Systematic Model-in-the-Loop Test of Embedded Control Systems

Alexander Krupp and Wolfgang Mueller

Paderborn University/C-LAB, Fuerstenallee 11, 33102 Paderborn, Germany
alexander.krupp@c-lab.de, wolfgang.mueller@c-lab.de

Abstract. Current model-based development processes offer new opportunities for verification automation, e.g., in automotive development. The duty of functional verification is the detection of design flaws. Current functional verification approaches exhibit a major gap between requirement definition and formal property definition, especially when analog signals are involved. Besides lack of methodical support for natural language formalization, there does not exist a standardized and accepted means for formal property definition as a target for verification planning. This article addresses several shortcomings of embedded system verification. An Enhanced Classification Tree Method is developed based on the established *Classification Tree Method for Embeded Systems* CTM/ES which applies a hardware verification language to define a verification environment.

1 Introduction

Verification cost has become a major cost factor in mechatronic systems development and in electronic design. Verification cost mitigation has become a priority, and any efficiency increase in verification contributes substantially to overall development efficiency.

This article describes an approach which supports the definition of a functional verification plan for mechatronic systems with full support for testbench automation, traceability, visibility, and repeatability. We introduce a methodology to close the gap between requirements and test definition by means of an enhanced classification tree method (CTM). It supports functional stimulus patterns, acceptance criteria which are compatible to the stimulus definition, and test quality criteria. The latter relate to requirements and they enable requirements coverage. Horizontal and vertical reuse is facilitated by the unified notation of the enhanced CTM. A concept for automation of the testbench execution is presented to reduce cost- and time -intensive manual human intervention in the verification process. Ideally, this leads to higher throughput of test cases due to reduced setup times for faster or more intense testing.

1.1 Mechatronic vs. Electronic Systems Design

Todays mechatronic systems development processes are increasingly dealing with a formal model of the mechatronic system, which enables code generation as well

as early verification of system features. Currently, model-based development is an accepted methodology in mechatronics systems design, which is being established in industry. The use of models and associated code generation replaces traditional manual coding for electronic control units. The increasing development productivity enables the creation of models of increasing complexity. While code generation removes many sources of coding errors, it cannot remove flaws in the models themselves. However, the use of models in mechatronic systems development opens up additional opportunities for verification. E.g., the automotive industry applies test and simulation environments at several levels of abstraction. Model-In-The-Loop (MIL) environments are applied to tests at model level with, e.g., MATLAB/Simulink. The integration of hardware and software on an embedded control unit (ECU) is being tested by means of a Hardware-In-The-Loop (HIL) environment. Sometimes additional abstraction levels, such as Software-In-The-Loop (SIL) and Processor-In-The-Loop (PIL) technologies are applied. Recently, with increasing computation power, the construction of virtual prototypes, i.e., a combination of behavioral and geometrical models, has become feasible. The focus of existing technology is on efficiency gains in the development and in the integration of models and hardware rather than extensive verification. The existence of a formal mechatronic model, however, paves the way towards extensive verification beyond the capabilities of a physical prototype. With formal verification for mechatronic systems being introduced mostly at research level, the most widespread mode of verification remains simulation and testing. Existing test tools by, e.g., National Instruments, dSPACE, Etas, Vector, and MBtech are rather specialized and apply proprietary languages and proprietary concepts.

Todays test patterns for mechatronic systems are either defined manually as fixed waveforms, or generated automatically from models. Automatic test pattern generation for mechatronic systems either derives test patterns from the model-under-test itself, or it requires the redundant creation of a reference model at the same level of abstraction [22]. The drawback of the first approach is that it does not support the generation of test patterns to detect missing functionality. The drawback of the second approach is that it requires the development of another model of similar complexity as the model-under-test. Both approaches derive their test patterns from models instead of from requirements. The relation of requirements to a model and, consequentially, to generated test patterns, remains unspecified.

Classification Tree Method CTM and CTM/ES *Classification Trees* were introduced during the early 90s by Grimm and Grochtmann for the structured representation of test cases [10, 11]. The construction of classification trees and their associated combination tables is supported by the Classification-Tree Method (CTM), which is derived from the category-partition method [19]. In its basic form, a classification tree and the accompanying combination table describe abstract high-level test cases in a graphical manner without an explicit notion of time. Since 1999, the method and notion has been enhanced by Con-

rad and Fey to accommodate the description of time-dependent test scenarios termed test sequences [5, 4]. These extensions are known as the Classification-Tree Method for Embedded Systems CTM/ES. The CTM/ES has recently been successfully employed in different control software development projects [17]. One of the main application areas is the testing of in-vehicle software developed in a model-based way [21]. Strengths of the CTM/ES approach are the description of time-continuous test patterns [6], it may also be applied as a front-end to Time Partition Testing [18].

The CTM/ES is mainly applied in the automotive domain. Several tools exist for editing classification trees (CTE/XL, Razorcat CTE) and for test data derivation support (MTest). The syntax of classification trees is a simple graphical notation. Its main advantage is the combination of discrete and continuous elements by means of interpolation for stimuli generation. Randomized stimuli instantiation is supported. However, it does not provide a gradual path towards directed test data definition.

Functional Verification In the domain of electronic design the concept of model-based engineering across several levels of abstraction has been employed for several decades. Formal verification, simulation and testing are employed on a regular basis. The increasing demand for verification at an early abstraction level, like system level, has led to the creation and introduction of methods and languages for *functional verification*. Functional verification is a methodology, which encompasses formal verification as well as simulation approaches. It builds on the declarative formulation of design properties as formal requirements, which provide a redundant path from natural language and semi-formal requirements to the design to enable consistency checks. Once defined, the formal properties can be applied for formal verification as well as for verification by simulation. Meanwhile, libraries and methodological guidelines have become available to supplement the tooling and standardization efforts, such as the *Verification Methodology Manual for SystemVerilog* and the *Open Verification Methodology*. Today, the domain of electronic design is able to apply a rather complete methodology for functional verification of digital designs [2].

2 Shortcomings of Mechatronic System Verification

Model-based development requires a thorough verification approach at modeling level before any code generation and implementation on an execution platform is performed. Generally, for verification purposes a requirements document in model-based development has to be accompanied by a verification document, the so-called *verification plan*. This document captures information, which does not belong into the requirements document, but is yet essential for successful implementation of a substantial verification task. As shown in figure 1, requirements guide the development of a mechatronic system model, whereas the verification plan determines verification goals, which are derived from the requirements as

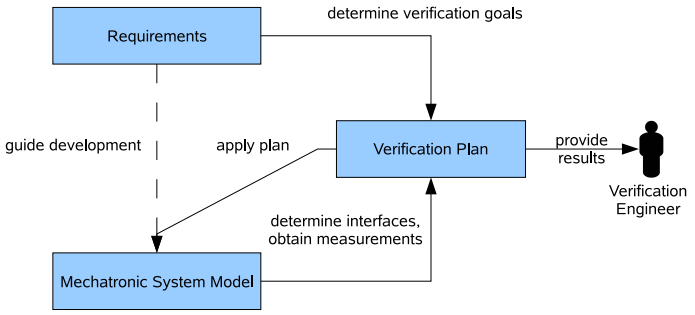

Fig. 1. Verification Plan, Requirements and Mechatronic Model

well. For testing purposes, the verification plan also needs to determine the actual system interfaces from the actual model-under-test. On application to the model-under-test, execution of the verification plan connects a testbench to the model interfaces, which then provides stimuli to the model-under-test, obtains measurements from it and provides verification results to the verification engineer.

In comparison to the electronic design verification approach, we observe several methodical shortcomings in mechatronic system verification.

1. Support for traceability and visibility is limited.
2. A methodical gap exists between requirements and testbench definition for mechatronic systems. This is due to
   (a) missing methodical support for the derivation of test descriptions from natural language requirements,
   (b) missing stimulus patterns which describe requirements,
   (c) missing requirements-based acceptance criteria,
   (d) missing test quality criteria for requirements coverage, which are easily derived,
3. Missing horizontal and vertical re-use of test definitions.

The limited support for traceability and visibility is due to the lack of functional coverage definitions, which can provide an independent means of requirement coverage measurement for tracing tested requirements, and for visibility of the current state of verification. The methodical gap from requirements exists for several reasons: a methodical support for derivation of test descriptions requires a suitable target. As most requirements allow an infinite number of possible stimuli, a directed stimulus definition cannot capture such a requirement, as it represents a single instance of stimulus only. However, the CTM/ES provides a first step for the definition of functional stimulus patterns. The definition of requirements based acceptance criteria for automatic acceptance evaluation is only possible usually by definition of accepting predicates. Existing predicate languages do not cover the definition of characteristic acceptance criteria for continuous systems. Existing proposals for test quality criteria are usually

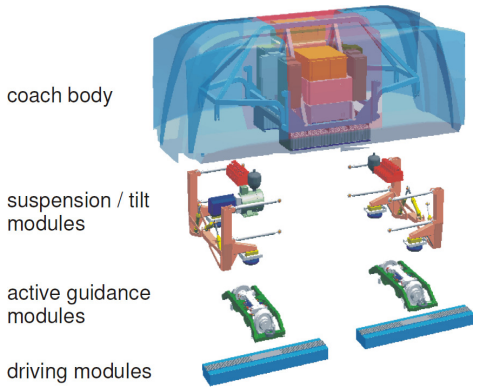

Fig. 2. Modules of the RailCab Shuttle

based on structural coverage criteria, which do not easily relate to requirements. Moreover, the re-use of test descriptions is only possible with high effort. Similar drawbacks have been described for mixed-signal verification in [3].

3 Example System: RailCab Suspension-Tilt Module

RailCab is a linear motor driven train system developed by the University of Paderborn [23]. RailCab is based on shuttles, which are composed of modules, which are arranged as shown in figure 2. The coach body is mounted on two suspension-tilt modules, which are used for active suspension and tilting. The suspension-tilt modules are coupled to the active guidance modules via air springs. The active guidance modules can actively rotate the axles relative to the rails to avoid striking the flange against the rail head, they also facilitate the driving through passive shunting switches. The rotors of the linear motor form the driving modules which provide propulsion and braking force. The active suspension system of the shuttle does without passive dampers in order to avoid the propagation of high-frequency disturbance into the coach body. The forces necessary for the damping are computed by the control and transferred to the body by displacing the spring bases via hydraulic cylinders [9].

The suspension-tilt module is the sub-system which links the active guidance modules and the coach body of the shuttle. A model of the system exists as a MATLAB/Simulink model. The model captures the functionality of one suspension-tilt module. It also contains a model of the coach body, and a model of the hydraulic system. The model controls the body position relative to the guidance modules. The elevation, the lateral position of the body, and its angle relative to the longitudinal axis is controlled. The controller and plant is influenced by the hydraulic pressure, and by disturbing forces. The coach body is to be controlled to provide maximum comfort for the passengers.

4 Concept for Systematic Testing of Mechatronic Systems

Main goal of this new approach to systematic testing of mechatronic systems is to narrow the methodical gap between requirements and testbenches. Existing methods for functional verification and testing of mechatronic systems lack in expressiveness and do not cover all areas of functional verification. Moreover, there is no accepted and standardized test definition language for control systems. An important precondition for a clean verification process is a plan. The concept of a verification plan is not new, however, current concepts and tools for testing mechatronic systems do only support a limited subset of the aspects of a verification plan. A verification plan has to support traceability, visibility, and repeatability. Traceability of the verification plan is provided by links between requirements and verification plan artifacts, namely stimulus definitions, test quality criteria and acceptance criteria. As the purpose of testing is bug hunting, requirement violations are traced from violated acceptance criteria. Test "completeness" is traced from test quality criteria, which describe covered requirements. Visibility of the state of verification is provided by such requirements coverage. Repeatability in model-based development is maintained through a deterministic simulation and test environment.

Similar to the functional requirements document as a design specification, the verification plan assumes the role of the verification specification. This document captures information, which does not belong into the requirements document, but is yet essential for successful implementation of a substantial verification task. While the requirements are being implemented, concurrently the verification plan has to be implemented. The implementation of a verification plan frequently consumes resources in the order of the design and model implementation. Increased efficiency in the process of verification plan generation and execution therefore results in substantial savings in the overall development process.

Figure 3 gives an overview of the new concept for systematic testing of mechatronic systems. The verification plan is based on the requirements and a so-called *principle solution*, which is a first step in requirements formalization [1]. For increased flexibility and precision of stimulus definition, constraints are applied for declarative stimulus definition, supported by a graphical notation similar to the CTM/ES. A new approach to acceptance criteria definition is introduced, which applies a graphical notation similar to CTM/ES. A new CTM/ES-like notation is applied for functional coverage definition.

Stimulus definition with *constraints* allows requirement based *stimulus pattern* definitions, which can be more accurately targeted for improved test quality: The declarative nature of the constraint-based stimulus patterns enables automatic generation of a wide range of stimuli. As more implementation details become available, the declarative stimulus patterns can be adapted in a straightforward manner. The new notation for *acceptance criteria* complements the stimulus pattern definition and it enables automatic acceptance criteria generation together with stimuli generation for fully automatic testbench execution.

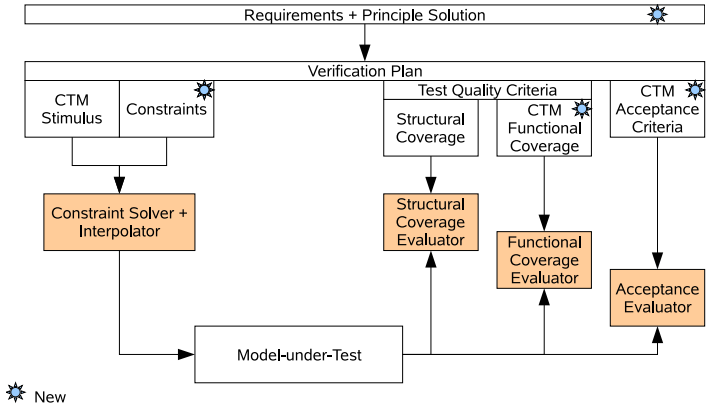

Fig. 3. Verification Plan for Systematic Testing

Moreover, the level of abstraction of the acceptance criteria definition is different from that of the model-under-test. The expensive creation of a reference model at the same level of abstraction can therefore be avoided for automatic acceptance evaluation. *Test quality criteria* define verification goals. These criteria encompass structural coverage metrics, usually. Structural coverage metrics, however, do not enable the derivation of requirements coverage. Recently, in the domain of electronic design, additional test quality criteria have been introduced by means of *functional coverage*. There are no approaches to functional coverage definition for mechatronic systems, which seamlessly fit into a verification process. A new approach to functional coverage definition for mechatronic systems is defined, which relates to requirements and enables requirements coverage derivation.

The definition of a unified CTM/ES notation for stimulus, acceptance criteria, and test quality criteria immediately enables exchange and re-use of information between, e.g., the stimulus and functional coverage aspects of the notation. Functional coverage definition is a tedious process usually, which can be alleviated by the derivation from previous stimulus pattern definitions.

The test control is described by means of the verification languages for execution of the testbench elements. Current industrial approaches to automatic testbench generation and execution are able to extract a subsystem from a simulation model for, e.g., MATLAB/Simulink. The extracted subsystem interfaces are then mapped into a testbench for automatic execution. One example of such a system is AutomationDesk from dSPACE. The automotive testbench features lack in expressivity though in comparison to state-of-the-art "intelligent testbenches" as exist for electronic design [2, 14].

4.1 Enhanced CTM for Constraint-based Stimulus Patterns

The original CTM/ES stimuli generation process is rather monolithic and inflexible. It takes a classification tree, instantiates it by means of certain heuristics,

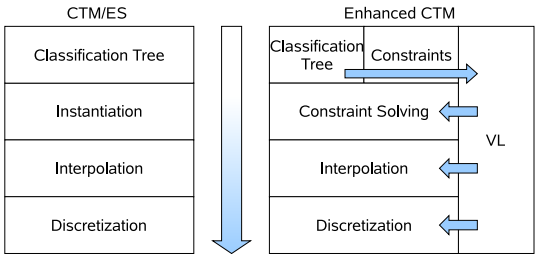

Fig. 4. CTM/ES vs. Enhanced CTM Stimuli Generation Process

interpolates between synchronization points and discretizes the stimulus for test input as illustrated in figure 4. The enhanced CTM provides a new syntax for the constraint based definition of synchronization points, which allows an the definition of constraints between synchronization point timing and value instantiation.

The new and enhanced stimuli generation process is based on a verification language (VL). The classification tree and constraints are mapped to the verification language, which controls the further generation process of constraint solving, interpolation, and discretization. A randomized constraint solver replaces the former instantiation step. Then, interpolation and discretization are performed under control of the verification language. The mapping and integration of the CTM syntax to a verification language allows to use verification language elements such as additional constraints with the classification tree. This provides a higher control over the value instantiation, as it enables the definition of dependencies between classifications and synchronization point times. The enhancements over CTM/ES are summarized in table 1.

	CTM/ES	Enhanced CTM
Randomized instantiation	+	+
Constraint based randomized instantiation	-	+
Synchronization Points with fixed time	+	+
Synchronization Points with constraint based time	-	+
Integration with Verification Language	-	+

+ : supported
- : unsupported

Table 1. Stimulus pattern definition with enhanced CTM

Stimulus Patterns This section illustrates the definition of stimulus patterns. First, the interface definitions are captured by a classification tree. Based on the interface definition, a raw classification tree is partly automatically derived as described for the CTM/ES, where interfaces become combinations, signals become

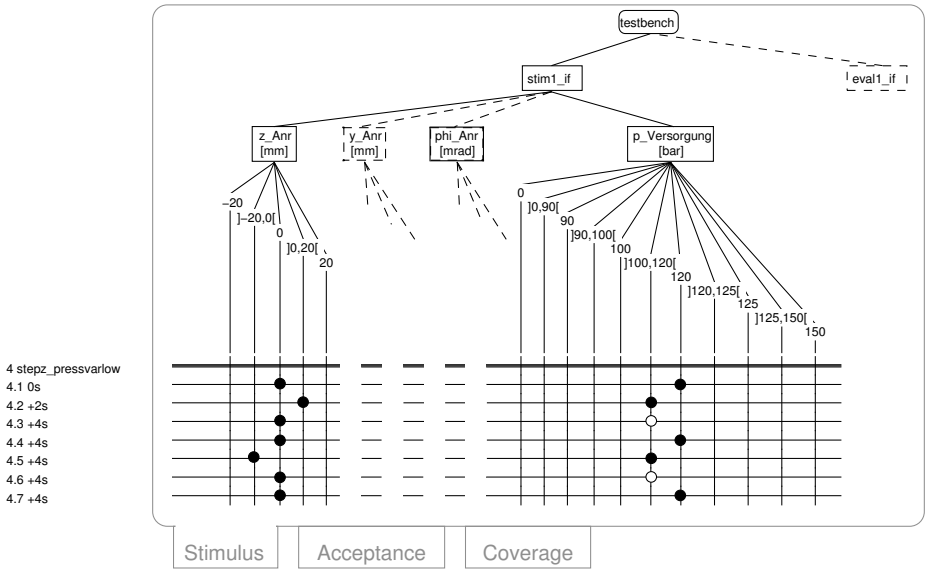

Fig. 5. Classification Tree for Stimulus

classifications and classes are setup heuristically for the signal range described for the interface. The tree is then manually readjusted to the feature descriptions of the interface. The resulting tree is shown in figure 5. It shows four classifications related to the input signals of the interface *stim1_if*. For the three signals *z_Anr*, *y_Anr*, and *phi_Anr* the tree shows the automatically derived classes. They cover the signal range as defined for the interface. Three corner cases are generated by the CTM/ES heuristics: the two corner cases of the range, and the value 0. The fourth classification for the hydraulics pressure, *p_Versorgung*, has been modified according to the requirements. In addition to the two standard corner cases, it defines, e.g., the nominal pressure of 120 bar. The nominal pressure range generates two more corner cases at 100 bar, and at 125 bar. Another corner case is determined by a low pressure emergency shutdown at 90 bar.

Figure 5 shows a classification tree with one stimulus pattern on the stimulus interface *stim1_if*. The test sequence performs step transitions on a single axis (*z_Anr*), while the hydraulic pressure drops with each movement. The test sequences are translated to SystemVerilog constraints as input to a stimulus generator with constraint solver. Multiple test runs are constraint-randomly generated from a single stimulus pattern with different value instantiations. Additional constraints formulated in SystemVerilog can be inserted for increased control over the stimuli generation. This provides a seamless transition from constraint-randomized to directed stimuli generation.

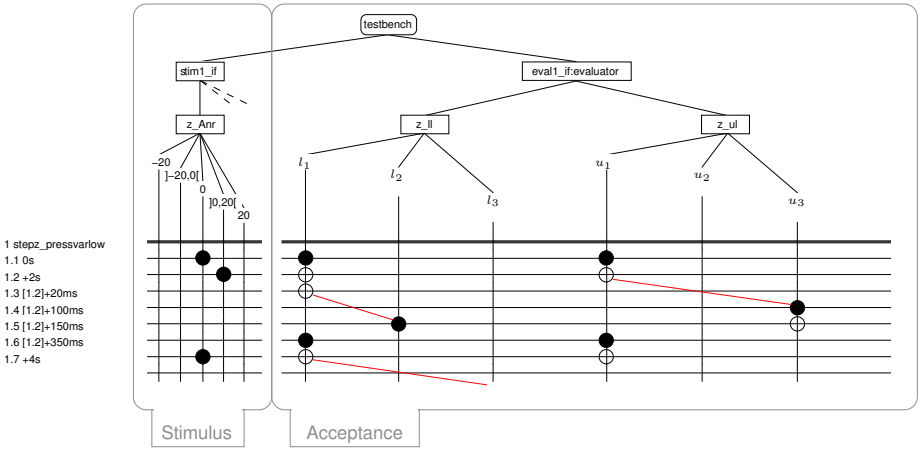

Fig. 6. Classification Tree for Acceptance Criteria

4.2 Enhanced CTM for Acceptance Criteria

Automatic acceptance evaluation is performed by correlation to a reference model or by predicate evaluation [12, 13]. The definition of continuous, interpolated behavioral boundaries is not covered by current assertion languages, therefore a reference model is required for correlation purposes. The new acceptance criteria based on an enhanced CTM notation can provide such a reference model at the same level of abstraction as the stimulus pattern definition with enhanced CTM. This replaces the effort of redundant creation of a reference model at the same abstraction level as the model-under-test. The acceptance criteria do not attempt to capture the exact behavior of the complete model-under-test, they rather set acceptable *behavioral boundaries* for a certain operational range of the model-under-test. From the CTM representation an acceptance evaluator in a verification language like SystemVerilog is generated. Assertions provided by the verification language supplement the acceptance criteria for reporting test results. They provide the link to the testbench evaluation infrastructure of the underlying execution environment.

This section illustrates the definition of acceptance criteria, which fit the stimulus defined in figure 5. The acceptance criteria define a functional relation between stimulus and response. They are defined by an additional classification tree synchronized to the stimulus classification tree for automatic generation of an evaluator (cf. figure 6). The acceptance tree is associated to the response interface *eval1_if* of the testbench. The acceptance aspect of the tree is derived from the interface such that for each signal, a supremum- and an infimum signal is defined as a class. Waveforms generated for these signals enable the formulation of control theoretic system response criteria, such as rise times, transient overshoot, and stabilization time as described in [8, 7].

The acceptance classification tree in figure 6 starts from a root node which represents the *testbench*. On the left hand side, the next lower node represents the stimulus interface *stim1_if*, and, on the right hand side, the next lower node represents the evaluation interface *eval1_if* for the evaluator. The *evaluator* annotation announces the different syntax and semantics of this part of the tree for generation of the evaluator. In the example tree, for the input interface *stim1_if* only the signal *z_Anr* is shown for brevity. Below this node the combination table shows the test sequence *stepz_pressvarlow*. The response signal *aufbauZ* of the interface *eval1_if* is to be evaluated. The signal itself is not present in the classification tree, visually. Instead, the limits for *aufbauZ* are defined below the combination node *eval1_if:evaluator*. A lower limit of *aufbauZ* is defined as classification *z_ll* in the classification tree, and an upper limit is defined as classification *z_ul*.

In classes, functions are specified (l_1, l_2, ...) instead of intervals. They describe the expected functional input-output relation of the model-under-test for a specific operational range. At each acceptance criteria synchronization point, a functional relation is selected, which is then evaluated in relation to the synchronization points of the stimulus. The acceptance criteria represent properties derived from the requirements and from control-theoretic quality criteria, such as transient overshoot, and stabilization time. The definition of a functional relation to certain stimuli definitions enables an exact and automatic evaluation of the system behavior for automatically generated stimuli.

4.3 Enhanced CTM for Functional Coverage

The concept of functional coverage definition [20] has been transferred to classification trees [15, 16] for the definition of functional coverage criteria for mechatronic systems. A classification tree with its value ranges and associated combination table provides the basis for the definition of relevant functional coverage criteria. The concept encompasses the coverage definition for value intervals on specific signals, the cross-coverage of value intervals on several signals, and the (cross-) coverage of transition sequences between the value intervals. The benefit of using classification trees for this purpose is twofold: they alleviate the task of initial formulation of functional coverage criteria and they enable hierarchical reuse of classification tree based stimuli definitions, e.g., from previous test definitions for sub-systems. The operational ranges of mechatronic controls can be captured as test quality criteria, without dependency on a concrete implementation of the system. In short, the CTM stimulus aspect *controls* the operational ranges of a system, whereas the CTM functional coverage aspect *observes* the activated operational ranges of a system for independent test quality evaluation.

The new CTM notation for functional coverage definition builds on the concept of coverpoints employed by the major hardware verification languages in electronic design. Coverpoints are associated to one or more signals and measure the occurrence of several ranges of values, or sequences thereof. Cross coverpoints deal with multiple signals and their value combinations. Figure 7 shows a cross coverpoint definition in classification tree notation as a sub-tree. The

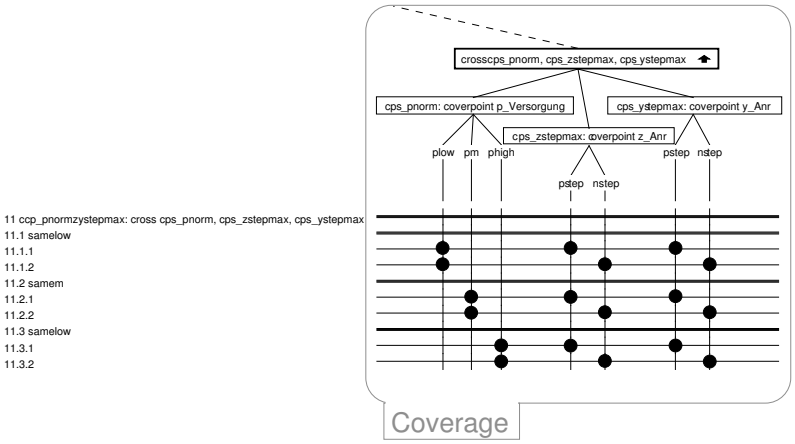

Fig. 7. Classification Tree for Sequence Cross Coverage

top node defines a cross coverpoint over three coverpoints associated to signals *p_Versorgung, y_Anr, z_Anr*. These coverpoints with their value sequences *plow,pm,phigh,pstep,nstep* have been defined elsewhere in the classification tree in the usual CTM notation for sequences. The combination table then describes a coverage of pairwise disturbance of the suspension-tilt platform axes y and z in the same direction. The pairwise movement is also crossed with the three nominal pressure classes. For the cross coverpoint *ccp_pnormzystepmax* in line 11, the coverpoints to be crossed are duplicated as nodes beneath an additional classification tree node (*cross cps_pnorm, cps_zstepmax, cps_ystepmax*) which selects the crossed signals. On line 11.1, the bin *samelow* captures disturbance movement in the same direction for the low nominal pressure range by selecting `<plow, pstep, pstep>`, or `<plow, nstep, nstep>` on the next two lines. The pattern is repeated for the next two bins *samem*, and *samehigh* for the two pressure ranges *pm*, and *phigh*. Similar patterns define functional coverage for opposite movement of the suspension-tilt platform. By means of measurements defined by this functional coverage metric, it can be determined, whether movement of the platform has been stimulated in selected directions with 3 different ranges of hydraulic pressure. Stimulus definitions can be used as a basis for such metric definitions, as the syntax of the combination table remains identical.

5 Conclusion

This article introduced a new methodology and formalism for the systematic verification of embedded control systems. The formalism enables the definition of formal behavioral properties for a model-based functional verification approach. The formalism applies the new Enhanced Classification Tree Method, which was developed based on the established *Classification Tree Method for Embeded*

Systems CTM/ES. A current hardware verification language was applied to definition and control of a verification environment. The new methodology provides improved traceability and visibility for the verification process. It closes the gap between requirements and testbench definition for embedded control systems (i) by support for stimulus patterns capturing requirements, (ii) by support for requirements-based acceptance criteria for automatic acceptance evaluation compatible to the stimulus definition avoiding the creation of a reference model at the same level of abstraction as the model, and (iii) by support for test quality criteria, which relate to specific requirements and enable requirements coverage. Furthermore, horizontal and vertical re-use of test definitions is enabled by means of a unified notation. The method has been implemented in the context of the CRC614, where it was used to verify a mechatronic function module of a railway shuttle system.

Acknowledgement This work has been partly supported by the DFG Sonderforschungsbereich 614 and by the German Ministry for Education and Research (BMBF) through the ITEA2 project TIMMO (01IS07002).

References

[1] P. Adelt, J. Donoth, J. Gausemeier, J. Geisler, S. Henkler, S. Kahl, B. Klöpper, A. Krupp, E. Münch, S. Oberthür, C. Paiz, H. Podlogar, M. Porrmann, R. Radkowski, C. Romaus, A. Schmidt, B. Schulz, H. Vöcking, U. Witkowski, K. Witting, and O. Znamenshchykov. *Selbstoptimierende Systeme des Maschinenbaus – Definitionen, Anwendungen, Konzepte.*, volume Band 234. HNI-Verlagsschriftenreihe, Paderborn, 2008.

[2] Brian Bailey, Grant Martin, and Andrew Piziali. *ESL Design and Verification: A prescription for electronic system-level methodology.* Morgan Kaufmann series in systems on silicon. Elsevier, San Francisco, CA, USA, 2007.

[3] Hamilton B. Carter and Shankar G. Hemmady. *Metric Driven Design Verification.* Springer, 2007.

[4] M. Conrad, H. Dörr, I. Fey, and A. Yap. Model-based Generation and Structured Representation of Test Scenarios. In *Workshop on Software-Embedded Systems Testing (WSEST)*, Gaithersburg, USA, November 1999.

[5] Mirko Conrad. The Classification-Tree Method for Embedded Systems. In *Dagstuhl Seminar Proceedings 04371*, 2005.

[6] Mirko Conrad and Alexander Krupp. An Extension of the Classification-Tree Method for Embedded Systems for the Description of Events. In *Second Workshop on Model Based Testing, MBT 2006*, Vienna, Austria, March 2006.

[7] F. Dörrscheidt and W. Latzel. *Grundlagen der Regelungstechnik.* Leitfaden der Elektrotechnik. B.G. Teubner, Stuttgart, second edition, 1993.

[8] Otto Föllinger. *Nichtlineare Regelungen II.* R. Oldenbourg, Wien, seventh edition, 1993.

[9] J. Geisler. Auslegung und Implementierung der verteilten Aktor- und Aufbauregelung für ein aktiv gefedertes Schienenfahrzeug. Master's thesis, University of Paderborn, Germany, 2006.

[10] K. Grimm. *Systematisches Testen von Software - Eine neue Methode und eine effektive Teststrategie (Systematic Software Testing – A new method and an effective test strategy).* Number 251 in GMD-Report. GMD, Oldenbourg, 1995.

[11] Matthias Grochtmann and Klaus Grimm. Classification Trees for Partition Testing. volume 3(2) of *Software Testing, Verification and Reliability*, pages 63–82, 1993.

[12] J. Grossmann, M. Conrad, I. Fey, A. Krupp, K. Lamberg, and C. Wewetzer. TestML – A Test Exchange Language for Model-based Testing of Embedded Software. In *Automotive Software Workshop '06*, San Diego, March 2006.

[13] J. Grossmann and W. Mueller. A Formal Behavioral Semantics for TestML. In *Proc. of IEEE ISoLA 06, Paphos Cyprus*, pages 453–460, 2006.

[14] ITRS. International technology roadmap for semiconductors 2008 UPDATE. http://www.itrs.net/Links/2008ITRS/Update/2008_Update.pdf, December 2008.

[15] A. Krupp and W. Müller. Classification Trees for Random Test and Functional Coverage. In *Design, Automation and Test in Europe (DATE 2006)*, Munich, Germany, March 2006.

[16] A. Krupp and W. Müller. Systematic Testbench Specification for Constrained Randomized Test and Functional Coverage. In *21st European Conference on Modelling and Simulation ECMS 2007*, Prague, Czech Republic, June 2007.

[17] Klaus Lamberg, Michael Beine, Mario Eschmann, Rainer Otterbach, Mirko Conrad, and Ines Fey. Model-based Testing of Embedded Automotive Software using MTest. *SAE 2004 Transactions, Journal of Passenger Cars - Electronic and Electrical Systems*, 7:132–140, 2005.

[18] E. Lehmann. Time partition testing: A method for testing dynamic functional behaviour. In *Proceedings of TEST2000*, London, UK, May 2000.

[19] T. J. Ostrand and M. J. Balcer. The category-partition method for specifying and generating fuctional tests. *Commun. ACM*, 31(6):676–686, 1988.

[20] Andrew Piziali. *Functional Verification Coverage Measurement and Analysis.* Springer, New York, USA, 2004.

[21] A. Rau. *Model-Based Development of Embedded Automotive Control Systems.* PhD thesis, Dept. of Computer Science, University of Tübingen, Germany, 2002.

[22] Jörg Schäuffele and Thomas Zurawka. *Automotive Software Engineering.* Vieweg, Wiesbaden, third edition, March 2006.

[23] A. Trächtler, E. Münch, and H. Vöcking. Iterative learning and self-optimization techniques for the innovative railcab-system. In *32nd Annual Conference of the IEEE Industrial Electronics Society — IECON'06*, Paris, France, 2006.