Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Agents Learn from Human Experts: An Approach to Test Reconfigurable Systems

Asem Eltaher[1] (PhD Student), Markus Maurer[1], Thomas Form[2], and Mohamed Ayeb[3]

[1] Technische Universität Braunschweig, 38106 Braunschweig, Germany
[2] Volkswagen AG (VW), 38436 Wolfsburg, Germany
[3]Universität Kassel, 34121 Kassel, Germany

{eltaher, maurer}@ifr.ing.tu-bs.de, thomas.form@volkswagen.de, ayeb@uni-kassel.de

Abstract— Faulty software is costly and possibly life threatening as software products permeate our daily life. Therefore, the test process formulates an indispensable component of the development cycle; yet it is a formidable task. In an effort to alleviate its challenges, this contribution outlines a novel approach to enrich traditional test techniques with intuition-based test strategies learned by observing skilled human testers during various test sessions. Consequently, the strategies learned would be verified, combined, and generalized to be further applied in similar test situations. Hence, a reasonable portion of the workload done by human testers would be shifted to the test system itself. This leads to a significant reduction in the development time and cost; yet the test efficiency is not sacrificed.

***Keywords*— learning, reasoning, embeded testing**

I. INTRODUCTION

Automotive domain involves a continuous increase in the quality expectations of the delivered products. Concurrently, business pressures demand a significant reduction in the development time and cost whereas ensuring more robust products. Therefore, the test process is considered as an indispensable component of the development cycle; yet it is an arduous task especially for embedded reconfigurable systems, e.g. infotainment systems, driver assistant systems, etc.

Generally, a reconfigurable Device-Under-Test (DUT) is a component-based system that involves the possibility to replace one or more of its component(s) [1]. This consequently permits slightly modified configurations of the same DUT, which make the corresponding test process an expensive burden in two diverse aspects. First, given the new delivered configuration, what has to be (re)tested? Second, assuming the inevitability of faults, when should the testing be stopped?

One solution is the automatic generation of test cases from the DUT's specifications according to a definite coverage criterion, which is shown in [2] to positively augment the test process. Nevertheless, this approach assumes the availability of formal and error-free specifications, which is –in case of reconfigurable DUTs- an overambitious assumption as one or more component(s) may be supplied by a third party.

Related work in [3] generates test suites by combining statistical approaches and redundancy techniques. Initial experiments show auspicious results, however, an empirical study in [4] shows that a substitution of a single component may entail -in some cases- to (re)execute the whole test cases.

Recently, diverse contributions -e.g. [5, 6] - tend to enrich traditional test techniques via imitating human intelligence during a test session. Specifically, the idea is to interview skilled human testers to get an insight into their experiences, which formulate the training sets for a learning system.

Consequently, a Bayesian Networks (BNs) model is trained to direct the test cases to the most likely defect areas [6]. The results illustrated show remarkable contributions, but this approach formulates a tough duty on the test experts since "it requires normally the analysis of a large number of cases, covering almost every possible combination of input variables" [6]. Furthermore, the theory of fuzzy logic shows that humans often describe their experiences in very imprecise and vague language that can hardly be formally described [7].

One other related approach is detailed in [8], in which skilled human testers are subsequently involved in the test process to adapt the predefined test strategies according to the test results obtained. Research achieved in [8] resulted in a corresponding cost model that can be formulated as:

$$C(T) = C_0(T) + C_1(1+P)m(T) + C_2\left[m(T_{LC}) - (1+P)m(T)\right] + C_3\int_0^T w_k(t)dt$$

Such that:
C(T): Overall cost function in the testing period (T).
C_0(T): Human engagement cost.
C_1: Cost of correcting an error during testing.
P: Additional fractional of detected faults.
m(T): Mean value function.
C_2: Cost of correcting an error during operation.
T_{LC}: Software life-cycle length.
C_3: Cost of testing per unit testing.
w_k(t): Current testing effort estimated by a logistic testing-effort function.

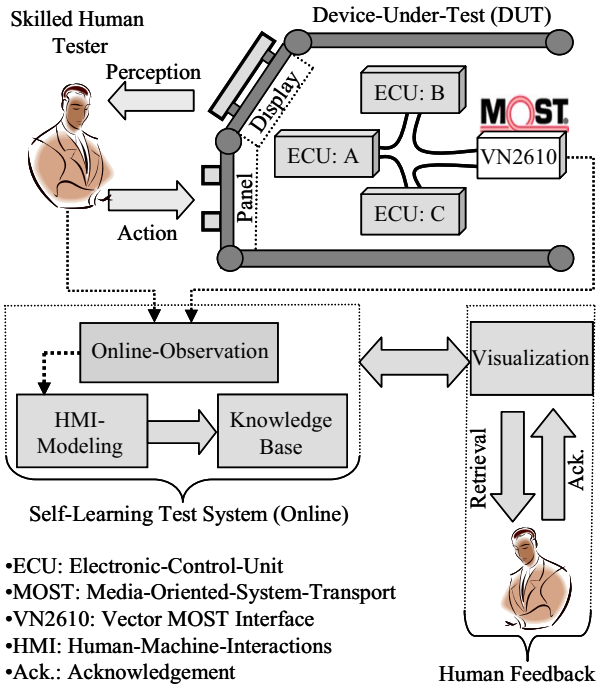

Fig. 1: Learning by observing skilled human testers.

Indeed, the periodical engagement of skilled human testers leads to an increase in the cost factor $C_0(T)$ that makes the process of involving human experts an expensive contribution. In response to this challenge, this contribution aims to develop a test system that can learn intuition-based test strategies by observing skilled human testers interacting directly with various releases of the DUT.

Consequently, strategies learned are verified, aggregated, and then generalized to be applied in any similar situations. This offers a constructive framework to -not completely but at least partially- reduce the necessity of the periodical engagement of skilled human testers. This leads to a non trivial reduction in the overall time and cost; yet the benefits of involving human experts in the test process are not sacrificed.

The realization of the proposed framework is outlined in the following sections. Section (II) illustrates the learning environment whereas Section (III) details our approach to learn behavioral primitives of the DUT. Then, Section (IV) describes the realization technique and the results achieved so far. Section (V) addresses the theory of generalization associated with a case study. Finally, Section (VI) summarizes the idea with an outlook about the future work.

II. Learning Test Systems

A. Observation of Human-Machine-Interactions (HMI)

Intuitively, facing the fact that "it is practically impossible to fully test a product" [9], skilled human testers posses the potential to work with an intelligent test strategy to design and adapt a significant range of test inputs under which a failure may arise. Then, they proceed further till they reach a decision that additional testing does not change the test results, which is defined in [10] as "good enough testing". Motivated by this assumption, the idea is to observe HMI take place during test sessions aiming to capture how skilled human testers behave in complex strategic situations.

The core conception is that HMI are based on a perception-action concept (see Fig. 1). In the perception phase, a human tester perceives the information obtained from the DUT that is followed by an action phase, in which the next best action(s) has/have to be decided to reveal any hidden failures.

Practically, observing HMI is accomplished by recording the communication data sent over the Media-Oriented-System-Transport (MOST) data bus. To this goal, a MOST-based interface, named VN2610 in Fig. 1, is attached to the DUT. Then, a preliminary step before storing the data recorded is to model it, which is illustrated in Subsection (B).

B. Modeling of Human-Machine-Interactions (HMI)

Situation-Operator-Model (SOM) in [11] models the changes of the considered part of the external environment as a sequence of effects. These effects are described by the items *scenes* and *actions*. A real world scene is modeled by a *situation* whereas an action is modeled by an *operator* [11].

The item situation (S) models the observed state of the DUT, which consists of a set of characteristics (C) and relations (r). Each characteristic describes a part of the DUT's state and possess a time-dependent parameter (P), which describes the current observed state, e.g. CD-Status(In/Out). A relation (r_i) describes the inner connection(s) between different characteristics of the same situation, if they exist.

The item Operator (O), defined by its name and parameter, is used to model the actions invoked by human testers that drive the situation of the DUT from an initial situation (S_I) to a final one (S_F), e.g. Set-Button (Button-ID).

In Fig. 2, a situation changed by an operator that leads to another situation is shown, which is denoted in [11] as an e*xperience* E^{SI}_{SF}. Then, the current final situation is defined as the initial one for the next experience, and so on. I.e. SOM offers a flexible data structure to model HMI, represented in the observed test cases, as a sequence of experiences.

Within the scope of this paper, SOM is detailed further through the following definitions:

Definition 1: Two situations are *equal*, if they posses the same characteristics with equal corresponding parameters.

Definition 2: Two operators are *equal*, if they have the same name and parameter.

Definition 3: Two operators are *separate*, if they affect different characteristics of the same situation when these operators are concurrently invoked by the human tester.

Definition 4: Two operators are *correlated*, if they trigger the same characteristic(s) within the same situation in case these operators are simultaneously activated.

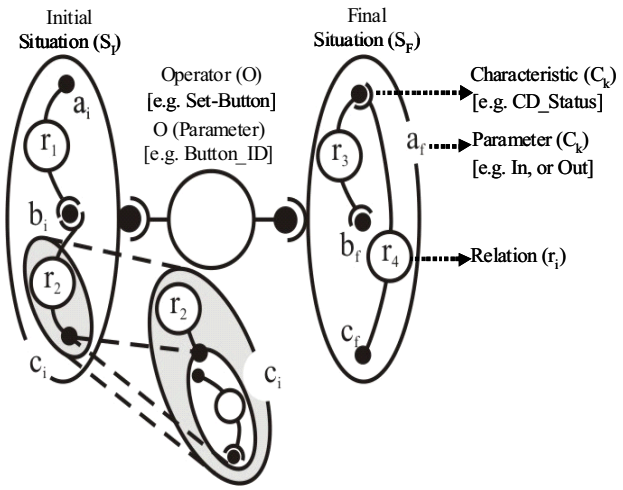

Fig. 2: Structure of Situation-Operator-Model (SOM) [11].

Definition 5: Two experiences are *similar*, if the initial situations are equal, operator(s) of the first experience is/are a subset of the second one, and the final situations do not match. I.e. $S_{I1}=S_{I2}$, $O_1 \subset O_2$, and $S_{F1} \neq S_{F2}$.

Definition 6: Two experiences are *homogeneous*, if they posses equal initial situations, their operators share the same name but with different parameters, and the final situations are not equal. I.e. $S_{I1}=S_{I2,}$ O_1(name) = O_2 (name), O_1(Parameter) ≠ O_2 (Parameter), and $S_{F1} \neq S_{F2}$.

C. Knowledge Base and Human Feedback

Observing skilled human testers is potentially effective provided that the learning environment conveys a flexible data structure that can be optimally exploited. This is especially true, if it is needed to retrieve similar experiences for further learning purposes or even to judge the behavior of the DUT. Additionally, a mean of communication between the test system and human testers must be provided to enrich the learning process with human feedback.

The objective of this module is to enable human testers to: a) acknowledge the consistency of the stored experiences and b) alter the stored data. Case (a) is triggered, if the test system announced the existence of inconsistent experiences in its knowledge base. This would be the case, if it observed the same test case with two different results (i.e. $S_{I1}=S_{I2}$, $O_1 = O_2$, and $S_{F1} \neq S_{F2}$), which may occur due to a human error.

On the other side, altering the knowledge base is demanded, if the DUT introduced wrong reactions during test sessions. Derived by these cases, the test system has been enriched with a 2D visualization module to facilitate the feedback process. Then, the provided acknowledgment is saved in the knowledge base, i.e. learned, for similar future cases.

A complete description of HMI-Modeling using SOM and the learning environment is given in [12], in which it is extended in this contribution to overcome the boundary from individual to cooperative learning as shown in Subsection D.

D. Cooperative Learning

Intuitively, the motivation behind cooperative learning is twofold. First, it is not practically possible to build a faithful model of HMI given an individual training session, which may suffer from any cognitive bias found in human testers. Second, it is not feasibly possible for a single individual to comprehend the entire scope of possible interactions and, therefore, some entire segments of the test process may be overlooked. Given this premise, cooperative learning is necessary to guard the learning process against overfitting or underfitting and to maximize individual results.

As a response, the idea is to aggregate several test strategies from skilled human testers interacted with various releases of the DUT. And to this goal, the notion of task relatedness has to be defined, which is the common test case(s) achieved by human testers during the learning phase.

Practically, the existence of common test case(s) among diverse test strategies is not a naive assumption. It has been shown in [13] that test scenarios performed by skilled human testers, regardless their test strategy; share some test cases that aim to stimulate the basic functions of the DUT.

Consequently, a coverage criterion has to be defined prior to the test process initialization. A conventional approach is to build a reachability graph to show all the transitions and configurations that are reachable from a defined initial state. Then, a commonly used test coverage criterion is to test each edge in the reachability graph at least once [14].

Finally, a preliminary step on the way to generalize the strategy learned is to define behavioral primitives of the DUT and their correlations to the actions invoked by the human testers during the learning phase.

III. Learning Behavioral Primitives

A. Idea

The necessity to learn behavioral primitives arises from the ability of the test system to differentiate the test cases into simple (deterministic) and compound (non deterministic) test cases. In deterministic cases, the action-reaction relationship is governed by a one-to-one correspondence whereas non deterministic test cases lead to a many-to-many relationship.

In the later case, the objective is to identify, which operator (O) has triggered which characteristic (C). To this end, supervised clustering using Rule-Based- Reasoning (RBR) is adopted in case of simple test cases. Whereas Case-Based-Reasoning (CBR) is used in case of compound test cases.

B. Supervised Clustering using Rule-Based Reasoning

RBR is to reason using a prior knowledge [15], which invokes in our case three algorithms. The first one compares each characteristic's parameter of S_I with its corresponding one of S_F. If they are equal, the system compares the remaining characteristics. Otherwise, it associates the operator to the triggered characteristic, i.e. the characteristic with the changed parameter.

Then, the defined association rule is learned to be recalled in case of future similar cases. The second algorithm takes place, if all the characteristics' parameters are equal. Then, a rule is learned that this operator (O) has no effect on this initial situation (S_I). Finally, the third algorithm activates the CBR component in case of compound test cases.

C. *Supervised Clustering using Case-Based Reasoning*

CBR is based on an analogical transfer approach, in which a new problem is solved by finding past similar cases and reusing them in the new domain [15]. The idea to use CBR in learning behavioral primitives of the DUT is initially proposed in [16]. In this contribution, the theory is extended, modeled, and applied to the framework obtained in [12].

Briefly, solving the problem of learning behavioral primitives using CBR can be formulated as following:

Definition 7: Given a compound test case (T) with an initial situation (S_I), find -with the aid of the knowledge base (K_B)- a finite set of association rules $\{r_1, r_2, \ldots r_n\}$, such that:

$$r_i : \left(C_i[P_i] \rightarrow C_i[P_k] \;\middle|\; O = O_i\right) \;\forall\; T[S_I] \subset K_B$$

Based on the theory of cause and effect, Definition 7 shows an association rule algorithm that assigns the operator (O_i) the responsibility to shift the parameter of the characteristic (C_i) from the value (P_i) to (P_k). Through applying this theory, the realized solution consists of three successive steps:

a) Given an observed-compound experience $E_{(OB)}$, the test system retrieves a similar simple one $E_{(KB)}$ (see Def. 5).

b) Then, the association rule introduced by $E_{(KB)}$ would be integrated into the current observed one $E_{(OB)}$.

c) Finally, the new rule learned would be verified using a Rule-Confident-Component (RCC).

For example, given the compound experience in Fig. 3a, the test system retrieves a *similar simple* experience $E_{(KB)}$ as in Fig. 3b. Then, a matching algorithm using the early described RBR technique is performed, in which $E_{(KB)}$ provides an association rule (e.g. r_1) such that:

$$r_1 : \left(C_1[a] \rightarrow C_1[a_k] \;\middle|\; O = O_1\right) \;\forall\; T[S_{I(KB)}] \subset K_B$$

Consequently, a new complementary rule (e.g. r_2) would be *initially assumed,* in which the other operator (O_2) is assigned the responsibility to shift the parameter of the characteristic (C_3) from the value (c) to (c_k) such that:

$$r_2 : \left(C_3[c] \rightarrow C_3[c_k] \;\middle|\; O = O_2\right) \;\forall\; T[S_{I(KB)}] \subset K_B$$

Intuitively, assuming the validity of the rule (r_2) would be a dubious assumption, if it is not statistically verified. Hence, the idea proposed in [16] is enriched in this contribution by a statistical-based approach to verify the consistency of the new learned rule (r_2) makes the results obtained more reliable.

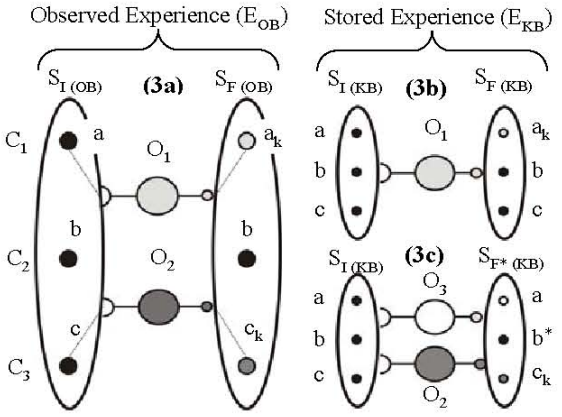

Fig. 3. Learning behavioral primitives of the Device-Under-Test.

Briefly, the test system retrieves similar test cases that share the same initial situation $S_{I(KB)}$ and the operator (O_2) as shown in Fig. 3c. Then, the cases retrieved would be fed to the RCC.

RCC is defined as the probability of success (P_s) to reveal the consistency of the new learned rule (r_2) with respect to the whole knowledge base. Referring to the example in Fig. 3, RCC would be:

$$P_s : \left(C_3[c] \rightarrow C_3[c_k] \;\middle|\; O = O_2\right) \;\forall\; T[S_{I(KB)}] \subset K_B$$

On one side, reaching a probability of 100%, the test system would consider both operators (i.e. O_1 and O_2) as *separate* operators (see Def. 3). And, therefore, a new *compound* association rule (R) is learned with an inner structure of both individual rules (r_1) and (r_2) such that:

$$\forall \;\; T[S_I] = S_{I(KB)} \subset K_B \Rightarrow R \equiv (r_1 \;\wedge\; r_2)$$

On the other side, reaching a probability of less than 100%, the test system would consider both operators as *correlated* operators (see Def. 4). And, therefore, a new compound rule will be learned with a *black box* inner structure.

Based on the above theory, the test system is capable to define the functional dependencies and correlations between human's actions (O_1 and O_2) and the corresponding DUT's reactions (C_1 and C_3). Hence, one significant benefit of the proposed test system is its ability to learn the specifications of the DUT even with incomplete knowledge about its internal structure. This offers a significant contribution over other approaches surveyed in [17], which assume a complete knowledge about the internal structure of the DUT.

IV. REALIZATION AND RESULTS

The early presented learning paradigm associated with the reasoning component has been experimentally verified. Briefly, the realization of the online-observation module as

well as SOM is done using *Microsoft C++*. The knowledge base module is realized in *Python* due to its strong linkage to free object-oriented databases like *DyBASE* used to realize the knowledge base. Additionally, the visualization module is realized using the visual libraries of Python (*VPython*).

The experimental testbed is an infotainment DUT, which consists of three Electronic-Control-Units (ECUs) connected through a MOST data bus. Then, several operators are defined to model the possible actions that might be invoked by human testers. Additionally, a situation is defined to include a vector of significant characteristics of the DUT, which have to be observed during the learning phase. The complete library of the operators and characteristics is detailed in [18].

For demonstration purposes, two individual test scenarios are conducted as shown in Fig. 4a. Then, the migration from individual to cooperative learning took place with the results shown in Fig. 4b.

Following the early mentioned coverage criterion in [14], the test system comes out with four diverse test scenarios:

1. E^0_5, E^5_3, E^3_4, and E^4_6.
2. E^0_1, E^1_3, E^3_4, and E^4_7.
3. E^0_5, E^5_3, E^3_4, and E^4_7.
4. E^0_1, E^1_3, E^3_4, and E^4_6.

Obviously, the first two test scenarios are just a replica of the scenarios learned from the human testers A and B respectively. However, the third and fourth scenarios formulate new test suites, since the chain in the third scenario (E^3_4-E^4_7) is tested under different initial conditions (E^0_5-E^5_3) rather than the conditions learned from the human tester B (E^0_1-E^1_3). Similarly, the chain in the fourth scenario (E^3_4-E^4_6) is tested under varied initial conditions (E^0_1-E^1_3) rather than the conditions learned from the human tester A (E^0_5-E^5_3).

Hence, the developed framework is capable of overcoming the limit from just imitating human behavior to optimize it, which definitely improves the coverage criterion of the test process through the *autonomous* generation of new test suites.

Consequently, the reasoning paradigm is activated to learn the behavioral primitives of the DUT. Interestingly, the results achieved match exactly the initial results described in [12] with even more confident in the approach applicability since the test system has been enriched, in this paper, with the early described RCC.

V. LEARNING VERSUS TESTING

Actually, the learning paradigm aims to provide an intuition-based oracle to test either DUTs that are similar to the one used in the learning session or the same DUT, but with a slightly modified configuration. Therefore, one more enhancement to the framework, introduced in [12], is how what is learned during the initial configuration can be analogically transferred to similar configurations.

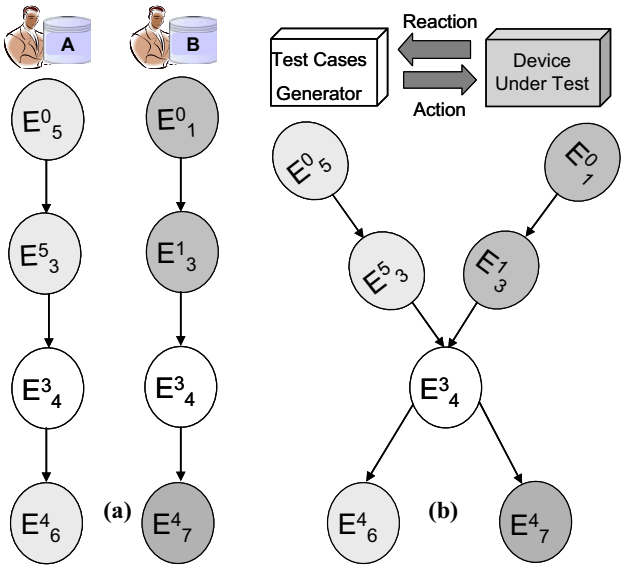

Fig. 4. Individual versus cooperative learning.

To this goal, generalization techniques like Neural Networks (NNs), Support Vector Machines (SVMs), and Rule Extraction (RE) are surveyed. Though the remarkable contributions of NNs and SVMs, one of the significant resistances against these techniques is their lack of interpretability. Specifically, it is difficult for a human analyst to understand the reasoning behind these models' decisions [19].

Conversely, generalization with logical rules is more acceptable to human analysts due to the comprehensibility of such an approach and the possibility to be validated [20]. Motivated by this fact, generalization using rule extraction is adopted and the theory can be summarized as following:

a) Classification: The domain knowledge is classified into sub regions of homogeneous experiences (see Def. 6).

b) Mapping framework: Stored data is described in terms of the Man-Machine-Interface (MMI) using linguistic variables.

c) Rule-based representation: Body of the learning module is established and trained by a set of domain examples followed by the RE phase.

A. Classification

The objective of this module is to classify the stream of test cases into sub regions of homogeneous experiences. And, therefore, it would be possible to obtain different rule sets that govern the DUT's behavior in various environmental conditions described by the initial situation (S_I) of the DUT.

A typical example would be gathering test cases that share the same initial situation, in which human testers tried to increase/decrease the volume status to various levels. I.e. these test cases share the same initial situation, the same operator, but the operator's parameter varies and the final situation as well. And the idea is to find a logical rule that faithfully describe a generalized behavior of the DUT, given this initial situation and this operator, based on the observed samples.

B. *Mapping Framework: Abstract Description*

Intuitively, it is not possible to learn without prior commitments over how the strategies learned can be encoded. Feature Space Mapping (FSM) in [21] is used to define features that identify the problem domain.

Practically, component-based systems entail -on one side- reconfigurable structure, which is the internal ECUs. On the other side, MMI formulates a settled part that is not frequently changed. And the idea behind adopting FSM is the usage of linguistic variables to encode the strategies learned in terms of the fixed part of the DUT rather than its reconfigurable one. Hence, the rules learned can be (re)applied; even if the DUT's internal structure is slightly modified.

Typically, linguistic variables associated with an MMI for an infotainment DUT would, on one side, describe the actions invoked by human testers. One the other side, linguistic variables have to describe the reaction(s) introduced by the DUT. The used variables are partially shown in Table 1.

C. *Rule Extraction (RE)*

An initial step towards RE is to differentiate diverse types of states observed during the learning session. Practically, observing an infotainment DUT involves observing three diverse arts of states [13]:

- Continuous states: e.g. volume level.
- Discrete states: e.g. CD track status (1, 2…n).
- Logical states: e.g. loud speaker (on/off).

Generally, a logical rule that describes the HMI take place during a test session would be:

For all S_I=S_x:

IF (n (index) = n_x ∧ d(index)=d_x ∧ P(index)=P_x)
THEN (ΔY=Y_x))

Table 1: Linguistic variables for the mapping framework .

MMI LINGUISTIC VARIABLE		DESCRIPTION	POSSIBLE PARAMETERS
DUT Reactions	V_i/V_f	Initial/Final state of the volume signal.	Decimal value.
	T_i/T_f	Initial/Final state of the CD-track.	Numerical value.
	L_i/L_f	Initial/Final state of the speaker (on-off).	Binary value.
Human Tester Actions	n (index)	It indicates the number of turns of the button identified by index.	A numerical value ranges from 0 to n.
	d (index)	It indicates the rotational direction of the button identified by index.	1: Clockwise -1: Anti-Clockwise 0: No rotation
	P (index)	It indicates the push state of the button identified by index.	1: Pushed 0: Released

Table 2: Rule Extraction: Structure of the information table.

Table 2a): Information table of continuous functions, e.g. Volume.

IP: Input OP: Output ΔV= $(V_f-V_i)/V_i$			COMMUNICATION DATA			LINGUISTIC DESCRIPTION			
			IP		OP	IP			OP
			V_i	O_1	V_f	n	d	P	ΔV
ΔV	A	r_1	28	44	44	1	1	0	0.5
	B	r_2	28	52	52	3	1	0	0.8

Table 2b): Information table of discrete functions, e.g. Track-Status.

$\Delta T=T_f-T_i$			COMMUNICATION DATA			LINGUISTIC DESCRIPTION			
			IP		OP	IP			OP
			T_i	O_2	T_f	n	d	P	ΔT
ΔT	A	r_3	1	3	3	2	1	1	2
	B	r_4	1	5	5	4	1	1	4

Table 2c): Information table of logical functions, e.g. On / Off

L_f: Final State			COMMUNICATION DATA			LINGUISTIC DESCRIPTION			
			IP		OP	IP			OP
			L_i	O_3	L_f	n	d	P	L_f
ΔL	A	r_5	1	0100	0	0	0	1	0
	B	r_6	0	0111	1	0	0	0	1

Such that: S_x is the observed initial situation whereas the body of the IF condition encodes the humans' action(s) with implicit timing parameters. Concurrently, ΔY indicates the reaction(s) introduced by the DUT with implicit timing parameters too.

D. *Case Study:*

Table (2a) represents different training samples for a continuous state, e.g. volume level, observed from diverse testers (A and B). The invoked human's action (O_1) is *Set Volume (Volume-Level)* to shift the volume state from an initial value of (28) to various levels, e.g. 44, 52, etc. Then, several individual rules (r_1, r_2) would be described as follows:

For all S_I=S_1:
r_1: IF (n (0) = 1 ∧ d(0)=1 ∧ P(0)=0) THEN (ΔV=0.5)
r_2: IF (n (0) = 3 ∧ d(0)=1 ∧ P(0)=0) THEN (ΔV=0.85)

Relying on individual rules that grow exponentially tends to obtain models that signify overfitting. Therefore, a fruitful way is to generalize the individual rules r_1, r_2, etc. A corresponding generalized rule would be:

For all S_I=S_1:
IF (n (0) = n_i ∧ d(0)=1 ∧ P(0)=0) THEN (ΔV_i=f (n_i) ± μ)

Such that, (f) encodes a non linear mapping between the numbers of turns (n_i) and the corresponding change in the volume output (ΔV_i) with a given tolerance (μ).

Similarly, Table 2b shows several training cases for a discrete state, e.g. CD track status. Here, the invoked human's action (O_2) is *Set Track (Track-Number)* from an initial value of (1) to different final states, e.g. 3, 5, etc. Hence, multiple individual rules (r_3, r_4) would be described as follows:

For all S_I=S_2:
r_3: IF (n (1) = 2 ∧ d(1)=1 ∧ P(1)=1) THEN (ΔT=2)
r_4: IF (n (1) = 4 ∧ d(1)=1 ∧ P(1)=1) THEN (ΔT=4)

Similarly, relying on numerous individual rules would be of no practical use. Therefore, a corresponding generalized rule would be learned by the test system such that:

For all S_I=S_2:
IF (n(1)=n_j ∧ d(1)=1 ∧ P(1)=1) THEN ($\Delta T_j = g\ (n_j) \pm \mu_o$)

Here, an equality function *(g)* encodes the relationship between the actions' sequence, which is the clock wise rotation of the button with the index of (1) by (n_j) turns followed by a push action, and the change of the track status (ΔT). In addition, (μ_0) is a tolerance only for the reaction time whereas the tolerance for the reaction itself is zero.

Finally, Table (2c) shows an example to learn the behavior of logical states in response to the human's action (O_3) *Set Mute* or *Set Lou*d. Similarly, a set of individual rules (r_5, r_6) is defined such that:

For all S_I=S_3 (L_i=1):
r_5: IF (n (2) = 0 ∧ d(2)=0 ∧ P(2)=1) THEN (L_f=0)

For all S_I=S_4 (L_i=0):
r_6: IF (n (2) = 0 ∧ d(2)=0 ∧ P(2)=1) THEN (L_f=1)

Then, the set of generated individual rules would be generalized like the following:

For all (S_I=S_3 ∨ S_I=S_4):
IF (n (2) = 0 ∧ d(2)=0 ∧ P(2)=1) THEN ($\Delta L = h\ (L_i) + \mu_*$)

Indeed, a negation function *(h)* encodes the expected reaction of the DUT (L_f) in case the button indexed by (2) is pushed. Similarly to discrete states, (μ_*) indicates an allowable tolerance concerning the reaction time whereas the tolerance for the reaction itself is zero.

Finally, it is worthwhile to mention that exact test cases are subjected first to a consistency check (see Section II-C), before they are used as training samples. Furthermore, individual rules are not only learned from deterministic test cases, but also from decomposing non deterministic cases using the reasoning algorithm described in Section (III).

VI. Conclusion and Future Work

A. Summary

This contribution outlines an ongoing research work that aims to enrich traditional test techniques with intuition-based test strategies learned by observing skilled human testers. Then, strategies learned would be verified, combined, and then generalized to be applied in similar problem domains.

B. Limitations versus Contributions

One of the intrinsic limitations of this approach is its relatively narrow domain of applicability. Indeed, it looses a reasonable portion of its effectiveness, if the human factor is not deeply involved in the test process. However, the proposed framework gets a wide acceptance in the automotive domain, where testing of reconfigurable DUTs, e.g. driver assistant systems, receives a fair deal of attention.

Moreover, the developed framework lacks the power to mathematically describe the motivations behind its own generated strategy. This is definitely true since strategies generated by human experts, which are a part of the self-generated strategy, stem from their rules of thumb derived by cognitive heuristics rather than mathematical formulas.

Nevertheless, the proposed approach enriches traditional test techniques with an optimal combination between automatic procedures and intuition-based test strategies to maximize individual benefits.

In addition, confronting the fact that manual construction of test data sets consumes a large part of the test effort during the development cycle [22]; the proposed framework provides the capability to teach technical systems test scenarios just by demonstrating them. This leads to a substantial reduction in the time and energy devoted in writing script-based scenarios. Adding to this, it eliminates the heavy burden thrown on the human experts while trying to verbalize their past experiences.

Second, cooperative learning reduces the workload thrown on human experts who have to test any similar configurations of the DUT. Moreover, it avoids just imitating one strategy learned, rather replicating the strategies learned with inertia through the autonomous generation of new test suites.

Third, the reasoning paradigm enables the test system to learn behavioral primitives of the DUT. This consequently leads to the automatic generation of its specifications without any extra workload on the quality assurance staff since the test process has to be any way accomplished.

Fourth, the proposed theory to generalize strategies learned reduces the necessity of the periodical engagement of skilled human testers in case the availability of new releases.

C. Research Plan versus Evaluation

Within the scope of this contribution, the availability of skilled human testers is assumed. However, it is planned to investigate the requirements on human testers to be considered as skilled testers. For instance, their history sheets, number of revealed errors versus execution time, etc.

In addition, it is necessary to define quantitative metrics to evaluate the strategies learned in order to minimize the exposure to overfitting or underfitting. For example, the overall states' coverage resulted from the training sessions with respect to the pre-designed specifications of the DUT.

One other key element of the future work is to generate the behavioral primitives (specifications) of the DUT using a standard data format, e.g. Finite-State-Machine (FSM). This would increase the degree of acceptance of the developed framework since no extra tools have to be learned.

Furthermore, a long-term goal is to offer the test system various training sessions with a real DUT and then test its generalized strategy against a simulated one. This is especially effective to judge how limited the generalized rules are since it is feasible to simulate new test situations that might be difficult to be done using a real DUT.

Finally, feasible evaluation metrics can determine the extent to which improvement has been reached. Generally, the designed metrics fall under two categories; how much the human efforts are reduced? And how far the test process has been improved?

Qualitative metrics are supposed to measure the degree of satisfaction from the rules expressive power. Additionally, it is worthwhile to test the fidelity of the extracted rules to mimic the strategies learned.

Quantitative metrics would be reasonable to evaluate the extent of the test process improvement. For example, the number of new generated test scenarios in comparison with the learned ones, percentage reduction in the time needed by human testers, percentage increase of the found errors, percentage increase of the states' coverage, how often the test system misfired the generalized rules, and its ability to predict the behavior of new configurations.

VII. REFERENCES

[1] G. Denaro, L. Mariani, and M. Pezze: Self-Test Components for Highly Reconfigurable Systems. In Electronic Notes in Theoretical Computer Science, Volume (82), Number (6), 2003.

[2] A. Hessel: Model-Based Test Case Selection and Generation for Real-Time Systems, PhD thesis. In Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology, ISSN 1651-6214; 301, 2007.

[3] E. A. Bezerra, F. Vargas, and M.P. Gough: Improving Reconfigurable Systems Reliability by Combining Periodical Test and Redundancy Techniques: A Case Study. In Journal of Electronic Testing: Theory and Applications, Volume (17), Issue (2), Pages 163-174, 2001.

[4] M.P.E. Heimdahl and D. George: Test-Suite Reduction for Model Based Tests: Effects on Test Quality and Implications for Testing. In the Proceedings of the 19th International Conference on Automated Software Engineering, IEEE Computer Society, 2004.

[5] D. A. Wooff, M. Goldstein, and F. P. A. Coolen: Bayesian Graphical Models for Software Testing. In IEEE Transactions on Software Engineering, Volume (28), Number (5), May 2002.

[6] J. Gras, R. Gupta, and E. Minana: Generating a Test Strategy with Bayesian Networks and Common Sense. In the Proceedings of the Testing: Academic and Industrial Conference- Practice and Research Techniques, IEEE Computer Society, 2006.

[7] L. A. Zadeh: Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. In IEEE Transactions on Systems, Man, and Cybernetics, Volume (3), Number (1), Pages 28-44, January 1973.

[8] C. Huang and M.R. Lyu: Optimal Release Time for Software Systems Considering Cost, Testing-Effort, and Test Efficiency. In IEEE Transactions on Reliability, Volume (54), Number (4), December 2005.

[9] C. Kaner: The Impossibility of Complete Testing. In Law of Software Quality Column, Software QA Magazine, Volume (4), 1997.

[10] J. Bach: A Framework for Good Enough Testing. In the Proceedings of the IEEE Computer Society, Volume (31), Number (10), Pages 124-126, 1998.

[11] D. Söffker: Interaction of Intelligent and Autonomous Systems-Part I: Qualitative Structuring of Interaction. In Mathematical and Computer Modeling of Dynamical Systems, Volume (14), Issue (4), Pages 303-318, August 2008.

[12] A. Eltaher, T. Form, M. Ayeb, and M. Maurer: A Generic Architecture for Hybrid Intelligent Test Systems. In the Proceedings of the 7th IEEE International Conference on Cybernetic Intelligent Systems, September 9-10, London, United Kingdom, 2008.

[13] M. Maurer, A. Eltaher, M. Reichel, T. Müller, M. Miegler, D. Niederkon, and B. Strasser: In the 1st Workshop on: New Tools for Automated Testing Techniques Applied to Infotainment Systems, AUDI AG, Ingolstadt, Germany, June 2008.

[14] Y. Levendel: Using Untampered Metrics to Decide When to Stop Testing Software. In the Proceedings of the IEEE International Conference on EC3-Energy, Computer, Communication and Control Systems, Volume (2), Pages 352-356, August 1991.

[15] R. Xu and D. Wunsch: Survey of Clustering Algorithms. In IEEE Transactions on Neural Networks, Volume (16), Number (3), May 2005.

[16] A. Eltaher: Towards Good Enough Testing: A Cognitive-Oriented Approach Applied to Infotainment Systems. In the Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering, September 15-19, 2008, L'Aquila, Italy.

[17] H. Liang, J. Dingel, and Z. Diskin: A Comparative Survey of Scenario-Based to State-based Model Synthesis Approaches. In the Proceedings of the 2006 International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools, ISBN: 1-59593-394-8, Pages: 5-12, 2006, China.

[18] A. Eltaher and T. Form: Modeling and Realization of Human-Machine-Interaction as an Approach for a Self-Learning Test System. In the Proceedings of the 3rd International Conference on Simulation and Testing, ISBN 978-3-8169-2818-8, 2008, Germany.

[19] C. Alippi and P. Braione: Classification Methods and Inductive Learning Rules: What We May Learn from Theory. In IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, Volume (36), Number (5), 2005.

[20] J. Huysmans, R. Setiono, B. Baesens, and J. Vanthienen: Minerva: Sequential Covering for Rule Extraction. In IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, Volume (38), Number (2), April 2008.

[21] W. Duch, R. Adamczak, and K. Grabczewski: A New Methodology of Extraction, Optimization, and Application of Crisp and Fuzzy Logical Rules. In IEEE Transactions on Neural Networks, Volume (12), Number (2), March 2001.

[22] B. Baudry, F. Fleurey, J. M. Jezequel, and Y. Le Traaon: Automatic Test case Optimization: A Bacteriological Algorithm. In Proceedings of the IEEE Computer Society, Volume (22), Number (2), Pages 76-82, 2005.