Model-based testing with χ and TorX [1]

A case study of the ASML laser subsystem

Niels Braspenning, Asia van de Mortel-Fronczak, Koos Rooda

Within the TANGRAM *project, a case study on model-based testing of the* ASML *laser subsystem has been performed. The approach used in the case study is based on the proposed model-based testing framework, instantiated with state-of-the-art tooling from the* TANGRAM *project partners:* χ *as specification language and* TorX *as test tool. A* χ *specification model of the laser state behavior and communication interface has been developed. After verification and validation, the model has been used for automatic model-based testing with* TorX*. Using this approach, discrepancies between the implementation and specification of the laser subsystem have been found.*

One research topic of TANGRAM is model-based testing (MBT in short), that has already been a topic of the XOOTIC MAGAZINE [1]. In model-based testing, the behavior specification of a system under test is given by a formal model, which is a precise, complete, consistent, and unambiguous basis for testing. Using formal specifications for testing enables automatic processing by means of tools. Using a test derivation algorithm implemented in a test tool, test cases are automatically derived from the specification model and executed on the system.

One of the TANGRAM case studies concerns model-based testing of the ASML laser subsystem using the specification language χ and the test tool TorX. The objectives of this case study are to show the applicability of automated model-based testing using TorX within ASML, to show that χ models can be used for model-based testing, and to investigate the limitations and shortcomings of the approach used.

MBT framework

The proposed MBT framework is shown in Figure 1 and consists of the following elements:

- An *informal specification* of the correct behavior of the system under test expressed in a natural language (documentation) and present in the minds of the designers (mental model).
- A *(formal) specification model* of the correct behavior of the system under test expressed in an unambiguous specification language.
- A *formal test model* of the correct behavior of the system under test expressed in a test formalism that is suitable input for the test tool. Note that the specification model and the test model can be (but are not necessarily) the same.
- A *test tool* that is able to automatically derive tests from the test model, to execute these tests on the system under test, and to compare the test results with the test model behavior.
- A *test environment* that provides access to the interfaces of the system under test and enables stimulation and observation of these interfaces.
- A *system under test (SUT)*, which is the actual implementation that is tested together with the required context that is needed for testing.

[1] 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.

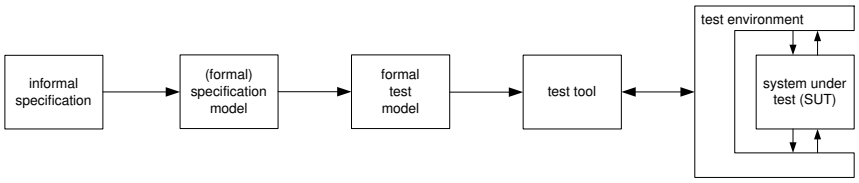

Figure 1: Model-based testing (MBT) framework

Tooling

When testing is to be performed automatically, some form of tooling is required. Looking at the MBT framework from Figure 1, the following tooling is needed:

- A specification language and test formalism in which the correct system behavior and the required test aspects can be expressed. The test formalism must be suitable input for the test tool.
- A test tool that is able to automatically derive tests from the test model using a test derivation algorithm and that is also able to automatically execute the derived tests on the SUT and compare the test results with the test model behavior.
- A test environment that connects the test tool to the SUT and enables stimulation and observation of interfaces of the SUT.

Looking at the TANGRAM project partners, good candidate tools for model-based testing would be χ [2] and TORX [3]. Within the Systems Engineering Group at the Eindhoven University of Technology, there is a lot of experience on the modeling, analysis, control, and optimization of manufacturing systems with the specification language χ, for both discrete-event and hybrid (i.e. including continuous behavior) systems. Using the high expressivity of χ for model-based testing will be beneficial in the future when the testing domain is extended towards time, data, and hybrid testing, because the current test formalisms are not expressive enough (discrete-event only) for specifying these other aspects.

The test tool TORX, developed at the Formal Methods and Tools research group at the University of Twente, is able to derive and execute tests on-the-fly, based on the *ioco* theory. Several case studies show successful application of the tool. Test formalisms that are currently supported by TORX are LOTOS and TROJKA, the latter one being a slightly adapted version of PROMELA [4]. The test domain of TORX is currently limited to the discrete-event domain, however extensions towards the data, time, and hybrid test domain are investigated within the TANGRAM project and other projects.

As it is a specific goal of the laser case study to investigate whether χ can be used for model-based testing, χ is chosen as specification language. However, χ cannot be used as a test formalism, as it is not a suitable input format for TORX. Because of this, and the fact that a direct connection between χ and TORX is considered as a future development, one of the supported test formalisms of TORX has to be selected to which the χ specification will be translated. Because of the resembling structures of χ and PROMELA and the existing experience in translating χ to PROMELA, TROJKA is chosen as test formalism for the laser case study. The usage of PROMELA also allows verification of certain properties of the model with the model checker SPIN.

Table 1: Properties of χ, PROMELA, TROJKA

Language property	χ	PROMELA	TROJKA
Simulation	√	√	√ (closed)
Verification	X	√	√ (closed)
Testing	X	X	√ (open)
Modeling expressivity	☺	😐	😐
Data	√	X	X
Functions	√	X	X
Time	√	X	X
Stochastics	√	X	X
Hybrid	√	X	X
Easy to modify	😐	☹	☹

The three specification languages mentioned above, χ, PROMELA, and TROJKA, are compared to each other according to certain properties in Table 1.

For the test environment, the current developments within TANGRAM on test infrastructure, also addressed in this XOOTIC MAGAZINE issue, are used, which provides easy access to the interfaces of ASML software components.

Case: ASML laser subsystem

For each exposure of an area (e.g. one chip) on a silicon wafer in a wafer scanner a beam of laser light is needed, that is provided by the laser subsystem. The laser subsystem is manufactured by another company than ASML, and has to operate together with the ASML wafer scanner to get good exposure results. To this end, a lot of communication is used between the scanner and laser, like commands, queries and responses, warnings and errors, control data, timing and synchronization triggers.

One condition of the case study is that only functional, untimed behavior is considered, so only the communication concerning commands, queries, and responses is taken into account. Although there are multiple (serial and parallel) communication interfaces between the wafer scanner and the laser, only the RS232 serial interface is used in the experiments, because this interface is easily accessible through the test environment.

Taking these limitations (functional behavior using the serial interface) into account and looking at the operational sequences in the laser subsystem documentation, the number of serial commands that can be tested is very limited. Many operational sequences use parallel commands (i.e. commands sent over the parallel communication interface) or can only be executed in the 'expose' state that requires parallel commands to reach. Therefore, only the laser state behavior (serial commands starting with 'LS') is considered, which limits the testable functionality to changing the laser state to standby and off, and to query the current state only. Nevertheless, this is still enough to show proof of concept.

Approach

From the informal specification in the form of documentation and mental models (revealed by talking with the ASML people involved), a χ model of the laser and an environment of the laser (necessary to get a *closed* system) is developed. To gain confidence in the model, the model is verified and validated against the informal specification by means of simulation.

Subsequently, the χ model is translated into PROMELA. Because PROMELA has several limitations concerning the modeling expressivity of χ, workarounds have to be found for the translation of certain χ model constructs. By means of simulation with SPIN, the PROMELA model is verified and validated against the informal specification and the simulation results of the χ model. Besides that, several model properties are verified using SPIN.

When there is enough confidence in the model, the PROMELA model is converted into the TROJKA test model, which involves only a few small modifications. It is important to mention that now only the laser part is converted, because testing is done using a system specification *without* environment (i.e. an *open* system). Finally, when TORX is connected to the TROJKA test model on one side and to the test environment that accesses the RS232 serial interface of the laser on the other side, the testing experiments are performed.

It is important to mention that for the first experiments, a hardware laser simulator (containing programmable electronics and cable connectors for the actual serial and parallel communication interfaces) is used instead of a real laser due to costs and safety issues. This hardware laser simulator is connected by cables to an ASML test rack, which is controlled by software. The approach of the laser case study, which is an instantiation of the MBT framework from Figure 1, is visualized in Figure 2.

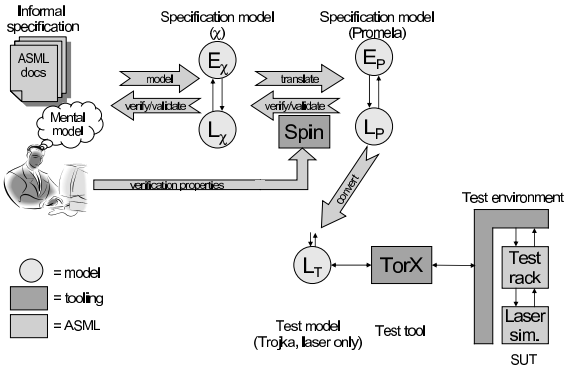

Figure 2: Case study approach

Modeling in χ

The χ specification model of the laser subsystem contains both the environment side and the laser side of the (serial and parallel) communication interface. The χ model, depicted in Figure 3, contains the following processes, which are interconnected by channels:

- The *environment* `Env` closes the system and can be configured (using an external configuration file) to generate specific command sequences for behavior validation, for example the operational sequences of the wafer scanner (as found in documentation).
- The *I/O interface* `IO` interfaces with the environment and passes through commands and responses to and from `LC` and `LS`.
- The *laser communication* `LC` process handles the commands from the environment (passed through by `IO`), performs the necessary actions (e.g. a state change), and creates the responses corresponding to the configuration that is loaded from an external file.
- The *laser state* `LS` process keeps track of the current laser state, in case the environment queries the current state.

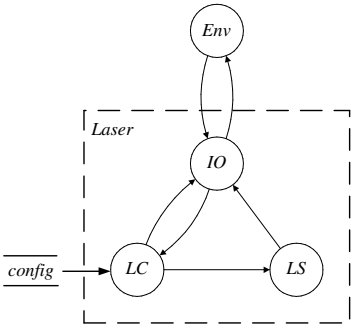

Figure 3: Processes and channels of the χ model

The χ model is *configurable* in a sense that the environment command sequences and the laser behavior can be changed easily in external files without changing and recompilation of the χ model itself. This easy changing of behavior already showed its advantage when it became clear that a certain laser type was not available in the laser simulator and another laser type had to be specified. Furthermore, the model contains *error handling* of 'unknown' commands (commands not understood by the laser) and 'bad context' commands (known commands that are not allowed in a certain state).

Figure 4 shows the behavior of the laser model that is to be tested by TORX. In this figure, the nodes depict the states of the model and the edges depict both commands/input (solid) and results/output (dashed). The central states at the top and bottom denote the actual laser states 'off' and 'standby' (numbered '00' and '03', respectively). The 'trans', 'error', and 'query' states are intermediate states between different LS (laser state) commands. Note that a state transition command to the current state results in a 'bad context' error ('??=02').

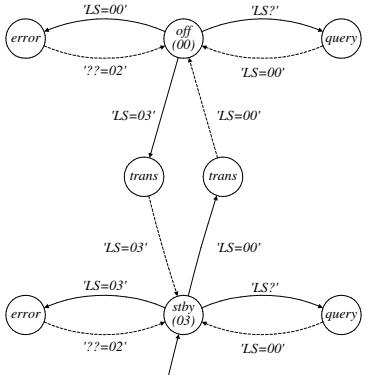

Figure 4: Laser behavior to be tested

The verification and validation of the χ model is performed by means of simulation. Several interesting scanner command sequences (e.g. operational sequences from documentation and bad weather (exceptional) behavior) are generated in process `Env` and the model is simulated. The simulation results show the same behavior as in the documentation and also the error handling functionality behaves as expected.

Translation to PROMELA

Because χ is not a suitable input for TORX, the χ specification model has been translated to PROMELA by hand, which is a laborious and error-prone task. For most of the χ constructs, a translation scheme from χ to PROMELA, developed in the TIPSY project [5], can be used. However, some specific χ constructs cannot be directly translated, for example lists, sets, (repetitive) selective waiting, and functions (e.g. the `pick` function to select one element from a set). For these cases, workarounds have been found and applied. As the translation is done manually according to some translation scheme, it is certainly not guaranteed that the translation is correct. Nevertheless, the resulting PROMELA model resembles the χ model as much as possible, which means that each statement in χ is translated into one PROMELA statement

or into one block of PROMELA statements that is preferably considered as one internal action (by using the `atomic` and `d_step` operators). The resulting PROMELA code is certainly not optimal and not the most efficient, which is due to the translation from χ.

Modeling the laser subsystem in PROMELA right away would probably result in a more elegant model, so in this case the benefits of using χ may not be really clear. However, one of the objectives of this case study was to investigate the possibility of using χ for MBT, and in this case the usable functionality of χ is limited to the functionality that is supported by PROMELA and TORX. So, the experience gained in this case study is beneficial when data, time and hybrid aspects are to be included, which are supported in χ, but not in PROMELA.

Table 2: Model properties of χ, PROMELA, and TROJKA

Model property	χ	PROMELA	TROJKA
Environment process Env	√	√	X
Laser processes IO/LC/LS	√	√	√
Serial interface	√	√	√
Parallel interface	√	√	X
Error handling	√	√	√
Configurable behavior	☺	☹	☹
#lines for model	350	800	350
Time to build	3 weeks	+3 weeks	+1 week

Verification and validation with SPIN

Just like the χ model, the translated PROMELA model is also verified and validated by performing simulation runs, in this case with the model checker SPIN. Again, several operational sequences and bad weather command sequences are generated in the environment and the results are as expected.

An advantage of having a specification model in PROMELA, is that SPIN can be used to verify certain model properties. Several generic properties like *deadlock freeness* and *no unreachable states* are successfully verified. Besides that, also some specific properties of translated χ constructs and of the laser behavior are verified and found to be correct, for example that:

- the PROMELA translation of the χ function `pick` (which takes an element from a set) always returns one set element;
- only certain state transition sequences are allowed;
- only certain replies are allowed to a command.

Conversion to TROJKA

With verification and validation, the confidence in a model grows. When there is enough confidence in the model, it is used for model-based testing. To this end, the PROMELA model needs to be slightly modified, which results in a TROJKA model that is suitable input for TORX. First of all, the TROJKA model is an *open* system, i.e. it does not contain the `Env` process from Figure 3. Another difference is that in a TROJKA model the channels that are *observable* to the outside world need to be defined, which is done by giving them the special attribute `OBSERVABLE`. Finally, the channel names have to conform to a certain naming convention to enable the connection of TORX to the system under test through the test environment.

Corresponding to Table 1 that shows properties of the specification languages χ, PROMELA, and TROJKA, a similar overview of laser model specific properties is given in Table 2.

Testing with TORX

Now that the specification side of the MBT framework (all elements on the left of the test tool in Figure 1) has been set up, the test tool has to be connected to the SUT. For the translation of the abstract commands from the TROJKA test model into the concrete commands of the SUT and vice versa, an adapter component (implemented in PYTHON) is used.

For each observable channel in the test model, a PYTHON adapter function has been created that handles the connection to the SUT, which involves translation from abstract commands into real commands, wrapping of specific command data (e.g. a left justified string of 128 characters). The other way around, also the real replies received from the SUT have to be unwrapped and translated back into the abstract replies as specified in the test model.

System under test: laser simulator

As already mentioned, a hardware laser simulator is used as system under test instead of the real laser due to safety and costs issues. This laser simulator is connected to a software controlled test rack and is developed by ASML to be able to test the wafer scanner software and electronics in the test

rack without a real laser connected to it. This saves a lot of expensive cleanroom time and is less dangerous. As the ASML wafer scanners are shipped to customers with different laser types, also the laser simulator can be configured for several (but unfortunately not all, as we experienced) laser types.

With a configured laser simulator, connected by cables to the ASML test rack where the software, electronics, and the test environment are up and running, the whole test setup as shown in the right bottom part of Figure 2 (consisting of the TROJKA test model, the TORX test tool, the test environment, and the laser simulator as SUT) is prepared for experimenting.

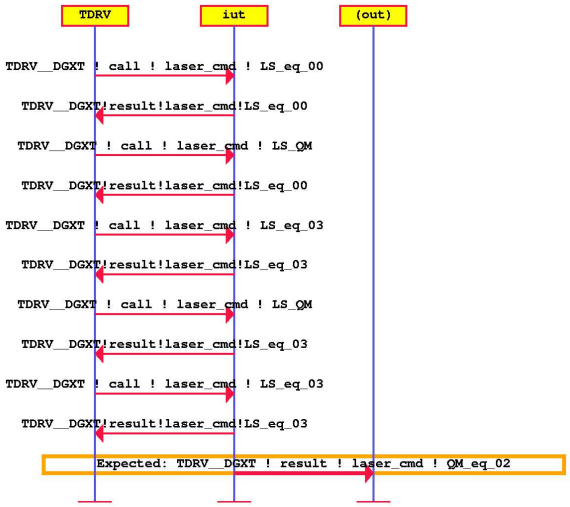

Figure 5: TORX has found a discrepancy!

Experiments and results

With the test setup as described above, the laser simulator has been tested automatically. Serial commands are selected from the TROJKA test model by TORX and sent to the laser simulator. The responses from the laser simulator are observed and compared with the behavior specified in the model. During the experiments two major discrepancies between the test model and SUT concerning state behavior have been found. One of these discrepancies is discussed in more detail below.

The specified laser behavior from Figure 4 shows that a state transition command to the current state (e.g. giving the command 'LS=03' in the 'standby (03)' state) should give a 'bad context' error ('??=02') as reply. However, the laser simulator replies with the current laser state instead ('LS=03' in this case). The TORX message sequence chart in Figure 5 shows the commands and replies leading to this discrepancy. Because the signs '=' and '?' are not allowed in PROMELA, they are replaced by 'eq' and 'QM', respectively.

Besides discrepancies in the implementation, also some errors and inconsistencies in the specification documents are found. Due to the general explanation in words, these specifications are incomplete, they can be interpreted in different ways, and sometimes they are even conflicting. Especially the specification of bad weather behavior (if it is specified at all) is not clear. For example, a lot of (operational) command sequences are specified separately, but nothing is explicitly stated about the remaining (e.g. bad weather) command sequences. Even if it is possible, it is very hard to extract this information from the informal specification. When making a specification model, the specification language explicitly forces a complete specification of all possible cases, for example in an if-elseif-else construct.

Conclusions

With the laser case study, a proof of concept is delivered that automatic model-based testing with TORX can be applied within ASML. Furthermore, it is also shown that χ models can be used for model-based testing. In this case the χ model is not directly used for model-based testing, however the structure of the χ model is maintained during the translation into the PROMELA and TROJKA models.

Developing a formal specification model starting from an informal specification is a difficult task, especially when a modeler is new to the system. The information is scattered over different documents, can be interpreted in different ways, is incomplete, and in some cases it is conflicting. Moreover, it is possible that parts of the informal specification are not documented, but stored in the minds of the designers (mental models). Therefore, talking to the people involved is very important to clear confusion, to reveal the mental models, and to validate your specification model.

The often heard argument that modeling a system takes a lot of time is not completely true. It is not the modeling (i.e. writing the specification down in some specification language) itself that takes a

lot of time, but the development of an *unambiguous specification*. The act of modeling itself forces the modeler to think harder about the system specification, which will result in a better understanding of the system and also in a more complete and less ambiguous specification.

Concerning the specification model of the laser subsystem in χ, the modeling is done according to the current way of working within the Systems Engineering Group at the TU/e. The configurability of the model can be considered as a new way of specifying behavior.

The translation from χ to PROMELA is a very laborious and error prone process that results in a loss of modeling expressivity, readability, and modifiability. Additionally, there is no certainty about the correctness of the translation, as it is done by hand.

One question that can be asked is whether it is beneficial to start modeling with χ instead of modeling directly in PROMELA. Currently the used functionality of χ is limited to what is possible with PROMELA and TORX, i.e. functional testing of discrete-event systems. The expressive modeling power of χ is yet untouched and all functionality that is used in the χ model is maintained in PROMELA and TROJKA (but certainly not in an optimal way). So for this case study, it would be reasonable to start modeling in PROMELA right away. However, when the data, time and hybrid test domain come into the picture (which will be the case in the near future), PROMELA will not suffice any more. Then the project can benefit from using χ and, therefore, this initial case study is useful and valuable for future research.

The approach described in this report enables automatic testing of the responses of the laser simulator, for both good and bad weather. The initial experiments concerning the laser state behavior tested limited functionality (because the interface accessibility was limited), however some discrepancies between implementation and specification of the laser simulator have been found.

Future work

A direct connection between χ and TORX is definitely required when χ specification models are to be used for model-based testing. Therefore, the first steps towards such a connection are being taken. To utilize TORX within ASML in a more easy way, the connection of TORX to the test environment (which now is done through the manually developed adapter component) will also be made more generic.

Besides that, more research is performed on model-based testing, especially regarding theory and tooling extensions towards the time, data and hybrid domain. First experiments show that a timed version of TORX is able to derive tests from a timed automata specification to test the functionality and some timing requirements (e.g. response time) of a system under test.

References

[1] XOOTIC MAGAZINE *'Testing' issue*, Volume 8, Number 2, November 2000.

[2] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers, *Syntax and Consistent Semantics of Hybrid Chi*, Computer Science Reports 04-37, Technische Universiteit Eindhoven, November 2004.

[3] J. Tretmans and E. Brinksma, *TorX: Automated model based testing*, In 1st European Conference on Model-Driven Software Engineering, December 2003.

[4] Gerard J. Holzmann, *The model checker SPIN*, Software Engineering, 23(5):279–295, 1997.

[5] E. Bortnik, N. Trčka, A.J. Wijs, B. Luttik, J.M. van de Mortel-Fronczak, J.C.M. Baeten, W.J. Fokkink, and J.E. Rooda, *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.

Contact Information

Niels Braspenning

Technische Universiteit Eindhoven
Department of Mechanical Engineering
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
n.c.w.m.braspenning@tue.nl