Failure Models for Testing Continuous Controllers – Tech Report –

***Abstract*—Continuous controllers exhibit several quality characteristics that are independent of their specific application. These properties include responsiveness, stability, and smoothness. Using the negation of these characteristics together with a natural restriction of the input domain, we first present a failure-based testing methodology, and a respective generic tool, for continuous controllers. We cast existing approaches in our framework, present several new failure models and provide empirical hints at the real-world scalability of this approach. Secondly, we investigate if this approach is useful from an engineer's perspective: Is it effective? Is it efficient? Because it makes use of randomization, is it reproducible? On the grounds of our comprehensive experiments, we find that effectiveness is given in that it provides better worst cases than manual testing; that efficiency is given in that the time required was adequate; and that strong reproducibility is given in any configuration used in our experiments.**

I. Introduction

A good test case detects a potential or likely fault with good cost-effectiveness [1]. One way to derive good test cases in this sense is to use defect models. Defect models encompass the notions of fault and failure models [2]. A fault model captures one relevant class of faults, which has been proposed to be done on the grounds of transforming behavior descriptions [1], [3]. A failure model, possibly induced by a fault model, characterizes those parts of the input domain that are more likely to provoke more costly failures than other parts of the input domain.

When testing continuous control systems, the underlying *faults* are, at first sight, often secondary. Control system engineers tend to use well-established methods to provoke *failures* that come as a violation of specific quality requirements. In this paper, we propose to describe and operationalize failure models to perform failure-based testing of control systems. The idea is to use generic quality criteria for controllers, and then establish realistic constraints on the input domain within which we search for trajectories (i.e. functions over time) that violate the quality requirements based on the criteria.

To arrive at a library of failure models that covers as many standard failures as possibly, we generalize common derivation methods of likely failure-causing trajectories found in literature and practice. We take the quality criteria from the work of Matinnejad et al. [4], [5] and extend them by two criteria gained from a literature survey and the feedback of three control system experts having academical and practical experience.

Once these failure models have been established, we are interested in how useful respective test case generators are. From an engineer's perspective, the test generator should be *effective* (it finds the relevant tests that potentially violate the quality requirements). It should also be *efficient* in that the time to compute test cases for common scenarios happens in the order of minutes rather than hours. Moreover, we would like to understand what the gain in effectiveness is if we add resources to the test case generator. Given that test case generation relies on random testing, they should be *reproducible*: Running the system twice should yield test cases of similar quality.

Problem: In sum, continuous control systems have a broad operating range spawning a large multidimensional input space of their signals. This range makes exhaustive tests infeasible and calls for cost-effective test case derivation: testing controllers is expensive due to their test case duration. We want to derive "good" test cases "efficiently" and be sure that respective test case generators are "reproducible," and that the deployment of resources can rationally be justified.

Solution: By negating domain-independent quality criteria, and by imposing additional practically accepted constraints on the input domain, we present failure models which characterize those parts of a controller's input domain that are likely to violate the quality criteria. We modify the ideas behind existing test case generators to get a generic tool for failure-based testing of continuous controllers.

Contribution: We show (1) that existing work [4], [5] is an instance of our schema, and therefore generalize these results. On the grounds of interviews with domain experts, we (2) present failure models that have not been described in the literature yet. To the best of our knowledge, the combination of (1) and (2) is the first creation of an extensive and comprehensive library of failure models for control systems. Finally, on the grounds of several experiments, we (3) provide evidence that existing work, and specifically our generalization, scales; is effective; efficient; reproducible; and predictable: thus yielding a solution that is a candidate for deployment in industrial practice.

The structure of this paper is as follows: In §II, we give an overview of control systems and testing to motivate rationales for the additional quality criteria and failure models. §III defines the failure models. In §IV, we evaluate the methodology and failure models w.r.t. to reproducibility, effectiveness and efficiency using experiments with nine control systems. §V puts our work in context. Finally, §VI presents a conclusion and an outlook.

II. Background

Control systems are an integral part of our everyday life. They range from steady temperature control in ovens to highly

reactive electronic stabilization programs (ESP) in commercial vehicles and precisely rotating brushless motors. The two essential parts of any control system are a controller on the one side, and a plant or process with a sensor on the other side (see Fig. 1). We consider controllers that are given a single desired value over time ($r(t)$) as reference-variable input signal and are supposed to drive ($u(t)$) the process to this value. In case the process's single control variable output signal or actual value over time ($y(t)$) is fed back to the controller ($\tilde{y}(t)$), the control system is referred to as closed-loop, and open-loop otherwise. A closed-loop control system enables the controller to correct or adapt to the changes detected in the process. In the experiments of this paper, we limit ourselves to closed-loop systems. However, our failure models can also be applied to open-loop control.

For closed-loop control systems, there are two fundamental responses to test: reference-value and disturbance response. Reference-value response is observed if a desired value is given to the controller, and it can drive the process without any environmental influences. Disturbance response is observed if environmental influences ($d(t)$) such as opposing forces, cooling or heating effects are present [6]. Testing each response separately allows control system engineers to make adjustments particularly pertaining to the considered response. In reality and particularly when non-linear, the controller is exposed to both desired value changes over time and environmental influences at the same time, thus leading to a hybrid of both behaviors.

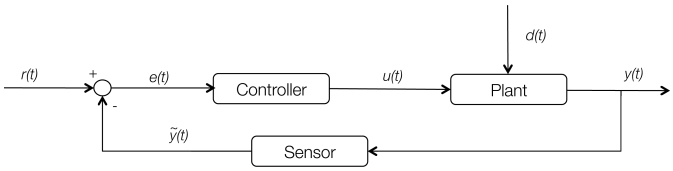

Fig. 1. Closed-loop control system

A. Design and Implementation

Originally implemented in continuous mechanical or electro-mechanical forms, today's controllers are mostly implemented using discrete microprocessors. The advantages are improved versatility and configurability towards a plethora of usage scenarios. In particular, one does not need to build an ESP controller for each vehicle model, but rather for one or multiple vehicle product lines. The controller software for microprocessors is often created using Matlab's Simulink [7] tool. Matlab Simulink uses a block-based data-flow-driven notation for its models, similar to block diagrams well-known by electrical and mechanical engineers. It contains standard blocks for all typical operations required for a controller. Moreover, it allows automatic source code derivation. Thus, controllers and plants can be designed and tested directly within Matlab Simulink. Such tests are referred to as Model-in-the-Loop (MiL) and are used to verify the functionality w.r.t. to the functional response of the controller. Note that, we consider discretized continuous responses, possibly with modes of the control system (i.e. hybrid responses). Using the derived code for testing is called Software-in-the-Loop (SiL) and aims at finding discrepancies between the behavior of the model and code (e.g. run time failures or incorrect data type conversions). On the Hardware-in-the-Loop (HiL) level, hardware behavior such as timing is tested on the actual hardware platform with a simulated environment. Typically, test cases are created on the MiL level and re-used/extended in each subsequent test level.

The process of designing a control system in Matlab Simulink consists of several steps. Firstly, the process characteristics have to be determined and have to be designed on the model level. Subsequently, a suitable type of controller must be chosen. Thirdly, the selected controller must be adapted to the application by adjusting its parameters. At this stage, the controller is implemented as continuous system in both the time and value dimensions. Matlab Simulink allows only the simulation of continuous controllers, but no production-ready code derivation for targeted hardware platforms, which is often done using other tools. To be able to implement it as software, it must be discretized w.r.t. time and subsequently w.r.t. values in a fourth step. In a last step, discrete values may need to be converted to fixed point as some microprocessors lack hardware floating-point units (FPUs) and do not support hardware floating point calculations.

B. Quality Assurance

A classic way of quality assurance for control systems is to examine the transfer function [6]. The transfer function describes the response of a control system to an input signal as mathematical function in a black-box way. Given today's complex controllers including feed forward and cascading controllers, the derivation of the transfer function is infeasible in the time given to test the controller. Thus, control system engineers commonly perform manual knowledge-based testing. Typically, they partition the input space of the desired and disturbance values into equal blocks. In addition, a block is allocated to all boundary values, thus yielding a form of limit testing. One representative from each is picked as test input. To judge requirement conformance, the engineer will inspect the trajectories of the desired, actual and disturbance values and form a knowledge-based verdict of requirements fulfillment.

One automated approach for testing continuous controllers, that our work fundamentally builds upon, was introduced by Matinnejad et al. [4]. It uses a search-based strategy in MiL testing in Matlab Simulink by performing a search space exploration, followed by a refining single-state search. The aim is to find the worst case control system response by driving the system towards a certain goal with the required quality. Their approach aims at finding two distinct desired values such that the controlled process's response violates the quality requirements. To this end, they introduce an extended step function using two desired values. In a first phase, the search space is explored by partitioning this two-dimensional input

space into regions and randomly selecting a number of points within each region as test inputs. This random selection can either be purely random, or adaptively random which tries to avoid clusters of points by maximizing the distance between the points in each region [5]. By using four quality criteria for the control system, parts of the trajectory of actual value in each test case are evaluated. The results are visually presented as a heat map to a control system engineer, who will have to choose regions to further investigate for the worst-case. In the second phase, a subsequent single-state search, an optimization algorithm is used to find the global maximum of an objective function in each selected region yielding (an approximation of) the worst case control system response. This again is implemented using various techniques, including forms of hill climbing and simulated annealing.

C. Quality Criteria

The quality criteria described by Matinnejad et al. [4], [5] are stability, liveness, smoothness and responsiveness. The quality criteria represent abstractions of expected outputs usable by control system engineers as oracles to form a verdict whether the tested control systems conforms to its quality requirements or a failure is present. Since the work of Matinnejad et al. presents an instance of our schema negating these quality criteria, we directly re-use them. In addition, two further criteria required for negation by our newly described failure models exist. The two additional criteria are steadiness [8] and reliability [6]. When presented to three control system experts of our project partner, they concluded the six quality criteria to be representative, adequate and minimal. While this is not representative, it gives us some confidence that we did not miss out on any relevant criteria.

1) Stability: Matinnejad et al. [5] introduce the standard deviation σ of the process's controlled variable as a measurement for stability. σ is measured after T_{stable} and should be $\sigma \approx 0$ for a stable real-world controller. However, there are steadily oscillating control systems (see §II-C5) due to an unpreventable steady oscillation of the process. Such systems will always lead to a standard deviation larger than 0.

2) Liveness: To compute the steady-state error of a control system, liveness [5] measures the maximum difference between desired and actual values after T_{stable}. Maximum liveness is reached when this measure is approximately 0. For a steadily oscillating control system (see §II-C5), the best liveness will be the amplitude of the steady-state oscillation.

3) Smoothness: To measure over- or undershoot, we utilize smoothness [4]. This is because it is particularly useful in our situation, where we use two (differing) intended values of the control variable (the step failure model, see §III-A). Smoothness is measured as the maximum absolute difference between the desired and actual values once the actual value gets as close as v_{s_close} to the desired value. Smoothness should be as low as possible and within a range specified by the requirements.

4) Responsiveness: To measure response time, we use responsiveness [4], which once again is particularly useful in the step model (§III-A). It measures the time it takes the actual value to get as close as v_{r_close} to the desired value. In a control system, responsiveness should be as low as possible and within a range specified by the requirements.

5) Steadiness: Steadily oscillating control systems are common when using multiple controllers in a cascade, relying on imprecise measurements or controlling processes with large internal disturbances. For a system steadily oscillating around the desired value as center of oscillation, the steady-state error provided by the liveness quality criterion provides the amplitude of the steady oscillation. In case the center of oscillation is different from the desired value, Ellis [8] proposes to measure the amplitude of the oscillation. Calculating the maximum deviation of the actual value after some defined T_{stable} is to result in a value twice the amplitude for a stable system. In case this measurement exceeds the expected amplitude, system bounds may have been violated.

6) Reliability: Each control system has a defined operating range. In case the actual value leaves this operating range, there may be physical damages to the system or its surroundings. This has far reaching implications concerning safety or other requirements of the control system. The quality criterion is presented by Goodwin [6]; its Boolean value checks the actual value signal to only contain values within the operating range. Reliability should always be given for any control system. If signals such as the control variable or internal signals of the controller are available for measurement, further Boolean checks can be defined accordingly to perform additional bounds checks.

III. Failure Models

When testing control system, engineers provide desired and/or disturbance values over time to the control system, in order to contrast the system's actual response with some intended response (that is usually left implicit and exists in the minds of the test engineer). Specific choices of the used desired and disturbance signals can provoke failures. Failures in our context are violations of the quality requirements judged by a control system engineer using the quality criteria introduced in §II-C. Since for all criteria higher values mean a possible violation of quality requirements, we aim at finding input values that maximize these quality criteria. A failure is the transgression of a specified upper bound. For technical reasons, the search space for input values needs to be restricted. This is why we add characterizations of those input values that are likely to yield high quality criteria values, and thereby may break the system requirements. Failure models provide these input characterizations together with an assertion when quality requirements are violated. In the following, we present several such failure models and show how to operationalize them for test case generation.

An input signal is a discretized sequence of values given to the control system over time. Formally, let $\tau = \{0, \ldots, T\}$ be a time series and $min_{desired}$ and $max_{desired}$ be the minimum and maximum desired values of the control systems, respectively. An input signal then maps each time step

to a value in the operating range of the control system: $Desired : \tau \rightarrow \left[min_{desired}, \ldots, max_{desired}\right]$ ($r(t)$ in Fig. 1). Correspondingly, the discretized output signal or actual value of the control system can be defined as $Actual : \tau \rightarrow \left[min_{actual}, \ldots, max_{actual}\right]$ ($y(t)$ in Fig. 1). Note that the range of actual values may be larger than the operating range of the control system. This manifests an out-of-bounds failure as described in §II-C6.

In the following, we present failure models for testing control systems. We concentrate on the characterization of the potentially failure causing inputs; the failures themselves always consist of violations of specified quality requirements.

One failure model was directly derived from [4], [5]. The others are the result of a literature [4]–[6], [8], [9] survey and discussions with three of our project partner's control system experts. For each presented failure model, we not only provide the input signals for testing, but also a rationale as to why and when it is sensible to use. Remember that a failure model consists of input values (intended values for the controlled variable in this case); expected output values (dissatisfied quality constraints in this case); and, for the sake of practicality, some explicit constraints on the input domain (e.g., "two intended values for the controlled variables are sufficient"). Every failure model constrains the space of trajectories of the desired value and disturbance signals. The search space exploration is then performed on this constrained typically two-dimensional search space. The surveyed control system experts concurred to specifically use boundary values as they are commonly examined. Thus, the limit testing failure model [1] is combined with each presented failure model, always leading to the creation of additional, non-random, limit test cases.

A. Step

Goodwin et al. [6] and Ellis [8] describe the step response of continuous control systems as the fundamental desired value signal to judge reference-value response. It uses an underlying step function defined as a partial function over discrete time. Matinnejad et al. [4] introduce a two-dimensional step function using *two* desired values. The initial desired value $InitialDesired \in \left[min_{desired}, \ldots, max_{desired}\right]$ is given as the desired value for the first half of the total simulation time T. The final desired value $FinalDesired \in \left[min_{desired}, \ldots, max_{desired}\right]$ is established for the second half. Thus, the desired value signal of this failure model is defined as $Desired(t) = \begin{cases} InitialDesired & \text{if } 0 \leq x \leq \frac{T}{2} \\ FinalDesired & \text{if } \frac{T}{2} < x \leq T \end{cases}$. In a closed-loop control system, it is impossible to start from any other value than the initial value (e.g. 0) due to the feedback loop. Any test case using a single desired value tests only a step originating from the initial value. However, in real-world systems a step from any desired value to any other may occur and should therefore be tested. For instance, an air conditioning system can be set from 19 degrees Celsius to 17 degrees Celsius. Note that if the initial value is 0, only using *one* desired value leads to solely performing tests using a positive step. However, failures are likely occur when performing a negative step according to the experts' feedback. Therefore, the rationale of this failure model is to start at an arbitrary initial value by setting it as $InitialDesired$. All quality criteria are then measured for the actual value signal when $FinalDesired$ is set as desired value. To steer the system into violating its quality criteria requirements, this failure model constrains the trajectories of the desired value signal to all possible two-dimensional steps. The disturbance signal is constrained to a constant 0 as only reference-value response is to be tested.

B. Sine

In a closed-loop continuous control system, a sine wave of $Desired(t)$ may result in a sine wave of $Actual(t)$. If the sine wave of $Actual(t)$ happens to amplify the sine wave of $Desired(t)$, the control system may become unstable. For control system engineers, it is utterly important to verify frequencies of instability to judge whether they are in the operating range. Thus, "frequency responses are a very useful tool for all aspects of analysis, synthesis and design of controllers and filters" [6]. The sine failure model is explicitly designed to show the frequency response for the reference-value response. The signal of the desired value is $Desired(t) = desired + a * \sin(2\pi f t)$, where $desired \in \left[min_{desired}, \ldots, max_{desired}\right]$ is the absolute term, f is the frequency and a the amplitude of the sine wave. The three-dimensionality of this fault model can be reduced to two dimensions as the amplitude only influences the time until the escalation of instability, but not the instability itself. $desired$ is required because signal clipping in boundary regions may have different effects on stability. Thus, $desired$ and f form a two-dimensional search space. To specifically violate the quality criterion of stability and steadiness, this failure model constrains the trajectories of the desired value signal to all possible two-dimensional sine waves. The disturbance signal is again constraint to 0 because only reference-variable response is to be tested.

C. Disturbance

To test the disturbance response introduced in §II, we define a disturbance failure model. The rationale behind using closed-loop control systems is to react to disturbances on the process by having a feedback loop. The goal is to test whether the controller can recover from the disturbance as specified in its requirements. In particular, the over- or undershoot produced by engaging/removing the disturbance as well as the overall stability and liveness are of interest. Using the disturbance failure model, the disturbance is introduced in the feedback loop as $Disturbance : \tau \rightarrow disturbance$ ($d(t)$ in Fig. 1), enabling testing of disturbances in the process and measurement errors at the same time. As we want to find the worst case reaction of a control system to a disturbance, we always use the maximum value $disturbance$ and duration T_{dist} of the disturbance according to the requirements. Variable factors are only the signal's start time t_{dist} and desired value $desired$. Once again, we can reduce the three dimensionality to two dimensions by

fixing the signal to the same trapezoidal ramp, pulse or sine wave for all tests. The desired value signal is $Desired(t) = desired$. The disturbance signal is $Disturbance(t) = \begin{cases} 0 & \text{if } t < t_{dist} \\ disturbance & \text{if } t_{dist} \leq t \leq t_{dist} + T_{dist} \\ 0 & \text{if } t_{dist} + T_{dist} < t \leq T \end{cases}$ for a pulse signal. The other signal types of $Disturbance(t)$ can be defined analogously. In this failure model, all quality criteria are measured after t_{dist} and $t_{dist} + T_{dist}$ to assess the control system behavior during and after the occurrence of the disturbance. As $Desired(t)$ is a constant trajectory and $Disturbance(t)$ is different from zero only after a start time t_{dist} as well as for a fixed duration of T_{dist}, the search space is again two-dimensional. Thus, this failure model aims at steering the system into violating all its quality criteria requirements by constraining the trajectories of the disturbance signal to having the specified form, value and duration. The trajectories of the desired value signal are constrained to all constant trajectories. Note that, the constraining of the desired value tries to separate reference-value and disturbance behavior. However, since the superposition principle does not hold for non-linear systems, a clear distinction may not be possible.

D. Comparison

A failure model derived interviews with control system experts is the comparison failure model. When developing control systems, they are commonly designed using a continuous time and value space and are later discretized in both dimensions. By performing a discretization, controller behavior previously assumed conforming to the requirements may now be violating them. Thus, it is not sufficient to only reuse continuous control system test cases as "one cannot simply treat digital control as if it were exactly the same as continuous control" [6]. For the purpose of comparing the response of two control system, we introduce the comparison failure model. The failure model is an extension of the step failure model where $Desired(t)$ is given to both control systems. Both $Actual(t)$ are assessed using the presented quality criteria. We then compute the difference in quality between the two controllers under comparison, in order to point out differences in response. Thus, this failure model aims at steering the two control systems into producing different responses for the same desired value signal. The constraints are the same as in the step failure model.

E. Allowed Oscillation

A further failure model derived given by the control system experts is the allowed oscillation failure model. Controllers used with steadily oscillating processes often have a deadband requirement. Within this deadband, the controller is supposed to show no reaction to changes in difference of desired or actual value. Typically, the deadband is specified by a minimal percentage $db_{[\%]}$ of change in difference to which the controller is to react. Any changes below this threshold should yield no reaction by the controller. The allowed oscillation failure model tries to provoke a reaction by using a step to just below $db_{[\%]}$ and just above $db_{[\%]}$. Thus, the desired value signal space is constrained into two blocks. One contains all step input signals just inside the deadband for a desired value; and another one contains all step input signals just outside of the deadband. This failure model does not require any quality criteria to be measured, but includes the binary oracle of whether the controller reacted or not. In case the controller reacts to a difference just below $db_{[\%]}$ or does not react to a difference just above $db_{[\%]}$, the verdict of the test case is fail; otherwise pass. $Disturbance(t) = 0$ is the constraint for the disturbance signal.

F. Summary

In sum, our failure models first present reasonable constraints on the input domain (input models for desired values and disturbances as a step, sine, ramp etc. function). Then, they describe what constitutes a failure by either directly relating to quality criteria and respective allowed maximum values, or, in the case of the *comparison* failure model, to the difference in quality between two controllers.

IV. Evaluation

Based on the above failure models, we developed an automated testing tool called ControllerTester for Matlab Simulink, following the ideas and methodology in [4]. Because we add the explicit perspective of failure-based testing, we contribute by generalizing that cited approach (a demo video, open-source version and installer of the tool are available at https://github.com/sac16controllertester/ControllerTester).

We evaluate the tool using a two Intel Xeon E5-2687W v2 processor machine having 16 logical cores clocked at 3.4 Ghz and 128 GB of memory. Matlab is running on Windows 8.1 in version 2014a. Because one controller requires 32-bit, we use a 32-bit version of Matlab in all experiments. The evaluation's goal is to show (1) reproducibility, (2) effectiveness/efficiency and (3) sensitivity of the methodology. Firstly, the search space exploration is non-deterministic as it is based on random selections. Thus, a reproducibility evaluation helps assess whether multiple executions lead to similar results—control system engineers need to be sure that independently of a chosen random seed, the tool is likely to yield similar results. Secondly, we want to better understand the relative and absolute effectiveness of the methodology. In terms of relative effectiveness, we compare the worst cases found by different search strategies. In absolute terms, we want to find out if our tool comes close to, or surpasses, the results independently obtained by three control system experts familiar with the systems. In a third step, we want to check for the sensitivity of the methodology by using different configurations of regions and points per regions. In particular, we want to know whether increasing points or regions in the search space exploration (§II-B) yields better reproducibility and higher relative effectiveness. We do not evaluate the second (optional) step of the search ("single-state search") because its effectiveness has already been thoroughly

No.	Input Range	T[1]	Max. resp. time	Over-/Under-shoot (max)	Steady oscil-lation	Critical fre-quency	Max. Dis-tur-bance
1	0 - 6150 rpm	0.85s	0.1s	5%	no	1 khz	0.05
2	0 - 6150 rpm	0.16s	0.02s	10%	no	-	0.05
3	0.02 - 0.1m	2.3s	2s	5%	no	-	0.05
4	0 - 6150 rpm	0.62s	0.1s	5%	no	1.1 khz	0.05
5	0.02 - 0.1m	2.3s	2s	5%	yes	-	0.05
6	0 - 1 m	1.16s	11s	5%	yes	-	0.05
7	.5 - 4 k rad/s	10.26s	1s	5 %	yes	-	0.2
8	2 - 7 m^3/h	20s	10s	35 %	no	-	-
9	2 - 7 m^3/h	20s	10s	35 %	no	-	-

TABLE I
MODELS OF CONTROL SYSTEMS USED IN OUR EVALUATION
[1] *Total simulation time (T) as described in § III*

examined and distributions for its reproducibility have been shown [5]. Thus, we contribute a thorough evaluation of the first step of the search ("search space exploration").

We focus the evaluation on the step, sine and disturbance failure models. As the comparison and allowed oscillation failure models re-use the step failure model, the results of the step failure model apply to them as well.

Our control systems for evaluation are six control systems used for training (numbered 1-6), one real-world complex control system of a brushless motor (numbered 7), and one real-world control system of a pump (with feed-forward enabled and disabled, numbered 8 and 9) given to us by an industry partner. Each control system has an input range either specified in rpm or rad/sec as the desired rotational speed of a motor, in cm of a pin to drive out, or in m^3/h as the flow rate. Each control system also has different requirements w.r.t. maximum response time (liveness), maximum over-/undershoot (smoothness) and steady oscillation (steadiness). These are described in Table I. The training systems are excerpts of real-world systems in a marginally simplified form to train new engineers. The controller of the brushless motor and the pump are real-world applications and deployed in a commercial context. According to the control systems experts of the industry partner, the training control systems are of medium complexity (avg. 18 Simulink operation blocks per controller) while the brushless motor control system is of maximum complexity (302 Simulink operation blocks for the controller itself) and the pump control system is of medium complexity (143 Simulink operation blocks for the controller itself). Note that an interesting aspect of the pump control system is the usage of characteristic curves making this control system highly non-linear.

A. Reproducibility

To address reproducibility, we question how reproducible the pure random search and adaptive random search are. Our baseline for reproducibility is using 10 regions with 10 points per region performing 30 repetitions of search space exploration using pure random and adaptive random search (remember that § II introduces the idea of searching in a two-dimensional search space). Note that, although statistically 30 repetitions appear to cover the search space of the controller, the signal values use double precision and the search space is not linear. For each point, and then for each region in aggregated form, we compute the standard deviation, the coefficient of variation and the quartile coefficient of dispersion for all quality criteria. These measurements are used to assess the dispersion of the worst cases over all repetitions for each quality criterion and control system. Good reproducibility is witnessed by low values for all three measures. In addition, we measure the percentage of how many times (out of the thirty replications of each simulation) the search space exploration came at least 95% close to the found worst case. This will be referred to as 95% closeness in the sequel. However, 95% closeness is a particularly strong measure and we believe 80% closeness to be sufficient in practice. In the following, we will further examine all cases where the measurements were not 0 and 95%, 90% and 80% closeness was not achieved since otherwise reproducibility is obviously given.

RQ1: Is pure random search reproducible?
For the step failure model, the standard deviation for the training control systems is 0 for all quality criteria except for smoothness in control systems 1 and 4. However, the coefficients of variation are 0.01 and 0.02 and the quartile coefficients of dispersion are 0.01 and 0.01 respectively yielding a marginal decrease in reproducibility. Only 95% closeness of the smoothness of control system 4 is impacted by 3%. As it stays within 90% closeness, the decrease is negligible. In the complex control system, the coefficient of variation and quartile coefficient of dispersion are 0.1 and .07 for liveness as well as 0.08 and 0.07 for steadiness leading to impacted results in 95% closeness. As 80% closeness is almost completely given, we believe this to be sufficient in practice. For all other quality criteria, the standard deviation is 0 in the complex control system and the pump control system.

For the sine failure model, the standard deviation for the training control systems is 0 for all quality criteria except for smoothness in control systems 1 and 4. The respective coefficients of variation are 0.04 and 0.04 and the quartile coefficients of dispersion are 0.03 and 0.02. Again, this yields a marginal decrease in reproducibility. The impact on the smoothness of control system 1 and 4 is 30% of 95% closeness, 10% at 90% closeness and 0% at 80% closeness. In the complex control system, the coefficient of variation and quartile coefficient of dispersion are 0.08 and 0.06 for stability, 0.07 and 0.08 for liveness, 0.03 and 0.01 for smoothness and 0.04 and 0.02 for steadiness. This impacts the 95% and 90% closeness, but yields 80% closeness. For the pump control system, the standard deviation is 0 when feed forward is disabled. When enabled, the coefficient of variation and quartile coefficient of the smoothness are 0.13 and 0.04. Again, 80% closeness is given.

For the disturbance failure model, the result for the standard deviation for the training control systems is identical to the sine failure model. In the complex control system, the coefficient of variation and quartile coefficient of dispersion are 0.07 and 0.06 for stability and 0.16 and 0.12 for liveness.

This impacts the 95% and 90% closeness, but yields 80% closeness.

From the results above, we tentatively conclude test case generation with pure random search to be highly reproducible for the presented control systems. However, we see the smoothness impacted when using feed forward control as "Feed-forward calculates a best guess; it predicts the signal that should be sent" [8]. This impact is not representative and needs to be further investigated.

RQ2: Is adaptive random search reproducible?

For the step failure model, the standard deviation is negligibly small except for the smoothness quality criteria in control systems 1 and 4. However, the coefficients of variation and the quartile coefficients of dispersion are again negligible small. For the complex control system, the coefficient of variation and quartile coefficient again decreases the reproducibility marginally, but 80% closeness is almost completely given.

For the sine failure model, the result for the training control systems is identical to the step failure model. In the complex control system, the coefficient of variation and quartile coefficient of dispersion again decreases the reproducibility marginally, but 80% closeness is completely given. When feed forward is enabled in the pump control system, our measures of reproducibility yield the same values as with pure random search space exploration for the smoothness. Otherwise, the standard deviation is 0.

For the disturbance failure model, the result for the training control systems is identical to the step failure model. In the complex control system, 95% closeness is impacted by 77% on average for liveness, but 80% closeness is given for all quality criteria.

We hence conclude the reproducibility of the worst case in adaptive random search space exploration to be high. We speculate the reason for the decrease of closeness to be the use of discrete time and feed forward for some controller components in control system 4, 7 and 9. However, further investigation is required.

B. Effectiveness

To address the aspect of effectiveness, we question how "good" the worst cases found by pure random and adaptive random search space exploration are. Our baseline for effectiveness is using 10 regions with 10 points per region performing 30 repetitions of pure random and adaptive random search space exploration. Our measure for *relative effectiveness* is the median worst case found for each quality criterion by both methods as it constitutes the average expected worst case. For *absolute effectiveness*, we use test cases derived by three control system experts and compare their worst case with the absolute worst case found for each quality criterion. Good absolute effectiveness is given if we find a better worst case than the manual tests. In addition, finding a better worst case violating the requirements would even demonstrate failure-finding ability of the methodology.

RQ3: What is the relative effectiveness of pure random and adaptive random search space exploration?

Failure model	Stab.	Live.	Smoo.	Resp.	Stea.
Step	0	0	0.7	0	0
Sine	0	0	3.4	0	0
Disturbance	0	0	3.6	0	0

TABLE II
PERCENT IMPROVEMENT/DETERIORATION OF ADAPTIVE RANDOM IN CONTRAST TO PURE RANDOM SEARCH SPACE EXPLORATION WORST CASE FOR CONTROL SYSTEM 1

Failure model	Stab.	Live.	Smoo.	Resp.	Stea.
Step	0	0	1.1	0	0
Sine	0	0	1.3	0	0
Disturbance	0	0	1.6	0	0

TABLE III
PERCENT IMPROVEMENT/DETERIORATION OF ADAPTIVE RANDOM IN CONTRAST TO PURE RANDOM SEARCH SPACE EXPLORATION WORST CASE FOR CONTROL SYSTEM 3

As values were 0 except for the values depicted in Tables II, III and IV, there is no winning strategy in terms of relative effectiveness in our experiments for the step and sine failure model. The best demonstration is the results of the complex control system as a maximum improvement and deterioration of 3-4 % occur at the same time. For the disturbance failure model, the adaptive random strategy is better by a maximum of 7%. We consider this a marginal increase not relevant in practice. Thus, we tentatively conclude both strategies to be equally relatively effective.

RQ4: What is the absolute effectiveness of the methodology compared to human test case derivation?

We analyze the worst case found for each requirement (see table I) and compare them to manual test in cases provided by the control system experts. The manual test cases for control systems 1, 2 and 4 were to try a step of 500/600, 1000/2000 and 2000/6150 rpm and a step from 0 to all values of the previous steps. For control systems 3 and 5, the tests cases involved moving the pin out by 2, 3, 5, 7 and 10 cm. Control system 5 was to move the pin out by 0, 0.5 and 1 m and control system 7 was to use a step from 500, 800, 2000 to 4000 rad/sec. The pump control system was to use a step from 2, 3, 5 to 7 m^3 with disabled and enabled feed forward. The worst case produced by the methodology on average was better than the worst case of the test cases of the control system experts for the training control systems 92.5% of the time. The median improvement by using the methodology was 44% compared to the experts' test cases. The best improvement was the smoothness of control system 4 being improved by 8000%. This worst case also violated the requirements and was on of the violations only detected by our methodology as seen in Table V. The requirements of the brushless motor controller were never violated although the control system experts did only propose 2 of 4 test cases leading to the worst cases. However, since this control system was extensively tested and is used in practice, we could increase the confidence of the control system experts w.r.t. the brushless motor controller functioning according to its requirements. The pump control system's requirement were also not violated, but the worst case was improved by 22% as decreasing the flow led to

Failure model	Stab.	Live.	Smoo.	Resp.	Stea.
Step	3.4	-3.8	-0.4	0	1.2
Sine	-1.8	3.7	0.7	0	2.7
Disturbance	7.1	3.7	2.1	0	7.3

TABLE IV
PERCENT IMPROVEMENT/DETERIORATION OF ADAPTIVE RANDOM IN CONTRAST TO PURE RANDOM SEARCH SPACE EXPLORATION WORST CASE FOR CONTROL SYSTEM 7

No.	Stab.	Live.	Smoo.	Resp.	Stea.
1	-	-	-	ex + ct	-
2	ex + ct	ex + ct	ex + ct	ex + ct	-
3	-	-	ct	ex + ct	-
4	-	-	ct	ex + ct	-
5	-	ex + ct	ct	ex + ct	-
6	ex + ct	ex + ct	ex + ct	ex + ct	ex + ct
7	-	-	-	-	-
8	-	-	-	-	-
9	-	-	-	-	-

TABLE V
VIOLATED CONTROLLER REQUIREMENTS BY EXPERT (EX) / CONTROLLER TESTER (CT) TEST CASES

worse values than increasing. Using the sine failure mode, we could detect instability at 1 and 1.1 kHz for controllers 1 and 4. For all other controllers any instability when using the sine failure model up to a frequency of 10 kHz (given by the control system experts) was negligible. An interesting aspect of the sine failure model was the clear visibility of the increase of instability described as particularly interesting for new control system engineers by the control system experts. For the disturbance failure model and a trapezoidal ramp (given by the control system experts), the worst case was marginally different to the worst cases of the step failure model. This is an indication towards the ability of the control systems to withstand disturbances. Thus, we consider failure models to be as effective or better in comparison to manual testing.

C. Efficiency

RQ5: Is the search space exploration executable in reasonable time?

Concerning efficiency, one training control system's search space exploration using 8 regions and 5 points per regions took approximately 2 minutes, while the baseline took approximately 3 minutes. The largest configuration evaluated was 12 regions and 20 points per region taking 8 minutes per control system. For the complex control system, the computation lasted 10 minutes for 8 regions and 5 points per region, 28 minutes for the baseline and 70 minutes for 12 regions and 20 points per region. The pump control system's search space exploration lasted approximately 3 minutes for 8 regions and 5 points per regions and 6 minutes for the baseline. The time required for each test case was only determined by the time Matlab required for simulation as generation of test cases and measurements take negligible time. Thus, the employed failure model did not influence the overall execution time. Note that, the test case generation and execution is highly parallelizable. When presented with the execution times, the control system experts found the time adequate for practical purposes. Thus, we tentatively conclude to have high efficiency and a linear scalability w.r.t. the overall number of points to be used.

D. Sensitivity

To assess the sensitivity of the methodology, we performed experiments using configurations differing from the 10 regions with 10 points baseline. In terms of the reproducibility of the search space exploration, we want to examine the correlation between the number of points, and number of regions, used during the search space exploration and their caused increase in reproducibility. In terms of the relative effectiveness of each configuration, we are interested in a correlation between the absolute worst case found for a quality criteria in each controller and the number of points and regions used during the search space exploration. The used configurations include varying numbers of regions (2x2, 8x8, 10x10, 12x12) and varying numbers of points per region (5, 10, 15, 20) with 10 repetitions for each configuration.

RQ6: How does the reproducibility change when using different configurations?

The results of our experiments using the step failure model showed a configuration of 8 regions and 5 points per region to be sufficient for control systems 2, 3, 5 and 6 reaching 95% closeness 100% of the time. The same was true for control systems 1 and 4 in all quality criteria except for smoothness. The smoothness of control system 4 yielded the worst case, but almost the same values were measured for control system 1. Again, the decrease of reproducibility was marginal and did completely disappear when using either 12x12 regions or at least 20 points per region. Interestingly, an increase in reproducibility was observed when increasing the number of points per regions as seen in Figure 2, but not when increasing the number of overall points. For the complex control system, the reproducibility increased with the number of points used in 90% closeness in all quality criteria, but is completely given in 80% closeness. For the pump control system without feed forward enabled, 8 regions and 5 points per region are sufficient. With feed forward enabled, smoothness is impacted in 95% and 90% closeness, but is given in 80% closeness. Putting this into perspective, we examined 5 quality criteria for 9 controllers leading to 45 experiments out of which only four had a very small dispersion possibly the result of jitter.

For the sine failure model, a configuration of 8 regions and 5 points per region is sufficient for control systems 2, 3, 5 and 6 as argued above. Again, the decrease of reproducibility of control system 1 and 4 was marginal. Also the reproducibility increased when increasing the number of points per region. The results for the complex and pump control system is the same as with the step failure model.

For the disturbance failure model, a configuration of 8 regions and 5 points per region is insufficient only for the smoothness of systems 1 and 4. However, this insufficiency is resolved when using 10 instead of 5 points per region int the 95% closeness. 80% closeness is always given. For the complex control system, 90% closeness is impacted for the

liveness. It is always given with 80% closeness, which is true for all other criteria as well.

Thus, we tentatively conclude all configurations to yield a high reproducibility. Therefore, a configuration of 8 regions and 5 points per region yields the highest cost-effectiveness due to its short run time. In addition, we speculate it to be better to increase the number of points per region and not the number of regions themselves based on our limited results.

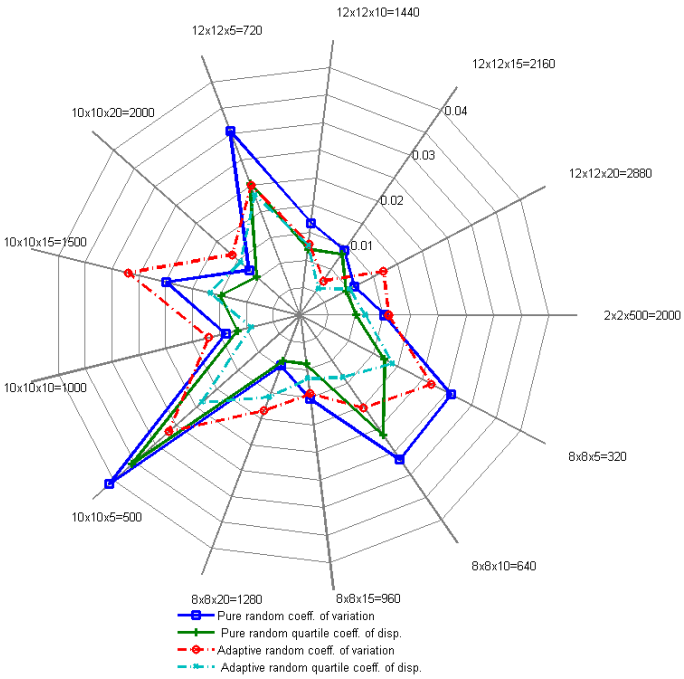

Fig. 2. Reproducibility of smoothness for control system 4 in the step failure model

To further examine whether the number of regions affects the reproducibility in the step failure model, we performed experiments using an overall number of 2000 points with 144 and 4 regions. As depicted in Figure 2, there is only a marginal difference when using this number of points as the maximum value is 0.04 and the minimum is 0. These results also applied to the sine and disturbance failure models. Putting our findings into the perspective of Weyuker and Jeng [10], we intuitively compare random to partition-based testing. Our comparison results give an empirical perspective of the theoretical model yielding (almost) the same results for both as the underlying landscape formed by the quality criteria is unknown. However, using more regions may improve usability for a control system engineer inspecting the heat map produced by the search space exploration.

Examining the complex control system using only 8 regions and 5 points per region in the step failure model, the coefficient of variation was 0.07 and the quartile coefficient of dispersion was 0.07 for stability. Again, these values show a marginal loss of reproducibility. When using 12 regions and 20 points per region (2880 points in total), the coefficient of variation and the quartile coefficient of dispersion had their best values also increasing the 80% closeness for all quality to 100%. Again, theses results also applied to the sine and disturbance failure models. Thus, we tentatively conclude the reproducibility to increase when using a larger number of points. This results in a rule of thumb to rather increase the number of point than the number of regions. However, this increase appears to be non-linear and correlated with the number of regions. Thus, a careful characterization of the search space in the future is required to draw a conclusion.

RQ7: How does the effectiveness change when using different configurations?

As the reproducibility for control systems 2, 3, 5, 6 and 8 was high in the last section in the step, sine and disturbance failure model, the median worst case was constant in all configurations for these control systems. Again, the same was true for control system 1 and 4 except for the smoothness quality criterion. However, as depicted in Figure 3, the median of the worst case of the smoothness of control system 4 in the step failure model stays in a range of 3.5% average and 5% maximum deviation yielding negligible differences. Examining the median worst case found using only 8 regions and 5 points per region for the complex control system, the ratio of median divided by the absolute worst case (median ratio) found already reaches a sufficient 87% for smoothness for compared to the baseline of 89%. When using 12 regions and 20 points per region, the ratio increases to 97% for smoothness being also the best ratio for this configuration. In the pump control system with feed forward, the median ratio is 75% for 8 regions and 5 points per region and 91% for the baseline. We speculate these results to be correlated with the use of feed forward control, but further examination is required.

For the sine and disturbance failure model, the median ratio of the liveness for the complex control system is 77% for 8 regions and 5 points per region 90% for the baseline and 91% for the maximum of 12 regions and 20 points per region. For the pump control system with feed forward and the sine failure model, the median ratio is 82% for 8 regions and 5 points per region and 87% for the baseline. For the maximum of 12 regions and 20 points per region, the median ratio is 97%.

Thus, all evaluated configurations yield a high relative effectiveness of 80% or more when using the baseline configuration. Using fewer regions and points per region influences the 80% closeness in medium and highly complex control system, but is sufficient in low complexity control systems.

E. Summary

In summary, we found pure random and adaptive random search space exploration to be reproducible and effective using the provided control systems. In particular, reproducibility and effectiveness of the search space exploration are essentially the same for both strategies. We see a larger increase when using more points per regions instead of using more regions and speculate there to be a correlation. This hints towards the number of regions to not affect the worst case, but only the readability of the resultant heat map. When examining the test cases given by the control system experts for the

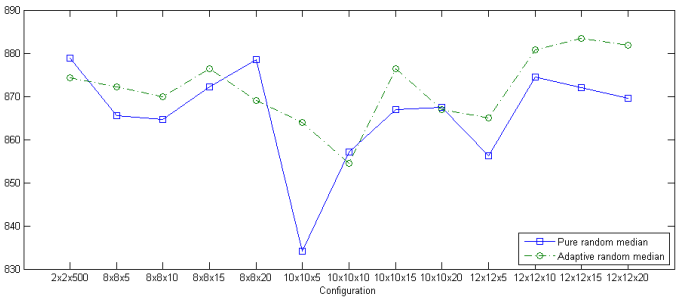

Fig. 3. Relative effectiveness of smoothness for control system 4 in the step failure model

training control systems, the methodology yielded better worst cases 90% of the time with a maximum increase of 8000%. For the complex control system, the methodology provided better worst cases for 50% of the test cases with a maximum increase of 25%. We deem the methodology to be effective and efficient when compared with manual testing. In addition, search space exploration of all control systems was possible within acceptable time constraints using the baseline setting.

V. Related Work

This work is based on the automated testing of continuous controller methodology introduced by Matinnejad et. al. [4], [5]. It aims to generalize and extend the approach by using failure models derived from the violation of quality criteria. A detailed description has already been given in § II as it is necessary to understand the generalization and extension performed. For an overview of testing of continuous control systems see [5]. We see the main difference in our perspective that is based on failure models and that we deem to be more general, as witnessed by more quality criteria (§ II-C) and more input constraints (§ III) that we provide in this paper. The employment of defect models for quality assurance is related to the idea of using defect taxonomies to improve software testing [11] and failure mode and effects analysis (FMEA) [12]. Due to the their nature, these methodologies deliberately focus on making defect knowledge explicit and allow the derivation of test cases specifically targeting these defects. This is similar to mutation testing, where the quality of test suites is assessed by their ability to detect explicit defects after injecting them into the system under test [13].

The notion of defect models [1], [2] including fault and failure models is the basis of this work. To our knowledge, this work constitutes the first operationalization of a failure model and shows its abilities to provoke failures in the area of control systems. Other operationalizations using fault models have recently been shown to be useful [14], but their practicality has not yet been demonstrated.

In the area of MiL control system testing tools, there are Simulink Design Verifier [15], Reactis [16] and TPT [17]. These tools allow the manual specification and automated execution of test cases. Design Verifier and Reactis can generate test cases for coverage. TPT uses a graphical test case notation abstracting from actual I/O in a keyword-driven way. Reactis is the only tool able to generate random test inputs. However, none of the tools use a search-based strategy or failure-based testing. To our knowledge, also no results w.r.t. reproducibility, effectiveness or efficiency have been published.

The effectiveness and reproducibility or predictability of random testing as well as its advantages and drawbacks have been largely discussed by Arcuri and Briand [18], [19]. In addition, they study various types of random testing and present potential practical application scenarios including tailorable evaluation methods. The effectiveness of random testing and particularly adaptive random testing is also discussed [20].

VI. Conclusion

In this paper, we have proposed to derive tests for continuous controllers on the grounds of potential failures. These potential failures are encoded as failure models. Failure models describe violations of various quality requirements together with explicit constraints on the input domain. We have shown that existing work [4], [5] can be cast in this framework, and have provided additional failure models.

As a second step, we have looked into the operationalization of failure models for test case generation. Again building on existing work, we have studied characteristics of a (re-implemented) test case generator [4], [5] for continuous controllers. These characteristics include reproducibility (because test generation in parts is random, results should not be fundamentally different in each run), effectiveness (because the tool should not yield worse results than a human expert), and efficiency (because results should be available quickly).

In our experiments, we found both random and adaptive random search space exploration to yield essentially the same reproducibility and effectiveness. However, this depends on the ability of the search space exploration to capture the landscape produced by the quality criteria. In our experiments, we used several training and two fully fledged real-world complex control system. Thus, we are unable to form a conclusion about the mapping of the landscape in general. However, the worst cases found by the methodology were often better than the manual test case worst cases, yet mostly did not violate the quality requirements.

In the future, we need to characterize the search space using many different configurations. Approaches to visualize large search spaces including respective changes when varying configurations have already been introduced [21] and may be transformable into our context. We were able to provide initial insights concerning the number of regions (seems to be negligible) and points per region (seem to matter more) in our evaluation. However, it was impossible for us to come to a conclusion regarding the effect of increasing either regions or points per region. In addition, we want to research ways of detecting faults in control systems such as implicit data type conversions, wrong integrator modulations, divisions by zero and state variable overflows possibly re-using existing solutions [14]. In addition, the methodology can be extended to control systems with multiple inputs/outputs by extending the number of dimensions of the search space. However,

this paper deliberately focuses on providing an extensive and comprehensive library of failure models for control systems.

Our work is restricted to one controlled variable. Future work includes the generalization to more than one.

Finally, the presented methodology is not limited to MiL testing, but tests can be re-used on other levels such as SiL and HiL. However, when testing at the MiL level, the quality of the test results depends on the accuracy of the plant model.

REFERENCES

[1] , "—," in *Proc. MODELS*.

[2] , "—," in *Dependable Software Systems Engineering*. IOS Press.

[3] L. Morell, "A theory of fault-based testing," *IEEE Transactions on Software Engineering*, 1990.

[4] R. Matinnejad, S. Nejati, L. C. Briand, T. Bruckmann, and C. Poull, "Automated model-in-the-loop testing of continuous controllers using search," in *SSBSE 2013*, 2013, pp. 141–157.

[5] R. Matinnejad, S. Nejati, L. Briand, T. Bruckmann, and C. Poull, "Search-based automated testing of continuous controllers: Framework, tool support, and case studies," *To be published in IST*, 2014.

[6] G. Goodwin, S. Graebe, and M. Salgado, *Control System Design*, 2001.

[7] MATLAB, *version 7.11.1.866 (R2010b SP1)*. MathWorks, 2011.

[8] G. Ellis, *Control System Design Guide*. Academic Press, 2004.

[9] N. S. Nise, *Control System Engineering*. John Wiley & Sons, 2007.

[10] E. Weyuker and B. Jeng, "Analyzing partition testing strategies," *IEEE Trans. Software Eng.*, vol. 17, no. 7, pp. 703 –711, jul 1991.

[11] M. Felderer and A. Beer, "Using defect taxonomies for testing requirements," *IEEE Software*, 2014.

[12] N. Ozarin and M. Siracusa, "A process for failure modes and effects analysis of computer software," in *Reliability and Maintainability Symposium*, 2003.

[13] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Trans SW Eng*, 2011.

[14] , "—," in *ASE*.

[15] MATLAB, *Simulink Design Verifier*. MathWorks, 2014.

[16] Reactis, *Reactis Product Suite*. Reactive Systems, 2014.

[17] TPT, *TPT - Time Partition Testing*. Piketec Gmbh, 2014.

[18] A. Arcuri, M. Z. Z. Iqbal, and L. C. Briand, "Random testing: Theoretical results and practical implications," *IEEE Trans. Software Eng.*, vol. 38, no. 2, pp. 258–277, 2012.

[19] A. Arcuri and L. C. Briand, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Softw. Test., Verif. Reliab.*, vol. 24, no. 3, pp. 219–250, 2014.

[20] ——, "Adaptive random testing: an illusion of effectiveness?" in *Proceedings of the 20th International Symposium on Software Testing and Analysis, ISSTA 2011, Toronto, ON, Canada, July 17-21, 2011*, 2011, pp. 265–275.

[21] B. Haugen and J. Kurzak, "Search space pruning constraints visualization," in *Second IEEE Working Conference on Software Visualization*, 2014.