A quantitative method to decide where and when it is profitable to use models for integration and testing

Systems Engineering Group
Department of Mechanical Engineering
Eindhoven University of Technology
PO Box 513
5600 MB Eindhoven
The Netherlands
http://se.wtb.tue.nl/

SE Report: Nr. 2007-14

A quantitative method to decide where and when it is profitable to use models for integration and testing

N.C.W.M. Braspenning, R. Boumen,
J.M. van de Mortel-Fronczak, J.E. Rooda

ISSN: 1872-1567

SE Report: Nr. 2007-14
Eindhoven, April 2008
SE Reports are available via http://se.wtb.tue.nl/sereports

Abstract

Industrial trends show that the lead time and costs of integrating and testing high-tech multi-disciplinary systems are becoming critical factors for commercial success. In our research, we developed a method for early, model-based integration and testing to reduce this criticality. Although its benefits have been demonstrated in industrial practice, the method requires certain investments to achieve these benefits, e.g. time needed for modeling. Making the necessary trade-off between investments and potential benefits to decide when modeling is profitable is a difficult task that is often based on personal intuition and experience. In this paper, we describe how integration and test sequencing techniques can be used to quantitatively determine where and when the integration and testing process can profit from models. An industrial case study shows that it is feasible to quantify the costs and benefits of using models in terms of risk, time, and costs, such that the profitability can be determined.

1 Introduction

High-tech multi-disciplinary systems, such as wafer scanners from ASML [1] which are used worldwide to manufacture integrated circuits, consist of many components from different disciplines, e.g., optics, mechanics, electronics, and (embedded) software. The close interaction between components results in system behavior that aims at satisfying the system requirements, e.g., providing some system functionality with a defined performance. In the current industrial practice of developing such systems, the main effort of system development is shifting from the design and implementation phases to the so-called integration and test (I&T) phases [2]. During the I&T phases, usually at the end of the system development process and on the critical path, the system is integrated by combining component realizations and, subsequently, tested against the system requirements. In the current industrial way of working, most of the system level problems can only be detected during the I&T phases, where the costs of repairing these problems are significantly higher than in earlier phases [3]. As a result, the I&T phases have a large and growing disadvantageous influence on the time-to-market and on the product quality, which are the main business drivers of lithographic equipment manufacturers such as ASML [4]. Our research within the TANGRAM project [5, 6] focuses on a *model-based integration and testing* (MBI&T) method which aims at a reduction of this disadvantageous influence.

In previous work [7, 8], we showed how formal and executable models can replace not yet realized system components (e.g., software, mechanics, electronics) allowing early integration and system testing, i.e., before the complete system is realized. Application of the MBI&T method to a realistic industrial case study showed that the use of models enabled early detection and prevention of several system design and integration problems, months before the real integration and test phases started. Although the model-based I&T activities form new possibilities to reduce the time-to-market or to increase the product quality, they also introduce additional costs, e.g., time needed to model the components. These investments in modeling need to be made before the actual benefits become clear, often without knowing if and to which extent the benefits outweigh the costs. In some cases, the investments in modeling are profitable, e.g., when the realization of a component is available only late in the development process or when testing with realizations is expensive. In other cases, it is wise not to invest in models but to perform the tests with realizations only, e.g., for mature or low risk components.

Making decisions on whether or not to use models for integration and testing can be supported by estimations on the risk involved in the system, on the development or delivery times and the availability of realizations, and on the costs of testing with realizations. In current industrial practice, the decision making process is usually based on personal intuition and experience. In this paper, we describe a quantitative decision making process that takes the costs into account to determine *where* and *when* the I&T process can profit from models. This quantitative decision is based on an I&T sequencing method [9], also developed within the TANGRAM project, in which 'assembly by disassembly' techniques [10] and sequential diagnosis techniques [11] are combined and adapted for application in the current industrial practice of high-tech multi-disciplinary system development. Although it is recognized in [12] that model-based activities like simulation should produce benefits that outweigh the costs of modeling, we are not aware of any method to determine the profitability of a certain model-based activity based on a quantification of its costs and benefits.

This paper is based on [13] and it is organized as follows. Section 2 first describes the current I&T process and distinguishes nine categories of I&T activities. Subsequently, this section shows *where* models can be applied in the current I&T process, by applying the MBI&T method and techniques to each of the nine categories. Section 3 shows *when* it is profitable to apply models in the I&T process, by quantifying the costs of various I&T processes using

the integration and test sequencing method from [9]. This paper extends the work described in [13] by including an industrial application of this quantitative decision making process, which is described in Section 4. Concluding remarks are given in Section 5.

2 Current and model-based I&T process

This section first describes the current I&T process, such as used at ASML, in more detail. Subsequently, we show which activities in the current I&T process can be supported by models. As an example, we use an I&T process that is common for many high-tech multidisciplinary systems: upgrading a system with new hardware and software to improve the system performance, e.g., adding a new sensor with accompanying control software to improve the measurement accuracy. In this example, the goals of integration and testing are to show the functionality and performance of the system upgrade as soon as possible, and to show that the system upgrade does not negatively affect the functionality and performance of the original system. We show how such a system upgrade is dealt with in both the current I&T process and the model-based I&T process.

2.1 Current I&T process

Let us consider an existing system that consists of several hardware and software components. The system is upgraded by implementing some new or improved functionality, which is denoted by a delta sign (Δ). To implement this Δ-functionality, certain components of the original system need to be upgraded, or new components need to be developed and added to the system. In our view, the development process of this Δ-functionality starts with the definition of the overall requirements R_Δ and the creation of the overall design D_Δ, as shown on the left-hand side of Fig. 1. After that, the software and hardware components for the Δ-functionality, denoted by ΔSW and ΔHW, respectively, are separately developed, starting with the requirements $R_{\Delta SW}$ and $R_{\Delta HW}$, followed by the designs $D_{\Delta SW}$ and $D_{\Delta HW}$ and, finally, the realizations $Z_{\Delta SW}$ and $Z_{\Delta HW}$. The right-hand side of Fig. 1 shows the integration of the Δ components $Z_{\Delta SW}$ and $Z_{\Delta HW}$ with the software and hardware components of the original system, considered here as one software component Z_{SW} and one hardware component Z_{HW}, respectively. The four components are integrated by means of an infrastructure I. In this paper, we abstract from the details and different forms of infrastructure described in [8] and only consider the generalized infrastructure I.

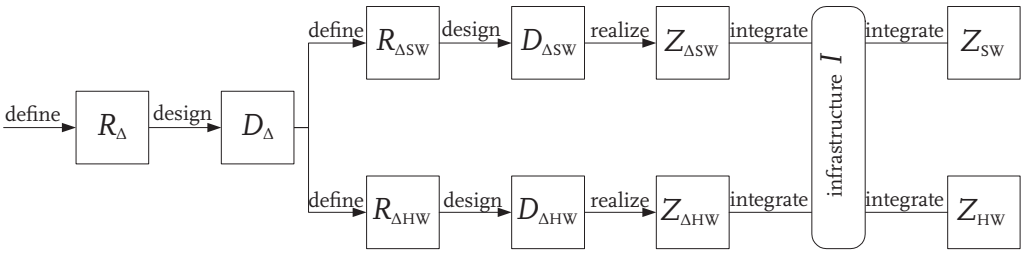


Figure 1: Current development and integration of a Δ-functionality

The four component realizations $Z_{\Delta SW}$, $Z_{\Delta HW}$, Z_{SW} and Z_{HW} can be integrated and tested in many ways during the I&T process. Within the current I&T process at ASML, nine different categories of I&T activities can be distinguished, which focus on different aspects of

the components or the system and require different combinations of realized and integrated components. These nine categories are listed in Table 1, where the integration of component realizations Z_i and Z_j by means of infrastructure I is denoted by $\{Z_i, Z_j\}_I$.

Table 1: Categories of I&T activities in current I&T process

Category	Required components	Explanation
1. SW qualification testing	$\{Z_{SW}, Z_{HW}\}_I$ and later $\{Z_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$	Periodic qualification of the so-called 'qualified baseline' (QBL) [14], a common repository for all new software developments that supports all machine types, by testing it on a set of representative hardware systems Z_{HW}
2. SW component testing	$Z_{\Delta SW}$	Testing the new software component in isolation
3. SW integration testing	$\{Z_{\Delta SW}, Z_{SW}\}_I$	Testing the new software component in combination with the original software system Z_{SW}
4. SW regression testing	$\{Z_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$	Testing whether any of the original system functions are negatively affected by the new software component, performed on the original hardware system Z_{HW}
5. HW component testing	$Z_{\Delta HW}$	Testing the new hardware component in isolation
6. HW integration testing	$\{Z_{\Delta HW}, Z_{HW}\}_I$	Testing the new hardware component in combination with the original hardware system Z_{HW}
7. Δ-functionality test bench testing	$\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}\}_I$	Testing the new Δ-functionality (also called progression testing) on a 'test bench', i.e., a partial hardware system including the new hardware component $Z_{\Delta HW}$, used for development tests
8. Δ-functionality system testing	$\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}, Z_{HW}\}_I$	Testing the new Δ-functionality on a complete system, i.e., Z_{HW} upgraded with $Z_{\Delta HW}$
9. System testing	$\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}, Z_{HW}\}_I$	Testing the functionality and performance of the complete system after all Δ-functionalities are integrated and tested, before system shipment

Fig. 2 shows a typical I&T process for a system upgrade. From left to right, the sequence of I&T activities, denoted by vertical lines, is shown. The numbers correspond to the nine categories of I&T activities in Table 1, and the dots indicate which components are integrated and tested. The horizontal lines depict the lifetime of each component: a dashed line means that the component is being developed; a flag symbol followed by a solid line means that the component realization is available. The flag symbols and the letters indicate the following milestones: (a) QBL Z_{SW} passes qualification tests; (b) development of $Z_{\Delta SW}$ and $Z_{\Delta HW}$ is started, possibly based on the original system (denoted by dashed upward arrows); (c) $Z_{\Delta SW}$ is available; (d) $Z_{\Delta SW}$ passes software tests and is integrated in the QBL Z_{SW} (denoted by downward arrow); (e) upgraded QBL $\{Z_{\Delta SW}, Z_{SW}\}_I$ passes qualification tests; (f) $Z_{\Delta HW}$ is available; (g) $Z_{\Delta SW}$ and $Z_{\Delta HW}$ pass test bench tests; (h) $Z_{\Delta HW}$ passes hardware integration tests and the hardware system Z_{HW} is upgraded to $\{Z_{\Delta HW}, Z_{HW}\}_I$ (denoted by downward arrow); (i) similar to the depicted Δ-functionality, the other Δ-functionalities are integrated and tested; (j) complete system with all Δ-functionalities passes tests and is shipped to customer. Note that the figure only shows a sequence and does not contain information on the possible start times and durations of the activities. For example, in the case that the hardware upgrade is available earlier than the software upgrade (i.e., milestone f before b), I&T activities 5 and 6 could be performed before I&T activities 2, 3, and 4.

Note that by removing particular components and related I&T activities from the system upgrade example, other I&T processes can also be represented. For example, a software only I&T process is obtained by removing the hardware components and all I&T activities that involve hardware. The I&T process of a completely new system is obtained by removing the original system software and hardware, implying that there is no initial software QBL, and that the first software component that is realized becomes the software QBL.

The main disadvantage of the current I&T process is that the I&T activities can only be performed when the realizations are available. Especially for testing on the system level (cate-

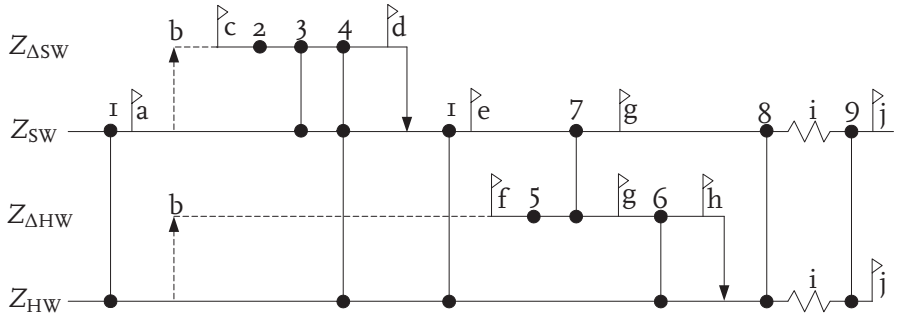


Figure 2: Typical I&T process for the system upgrade example

gories 7, 8, and 9) this is problematic, because it means that feedback on the system behavior and performance is obtained late in the process, where fixing the problems is expensive. In our research, we investigate how models can be used to enable *early integration and testing*, i.e., before all components are realized and integrated. The remainder of this section shows which activities of the I&T process can be supported by models, using the same system upgrade example.

2.2 Model based I&T process

Fig. 3 shows the development and integration of a Δ-functionality using the MBI&T method of [7]. Here, the models $M_{\Delta SW}$, $M_{\Delta HW}$, and M_{HW} are used as early representations of the Δ software component, the Δ hardware component, and the original complete hardware system, respectively. The reason for having M_{HW} but not M_{SW} is explained in [13]. The choice of integrating either the model or the realization of a component, or none of them, is depicted by the integration 'switches'.

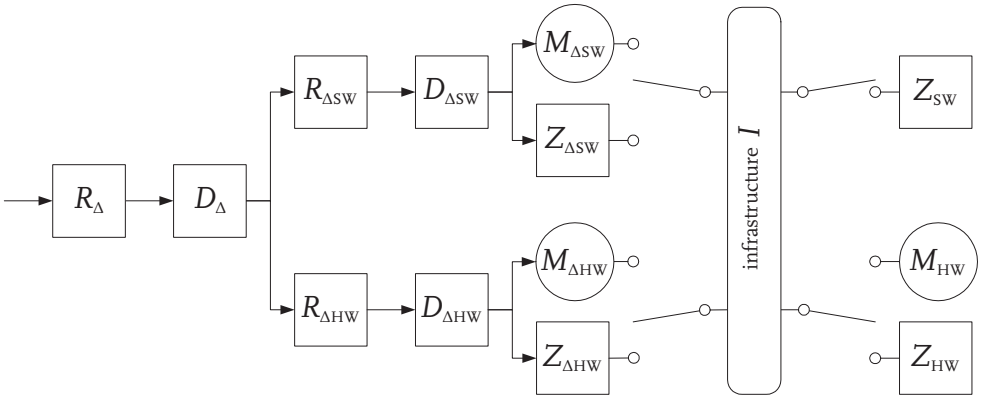


Figure 3: Development and integration of a Δ-functionality in the MBI&T method

When only models are integrated, the behavior of the resulting system model can be analyzed by several model-based techniques. For example, simulation techniques can be used to analyze particular scenarios regarding both nominal and exceptional system behavior. When the models have a formal semantics, formal verification techniques such as model checking [15] can be used to prove particular properties of the system, e.g., deadlock and livelock freeness, as well as system specific behavioral properties such as a required sequence of actions or a

required system performance. When both models and realizations are integrated, the models need to be executed in the realization environment, which involves real-time simulation and adapting the communication mechanisms used by the models to the communication mechanisms used by the realizations and vice versa. The resulting model-based integrated system is tested on the system level using tests derived from the system requirements R_Δ and the system design D_Δ.

An industrial application of these MBI&T activities to an ASML wafer scanner is described in [7] (simulation and model checking) and in [8] (model-based integration and system testing). These applications showed that the use of models for certain I&T activities has several advantages when compared to the current I&T activities in which only realizations are used. First, models are usually available before the realizations, which means that the total integration and test effort can be distributed over a wider time frame and that the effort to be invested during the real integration and testing phases at the end of the development process can be reduced. Second, the use of models enabled detection of system design and integration problems at an earlier stage of system development, which means that the corresponding diagnosis and fix costs are lower [3] and the quality of the system can be improved at an earlier stage. Third, formal models enable the use of sophisticated model-based analysis techniques such as simulation and verification, which increase the insight in the system behavior for the engineers, which in turn improves the system quality as well. Finally, looking at the costs of testing, many of the MBI&T activities can be performed on a common computer system using modeling and analysis software tools. The test costs in such a desktop environment are generally lower than the costs of realization tests, which, in the case of ASML, may require expensive machine time and cleanroom facilities.

Although models can support all nine categories of I&T activities as shown in Table 1, they cannot fully replace testing with realizations, since models are always abstractions of reality and usually do not cover all aspects of a component. For example, the models in the industrial application of [7, 8] focused on the interaction and time behavior of several wafer scanner components, which is suitable to test and detect problems related to these aspects but unsuitable to test other aspects such as the quality of the light needed to expose wafers in the wafer scanner. Sooner or later, when they become available, the realizations of the components and the system will be used to test at least the remaining aspects. However, these realization tests can probably be performed faster and with less costs, since the MBI&T activities already reduced some risk by detecting and preventing several problems. This was also experienced in the wafer scanner case study: in the period after the successful MBI&T activities for a particular wafer scanner component, no further problems were detected in the interaction and time behavior of this component, i.e., all test time could be spent on other aspects such as the quality of the light for exposure.

The following list identifies all possible I&T activities for each category, including both the current I&T activities from Table 1 and the new MBI&T activities of the MBI&T method. For each of the nine categories, the I&T activities with realizations only are marked with a 'Z', and the MBI&T activities are marked with an 'M', followed by a letter in the case of multiple MBI&T activities. Note that MBI&T activities 2Mb and 5Mb involve model-based testing techniques [16, 17], in which tests are automatically generated from a model M and executed on a realization Z, in order to determine the conformance of Z with respect to M. This paper only describes the MBI&T activities of categories 1, 4 and 7 in more detail. We refer to [13] for a complete overview.

1. Software qualification testing:
1Za: $\{Z_{SW}, Z_{HW}\}_I$
1Ma: $\{Z_{SW}, M_{HW}\}_I$
1Zb: $\{Z_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$
1Mb: $\{Z_{\Delta SW}, Z_{SW}, M_{HW}\}_I$

2. Software component testing:
2Z: $Z_{\Delta SW}$
2Ma: $M_{\Delta SW}$
2Mb: $Z_{\Delta SW}$ vs. $M_{\Delta SW}$

3. Software integration testing:
3Z: $\{Z_{\Delta SW}, Z_{SW}\}_I$
3M: $\{M_{\Delta SW}, Z_{SW}\}_I$

4. Software regression testing:
4Z: $\{Z_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$
4Ma: $\{M_{\Delta SW}, Z_{SW}, M_{HW}\}_I$
4Mb: $\{M_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$

5. Hardware component testing:
5Z: $Z_{\Delta HW}$
5Ma: $M_{\Delta HW}$
5Mb: $Z_{\Delta HW}$ vs. $M_{\Delta HW}$

6. Hardware integration testing:
6Z: $\{Z_{\Delta HW}, Z_{HW}\}_I$
6M: $\{M_{\Delta HW}, M_{HW}\}_I$

7. Δ-functionality test bench testing:
7Z: $\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}\}_I$
7Ma: $\{M_{\Delta SW}, M_{\Delta HW}\}_I$
7Mb: $\{M_{\Delta SW}, Z_{SW}, M_{\Delta HW}\}_I$
7Mc: $\{Z_{\Delta SW}, Z_{SW}, M_{\Delta HW}\}_I$
7Md: $\{M_{\Delta SW}, Z_{SW}, Z_{\Delta HW}\}_I$

8. Δ-functionality system testing:
8Z: $\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}, Z_{HW}\}_I$
8Ma: $\{M_{\Delta SW}, M_{\Delta HW}, M_{HW}\}_I$
8Mb: $\{M_{\Delta SW}, Z_{SW}, M_{\Delta HW}, M_{HW}\}_I$
8Mc: $\{Z_{\Delta SW}, Z_{SW}, M_{\Delta HW}, M_{HW}\}_I$
8Md: $\{M_{\Delta SW}, Z_{SW}, Z_{\Delta HW}, Z_{HW}\}_I$

9. System testing:
9Z: $\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}, Z_{HW}\}_I$
9M: $\{Z_{\Delta SW}, Z_{SW}, M_{\Delta HW}, M_{HW}\}_I$

In the current I&T process of ASML, the software qualification tests (category 1) consume quite some machine time, approximately one full day of testing each week. Besides that machine time is limited and expensive, experience shows that also setting up the system for testing may consume a considerable amount of time. Moreover, much time may be lost on solving minor machine problems that are unimportant for the tests, e.g., a malfunctioning sensor that is not involved in the test but prevents the system from initializing. Test time and costs may be reduced by using hardware models instead of hardware realizations for certain parts of the qualification tests. For example, the qualification of the system throughput in principle depends only on the sequence and durations of all hardware actions. When the durations of these hardware actions are modeled as time delays in a model M_{HW} of the hardware system Z_{HW}, and when the software Z_{SW} executes the sequence of actions on the model M_{HW}, the system throughput can be qualified without a hardware realization Z_{HW}. In this way, software qualification tests can be performed in a low cost desktop environment with a hardware model M_{HW}, i.e., using $\{Z_{SW}, M_{HW}\}_I$ instead of $\{Z_{SW}, Z_{HW}\}_I$. Furthermore, models require less test setup time, and they do not suffer from the minor problems that may occur in other components not involved in the tests, since the hardware model only contains the behavior important for the tests and abstracts from these problems.

A model $M_{\Delta SW}$ of the Δ software component can be used as replacement of $Z_{\Delta SW}$ for software regression testing (category 4), i.e., i.e. $\{M_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$ (4Mb) instead of $\{Z_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$ (4Z). By real-time simulation of the model in combination with the other software Z_{SW}, tests can be performed on the original hardware system Z_{HW} to check whether any of the original system functions are negatively affected by the new software component. Similar to model-based software qualification testing in test activity 1Ma, the hardware realization Z_{HW} could also be replaced by its model M_{HW}, i.e. $\{M_{\Delta SW}, Z_{SW}, M_{HW}\}_I$ (4Ma).

Testing the complete Δ-functionality using a test bench (category 7) can be supported by four MBI&T activities. First, the Δ-functionality can be tested by using the integrated models of the Δ components, i.e., $\{M_{\Delta SW}, M_{\Delta HW}\}_I$, in which $M_{\Delta HW}$ is a model of the test bench including $Z_{\Delta HW}$. Since only models are used in this test, model-based analysis techniques such as model checking can be used for exhaustive analysis of all possible behaviors of the system model, as shown in [7]. Second, the model $M_{\Delta SW}$ can be integrated with the other software Z_{SW}, and tested on the test bench model $M_{\Delta HW}$. Third, the realization of the upgraded software system, i.e., $\{Z_{\Delta SW}, Z_{SW}\}_I$, can be tested on the model of the test bench $M_{\Delta HW}$. Finally, in the case that $Z_{\Delta HW}$ is available before the software realization $Z_{\Delta SW}$, the model $M_{\Delta SW}$ can be tested with Z_{SW} on the test bench realization $Z_{\Delta HW}$, as shown in [8].

Fig. 4 shows all I&T activities of the MBI&T process, in a way similar to Fig. 2. The flag symbols and the letters indicate the following milestones: (a) QBL Z_{SW} passes qualification tests; (b) modeling of $M_{\Delta SW}$ and $M_{\Delta HW}$ is started, possibly based on the original system (denoted by dashed upward arrows); (c) $M_{\Delta SW}$ and $M_{\Delta HW}$ are available; (d) $M_{\Delta SW}$ and $M_{\Delta HW}$ pass model tests; (e) development of $Z_{\Delta SW}$ and $Z_{\Delta HW}$ is started, possibly based on the original system and the models (denoted by dashed upward arrows); (f) $Z_{\Delta SW}$ is available; (g) $Z_{\Delta SW}$ passes software tests (automatic model-based component testing, i.e., testing realization against model, is denoted by a double headed arrow) and is integrated in the QBL Z_{SW} (denoted by downward arrow); (h) upgraded QBL $\{Z_{\Delta SW}, Z_{SW}\}_I$ passes qualification tests; (i) $Z_{\Delta HW}$ is available; (j) $Z_{\Delta SW}$ and $Z_{\Delta HW}$ pass test bench tests; (k) $Z_{\Delta HW}$ passes hardware integration tests and both Z_{HW} and M_{HW} are upgraded (denoted by downward arrows); (l) similar to the depicted Δ-functionality, other Δ-functionalities are integrated and tested; (m) complete system with all Δ-functionalities passes system tests and is shipped to the customer.

As an example, the circles indicate the I&T activities of category 7, Δ-functionality test bench testing. Their positions in the development process clearly illustrate how models enable earlier testing on the system level when compared to the current I&T process, in which only the realization test 7Z can be performed late in the process.

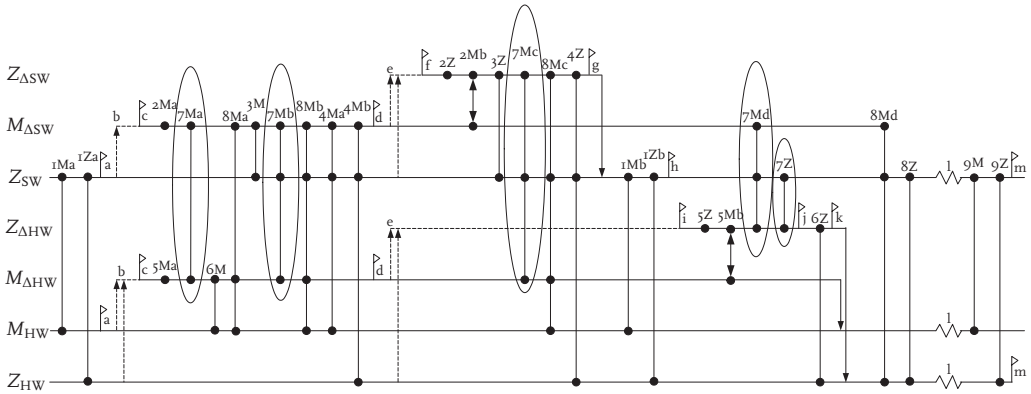


Figure 4: Model-based integration and testing process

Although Fig. 4 shows more I&T activities than Fig. 2, the number of activities does not relate to the total duration of the I&T process, because the possible start times and the durations of the I&T activities are not included. For example, in the case that the hardware upgrade is realized earlier than the software upgrade (i.e., milestone i before f), the MBI&T activities 7Md and 8Mc could be performed before 2Z.

3 Integration and test sequencing

Although all possible (model-based) I&T activities have been defined in the previous section, there is still a problem that needs to be addressed before the MBI&T process can be applied to a real I&T problem. This problem, which also exists in the current I&T process, is called integration and test sequencing, which was investigated in another part of the TANGRAM project [9, 6]. Integration and test sequencing involves making decisions on which components should be integrated when, and which tests should be performed in which order on which components. These decisions result in a sequence of I&T activities that can

be optimized towards criteria such as lead time, total test time, test costs, and remaining risk (i.e., quality) in the system, depending on the importance of the business drivers, i.e., time-to-market, product quality, and costs.

For the MBI&T process, there is not only the problem of deciding on the sequence of the I&T activities, but there is also a choice whether or not to use models for certain I&T activities. This choice involves a trade-off between the potential benefits of applying the MBI&T method (e.g., shorter time-to-market and improved product quality) and the additional costs needed to enable the MBI&T activities (e.g., time needed to model and integrate the components). We use the I&T sequencing method from [9] to determine the (nearly) optimal I&T sequences and to quantify the related costs for various I&T processes. By comparing the costs of I&T processes with and without models, this provides a quantitative decision making process to decide when it is profitable to use models for integration and testing.

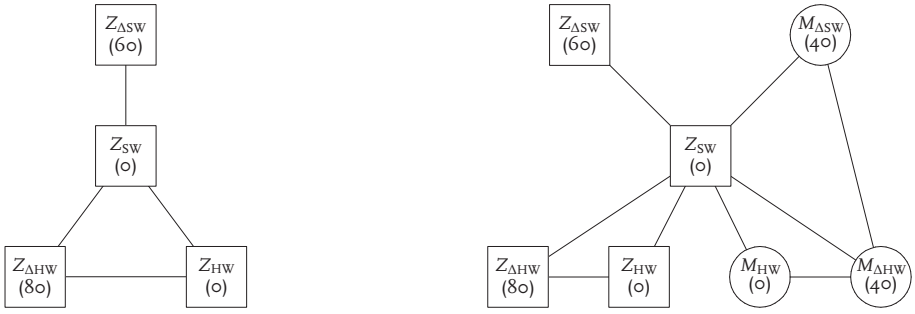
In this section, we discuss this quantitative decision making process by giving an illustrative example that is based on the system upgrade I&T process from the previous section. The system upgrade example is instantiated once with realizations only (as in the current I&T process), and once with the possibility to use models as well (as in the MBI&T process), such that the resulting I&T sequences and related costs can be compared.

The input for the I&T sequencing method is an I&T process model that contains an abstract representation of an I&T problem. An I&T process model defines properties of and relations between the components and interfaces of a system, the potential faults related to the components and interfaces, and the tests that can be performed on certain combinations of integrated components to detect certain faults.

Fig. 5 and Table 2 show the information used for the I&T process models of the system upgrade example. Figs. 5(a) and 5(b) depict the components (boxes for realizations and circles for models) and interfaces (lines) for the current and for the model-based I&T process, respectively. The numbers between the brackets denote the development or delivery times of each component. This example uses fictitious but representative development times for the system upgrade I&T process, in which the original hardware and software are available from the start (development time is zero) and the Δ software component is available after 60 time units, which is 20 time units before the Δ hardware component. In the MBI&T process, the models of the Δ components are available after 40 time units. Note that an interface can also indicate a possible integration of components that is only needed for a certain I&T activity. For example, the interface between Z_{SW} and $Z_{\Delta HW}$ is only needed for test bench testing and does not necessarily exist in the actual system. Also note that the model-based interface layout is symmetrical except for the additional interface between $M_{\Delta SW}$ and $M_{\Delta HW}$. This interface is used for model-based analysis of the Δ-functionality (I&T activity 7Ma), which can be performed early in the I&T process without realizations (see Fig. 4).

Table 2 shows the available tests in the I&T process model of the system upgrade example, including the components that need to be available and integrated for each test, and the test durations. The tests in the table relate to all 29 I&T activities of the nine categories listed in the previous section, including both the current and the model-based I&T activities. In the I&T process model, the choice of using models or realizations for a certain test can be expressed in several ways. For the system upgrade example, we express it in a simple way which is sufficient to explain the quantitative decision making process in this section. In the industrial application described in the next section, we use a more detailed and more realistic way of expressing this choice.

For the system upgrade example, we subdivide all tests into tests that can be performed with realizations only (denoted with a 'Z' in Table 2) and into tests that can possibly be performed with models (denoted with an 'M' in Table 2). This subdivision is done for both the current



(a) Current I&T process (b) MBI&T process

Figure 5: Components and interfaces for the system upgrade example

and model-based I&T process, under the assumption that the aspects covered by an 'M' test can also be tested using realizations, i.e., also in the current, non model-based, I&T process. This assumption is valid in most cases since the models are abstract representations of the realizations and their behavior, implying that the tested behavior of a model will also occur in the corresponding realization. As a result, the 'M' tests contain a choice of using realizations as an alternative to the models, which is denoted by the round brackets in the second column in Table 2. For the current I&T process, only the realization alternatives can be chosen, while for the MBI&T process, both alternatives can be chosen. In this way, we can use a single set of tests to compare the current and model-based I&T process. Taking category 1 as an example, we see that I&T activity 1Za requires Z_{SW} and Z_{HW} to be available and integrated. I&T activity 1Ma always requires Z_{SW}, but gives a choice to use either the hardware system model M_{HW} or the realization Z_{HW}. For the current I&T process, only the equivalent realization test with Z_{HW} can be used for 1Ma, while in the MBI&T process, the model M_{HW} can also be used for 1Ma if this is profitable.

Table 2: Available tests for the system upgrade example

Test	Required components	Time
1Za	$\{Z_{SW}, Z_{HW}\}_I$	6
1Ma	$\{Z_{SW}, (M_{HW}/Z_{HW})\}_I$	2
1Zb	$\{Z_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$	6
1Mb	$\{Z_{\Delta SW}, Z_{SW}, (M_{HW}/Z_{HW})\}_I$	2
2Z	$Z_{\Delta SW}$	1
2Ma	$(M_{\Delta SW}/Z_{\Delta SW})$	1
2Mb	$Z_{\Delta SW}$ vs. $M_{\Delta SW}$	1
3Z	$\{Z_{\Delta SW}, Z_{SW}\}_I$	2
3M	$\{(M_{\Delta SW}/Z_{\Delta SW}), Z_{SW}\}_I$	1
4Z	$\{Z_{\Delta SW}, Z_{SW}, Z_{HW}\}_I$	3
4Ma	$\{(M_{\Delta SW}/Z_{\Delta SW}), Z_{SW}, (M_{HW}/Z_{HW})\}_I$	1
4Mb	$\{(M_{\Delta SW}/Z_{\Delta SW}), Z_{SW}, Z_{HW}\}_I$	2
5Z	$Z_{\Delta HW}$	2
5Ma	$(M_{\Delta HW}/Z_{\Delta HW})$	1
5Mb	$Z_{\Delta HW}$ vs. $M_{\Delta HW}$	1
6Z	$\{Z_{\Delta HW}, Z_{HW}\}_I$	3
6M	$\{(M_{\Delta HW}/Z_{\Delta HW}), (M_{HW}/Z_{HW})\}_I$	1
7Z	$\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}\}_I$	4
7Ma	$\{(M_{\Delta SW}/Z_{\Delta SW}), (M_{\Delta HW}/Z_{\Delta HW})\}_I$	2
7Mb	$\{(M_{\Delta SW}/Z_{\Delta SW}), Z_{SW}, (M_{\Delta HW}/Z_{\Delta HW})\}_I$	1
7Mc	$\{Z_{\Delta SW}, Z_{SW}, (M_{\Delta HW}/Z_{\Delta HW})\}_I$	2
7Md	$\{(M_{\Delta SW}/Z_{\Delta SW}), Z_{SW}, Z_{\Delta HW}\}_I$	1
8Z	$\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}, Z_{HW}\}_I$	4
8Ma	$\{(M_{\Delta SW}/Z_{\Delta SW}), (M_{\Delta HW}/Z_{\Delta HW}), (M_{HW}/Z_{HW})\}_I$	2
8Mb	$\{(M_{\Delta SW}/Z_{\Delta SW}), Z_{SW}, (M_{\Delta HW}/Z_{\Delta HW}), (M_{HW}/Z_{HW})\}_I$	2
8Mc	$\{Z_{\Delta SW}, Z_{SW}, (M_{\Delta HW}/Z_{\Delta HW}), (M_{HW}/Z_{HW})\}_I$	2
8Md	$\{(M_{\Delta SW}/Z_{\Delta SW}), Z_{SW}, Z_{\Delta HW}, Z_{HW}\}_I$	2
9Z	$\{Z_{\Delta SW}, Z_{SW}, Z_{\Delta HW}, Z_{HW}\}_I$	60
9M	$\{Z_{\Delta SW}, Z_{SW}, (M_{\Delta HW}/Z_{\Delta HW}), (M_{HW}/Z_{HW})\}_I$	20

The test durations in the third column of Table 2 are fictitious but give a representative distribution of the test time over all I&T activities of the system upgrade I&T process. For simplicity, this example I&T process model uses constant test durations, which is sufficient to explain the quantitative decision making process and to compare the current and model-based I&T process for the system upgrade example. In reality, the duration of a test depends on the remaining risk in the system, i.e., on how much risk is already reduced by previous tests, on different risk reduction rates of certain tests, e.g., tests with realizations or with models, and on the stopping criterion of a certain test, e.g., reduce all risk or stop at a certain remaining risk threshold. The industrial application in Section 4 uses a more detailed I&T process model that incorporates potential faults and their risks to determine test durations.

Although the I&T process model for the system upgrade does not express differences in durations or costs for testing with realizations or with models, the differences can be determined and analyzed afterwards, based on the generated I&T sequences and on different test costs per time unit for testing with realizations or with models. Taking software qualification testing (category 1) as an example, Table 2 shows that the main part of this category can be executed with realizations only (I&T activity 1Za, 6 time units), while some aspects could possibly be tested with models (I&T activity 1Ma, 2 time units). For the current I&T process, the aspects tested in 1Ma are covered by equivalent realization tests with the same test duration (also 2 time units). Since testing with realizations is usually more expensive than testing with models, we see that the test costs of 8 time units of expensive realization testing in the current I&T process are higher than 6 time units of expensive realization testing and 2 time units of low cost model testing in the MBI&T process. In this way, we can determine and compare the test costs for specific I&T sequences without using different test times in the integration model, which is sufficient for this example.

Based on an I&T process model as described above, the I&T sequencing algorithm from [9] determines all feasible I&T sequences. For the example in this section, an I&T sequence is feasible when all components are integrated via the defined interfaces, when all defined tests are performed, and when the tests are not performed before the required components are integrated. Determining these sequences is based on an assembly-by-disassembly technique [10], which starts with a complete system and iteratively subdivides the system into smaller parts by removing interfaces until only components are left. Reversing the result gives all possible integration sequences from single components to a complete system. Since the number of feasible sequences can be very large, heuristics can be applied to remove sequences that are expected to have a low performance according to the optimization criteria used. For more details on the I&T sequencing algorithm, we refer to [9].

As explained in [4], time-to-market is the most important business driver for ASML, therefore we use the duration of the complete I&T process, the lead time, as the optimization criterion for I&T sequencing. Note that the lead time is different from the total time used for integration and testing, which is the sum of the durations of all separate I&T activities. By performing multiple I&T activities in parallel, the total time used for integration and testing remains equal but the lead time is reduced. To reduce the number of sequences and the corresponding computation time, we applied a heuristic that prefers I&T sequences with as much parallelism as possible.

Fig. 6 shows the determined I&T sequences for the current (top) and model-based (bottom) I&T process models, in the form of a Microsoft Project Gantt Chart. The figure shows all activities related to component development (dashed bars), component modeling (white bars at the bottom), integration (diamonds), and testing (black bars) over time, and the precedences between the activities (arrows). On the topmost line of the I&T sequences, the long white bar with triangular line ends indicates the lead time.

Several conclusions can be drawn from these sequences. First, the lead time of the MBI&T

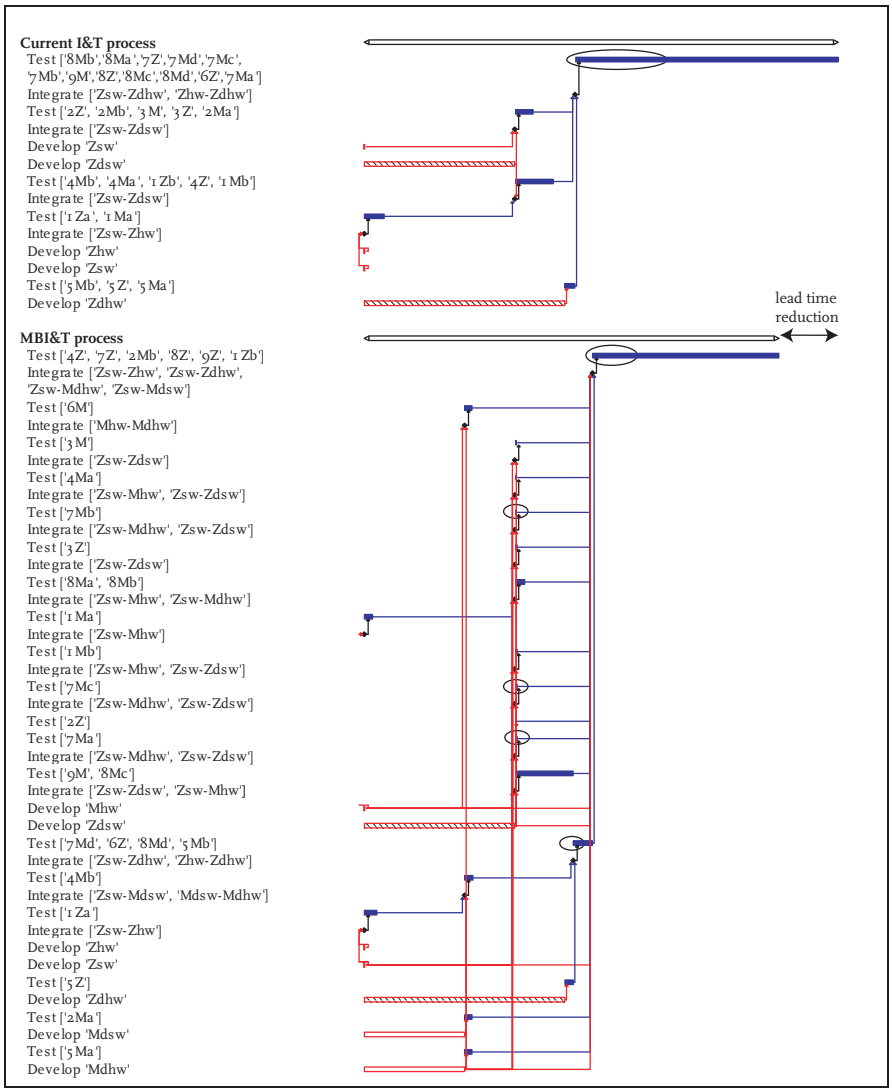


Figure 6: Current (top) and model-based (bottom) I&T sequences

sequence is shorter, 167 time units against 190 time units for the current I&T sequence, a reduction of 12%. Besides lead time, also the duration of the final system test phase (the long black bars at the right-hand side of Fig. 6) is important for ASML, since this phase is on the critical path and has a major influence on the time-to-market. The final system test phase is 78 time units for the MBI&T sequence, 26% less than the 106 time units for the current I&T sequence.

The profitability of using models in the I&T process is determined by quantifying the costs of enabling the above mentioned benefits of shorter lead time and shorter final system test phase. As previously mentioned, costs were not explicitly expressed in this system upgrade example but can be compared afterwards, using different costs per time unit for testing with realizations or with models. At ASML, the costs per time unit for realization testing are orders of magnitude higher than for testing software or models in a desktop environment. The use of models influences the costs of the I&T process in both a positive and in a negative sense. On the one hand, the costs of the I&T process increase due to the effort invested in

modeling the components, e.g., 80 time units in a relatively low cost desktop environment for the system upgrade example. On the other hand, the costs of the I&T process decrease due to the reduction of time spent on realization testing and due to the possibility to diagnose and fix problems earlier. In Fig. 6, the current I&T process uses 138 time units of realization testing, while the MBI&T process uses 91 time units of realization testing (34% less) and 47 time units of testing with models, with relatively lower costs. Besides a reduction in these direct costs of testing, the use of models in the I&T process also reduces indirect costs such as diagnosis and fix costs when problems are detected. Compared to the current I&T process, the I&T activities in the MBI&T process are performed earlier and more in parallel, see for example the position of the I&T activities of category 7 in both sequences, indicated by the circles in Fig. 6. As a result, design and integration problems can be detected and prevented at an earlier stage of development where the costs for fixing them are lower. For example, in the wafer scanner case study described in [7, 8], the MBI&T activities prevented problems that, if they would remained undetected, would result in machine damage in the real I&T phases, with high costs for fixing the problems and for the accompanying downtime. Although these benefits of early testing cannot directly be expressed in the I&T process model of the system upgrade example, they can be expressed in terms of risk, as shown in the industrial case study in the next section. Summarizing, besides the benefits of a shorter lead time and a shorter final system test phase in the I&T process, using models is also beneficial for the costs of testing, diagnosis, and fixing.

This quantification of benefits and costs can be used as a basis for deciding whether it is profitable to use models in the I&T process. In the system upgrade example, the estimated benefits of using models, e.g., a lead time reduction of 12%, a reduction of realization test time of 34%, and reduced costs for testing, diagnosis, and fixing, probably justify the onetime investments needed for model development, which are 80 time units of modeling in a (low cost) desktop environment.

As an overall result, this example showed the ingredients of the following quantitative decision making process, which can be used to determine when it is profitable to use models in the I&T process:

1. Create an I&T process model with component realizations only.

2. Create a second I&T process model by extending the first I&T process model with component models and by giving a choice of using either a model or a realization for certain I&T activities.

3. Determine the I&T sequences for both I&T process models using the I&T sequencing method.

4. Compare the resulting I&T sequences based on the quantified benefits and costs, e.g., on lead time and test costs.

5. Decide whether the benefits of using models outweigh the additional costs to achieve the benefits.

In the next section, this quantitative decision making process is used in a case study to determine for which parts of a new version of an ASML wafer scanner it is profitable to create a model for early integration and testing.

4 Practice: which components of a new ASML wafer scanner should be modeled?

In [7, 8], different activities of the MBI&T method were successfully applied to a current version of an ASML wafer scanner, in order to provide a proof of concept showing that models can effectively be used for early integration and testing. Although the MBI&T method proved to be both applicable and profitable in industrial practice, the estimated profitability was not a main criterion for deciding where to apply the method for this proof of concept. Instead, these decisions were mainly based on the estimated applicability, e.g., problem size and characteristics, and on the personal involvement of ASML engineers in both the TANGRAM project and in the development of the current version of the wafer scanner.

After seeing the applicability and potential profitability of the MBI&T method, the developers of the current version of the wafer scanner would like to know, for future versions of the wafer scanner, which components are the most interesting and profitable candidates for model-based integration and testing. Assuming that a new version of a wafer scanner should be developed, the next subsections describe how the five steps of the quantitative decision making process were executed in order to decide which components should be modeled.

4.1 Step 1: Create an I&T process model with component realizations only

In a new version of a wafer scanner, some components of the current version may be reused or adapted, while other components may be new and need to be developed from scratch. These reused, adapted, and new components should then be integrated according to a predetermined system architecture. Fig. 7 shows an example of a part of such a system architecture, in which the boxes represent the components and the lines represent the interfaces. The figure shows five subsystems, A through E, for which the grey values of the boxes denote different characteristics of the components. The light grey component realizations (subsystems A and B) are reused from the current version of the wafer scanner, which means that these components will be available earlier and have less risk since they are more mature. In the I&T process model, the development time of a light grey component is 40 time units. The dark grey components (subsystems C, D and E) are newly developed component realizations, which means that they will be available later and have more risk since they have never been used and tested before. In the I&T process model, the development time of a dark grey component is 160 time units. Finally, the white 'switch' components, denoted by S, are not part of the real system but they are 'dummy' components to express the risk reduction of testing with models in a more realistic way, which is explained later in this section. Since they are 'dummy' components, they have zero development time and add no risk to the system.

The interfaces in Fig. 7 connect the components using different interaction types as explained in [8]. Most interfaces are used to connect components of the same subsystem. Connections between components of different subsystems are mostly established via the interfaces shown at the top of the figure that represent software interfaces, although three other connections exist as well: between Z_{C_3} and Z_{D_2}, between Z_{B_7} and S'_E, and between Z_{D_4} and S'_E.

The different characteristics of the components and the interfaces influence the amount of risk that they introduce in the system, as well as the amount of time that is needed to reduce this risk by testing and by fixing the detected problems. In contrast to the system upgrade example, in which Table 2 expressed the simple relations between tests, components, and fixed test durations, the I&T process model in this case study incorporates *risk* to express these relations and to determine test durations. In the I&T process model, risk is expressed by multiplying the probability and the impact of *fault states* [18]. Fault states denote possi-

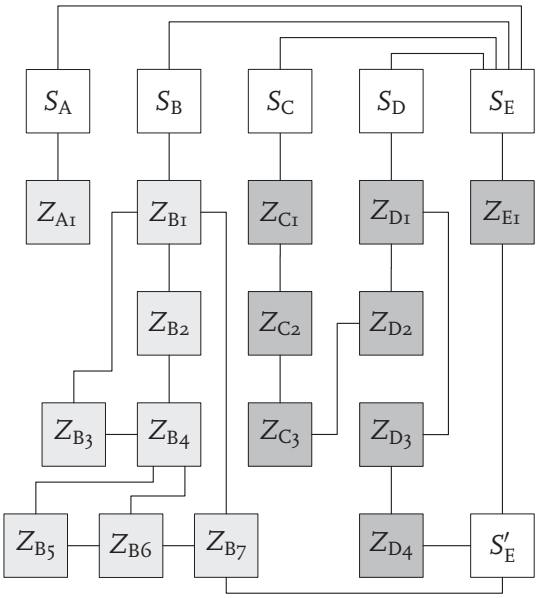


Figure 7: Components and interfaces of a new version of the wafer scanner

ble problems that may be present in the system, e.g., a broken component, an error in the interface, missing or erroneous functionality, or insufficient performance. A fault state has a certain probability that it is present in the system. When a fault state is present in the system and when it manifests itself in the system behavior, it has a certain impact on the system development process, e.g., restarting the system, replacing a broken component, or even worse, revising the component or system design to prevent the problem. The impact of a fault state usually also depends on the point in time at which its presence is detected: later detection usually means more impact. In the I&T process model of a new version of the wafer scanner, however, differences in fault state impacts are not considered, i.e., impact = 1, which means that the risk only depends on the fault state probability. Table 3 defines the fault states and shows the risk contribution of each component and interface to these fault states. To reduce the size and the complexity of the table, only the components and the interfaces related to subsystems A, B, and E are shown. The components and internal interfaces of subsystem B are grouped and denoted by Z_B* and B-internal*, respectively. On the left-hand side, the table shows five fault states denoted by an f with a subscript: three (internal) fault states for the individual subsystems, one fault state for the interaction between subsystems A and E, and one fault state for the interaction between subsystems B and E. The numbers in the table denote the relative risk contribution of the components and interfaces to each fault state, where different numbers imply different levels of risk contribution, e.g., based on the maturity of a component or on the partitioning of interfaces, which is explained later. For example, Z_{A1} introduces less risk than Z_{E1}, since subsystem A is reused from the current version of the wafer scanner, while subsystem E is newly developed, as indicated by the grey values in Fig. 7. Note that for Z_B* and B-internal*, each of the grouped components and interfaces has an individual risk contribution of 0.3 to fault state f_B.

In a way similar to Table 3 that expresses how risk is introduced via components and interfaces, Table 4 expresses how risk can be reduced by performing tests. The table shows the fault states f on the left-hand side and possible tests, denoted with t, at the top. The individual subsystems can be tested by t_A, t_B, and t_E. To test the interaction between subsystems, t_{A+E} and t_{B+E} can be used. Finally, the complete system can be tested by t_{bench} using a test bench with only a part of the system, or by t_{system} using the complete system. The numbers

Table 3: Contribution of components and interfaces to fault states

	components			interfaces								
	Z_{A1}	Z_B*	Z_{E1}	$S_A - Z_{A1}$	$S_A - S_E$	B-internal*	$S_B - Z_{B1}$	$S_B - S_E$	$Z_{B7} - S'_E$	$S_E - Z_{E1}$	$Z_{E1} - S'_E$
f_A	0.3										
f_B		0.3				0.3					
f_E			0.6								
$f_{A\text{-}E}$				0.2	0.6					0.2	
$f_{B\text{-}E}$							0.2	0.6	0.6	0.2	0.2

in the table denote the coverage of each test, i.e., the probability that a certain test detects a certain fault state. For example, the internal subsystem fault states are completely covered by the subsystem tests and partially by the interaction tests, which in turn completely cover the interaction fault states. Besides the relation between tests and fault states, the table also shows the duration of each test at the bottom. Different tests may have different durations, e.g., an interaction test takes more time than an internal subsystem test, but less time than a system level test. Note that information about which components need to be realized and integrated to perform a certain test, similar to the second column of Table 2, is omitted here, because it is straightforward in this case study: all tests except t_{bench} require all components of the involved subsystems, i.e., either one, two, or all subsystems. The test bench used in t_{bench} consists of all components except for the components on the lowest level of Fig. 7.

Table 4: Tests and their coverage on the fault states

	t_A	t_B	t_E	t_{A+E}	t_{B+E}	t_{bench}	t_{system}
f_A	1			0.5		0.5	1
f_B		1			0.5	0.5	1
f_E			1	0.5	0.5	0.5	1
$f_{A\text{-}E}$				1		0.5	1
$f_{B\text{-}E}$					1	0.5	1
time	8	8	8	16	16	16	24

In this case study, the test durations in the I&T sequences are determined in another way than in the system upgrade example of Section 3, which used constant test durations as defined in Table 2. In this case study, the test durations are not constant, but they depend on the remaining risk at the moment that a test is executed and thus on the preceding I&T sequence. Here, we only give an informal explanation of how the remaining risk and the expected test duration at a certain point in the I&T sequence are calculated, we refer to [9] for more details. When two system parts, i.e., one component or multiple integrated components, are integrated by connecting an interface between these system parts, the probability of a related fault state, and thus the risk, increases. This increased risk is calculated using the fault state risks of the separate system parts and the risk contribution of the connected interface as defined in Table 3. For example, when two system parts with risks of 0.6 and 0.4 for a certain fault state are integrated via an interface that contributes 0.2 risk to that fault state, the increased fault state risk after integration is $1-(1-0.6)*(1-0.4)*(1-0.2) = 0.808$. When a test is performed on a system part, the risk of a fault state covered by the test decreases. This decreased risk is calculated using the fault state risk before the test and the test coverage as defined in Table 4. For example, when a system part has a risk of 0.8 for a certain fault state, and when it passes a test that has 0.5 coverage for that fault state, the risk after testing is $0.8 * (1 - 0.5) = 0.4$. Using risk calculations like these, the fault state risks continuously change with each integration or test activity in the I&T sequence. This influences the expected duration of a particular test activity in the following way. A test activity may involve the execution of multiple tests, for which the optimal test sequence is determined using a test sequencing algorithm [18]. This algorithm is based on a sequential diagnosis method [11] and uses AND/OR graphs to

represent test trees, showing which tests should be executed depending on whether previous tests passed or failed. The expected duration of such a test tree is calculated by multiplying the test durations defined at the bottom of Table 4 with the probabilities that each test in the tree will be executed, which in turn depend on the 'pass' and 'fail' probabilities of previous tests in the tree. The 'pass' and 'fail' probabilities of a test depend on the test coverage as well as on the fault state risks before the test, which are calculated as described above.

4.2 Step 2: Extend the I&T process model of step 1 with component models

As shown in Fig. 7, subsystem E contains a new component E1 that needs to be developed from scratch. Besides that this means relatively long development times as described in the previous subsection, subsystem E also interacts with many other subsystems, which increases the risk of problems. Considering this high risk, it is expected that a model of component E1 would be a good candidate to improve the I&T process of a new version of the wafer scanner. In contrast to component E1, component A1 of subsystem A is reused from the current version of the wafer scanner and it has only one interface, i.e., it is available earlier and has a lower risk. This means that including a model of component A1 probably has a low profitability. By creating I&T process models that include these component models and by analyzing the resulting I&T sequences, we investigate whether the quantitative decision making process can be used to confirm these expectations on the profitability of using a model of components A1 and E1.

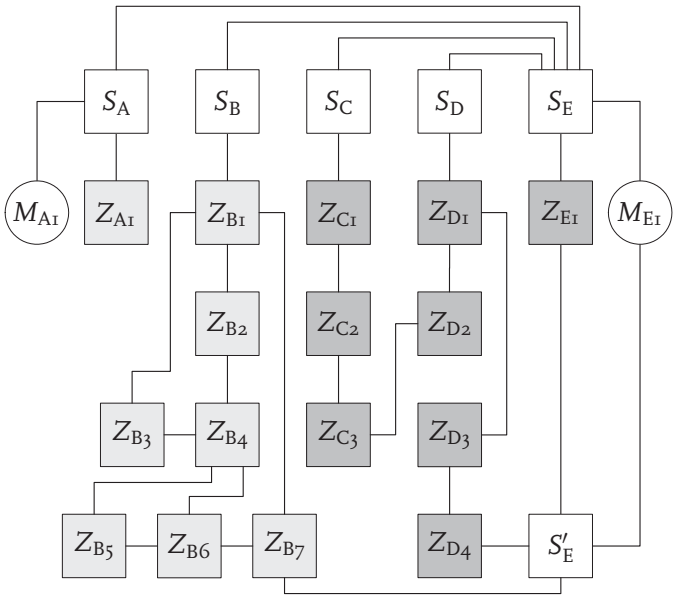


Figure 8: Including models of components A1 and E1

Including component models in the I&T process model results in several changes with respect to the information shown in Fig. 7, Table 3 and Table 4. As shown in Fig. 8, additional components (circles) and interfaces are introduced to represent the models M_{A1} and M_{E1}. In this case study, the development time for the models is defined at 16 time units, which is shorter than the 40 and 160 time units of development time for the corresponding realizations Z_{A1} and Z_{E1}. The models have similar interfaces as the corresponding realizations,

connecting them to the same 'switch' components, one for M_{AI} and two for M_{EI}. As previously mentioned, these 'switch' components are not part of the real system but they are 'dummy' components to express the risk reduction of testing with models in a more realistic way. Without the 'switch' components in the I&T process model, a model of a component would have interfaces to all other components to which the corresponding realization is connected as well, e.g., M_{EI} in Fig. 8 would have interfaces to Z_{AI} through Z_{DI}, as well as to Z_{B7} and Z_{D4}. Testing the combination of M_{EI} and some of these other components would then require that the model interfaces, e.g., between M_{EI} and Z_{AI}, are integrated, while the realization interfaces, e.g., between Z_{EI} and Z_{AI}, are not integrated. This means that the risk of the realization interface is not tested with the model, and will completely be introduced when the component realization is integrated via the interface. This does not correspond to reality, since an important aspect of using models for integration and testing is that the interaction between components, i.e., the risk in the interfaces, is at least partly tested at an early stage. This is the reason why the 'switch' components were introduced and the interfaces were partitioned. For example, a test that uses M_{EI} and Z_{AI} already includes a large part of the interface risk between Z_{EI} and Z_{AI}, namely the interface between S_E and S_A (with a relatively large risk contribution to fault state $f_{A\text{-}E}$, see Table 3) and the interface between S_A and Z_{AI}. Together with the interface between S_E and M_{EI}, a large part of the final realization interface and its corresponding risk is reduced when testing with models. Later, when M_{EI} is replaced by Z_{EI}, only the risk related to the interface between the Z_{EI} and S_E is added to the system risk and needs to be reduced by testing.

Table 3 changes in a sense that new columns for the models and the related interfaces are added and their relative risk contributions to each fault state are defined. In principle, models should be abstract representations of the realizations and they should not introduce additional risk on top of the risk introduced by the realizations themselves. In practice, however, there is a possibility that models differ from the actual realization, which introduces some additional risk in the system. In the changed table (with new columns for the models and the related interfaces as described above), this additional risk is expressed by giving the models and related interfaces a small relative risk contribution to the same fault states as their realization counterparts. For example, M_{EI} gets a relative risk contribution of 0.1 to fault state f_E, and the interface between S_E and M_{EI} gets a relative contribution of 0.1 to both $f_{A\text{-}E}$ and $f_{B\text{-}E}$. This additional risk introduced by the models should also be reduced by testing, which is part of the additional costs of using models in the I&T process. Table 4 remains unchanged when models are added. The information about the required components for each test changes in a sense that whenever a test requires Z_{AI} or Z_{EI}, there is also a choice of using the corresponding models M_{AI} and M_{EI}, similar to the choices denoted by the round brackets in Table 2. The I&T sequencing algorithm as used in the next step of the quantitative decision making process decides which of the choice alternatives is the most profitable with respect to the used optimization criteria.

4.3 Step 3: Determine the I&T sequences for all I&T process models

Applying the changes described in the previous subsection results in three different I&T process models: one without component models (A), one with model M_{EI} (B), and one with model M_{AI} (C). For each I&T process model, the (nearly) optimal I&T sequence is determined using the I&T sequencing algorithm from [9], using the lead time as optimization criterion, a 'reduce all risk' stopping criterion for each test activity, and a heuristic that reduces the number of I&T sequences by preferring those that have as much parallelism as possible.

For I&T sequencing, the heuristics should be used with care since the results showed to be quite sensitive to the heuristic settings regarding the number of considered alternatives, especially in combination with the fact that the I&T sequencing algorithm uses estimations to choose between alternatives. By increasing the number of considered alternatives for a

heuristic (with the expectation that better sequences may be found), certain alternatives may be 'overruled' by alternatives that have a higher estimated profitability at the time of making the choice (e.g., shorter lead time by skipping a test), but in fact these alternatives result in a lower profitability when the complete sequence is determined (e.g., since a skipped test leaves more risk that needs be reduced later). To overcome this problem, the I&T sequencing algorithm was executed multiple times with different heuristic settings regarding the number of alternatives taken into consideration. The best results for each I&T process model are shown in Table 5, including the lead time (total duration of the I&T sequence), the test time (the amount of time spent on all I&T activities), and the total time (the amount of time spent on all activities, i.e., including development) as well as the relative differences when compared to I&T process model A.

Table 5: Results of the three I&T process models

I&T process model	Lead time	Difference	Test time	Difference	Total time	Difference
A. without models	438	-	426	-	1922	-
B. with M_{E1}	396	-9.6%	508	+19.2%	2020	+5.1%
C. with M_{A1}	439	+0.2%	438	+2.8%	1950	+1.5%

Fig. 9 shows the resulting I&T sequences for I&T process models A and B, in the form of a Microsoft Project Gantt Chart. The figure shows all activities related to development and integration (dashed bars), modeling (white bar at third line from below) and testing (black bars) over time, and the precedences between the activities (arrows). On the topmost line of the I&T sequences, the long white bar with triangular line ends indicates the lead time of the sequence. For simplicity, the development and integration activities of some subsystems and the 'switch' components are grouped (denoted with < and > brackets). Note that the main 'branches' of the two I&T sequences are different. In I&T process model A, one main branch contains subsystems B and D, while the other main branch contains the 'switch' components and the subsystems A, C, and E. In I&T process model B, subsystems C and E are 'moved' from the second main branch to the first main branch with subsystems B and D, while subsystem A and the 'switch' components remain in the second main branch, which now also includes M_{E1} for early testing of the interface between subsystems A and E.

4.4 Step 4: Compare the resulting I&T sequences on quantified benefits and costs

Several conclusions can be drawn from the results shown in Table 5 and Fig. 9. First, these quantitative results support the initial expectations that creating M_{E1} would be profitable, i.e., the lead time is reduced by 9.6%, and that creating M_{A1} would be less profitable, i.e., the lead time even increases with 0.2% due to the additional risk introduced by the model. As expected, M_{E1} enables early testing of the interaction between subsystem E and the other subsystems. For example, by looking at the numbered test trees in Fig. 9, the interface between subsystems A and E can be tested after 60 time units when M_{E1} is used (test tree B-2), while this is only possible after 192 time units when no models are used (test tree A-4). Second, the test time results in Table 5 show that more time is available for testing when a model is used, which increases the system overview and the product quality. This is not only caused by the possibility of earlier testing with the model, but also by the possibility of more parallel testing, as shown by the black testing bars in Fig. 9. This increased test time is also responsible for the major part of the increased total time as shown in the last column of Table 5. The other and much smaller part of the increased total time is caused by model development, 16 time units in this case study.

Another view on the effects of using models in the I&T process is shown in Figs. 10(a) and 10(b), which contain the so-called risk profiles showing how risk evolves over time for

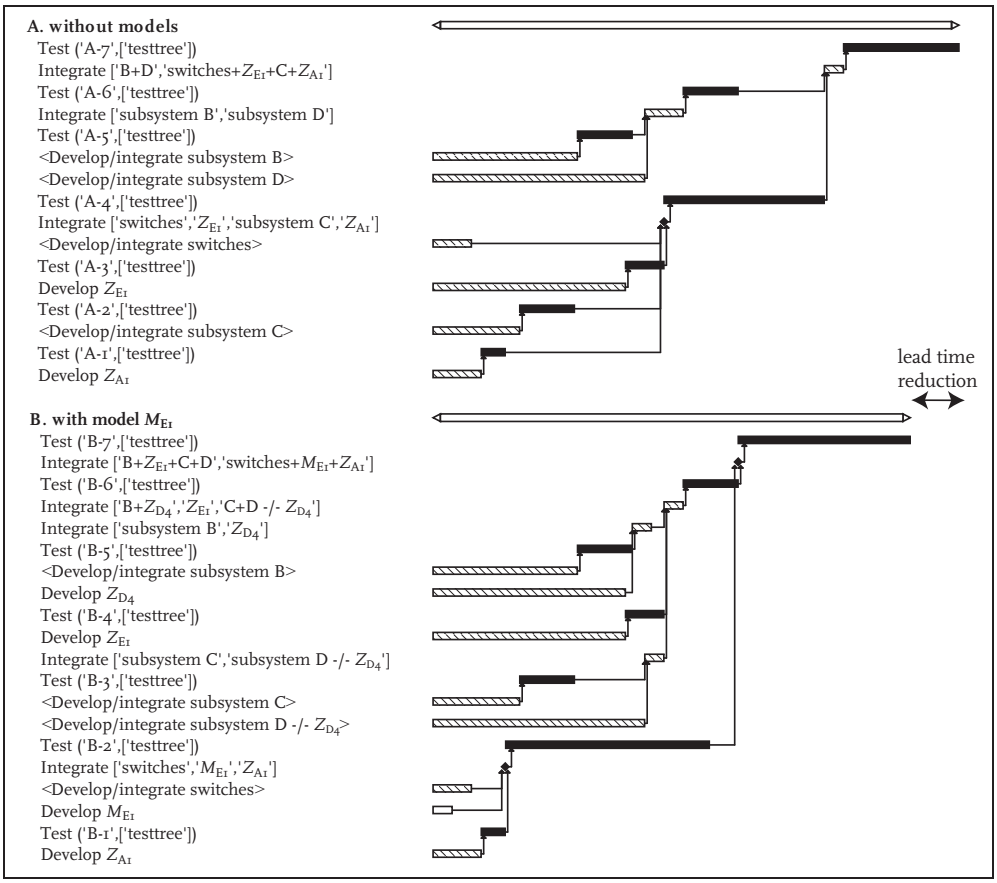
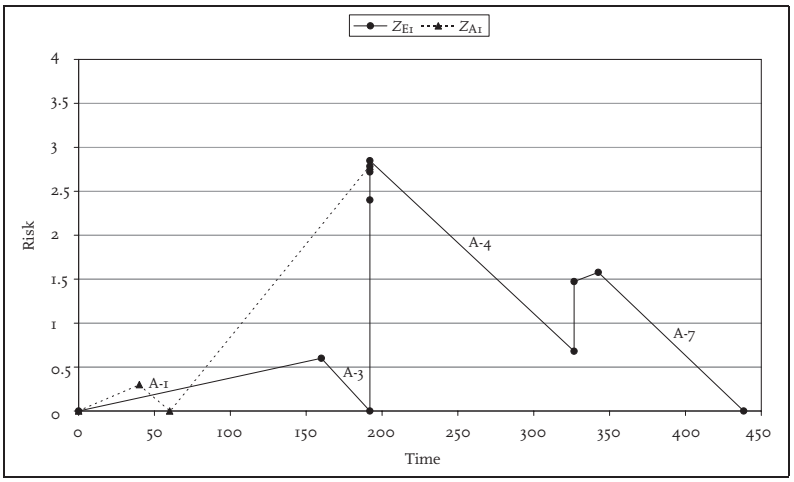


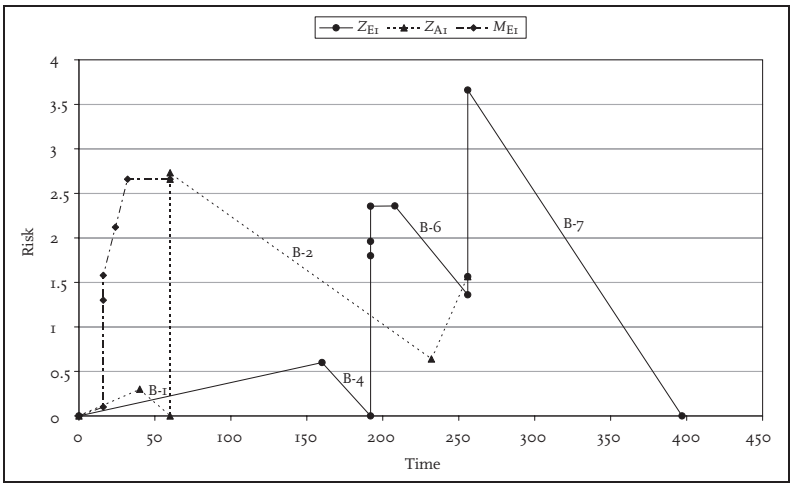
Figure 9: I&T sequences for I&T process model A (top) and B (bottom)

I&T process models A and B, respectively. The risk increases when components are developed and integrated, because potential problems related to a fault state may reveal themselves. The risk decreases when a system part is tested and the detected problems are fixed (the numbers at the slopes correspond to the test trees in Fig. 9). For simplicity, the figures only show the risk profiles for Z_{EI}, Z_{AI}, and M_{EI}, because using M_{EI} has the most influence on these risk profiles. Furthermore, linear abstractions of the risk profiles are used since the I&T sequencing algorithm only calculates the risk at the start and at the end of each integration or test activity (the points in the figure); especially the decrease of risk during testing will have a more exponential shape in reality. Also note that when system parts are integrated in the I&T sequence, the corresponding risk profiles are combined and continued as one risk profile. This also explains the higher risk peak at the end of the I&T sequence in Fig. 10(b), where all components and thus all risk profiles are integrated just before the last test activity (test tree B-7 in Fig. 9). In Fig. 10(a), however, the main risk of Z_{EI} and Z_{AI} is tested halfway the I&T sequence (test tree A-4 in Fig. 9). At this point, subsystems B and D of the other main branch are not integrated yet, hence also their risk profiles are not yet combined with the Z_{EI} risk profile in Fig. 10(a).

The results of the case study are preliminary and need further investigation before real decisions regarding the use of models in the I&T process (step 5) can be taken. For example, the relative probabilities and impacts of fault states, as well as the coverage and durations of the tests should be validated, the use of heuristics should be investigated in more detail, and more risk data should be obtained, e.g., using I&T process simulation techniques [19]. Neverthe-



(a) I&T process model A (without models)



(b) I&T process model B (with M_{E1})

Figure 10: Risk profiles

less, these preliminary results show that the shapes of the risk profiles in Fig. 10 correspond to the initial intention of early integration and testing, namely reducing the disadvantageous influence of the I&T phases on time-to-market and product quality. By using a model M_{E1}, the system risk is revealed at an earlier stage, after which it is immediately reduced by testing (in test tree B-2), resulting in a lead time (and thus time-to-market) reduction for the I&T process and an earlier improvement of the product quality.

5 Concluding remarks

This paper started with a description of the current I&T process, using a system upgrade example that is common in industry. Nine different categories of I&T activities were identified that cover different system aspects. Since tests can only be performed with realizations, the test costs are relatively high and the tests can only be performed late in the process, where

fixing the detected problems is relatively expensive. Subsequently, it was shown how the model-based analysis and testing techniques of the MBI&T method can be applied in each category of I&T activities to enable earlier and more parallel testing with lower costs.

By using the I&T sequencing method from [9], we showed how (nearly) optimal sequences of I&T activities can be determined and how the costs of using models in the I&T process can be quantified. This quantification of costs supports the decision making process of when the use of models is profitable. The results of a basic system upgrade example showed that the lead time and costs of the current I&T process can be reduced by performing tests earlier with models.

Finally, the proposed quantitative decision making process was applied to the I&T process of a new version of an ASML wafer scanner, showing that it is feasible in current industrial practice to quantify the costs of using the MBI&T method and to decide where and when the method should be applied. Three different I&T process models were created that described different scenarios regarding the use of component models. The resulting I&T sequences showed that the profitability of using models can be quantified in terms of reduced lead time, increased and more parallel test time, and risk profiles that show the early revealing of risk which can subsequently be reduced early by testing. The quantitative decision making process supported the initial expectations on the profitability of using two particular models in the I&T process. Using a model of a new component with many interfaces proved to be profitable (9.4% lead time reduction), while using a model of a reused and more mature component even increased the lead time by 0.2%, due to the additional model risk that has to be reduced by testing as well.

Using this quantitative decision making process in practice requires estimations of the development or delivery times of realizations and models. Furthermore, the knowledge about possible fault states of the system with their (relative) probability and impact, and about the available tests with their (relative) coverage and duration needs to be expressed explicitly in the I&T process model, which thus acts as a knowledge container. As described in [9], ASML test engineers that currently use the I&T sequencing method for periodic software qualification testing (I&T activity 1 of Table 1) are able to make these estimations and to maintain the corresponding I&T process model. However, I&T sequencing should not be considered as a 'push the button' technique, since the results may be quite sensitive to the heuristics and their configuration.

In this paper, we focused on quantifying the costs of reducing the time-to-market using the MBI&T method. As shown in previous work [7, 8], reducing time-to-market is not the only advantage of using the MBI&T method. Creating models allows for increasing the system overview and the product quality, and it allows the analysis of all possible behaviors using model checking. Furthermore, models enable a reduction of costs and lead time for diagnosing and fixing problems, and they enable easier and less expensive testing of exceptional behavior, e.g., by simulating a broken component without the risk of real system damage. Although these advantages were not considered in the case study of the quantitative decision making process, they can be expressed in the I&T process model in the following way. As previously mentioned, the effects on product quality can be expressed in terms of risk, e.g., by optimizing the I&T sequences towards minimal risk at a fixed system shipment date. When model checking is considered as a form of testing the system model with very high coverage (all possible behaviors), this can be defined in the I&T process model by a high coverage test that can be performed with models only. Although not incorporated in the case study, the costs for diagnosing and fixing a fault state can also be expressed in the I&T process model. By letting these costs increase over time, the effects of lower diagnosis and fix costs when models are used for early testing can be taken into account. Furthermore, the tests in the I&T process model can be defined such that they can be performed using only models or only realizations, possibly with different test costs. This allows the modeling of, for exam-

ple, tests related to machine damage control that can be performed at low costs with models (machine damage situations can be simulated) but not at all or only at high costs with realizations (with the risk that real machine damage occurs). Finally, 'what if' scenarios can be used to investigate, for example, the effects of developing more detailed models, implying higher model development times, but also a higher coverage of the MBI&T activities and less I&T activities that can be performed with realizations only. In a similar way, motivated choices between longer but low cost model testing and shorter but more expensive realization testing can be made.

These examples of possible extensions show that I&T sequencing is a suitable technique to get insight in and to analyze possible scenarios for industrial I&T processes at an early stage. It is also suitable for making more objective decisions on which I&T activities should be performed in which order, and whether or not models should be used for these I&T activities. The I&T process models used in the quantitative decision making process are rather general, in the sense that only the names of the components, interfaces, fault states, and tests are used, as well as numbers to express their relations and the associated risk, time, and costs. Since no application specific content is required, an I&T process model and I&T sequencing can be used in a wide variety of applications. The only requirement for an application is that the knowledge about components, interfaces, fault states, and tests needs to be made more explicit and expressed in a model.

Acknowledgements

This work has been carried out as part of the TANGRAM project under the responsibility of the Embedded Systems Institute, partially supported by the Netherlands Ministry of Economic Affairs under grant TSIT2026. The authors would like to thank Johan Neerhof for his support in the wafer scanner case study. Furthermore, we would like to thank all TANGRAM project members for the fruitful discussions and valuable comments.

Bibliography

[1] ASML. Website, 2007. http://www.asml.com.

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

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

[4] I.S.M. de Jong, R. Boumen, J.M. van de Mortel-Fronczak, and J.E. Rooda. An overview of integration and test plans in organizations with different business drivers. In *Proceedings of the 5th Annual Conference on Systems Engineering Research (CSER'07), Hoboken, NJ, USA*, 2007. CD-ROM. Available online at http://www.stevens.edu/cser/.

[5] TANGRAM project. Website, 2007. http://www.esi.nl/tangram.

[6] J. Tretmans, editor. TANGRAM: *model-based integration and testing of complex high-tech systems*. Embedded Systems Institute, Eindhoven, the Netherlands, 2007. ISBN: 978-90-78679-02-8. Available online at http://www.esi.nl/tangram/.

[7] N.C.W.M. Braspenning, E.M. Bortnik, J.M. van de Mortel-Fronczak, and J.E. Rooda. Model-based system analysis using Chi and UPPAAL: an industrial case study. *Computers in Industry*, 59(1):41–54, 2008.

[8] N.C.W.M. Braspenning, J.M. van de Mortel-Fronczak, and J.E. Rooda. Modeling, analysis, and implementation of infrastructure for model-based integration and testing. Systems Engineering report 2007–08, Eindhoven University of Technology, 2007. ISSN: 1872–1567. Submitted for publication in IEEE Transactions on Automation Science and Engineering.

[9] R. Boumen. *Integration and test plans for complex manufacturing systems*. PhD thesis, Eindhoven University of Technology, 2007.

[10] L.S.H. de Mello and A.C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, 7(2):228–240, 1991.

[11] K.R. Pattipati, S. Deb, M. Dontamsetty, and A. Maitra. START: system testability analysis and research tool. *IEEE Aerospace and Electronic Systems Magazine*, 6(1):13–20, 1991.

[12] R.E. Shannon. *Systems simulation: the art and the science*. Prentice-Hall, 1975.

[13] N.C.W.M. Braspenning, D.O. van der Ploeg, J.M. van de Mortel-Fronczak, and J.E. Rooda. Model-based techniques for intelligent integration and testing in industry. In *Proceedings of the 17th International Symposium of INCOSE (INCOSE'07), San Diego, CA, USA*, 2007. CD-ROM.

[14] J.W. Horch. *Practical guide to software quality management*. Artech House, 2nd edition, 2003.

[15] J-P. Katoen. *Concepts, algorithms and tools for model checking*, volume 32–1 of *Arbeitsberichte der Informatik*. Friedrich-Alexander-Universität Erlangen-Nürnberg, 1999.

[16] E. Brinksma and J. Tretmans. Testing transition systems: an annotated bibliography. In *Revised tutorial lectures of the 4th Summer School on Modelling and Verification of Parallel Processes (MOVEP'00), Nantes, France*, volume 2067 of *Lecture Notes in Computer Science*, pages 187–195. Springer-Verlag, 2001.

[17] N.C.W.M. Braspenning, J.M. van de Mortel-Fronczak, and J.E. Rooda. A model-based integration and testing method to reduce system development effort. *Electronic Notes in Theoretical Computer Science – Proceedings of the 2nd workshop on Model-Based Testing (MBT'06), Vienna, Austria*, 164(4):13–28, 2006.

[18] R. Boumen, I.S.M. de Jong, J.W.H. Vermunt, J.M. van de Mortel-Fronczak, and J.E. Rooda. Test sequencing in complex manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 38(1):1–13, 2008.

[19] I.S.M. de Jong, R. Boumen, J.M. van de Mortel-Fronczak, and J.E. Rooda. Test strategy analysis for manufacturing systems. Systems Engineering report 2007–10, Eindhoven University of Technology, 2007. ISSN: 1872–1567. Submitted for publication in IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans.