TESTING AGAINST REQUIREMENTS — A SYSTEMATIC APPROACH FOR IDENTIFYING REASONABLE TEST CASES

Carsten Stechert[a] and Thomas Vietor[b]

Technische Universität Braunschweig, Institute for Engineering Design
Langer Kamp 8, 38106 Braunschweig, Germany.
Email: [a]*stechert@ikt.tu-bs.de,* [b]*vietor@ikt.tu-bs.de*

Project monitoring is a crucial component of product development. The product development has to focus on the actual market needs (requirements), avoid errors, and guarantee the required high quality. Testing against requirements is the key factor to develop the right product at the right time. Requirements are documented in lists or even modelled in special modelling languages. Test criteria must evolve from these requirements and they must be allocated to test cases and suitable test tools. This contribution shows a systematic approach to classify test cases for the development of a group of complex mechatronic products: parallel robots. With the help of design catalogues, a classification that covers mechanical, electronic, and software related test cases was built up. This results in a shared understanding of activities, interfaces, and terms that will assist in more effective product development.

Keywords: Design Theory and Methodology, Design for X, Knowledge and Product Life Cycle Management.

1. INTRODUCTION

Validating and testing are important activities in product development and project monitoring. Development risks are to be reduced by eliminating error sources on the one hand and by focussing on the market needs on the other hand. In order to focus on the market needs, the fulfilment of all afore established requirements has to be tested. The more risky a requirement is, the earlier in the product development process it has to be tested. Regarding testing from an economic perspective, one should strive for the minimum number of different tests. Hence, test cases should be able to cover a large number of different requirements. Ideally, test cases can be automated, but which requirement is crucial to be tested and in which phase should it be tested? What is the appropriate method and tool to test it?

The contribution shows a systematic approach to classify test cases. These test cases are further described by important attributes (e.g., expense, coverage, test tools) and the information is stored in a knowledge based database, a so-called design catalogue [1, 2]. An earlier developed life cycle oriented requirements modelling approach [3, 4], based on the Systems Modelling Language (SysML) [5], is extended to develop test criteria from requirements. The systematically described test criteria are compared with the classes of test cases in a way that the most suitable test case class can be found. From the generic test case, a concrete test case is developed and modelled in the modelling environment.

The described approach was developed in the field of high dynamic parallel robots for handling and assembly [3, 6]. A development environment [7] was programmed, enabling configuration of robots and the exportation of pattern data to and the importation of concretised data from diverse expert tools, such as CAD, MBS, CAS. This allows for continuous use of model data. The next step is to identify test cases that can be integrated into the development environment. A short example will demonstrate the

Research into Design — Supporting Sustainable Product Development. *Edited by* Amaresh Chakrabarti
 :: *Published by* Research Publishing
ISBN: 978-981-08-7721-7

applicability of the above method to identify automatable test cases for this development environment. However, the method itself is not restricted to this specific area. Ongoing research deals with the transfer of these methods to other areas, such as automotive engineering.

2. STATE OF THE SCIENTIFIC AND TECHNICAL KNOWLEDGE

Most product development processes start with the clarification of the task and the creation of requirements lists [8–10]. Requirements are taken into account during the whole product development process for decision making. In addition, as the project proceeds further, a larger amount of information becomes available and additional requirements will be stated. After synthesis, the product has to be tested against the initial requirements. Thus, testing is the mirror image of clarification of task. Test criteria are the reflections of requirements.

In the following section, a short overview of testing is given followed by a short introduction into the systematic modelling approach used for this work.

2.1. Testing

With regard to project monitoring, resources such as time and costs have to be controlled for the duration of the project. When searching under the keyword “project management”, much is available in terms of literature, methods, and software tools.

Within the closer circle of product development, a variety of methods are available. These are suitable for specific phases and different objectives. In early phases, it is necessary to review target achievement by considering estimated product characteristics. Non-suitable concepts can be eliminated from the solution space and suitable concepts can be ordered in reference to applicability for a special task.

In later phases, and especially for products with a large amount of software and electronics components, or those posing a possibly higher risk to the safety and well being of humans, as well as those products where a high monetary risk is existent, formalised and well thought out test methods are necessary, and in part mandatory. The use of Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA) to identify error sources and estimate risk for pump units of nuclear reactors is one example.

On the one hand, risks are due to unexpected or unintended uses of the system (e.g. collision of a car with a kerbstone). On the other hand, risks can also be due to insufficient product quality, poor design, or faulty manufacturing. On the basis of risks, or rather risk priority, it must be decided which requirements have to be tested, when to test, and the amount of labour necessary to carry out the testing. Then, test methods are defined for components and activities, e.g. design review, inspection, prototyping, simulation, walkthrough, agile specification quality control [11].

Every test method needs test criteria and it makes sense to evolve them from requirements. Thence, they can be related to requirements lists and models (e.g. [12]). In early phases, abstract requirements are most suitable, while in later phases more detailed requirements should be used. Requirements are prioritised as strict, minimum (maximum) or wish (e.g. [8]). Concepts that do not fulfil strict requirements or that are not within the limits of minimum requirements are discarded. Minimum requirements and wishes are used to rank the remaining concepts. This allows for the consideration of the most capable concepts for further development. In the end, the fulfilment of all requirements has to be confirmed. Especially, those requirements that were fixed as acceptance criteria in the contract have to be tested in-depth. A good test criterion has the following characteristics [11]:

- The criterion can be tested.
  - — The test method can be carried out.
  - — The value to be tested can be measured.
  - — The test method is reproducible.
- The criterion is complete according to the respective requirement.
- The criterion is minimal according to the respective requirement.

All of the different test criteria are bundled in test scenarios or test cases that describe the pre-condition, testing procedure, and post-condition. The aim is to reduce the overall amount of labour and working time by covering a large amount of test criteria with a single test case.

2.2. Modelling

According to [13] a model should represent the subject to be modelled, ignore unimportant details (abstraction), and allow for pragmatic usage. The purpose of a model is to support and improve the understanding of the matter and build a common basis for discussion and information exchange. Moreover, models should allow for a comparison of different solutions as well as an analysis and prediction of behaviour and characteristics of the system to be designed. The organisation of a model should contain its structure and architecture. Furthermore, interactions between components, component interdependencies, and important external relations should be taken into account.

The Systems Modelling Language (SysML) [5] is an object-oriented approach to model a product on different levels of abstraction and from different viewpoints. It is a widely known notation within the fields of software development, electronic design, automation, and (in parts of) mechanical engineering. A variety of different (commercial) software tools are available. Here, SysML is used as a notation understood across multiple disciplines for early modelling and knowledge structuring. Within this project, SysML is extended to be able to consider the following components of the early phases: goal-system, product-lifecycle, stakeholder-network, product-surroundings, project-/product-monitoring, and system-context. The test model is part of the project-/product-monitoring. More detailed information about the whole modelling concept and SysML extensions can be found in earlier work of the authors [3, 4, 14].

In Figure 1 a brief overview of a model for parallel robots is shown to illustrate the approach. On the right side, use cases are shown as objects. One use case in the product lifecycle phase "Use" describes the handling of muffins. This use case leads amongst others to a refinement of the requirements workspace and payload. Besides fulfilling the specific use case "handling muffins", one important goal is a short cycle time. There are a number of targets that support this goal. However, not every target is related to the development of the robot, i.e. not within the projects boundaries. The target "high dynamic" is directly related to the robot and supported by the requirements "high acceleration" and "high speed". As a simplified example, the tripolar quantitative relationship between acceleration, speed, and workspace can be described by an equation considering constant acceleration at the tool centre point (TCP). As long as the concretion level is low, this simplified equation can give a reasonable approximation. However, it contains the danger of prejudgement.

3. TEST MODEL

The test model is used to review target achievement as early as possible and to guarantee a high-quality product that fulfils the customer wishes. To achieve this, a description is given on how the system and

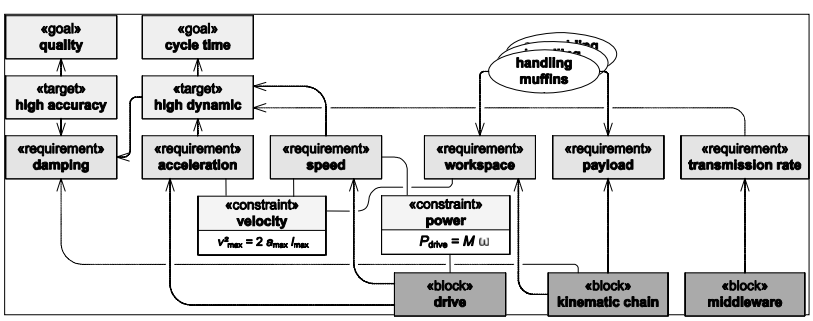

Figure 1. Part of a diagram for the relational network of an extended SysML model for parallel robots.

its components should be tested. After a description of test criteria, the systematic classification of test cases is discussed. Then, some test tools for parallel robot testing are introduced and a brief example of a concrete test case is shown.

3.1. Test criteria

The test model consists of three different types of objects: test criteria, test cases, and test tools. Test criteria evolve from requirements and are related to the requirements model via "*satisfy*" relations. If the test team is not able to define a reasonable test criterion, the corresponding requirement must be revised. Either the formulation of the requirement was not precise enough or no reasonable way to control its fulfilment exists. In the latter case, the requirement should be removed from the requirements model.

Test criteria are created in the model as "*requirement*" and the newly developed stereotype "*Test criterion*" is added. In this way, a test criterion acquires the following attributes:

- *What is tested?* This attribute describes the type of characteristic of the test object being tested (function, structure, or behaviour).
- *Direct testing*: The criteria is directly verifiable or indirectly with an in-between parameter, e.g. testing force by deflection of a spring.
- *Discipline*: Several different disciplines are necessary to develop a complex, mechatronic product. Test criteria from a special discipline sometimes need special test methods, e.g. vibration test, software-in-the-loop.
- *Risk*: Different test criteria are unequally risky. In this contribution risk is described following FMEAs risk priority number (RPN) that is the product of the following numbers:
  - — Severity rating,
  - — Occurrence rating, and
  - — Detection rating.
- *Author*: The person that created the test criterion.
- *Responsible*: The person that is responsible for the test and fulfilment of the test criterion.
- *State*: Describes the current state of the object:
  - — Created,
  - — Signed,
  - — Revise,
  - — Proofed, and
  - — Refused.

3.2. Test case classes

Test criteria are tested through test cases. The relation is implemented in the model via the SysML-relation "*verify*". A test case is created as "*UseCase*" and the SysML-stereotype "*Test case*" is added. For a systematic proceeding, test cases are structured. Many different classifications exist within the different disciplines (e.g. code and unit test, module test, and system test in software engineering). A consistent classification that can be used for complex mechatronic products is not available at the moment.

In Figure 2 a design catalogue that was developed for a systematic description of test cases for parallel robots is shown. It is realised in Excel and contains substantial descriptions and annotations, all easily accessible via hyperlinks. Design catalogue (cp. [1, 2, 15]) is a tool to systematically structure knowledge, make it easily accessible, and provide a means to identify new solution ideas ("white fields"). It consists of classification-part, main-part, access-part, and annex. Each part consists of a limited pre-defined set of attributes. The test case catalogue shown has the following three attributes in its classification-part, hence 36 distinguishable classes of test cases.

classification part			main part			access part												
1.1	1.2	1.3				1.4	1.5	1.6	1.7	1.8	1.9	1.10	1.11	1.12	1.13	1.14	1.15	1.16
What is tested?	Test object	Aggregation level	name	description	Nr.	dynamic	combined testing	previous knowledge	who carries out the test?	phase of the development process	test loop	test environment	interface	automatability	time period	effort	transferability	coverage
1	2	3	1	2	Nr.	1	2	3	4	5	6	7	8	9	10	11	12	13
function	abstract	part	concept tests	evaluation with general information…	1	3	1 +2 +3	1	1	1	1 +2	1	1	2 +3	3	3	3	3
		assembly		evaluation with general information…	2	3	1 +2 +3	1	1	1	1 +2	1	1	2 +3	3	3	3	3
		machine		evaluation with general information…	3	3	1 +2 +3	1	1	1	1 +2	1	1	2 +3	3	3	3	3
		plant		evaluation with general information…	4	3	1 +2 +3	1	1	1	1 +2	1	1	2 +3	3	3	3	3
	virtual	part	function simulation	evaluation with simulation…	5	1	1 +2	2	1	2	1 +2	1 +2	1 +2	2	2	2	1	1

Figure 2. Design catalogue for general test case classes for complex mechatronic products. Numbers in the access-part represent the special characteristic of the attribute.

- *1.1 What is tested?* The first classification attribute describes the abstract field of what is tested. The IEEE suggests a division into object-oriented, process-oriented, and behaviour-oriented approach for tests of software systems [16]. For a more general, interdisciplinary applicable division, this contribution suggests the terms function, structure, and behaviour (cp. [17]).
- *1.2 Test object*: The second classification attribute describes the level of abstraction of the test object at test time: abstract (e.g. function structure, sketch), virtual (e.g. CAD model, DMU), or real (e.g. rapid prototype, conventional prototype). This attribute implies the point in time when the test can be carried out, since virtual and real test objects are not available until later phases of the product development process.
- *1.3 Aggregation level*: The aggregation level describes the complexity of the test object from "assembly part" to "plant". This is because at a level "assembly part" it is possible to test in more detail (e.g., FEM) than at a level "plant", where (e.g., due to computing power) often only integration tests are possible.

The access-part contains further attributes that are not appropriate for the classification-part because of mutual dependencies (e.g., coverage influences effort and transferability). In addition, general test case classes can not always be assigned with one single value of an access-attribute. In the following the attributes of the access-part are described:

- *1.4 Dynamic*: Tests can be carried out statically (or rather quasi-statically) or dynamically.
- *1.5. Combined testing*: Combined testing describes whether a test criterion can or should be tested on its own or in combination with others. As an example, maximum dimensions of an object (pack size) can be measured for each direction in space and compared with a reference value. If the object is put into a form, the allowed limits of dimensions are tested in combination (cp. control of hand baggage size at the airport). In the latter case, the test result is simply positive or negative. If test criteria are tested individually, information about the gap is possible (e.g. the bag is 2 cm to wide).
- *1.6 Previous knowledge*: This attribute describes how much system internal information must be known to be able to carry out the test. In a black-box test, just the interfaces of the test object are known (e.g. hardware-in-the-loop). In a white-box test, the whole system structure is known (e.g. FEM).

- *1.7 Who carries out the test?* The product, or rather the product's components, is tested during product development by different groups of people. It can be distinguished between tests of the developer, a test team, or the user. The control loop grows bigger from developer to user tests; therefore, the response time becomes longer. A developer integrates the necessary changes on the basis of his own tests simultaneously during the development process. Changes that are induced not until tests of the user (acceptance test) are very extensive and expensive. For example, expensive adaptronic components have to be integrated in a finished machine tool design to compensate oscillations.
- *1.8 Phase of the development process*: The product development process is commonly divided into different phases. This attribute describes which phase the test is carried out in.
- *1.9 Test loop*: The test object is embedded into a test environment that provides input data. In a closed test loop, output from the test object is used in the test environment to determine new input data. In tests without a closed loop the environment only delivers predefined data sets as input and simply records the output.
- *1.10 Test environment*: The test object has to be embedded into a test environment. This test environment can be of a different level of abstraction (similar to the test object).
- *1.11 Interfaces*: Describes the kind of interface between test object and test environment. Interfaces can be abstract or real. In between abstract and real interfaces, so-called stubs are used. Stubs provide input and accept output data from the test object on that abstraction level and transform it to the (different) abstraction level of the test environment. For example, real computer cards are plugged into real computer slots. The environment virtually simulates different systems and different system states.
- *1.12 Automatability*: The execution of a test is, in general, time consuming. Hence, an attempt is made to automate as much of the test procedure as possible. This attribute describes whether a test is fully or partly automatable.
- *1.13 Time period*: The duration of the tests can vary. For example, a whole car corrosion test is long term (month) while the function test of a single procedure in a software code is generally short term (minutes).
- *1.14 Effort*: This attribute describes qualitatively the effort to be invested. Effort is needed for preparation (e.g. building the test bench), execution (e.g. record data), and pre-process (e.g. interpretation of test data). At this level of classification, a distinction is not made between effort in time or money.
- *1.15 Transferability*: This attribute considers the possibility to transfer the test results to real-life conditions of the system. Every test underlies special boundary conditions and emulates reality with simplifications. If real conditions are extremely simplified, (e.g. in early phases) a test can just suggest a rank order of better or worse solutions.
- *1.16 Coverage*: The coverage describes the testable state space in comparison to the possible state space. For example, for net-like software systems where different states can be passed through repeatedly, complete coverage is hardly possible. Errors that just appear for unlikely and unforeseen sequences of states are rarely found. The same is true for experiments with numerous influencing factors. The combination of factors results in a huge (for analogue systems almost infinite) number of parameter collectives. Statistical test planning helps to find the most likely states for testing.

3.3. Test tools

As a third component test tools are constituted as blocks ("*block*") in the model. Their basic job is to document the characteristics of the test tool and serve as knowledge storing elements. For example, Unigraphics, as a CAD system, provides interference control of the assembly at defined configurations and poses.

Specific test tools can be used to review test criteria of special test case classes. On the left side of Figure 3, test case classes are shown. For example in TC7, function is tested on machine level.

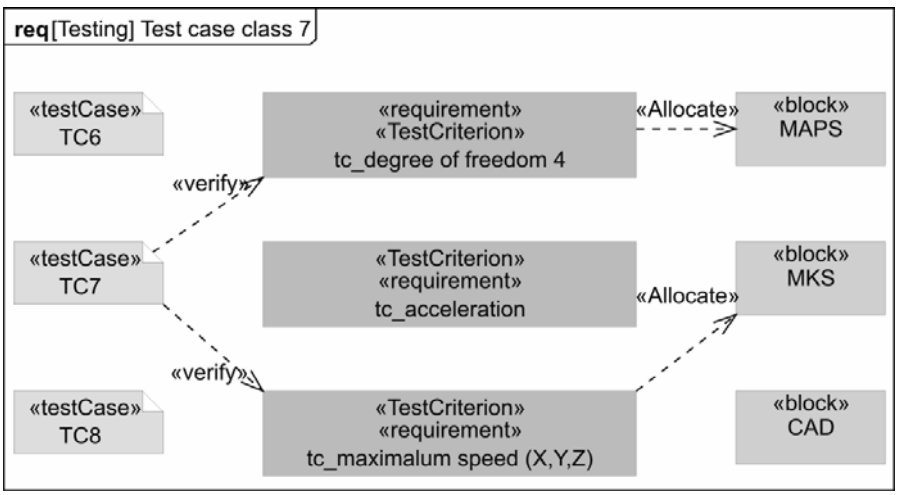

Figure 3. Part of a diagram for the relational network of an extended SysML model for parallel robots.

From the design catalogue (cp. Figure 2) it can be learned that results of this class can be transferred to reality quite certainly while the effort is manageable. In the middle of the diagram test criteria are displayed ("*Test criterion*"). These can be verified by test cases of test case class TC7. This fact is modelled through "*verify*"-relations. For example, it should be verified that the robot system provides a degree of freedom (DOF) 4 at the tool centre point (TCP). On the right, the possible test tools are shown. "*allocate*"-relations show that a test criterion can be tested with a special test tool. The DOF can be tested through a test tool called MAPS, which is a Matlab-based program for kinematic analysis of parallel structures.

3.4. Test case

Within the test case class TC7, the function of a machine is tested on the virtual test object. In the following, a concrete test case of this class is described:

Pre-condition: The CAD model of the kinematic structure is available and all basic boundary conditions and constraints (e.g. allowed DOF of the joints) are implemented in the model. The input values (angle of drives) are controllable by parameters. A test trajectory is integrated into the model and the end-effector is fixed to this trajectory by constraints in such a way that the desired movements can be carried out. The end-effector is located at the defined starting point.

Execution: The working platform can be manually "pulled" along the trajectory, so that the structure follows the movement due to the implemented constraints. Using on-board means of Unigraphics it is confirmed that no collisions appear during the movement. In addition, it is proved that the movement can be carried out along the whole trajectory.

Post-condition: The working-platform is located at the end of the test trajectory. No collisions or blockings were detected. → The function is satisfactory.

Limitation: This test case does not consider any existence of singularities.

4. CONCLUSION AND OUTLOOK

Requirements management is the core of successful product development. However, requirements management accompanies testing. If the product was not tested against requirements, it can not be guaranteed to be successful.

A modelling approach is presented that is based on SysML. The SysML is extended to the needs of methodical product design. This contribution focuses on the integration of testing into the interdisciplinary modelling approach. Therefore, test case classes were identified and systematically structured in design catalogues. For the example of parallel robotic systems, test cases were developed. These verify test criteria that evolve from the requirements model and can be tested through those test tools available in the project.

In ongoing research activities an Excel tool is programmed for analysis of the SysML model. With this tool, test criteria and test cases are automatically displayed in a matrix making it possible to easily decide if a test criteria is being tested at the right time. Furthermore, the tool shows how many requirements can be validated by test criteria. Hence, it guarantees that the project is on time and that the product focuses on the market and customer needs.

ACKNOWLEDGMENTS

The authors gratefully thank the German research association (DFG) for supporting the Collaborative Research Centre SFB 562 “Robotic Systems for Handling and Assembly — High Dynamic Parallel Structures with Adaptronic Components”.

REFERENCES

1. Roth, K., Franke, H.-J. and Simonek, R. “Aufbau und Verwendung von Katalogen für das methodische Konstruieren.“ Konstruktion, 24, 449–458, 1972.
2. Franke, H.-J., Löffler, S. and Deimel, M. “Increasing the Efficiency of Design Catalogues by Using Modern Data Processing Technologies.” International Design Conference (DESIGN 2004), pp. 853–858, 2004.
3. Schmitt, J., Stechert, C., Raatz, A., Hesselbach, J., Franke, H.-J. and Vietor, T. “Reconfigurable Parallel Kinematic Structures for Spreading Requirements.” IASTED International Conference on Robotics and Applications (RA09), 2009.
4. Stechert, C. and Franke, H.-J. “Requirements Models for Modular Products.” International Conference on Research into Design (ICoRD ’09), pp. 411–418, 2009.
5. Object Management Group. Homepage of the OMG Systems Modeling Language. http://www.omgsysml.org/, 02.7.2010.
6. Stechert, C., Pavlovic, N. and Franke, H.-J. “Parallel Robots with Adaptronic Components — Design Through Different Knowledge Domains.” IFToMM World Congress, 2007.
7. Stechert, C., Alexandrescu, I. and Franke, H.-J. “Modeling of Inter-Model Relations for a Customer Oriented Development of Complex Products.” International Conference on Engineering Design (ICED07), 2007.
8. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H., Wallace, K. and Blessing, L. “Engineering Design: A Systematic Approach.” Springer, 2007.
9. Ulrich, K. and Eppinger, S. “Product Design and Development.” McGraw-Hill/Irwin, 2007.
10. Mello, S. “Customer-Centric Product Definition: The Key to Great Product Development.” Amacom, 2001.
11. Rupp, C. “Requirements-Engineering und Management — Professionelle, iterative Anforderungsanalyse für die Praxis.” Carl Hanser Verlag, 2007.
12. Kickermann, H. “Rechnerunterstützte Verarbeitung von Anforderungen im methodischen Konstruktionsprozeß.“ Ph.D. Thesis, Institute for Engineering Design. Technischen Universität Braunschweig, 1995.
13. Avgoustinov N. “Modelling in Mechanical Engineering and Mechatronics —Towards Autonomous Intelligent Software Models.” Springer, 2007.
14. Stechert C. and Franke H.-J. “Managing requirements as the core of multidisciplinary product development.” CIRP Journal of Manufacturing Science and Technology, Vol. 1, Issue 3, 2009.
15. Kirchner, K., Drebing, U. and Franke, H.-J. “Konstruktionskataloge für den effizienten Einsatz physischer Modelle im Produktentwicklungsprozess.“ Konstruktion, 10, 62–66, 2009.
16. IEEE Standard 830. IEEE Recommended Practice for Software Requirements Specifications. IEEE Software Engineering Standards Committee, 1998.
17. Alvarez Cabrera, A.A., Erden, M.S. and Tomiyama, T. “On the Potential of Function-Behavior-State (FBS) Methodology for the Integration of Modeling Tools.” CIRP Design Conference — Design Synthesis. pp. 412–419, 2009.