Testing Electronic Control Units with Support of Ontologies and Rules

Willy Chen[1] and Thomas Syldatke[2]

[1] mail@chenwilly.info
[2] thomas.syldatke@audi.de

Abstract. Due to the steadily growing percentage of software and electronic components in cars today the testing of these systems is getting more and more complicated. Intelligent IT solutions could help to face these challenge. The application of ontologies and rules has thereby yield very promising results for automating important parts of the overall testing process.

1 Introduction

Thorough testing is an essential part in the development of a new car to ensure an outstanding quality of the final product. This is also true for embedded software and electronic components. While such systems have been quite simple in the past, they are nowadays getting more and more complex (e.g., driver assistance systems, comfort vehicle controls). The complexity is thereby not only driven by the actual functionality but also by the application in different car configurations and environments. Furthermore, their number has grown up to 60 electronic control units (ECU) in a single premium class car.

An important task in the overall testing process is the simulation of a physical ECU in a virtual car environment, which is also known as Hardware–in–the–Loop (HiL) test. Unlike tests in real prototypes, critical situations and system states can be covered without any risk. Targets during HiL tests are the validation of the ECU against its specification and other given requirements. The main problem thereby is the huge amount of data, which is being generated and recorded during such a test run. Therefore, the analysis of that data is not only very time-consuming, but one could easily overlook a fault.

Existing software solutions do assist the engineers in performing the analysis by providing different visualizations of the recorded data. However, they lack a standardized way to define the desired system behaviour on the one hand and known error cases on the other. The application of ontologies to provide an uniform vocabulary for that domain and rules to specify the system behaviour as well as error cases have shown promising results to face these issues and automate this part of the testing process.

This paper will give a brief survey over a prototype realized for Audi. Section 2 summarizes the concrete business use case. The prototype itself is presented in section 3 and discussed in section 4. Finally, section 5 concludes the paper and gives some plans and ideas for the future work on this topic.

2 Business Use Case

In 2006 Audi has announced the first engines including the Audi valvelift system. “This innovation varies the valve lift between two levels. To achieve this, sets of sliding cams are mounted directly on the intake camshafts. These feature two sets of adjacent cam contours for small and large valve lift. Which cam is used to open the intake valves depends on the power demand at any one time. The effect is an appreciable increase in engine efficiency. The driver benefits from greater power and improved driveability, while enjoying a marked reduction in fuel consumption“ [1].

The valve lifts are controlled by the engine management system. A deterministic finite automaton[3] internally represents the possible states of the valvelift system and specifies the conditions for switching the valve lifts. This automaton has 6 different states $S_1..S_6$ whereas S_1 is the start state. S_1 and S_4 represent the states in which the small respectively the large valve lift is being used. The other ones are transition states that have to be passed in a given order. The state transition functions δ_n finally define when to switch between the valve lifts. An example for such a transition function is: “If the engine speed is greater than 4000 the valvelift system must switch to S_4 if it is in S_1”. These details are defined in the specification of the engine management system.

Without knowing any internal details about the engine management system an engineer could also specify the expected behaviour of the valvelift system from another point of view - e.g.: “at idle speed the small valve lift must be used”. During expert interviews we have been able to elicit 18 of these kind of rules that need to be kept in mind when testing the engine management system.

3 Prototype Development

The main challenge for a first prototype is to merge both sources of knowledge about the valvelift system and provide a way to automatically validate and analyze the data recorded during the HiL tests. Therefore, a flexible way of formalizing the given rules for the transition functions as well as the system constraints is required.

Adopting ontologies for defining an uniform domain vocabulary and rules is a promising way to meet these demands. Both have recently gained increasing attention from researchers as well as the industry, because they are considered key technologies for enabling the vision of a Semantic Web. First companies are also starting to provide professional support in that technology niche.

Since researchers still disagree about the best way of combining (OWL) ontologies and rules, we have selected the representation language that comes with the best professional support. F–Logic [2] together with the inference machine OntoBroker [3] from Ontoprise[4] turned out to be most suitable for our demands.

3.1 Domain Ontology

The first task in the prototype development is the modelling of a domain ontology that provides a common vocabulary for the definition of rules. We thereby need to take the

[3] Because of confidentiality reasons we will refer to an abstracted model thereof.

[4] see http://www.ontoprise.de

format of the recorded data from HiL tests into account, since they will in the end be the facts of our domain ontology.

Using the tool INCA it is possible to take snapshots of internal variables of an ECU (e.g., current state of the finite automaton, engine speed, oil temperature) during the HiL tests with a minimum time interval of approx. 10ms. Table 1 gives an example of such a recorded data set.

$time(ms)$	var_1	var_2	var_3	var_4	..	var_{24}
0.003264454	2519	1	2	90	..	15005
0.013361665	2519	1	2	90	..	15006
..	..	..	..	..	..	..

Table 1. Example data set from INCA

To keep the first prototype quite simple, we define two concepts:

– `Situation`: represents a single snapshot of the ECU at a given time including the recorded variables and a relation to the successor snapshot if present.
– `State`: the facts of this class represents the possible states of the finite automaton.

3.2 System and Expert Rules

The basic idea for defining rules on top of the domain ontology is on the one hand to simulate the expected state that should be present at a given snapshot. Therefore we add a relation `nextState` to the concept `Situation` and can derive its value based on the given transition functions. The above mentioned rule ("If the engine speed is greater than 4000 the valvelift system must switch to S_4 if it is in S_1") is for example represented in F–Logic as follows:

```
FORALL S,SS,V S[nextState->S4] <-
   S:Situation[state->S1;engineSpeed->V] and
   greater(SS,4000).
```

On the other hand we can use this derivation results to check, whether it is equal to the measured state. Since F–Logic does not explicitly support integrity rules, we use a predicate therefor:

```
FORALL S,SS,X,Y Error(S) <-
   S:Situation[nextState->X;successor->SS] and
   not SS[state->Y].
```

Based on this approach we can seamlessly integrated the expert rules by formulating them as error statements:

```
FORALL S ERROR(S) <- S:Situation[zustand->>S4;IDLE->>1].
```

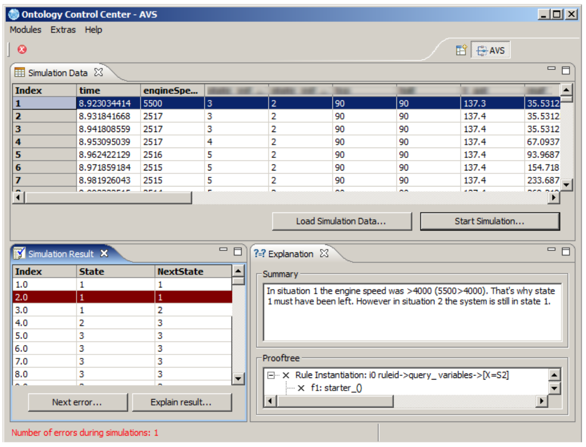
The OntoBroker offers the possibility to retrieve the proof tree for a specific query result – i.e. which variables and values have been used in which rule. This feature can be utilized to define explanations for rule. A detailed description can be found in [3].

3.3 The Prototype System

Figure 1 shows the prototype system, which is embedded as a plugin in an Eclipse RCP-based application[5], which also includes a module to manage ontologies and rules. The main features are:

- Import HiL test data: The data recorded with INCA can be transformed into instances of the domain ontology.
- Simulation Run: By applying the defined rules, the test data instances are checked for possible errors, which are then highlighted.
- Explanation of results: If available, an explanation for a detected error is presented.

Tests with real data from HiL tests have shown reasonable results. Furthermore, some minor errors could be detected as well as explained. Even with a large number of instances (approx. 50.000) the prototype performed very well.



Index	time	engineSpe...	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
1	8.923034414	5500	3	2	90	90	137.3	35.5312
2	8.931841668	2517	3	2	90	90	137.4	35.5312
3	8.941808559	2517	3	2	90	90	137.4	35.5312
4	8.953095039	2517	4	2	90	90	137.4	67.0937
5	8.962422129	2516	5	2	90	90	137.4	93.9687
6	8.971859184	2515	5	2	90	90	137.4	154.718
7	8.981926043	2515	5	2	90	90	137.4	233.687

Index	State	NextState
1.0	1	1
2.0	1	1
3.0	1	2
4.0	2	3
5.0	3	3
6.0	3	3
7.0	3	3
8.0	3	3

Summary

In situation 1 the engine speed was >4000 (5500>4000). That's why state 1 must have been left. However in situation 2 the system is still in state 1.

Prooftree

```
⊟ X Rule Instantiation: i0 ruleid->query_ variables->[X=S2]
    X f1: starter_()
```

Number of errors during simulations: 1

Fig. 1. Screenshot of prototype

4 Discussion of the Prototype

The prototype system has been accepted very positively by the responsible engineers and do address their current needs. We have also been able to show the value of the application of ontologies and rules. The most important benefits are thereby:

- Integration of specification and expert knowledge: With ontology and rules, it has been possible to merge two completely different sources of knowledge and make them processable by computers. Therefore, a high automation of performing data analysis of HiL test results could be achieved.

[5] see http://www.eclipse.org

– Evolving rule bases: The rule base does not necessarily need to be complete at any time. In fact the better the rules are the more errors could be detected automatically. Changing a rule can also be done very quickly. It is important to stress that the presented prototype does not aim to replace current testing technologies or experts but to support the engineers in doing their job more efficiently.
– Sharing and reuse of knowledge: Rules that capture specific knowledge can be centrally stored and made available to a broader audience. Knowledge islands could thus be avoided. In addition, each data analysis is able to apply all previously defined rules. Thus the knowledge about why an error has occurred is not get lost.
– Explanation of results: The usage of appropriate inference machines allows a detailed explanation of the used rules with only a minimal amount additional work. This may give an engineer a clue for finding and solving an error.

5 Conclusion and Outlook

The application of ontologies and rules for supporting the testing process of ECUs has shown promising results. We have successfully convinced the decision makers and are now working on the integration of additional ECU functions.

Future plans and ideas for evolving the presented approach are:

– Support of different vendors: Various electronic control units are offered by different vendors with exactly the same functionality but different internals. Thus, an adequate support by layered vendor ontologies would be desirable.
– Support regulatory requirements: Every market (e.g. European Union, US) has different regulatory requirements. Instead of defining rules for each market, the regulations could be modelled in market ontologies. Depending on the market an ECU should be tested for, the appropriate ontology can be applied dynamically during data analysis.
– Automated extraction of rules: Since technical specifications are highly formalized, it could be possible to automate the extraction of the essential rules and therefore provide a seamless integration into today's testing infrastructure.

Acknowledgement

Achievo Inproware GmbH has been responsible for the development of the presented prototype. Their work has been supported by Ontoprise. Both companies are now cooperating as partners to bring that approach to a broader audience.

References

[1] Audi. The new Audi A5 / Audi S5. http://tinyurl.com/3bv6u3, 2007.
[2] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *JACM: Journal of the ACM*, 42:741–843, 1995.
[3] Ontoprise. *How to Use OntoBroker - Users and Developers Guide for the OntoBroker - System / Version 4.3*, 2006.