Designing the Brains of Complex Manufacturing Systems

By: Bram van der Sanden

Efficient manufacturing of products has become more important than ever in our global society. Have you ever thought about how these systems themselves are designed? Manufacturing systems that produce these products are becoming increasingly complex. They need to be flexible, in order to quickly adapt to changing market demands. Products need to be produced faster, while ensuring their quality. All these constraints make the design of present-day manufacturing systems a true challenge.

Achieving the economic and quality goals of manufacturing requires the use of a sophisticated controller that directs the scheduling of operations in the system and handles on-line changes in the production process. The controller in a sense functions as the brains of the system, and controls its behavior. In this article, we describe how these controllers can be automatically generated from domain-specific models that specify the system and its requirements.

Domain-Specific Models

To make a good controller design, we first have to know the manufacturing system for which we are designing a controller. The behavior of these systems becomes so complex, that it cannot be totally understood by a single system designer or operator. To deal with the complexity, we use model-based engineering, where we create formal and executable models of the system. These models describe precisely the possible system behavior, i.e. what the system can do, and the requirements, i.e. what the system should do. In the end, the complete model is composed of many building blocks. To understand the full system behavior, we need to understand the individual parts and how they interact.

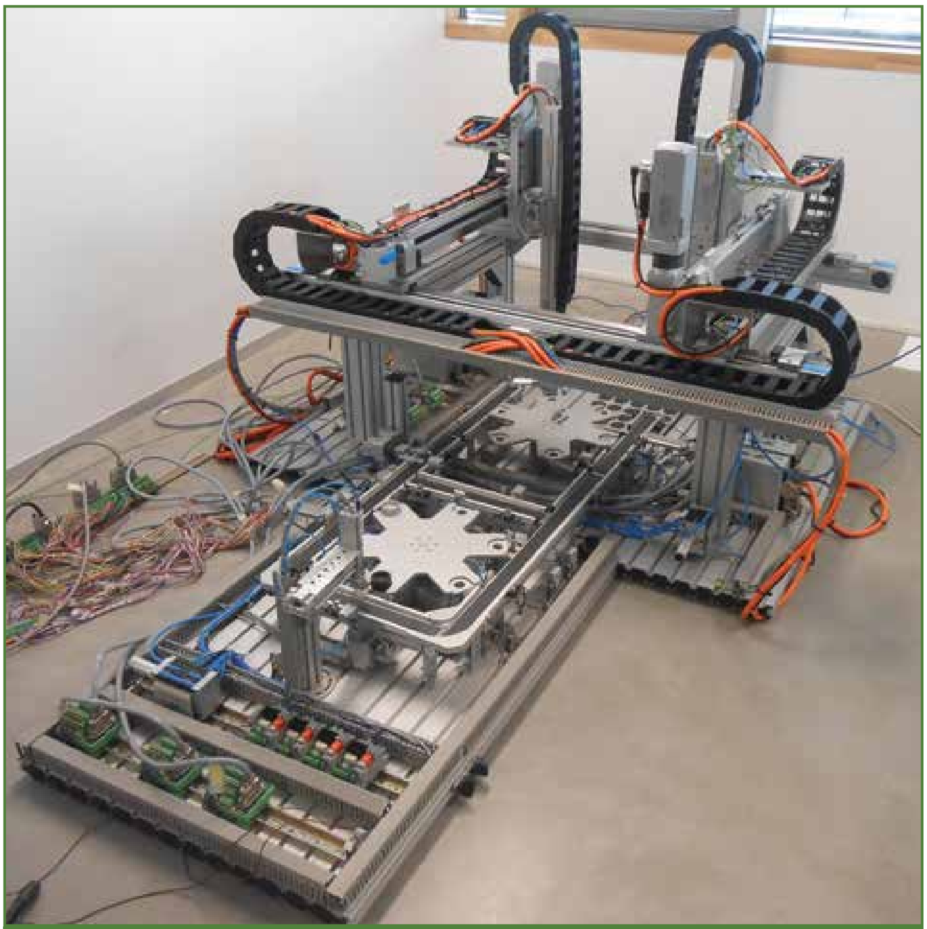


Figure 2: xCPS platform, a small-scale machine mimicking a production line to assemble and disassemble objects.

Then we can study the complete behavior using computer simulations and interactive visualizations.

The models are used throughout the development process of the controller. Since the models are formal, we can provide guarantees on functionality and performance if the system is built accordingly. After specifying the possible system behavior and the requirements, we can automatically generate a controller that enforces all requirements upon the system. After including timing details of the operations, we can also optimize this controller according to one or more performance indicators. For example throughput, which is an indicator that specifies how many products per time unit are produced. Figure 1 shows this work flow.

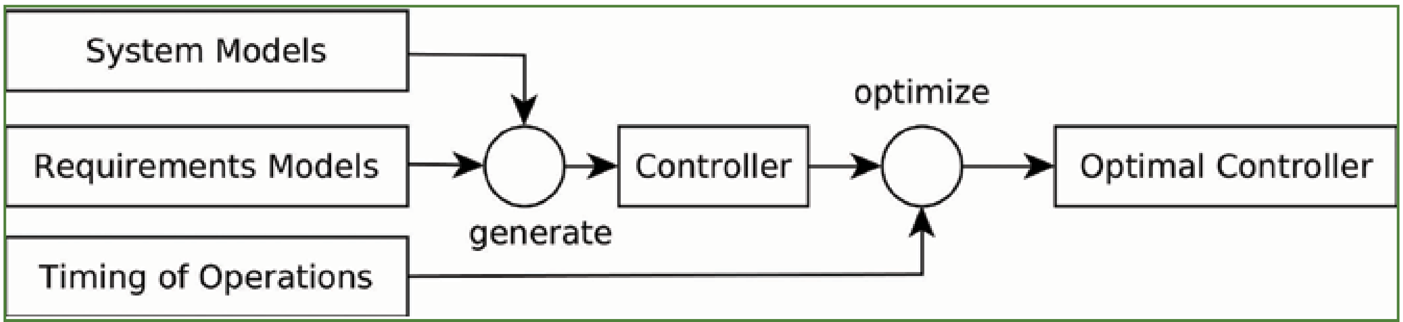


Figure 1: Work flow to automatically generate an optimal controller.

An advantage of the proposed work flow is that we can do early design-space exploration, where we compare different designs of the same system. This is enabled by the model-based approach. If we adapt the model to formalize a different system configuration, we can generate a new controller for the system with a push of a button. This new controller can then be compared with other controllers for different system configurations. In the next sections we describe each step of the work flow in more detail.

System Models: What can the system do?

The first step in the design flow is modeling the capabilities of the system to describe what the system can do. This means that we define the components in the system, and the actions that each component can perform. We also model system activities, that may contain multiple actions and dependencies between those actions. These activities can be scheduled by the controller. An example of such an activity is performing an operation on a product, or picking up a product at one processing station, and bringing it to another processing station.

As an example, consider the xCPS platform (www.xcps.info) shown in Figure 2. This platform mimics a production line capable of assembling and disassembling objects. One of the components in this platform is the turner, shown in Figure 3, which can turn objects. To model this turner, we need to model both the actuators and sensors. The actuators are the motor of the vertical arm, the gripper, and the rotator. Two sensors are modeled to detect when the vertical arm is up or down. The Turn activity specifies the order of actions to correctly turn a piece. First, we ensure that the vertical arm is down. Then, we pick up the product by closing the gripper and moving up the arm. When the arm is up, we can rotate the piece. After rotating, the arm is moved down, and the gripper is opened to release the now correctly oriented product.

Given the operations in the system, we describe the behavior of the system in terms of system states and transitions between these states. We use state machines to formally capture these states and transitions. The information of these states is for instance related to which processing stations are currently processing some product, or the location of products. The transitions represent the operations that can be executed from a given system state. For instance, when a product is on a processing station, this station can start processing the product. Since we use a modular approach to describe the system, we define different state machines for different system components and their behaviors. This allows us to consider each aspect in isolation. If one looks at the full system behavior, all these different state machines interact and influence each other.

To get a full system specification, we also model the environment of the system. The environment can change the system state, and the controller has no control over these changes. For example in the xCPS platform, the environment influences the orientation of pieces that enter the system. Dependent on the orientation of a piece, a Turn activity needs to be executed. To distinguish the controller and the environment, there are two types of transitions in the state machines. Transitions with operations are called controllable transitions, and transitions with actions of the environment are called uncontrollable transitions.

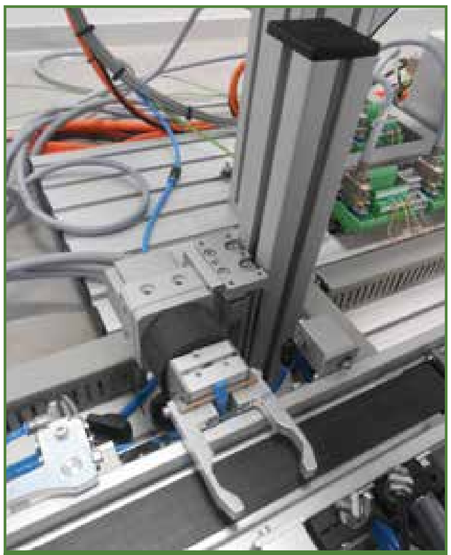


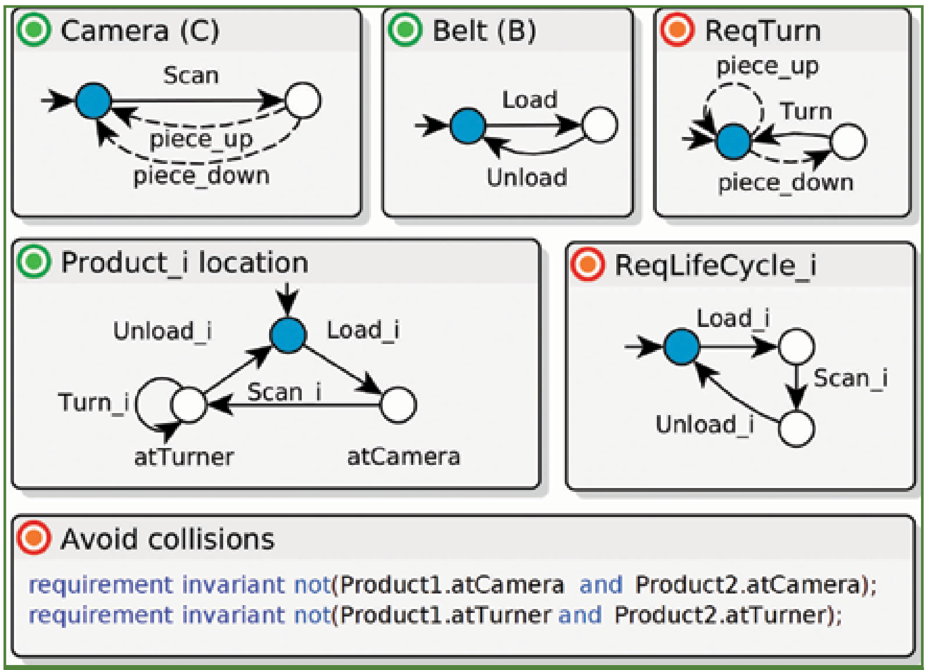
Figure 3: Turner in the xCPS platform and the corresponding models.

Requirement Models: What should the system do?

To generate a controller for the system, we need to specify the requirements that the controller needs to enforce. These requirements specify in which order certain operations or actions are allowed to occur, and thereby restrict the behavior of the system to prevent undesirable behavior. For instance for safety, we want to ensure that two robot arms do not start moving towards each other and collide, or that products are damaged when placed on top of each other. Requirements also prescribe the life cycle of operations, such that the machine only outputs a product once it has gone through all production steps in the right order.

Generating a controller

We have a method to automatically derive from the state machine models of the system and the requirements, a state machine model of a controller that respects all requirements, if such a controller exists. Therefore, no model of the controller has to be made by hand. The generated controller model enforces the requirements on the system, and is correct-by-construction. Figure 5 shows the controller for our example. ▶



```
requirement invariant not(Product1.atCamera and Product2.atCamera);
requirement invariant not(Product1.atTurner and Product2.atTurner);
```

Figure 4: System models of the camera and belt in the xCPS platform, and two requirements.

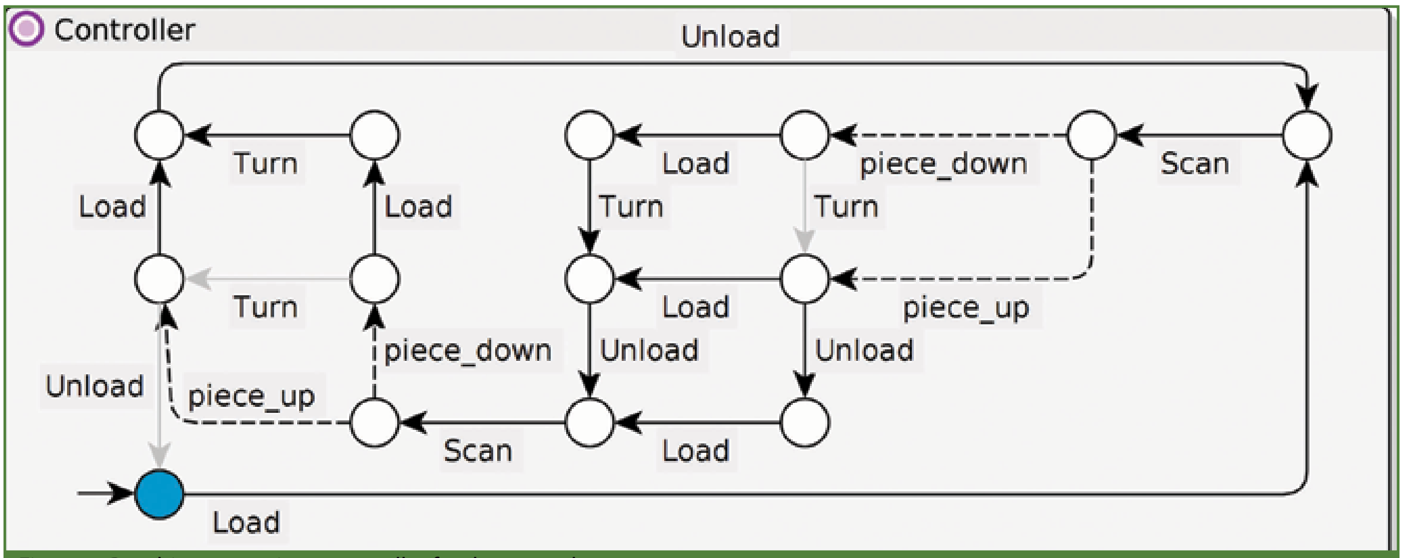


Figure 5: Resulting supervisory controller for the example.

We obtain a controller that allows all control decisions that do not violate any of the requirements. This means that in some system states, there can be multiple control decisions enabled. A controller implementation may select an arbitrary operation among the enabled ones.

Optimizing Productivity

After generation of a controller, there can be system states where multiple transitions are possible. In such states, we look for the optimal decisions that maximize productivity. To find these decisions in each system state, we include the timing information of the operations. Since transitions from the environment are not controllable, we can only choose the controllable transitions. The uncontrollable transitions all remain, because the controller cannot disable those. Therefore, we look for the best control decisions for productivity under consideration of all uncontrollable transitions.

In Figure 5, the optimal control decisions are marked in black. The gray transitions correspond to suboptimal control decisions. The optimal controller already starts with the load as soon as the scan of the previous product finished, instead of waiting for the whole system to be empty first.

Just like in describing the system, we also use state machines to capture the requirements. Each requirement is modeled as an individual state machine. This makes it easier to remove, add or adapt requirements.

To complete our example on xCPS, we introduce three additional system models and three requirements. These models are shown in Figure 4. The first two system models describe the behavior of the camera and belt respectively. The third system model keeps track of the location of a product with identifier I, and system activities involved with this product. Each product that enters the system is given an identifier. Activities with no identifier suffix in the model refer to any product. In the system we can have at most two products; one at the camera, and one at the turner. Therefore, the transition with activity Scan for example refines to Scan_1 and Scan_2.

The first requirement (ReqTurn) describes that a piece must be turned if it is down (piece_down), and not if it is up (piece_up). The second requirement (ReqLifeCycle_i) describes the work flow that a piece shall first be loaded, then scanned, and finally unloaded. The third requirement (Avoid collisions) makes sure that we do not have two products at the same location in the system. These requirements ensure that pieces will always be unloaded in the correct orientation, and no product collisions will occur.

Designing a controller for wafer scanners

We apply the proposed design flow at ASML, which is the world-leading provider of lithography machines. These machines (see Figure 6) are used in the production process of electronic chips. You can find those chips in your mobile phone, tablet, laptop, or smart television. Designing controllers for these systems is a challenging task, due to the large scale of the actions and operations, and all the use cases that the controller needs to be able to handle. In our research project, we focus on the product (wafer) logistics in these machines. The logistics controller ensures that each product is handled correctly, and optimizes production over time.

So far, we have created system models and requirement models of the logistics under assumption of nominal behavior. We take into account the life cycle of products, how many products can be at a processing station, robot collisions, product order requirements, and situations where the user of the system explicitly tells how the product should be handled. This model is linked to a formal model of the real system, that describes in more detail and with timing information what the system can do.

Due to the complexity of the problem, we have not yet been able to derive a controller that enforces all the requirements mentioned before upon the system. Currently, a controller can be generated if we leave out the opportunity for the user to specify how the product should be handled, and assume the same recipe for each product. In next steps, we want to extend the model to cover other use cases, tackle the challenge of controller generation, and realize the automatic generation of controller implementations from the controller models.

The work flow of automatically generating a controller has various advantages. It allows system designers to take a step back from unnecessary details, and focus on actually designing system requirements. Each system aspect is modeled formally, which forces the designer to be precise. The models can be validated already in an early stage, because they can be simulated and verified. Designers can investigate the impact of requirements, by choosing a selection from all requirements and looking at the system behavior that results after generation. Another use case is design-space exploration, where we can change the duration of certain actions and see the impact on the overall system performance, by repeating the step of finding an optimal controller.

Research

In the first two years of the research project, we have developed a formal modeling approach to specify the system. We have shown that this approach can capture the requirements in wafer logistics in a very natural way. The resulting model maps intuitively onto the requirements of the system designers. We have also developed new algorithms to optimize a controller model for throughput.

There are many research questions on the road ahead for the next two years. We want to investigate the scalability aspect of the automatic generation. In case of logistics in lithography machines, there are many requirements and use cases that have to be covered. This results in a large model, for which automatic generation is infeasible with the current techniques. There are ways that try to tackle this problem by restructuring the model, or generating a set of controllers that work together to control the system. Another interesting aspect is determining optimal control decisions. In the current flow we first look at all valid control decisions and then find the decisions that are optimal from a performance point of view. Combining these two steps might be more efficient. The model describing these optimal control decisions finally needs to be translated in a controller that can be implemented in software. In the end we would like to have a controller that is able to cover all use cases and optimizes productivity of the system.

Conclusion

In the design of the next generation manufacturing systems, we will no longer design our controllers by hand, but rather describe what it should do with a model. From this specification, we can automatically generate an optimal controller. The initial work flow already gives nice results, and we are working hard to solve the next interesting research questions to make it ready to be used by industry.

Further Reading

If you are eager to know more about the research, then you can have a look at the website www.bramvandersanden.com. In our research we use CIF3 (http://cif.se.wtb.tue.nl) for synthesis of controllers, and SDF3 (http://www.es.ele.tue.nl/sdf3/) for performance analysis. Some relevant papers are listed below.

van der Sanden, Bram, et al. "Modular Model-Based Supervisory Controller Design for Wafer Logistics in Lithography Machines." Model Driven Engineering Languages and Systems (MODELS), 2015 ACM/IEEE 18th International Conference on. IEEE, 2015.

van der Sanden, Bram, et al. "Compositional Specification of Functionality and Timing of Manufacturing Systems." Forum on specification and Design Languages 2016 (FDL).

Adyanthaya, Shreya, et al. "xCPS: a tool to eXplore Cyber Physical Systems." ACM SIGBED Review, Vol. 14, Issue 1, Oct. 2016. ■

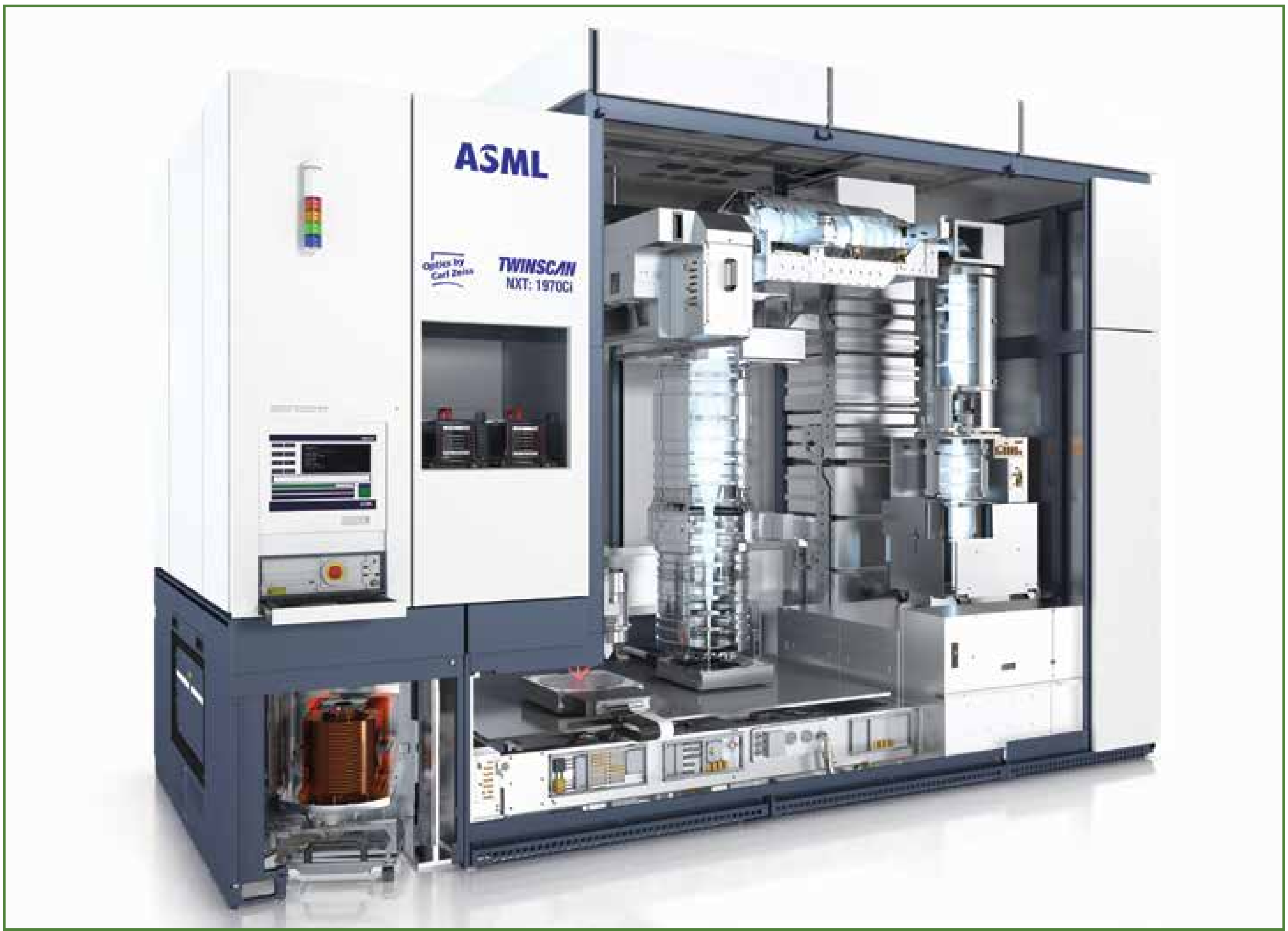


Figure 6: Lithography Machine of ASML.