DESIGN OF SUPERVISORY MACHINE CONTROL

N.J.M. van den Nieuwelaar*†, J.M. van de Mortel-Fronczak†, J.E. Rooda†
*ASML, De Run 1110, 5503 LA Veldhoven, The Netherlands
†Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
email: {n.j.m.v.d.nieuwelaar, j.m.v.d.mortel, j.e.rooda}@tue.nl

Keywords: supervisory control, complex manufacturing machines, task resource systems, design framework, optimal behaviour.

Abstract

This paper presents a basis for a framework for the design of supervisory machine control. Machine behaviour is described using the task resource system (TRS) paradigm. Analysis is performed to classify system descriptions leaving room for control choices up to a certain extent: selected untimed TRS, instantiated unselected TRS and TRS constraints. As a consequence of the physical nature of machines and products, some machine-specific issues are involved. Furthermore, an overview of existing techniques that are suitable to support the design of supervisory control is given. This overview involves techniques from different research areas that help to make well-founded design decisions and identifies relevant issues.

1 Introduction

The purpose of a manufacturing machine is to add value to products. To this end, physical manufacturing processes must be carried out. To actually do the work, mechatronic systems in the machine must be deployed. In this context, the machine can be considered as a task resource system [8]. A manufacturing process can be associated with a task, whereas a mechatronic system can be associated with a resource. Complex manufacturing machines consist of multiple mechatronic systems. Control of the separate mechatronic systems is referred to as low-level control, and is primarily not considered in this paper. Many possibilities exist to deploy the available resources to perform tasks that lead to the desired manufacturing purpose, resulting in different machine behaviour. In this paper, supervisory machine control is considered that co-ordinates mechatronic systems operating in parallel.

Supervisory machine control is an emerging application area as machines become more and more flexible, complex, and expensive. This effect is boosted by the rapidly increasing processing power that can be applied in machine control systems. Minimization of the machine's Cost of Ownership (CoO) [7] is an important issue in industry. It can be achieved by optimising machine performance through efficient usage of available resources, which is the purpose of supervisory machine control. Control of mechatronic systems has received much attention in literature [5]. Also the analysis of discrete-event and timed systems has been an important subject in the area of computer science and operations research [13]. The application area of supervisory machine control involves multiple research areas. Therefore, a framework is required that combines the relevant issues for supervisory machine control from these areas.

The purpose of this paper is twofold. First, as a basis for reasoning, temporal machine behaviour is described that results from a definition of the system according to the task resource paradigm. Several system definitions are discussed, that leave more or less room for choices that affect machine behaviour. These choices can be made during the design, which implies static scheduling. When dynamic scheduling is applied, these choices are made run-time by supervisory control. Second, an overview of the existing techniques that can support the choice-making process and design decisions is given. This paper provides a firm foundation for a framework for the design of supervisory control. It gives an overview of issues involved and relevant techniques known. Open issues, which are currently under investigation, are pointed out. For illustration, examples and cases from a wafer scanner [1] are used. A wafer scanner is a machine used in semiconductor industry. It projects a mask on a wafer (a slice of polished silicon) covered with a thin layer of photo-sensitive emulsion, using light. A wafer scanner is a good example of a complex manufacturing machine, in which numerous actuators and sensors are involved that have to be co-ordinated and controlled within nanometer accuracy. Several control problems related to material flow also play an essential role.

This paper is structured as follows. Section 2 describes the time behaviour of a TRS in terms of a sequence of state transitions for each resource, imposed by tasks in a certain order. First, the time required for execution of a single task is determined based on the behaviour of the individual resources. After that, time required for execution of the total system is determined based on resource behaviour imposed by the tasks in a certain order. In Section 3, two classes of more relaxed system definitions are discussed within the context of supervisory machine control. First, a system definition is considered that leaves room for some alternatives with respect to tasks, resources, and precedences. Subsequently, a system definition describing the constraints on task precedences is introduced, based on integrity constraints on task pre- and post-conditions. Furthermore, several aspects to be taken into account during the design of supervisory machine control are discussed. Section 4 discusses suitability of known techniques to support the design of supervisory control. Finally, concluding remarks are presented in Section 5.

2 Time behaviour of a task resource system

Machine resources generally show continuous behaviour. At supervisory level, resource behaviour is considered at discrete points in time only: at the start and at the end of a task. Supervisory control imposes certain resource states at these points in time, and puts some additional restrictions on the transition behaviour. In this context, only the required duration of the resource state transitions is relevant, which is restricted by the resource performance parameters or capacities. In this section, first the time required for a single resource state transition is discussed. In fact, this forms a bridge from the continuous behaviour of machine resources to the abstraction level required by supervisory control. After that, the time required for a single task involving multiple resources is determined based on the behaviour of the single resources involved. The resources involved have to co-operate in order to carry out the task. In the sequel we refer to this as synchronous concurrency. Finally, the time required for execution of the total system is determined, involving multiple sequential and partly parallel tasks and two forms of resource concurrency: synchronous concurrency inside tasks, and asynchronous concurrency between tasks.

Let $\mathcal{R}$ be the set of resources in the system, and let $\mathcal{T}$ be the set of tasks in the system. Let $\mathcal{S}$ be the set of possible states of all resources in the system. As stated before, first a single state transition of a single resource *r* is considered. Let $\mathcal{S}(r)$ be the set of states of resource *r*. The capacity of resource *r*, $c(r)$, can be described by a certain number of parameters: $(\forall r : r \in \mathcal{R}\text{: } (\exists n : n \geq 0 : c(r) \in \mathbb{R}^n))$. Let $\mathcal{B}(r,t)$ be a set of Differential Algebraic Equations (DAEs), including inequalities, describing the allowed continuous behaviour of resource *r* for task *t*. The behaviour description of resource *r* depends on the capacity restriction $c(r)$. Let $\Pi(\mathcal{B}(r,t))$ be the set of possible solutions of $\mathcal{B}(r,t)$. Furthermore, the minimal time $\tau \in \mathbb{T}$ required for a state transition of *r* depends on the begin and the end states $s_b(r)$, $s_e(r) \in \mathcal{S}(r)$. Here, $\mathbb{T}$, stands for time and equals the set of non-negative real numbers $\mathbb{R}^+$. This minimal time $\tau \in \mathbb{T}$ is the time for which holds:

$$\begin{aligned}&(\exists \varphi\text{: } \varphi \in (\mathbb{T} \rightarrow \mathcal{S}(r))\\ &\quad : \varphi(0) = s_b(r) \wedge \varphi(\tau) = s_e(r) \wedge \varphi \in \Pi(\mathcal{B}(r,t)))\end{aligned} \qquad (1)$$

As an example, consider one-dimensional motions. Let *x* be the considered dimension, which can be associated with a position, depending on time. Then, the first, second and third time derivative of *x* can be associated with speed *v*, acceleration *a*, and jerk *j*. First, no transition restriction is considered. Suppose that the resource capacity $c(r)$ is described by constraints on the absolute value of *x* and its first, second and third derivative, *X*, *V*, *A*, and *J*, respectively. Furthermore, the begin position, acceleration, and jerk are zero, as well as the end acceleration and jerk. The begin speed is v_b, whereas the end position and speed are x_e and v_e, respectively. Then, minimal time needed for this particular resource state transition can be defined by substitution in Expression 1 as follows:

$$s_b(r) = \begin{bmatrix} 0 \\ v_b \\ 0 \\ 0 \end{bmatrix}, s_e(r) = \begin{bmatrix} x_e \\ v_e \\ 0 \\ 0 \end{bmatrix}, \text{and } \mathcal{B}(r,t) = \begin{cases} x \leq |X| \\ \dot{x} \leq |V| \\ \ddot{x} \leq |A| \\ \dddot{x} \leq |J| \end{cases} .$$

An example of a transition restriction imposed by a task would be constant speed, which means that *A* and *J* are replaced by 0.

Subsequently, minimal time required for execution of a task is considered. If only one resource is involved, the minimal time required for the task is equal to the minimal time required for the state transition of this resource. However, for one task multiple resources might be involved. These resources have to transform their state synchronously for this task. A typical example from a wafer scanner is an exposure scan, during which the reticle and the wafer are moved synchronously in opposite directions. Let *I* be a function that assigns to each task *t* the set of resources involved: $I \in (\mathcal{T} \rightarrow \mathcal{P}(\mathcal{R}))$, where $\mathcal{P}(\mathcal{R})$ is the powerset of $\mathcal{R}$. Let S_b, $S_e \in (\mathcal{T} \times \mathcal{R} \rightarrow \mathcal{S})$ be functions assigning a begin and an end state, respectively $S_b(t,r)$, $S_e(t,r) \in \mathcal{S}(r)$, to each resource involved in task *t*. Let $\tau_a \in (\mathcal{S}(r) \times \mathcal{S}(r) \times \mathbb{B} \rightarrow \mathbb{T})$, where $\mathbb{B}$ is the set of all possible DAEs, be the function determining the minimal time required for a state transition of resource *r* from state $s_b(r)$ to state $s_e(r)$ within the constrained behaviour, as is described previously. Usually, the time taken for a state transition is allowed to be anything larger than the result of τ_a. In [10] an exception to this assumption can be found, which is not relevant for this paper. Under this assumption, minimal time required for the synchronous resource state transitions of task *t* can be derived from the involved asynchronous state transition durations as follows:

$$(\max r : r \in I(t) : \tau_a(S_b(t,r), S_e(t,r), \mathcal{B}(r,t))) \qquad (2)$$

Finally, minimal time required for the realisation of an entire system implying execution of multiple tasks in a predefined order is discussed. Let $P \in \mathcal{P}(\mathcal{T} \times \mathcal{T})$ be the precedence relation between tasks defining this order, such that the precedence relation does not contain any cycles, and the sequence of tasks per resource is a chain.

Suppose that $P' \subseteq P$ is the union of all these resource task chains. The remaining task precedences are then $P \backslash P'$. The system behaviour from supervisory machine control point of view can be described by the start and finish times of (synchronous) resource state transitions, related to the tasks in their predefined order, which can be graphically displayed in a Gantt chart. Let $\tau_s : (\mathcal{T} \rightarrow \mathbb{T})$ be a function determining the minimal time required for a task, that is, for the synchronous state transitions of this task, such that $\tau_s(t)$ is defined by (2). Let $\tau_{Ss}(t)$ and $\tau_{Fs}(t)$ be the start and end time of the synchronous resource state transitions of task *t*, respectively. Between tasks, resources behave without any transition behaviour constraints implied by tasks. Let $\mathcal{B}(r)$ be the set of DAEs describing the behaviour of resource *r* in this case. Let $\tau_{Sa}(r, t_1, t_2)$ and $\tau_{Fa}(r, t_1, t_2)$ be the start and end time of a resource state transition of resource *r* between tasks t_1 and t_2, respectively. Suppose that the system contains an initialisation task t_I, preceding all other tasks and having an initial resource state assigned to it as the begin state and as the end state as well. Then, for the minimal time $\tau \in \mathbb{T}$ required for the realisation of the total system state transition holds:

$$\begin{aligned}
&(\forall t : t \in \mathcal{T}\colon \tau \geq \tau_{Fs}(t) \wedge \tau_{Ss}(t) \geq 0 \\
&\qquad\qquad \wedge \tau_{Fs}(t) = \tau_{Ss}(t) + \tau_s(t)) \\
&\wedge(\forall t_1, t_2 \;\; : (t_1, t_2) \in P \backslash P' : \tau_{Ss}(t_2) \geq \tau_{Fs}(t_1)) \\
&\wedge(\forall t_1, t_2, r : (t_1, t_2) \in P' \wedge r \in I(t_1) \cap I(t_2) \\
&\qquad\qquad : \tau_{Ss}(t_2) \geq \tau_{Fa}(r, t_1, t_2) \\
&\qquad\qquad \wedge \tau_{Sa}(r, t_1, t_2) \geq \tau_{Fs}(t_1) \\
&\qquad\qquad \wedge \tau_{Fa}(r, t_1, t_2) = \tau_{Sa}(r, t_1, t_2) \\
&\qquad\qquad\qquad\qquad + \tau_a(S_e(t_1, r), S_b(t_2, r), \mathcal{B}(r)))
\end{aligned} \tag{3}$$

3 Supervisory control analysis

In this section, the underlying design decisions are discussed that eventually lead to the desired machine behaviour. First, to outline the decisions involved, system definitions are introduced that leave more or less room for choices with respect to resources, tasks, and task precedences. After making these choices, system behaviour can be determined as has been described in the previous section. TRS definitions can be classified by the level of restriction of these choices. The most restrictive TRS definition has already been discussed in the previous section: an selected, untimed system. The only room for choices not affecting minimal system realisation time concerns tasks that have floats. A TRS definition that is less restrictive defines the alternatives to choose from. A TRS definition that is even more relaxed defines only the constraints that must be satisfied. By making choices, less restrictive TRS definitions can successively be transformed into more restrictive TRS definitions, finally resulting in timed machine behaviour. This layered choice framework is visualised in Figure 1. In the sequel, the TRS definition classes and choices that transform TRS definitions into more restrictive definitions are referred to by digits and letters, respectively, as in Figure 1. Additionally, due to

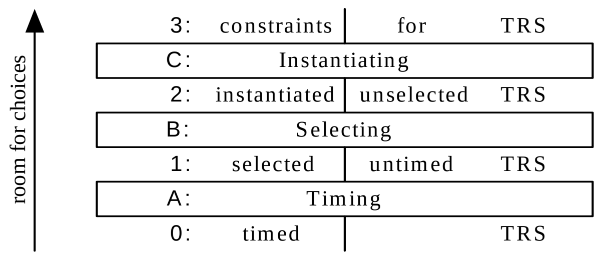

Figure 1: Layered Task Resource System framework

the physical nature of machines and products being manufactured by them, some special issues arise. These issues involve product configuration, material logistics, material capacity of resources and hardware limitations. Finally, the aspects to be taken into account during the design of supervisory control are addressed.

In the previous section, a restriction on the precedence relation is that it must define a chain of tasks per resource. When disregarding resources, only a subset of these task precedences is necessary to ensure a feasible order of tasks. However, a resource can execute only one task at a time: mutual exclusiveness. For each pair of tasks that share some resource, it must be decided which one to do first. In some cases, in a system multiple resources capable of the same job might exist. Examples from a wafer scanner are handling robots and wafer chucks. In some cases, in a system multiple tasks having the same effect might exist. Examples from a wafer scanner are exposure scans that can be executed in either scanning direction. After selection from the various alternatives (B), the system must satisfy the restrictions described earlier. To illustrate the fact that behaviour can be influenced considerably by these choices, consider the following case from a wafer scanner involving choices from task precedence alternatives implied by mutual exclusiveness of resources. The system in this case consists of 30 calibration tasks and 15 resources. Considering task order feasibility, it appears that 41 task precedences are involved, excluding redundant ones. Total amount of work (the summed-up duration of the calibration tasks) equals 38.8 time units, and time needed between tasks can be neglected in this case. In Section 4 it is shown that optimal selection of precedences results in a system realisation time that is considerably less than 38.8 time units.

In some cases, products are produced in batches. For example, wafers are produced in *lots*. Furthermore, a product can consist of or can be part of a number of other manufacturing entities. For example, a wafer consists of a number of chips. The same goes for the other entities required for manufacturing. For example, one or more IC circuit masks are required for lithographic manufacturing of one layer of a chip and multiple masks can be placed on one reticle. For the purpose of this paper, an instance of the aggregate of these entities including a product batch is called a recipe. In general, some manufacturing processes are related to some of these entities, and have to be executed for several instances of these entities. In fact, a combination of such a process and such an instance of a manufacturing entity forms a task. For example, the *lot* size limits the range of instances of the *wafer* manufacturing entity. The manufacturing process *'load wafer'* has to be executed for all wafers. A generic unselected TRS definition (2) for any number of instances of entities should be based on processes in a defined order for any number of instances of entities. This system definition is not elaborated further as it is currently under investigation.

Besides general precedence restrictions some additional specific restrictions might be appropriate for the unselected TRS definitions (2), for instance the ones imposed by material logistics. These restrictions are referred to as logistic (integrity) constraints in the sequel, and have to do with the fact that the system handles material. This can be either material ending up in the product to be manufactured or material that is required to enable manufacturing. Examples from a wafer scanner are wafers and reticles, respectively. Some of the tasks can be associated with material. Material resides in resources, which have limited capacity to hold it. For the purpose of this paper, only one type of material is considered, and one task gets only one material instance associated to it. In this case, material capacity of a resource can be described by an integer number. Moreover, we assume that there is no task concurrency possible for one material instance. Taking logistic integrity into account, the following additional restrictions are imposed on the precedence relation resulting after selection:

- The tasks involving the same material instance must form a chain, and must be consistent with respect to resources: material can be processed by some resource only if it was put there before.
- The actual material content of a resource implied by the tasks involving this resource may not exceed its material capacity.

Besides task precedence restrictions also resource state transition restrictions might exist, imposed by the physical machine. It might be the case that a resource state transition imposed by a chosen precedence relation is not possible asynchronously. Due to physical limitations, such a resource state transition implies a resource state transition for one or more other resources to take place synchronously. An example from a wafer scanner is reticle manipulation using a single manipulator that can contain two reticles. Movement of this manipulator for one reticle implies a synchronous move of the other reticle. As one task gets only one material associated to it (see above), the manipulator must be divided into two resources. Manipulation of one reticle involves synchronous state transitions of two resources.

In an unselected TRS definition (2), some basic precedence relation exists. Previously in this section, the statement has been made that this precedence relation is necessary to ensure a feasible task sequence. However, in some cases, several feasible task sequences might be possible. Definition of the set of all possible precedence relations implies a less restrictive system description. The precedence relation of this system description is completely based on integrity constraints (3). Above, logistic integrity constraints have been described, concerning material flow and material capacity. Additionally, other relevant integrity constraints must be taken into account. This implies the introduction of task conditions for all relevant aspects.

A TRS definition based on integrity constraints (3) might reveal possibilities for timing optimisation that do not exist in an unselected TRS definition (2). Moreover, up to now it has been assumed that the system behaves as expected. When considering exceptional system behaviour, it appears that, to enable recovery from various exceptional system states, a system definition should be based on integrity constraints (3).

The choices (A-C) of what task to execute when and on what resource relate to scheduling, which can be performed statically or dynamically. In case of static scheduling, choices are made during design by the designer, whereas in case of dynamic scheduling choices are made run-time by supervisory control. The decision to apply dynamic scheduling has impact on the behaviour of the machine, as well as on the complexity of supervisory control. This is why this decision should be a careful trade-off. If a highly flexible behaviour with respect to recipes, as well as a highly optimised timing behaviour for a diversity of recipes are desired, this suggests a decision for dynamic scheduling based on unselected TRS definitions (2). If a highly robust behaviour with respect to exceptions is desired, this suggests dynamic scheduling based on integrity constraints concerning pre- and postconditions related to tasks (3). In this case, the scope of supervisory control reaches from the lowest layer to the highest layer of Figure 1. The most important drawbacks of dynamic scheduling for supervisory control are the complexity and the amount of domain knowledge that must be explicitly defined. In practice, a mix of static and dynamic scheduling for certain control scopes and abstraction levels is thinkable. In any case, it is important to have techniques at one's disposal that support the design decisions in this context. In case of dynamic scheduling, these techniques even have to be embedded in supervisory control in order to achieve the required functionality.

4 Supervisory control synthesis

This section discusses known techniques suitable to perform timing analysis based on the system definitions as discussed in the previous section.

A task resource system can be classified as a hybrid system as it contains both continuous-time and discrete-event characteristics. In computer science, several generic hybrid paradigms and associated languages exist which are accompanied by various analysis tools [9, 4]. After a mapping of the original system onto such a paradigm and language, timing analysis can be performed. Supporting tools can be classified as either model checkers or simulators, in case of exploration of the complete state space (all realisations) or exploration of just any realisation, respectively. To determine suitability of these tools for the analysis of a completely predefined system, consider, for instance, the one-dimensional motion problem. Only one trajectory corresponds with the minimal solution of Expression 1. However, the set of DAEs leaves also room for other solutions. Therefore, a simulator is not suitable for derivation of the minimal duration. A model checker is able to determine whether a certain property holds. This property could be whether a solution exists that takes no more than a certain amount of time. Embedding a model checker in an optimisation algorithm that iterates towards the optimal solution would be a possible though inefficient solution to finding the minimal time for which Expression 1 holds. Some model checkers have limited optimisation extensions.

However, from supervisory machine control point of view, only task start states and end states are considered. Therefore, abstraction from continuous behaviour is possible by embedding specific mathematical functions in the model, assigning the required duration to the resource state transitions defined. Using these mathematical functions, the model is simplified to the class of discrete-event systems. In the one-dimensional motion example, the domain knowledge can be used that the optimal trajectory consists of sub-trajectories for which alternatively $\dot{x} = V$, $\ddot{x} = A$, or $\dddot{x} = J$ holds. The determination of the required duration of the resource state transition can be performed by combining analytical functions for most cases. Only in very special cases, an approximation algorithm is required. As this concerns a very restricted solution area, a very simple bisection algorithm suffices [10]. These dedicated mathematical functions take less computing power to find a solution than a generic solver. Also for discrete-event systems, a wide range of paradigms and languages exists, e.g. [3]. Supporting tools can be classified analogously to the tools supporting hybrid languages. Because there are no alternatives in case of a completely predefined system, both tool classes are suited for analysis. This leaves the disadvantage of mapping of the original system onto the generic discrete-event paradigm and language. Even this mapping can be prevented, as calculation of the minimal time for which Expression 3 holds is in fact a linear programming (LP) problem [16]. This enables the usage of a variety of mathematical tools, to derive the total system behaviour from the durations of resource state transitions.

In case of dynamic scheduling, it is desirable that computation of system behaviour starts with tasks that can be dispatched first, called forward computation. In this way, some tasks can already be released, while timing determination of the rest of the tasks is still under progress. Forward computation is not applicable for generic LP problem solvers. A mathematical approach for which this is applicable is the Heap of Pieces approach [15]. A restriction of this approach is that it cannot cope with precedence relations as discussed in this paper.

TRS definitions of class (2) and (3) can have several realisations. Therefore, in principle a simulator is not suited for analysis, whereas a model checker is because it evaluates all possible realisations. However, the combination of possible choices blows up the number of realisations exponentially. In model-checking terminology, this phenomenon is known as state-space explosion. Therefore, most model checkers are equipped with state-space reduction techniques. Intuitively, it should be possible to find recurrent patterns concerning different instances of the same manufacturing entity. However, it appears that this is not the case, as these techniques are based on interrelations between clock variables rather than repetition counter variables. Furthermore, the mapping of the original model onto a generic paradigm and language usually introduces even more possibilities than exhibited by the original model. Therefore, model checkers are in principle not suited to analyse practical cases. In operations research, several approaches can be found that address certain aspects of the choices from alternatives. The choice of tasks and task precedences for one resource results in different realisation times. This issue is widely discussed in literature, and is referred to as the Travelling Salesman Problem [11], or more specific: the Rural Postman Problem or the Vehicle Routing Problem [14]. Because only one resource is considered, there is no parallelism. The choice from resource alternatives and task precedence alternatives for resources is also widely discussed in literature, and is referred to as the (Generalized) Job Shop Scheduling problem (JSS) [16]. In this context, time required between the tasks gets very little attention. Using JSS, optimisation of the calibration case can be addressed. A sequential realisation would take 38.8 time units (Figure 2a), whereas an optimised realisation takes only 32.8 time units (Figure 2b), which gives an improvement of 15%. Due to the combinatorial nature of these scheduling problems, they are NP-hard to solve [13]. TRS definitions in terms of integrity constraints are analogous to rules of games [2]. Application of game theory in this domain is unknown.

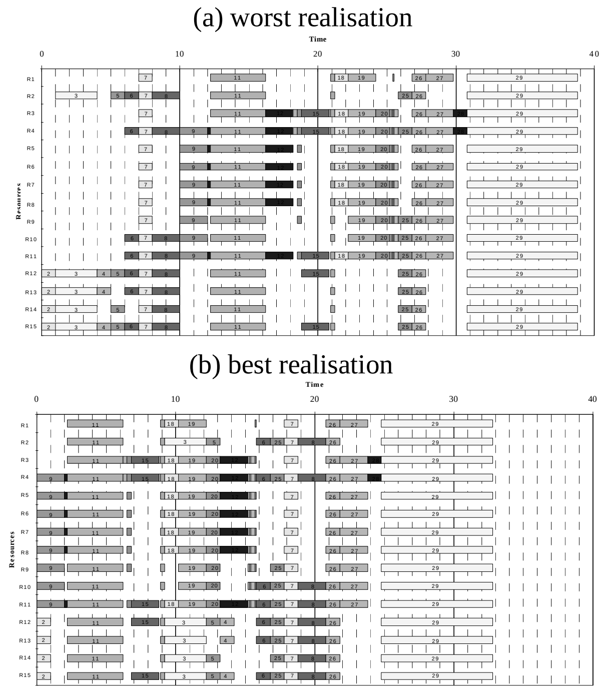

Figure 2: Case: wafer scanner calibration

However, the machine-specific issues of logistic integrity and physical limitations are not well addressed in scheduling literature. The same goes for recognition of patterns concerning repetitive behaviour for different instances of the same manufacturing entity.

Usage of predictions based on a model in a control system is known in control theory literature as model-based predictive control [12]. The underlying line of thoughts can also be used in supervisory control. In some cases, a well-founded decision can only be made if future behaviour can be predicted, for instance, by a model of the system that is embedded in supervisory control.

5 Conclusions

Machines are hybrid systems: discrete-event and continuous-time. In the context of supervisory control, the continuous behaviour of the machine between discrete states can be captured by durations of tasks. Therefore, system definitions based on tasks and resources are suited for description of the dynamic behaviour of machines.

TRS definitions are classified by the level of restriction of the control choices. In the case of unselected TRS definitions (2), some machine-specific restrictions are applicable: hardware limitations and logistic integrity constraints. Within the context of supervisory control, a further relaxation considering task precedences can be obtained by definition of integrity constraints for all relevant aspects.

Scheduling can be performed statically or dynamically. Dynamic scheduling allows better performance than static scheduling with respect to timing behaviour and robustness for exceptions. In the design process, trade-offs should be carefully evaluated considering the aspects mentioned and the complexity of the resulting supervisory control.

Techniques to support the design of supervisory control originate from several research areas. Generic techniques from computer science are not well suited for analysis of hybrid models, not even of class (1), because optimisation is involved. Dedicated mathematical functions assigning the minimal duration to resource state transitions are better suited, as they simplify the problem to the class of discrete-event systems. Again, generic techniques from computer science are not well suited for analysis of practical cases described by unselected TRS definitions (2), because of the state-space explosion problem. Although the choices result in an NP-hard problem anyhow, state space for model checkers is generally even bigger. This is caused by the mapping onto a generic timed paradigm and language, and by the fact that patterns concerning different instances of the same manufacturing entity are not recognised. More dedicated techniques originating from operations research are better suited for analysis of unselected TRS definitions. They stick to the bare problem and have intelligent search algorithms for optimisation purposes. However, the machine-specific issues concerning hardware limitations and integrity constraints (3) are not well addressed in literature.

The following open issues remain. To efficiently reason about system definitions with alternatives, techniques are needed for pattern recognition concerning different instances of the same production entity. Moreover, techniques are required that properly cope with machine-specific restrictions. To achieve better machine performance, model-based predictive supervisory control concepts should be developed. Additionally, the impact of integrating integrity constraints -related to game theory concepts- in supervisory control on exception-robust behaviour should be investigated.

6 Acknowledgments

The authors would like to acknowledge Cor Hurkens for his useful comments.

References

[1] www.asml.com

[2] R.J. Aumann, S. Hart. “Handbook of game theory: with economic applications”, Amsterdam, North-Holland, (2002).

[3] J.C.M. Baeten and W.P. Weijland. “Process Algebra”, *Cambridge Tracts in Theoretical Computer Science,* **no. 18**, Cambridge University Press, (1990).

[4] D.A.van Beek, J.E.Rooda. “Languages and applications in hybrid modelling and simulation: positioning of chi”, *Control Engineering Practice*, **vol. 8, no. 1**, pp. 81-91, (2000).

[5] P.R. Belanger. “Control Engineering: a modern approach”, Saunders College Publishing, Fort Worth, (1995).

[6] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. "A system of patterns", Chichester, Wiley, (1995).

[7] L.M. Ellram, S.P. Siferd. “Total Cost of Ownership: a key concept in Strategic Cost Management”, *Journal of Business Logistics*, **vol. 19, no. 1**, (1998).

[8] S. Gaubert and J. Mairesse. “Task Resource models and (max, +) automata,” *Idempotency*, Cambridge, U.K. Cambridge University Press, pp. 133-144, (1998).

[9] H. Gueguen, M. Lefebvre. “A comparison of mixed specification formalisms”, *Proceedings of ADPM2000*, pp. 133-138, (2000).

[10] C.M.H. Kuijpers, C.A.J. Hurkens, J.B.M. Melissen. “Fast movement strategies for a step-and-scan wafer stepper”, *Statistica Neerlandica*, **vol. 51, no. 1**, pp. 55-71, (1997).

[11] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B, Shmoys. “The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization”, Chichester, Wiley-Interschience, pp. 177-178, (1985).

[12] J.M. Maciejowski. “Predictive control with constraints”, Harlow, Prentice Hall, (2002).

[13] M. Pinedo. “Scheduling: Theory, Algorithms, and Systems”, Prentice Hall, Englewood Cliffs, (1995).

[14] P. Toth, D. Vigo. “The Vehicle Routing Problem”, Philadelphia, SIAM, (2002).

[15] G.X. Viennot. “Heaps of Pieces, I: Basic Definitions and combinatorial lemmas,” *Combinatoire Enumerative*, Labelle and Leroux, Eds., **no. 1234** in Lect. Notes in Math., New York: Springer, pp. 321-350, (1986).

[16] M. Wennink. ”Algorithmic support for automated planning boards ”, PhD thesis, Eindhoven University of Technology, (1995).