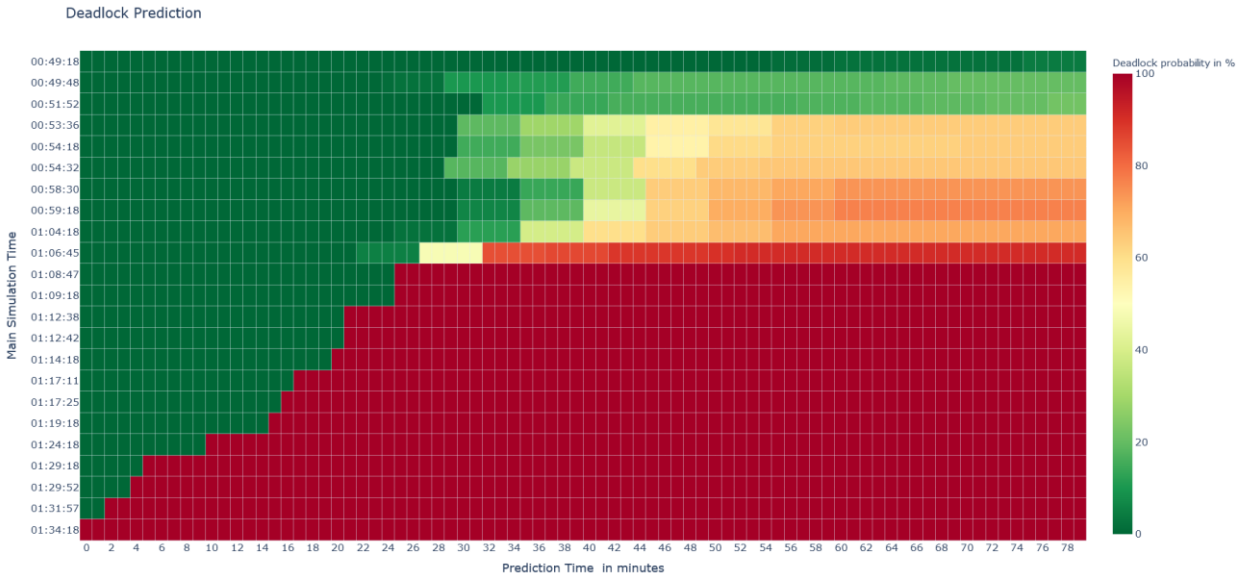

Figure 19: Deadlock Prediction Heatmap

The heatmap in Figure19 shows a possible output of this method. The y-axis represents the time of the simulation, starting at the top. Every point in time of the main simulation is a starting point for the prediction simulation which calculates into the future shown on the x-axis. At the start the deadlock probability is still close to 0 for the next 80 minutes. But the next step in the main simulation already changes the system in a way that the deadlock probability raises to around 10% in the next 28 minutes and about 30% in the next 44 minutes. The deadlock probability mainly rises constantly as the main simulation progresses in time. In some cases, also a probability decrease can be seen. This may be caused by random events that make deadlock less likely. But this can also happen the other way around where the deadlock probability suddenly increases a lot until the deadlock is very likely and only a matter of time since all prediction simulations run into a deadlock. It as to be noted that even though the deadlock probability can change and vary, the predicted time for the deadlock progresses constantly which makes the system still useful. To guarantee this behaviour the main simulation time progression intervals should be small to get constantly information about the evolution of the prediction. This example shows the prediction of a system with many different possibilities for a deadlock to occur. If the system is more complex it would be also possible to make different prediction graphs for distinct section in the system.

With this information the real simulation can be influenced to behave in a different way or switch into an alert state.

3.5.3 Combinations with the Deadlock Prediction Strategy

As already mentioned, this method is only a pre-emptive detection that needs to be paired with other methods to either recovery or avoid deadlocks. Two important values have to be defined. First, how high the predicted probabilities are for a deadlock to occur and second, at which point in time the probability reaches this limit. This depends on how fast the system can get into a critical state. In some systems it might be easier to predict deadlocks a long time in advance and in other systems deadlock occurrence is volatile and its occurrence cannot be predicted a long time in advance.

3.5.4 Deadlock Recovery Combination

An example would be to pair the deadlock prediction strategy with the “locked-buffer strategy”, explained before. This way, performance could be improved by only blocking a buffer if a deadlock is imminent. In general, it is recommended to set the system in an alert state if the possibility of a deadlock is high and the deadlock is upcoming in the near future. By this, performance will not be worsened until shortly before the deadlock.

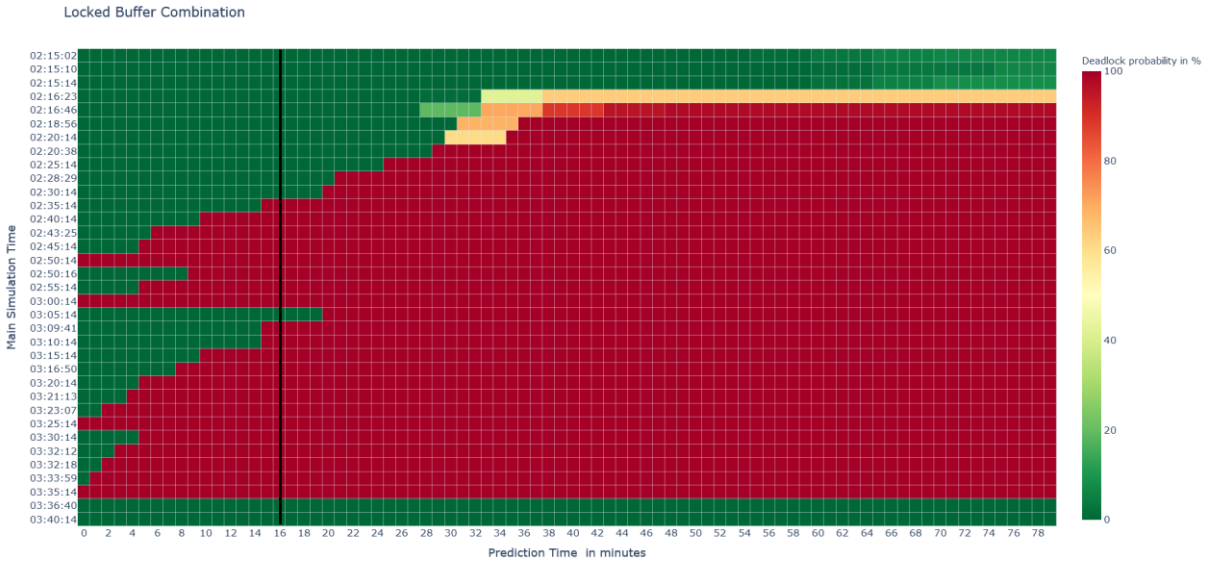

Figure 20: Deadlock Prediction Heatmap, Combined with Locked-Buffer Strategy

In Figure 20 the effect of this method is shown. This example starts at 2:15 h into the main simulation. At first, nearly no deadlocks are predicted. This changes in the next few minutes and at 2:16:23 all prediction simulations report back a deadlock. When the main simulation reached 2:20:38 not only the occurrence of a deadlock is very likely but also the time of its occurrence. The locked buffer policy is activated by locking a buffer as soon as the probability is above 50 percent in 16 minutes into the future, marked by the black line. Policies have to be adjusted depending on what the prediction simulation reports back. In this case this could be the deadlock location. It can be seen that the system state does not change until the deadlock occurs but is than recovering from it. In some cases, a new deadlock state is imminent only a few minutes after the recovery but also a full recovery is possible as seen at the bottom of the heatmap. It has to be noted that the prediction simulations have no recovery policies implemented and are therefore reporting back what the outcome of the main simulation would be without recovery policy.

3.5.5 Policy Change Combination

Another, more expandable combination is to change the system behaviour through policies and rules in advance, to resolve possible imminent critical states and deadlocks early. This would allow a smoother deadlock control.

A possible intervention could be to change the dispatching of products through prioritisation. Since the prediction simulations do not only get information about when, but also how and where, deadlocks are likely to occur, the prioritisation can be applied to the right products. In general, it is wise to change policies much more in advance compared to the “locked-buffer”

strategy, since the interventions effects should have less impact but require time to influence the system.

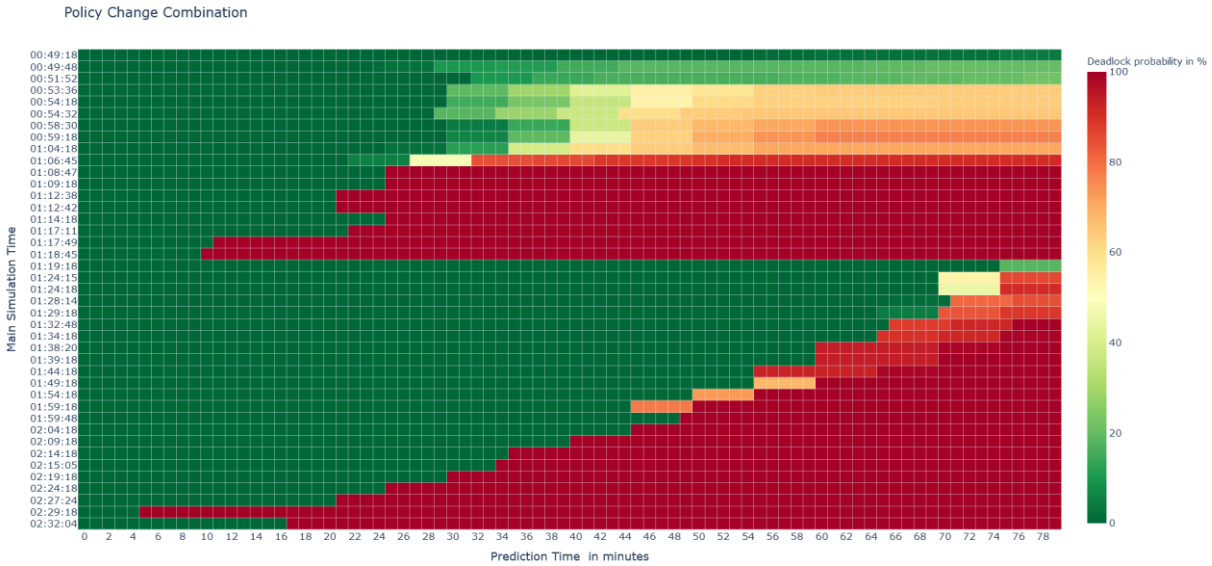

Figure 21: Deadlock Prediction Heatmap, Combined with Policy Change Strategy

In Figure 21 the deadlock probability is raising and therefore a policy change is applied. In this example the products which happen to be most often in a deadlock are prioritised. The intent is to get products that are causing critical states out of the system as fast as possible. In some cases, the effect can be seen by a delay of an imminent deadlock. But also, nearly full resolutions of critical states are possible. Even though policy changes are active, the influence can sometimes only be seen after a while, since the changes influence the system not always instantly. It can also happen, that multiple changes are applied shortly after another, changing the prediction multiple times. This deadlock prediction again, shows only the development of the system without policy changes, which means that changes are only shown if already happened in the main simulation.

Other possible policy changes could be to block the input of new products into the system or into different sections and therefore keep the used capacity low. A less intervening policy would be to change the performance of a machine or multiple machines to relive critical sections. Or to change the routing of products so that they take less critical paths if variation is possible. Another option is to change the incoming products in a way that critical sections will be used less until the deadlock probability is low again.

For most of these policies it is easy to find out which values are needed to be changed to achieve a certain improvement. In some cases, it would also be possible to take an analytical approach and start a set of prediction simulations with different policy changes to than choose the most beneficial one. In the end also the buzzword machine learning should be mentioned which could be used to find an optimized set of variables and their values.

Since the predicted simulations possesses the same variables as the real simulation a lot more of information could be extracted and more complex policies and rules can be applied. The system state could be divided into different alert states depending on when and how likely a deadlock is to occur (Figure 22). Depending on in which state the system is, more intervening

policies could be applied. There are many possibilities, and all have to be adjusted to a specific system for the full potential.

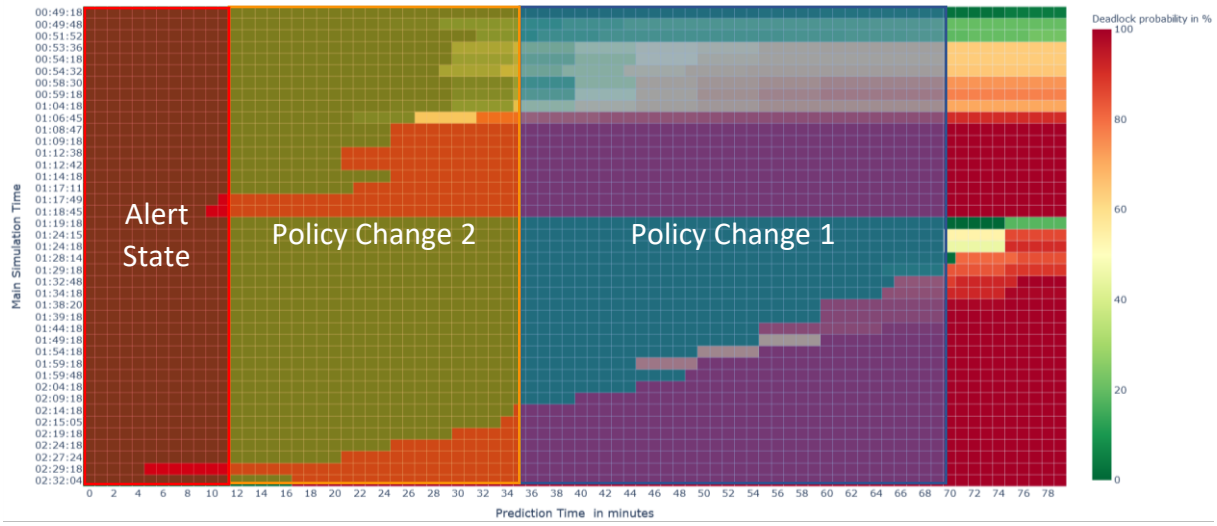

Figure 22: Deadlock Prediction Heatmap, Alert States

4 Case Study

The following case study analyses two layouts of a production system in two different scenarios and the effect of the proposed deadlock control methods with the use of a simulation, seen in Figure 23. Deadlock control methods that will be used are:

- Method 1: locked buffer – deadlock recovery strategy
- Method 2: deadlock prediction combined with the locked buffer strategy
- Method 3: deadlock prediction combined with a policy change
- Method 4: method 2 and method 3 together

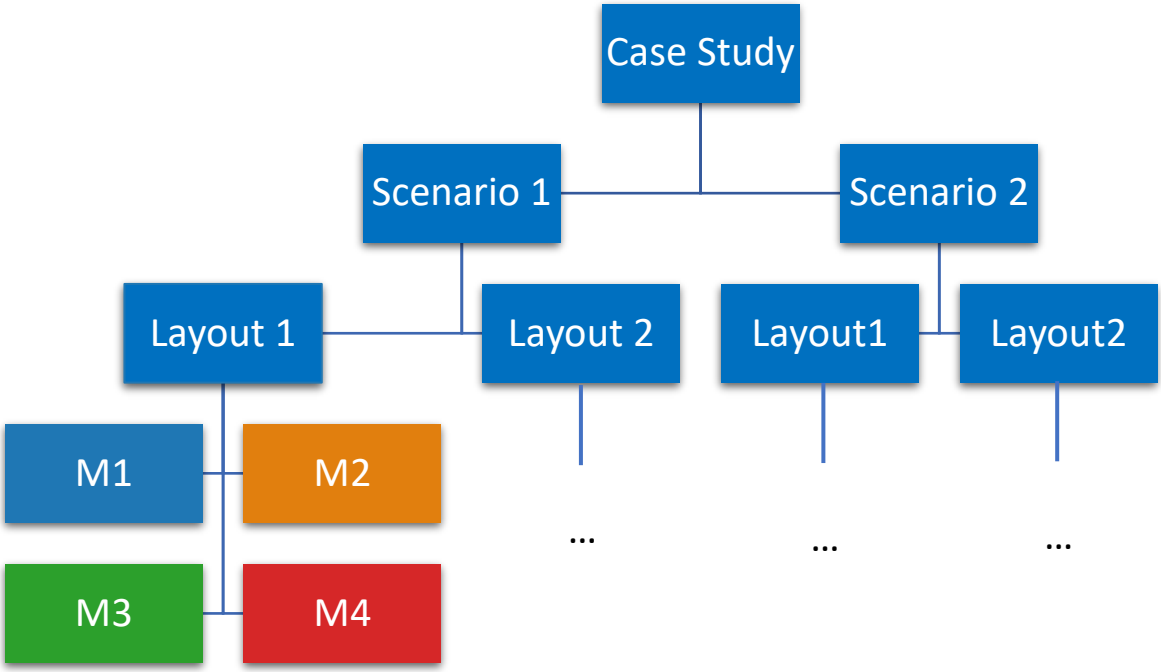

Figure 23: Case Study Scenarios

The effect of the methods on the production system will be tested on two different scenarios. The first one will focus on the ability to handle deadlocks. The design of the system is chosen in such a way that deadlock occur relatively frequently. The simulation is carried out until a deadlock occurs. This scenario will give information on how good the methods can handle many, repeatedly occurring deadlocks. The second scenario focuses on the performance of the deadlock control methods with respect to a more stable system design, where deadlocks will occur less frequent and the simulation only runs for a certain time.

4.1 Output values

Key measures for comparison are the deadlock time, parts per deadlock and average throughput time of the system. The deadlock time represents how long the system is running until it is in a deadlock state which cannot be recovered anymore. Parts per deadlock, giving information about how many parts got finished until the unrecoverable deadlock occurred. Through the combination of these two values the average throughput time of the system can

be calculated. To get a significant comparison, each combination of test setups will be done 200 times. This is also needed because of the stochastic behaviour of the system.

For scenario 2 an additional measure is added, which will be named no deadlock percentage. Since the simulation in scenario 2 only runs for a certain amount of time, every test run that reaches this time is marked as “no deadlock”.

4.2 Entities

The systems entities are products, routes, machines and buffers. In the following part the most important attributes and their functions are described. Figure 24 shows all used attributes in this case study. It as to be noted that the product is connected with the route through the product type and the machine has an attribute for buffer referencing.

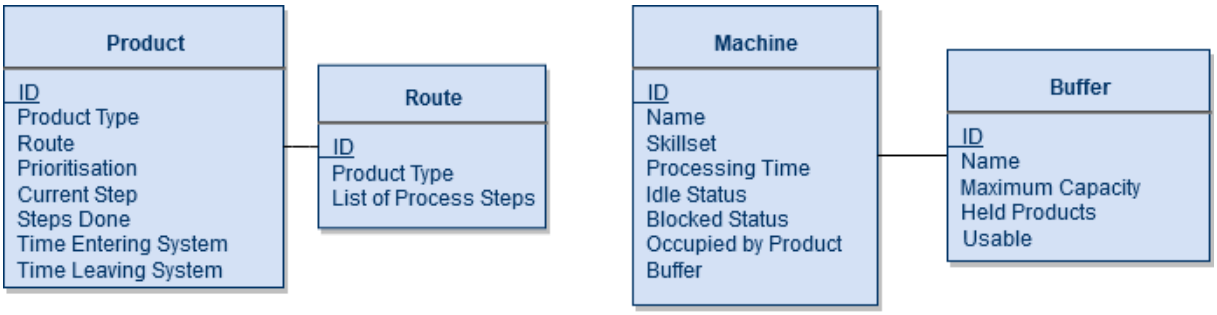

Figure 24: Entities and their Attributes

4.2.1 Products and Routes

In this case study there are three different product types, determining what stages the product must proceed along the route. Each route is a list of processes or required skills that can be fulfilled by the machines. The three process routes shown in Figure 25 for layout 1 are:

- Route A: In → A → B → D → Out
- Route B: In → A → F → B → C → A → Out
- Route C: In → A → B → D → E → C → B → D → Out

Route A is characterized as a line and therefore no circles, getting processed by three machines. Route B forms a circle and needs to use machine A two times. On Route C, products get processed by seven machines of which 2 machines, B and D are used twice and forms a circle too like route B.

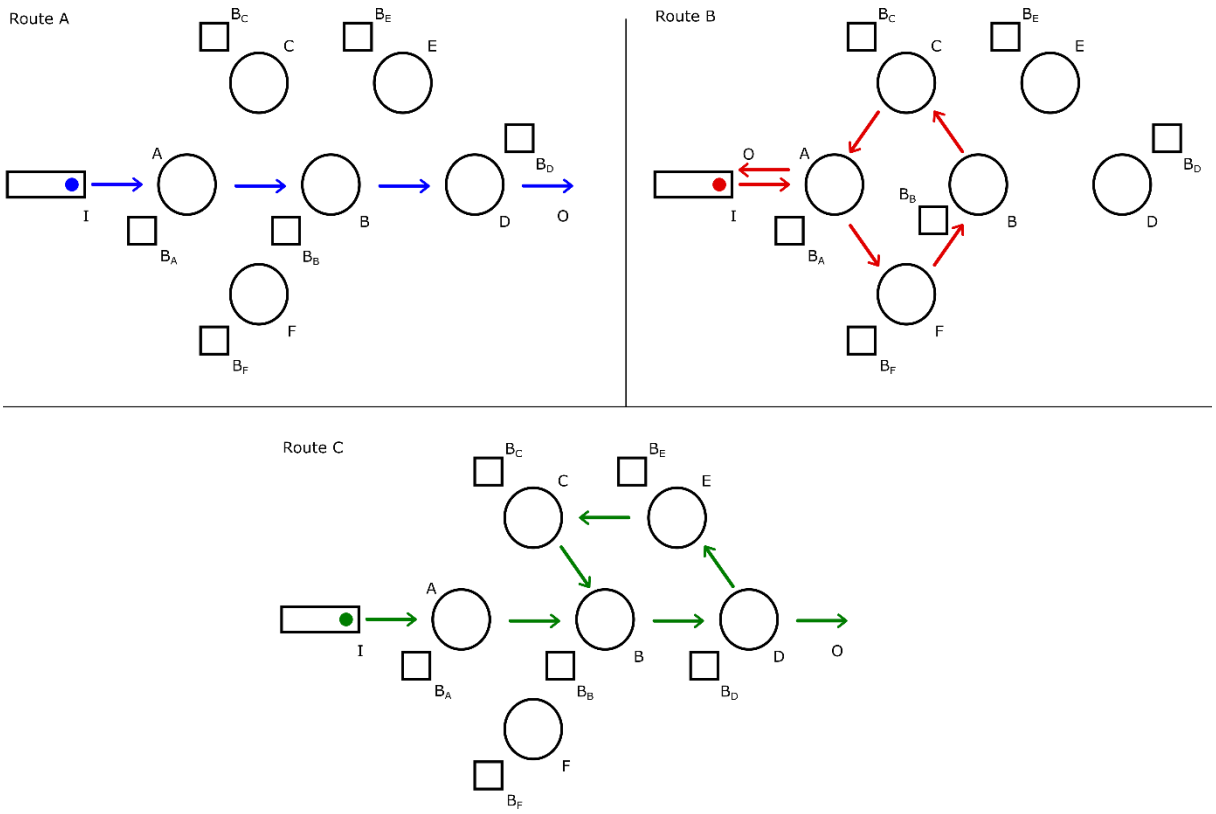

Figure 25: Route of the Products for Layout 1

For layout 2, machine G is added to extend the route of product type A, leading to another circuit for deadlocks to occur. This will be discussed in more detail in the deadlock section.

- Route A: In → A → B → D → E → G → D → Out

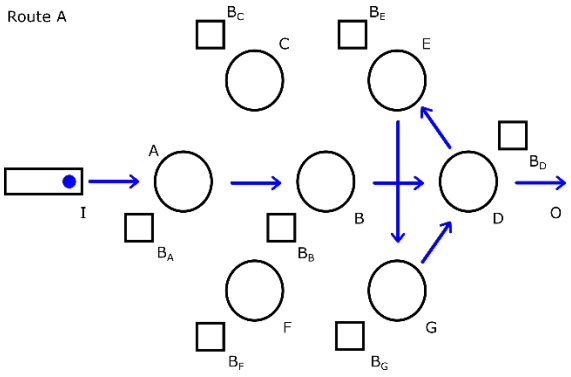

Figure 26: Route of Product A for Layout 2

For each scenario four different product mixes are used. The change of the product frequency distribution influences the likelihood of deadlock occurrence and the affected machines – where the deadlocks occur. Products of type A with route A in layout 1 have the shortest route and are therefore the shortest amount of time in the system. They also move in a line and therefore without a circuit, through the system. This makes product type A the safest product and the least critical for deadlock causing. For this reason, the frequency of product A is getting

progressively lowered and the frequency of product B and C are increased for each product mix, to increase the probability for the system to run into a deadlock state. In layout 2, products of type A with route A increase the possibility for a new deadlock at an additional place. The product mixes used are:

- 50% A, 25% B, 25% C
- 40% A, 30% B, 30% C
- 33% A, 33% B, 33% C
- 24% A, 38% B, 38% C

Besides the product type and the route, products carry information about their priority. All products are by default prioritized equally. The priority of products can be changed to influence the system flow. This will only be done in deadlock control method 3 and 4.

4.2.2 Machines

Each machine is defined by a skill or a set of skills that determines which process step can be carried out on it. As can be seen by the routing, in this case study, each machine has only a single skill to offer. This can be checked when products search for routes to find the next machine. Depending on what skill a machine has, different processing times are set. For this case study, either ten minutes for machine C and G or five minutes for all other machines are used.

Machines also have information about by which product they are currently occupied. If there are currently no products in the machine it is set to idle and awaits a product to process. As long as there is a product occupying the machine, it is not idle even if it is not processing. In this case the product cannot move forward because it is blocked, as already explained before, setting the machine also to blocked. These two properties define all possible states of a machine. The idle property for the product flow, the blocked property for deadlock detection.

Machines also have a buffer assigned, so that, if a product cannot get into the machine, it can be send to this buffer that is in front of the machine.

4.2.3 Buffer

Each buffer has a capacity to define how many products can be stored in it. For simplification reasons, in this case study the capacity of all buffers is assumed to be one. One exception is the overflow buffer that is located at the start of the system which has no capacity limit. It is used to store the arriving products before they move to the first station. Buffers also have information about which product is currently stored in it. To enable Method 1 and 2 there is also a property that sets the buffer temporarily to being not available.

4.3 Definition of the System

Now that the entities are described, the use of the system and the underlying assumptions will be explained. Again, no transportation device like automated guided vehicles are included. If products change their location, they will immediately. This simplification is based on the assumption that transport does not have any major effects on the system or especially for what is tested and reduces the complexity that makes focusing on the problem easier.

This case study does also not consider random breakdowns of any resource, even though it could be implemented easily the effects of breakdowns will not be analysed here.

4.3.1 Influence of Variables and System Behaviour

In this section thoughts are made on how the system behaviour changes if certain variables are changed. This is only done to see how this systems dynamic could be changed in during the definition phase, since these variables stay constant afterwards. Some variables were already mentioned before since they are attributes of entities.

Another variable that is defined by the system and not by an entity is the inter arrival time of products. It is the time between two product arrivals. The inter arrival time is an exponential distribution ($x = t_{Ar} * log_e(1 - y)$) with the centre t_{Ar} at 10 minutes for first scenario and 15 minutes for the second. The inter arrival time influences the system in a way that, if all other variables are constant and the inter arrival time is decreased, more products will fill the system. Therefore, blocking of machines will happen more often and since more products are in the system, the state of the system is more critical. This is because the maximum capacity is more likely to be reached.

The same behaviour can be seen in combination with the performance or the utilization of machines and if the processing time of machines gets longer. The system gets more frequently into a deadlock state if the throughput is lower, since there are more products in the system. On the other hand, if the performance is higher and hence the throughput is higher, it is less likely that the system gets into a deadlock state.

Another attribute of the buffer that influences the systems behaviour is the maximum capacity. Increasing the buffer capacity can help making deadlock prediction easier because the systems state is more differentiated and can be categorized in safe, alert and other states.

The only stochastic variables are the inter arrival time and the product mix, hence the type of products arriving. All other variables, particularly mentioned the processing time of machines, are discrete and deterministic.

4.3.2 Possible Deadlocks

If the route of the products is investigated, see Figure 25, three possible circles can be identified. The first circle can occur with 3 machines involved or with 4 machines involved. Since the circle is the same and only the numbers of involved machines change this can be seen as one circle. The circles are:

- A → (F) → B → C → A
- B ↔ C
- B → D → E → C → B

In layout 2 an additional circle can be found:

- D → G → E → D

Special notice should be given to the circle where only two machines are involved. In this case the maximum capacity of the circle can be reached quickly, and the deadlock prediction gets difficult. Also, there is a lot of traffic for the machines A and B, making the circles where these

machines are involved easier to get into a deadlock, since they are most of the time occupied and the capacity is therefore often closer to being reached.

4.4 Hierarchical Control Structure

The structure of the model is rather simple, since the functionality and complexity are kept low. The root control unit is responsible for the whole production control, for handling requests, moving products and therefore the state changes through a rule set but also the control methods are executed by it. The detailed procedure of the production control will be explained later. The products move through the system by requesting the next resource. By implementing the prediction simulation, the production control unit takes responsibility for the prediction control unit, which is mainly a copy of the production control.

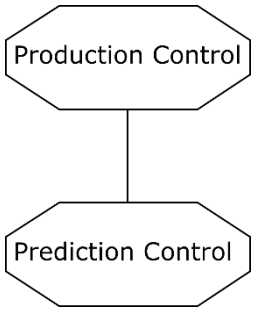

Figure 27: Hierarchical Control Structure

4.5 Procedure of the Simulation

The Simulation uses the HCCM concept described in chapter “Hierarchical Control Conceptual Model”. Since the time advancement has already been explained there, this section will focus on the main control rules, the requests and activities which are specific for this case study. First, only the regular flow of products through the system will be explained and after that, the deadlock control methods will be added.

4.5.1 Requests and Activities

The simulation consists of two different requests and 2 different activities. The activities are “get machined” and “wait in machine”. In the get machined activity the product is processed by the machine with the required skill, according to the route of the product. As soon as the activity starts the machines is set to not idle and stores the information by which product it is occupied. The product stores its current processing step and an end event is scheduled that determines the end of the activity, depending on the processing time of the machine. When the end event is triggered the product searches for the next required skill by going through its route. If there are no more needed steps and the product is finished, it moves out of the system and the machine is set to idle again. If there are still steps to be performed, the “wait in machine” activity is carried out. This behaviour can later also be used to identify states of the system where a product is waiting in a machine and to distinguished blocked machines from others. If the product can get moved out of the machine in the same timestep, which as explained before happens after the events are triggered, the practical waiting time is zero. This happens in the machine request handling procedure if an idle machine is found and the product is currently in a machine. Nevertheless, the machine is first set to blocked and is still not idle. A request for the next machine with the needed skill is filed and the activity is carried out until the product is moved. At the end of the waiting activity, when the product gets moved, the machine is set to idle. Further, the machine is no more blocked and the occupied by product property is cleared.

The requests are to get placed into a machine and to get placed into a buffer. There is a special case for each of them during the arrival of a product. In this starting phase, both requests are predetermined and always request the same. Required information to file a request are the needed skill, the product filing it and the time of the request.

For better understanding of this behaviour, Figure 28 shows a simplified flowchart from a product perspective with the focus on activities and requests. Requests are marked in blue, while activities are marked in green. It has to be mentioned that a product stays in the waiting activity as long as it gets placed somewhere else.

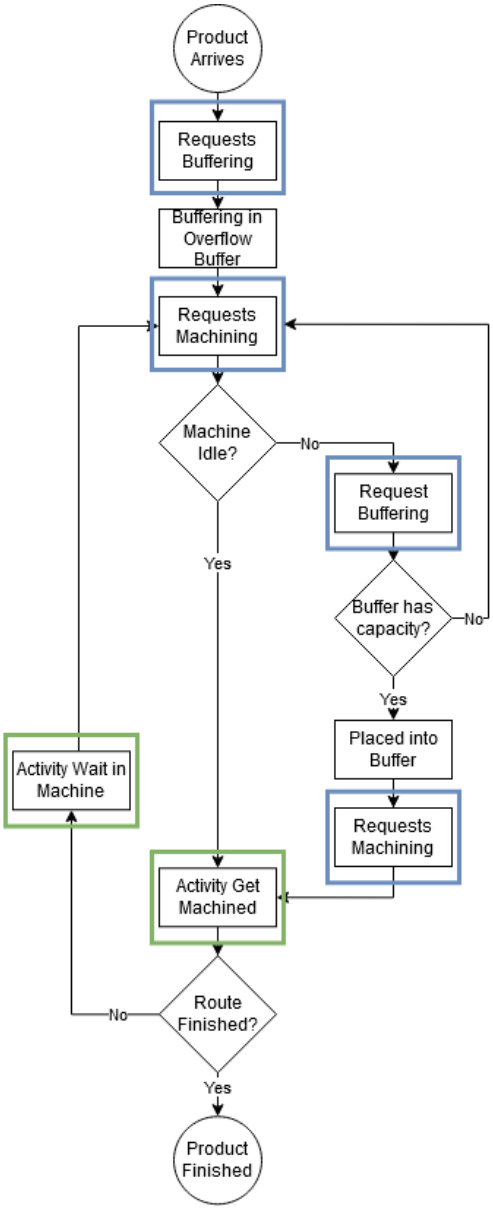

Figure 28: Flowchart of Requests and Activities

4.5.2 Initialization and Start

At first, the simulation, all its entities and the starting state are initialized. The starting state establishes the flow of products by creating the first product and scheduling the first product arrival event. After the time advances to that point the product arrival event is triggered, and the main control rules are executed (Figure 29). Each time a product arrives a new product is generated, and its arrival is scheduled, this way the simulation keeps on running until a stop condition is satisfied. The needed information to generate a product is the probability distribution of products from the product mix. To generate the next product arrival event the time of arrival is needed. The time is the current time plus the before mentioned exponential distribution of the simulation's inter arrival time. After scheduling the new product arrival, the currently arriving product is handled by making a request. The first Request of an arriving product is always to get placed in a buffer with no machine in front, which is used as an overflow buffer if products cannot be placed into the system.

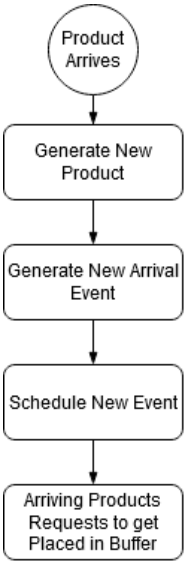

Figure 29: Initialization and Start Procedure

4.5.3 Basic Overview of the Main Control

The next procedure that will be explained are the main control rules of the system, seen in the flowchart (Figure 30), where products are moved through the system. This will be executed after every triggered event. The intention is to give a basic overview and focus on details later.

The main control starts by searching the request list for the requests where products want to get placed in the overflow buffer. The first one is picked and will be placed into the overflow buffer. The request is fulfilled and can be removed. To conclude the overflow buffer request handling, the next request for the next process is filed. The details to this will be explained later. This part will be repeated until there are no more requests for placing a product into the overflow buffer.

The procedure is continued by sorting the request list. The first sorting is done by product priority. As mentioned before the product priority is by default the same for every product

and gets only changed by deadlock control algorithms. Then the products are sorted by request time to pick the ones which waited the longest for their request to get accepted. The last attribute by which is sorted is the product arrival time, to move products that where for a longer time than others in the system. After the sorting of requests, the machining requests are handled. Here, products are assigned to machines. This will be explained in detail later.

The next step is then, to again sort the request list. This time, the buffering requests and with the same policies as described before. The buffer requests will then be handled, and products are assigned to the buffers of the machines to which they would have requested to move but could not. The last step is to check if any change as happened, which means that any products were moved. If so, the procedure starts again from sorting the machine requests. This is done, since if a product moves into a buffer, the possibility exists that a machine is again idle, and a product could move again. If the steps are done and no change has happened it is certain that all possible assignments have been done.

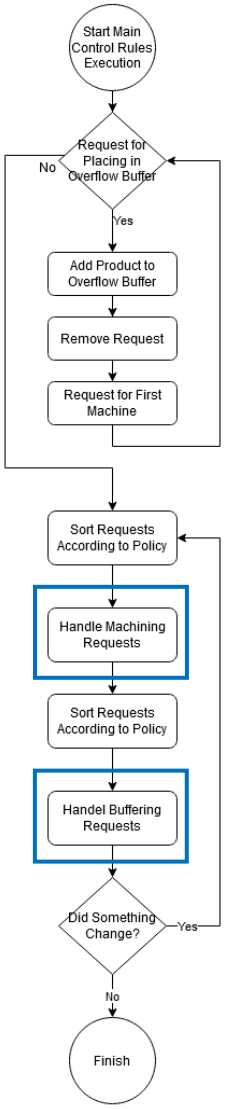

Figure 30: Main Control Procedure

4.5.4 Handling Machine Requests

This concludes the basic procedure of the main control. More detailed things are happening during the request handling of machining and buffering requests, which will be explained here. The steps performed can be seen in the flowchart in Figure 31. The first request from the sorted request list is picked and a machine that can fulfil the needed skill is searched. The machine is checked for being idle. If not, this request is skipped and the next one from the request list is taken. If the machine is idle, the product can be placed in it. For this to happen it needs to be checked where the product is currently located. If it is in a buffer, the product is removed from it. Should the product be currently in a machine it is in a waiting activity until it is moved. In this case the end event of the activity is triggered. The product is now located in the next machine and the start event of the machining activity is triggered. Finally, the request can be removed. It is important to start picking requests of the request list from the first position again, also the skipped ones. This is because, since a product has changed location the possibility exists that a machine, that was occupied before is now free. Products with higher priority, the ones who are first in the request list should be preferred. These steps are done until no requests are left in the list or every request in the list has been skipped. If there are skipped request in the list, they are changed into buffering request, since products that cannot get into machines try to get into the buffers in front of the needed machine.

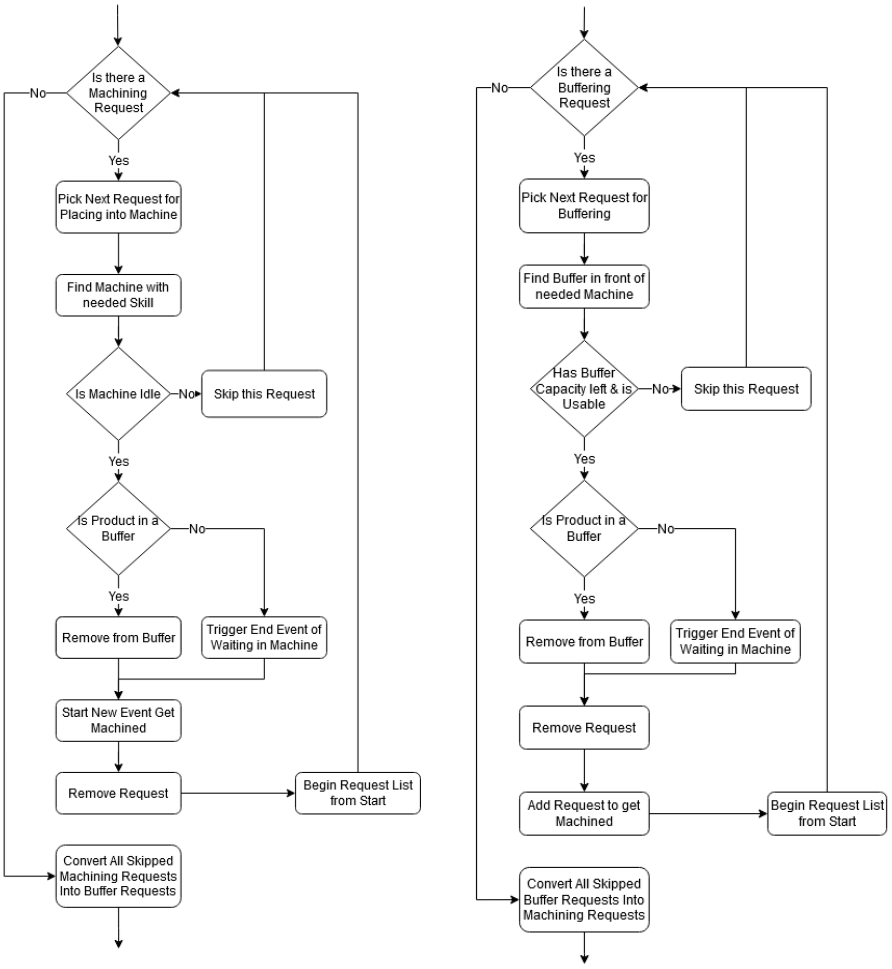

Figure 31: Handling of Machining and Buffering Requests

4.5.5 Handling Buffering Requests

The handling of the buffering requests is nearly the same as the handling of machining requests, as can be seen in Figure 31. A difference is the checking for resource availability. While the machine is checked for the status idle, the buffer is checked for available capacity and if the buffer is allowed to be used (depending on later used deadlock control policies). In the last step the buffering requests are again converted back into machining requests, since this procedure is most likely to be repeated. It has to be noted that, even though the requests are changed from requesting to get machined to requesting to get buffered the time of when the request was filed stays the same, which is the time when first filed. This applies also to for the next time steps until one of these requests gets accepted. Otherwise the prioritisation would be mixed up. In other words, the requested time of a buffering request is the same as for the machining request when converted and vice versa.

4.5.6 Deadlock Detection

The deadlock detection follows after the main control rules and executed right afterwards. After the system state has changed and all products were moved, it is checked if there are any sections in deadlock. The deadlock condition before explained is used, where if a set of blocking machines is a subset of all blocked machines, a deadlock is guaranteed. Since the machines already have information about if they are blocked through the waiting state, only the blocking machines have to be identified. Since the blocked machine is blocked because the product cannot move, the route of the product can be checked to see where the product would like to go and therefore find the blocking machine. This is done for all blocked machines to find all blocking machines. If the deadlock condition is true a deadlock is identified.

4.6 Deadlock Control Methods

The procedure until now concludes the basic function of the system and how the production model is simulated. It shows how products enter, how they are moved while fulfilling their processing routes and, in the end can exit the production as finished products. Deadlocks can be detected but no deadlock handling has been included yet. In this next section, the procedures of deadlock handling will be added. The locked buffed strategy will be explained first and after that the prediction strategy. This second one will then be extended with a combination with the locked buffer strategy and then a combination with a policy change.

4.6.1 Locked Buffer

As mentioned before the locked buffer strategy works by locking at least one resource and set it free in the case of a deadlock. In this case study two buffer are locked in layout 1, which are buffer B_A and B_B. This decision has been made, since machine B is involved in all circles and machine A has second most products to process. For layout 2, buffer B_D is additionality locked because the other machines are not involved in this second location for deadlock occurrence.

The example in Figure 32 shows on the left side a deadlock. As soon as the deadlock is detected Buffer B_B is set free and the Product marked in blue can move. Now a next step is possible by moving product from machine C and therefore the deadlock is recovered.

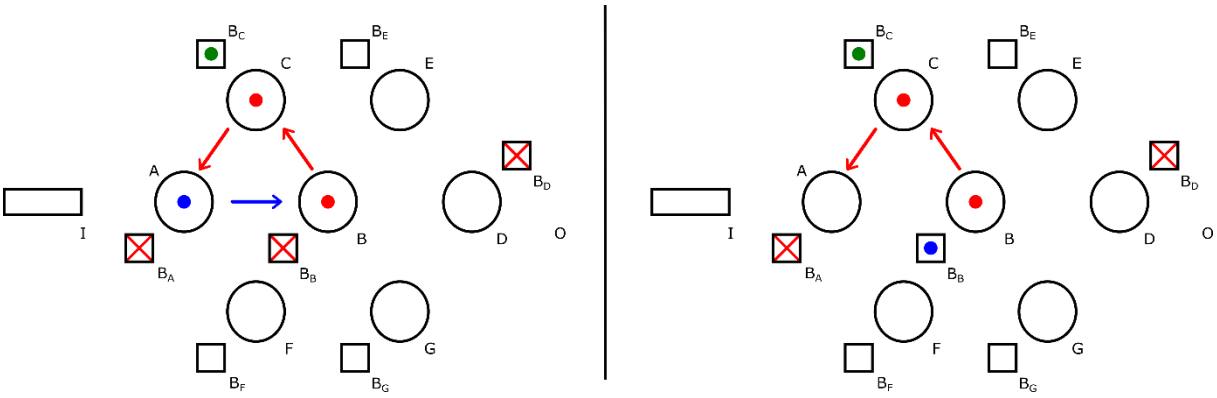

Figure 32: Locking of Buffers

The procedure can be seen in Figure 33 and is as follows. At every execution of the main control rules the mentioned buffers are locked by setting them to not usable. This is also important for relocking buffers that were set free in the step before, after a recovery where the flow of products got re-established. Next, the main control rules and deadlock detection is executed like explained before, but with the difference that requests for buffers can sometime be not fulfilled, since there may be capacity left but the buffers are set to be not usable. If a deadlock is detected this procedure starts from again from the start. This repetition is done since two buffers can be set free. In the first loop buffer B_B is unlocked and the main control procedure is repeated, this time a product can most likely be placed in this buffer. If not, the next buffer B_A is set free and the procedure is repeated. This does not always guarantee a deadlock recovery since if all buffers are set free and a deadlock is still detected no further measures can be taken to resolve the situation.

For layout 2 an additional step is performed during deadlock detection, where it is checked which machines are involved in a deadlock to determine the deadlocked circle. Depending on where the deadlock occurred, the corresponding buffers are set free. In case of the layout 2 deadlock, this would be buffer B_D.

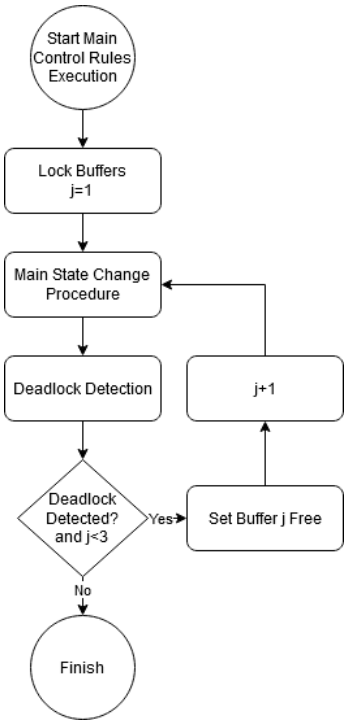

Figure 33: Flowchart of Locked Buffer Strategy

4.6.2 Prediction Simulation

The underlaying principle of the prediction simulation is to run a copy of the main simulation as the first procedure of the main control rules. Through this, decision can be made that influence the following steps to avoided deadlocks. Figure34 shows the procedure. Every time a main control rules execution is triggered a prediction simulation is started. It is an exact copy of the main simulation in its current state. All entities and their properties, scheduled events, currently running activities and filed requests are replicated. Special care has to be taken in copying currently running activities, since the time they have already been executed must be considered. The prediction simulation proceeds in the same way as the main simulation would and until a deadlock is detected or it exceeds a simulation time of 80 minutes. The prediction simulation will most likely have a different outcome as the main simulation would have. This is because the chances are high, that a new or several new products will arrive, which have probability involved since the new products are not predetermined but depend on the product mix and therefore are not the same as in the main simulation. Due to this it is possible that the flow and outcome of the prediction simulation is different from the main simulation. The probability, which product will be generated is influenced by the product mix of the prediction simulation, that is the same as in the main simulation. If a deadlock happened the time of its occurrence is reported back to the main simulation and put into a cluster divided into 2-minute time steps. A new prediction simulation is then started until 150 reports are made. Through this cluster the deadlock probability can be estimated.

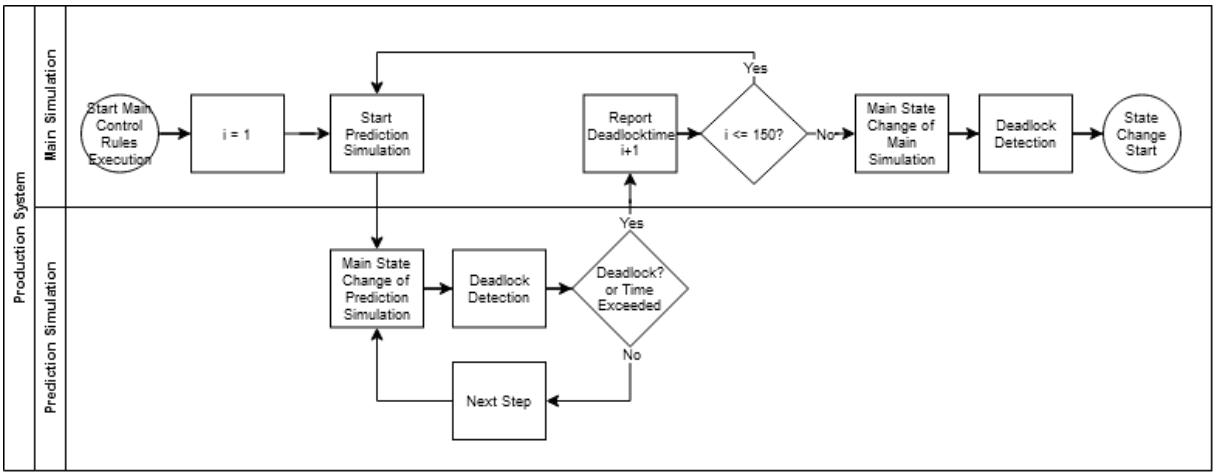

Figure 34: Flowchart of Prediction Simulation Strategy

Table 3 shows a graphical representation of such a cluster. On the left side, top to bottom is the main simulation time and on the top are the 2-minute time intervals. The two highlighted cells in red can be interpreted as follows. At 6:00:27, looking 20 minutes into the future a deadlock will happen with a likelihood of 6 %. Looking 24 minutes into the future the likelihood of a deadlock is 16 %. The explained steps conclude the basic function of the prediction simulation strategy on which two deadlock control methods will be build.

2 Minute Interval Main Simulation Time	2	4	6	8	10	12	14	16	18	20	22	24	26	28	…	80
05:53:28	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
05:55:32	0	0	0	0	0	0	0	0	0	0	0	0	1	1		4
05:58:28	0	0	0	0	0	0	0	0	0	0	0	2	2	2		5
06:00:27	0	0	0	0	0	0	0	0	0	0	6	6	16	16		26
06:00:32	0	0	0	0	0	0	0	0	0	0	6	6	12	12		20
06:00:46	0	0	0	0	0	0	0	0	0	4	4	4	8	8		30
06:03:28	0	0	0	0	0	0	0	0	11	11	11	11	11	58	…	58
06:05:32	0	0	0	0	0	100	100	100	100	100	100	100	100	100		100
06:07:39	0	0	0	100	100	100	100	100	100	100	100	100	100	100		100
06:08:28	0	0	0	100	100	100	100	100	100	100	100	100	100	100		100
06:09:05	0	0	0	100	100	100	100	100	100	100	100	100	100	100		100
06:10:32	0	0	100	100	100	100	100	100	100	100	100	100	100	100		100
06:15:32	100	100	100	100	100	100	100	100	100	100	100	100	100	100		100

Table 3: Deadlock Probability Cluster

4.6.3 Deadlock Prediction Combination with Locked Buffer Method

The combination of the locked buffer strategy with the deadlock prediction has the advantage of not having to lock a buffer all the time but only if a critical state is reached. This critical state is detected by a deadlock probability higher than 50 % in the next nine minutes. The deadlock probability is the how likely a deadlock occurs at a certain point in time. Until this critical state has been reached, the main simulation is executed as if there is no deadlock control method implemented. If the critical state is reached it behaves like the locked buffer strategy explained above. It is noted that for layout 2, also not just the criticality of the system is reported but also where the deadlocks are predicted, this way a buffer in circle one, circle two or in both circles is locked. The main intend of this method is, to see if a deadlock prediction can be used to improve performance and to make the system more deadlock safe.

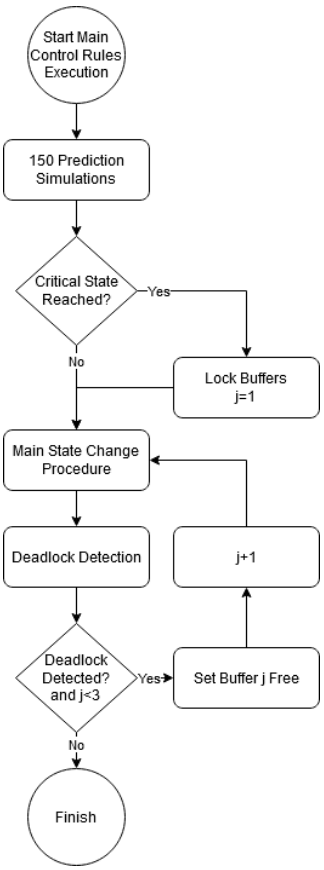

Figure 35: Flowchart of Deadlock Prediction Combined with Locked Buffer Strategy

4.6.4 Deadlock Prediction Combination with Policy Change

This method changes the flow of the main simulation by prioritizing products which are the reason for deadlocks to get them out of the circles faster. The prioritization starts as soon as the deadlock probability is higher than 50 % in the next 22 minutes. Besides the deadlock time the prediction simulations also generate a list, ranking the products that where most often in a deadlock. The two highest ranked receive a higher prioritization. As explained in the main control rules procedure, the prioritization changes the order of requests to let these products move before all others.

5 Results

For each method 200 replications were made, and the average of each key measure has been calculated. The measures by with which the methods are compared are the deadlock time, the sum of products per deadlock and the average time it takes until a product gets out of the system which is a combination of the first two. The latest one will be called system throughput* since it is not a standard product throughput and is calculated by dividing the deadlock time by the products per deadlock.

The deadlock time is measured in hours, rounded to the first digit and is the time from the start of the simulation until an unrecoverable deadlock. The longer the simulation lasts and the higher the deadlock time is, the better.

But a production system that only lasts long does not have to be the best, since another setup that might run faster into a deadlock could still process more products until then. In this case the products per deadlock gives information about the performance of the production system during its runtime. The products per deadlock is rounded to whole products. The more products get out of a system, until a deadlock occurs, the better.

In some cases, it might be hard to see which of these two values outweighs the other since the values are not normalized and therefore not easy to compare directly. This is the reason for calculating the system throughput*. This value is presented in minutes and rounded to the first digit. If the average time for a product to come out of the system is lower, the better.

For all 200 replicaitons a random set of products was generated according to the product mix beforehand, so that each method gets the same set of products. This has been done to eliminate a possible distortion of the results when comparing.

Deadlock control methods that will be used are:

- Method 1: locked buffer – deadlock recovery strategy
- Method 2: deadlock prediction combined with the locked buffer strategy
- Method 3: deadlock prediction combined with a policy change
- Method 4: method 2 and method 3 together

5.1 Scenario 1 / Layout 1

The first notable point is that the impact of the product mix is strong and makes it easier to get into a deadlock, as was planned. Especially the change from the first to the second product mix influences the system a lot. Looking at the simulation, where the product mix favours product A, method 1 achieves the highest deadlock time. But as soon as the system gets easier into a deadlock, method 2 and 4, which both use the prediction strategy do not fall of as hard as method 1. At the equilibrium of the product mix, method 1 is once again better than method 4 but in the next product mix, method 4 has a higher deadlock time by around two and a half hours. When comparing the methods by the deadlock time, method 2 seems to be the best.

Deadlock Time in h				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	46,3	45,2	17,8	35,3
A:40 % B:30 % C:30 %	18,2	22,5	10,3	19,7
A:33 % B:33 % C:33 %	14,7	17,3	7,2	12,3
A:24 % B:38 % C:38 %	5,5	9,5	4,6	8,1

Table 4: S1-L1 Deadlock Time

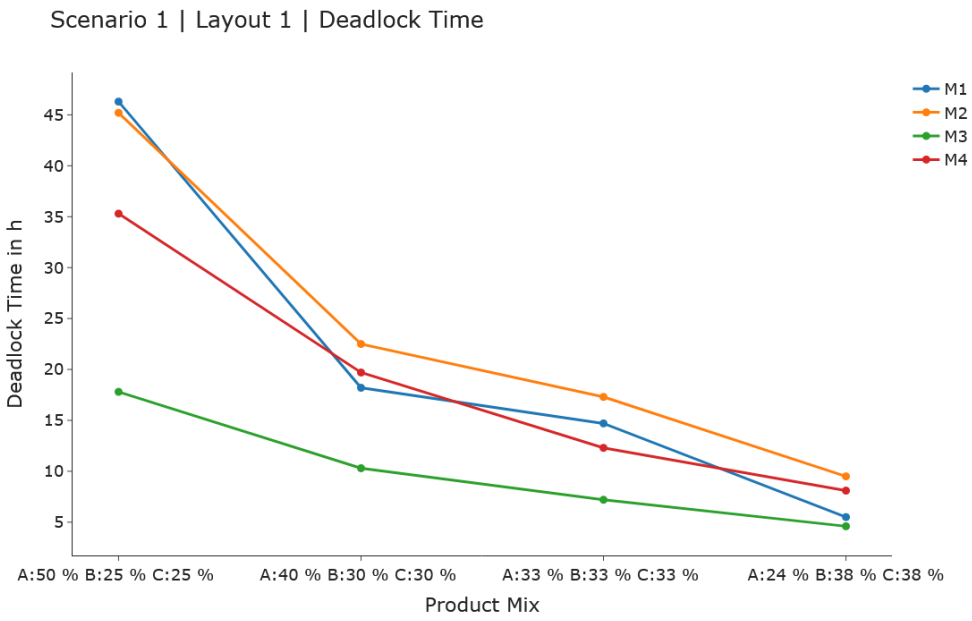

Figure 36: S1-L1 Deadlock Time

The same result can be seen for the products per deadlock. Judging only by the products per deadlock measure, method 2 seems overall superior and method 3 way off. In both measures method 1 falls off the fastest.

Products/Deadlock				
Method Product mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	268	263	100	203
A:40 % B:30 % C:30 %	100	126	54	108
A:33 % B:33 % C:33 %	78	93	35	65
A:24 % B:38 % C:38 %	27	48	21	41

Table 5: S1-L1 Products/Deadlock

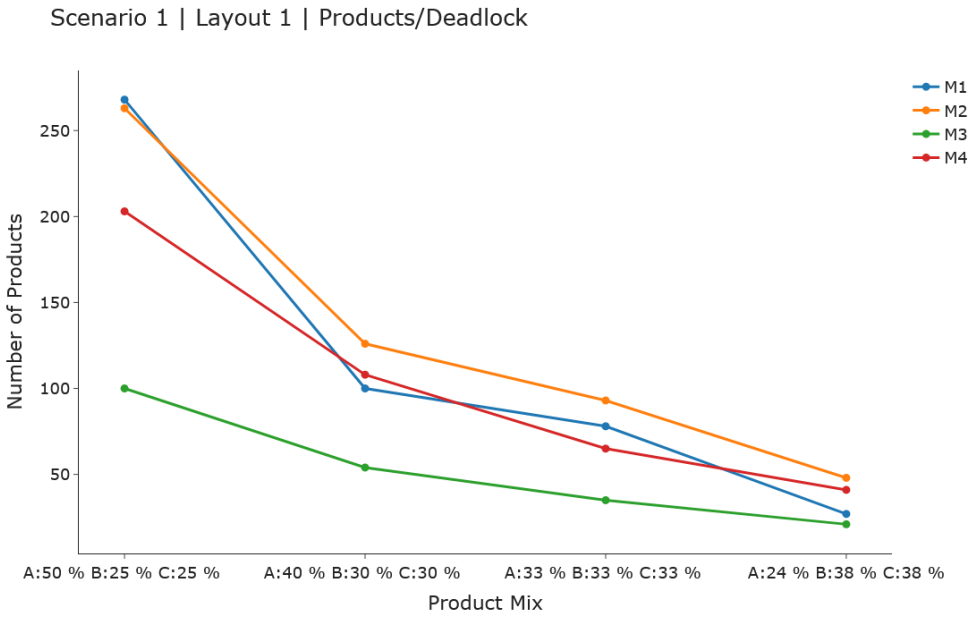

Figure 37: S1-L1 Products/Deadlock

Since the two measured values do show nearly the same results and do not point in opposite directions, the calculated value should and does show the same result. Method 3 is worse than the others. Method 2 seems overall better than Method 1 and 4, even though they are most of the time very close to each other. The difference is below half a minute. Also, of interest is that method 2 excels the others at the hardest product mix, where also method 3 has the worst outcome.

System Throughput* in min				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	10,4	10,3	10,6	10,4
A:40 % B:30 % C:30 %	10,9	10,7	11,4	11,0
A:33 % B:33 % C:33 %	11,2	11,1	12,2	11,4
A:24 % B:38 % C:38 %	12,2	12,0	13,1	11,9

*Table 6: S1-L1 System Throughput**

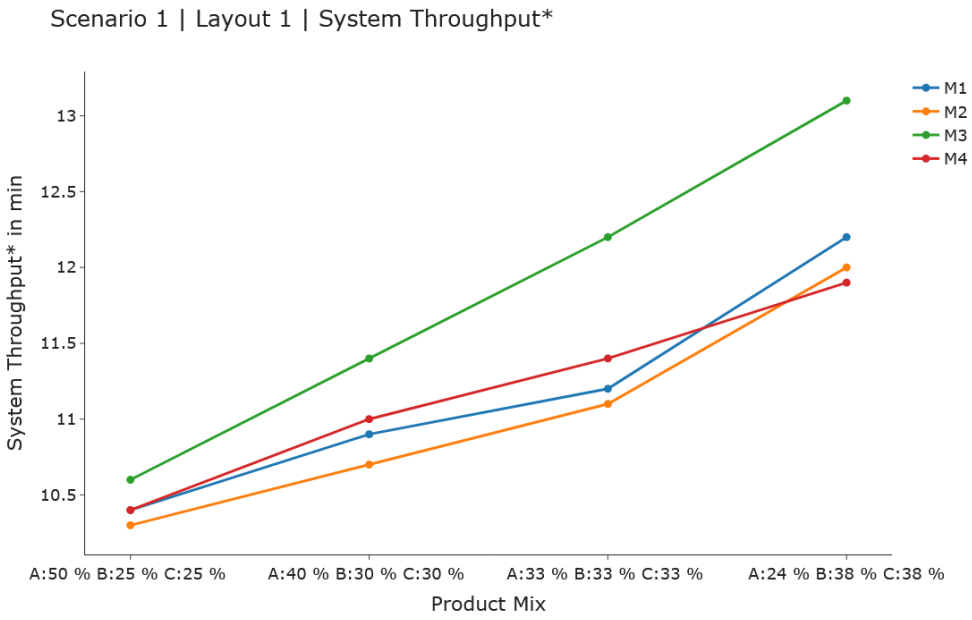

*Figure 38: S1-L1 System Throughput**

Summarizing the first results, it can be said that method 3 is not competing with the others, even though the system throughput* in the first product mix is close to the others, looking at the other measures this method might not be practically usable. Comparing the other 3 methods shows that method 1 is best for systems where the system does not get easy into a deadlock. Method 2 excels if the system does get easy into deadlocks and therefore seems to be the most promising one for this scenario and layout. Method 4 lies in between method 1 and 2.

5.2 Scenario 1 / Layout 2

In the second layout a circle for possible deadlocks was added. Because of this it has to be mentioned that the decrease of product A over the product mixes does not mean a less deadlock safe system but rather that a deadlock is more likely to occur at the added circle.

The values for deadlock time show that method 2 is overall the best. Method 1 is at the first product mix the most useful one but gets worse with changing product mixes. For the last two mixes method 4 is better than method 1 and close to method 2. Method 3 is again not near to the other ones.

Deadlock Time in h				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	33,0	31,5	15,5	28,2
A:40 % B:30 % C:30 %	15,3	18,3	10,2	15,6
A:33 % B:33 % C:33 %	11,2	13,6	7,6	13,2
A:24 % B:38 % C:38 %	7,1	9,1	5,2	8,8

Table 7: S1-L2 Deadlock Time

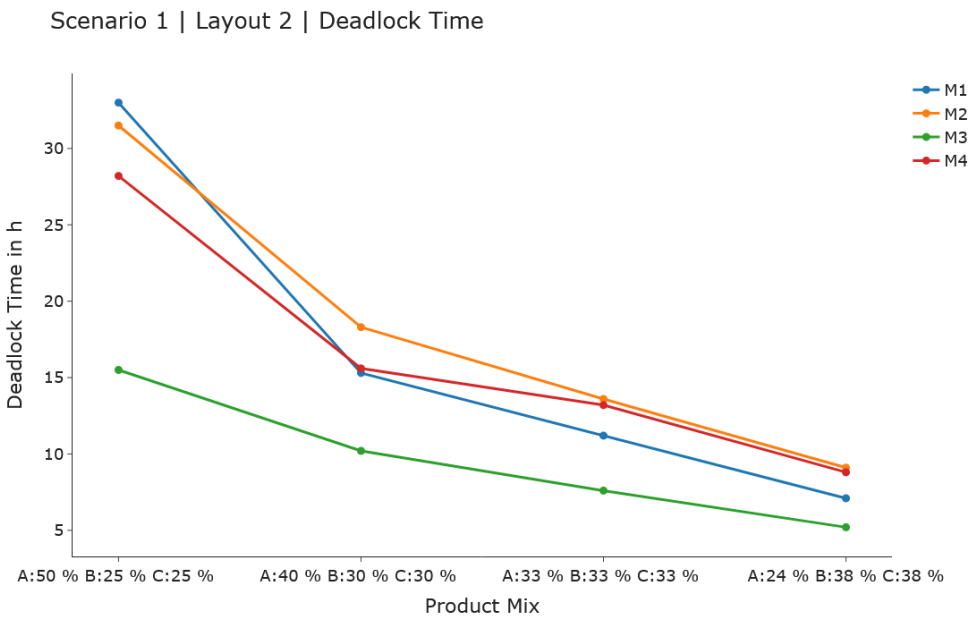

Figure 39: S1-L2 Deadlock Time

When comparing the results of the products per deadlock to the deadlock time it gets obvious that they are dependent on each other since they are nearly the same. This means that, in this layout, the longer a system lasts and does not get into a deadlock the more products get through it. Because of this, the same outcome as explained for the deadlock time values are valid for the products per deadlock.

Products/Deadlock				
Method Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	183	179	84	159
A:40 % B:30 % C:30 %	82	100	52	83
A:33 % B:33 % C:33 %	57	72	37	69
A:24 % B:38 % C:38 %	33	45	23	43

Table 8: S1-L2 Products/Deadlock

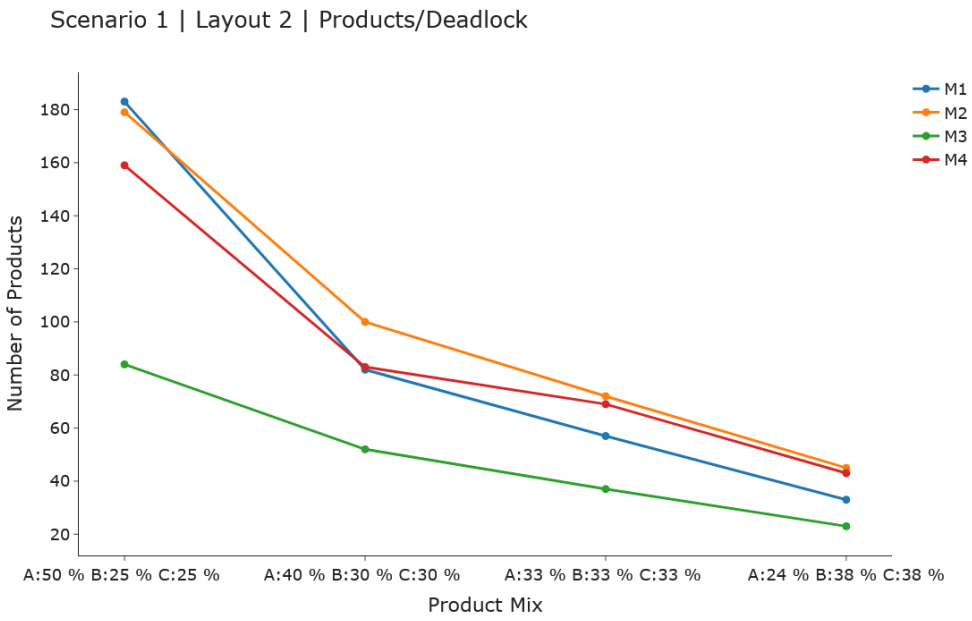

Figure 40: S1-L2 Products/Deadlock

As again the two measured values do not point in opposite directions the calculated system throughput* follows them and shows method 2 as the best. In the last product mix method 4 catches up to method 2 and has the same result.

System Throughput* in min				
Method Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	10,8	10,6	11,1	10,6
A:40 % B:30 % C:30 %	11,2	11,0	11,8	11,2
A:33 % B:33 % C:33 %	11,8	11,4	12,4	11,6
A:24 % B:38 % C:38 %	12,9	12,2	13,5	12,2

*Table 9: S1-L2 System Throughput**

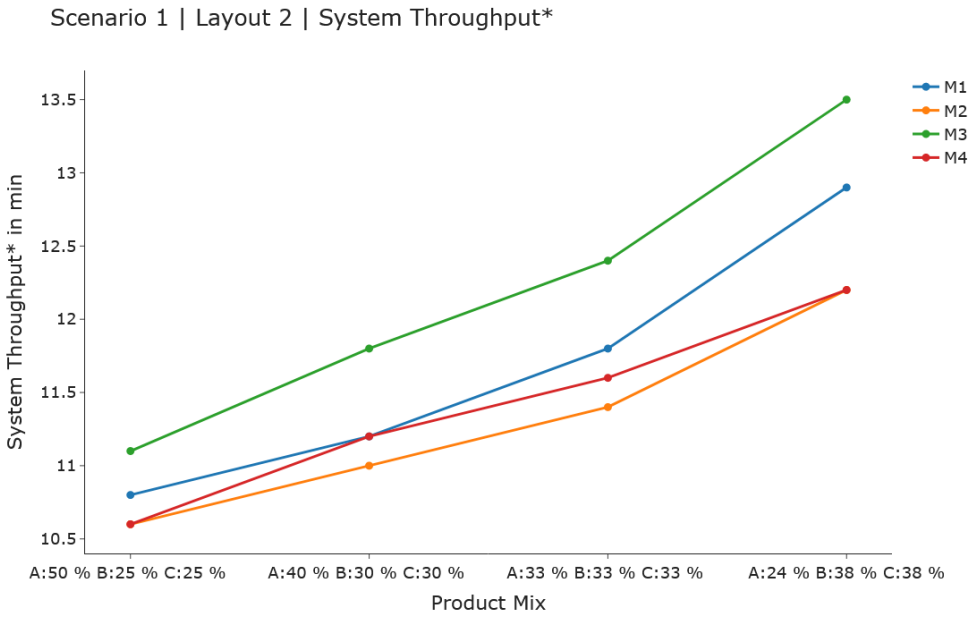

*Figure 41: S1-L2 System Throughput**

To conclude layout 2 of scenario 1 it can be said that method 2 is again the best option but method 4 is coming close for the last two product mixes. Method 3 is also again not competing with the others and method 1 gets good results in the at first but falls of as the deadlock occurrence shifts closer to the second circle.

5.3 Scenario 2 / Layout 1

In scenario 2 the average arrival time gets set to 15 minutes instead of 10 from scenario 1. Through this the system does not get easy in a deadlock and lasts longer. If a test run exceeds 48 hours, it is marked as no deadlock. Through this a new value for comparison is added which is the no deadlock percentage. This value is in close relation with the deadlock time. If for example, the no deadlock percentage is 100 %, the deadlock time must be 48-hours. This correlation can be used for a detailed analysis to find out if in some cases, deadlocks happen unusually early.

Looking at the deadlock time it can be seen that the system operates longer than the others with the use of method 1. Method 2 and 4 are close but this time method 4 is better than method 2. When observing the last product mix, which favours deadlock occurrence the most, method 1 its deadlock time declines stronger than method 2 or 4, making them equally good for this mix. For another perspective, the values can be evaluated by how close they get to the 48-hour mark.

Deadlock Time in h				
Method Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	45,4	45,7	31,3	41,4
A:40 % B:30 % C:30 %	42,7	39,2	26,5	39,6
A:33 % B:33 % C:33 %	41,0	31,6	25,4	34,2
A:24 % B:38 % C:38 %	30,2	28,7	17,3	29,9

Table 10: S2-L1 Deadlock Time

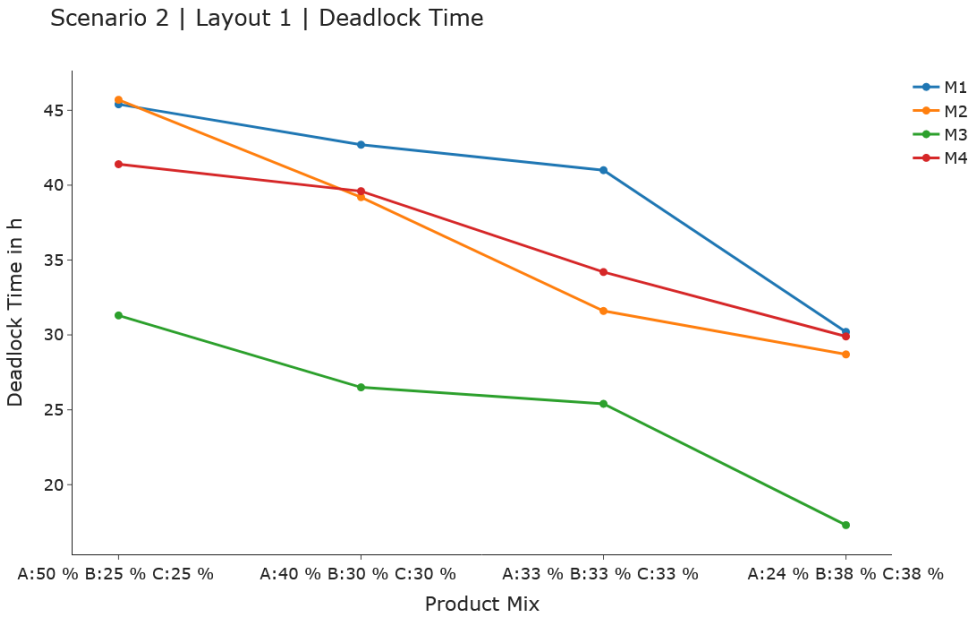

Figure 42: S2-L1 Deadlock Time

The products per deadlock measure delivers a nearly identical comparison of the methods as the deadlock time.

Products/Deadlock				
Method Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	184	185	151	175
A:40 % B:30 % C:30 %	169	153	101	154
A:33 % B:33 % C:33 %	164	127	100	138
A:24 % B:38 % C:38 %	115	109	62	114

Table 11: S2-L1 Products/Deadlock

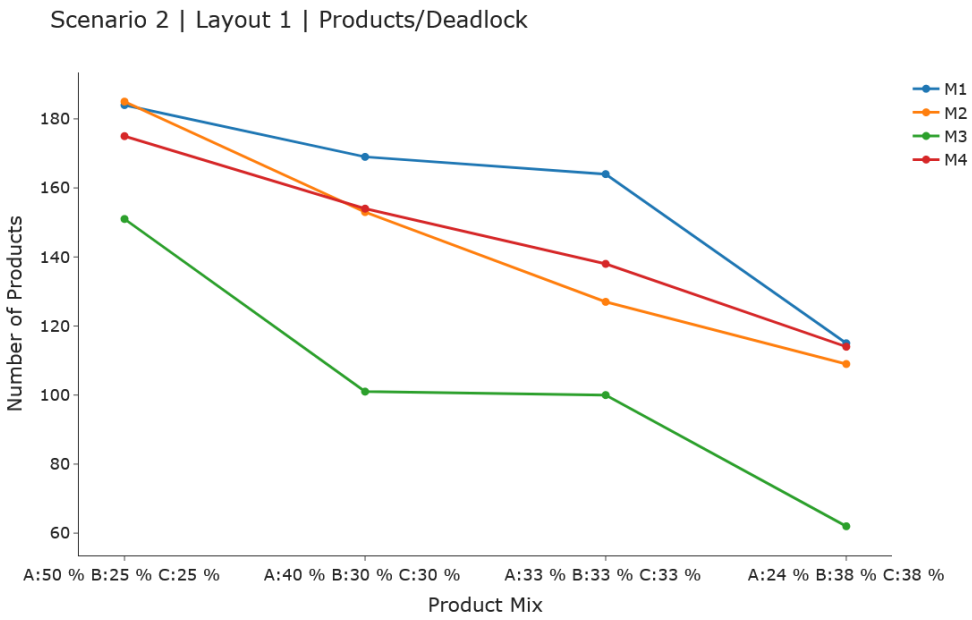

Figure 43: S2-L1 Products/Deadlock

In the system throughput* comparison, many changes can be seen. First, method 3 has the best result for the easiest product mix, which then gets worse fast. When looking at the values of method 1, 2 and 4 it gets clear that there is close to no difference since the difference is around 15 seconds to each other. This can be explained through the limited run time and the fact that many runs of the simulation end after 48-hours simulated time.

System Throughput* in min				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	14,8	14,8	12,4	14,2
A:40 % B:30 % C:30 %	15,2	15,4	15,7	15,4
A:33 % B:33 % C:33 %	15,0	14,9	15,3	14,9
A:24 % B:38 % C:38 %	15,7	15,9	16,6	15,8

*Table 12: S2-L1 System Throughput**

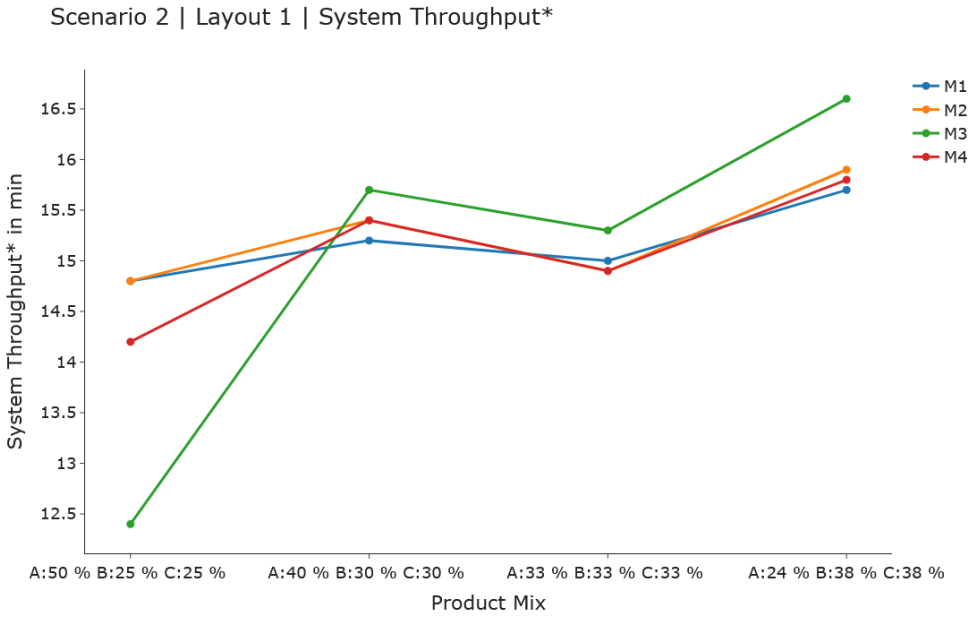

*Figure 44: S2-L1 System Throughput**

The no deadlock values show the same results as the deadlock time, as expected due to the correlation of these two values. Simulation runs that do not run into a deadlock during the 48-hour period of time, decline very drastically. At first, methods 1, 2 and 4 keep the system nearly deadlock free at around 90 %. For the most difficult situation, only around 30 % of the simulations do not get into a deadlock.

% no Deadlock				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	94%	95%	64%	86%
A:40 % B:30 % C:30 %	78%	65%	27%	68%
A:33 % B:33 % C:33 %	67%	41%	21%	47%
A:24 % B:38 % C:38 %	34%	30%	8%	34%

Table 13: S2-L1 % no Deadlock

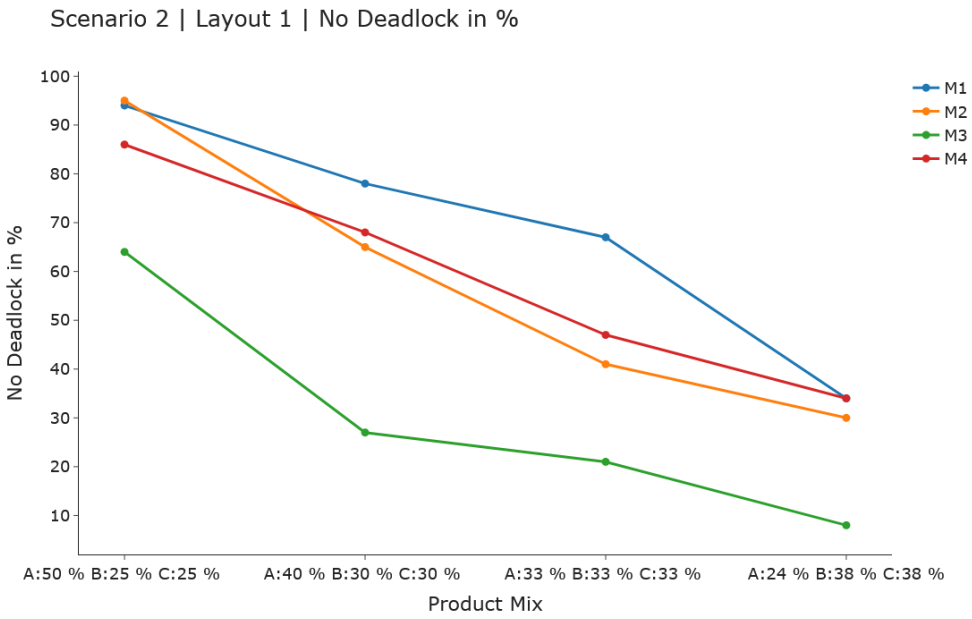

Figure 5.45: S2-L1 % no Deadlock

Summarizing this set up, the overall most useful method might be method 1. Method 2 and 3 are very close to each other. Method 3 is the most unusable. For the last product mix, which favours deadlock occurrence the most, the results of the methods get very close.

5.4 Scenario 2 / Layout 2

In this test run the easier system in relation to deadlock occurrence is paired with the layout that has an additional circle added.

Looking at the measures for the deadlock time, method 2 is overall the best and method 3 the worst. For the first product mix the results are close to each other. With advancing product mixes the difference gets more significant. The same can be said for the products per deadlock measures.

Deadlock Time in h				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	44,7	45,9	36,7	44,4
A:40 % B:30 % C:30 %	40,9	42,9	28,2	40,8
A:33 % B:33 % C:33 %	35,6	40,1	23,8	36,8
A:24 % B:38 % C:38 %	28,6	34,9	18,1	32,4

Table 14: S2-L2 Deadlock Time

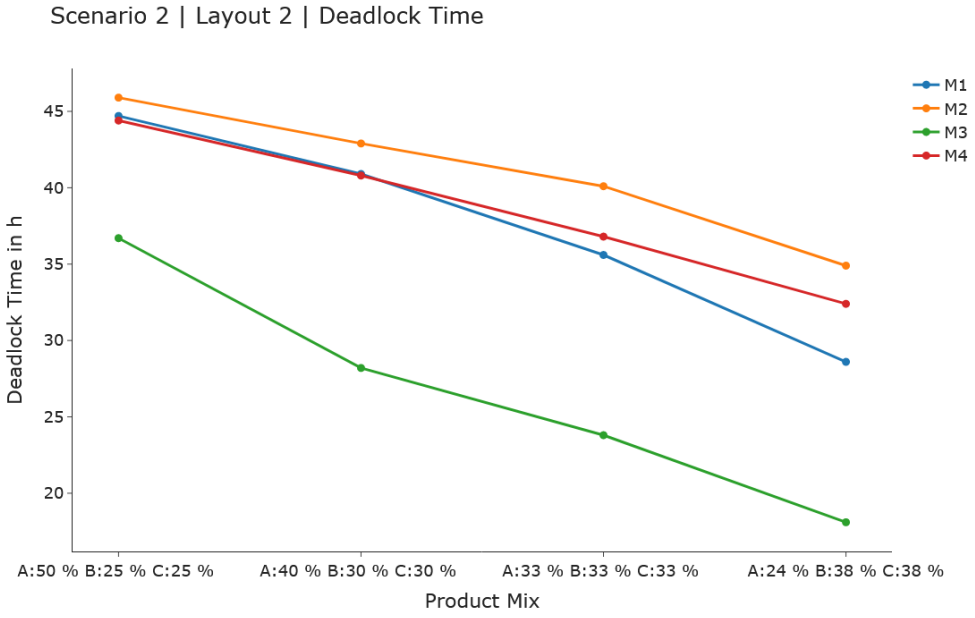

Figure 46: S2-L2 Deadlock Time

Products/Deadlock				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	177	181	142	174
A:40 % B:30 % C:30 %	160	167	107	161
A:33 % B:33 % C:33 %	138	146	88	143
A:24 % B:38 % C:38 %	109	134	65	122

Table 15: S2-L2 Products/Deadlock

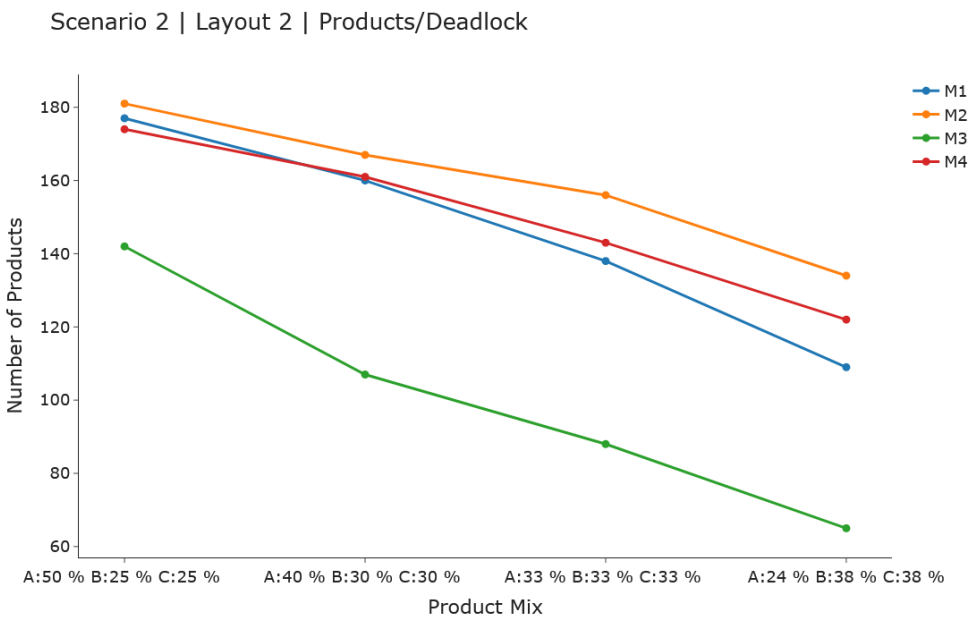

Figure 47: S2-L2 Products/Deadlock

Method 3 is compared by the system throughput* the worst. The others are very close to each other and the difference is below half a minute.

System Throughput* in min				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	15,2	15,2	15,4	15,3
A:40 % B:30 % C:30 %	15,3	15,4	15,8	15,2
A:33 % B:33 % C:33 %	15,5	15,4	16,3	15,5
A:24 % B:38 % C:38 %	15,7	15,6	16,8	16,0

*Table 16: S2-L2 System Throughput**

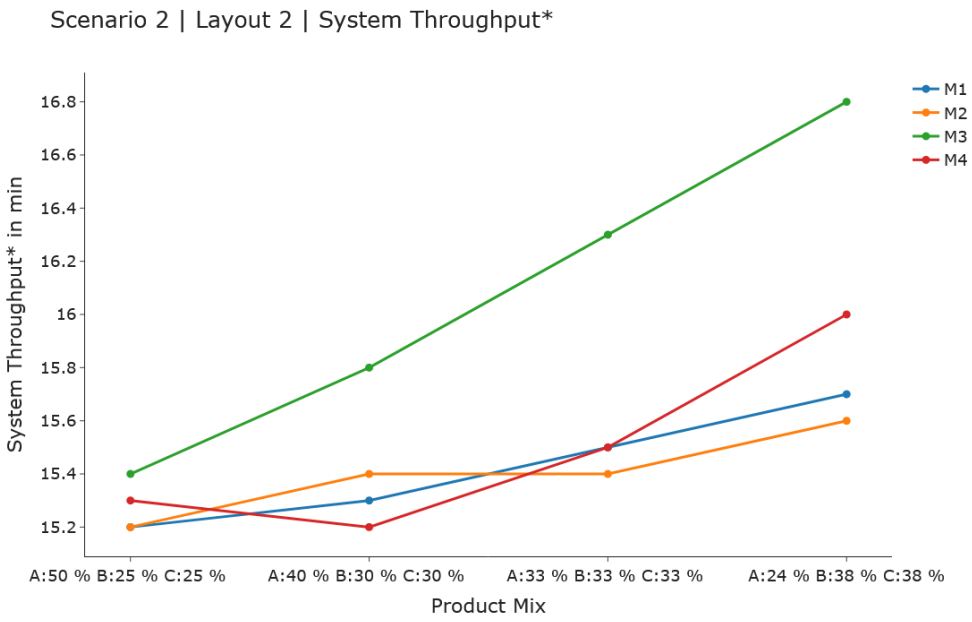

*Figure 48: S2-L2 System Throughput**

The no deadlock percentage starts again with close to 90 %. But do not decline as much for layout 2 as in layout 1. Method 2 is the best in avoiding deadlocks of the methods and exceeds the 48-hour mark nearly 50 % of them for the worst product mix.

% no Deadlock				
Method / Product Mix	M1	M2	M3	M4
A:50 % B:25 % C:25 %	85%	91 %	56%	85%
A:40 % B:30 % C:30 %	70%	81%	30%	71%
A:33 % B:33 % C:33 %	51%	65%	17%	55%
A:24 % B:38 % C:38 %	31%	49%	10%	39%

Table 17: S2-L2 % no Deadlock

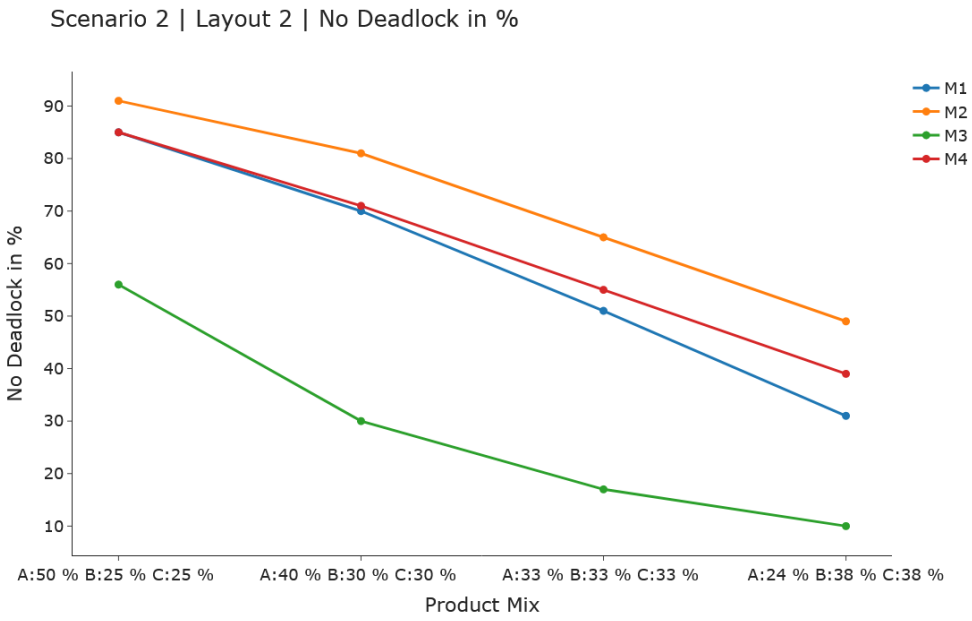

Figure 49: S2-L2 % no Deadlock

5.5 Summary of the Results

In general, it can be said that the measured values get worse with the decline of product A and the rise of product B and C. Through this behaviour it can be seen how the deadlock control methods develop in a system in which the deadlock probability is rising step wise.

Method 1 has the best outcome when evaluating the deadlock time and the products per deadlock for the first product mix. Even though the system throughput* is not the best at this mix, it is very close. This suggests that in a system where the deadlock occurrence is low, method 1 that does not intervene much, seem to be superior. By intervening it is meant that method 1 only releases locked buffers and until then the system stays the same, in comparison to method 2 and 4 where buffers get locked before deadlocks are detected and change the flow. But soon as the deadlock occurrence gets higher method 1 falls of rapidly, meaning that for a more critical system with a higher probability for deadlock to occur, this method might not be the most suitable.

Method 2 seem to be overall the best. It achieves good results for the for the first product mix and falls of the least when the deadlock occurrence rises. Also, the system throughput* is most of the time the best, even though the difference to the others is marginal.

Method 4, which is the combination of method 2 and 3 is also close, which leads to the suggestion that the locked buffer strategy, which all of them have in common, has a big impact on the deadlock avoidance. It can also be seen that methods 4 is slightly worse for the first product mix than the others with the locked buffer strategy but gets close to method 2 for more difficult systems. This corroborates the suggestion that more system intervention worsens the outcome for systems with lower deadlock occurrence, since method 4 changes the flow of the system through prioritization and the locked buffer strategy. On the other hand, the values for method 4, same as method 2, do not fall of as much as method 1 when deadlock occurrence rises. This suggests that the implementation of a simulation prediction is beneficial for more critical systems.

Method 3 is in no scenario as useful as the others. In some cases, this method does also not decrease in performance as much as method 1 does, like the other prediction methods, and in some cases its behaviour is similar to method 1. It seems to be likely that this happens since the method is not targeted in the right direction. It is intervening in the system to not have good results when the deadlock occurrence is low but also intervening to little when possible deadlocks get detected.

5.6 Control Method vs. No Method

In this section the deadlock method 2, which is the overall best, is compared to running the system with no deadlock control methods.

Figure 50 shows the deadlock time and the products per deadlock results for all combinations of scenarios and layouts. In scenario 1 the results with no methods stay nearly the same, since it can most likely not get much worse and this time is needed to fill the system and the circles with products. In other words, the system runs into a deadlock shortly after enough products got into the systems. Method 2 achieves way better results for the first, easy product mix and is still around twice as good as no methods for the last and most difficult product mix.

Scenario 2 is easier and does not get into a deadlock as quickly as scenario 1. Due to this the results with no method show a decline over the more difficult product mixes. This means that with no method applied in scenario 2, it is not as close to its worst state as in scenario 1. But still method 2 is overall around twice as good. Especially for the more difficult product mixes method 2 shows its adventages.

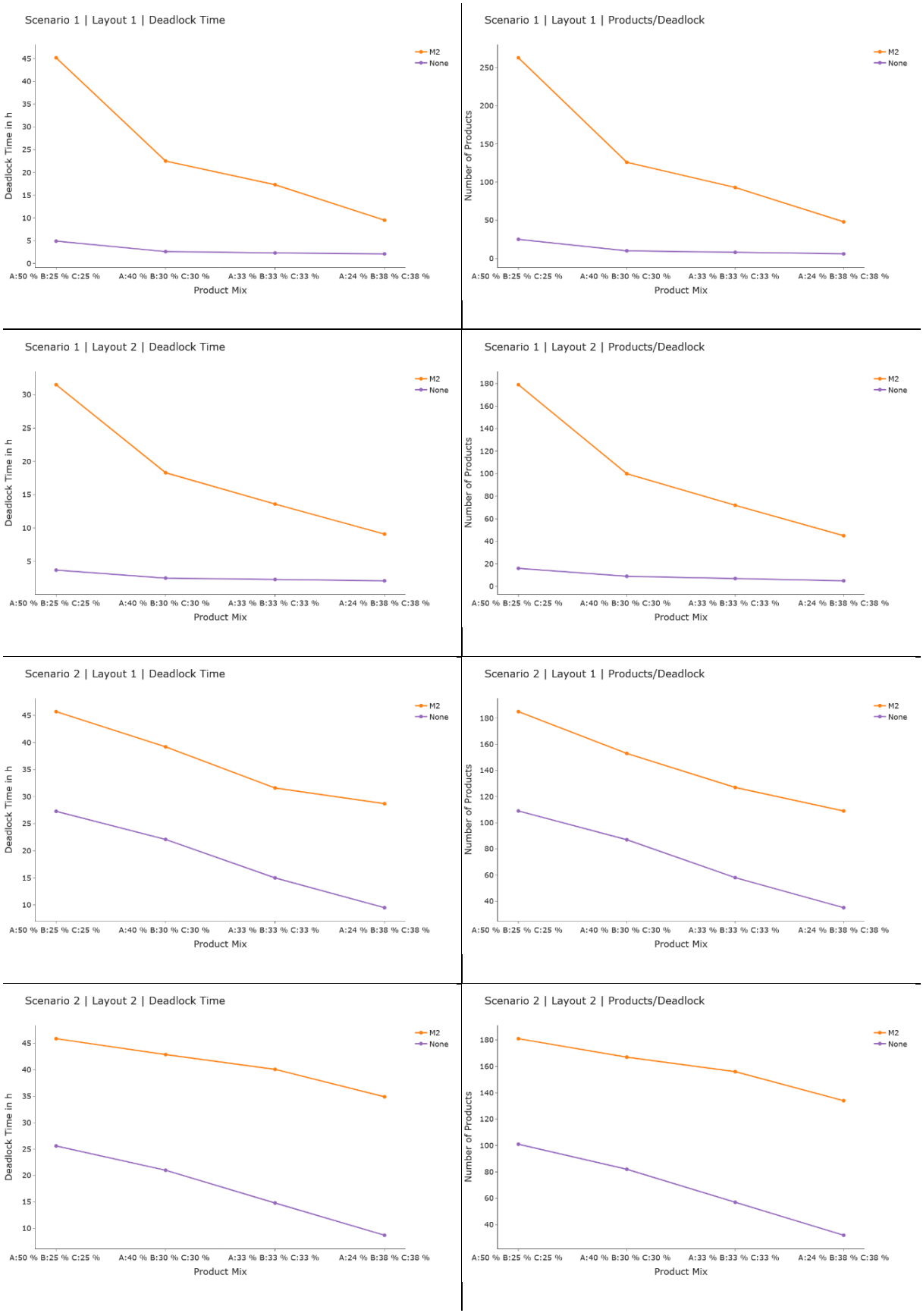

Figure 50: Method 2 vs. No Method

6 Outlook

This case study shows that deadlock prediction gets better results for system with higher deadlock occurrence. The difference can be seen even with simple and known methods like the locked buffer strategy. It also gets clear that it is difficult to apply a beneficial policy change and that it is important to adjust it to the system. The results of this case study have to be proven on a model of a real manufacturing system to get better sight on what can be achieved with this method, especially when correctly adjusted and the practically of it. Another point that has yet to be proven feasible is the processing time for bigger systems and the ability to get real-time results. More work has to be done especially on policy changes to influence the system more targeted in the right direction. In this area also the combinations of policy changes and the interaction of their characteristics has to be observed. Another point of interest is what results could be achieved with the use of machine learning. A successful implementation of prediction simulation with adjusted deadlock control strategies is able to make deadlocks controllable with only little lowering of the performance of the production system.

List of Figures

List of Tables

Bibliography

Baruwa, O., Piera, M.A. and Guasch, A. (2014) 'Baruwa, Olatunde T., Miquel Angel Piera, and Antoni Guasch. "Deadlock-free scheduling method for flexible manufacturing systems based on timed colored Petri nets and anytime heuristic search', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 831-846.

Bonney, M.C., Zhang, Z., Head, M.A., Tien, C.C. and Barson, R.J. (1999) 'Are push and pull systems really so different?', *International journal of production economics*, pp. 53-64.

Browne, J. (1984) 'Classification of flexible manufacturing systems', *The FMS magazine 2.2*, pp. 114-117.

Cardoso, J. and Heloisa, C. (1998) *Fuzziness in Petri*, 22nd edition, Heidelberg: Physica-Verlag.

Coffman, E.G., Elphick, M. and Shoshani, A. (1971) 'System deadlock', *ACM Computing Surveys (CSUR) 3.2*, pp. 67-78.

Cox Jr, T. (1989) 'Toward the measurement of manufacturing flexibility', *Production and Inventory Management Journal 30.1*, p. 68.

ElMaraghy, H.A. (2005) 'Flexible and reconfigurable manufacturing systems paradigms', *International journal of flexible manufacturing systems 17.4*, pp. 261-276.

Fanti, M.P., Giua, A. and Seatzu, C. (2006) 'Monitor design for colored Petri nets: An application to deadlock prevention in railway networks', *Control Engineering Practice*, pp. 1231-1247.

Fanti, M., Maione, G. and Turchiano, B. (1996) 'Fanti, M. P., Maione, G., & Turchiano, B. (1996, May). Deadlock detection and recovery in flexible production systems with multiple capacity resources', *In Proceedings of 8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications* , pp. 237-241.

Furian, N., O'Sullivan, M.J., Voesnner, S. and Walker, C. (2014) 'HCCM-A Control World View For Health Care Discrete Event Simulation', *In ECMS*, pp. 206-213.

Furian, N., O'Sullivan, M., Walker, C., Vössner, S. and Neubacher , D. (2015) 'A conceptual modeling framework for discrete event simulation using hierarchical control structures', *Simulation modelling practice and theory 56*, pp. 82-96.

Garrett, S.E. (1986) 'Strategy first: A case in FMS justification', *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems. Elsevier, Amsterdam, The Netherlands*, pp. 17-29.

Gu, C., Li, Z., Wu, N., Khalgui, M., Qu, T. and Al-Ahmari, A. (2018) 'Improved multi-step look-ahead control policies for automated manufacturing systems', *IEEE Access 6*, pp. 68824-68838.

Hu, S.J. (2005) 'Paradigms of manufacturing—a panel discussion', *3rd Conference on Reconfigurable Manufacturing*.

Jain, A. (2013) 'A review on manufacturing flexibility', *International Journal of Production Research 51.19*, pp. 5946-5970.

Kim, -O. (1997) 'An efficient real-time deadlock-free control algorithm for automated manufacturing systems', *International journal of production research*, pp. 1545-1560.

Koren, Y. (2006) 'General RMS characteristics. Comparison with dedicated and flexible systems', *Reconfigurable manufacturing systems and transformable factories*, pp. 27-45.

Lawley, M. and Sulistyono, W. (2002) 'Robust supervisory control policies for manufacturing systems with unreliable resources', *IEEE Transactions on Robotics and Automation 18.3*, pp. 346-359.

Lenz, J.E. (1992) 'The need for both labor and machine flexibility in manufacturing', *Industrial Engineering 24.10*, pp. 22-23.

Li, Z. (2011) 'Deadlock Control of Automated Manufacturing Systems Based on Petri Nets—A Literature Review', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, pp. 437-462.

Narain, R., Yadav, R.C., Sarkis, J. and Cordeiro, J.J. (2000) 'The strategic implications of flexibility in manufacturing systems', *International Journal of Agile Management Systems*, pp. 202-213.

Robinson, S. (2004) 'Simulation: the practice of model development and use', in Robinson, S. (ed.) *Simulation: the practice of model development and use*, 50th edition, West Sussex PO19 8SQ, England: John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester.

Ropohl, G. (1967) 'Zum Begriff der Flexibilität', *Werkstattstechnik Vol. 57*, p. 644.

Seidl, M. and Günther, S. (2000) *Avoiding deadlocks in flexible manufacturing systems*, 569th edition, Boston, MA: Springer.

Sethi, A.K. and Sethi, S.P. (1990) 'Flexibility in manufacturing: a survey.', *International journal of flexible manufacturing systems 2.4*, pp. 289-328.

Stecke, K.E. (1983) 'Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems', *Management science 29.3*, pp. 273-288.

Unisig (2018) *www.unisig.com*, Sep, [Online], Available: https://www.unisig.com/wp-content/uploads/2018/09/rifle-barrel-manufacturing-production-cell.jpg [May 2020].

Upton, D.M. (1994) '72-89', *California management review 36.2*, pp. 72-89.

Upton, D.M. (1995) 'What really makes factories flexible', *Harvard business review 73.4*, pp. 74-84.

Viswanadham, N., Narahari, Y. and Johnson, (1990) 'Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models', *IEEE Transactions on Robotics and Automation 6.6*, pp. 713-723.

Wainer, G.A. (2009) 'Computer simulation', in Group, T.&.F. (ed.) *Discrete-event modeling and simulation: a practitioner's approach*, Baco Raton, FL: CRC press.

Wysk, R.A., Joshi, S. and Yang, N.-S. (1991) 'Detection of deadlocks in flexible manufacturing cells', *IEEE Transactions on robotics and automation*, pp. 853-859.

Xing, K., Han, L., Zhou, M. and Wanf, F. (2011) 'Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, pp. 603-615.

Xing, K., Han, L., Zhou, M. and Wang, F. (2011) 'Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, pp. 603-615.

Zajac, J. (2004) 'A deadlock handling method for automated manufacturing systems', *CIRP Annals-Manufacturing Technology 53.1*, pp. 367-370.

Zhu, Q., Wu, N., Qiao, Y. and Zhou, M. (2014) ' Scheduling of single-arm multi-cluster tools with wafer residency time constraints in semiconductor manufacturing', *IEEE Transactions on Semiconductor Manufacturing*, pp. 117-125.