Close

Model-Based Support for Integration and Testing of a Multi-Disciplinary Industrial System*

N.C.W.M. Braspenning[1**], D. Kostić[2], J.M. van de Mortel-Fronczak[1], and J.E. Rooda[1]

[1] Eindhoven University of Technology
Mechanical Engineering Department
P.O. Box 513
5600 MB Eindhoven, the Netherlands

[2] Embedded Systems Institute,
P.O. Box 513,
5600 MB Eindhoven, the Netherlands

Abstract. To reduce the integration and test effort for high-tech multi-disciplinary systems, we are developing a method called *model-based integration and testing*. The method allows integration of models of not yet realized components (e.g. mechanics, electronics, software) with available realizations of other components. The combination of models and realizations is then used for early system analysis by means of validation, verification, and testing. The analysis enables early detection and prevention of problems that would otherwise occur during real integration, resulting in a significant reduction of effort invested in the real integration and testing phases. This paper concerns the application of the method to a relevant industrial case study, involving several model-based analysis techniques that support in clarifying, evaluating, and improving integrated system and component designs, and in evaluating and improving candidate test cases. The models developed allow early prediction of integration and test problems, and the same models help in finding and fixing the root cause of such problems. The case study results contribute to the effort reduction for integration and testing, and encourage further research on the model-based integration and testing method.

INTRODUCTION

High-tech multi-disciplinary systems like wafer scanners, electronic microscopes and high-speed printers are becoming more complex every day. These systems consist of numerous hardware (e.g. optics, mechanics, electronics) and software components, connected through many interfaces. Furthermore, these systems have to meet the strict quality requirements set by the customer, in market conditions where lead time (in the context of time to market) is critical. The growing system complexity also increases the effort (in terms of lead time, costs, and resources) needed for the, so-called, *integration and testing phases*. During these phases, the system is realized by combining component realizations (implementations) and, subsequently, tested against the system requirements.

In current industrial practice, the system complexity problem is usually tackled by using a system development process with a ‘divide and conquer’ strategy, in which the system is decomposed into smaller components that are separately developed. In most cases, integration and testing are incremental processes that run in parallel with the development of the system components. Soon after the realization of a component becomes available, it is integrated with already assembled components of the system. Each integration increment is followed by corresponding testing. System integration and testing should be completed before the date of the system shipment agreed with the customer. Existing industrial practice shows that the main effort of system development is shifting from the design and implementation phases to the system integration and testing phases (Bratthall et al. 2000). Furthermore, finding and fixing problems during integration and testing can be up to 100 times more expensive than finding and fixing the problems during the requirements and design phases (Boehm and Basili 2001). In order to limit this trend of increasing integration and test effort, research is required that allows earlier and cheaper finding and fixing of integration problems (Prins 2004).

Improving the total system development process is the subject of *systems engineering* research (INCOSE 2004). While lots of the research in this area is focused on improving the requirements, the design and the implementation phases of system development, less attention is paid to the integration and test phases. In most cases, systems engineering research describes procedures and guidelines for system development, in which documentation is used for keeping track of and communicating about the system requirements and design. Unfortunately, documentation is often ambiguous and incomplete and the dynamic system behavior cannot be easily expressed in it. While using a document-based approach might help to improve the integration and test

* This work has been carried out as part of the TANGRAM project under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under grant TSIT2026.

** Corresponding author: n.c.w.m.braspenning@tue.nl

Close

phases, documentation can not be used to replace component realizations for early integration with other components. This means that the integration and test phases still require that the component realizations are available, and therefore these phases remain in their critical position.

An emerging alternative to using documents in the systems development process is to use computer models to represent the system components, and to use a range of model-based techniques and tools to support the system development process. There is a wealth of research relating to model-based techniques that aim at countering the increase of system development effort, like requirements modeling (Broy and Slotosch 2001), model-based design (Gomaa 2000, Liu et al. 2003), model-based code generation (Budinsky et al. 1996), and hardware-software co-simulation (Rowson 1994). In most cases, however, these model-based techniques are investigated in isolation, and little work is reported on combining these techniques into an overall method. Although model-based systems engineering (Ogren 2000) and OMG's model-driven architecture (Kleppe et al. 2003) (for software only systems) are such overall model-based methods, these methods mainly focus on the requirements, design, and implementation phases, rather than on the integration and test phases (which also holds for the research in document-based systems engineering). Furthermore, literature barely mentions realistic industrial applications of such methods, at least not for high-tech multi-disciplinary systems.

Our research within the TANGRAM project (TANGRAM 2003) focuses on a method of *model-based integration and testing* (MBI&T for short), introduced in (Braspenning et al. 2006), in which model-based techniques are used to reduce the effort of integration and testing. In this method, executable models of system components that are not yet physically realized or implemented in software are integrated with available realizations and implementations of other components, establishing a *model-based integrated system*. Such integration is used for early model-based systems analysis and integration testing, which has three main advantages. First, the fact that it takes place earlier means that the integration and test effort is distributed over a wider time frame, which in turn reduces the effort to be invested during the real integration and testing phases. Secondly, it allows earlier (and thus cheaper) detection and prevention of problems that would otherwise occur during real integration, which also increases system quality at an earlier stage. Finally, the use of (formal) models enables the application of powerful (formal) model-based analysis techniques, like simulation for performance analysis and verification for proving correctness of a system model. These model-based analysis techniques help in clarifying and improving the decomposition of the system requirements and system design into the requirements and designs of the system components. While the former are usually clear and certain, the latter are often based on assumptions and difficult to formulate. Model-based analysis improves the insight into this system decomposition, which is beneficial for the quality of the system realization.

In this paper, we present the MBI&T method and we instantiate it for two different system views: concurrent discrete-event system behavior and continuous-time and discrete-time system behavior. We also give a practical illustration of the application of the method to a realistic industrial case study concerning the ASML wafer scanner (ASML 2006). In the case study, the focus is on model-based analysis that supports integration and testing of system realizations. Using comprehensive models of the system components, we illustrate the analysis of the integrated system behavior, and the early prediction of problems that are encountered during the real integration phase. Furthermore, we illustrate how the models were used for early validation/falsification of test cases, for diagnosing the root cause of a test case failure, and for design of better alternatives to the tests that were falsified.

The structure of the paper is as follows. First, the system development process currently used in industrial practice is described in the next section. Subsequently, the MBI&T method is presented and instantiated for different system views in the third section. The application and results of the method in the case study are described in the fourth section. Finally, the conclusions are drawn and discussed in the last section.

CURRENT SYSTEM DEVELOPMENT

In current industrial practice, the system development process is subdivided into multiple concurrent component development processes. Subsequently, the resulting components are integrated into the system. The development process of a component C_i consists of a requirements definition phase, a design phase, and a realization phase. Each of these three phases results in a different representation form of the component, namely the requirements, the design, and the realization of the component, denoted here as R_i, D_i, and Z_i, respectively. In the development process of a system S that consists of multiple components, for instance two components C_1 and C_2, the system requirements and system design, denoted here as R and D, respectively, precede the development processes of each of the components. The realization of system S is the result of the integration of realizations Z_1 and Z_2 of components C_1 and C_2. This integration is denoted as $\{Z_1, I_{12}, Z_2\}$, where I_{12} denotes the infrastructure connecting Z_1 and Z_2. Figure 1 shows a graphical representation of the development process of system S.

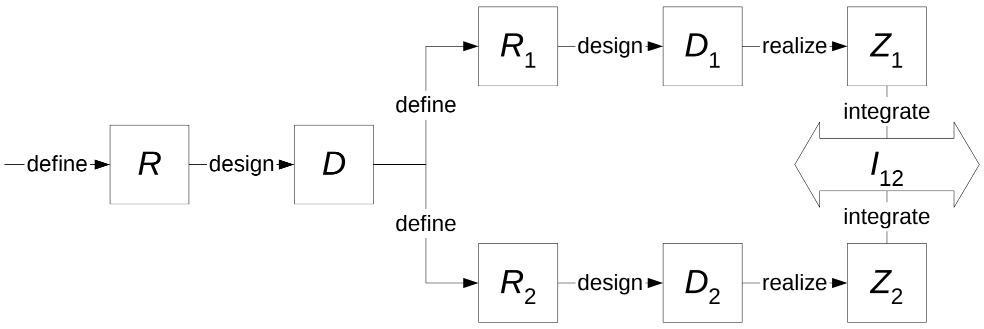

Figure 1. Current system development process

In this way of working, only two types of system level analysis can be applied. On the one hand, the consistency between requirements and designs on the component level and on the system level can be checked, e.g. R_1, R_2 versus R and D_1, D_2 versus D, which usually boils down to reviewing lots of documents. On the other hand, the integrated system realization, e.g. $\{Z_1, I_{12}, Z_2\}$, can be tested against the system requirements, R, which requires that all components are realized and integrated. This requirement that all components need to be realized and integrated means that when problems occur and need to be fixed, the effort invested in the integration and test phases immediately increases, directly threatening on-time system shipment.

For the development of high-tech multi-disciplinary systems such as the wafer scanners from ASML, the complex and multi-disciplinary nature of these systems obstructs specification of complete sets of requirements at each abstraction level (system, subsystem, unit). Completeness of requirements is especially difficult to achieve if emerging and not yet mature technologies have to be introduced in the system for the first time, because of limited experience with these technologies. Practice shows that obtaining detailed sets of requirements at the beginning of system development is a process which costs time, effort and money. Manufacturers of high-tech systems with time-driven business models, such as ASML, cannot wait until the process of completing the requirements is finished. In fact, they deliberately start designing even if some specifications are not yet known. During the design, realization, and integration phases, their understanding of the system improves, which helps to specify the requirements that are still missing. The testing phase already starts before the majority of the requirements are known, and, moreover, some specifications do not become clear until some tests are executed on the system that is integrated. One can observe a paradox in the fact that the testing can run even when not all requirements are known, while, by definition, testing should validate whether the requirements are met by the system or not. In practice, this paradox is resolved by engineers who, based on previous experiences, analogies, and engineering intuition, suggest test cases even for parts of the system that are not yet sufficiently covered by the requirements. For instance, a new design is analyzed at a brainstorm session of engineers trying to identify critical parts of the design and anticipate possible integration problems originating in these parts. Based on such an analysis, engineers give priority to test cases they assume capable of validating/falsifying consequences of anticipated integration problems. Practice shows that some of these test cases appear to be quite successful for the validation of the desired system operation at the system level and for clarifying some of the requirements that are still missing. Other test cases, however, yield little information and, as such, can be characterized as unfortunate waste of time, money and resources.

MODEL-BASED INTEGRATION AND TESTING

In (Braspenning et al. 2006), we introduced a method of model-based integration and testing to reduce the integration and test effort, which is an important aspect of the development effort. In this MBI&T method, depicted in Figure 2, the designs of the components (e.g. mechanics, electronics, software) are represented by executable models, denoted here as M_i for a component C_i. The infrastructure I_{12} allows the integration of components C_1 and C_2, both represented by either a model or a realization, such that all possible combinations of models and realizations can be integrated. Note that with code generation, the realization of a software component, Z_i, could also be based on its model M_i.

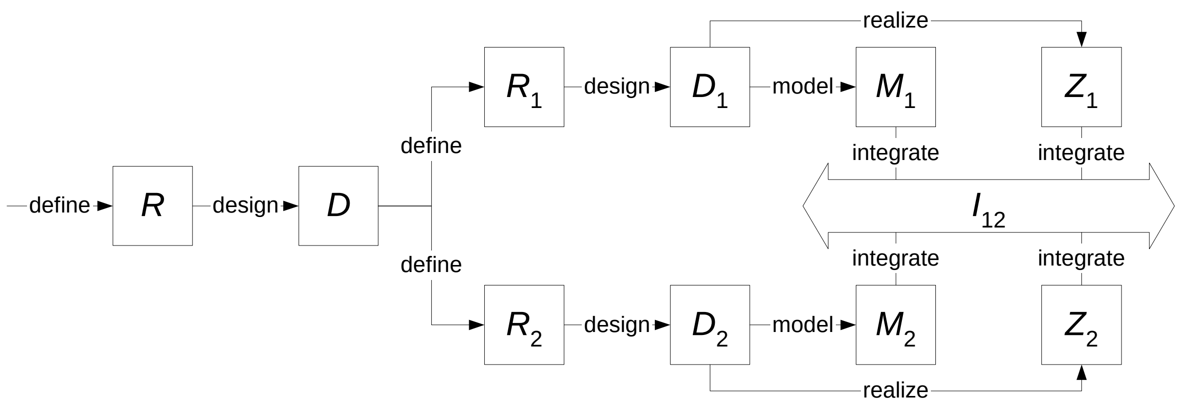

Figure 2. System development process in the MBI&T method

Several model-based techniques can be applied to analyze of the integrated system model, i.e. $\{M_1, I_{12}, M_2\}$. First, model validation by means of simulation can be used to inspect the behavior of certain traces of the system model. The simulation results can be compared with the intended system design D, and checked against properties derived from the system requirements R and the system design D (for single traces only, so not proving the property for the complete system). In order to prove the correctness of a system model in general, verification (e.g. model checking) can be used, in which the validity of a given property (derived from the system requirements R and system design D) for a given model of a system can be proven automatically. Analysis by validation and verification helps in evaluating and improving the correctness of the decomposition of the requirements and design of the system into the requirements and designs of the components.

Furthermore, test case simulation (i.e. simulation of the execution of test cases on models) enables the evaluation of whether certain test cases will produce the desired and expected test results when the test cases are executed during the real integration and test phases. These test cases include those derived from the system requirements, R, as well as test cases for requirements clarification and completion as discussed in the previous paragraph. With test case simulation, candidate test cases can be falsified when they do not produce the desired results or further improved in order to optimize the quality of the test results.

Finally, when model-based system analysis as described above has shown correctness of the system model, a model of a component can be used for both model-based component testing and for model-based integration testing. Model-based component testing involves automatic testing of the realization of a component against a model of the same component, and uses techniques and tools from model-based testing research (Brinksma and Tretmans 2001) for automatic generation of tests from the model and automatic execution of these tests on the realization. Model-based integration testing means that integrations of models and realizations are tested against the system requirements R and the system design D, i.e. without the necessity that all component realizations are available. As models are usually available earlier than realizations, testing on the system level can start earlier, and system integration problems can be detected and prevented earlier, which both contribute to a reduction of the effort invested during real integration and testing.

The procedure for the MBI&T method is as follows:

1. Modeling of components, e.g. M_1 and M_2, based on their designs D_1 and D_2.
2. Simulation, verification, and test case simulation using the model-based integrated system with models only, e.g. $\{M_1, I_{12}, M_2\}$.
3. As soon as the realization of each component becomes available:
    a. Replacement of model by realization, e.g. M_2 by Z_2, using an infrastructure that enables the integration with the other components.
    b. Model-based testing of component realization with respect to model, e.g. Z_2 with respect to M_2, using model-based testing techniques and tools.
    c. Model-based integration testing of combined models and realizations, e.g. $\{M_1, I_{12}, Z_2\}$.
4. After all models have been substituted by realizations: integration testing of the complete system realization, e.g. $\{Z_1, I_{12}, Z_2\}$.

In principle, the MBI&T method could support system development in any industry that has a separate development process for the (multi-disciplinary) system components, and for which it is difficult to perform thorough analysis on the system level without having a realized and integrated system available. To be able to use the MBI&T method in practice, however, the method needs to be instantiated with paradigms, techniques, and tools for all system views that are taken into consideration. The applicability of the method to a certain industrial system and its development process depends on the availability of appropriate paradigms, techniques, and tools for the system views that are important for the particular system and development process.

As mentioned in the previous paragraph, the MBI&T method has to be instantiated with appropriate paradigms, techniques, and tools. A paradigm is a way of thinking and reasoning about a system view, for which techniques (e.g. mathematical modeling formalism and analysis techniques) are needed that allow the specification and analysis of the system view according to the paradigm. To be useful in practice, the techniques should be implemented in modeling and analysis tools. The applicable analysis techniques for a system view depend on the paradigm, techniques and tools chosen to model that system view. For example, to be able to perform system analysis by means of verification, several requirements need to be satisfied. First, mathematical techniques are required to derive and analyze all possible behaviors (i.e. the state space) of a model. Secondly, the properties to be verified must be specified in some mathematical form (e.g. in a temporal logic). Finally, a tool should be available for efficient verification of these properties over the complete state space.

For the ASML case study considered in this paper, two views on the system under investigation are important. The first one, the *concurrent discrete-event system view*, is used for model-based analysis of concurrent behavior of all involved processes, in order to validate and verify whether the order of events and communication actions is correct for the whole system of processes. Furthermore, this analysis should ensure that the data handling of the system is correct, i.e. all data needed for each process is available and correct data is created by each process. The second system view, the *continuous-time and discrete-time system view*, deals with model-based analysis and prediction of an emergent system property: system performance. The analysis is focused on the system component which is the most critical for meeting the performance requirements. Performance analysis is facilitated by comprehensive kinematical, continuous-time, and discrete-time dynamic modeling of the critical system component, under the assumption that the event and communication order, as well as the data handling of the system, are found to be correct based on analysis of the first system view.

For the concurrent discrete-event system view, the paradigm of concurrent processes that communicate data is used. For static data modeling, graphical modeling techniques well known from structural system analysis (Whitten et al. 2001) are used, like Entity Relationship Diagrams (ERD) and Data Flow Diagrams (DFD). Many graphical drawing tools (e.g. the tools in the INCOSE Systems Architecture Tools Survey (INCOSE Tools Database Working Group 2006) and Microsoft Visio (Microsoft Corporation 2006)) support the design of logical ERD and DFD models, and many languages and tools like C (Kernighan and Ritchie 1988), Python (Python Software Foundation 2006), and Matlab (The Mathworks 2006) are suitable for implementation of the physical data models. Static data analysis is performed by checking certain rules of thumb for ERD and DFD models, and by using additional analysis techniques like Create-Read-Update-Delete (CRUD) tables (Politano 2001). The concurrent processes that communicate data are expressed in a timed process algebra called χ (Baeten and Weijland 1990, van Beek et al. 2005). For each component, the internal behavior of the process (assignments, guarded alternatives, guarded repetitions, delays) and the external communication of data with other processes (sending, receiving) are modeled. Subsequently, the integrated system is modeled as the parallel composition of all component processes, connected by communication channels. For analysis of such a system model, the χ toolset contains a simulator and back ends to the formal verification tools Spin, Uppaal, and μCRL (Bortnik et al. 2005). A tool called TorX (Tretmans and Brinksma 2003) is used for model-based component testing, which can also be applied in combination with χ models of a system (Braspenning et al. 2006).

For the continuous-time and discrete-time system view, the paradigms used are kinematics and dynamics. The mathematical techniques used for modeling include algebraic, differential and difference equations, homogeneous coordinates and transforms, transfer functions based on Laplace and Z transforms, state space models, and frequency response functions (Franklin et al. 2001). For analytical and graphical analysis and prediction of the system performance in the time domain or in the frequency domain, many techniques can be applied. These techniques include analytical techniques like sensitivity functions for a feedback control system and Laplace theorems on the initial/terminal values of a transfer function response (Franklin et al. 2001), signal processing techniques like Fast Fourier Transform and coherence functions (Pintelon and Schoukens 2001), and simulation techniques to determine responses to certain inputs. The common modeling and analysis tool for this system view is Matlab/Simulink (The Mathworks 2006) with the corresponding toolboxes.

CASE STUDY: LEVELING SUBSYSTEM OF AN ASML WAFER SCANNER

The MBI&T method, instantiated for the two system views as described in the previous section, has been applied to a realistic industrial case study from ASML, the industrial partner of the TANGRAM project. ASML is the world's leading provider of lithography systems for the semiconductor industry, manufacturing complex machines that are critical to the production of integrated circuits or chips. It is a medium sized company (around 5000 employees) with a large research and development department, which has the challenging task of developing and integrating leading edge technologies from many disciplines (optics, mechanics, electronics, motion control, software) into one system. The system development process of ASML is very time-driven; providing customers with leading edge lithography systems that are production-ready at the earliest possible date is critical in this area of semiconductor equipment.

In an ASML wafer scanner, laser light transfers a lithographic image (a pattern corresponding to one layer of a chip) onto the surface of a silicon wafer. This light passes through a lens that shrinks the pattern image before it is projected onto the wafer. The leveling function is responsible for keeping the wafer surface, which is not perfectly flat on the micrometer scale, in focus of the lens during exposure. Keeping the wafer surface in focus is achieved by the a priori measurement of the wafer topology using an appropriate leveling sensor and subsequent correction by wafer movements in accordance to the calculated height compensation during exposure. Currently, a new sensor for the leveling function is being developed to improve the accuracy of the leveling subsystem. Figure 3 schematically shows the leveling subsystem of the wafer scanner with all components mentioned above.

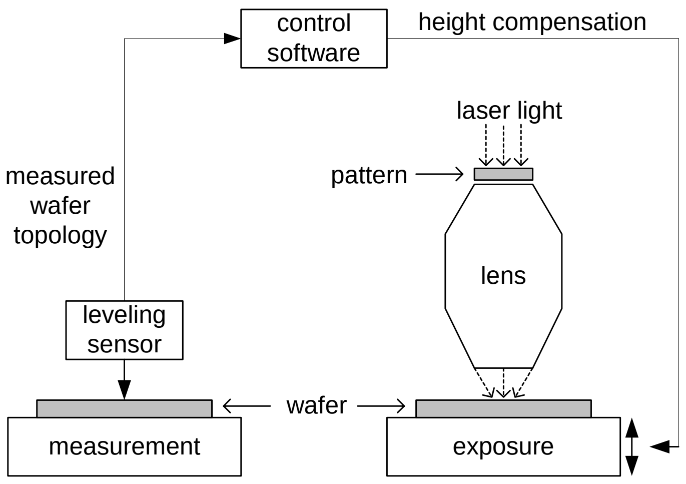

Figure 3. Leveling subsystem with involved components

In the case study, we apply the MBI&T method and accompanying techniques and tools to the integration of the new sensor into the existing leveling subsystem. Note that the activities in this case study do not cover the complete MBI&T method, we particularly give a practical illustration of steps 1 and 2 of the procedure described in Section 3, involving early evaluation of the system design by model-based analysis techniques. We also demonstrate model-based support to the real integration and testing in step 4 of the procedure, by evaluating and improving candidate test cases and by helping in the uncovering of integration problems. Since the focus in this case study is on model-based techniques that consider only models, activities in the context of step 3 (where both models and realizations are used for early analysis) were not covered. However, activities in the context of step 3 were successfully applied in other ASML case studies, in which the capability of detecting and preventing design and integration errors at an early stage has been shown. Case study results for model-based component testing (step 3b) have been reported in (Braspenning et al. 2006), while case study results for model-based integration testing will be reported in the future.

As mentioned in the previous section, two views on the leveling subsystem are considered in the case study: the concurrent discrete-event system view and the continuous-time and discrete-time system view. The case study activities for the first system view involve modeling and static/dynamic analysis of the data and the processes involved in the complete leveling subsystem, corresponding to steps 1 and 2 of the MBI&T method:

Activity I. Modeling of all data involved in the leveling subsystem and static analysis of the logical data model.

Activity II. Modeling of the processes involved in the leveling subsystem (including a physical version of the data model from activity I), dynamical simulation, and verification of the system of concurrent processes.

While activities I and II cover the complete subsystem (using a rather high-level and abstract model of the data and processes), the case study activities for the second system view involve lower level and more detailed modeling and analysis (again corresponding to steps 1 and 2 of the MBI&T method). The activities cover only the leveling subsystem components that are critical for the emergent system property, the leveling performance. Besides using the models for early model-based analysis of the system design, they are also used to support the real integration and testing phases, corresponding to step 4 of the MBI&T method:

Activity III. Detailed modeling and model-based performance analysis of the critical leveling subsystem components, and test case simulation using this more detailed model.

Activity IV. Supporting the real integration and test phases with model-based generated test cases and model-based support to uncover integration problems. Note that this test case generation and diagnosis are model-based but not automatic, although other research within the TANGRAM project addresses these topics (Braspenning et al. 2006, Pietersma et al. 2004).

In the leveling subsystem, different disciplines and technologies coincide: physics, mechanics, electronics, computer hardware, software, etc. Intrinsic heterogeneity complicates detailed understanding of this subsystem, especially during the development of new system generations that introduce the latest technologies influencing the leveling function. Each cutting edge technology may feature phenomena that are not yet thoroughly explained. Moreover, combining heterogeneous technologies induces unintended coupling effects that cannot easily be anticipated, while these effects may have a serious impact on the requirements at the system level. Due to uncertainties intrinsic to the latest technologies as used in ASML systems, problems may arise during the integration and testing phases and system tests may fail because of faults whose existence has not been foreseen during the system development. For instance, the sources and mechanisms of some faults are not known, the integration of some components does not operate according to performance and reliability requirements, or unintended couplings of unknown underlying mechanisms are not predicted in advance.

On the one hand, remedies for the given problems can be found in pragmatic solutions: for instance, installing additional shields to isolate magnetic and thermal interferences, or applying empirically determined inputs for drift and offsets correction. Engineering practice has invented a rich toolset of such pragmatic solutions that cover a wide range of problems. Once they realize that a problem encountered during integration and testing allows a solution from the given toolset, engineers apply this solution as the most straightforward remedy they have at that moment. However, one should be aware that pragmatic solutions do not eliminate the problem's cause but only deal with its consequences. Since these solutions do not require a deep understanding of the problem's origins, they cannot prevent the reappearance of similar problems in future system releases, especially since these solutions are tuned for given operational conditions and are hardly effective if operational conditions are changed. Limited generality of pragmatic remedies prevents their direct reuse in different system releases. Their implementation always requires extra time, resources, and effort for customization and adaptation to new circumstances and operating points.

On the other hand, problems encountered during the integration and testing can be tackled by getting a more detailed understanding of the problem cause. Once the cause has been identified, design of appropriate remedy should become more straightforward. Also, knowing the cause helps preventing it in new system designs. This systematic and theoretically grounded approach requires extra effort in system modeling, analysis, simulation, and experimentation during the system development processes preceding the integration and testing. However, this extra effort can be more than compensated for by the advantages of faster, cheaper, and higher quality accomplishment of the integration and testing phases.

An example of the difference between this pragmatic and systematic approach to integration and test problems was experienced in activity IV of the case study. The four case study activities mentioned above are discussed in more detail in the following.

Activity I: data modeling and analysis. This activity corresponds to steps 1 and 2 of the MBI&T method for only the data part of the concurrent discrete-event system view. In order to create a data model of the leveling subsystem, the requirements and design documentation of the leveling subsystem, corresponding to *R* and *D* in Figures 1 and 2, were examined. A general description of the leveling function is given below, where the data entities are in italics:

– A *lot* (a job definition for a wafer scanner) consists of multiple *wafers* with a *wafer id* and one *wafer* has multiple *wafer fields* with each a *field id* and *field position*.

– A *field id* of a *wafer field* corresponds to a certain type of *field* (as defined in the *lot*), which has a certain *field size*.

– For the leveling function, a *wafer map* of a *wafer* (the measured topology of the complete *wafer*) needs to be created, which consists of multiple *stroke maps* (the measured topology of a part (*stroke*) of the *wafer*) containing the measured height data of each *stroke*.

– A *stroke* has a *stroke id* and is defined by a *begin point* and an *end point*.

– Finally, the *height compensation*, required for the exposure of a *wafer*, is calculated from the *wafer map* of that *wafer*.

These entities and their relations were modeled (corresponding to step 1 of the MBI&T method) in an entity relationship diagram (drawn in Microsoft Visio using an appropriate template), as shown in Figure 4.

Close

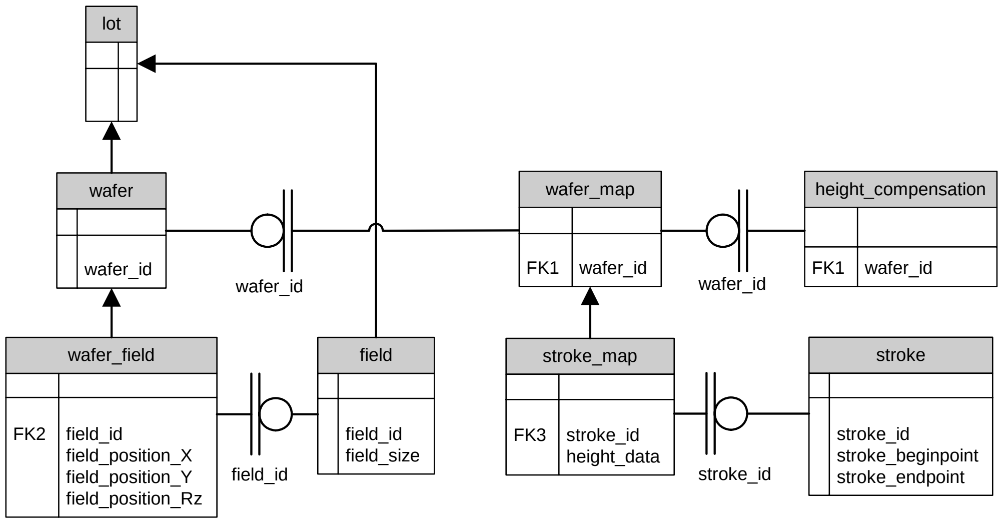

Figure 4. Entity relationship diagram of leveling subsystem

In the investigated system requirements and design documentation, no information could be found about which components are responsible for the handling of the data, for instance which component combines the stroke maps into a complete wafer map. This responsibility and the decomposition into processes and components was found in other, lower level documentation, corresponding to R_i and D_i in Figures 1 and 2. In this documentation, the functionality, behavior, and interfaces are defined for the following leveling subsystem components:

U The user, which provides the lot definition to the main controller

C The main controller, which controls all activities needed to process (measure and expose) the wafers of the lot definition provided by the user

RC The routing controller, which calculates all routings needed in the wafer scanner, e.g. strokes for topology measurement

MC The measurement controller, which controls all activities needed for the measurement of a wafer

EC The exposure controller, which controls all activities needed for the exposure of a wafer

LS The leveling sensor, which measures the height of the wafer surface

From this system decomposition and these component definitions, a process layout and a data flow diagram were also modeled in Microsoft Visio using appropriate templates. The process layout in Figure 5 shows the processes (circles) of the leveling subsystem and the interfaces (arrows) between the processes. The data flow diagram in Figure 6 shows all transformations (circles) on all data entities (arrows), as performed by one of the processes (indicated by the process name in front of each transformation). Note that the two data flows marked with a star, the 'stroke' data flow from RC to MC and the 'wafer map' data flow from MC to EC, pass through the main controller C. Furthermore, note that in both figures the creation and storage of virtual wafer data by main controller C is also depicted, which is not the case in the real system (in which the real wafer 'contains' the wafer data, and the leveling sensor LS 'reads' the wafer data from the real wafer). This virtual wafer data is added to the figures in order to let them correspond to activity II of the case study, where the virtual wafer data is used in system modeling and analysis.

Although the documentation that served as a basis for modeling was scattered, not complete, and contained several ambiguities, the activity of modeling the design described in these documents helps to clarify the design and indicates incompleteness and ambiguity issues. For example, we found that issues encountered during modeling (e.g. behavior that was difficult to model) were generally related to issues in the design as well. In this manner, valuable feedback from the modeling activities was provided to the involved engineers, and the quality of the documentation (and, accordingly, the model) was improved regarding completeness and ambiguity issues. Furthermore, by explaining and discussing the model with the involved engineers, the correctness of the model was validated against their 'mental model' of the system.

Close

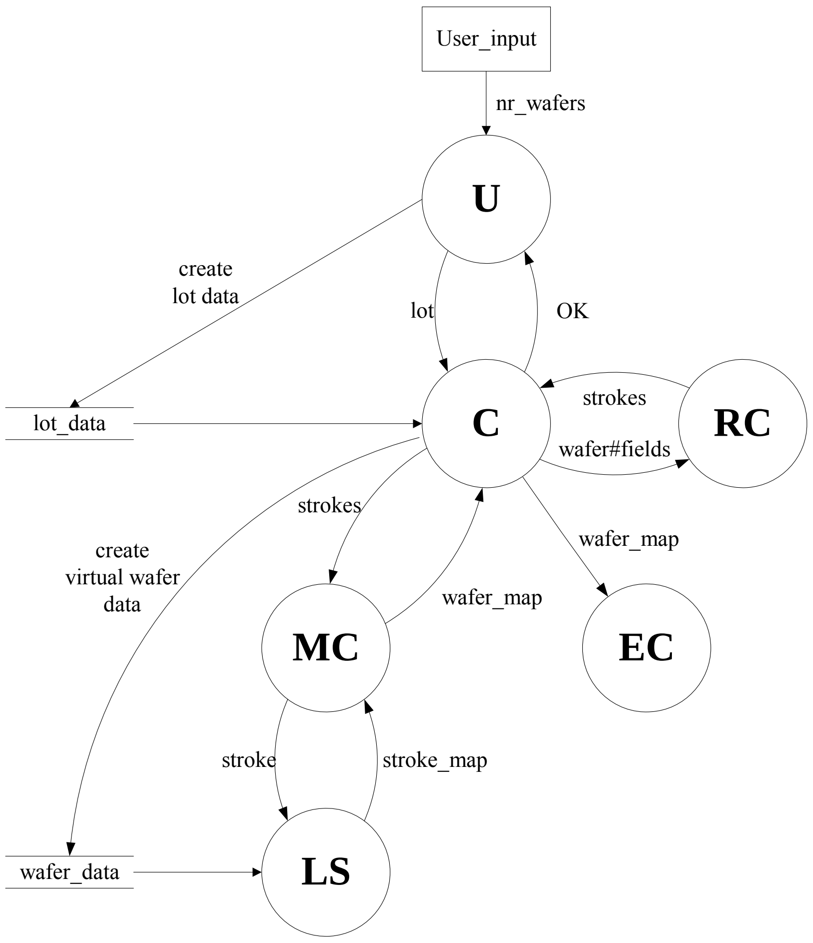

Figure 5. Leveling subsystem processes and their interfaces

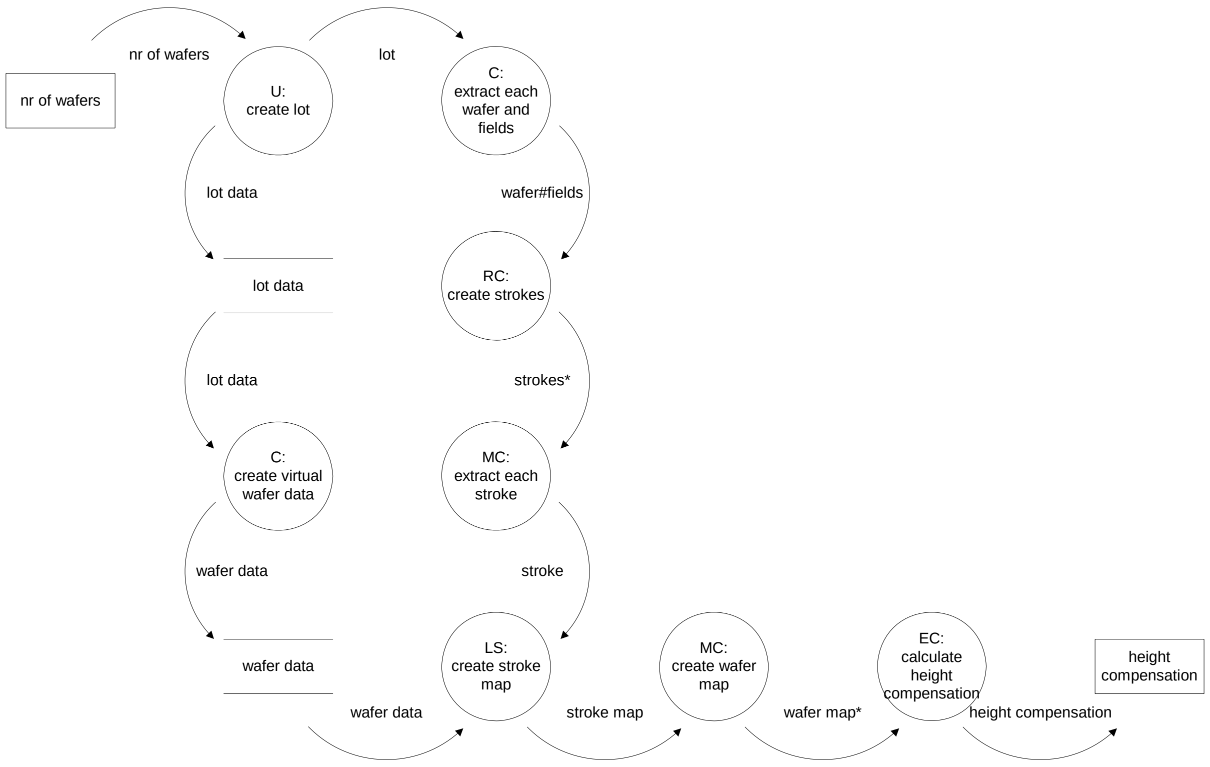

Figure 6. Data flow diagram of the leveling subsystem

Close

Corresponding to step 2 of the MBI&T method, the resulting entity relationship diagram, process layout, and data flow diagram were used for analysis of the system behavior. In general, the models help in clarifying the system behavior and give a better system overview when compared to the textual descriptions in the documentation, which is often not very well structured and therefore difficult to extract the system overview from. Static analysis of the diagrams shows that the necessary data entities are defined with clear relations between them, and that the process decomposition and interface definitions of all components are clear as well.

Another static analysis technique that was used in the case study is a Create-Read- Update-Delete (CRUD) table, showing which components (in the rows) create, read, update, and delete the data entities (in the columns). The CRUD table for the leveling subsystem is shown in Table 1.

Table 1. CRUD-table for analysis of interaction between processes and data

Component	Entity								
	Lot	Wafer	Field	Lot_data	wafer_data	stroke	stroke_map	Wafer_map	height_comp
U	CUD	CUD	CUD	CUD					
C	R			R	CUD		D	R	
RC		R	R			CU			
MC							R	CU	
EC								RD	C
LS					R	RD	CU		

Analysis of the CRUD table was done by checking certain rules of thumb that should be satisfied by a CRUD table (Politano 2001):

– There should be at least one Create for each data entity (data population)

– There should be at least one Delete for each data entity (data necessity)

– Each process should interact with at least one data entity (system granularity/completeness)

– Each data entity should interact with at least one system (process decomposition completeness)

– Multiple Creates for a data entity are not preferred (data quality)

– All Creates should be related to Reads and most entities should have at least one Create and more than one Read

– Grouping of data and processes indicate clusters in the system

The CRUD table in Table 1 satisfies these rules of thumb with one exception: the height compensation entity has no Read action, which is obvious because the usage of this entity (i.e. in the exposure process of the wafer scanner) is outside the scope of this subsystem. In the CRUD table, a cluster can be identified for the U, C, and RC processes, which handle the lot, wafer, and field entities. Besides checking these rules of thumb, a design decision was identified regarding the responsibility of calculating the height compensation for exposure from a measured wafer map. After process C receives the measured wafer map from process MC, it can either send the wafer map to process EC, which should then calculate the height compensation, or it can calculate the height compensation itself and send the results to process EC. In the leveling subsystem design, the second alternative was chosen, which is preferred because the main controller C is responsible for control tasks only, and the amount of calculation tasks should be minimized.

Summarizing the results of this activity, we can say that the ERD and DFD data modeling techniques used are well suited for modeling the system view considered, and that the static analysis techniques are capable of providing more insight into the data responsibilities of all system processes, compared to a textual description used in the documents. The resulting CRUD table can be used to check certain rules of thumb that possibly indicate design incompleteness and integration problems. Furthermore, the CRUD table supports in the identification of possible design alternatives regarding the data handling of the involved processes. Nevertheless, these analysis techniques are not capable of showing the dynamic behavior of the system, e.g. the ordering of events and communication actions. Modeling and analysis of the dynamic behavior of the leveling subsystem is covered in activity II of the case study.

Activity II: process modeling and analysis. This activity again corresponds to steps 1 and 2 of the MBI&T method, but now includes both the data and the process part of the concurrent discrete-event system view, in order to analyze the leveling subsystem for its dynamic behavior. The processes and data communications of Figure 6 were modeled in χ. For the underlying physical data model of Figure 4, some Matlab functions were included in the χ processes. Each χ process, described below, corresponds to a model M_i in Figure 2.

The User process U, which should provide the main controller with a lot definition, includes a Matlab function developed by ASML to create data structures that contain a lot with wafers and field definitions. The resulting Matlab data structures are converted into equivalent χ data structures. The same Matlab function also creates a virtual topology for each wafer according to the wafer and field definition. Using a virtual topology is sufficient for the model-based systems analysis in this case study activity, as the focus here is on the order of events and communication actions and on the correct data handling of all processes, and not on the correctness of the data itself (which is modeled and analyzed more extensively in activity III and IV of the case study).

The behaviors of the other processes, C, RC, MC, EC, and LS are modeled according to the requirements and design documentation of these components and can be described as follows (corresponding to the data flow diagram in Figure 6). After receiving a lot definition from process U, main controller process C subsequently stores the virtual wafer height data of the current wafer in the data store (to be used by process LS) and sends each wafer definition and the field definitions from the lot definition to routing controller process RC. Process RC uses these wafer and field definitions to calculate the stroke to be measured. The calculated strokes are sent through C to the measurement control process MC, which commands the leveling sensor LS to measure one stroke at a time. The leveling sensor measurements are based on the virtual wafer topology data in the data store. All measured stroke maps are concatenated by MC into a wafer map, which is returned to C. Note that the calculation of the height compensation from the wafer map is not modeled here, because it does not add much value to the model-based system analysis (under the assumption that when the wafer map is available, the height compensation can be calculated) and because the quite complex mathematics are not suited for the virtual wafer topology as generated by the Matlab function.

The integrated system model is obtained by parallel composition of all χ processes. Here, the parallel composition operator plays the role of infrastructure *I* (corresponding to Figure 2) that couples the modeled χ processes. Corresponding to step 2 of the MBI&T method, the behavior of this model-based integrated system is used for model-based systems analysis by means of simulation and verification.

Several simulation experiments were performed, where a lot with wafers is subsequently measured by the leveling sensor and the resulting wafer maps are received by C. All experiments showed a correct order of events and communication actions of the processes, resulting in an actual wafer map, indicating that the data handling of the processes is correct as well. Closer inspection of the wafer map (using visualizations similar to Figure 7 in activity III of the case study) shows that the strokes are correctly generated (concerning width, length, and measurement positions) and the sensor measurements are performed at the correct locations.

Unfortunately, verification of the leveling subsystem model was impossible due to the data complexity in the model. The verification tools for which translation schemes from χ have been developed, Spin, Uppaal, μCRL (Bortnik et al. 2005), only support simple and limited data structures and cannot handle the complex data structures that are used in the process model. For the same reason, these verification tools are generally not suited to verify the correctness of calculations on complex data structures. By applying sufficient abstractions from the complex data in the model, model verification might become applicable in order to verify whether all possible execution sequences of the abstract model are correct. However this was not further investigated in the case study, because the data abstractions require quite some effort and because the sequence in the model is rather fixed and can also be examined by sufficient simulation of the original model with the complex data structures.

Summarizing the results of this activity, we can say that the model-based way of working gives a good system overview and steers the engineer towards more accurate specifications, which in turn helps in the correction of inadequate requirements and in the capturing and clarification of requirements that were not known or unclear at the beginning of system development. Furthermore, static and dynamic analysis of the integrated system model provides support in evaluating design alternatives and allows early validation of the correctness of the data structure, data flow and data handling by all components of the system.

Since the model-based systems analysis in activities I and II of the case study, focusing on the system view of concurrent discrete-event behavior, has shown that the main system design was correct, there was no direct reduction of integration and test effort. This is mainly due to the fact that the considered part of the leveling subsystem is rather well-known and mature. Nevertheless, the applicability of early model-based systems analysis without the need for a system realization was shown. Furthermore, these case study activities show the potential value of the MBI&T method in the case that errors had been found during the model-based systems analysis.

Close

In activities III and IV of the case study, another system view on the leveling subsystem regarding the continuous-time and discrete-time system behavior was considered. Based on the results of the first two case study activities that showed correctness of the event and communication order and the data handling of the system, we now focus only on the system component that is most critical for the system performance, the new leveling sensor. Since this new sensor was still under development at the time of the case study activities, it provides a better context for showing the potential value of the MBI&T method, when compared to the part of the leveling subsystem considered in activities I and II.

Activity III: performance modeling and analysis. Similar to activities I and II, this activity corresponds to steps 1 and 2 of the MBI&T method, but now the modeling and analysis is focused on the second system view of continuous-time and discrete-time behavior. As mentioned earlier in this section, the case study considered deals with integration of the new leveling sensor which should increase accuracy of the leveling function. This accuracy represents the emergent performance characteristics of the leveling subsystem. In the case study, both pragmatic and systematic approaches to the development and integration of the new sensor were pursued in order to confront the advantages and the disadvantages of these approaches. To facilitate early assessment of the emergent property of the leveling function, comprehensive modeling of the new sensor was carried out. This modeling also covered elements of the leveling subsystem that have the most critical influence on the emergent performance property. On the one hand, the modeling was based on the sensor manufacturer's documentation and inputs from engineers who are responsible for the sensor's design and development. On the other, the modeling was based on the leveling documentation and inputs from engineers who are responsible for the leveling function. In the following paragraphs, all elements that were modeled are discussed in more detail: the leveling sensor, the wafer topology, the sensor output, the motion control, and finally the analog-digital filters.

The leveling sensor was modeled as a product of a static spatial sensitivity function and of a dynamic sensor's sensitivity. The sensor measures the average height of the wafer surface within the narrow spot which is directly below the sensor. The spot size is designed to be small enough to accommodate the required accuracy of the height measurements. The static sensitivity is represented by a scalar algebraic function of several position coordinates. The dynamic sensitivity is modeled with a continuous-time transfer function in order to facilitate application of different Matlab tools for dynamic system analysis.

The sensor model processes topology of the wafer surface within the encompassed spot. Wafer topology can be modeled using matrices that represent Cartesian coordinates of points on the wafer surface and surface heights at these points. For illustration, Figure 7 shows a field on a simulated surface which contains a sharp step in heights. The surfaces at lower and higher heights qualitatively reflect realistic wafer topologies. With proper tuning of matrix data, roughness of the modeled surfaces can also quantitatively describe realistic wafer topologies.

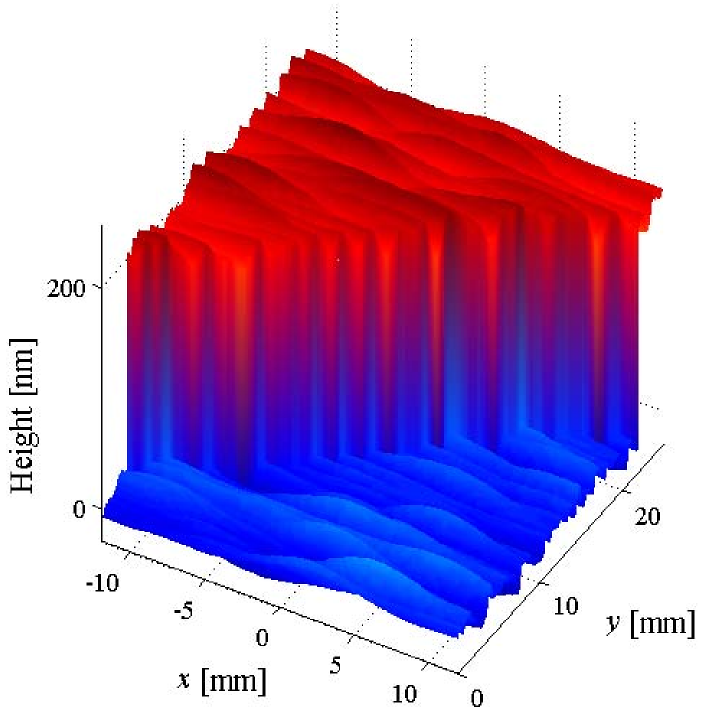

Figure 7. Topology of a field on wafer surface

Close

The output of the sensor model is determined as a convolution of the sensor sensitivity function (product of static and dynamic sensitivities) and of the wafer surface topology encompassed by the spot lying below the sensor. As the sensor measures wafer surface heights in the so-called scanning mode, i.e., while the wafer is moving with respect to the sensor, kinematics of wafer movements should be modeled as well. These movements are in general performed in six dimensional space: three Cartesian angles and three rotations. Conceptual kinematic setting for a wafer field and the sensor during the scanning is shown in Figure 8. Here, three coordinate frames are presented: of a wafer (blue), a reference coordinate system (red), and of the sensor (green). Transformations of Cartesian and angular coordinates among these frames are performed using homogenous coordinates and transforms (Fu et al. 1987). The scanning process was visualized by means of Matlab animations (Figure 8 is actually a snapshot of one such animation).

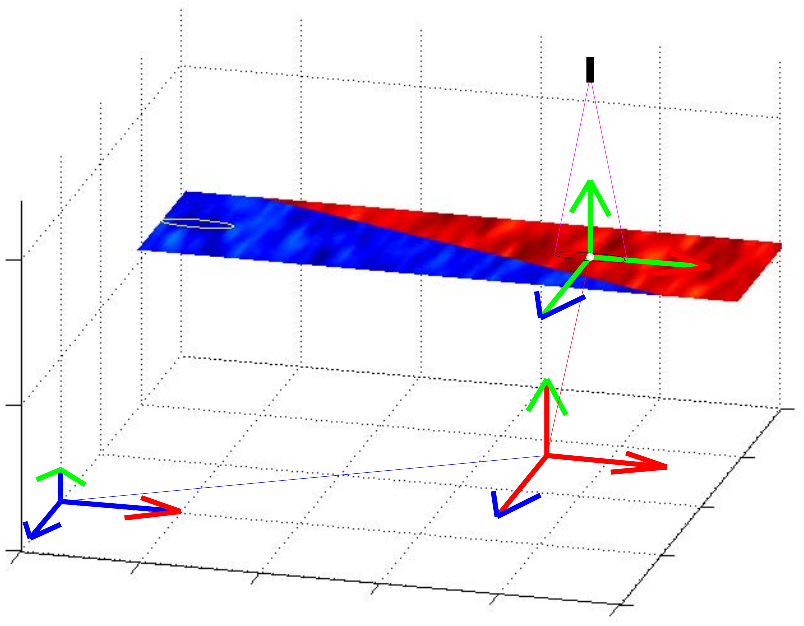

Figure 8. Kinematics of scanning a field on wafer surface

In a wafer scanner, movements of the wafer are achieved by means of an appropriate motion platform. This platform is a robotic mechanism in which motions are feedback controlled using advanced control laws. Because of the digital implementation, the control laws were modeled using difference equations. For convenience, these equations have subsequently been transformed into discrete-time transfer functions and state-space models. Similar mathematical formalisms (difference equations, discrete-time transfer functions and state-space models) were used for modeling dynamics of the motion platform. For illustration, Figure 9 shows a Simulink block diagram used for simulation of feedback controlled dynamics of the motion platform along one Cartesian axis of a wafer. Such a simulation enables analysis of the motion control performance (accuracy of wafer movements) in the time domain. The transfer functions of motion platform dynamics and of feedback controllers were used for analysis of control designs in the frequency domain (Franklin et al. 2001), for instance stability assessment based on locations of closed-loop poles, performance prediction by analysis of Bode plots of the open-loop gain, of the closed-loop sensitivity function, and of the closed-loop complementary sensitivity function.

Close

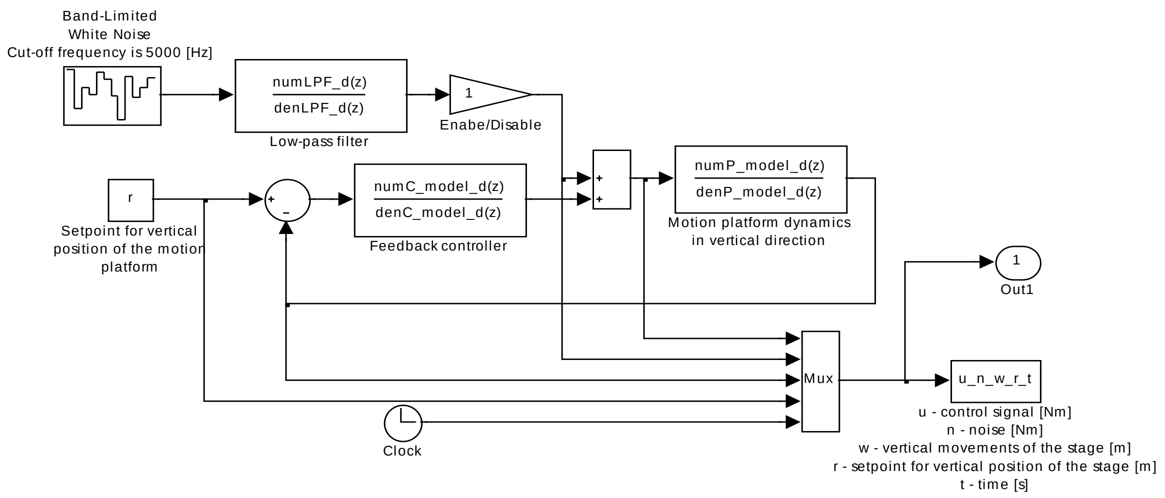

Figure 9. Simulation of feedback motion control in Simulink

The real sensor generates analog electrical signal proportional to the scanned wafer topology. This signal is digitalized via analog to digital conversion and then processed to filter out the measurement noise and to align the sensor readings with other data relevant to the leveling function. Signal processing is performed in software by a sequence of digital filtering operations. All filtering operations were modeled using difference equations, and these equations complete the set of models needed for performance analysis and prediction in our case study.

All derived models were implemented and integrated in Matlab and Simulink for the prediction and analysis of the sensor performance in the time domain. The integrated models were validated by simulating the behavior of the sensor for several typical wafer topologies, with expected results for the predicted sensor output signals. For illustration, Figure 10 shows typical plots generated by execution of the model. The plots on the left hand side present the scanned topography of a wafer surface and the analog sensor readings rendered by the model. The plots on the right hand side present the model-based predictions of outcomes of two signal processing steps to be performed in the software. Given plots reveal finite bandwidth of the sensor (low pass filtering characteristics), since its accuracy in measuring wafer topology variations is reduced as the frequency of these variations increases. The integrated models are configurable, i.e., the model allows for variations of wafer topology, scanning velocity, feedback control design, signal processing algorithms, etc. Configurable models facilitate analysis of how different behavioral aspects (e.g. kinematics, wafer topology, control design, filter design) influence the accuracy of the sensor readings, and consequently the leveling performance. Such model-based analysis has shown to be especially valuable for prediction of the leveling performance of the realized integrated system and for selection of test cases to be executed during the real testing phase.

Another effective usage of the Matlab/Simulink model of the leveling sensor lies in the testing of the leveling software, which was modeled in the first system view. Currently, testing the quality of the leveling software (e.g. the correctness of the wafer map and height compensation calculations) can be performed both offline and online. Offline testing means that the software is tested outside the wafer scanner, e.g. on a desktop system, using artificial sensor values, which has the disadvantage that the level of realism of the artificial sensor values (especially for any wafer topology) is unsatisfactory. Online testing, on the other hand, means that the leveling software runs on a wafer scanner and that custom made wafers with specific wafer topologies have to be measured by a real leveling sensor. The disadvantages of this test method are that a wafer scanner with a realized and integrated leveling sensor and the custom made wafers must be available, that the available machine time for testing is limited and very expensive, and that not only performing the tests but also setting up the wafer scanner for the tests can be very time consuming. The Matlab/Simulink model developed in this case study activity can be used for more extensive and more realistic offline testing of the leveling software, since the model can generate realistic sensor values for any virtual wafer topology, without the need for the expensive machine time used for setting up and performing the tests. The developers of the leveling software are actually using the leveling sensor model for this purpose, saving valuable machine time and costs.

Summarizing this step of the case study, we explained all aspects involved in the integrated leveling sensor model. The Matlab/Simulink model was validated by typical behavior simulation experiments and found to behave as expected. Now the model is available and validated, it can be used for several model-based analysis and testing techniques to support real integration and testing, as explained in activity IV of the case study.

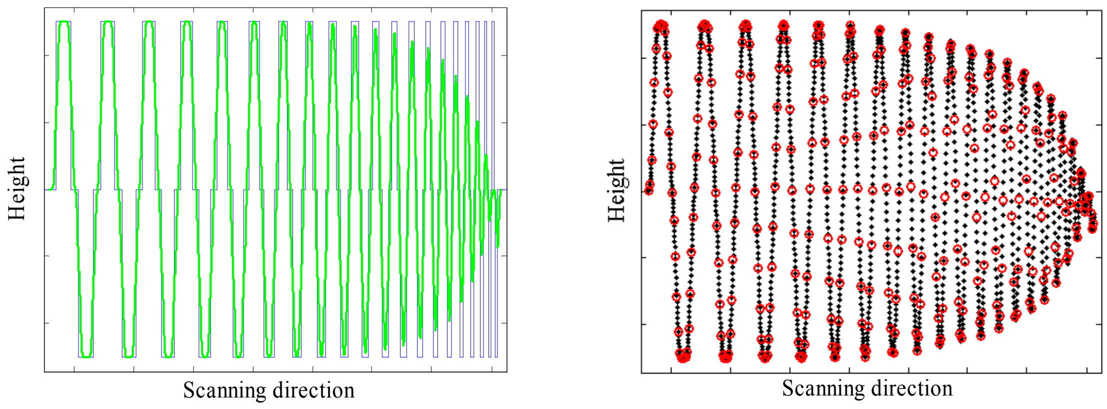

Figure 10. Model-based prediction of sensor performance: left - scanned surface topology (blue) and sensor model output (green); right - results of different filtering steps (points and circles)

Activity IV: model-based support to real integration and testing. This activity involves step 4 of the MBI&T method in the sense that we provide model-based support to the real integration and testing of the new leveling sensor. Our hypothesis is that model-based analysis can help in the selection of appropriate test cases in the presence of uncertainties in the behavior of an integrated system. To validate or falsify this hypothesis, in our case study we used the systematic model-based approach for test selection as an alternative to the pragmatic and experience-based way of engineers in industry. In this subsection, we show the illustrative results of confronting these two reasonings: systematic versus pragmatic.

The model of the new leveling sensor, as mentioned in the previous subsection, contains two components: static and dynamic sensitivity functions. One of the objectives of our case study is to obtain physical values of parameters of these functions. These parameters belong to the group of the system specifications that are not known during the system development, but are supposed to be determined during the phase of testing on the integrated system. Actually, these parameters are needed to enable model-based prediction and analysis of the leveling performance. To capture model parameters, suitable test cases need to be derived.

Here we report the derivation of the test cases for estimation of the dynamic sensitivity function. Describing a dynamical process, this function can be estimated by measuring the frequency response function (FRF) of the sensor (Franklin et al. 2001). The FRF measurements are standard in system identification theory. They are carried out by proper excitation of the dynamic process and by measuring the corresponding response of the process. The excitation is appropriate if it is rich in frequencies of interest for identification, i.e., its frequency content spans the frequency spectrum within which the dynamical process needs to be identified. The FRF of the dynamical system is computed based on the applied excitation input and the observed response data using algorithms reported in, e.g., (Pintelon and Schoukens 2001).

To perform the FRF measurements, documentation written by industrial engineers suggested a test case based on scanning a test wafer with a chirp-like surface profile. Such a profile is similar to the scanned surface topology shown by the blue plot on the right hand side of Figure 10. The given suggestion was based on common experience in system identification, since chirp-like excitation is often used for identification of dynamical processes (Franklin et al. 2001). To evaluate effectiveness of the test case with the chirp-like excitation, this test case was simulated on the model-based integrated system described in the previous subsection. As in the model-based integrated system, parameters of the sensor model are not known exactly, these parameters were assigned values based on some early design data. There was no guarantee that the design data coincide with parameters of the realized system.

Spectral contents of the excitation and response data obtained during model-based simulation of the pragmatically formulated test case are shown on the left hand side of Figure 11. In particular, the blue plot presents the power spectrum density (PSD) (Pintelon and Schoukens 2001) of the simulated scanned chirp-like surface topology, which has the role of excitation input for the FRF identification. The red plot is the PSD of the simulated sensor response based only on the static sensitivity function. Finally, the green plot is the PSD of the simulated sensor response based on both the static and dynamic sensitivity functions. By inspection of the given plots, one may observe that the static sensitivity acts as a low pass filter and itself contributes to the processing of the excitation data. The actual response, i.e., the green line, is thus the result of signal processing performed by both sensitivity functions. If such a test were applied on the real system, and the observed excitation and the measured sensor response were used to compute the FRF, then the determined FRF would incorporate both the static and dynamic sensor behaviors since the measured response is the result of both sensor behaviors. In other

Close

words, the determined FRF would deviate from the one of the dynamic sensitivity function. If industrial practitioners working on this identification experiment were not aware of the described consequences of the pragmatically formulated test case, then they would apparently incorrectly identify the sensor's dynamic sensitivity. If they were aware, then they might try to compensate for the contribution of the static sensitivity function. For this it is necessary to know the static sensitivity exactly. Moreover, the static sensitivity itself has low pass filtering characteristics, which limits the frequency spectrum within which the FRF of the dynamic sensitivity can be identified. Hence, knowing the static sensitivity is not sufficient for the identification of the desired FRF in the wider frequency range. Consequently, the performed model-based test case simulation has shown that the test case suggested based on the pragmatic reasoning was not as effective (informative) as the engineers wanted, which in turn falsifies this test case. As a result of this falsification, the test case considered was removed from the test set to be executed on the system realization, saving valuable test time.

The available model-based integrated system enabled us to determine a more effective test case. Such a test case is the one which decouples influences of the static and dynamic sensitivities. This can be achieved if the sensor measures distance to a flat field on a test wafer, while the wafer is vibrating in the direction orthogonal to the field. Since the wafer field is flat, simulated sensor output based on the static sensitivity is only identical to the actual distance between the sensor and the field, which in turn decouples the influence of the static sensitivity function from identification of FRF of the dynamic sensitivity. Having the role of excitation input, vibrations of the field should be in the reach of frequencies that are of interest for identification. This can be achieved by injecting the noise after the controller in the feedback loop responsible for wafer movements orthogonal to the flat field. Such a noise injection is used in the Simulink block diagram shown in Figure 9. Since the influence of the static sensitivity is decoupled, the sensor response is only due to the dynamic sensor sensitivity. Spectral contents of excitation and response data obtained during model-based evaluation of this alternative test case are shown on the right hand side of Figure 11. As expected, PSDs of the simulated excitation (blue) and of the response that is based on the static sensitivity only (red) are identical, and their plots cannot be distinguished from each other. Consequently, the frequency content of the simulated response (green) that is based on both the static and dynamic sensitivities depends only on the frequency content of the excitation. The FRF of the sensor dynamic sensitivity function can accurately be identified based on excitation and response data obtained using the suggested alternative test case. This test case to capture the unknown system requirements was designed based on systematic model-based analysis, which provided a better result than the dominantly utilized pragmatic approach. Execution of the alternative test case on the system realization indeed proved to be successful in identifying the required FRF.

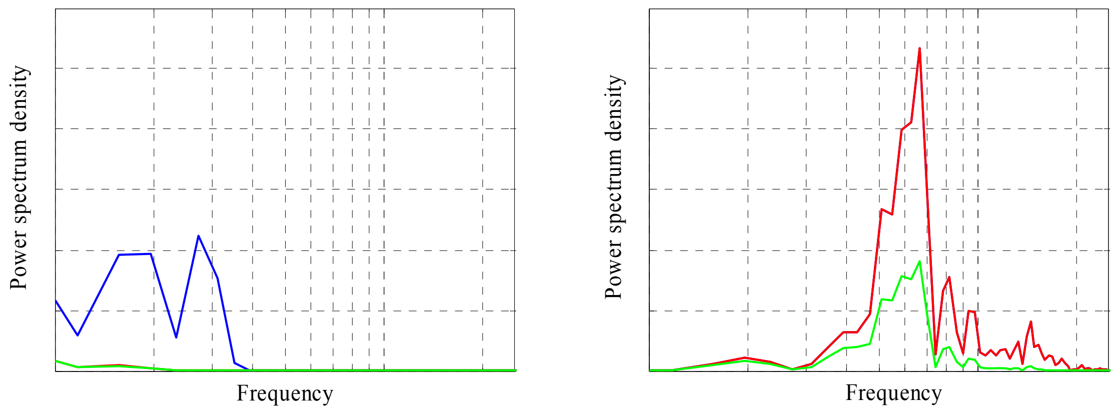

Figure 11. Frequency contents of excitation (blue) and response signals (red is based on static sensitivity only; green is based on both static and dynamics sensitivities) in test cases suggested based on: left - pragmatic reasoning, right - systematic reasoning

The systematic approach has shown advantages with respect to the pragmatic one regarding several more aspects in the considered case study. One of them was discovering the non-minimum phase behavior (Franklin et al. 2001) in the sensor dynamics after integration of the realized sensor in the leveling subsystem. The realized sensor has featured minimum phase behavior when tested in isolation. A change from minimum to non-minimum phase behavior has consequences for the sensor performance, i.e., for the accuracy of measuring the wafer topology. With the pragmatic way of working, it would be difficult to explain the cause of performance differences before and after integration. The systematic model-based analysis, however, has directly identified the non-minimum phase behavior as the cause of this difference.

The experience gathered in the considered industrial case study shows that the MBI&T method indeed improves the integration and testing phases of system development. The model-based analysis can falsify inadequate test

Close

cases and help determining more appropriate test alternatives, before the actual testing takes place. By early problem prevention, real testing can be accomplished with less effort. The systematic model-based approach can support real integration by faster uncovering the problems hidden in the integrated system.

Although it is very difficult to give a quantification of the actual reduction of the integration and test effort in the case study, we give, as an example, an indication of the time and cost savings for the FRF identification as discussed in activity IV of the case study and shown in Figure 10. In the current, pragmatic way of working, the only way to perform FRF identification would be to perform measurements on a wafer scanner equipped with a real sensor, using custom made wafers that contain a wafer topology suitable for the FRF identification (e.g. a chirp-like surface profile similar to the blue plot on right hand side of Figure 10). From testing experience, it is known that before a test on the machine can actually be performed, it takes quite some effort to setup the machine and to create the right conditions to perform the test. Furthermore, test results need to be analyzed and perhaps tests need to be repeated to get more confidence in the quality of the test results. Altogether, we estimate that performing the FRF identification at the machine would cost three to four hours of machine and clean room time, which is very expensive time. In the model-based way of working, a virtual wafer topology with the same chirp-like surface profile is used as input for the level sensor model. Setup time and execution time for the simulation of the model are in the order of minutes. Including the analysis of the simulation results and several repetitions of the simulation experiments, the FRF identification was completed in half an hour on a regular desktop system. Compared to the three to four hours of expensive machine and clean room time in the pragmatic way of working, the half hour of much cheaper desktop time clearly shows the time and cost reduction of the model-based way of working.

CONCLUSIONS

In this paper, the presented MBI&T method has been instantiated with paradigms, techniques, and tools for modeling and analysis of two system views: concurrent discrete-event system behavior and continuous-time and discrete-time system behavior. The instantiated method has practically been illustrated in a realistic industrial case study in order to show how a model-based way of working and model-based analysis techniques supports the integration and testing of an industrial system.

In the case study, the model-based way of working provided a better system overview and helped in clarifying and improving the component and system specifications, and consequently component and system quality. The availability of models enables powerful model-based analysis techniques that can be used as a support in evaluation of design alternatives and early validation and falsification of the correctness of the design. When applicable, model verification is able to prove the correctness of the system design. However, further improvements regarding complex data structures are needed to make model verification applicable for data intensive systems.

In the case study, inefficiencies of the standard, pragmatic way of working were revealed concerning the late finding and fixing of possible integration problems, implying an increased effort invested in the integration and test phases compared to the MBI&T method. The model-based way of working showed to be capable of predicting possible threats to the integration and testing by analyzing and subsequently falsifying inadequate test cases suggested based on pragmatic reasoning, saving valuable test time. Moreover, the systematic model-based approach provided a means to derive and analyze alternative and more effective test cases. Finally, model-based analysis assisted in the identification of the root cause of other integration problems, which reduces effort invested on fault diagnosis and fixing.

In the TANGRAM project, the MBI&T method has been successfully applied to several case studies in the domain of ASML wafer scanners. However, the method can be applied to any industry that has a separate development process for the (multi-disciplinary) system components and for which it is difficult to perform thorough analysis on the system level without having a realized and integrated system available. The prerequisite for a practical application of the method is that it needs to be instantiated with appropriate paradigms, techniques, and tools for the system views that are taken into consideration.

Even without a complete instantiation of the method, a systematic way of thinking and creating models of a system under investigation helps in clarifying the system behavior, in getting a better and more complete overview of the system, and in capturing, correcting, and improving requirements and design issues that were unknown at the start of the development of the system. Furthermore, it indicates incompleteness and ambiguity issues in the requirements and design documentation that are used as a starting point for modeling. However, the main advantage of the model-based way of working is that once a model of the system is available, many powerful model-based analysis techniques become applicable for a thorough system analysis by means of validation, verification, and testing, without the need for a completely realized and integrated system.

A risk in the method is that the investments in modeling are larger than the effort reduction achieved by it, which can be prevented by an appropriate risk assessment of applying a certain MBI&T method instantiation to

Close

a certain system. Risk assessment and choosing which components to integrate and test in which order and with which test method are important aspects of the overall integration and test strategy for a system, which is also a research area of the TANGRAM project (Boumen et al. 2006, de Jong et al. 2006).

The MBI&T method seems to be more effective for new, unknown systems that still have to be developed (e.g. the new leveling sensor in activities III and IV of the case study), rather than a well-known and mature system (e.g. the leveling subsystem software in activities I and II of the case study), at least for the model-based systems analysis part of the method. However, the part of the method that has not been addressed in this paper, the model-based integration testing part (step 3 of the method), may still be useful in systems of which some components are well-known and mature. For example, models may serve as a cheap and configurable test environment for acceptance testing or robustness testing of the realizations of the mature components, or models of alternative designs for a new component can be used for early integration testing of the system extended with the new component.

The objective of our model-based integration and testing research is to reduce effort invested in integration and testing of high-tech multi-disciplinary systems, while maintaining system quality. The results in this paper do contribute to this reduction, however it is expected that a more significant effort reduction can be achieved by early integration of models with realizations and by testing such model-based integrated systems. An example of this model-based integration testing is the integration of the leveling sensor model presented in this paper with the real leveling software, in which the leveling sensor model sends the predicted sensor signals for a virtual wafer topology to the leveling software for further calculations. This model-based integrated system is currently used by the leveling software developers to test the leveling software for a wide range of (virtual) wafer topologies at their desk, without the need for expensive machine time and custom made wafers that contain the specific wafer topologies. This integration of models and realizations and further improvements to the MBI&T method and its techniques and tools are the subject of our current work and will be reported in the future.

ACKNOWLEDGEMENTS

The authors would like to thank the engineers of the ASML leveling group for their support in the case study and all TANGRAM project members for their valuable comments and fruitful discussions.

REFERENCES

ASML, Website, 2006. http://www.asml.com.

Baeten, J.C.M. and Weijland, W.P., *Process algebra*. Volume 18 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1990.

van Beek, D.A., Man, K.L., Reniers, M.A., Rooda, J.E., and Schiffelers, R.R.H., Syntax and semantics of timed Chi. Computer Science Reports 05-09, Eindhoven University of Technology, March 2005.

Boehm, B.W. and Basili, V.R., Software defect reduction top 10 list. *IEEE Computer*, 34(1):135–137, January 2001.

Bortnik, E.M., Trčka, N., Wijs, A.J., Luttik, S.P., van de Mortel-Fronczak, J.M., Baeten, J.C.M., Fokkink, W.J., and Rooda, J.E., Analyzing a χ model of a turntable system using SPIN, CADP and Uppaal. *Journal of Logic and Algebraic Programming*, 65(2):51–104, November 2005.

Boumen, R., de Jong, I.S.M., van de Mortel-Fronczak, J.M., and Rooda, J.E., Test time reduction by optimal test sequencing, accepted for INCOSE 2006 International Symposium, July 9-13, Orlando, Florida, USA, 2006.

Braspenning, N.C.W.M., van de Mortel-Fronczak, J.M., and Rooda, J.E., A model-based integration and testing approach to reduce lead time in system development. In: Proceedings of the 2nd workshop on Model-Based Testing (MBT2006), March 25-26, Vienna, Austria, 2006. To appear in *Electronic Notes in Theoretical Computer Science*.

Bratthall, L.G., Runeson, P., Ädelsward, K., and Eriksson, W., A survey of lead-time challenges in the development and evolution of distributed real-time systems. *Information and Software Technology*, 42(13):947–958, September 2000.

Brinksma, E. and Tretmans, J., Testing transition systems: an annotated bibliography, in: *MOVEP 2000 – Modelling and Verification of Parallel Processes,* Lecture Notes in Computer Science 2067, pp. 187–195, 2001.

Broy, M. and Slotosch, O., From requirements to validated embedded systems. In *Embedded Software: First International Workshop (EMSOFT 2001)*, Tahoe City, CA, USA, pages 51–65. Springer-Verlag, October 2001.

Budinsky, F.J., Finnie, M.A., Vlissides, J.M., and Yu, P.S., Automatic code generation from design patterns. *IBM Systems Journal*, 35(2):151–171, 1996.

Franklin, G.F., Powell, J.D., and Emami-Naeini, A., *Feedback control of dynamic systems*. Prentice-Hall, 2001.

Fu, K.S., Gonzales, R.C., and Lee, C.S.G., *Robotics: control, sensing, vision, and intelligence*. McGraw-Hill, 1987.

Gomaa, H., *Designing concurrent, distributed, and real-time applications with UML*. Addison-Wesley

Professional, 1st edition, August 2000.

INCOSE, *Systems Engineering Handbook*, version 2a, June 2004.

INCOSE Tools Database Working Group, INCOSE systems architecture tools survey. Website, 2006. http://www.paper-review.com/tools/sas/read.php.

de Jong, I.S.M., Boumen, R., van de Mortel-Fronczak, J.M., and Rooda, J.E., Integration and test strategies for semiconductor manufacturing equipment, accepted for INCOSE 2006 International Symposium, July 9-13, Orlando, Florida, USA, 2006.

Kernighan, B.W. and Ritchie, D.M., *The C programming language*. Prentice-Hall, 2nd edition, May 1988.

Kleppe, A., Bast, W., and Warmer, J., *MDA Explained: The Model Driven Architecture: practice and promise*. Addison-Wesley Professional, 1st edition, April 2003.

Liu, X., Liu, J., Eker, J., and Lee, E.A., Heterogeneous modeling and design of control systems. In *Software-Enabled Control: Information Technology for Dynamical Systems*, chapter 7, pages 105–122. Wiley-IEEE Press, May 2003.

The MathWorks. Website, 2006. http://www.mathworks.com/.

Microsoft Corporation. Visio website, 2006. http://www.microsoft.com/office/visio.

Ogren, I., On principles for model-based systems engineering. *Systems Engineering*, 3(1):38–49, February 2000.

Pietersma, J., van Gemund, A.J.C., and Bos, A., A model-based approach to fault diagnosis of embedded systems. In *Proceedings of the Tenth Annual Conference of the Advanced School for Computing and Imaging (ASCI)*, pages 189–196, June 2004.

Prins, M., Testing industrial embedded systems – an overview, INCOSE 2004, Toulouse, France, June 20-24, 2004.

Politano, A.L., Salvaging information engineering techniques in the data warehouse environment. *Informing Science*, 42(2):35–43, 2001.

Pintelon, R. and Schoukens, J., *System identification: a frequency domain approach*. IEEE Press, 2001.

Python Software Foundation. Python website, 2006. http://www.python.org/.

Rowson, J.A., Hardware/software co-simulation. In *DAC '94: Proceedings of the 31st annual conference on Design automation*, pages 439–440. ACM Press, 1994.

TANGRAM Project. Website, 2003. http://www.esi.nl/tangram.

Tretmans, J. and Brinksma, E., TorX: Automated model based testing, in: First European Conference on Model-Driven Software Engineering, AGEDIS project, 2003.

Whitten, J.L., Bentley, L.D., and Dittman, K.C., *Systems analysis and design methods*. McGraw-Hill, 5th edition, 2001.