Journal of Universal Computer Science, vol. 5, no. 10 (1999), 730-741
submitted: 14/4/99, accepted: 7/9/99, appeared: 28/10/99 © Springer Pub. Co.

Electronic Throttle Control – A Dependability Case Study

Hans Mauser
(Siemens AG, ZT PP 2
hans.mauser@mchp.siemens.de)

Erwin Thurner
(Siemens AG, ICN WN ES HW 42
erwin.thurner@icn.siemens.de)

Abstract: The so-called Electronic Throttle Control unit was a big step towards reducing important parameters like fuel consumption or exhaust emmission. Due to its safety-criticality, a dependability study was initiated by the manufacturer Siemens Automotive. As the most important result, values could be stated for the quantitative estimation of the safety-critical and the availability-relevant cases. The study was based on the existing safety concept, but after this study, a slightly changed system architecture of the ECU was proposed to VDA (Verband der Automobilindustrie), which enhances availability and safety of the ECU significantly, at about the same cost. For this study, a new kind of Markov evaluation method was used, called TEFT (Time-Extended Fault Trees). This was necessary to deal with concepts like multiple faults, faulty states, and time. In this paper, the questions raised by the Electronic Throttle Control system are described, together with our way to solve these problems.

Key Words: Dependability, Availability, Safety, Markovian methods, TEFT (Time-Extended Fault Trees), Cars, Powertrain, ECU (Electronic Control Unit), Electronic Throttle Control (ETC)

1 Problem Description

Higher requests caused by public and by law, to improve fuel consumption, exhaust emmission, or comfort, forced the development of the Electronic Throttle Control (ETC) system. This solution removes the traditional Bowden cable, and uses the torque request instead as relevant parameter [see Fig. 1]. The torque request can be generated by the driver or by car comfort systems.

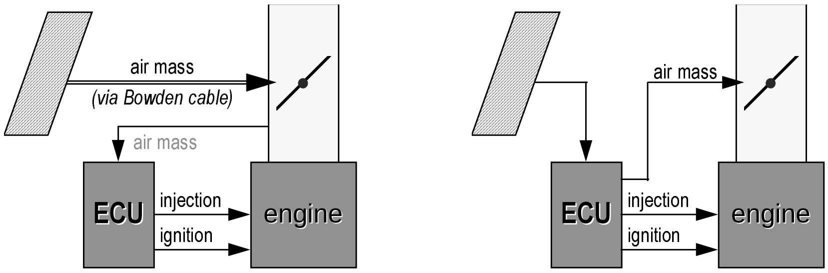

Fig. 1: Traditional Bowden cable solution versus Electronic Throttle Control

This approach makes it possible to give the car engine in every moment exactly the necessary mixture of air, fuel, and ignition angle, in contrast to the Bowden cable solution, in which fuel and ignition angle try to follow the air mass provided by the throttle. Due to this exact and consistent engine control, ETC reduces fuel consumption and reaches low-emission requirements. In addition, this concept eases the integration of comfort functions like cruise control or anti-ice measures, and environmental measures, e.g. heating the catalyst.

The ETC unit, however, is considered to be safety-critical. Due to this, a dependability study was initiated by the ETC manufacturer Siemens AG, section Automotive. Goals were to scrutinise safety and availability of the considered Electronic Throttle Control system solutions, in order to detect weaknesses and to select the best architecture. The considered architectures were compared to the existing ETC system proposal. With our approach, it was possible

- to make quantitative statements about the safety and availability of the system, also considering degrading states,
- to compare the considered architectures quantitatively,
- to detect weaknesses of the system already in the design phase,
- to give a base for system tests.

2 Considered Faults

As a first step, possible faults and their system interference were analysed by a FMEA (Failure Mode and Effects Analysis, [3]). The fault sources were roughly structured as faults appearing in sensors, actuators, and ECU (Electronic Control Unit). In the next section, we will highlight the differences in the structure of the faults. By using our Markovian method TEFT, it became easier to describe the influence of any faults on the system behavior, also integrating system states and time.

2.1 Sensor Faults

From the beginning, the pedal sensor was considered to be safety-critical. Due to this, the sensors generally are built with redundancy, to be able to detect and to mask appearing faults. Note that common-mode failures like a mechanical sensor break has the same effect as they have at Bowden cables, namely stuck at a random sensor value – e.g. at highest value! The measures to avoid faults in the A/D converters were not clear at the project beginning: the standard architecture only checks the A/D converters regularly, in the other architectures we also considered alternatives with redundant components.

Other sensors – like the oil temperature sensor – were not considered safety-critical. Here, the goal of this study was to check the plausibility of these assumptions.

Other "sensor" signals come via the CAN bus. These signals were also considered, and their influence on the system behavior appeared as being under-estimated.

2.2 Actuator Faults

Actuators do directly influence the engine behavior. Examples are ignition angle and fuel injection mass. The throttle seemed to be most significant: Here, both actuators faults *and* sensor faults may appear, and this device is critical due to high temperature or ice.

Actuator faults have in common, that they are rather an availability problem than safety-critical: They rather *decrease* the engine performance than increasing it.

2.3 Faults of the Electronic Control Unit (ECU)

The remaining faults appear in the ECU: Examples are processor faults, computing faults caused by hardware defects, or driver faults. These faults cause random values. Some of these faults are transient ones, so measures (like plausibilty checks) can be taken to detect and possibly mask them.

In the ECU, the most failure-avoiding concepts of the ETC system are found. These will be considered next.

3 Monitoring Concepts

To make sure that the system is functioning correctly also in case of faults, several checking strategies were designed. The overall goal was to discover as many faults as possible and to trigger an adequate reaction, in order to avoid critical states. Particularly the *runaway* case, i.e. undesired high engine performance, has to be avoided.

To get the desired torque (performance) the ECU controls the parameters *air mass, fuel mass*, *ignition angle*. The ECU also controls the checking concepts. Due to this important role, the main processor is considered to be very important. The software of the ECU was considered as correct, but hardware faults that lead to software errors were taken into account:

– Faults that lead to a partial processor failure.

– Memory faults that influence critical variables.

– I/O errors, e.g. given by the A/D converters.

3.1 Processor Monitoring

Processor monitoring are measures that guarantee the correct function of the ECU processors. Examples are watchdog-timers. Due to a two-processor concept, each ECU processor can be checked by the other. Implemented *function tests* are:

- Instruction set tests execute representative instruction sequences and check the result.
- Memory tests detect stuck-at faults in the processor memory.
- Test computings check the complete function of a software part, by executing tests and comparing the results.

These monitoring-means efficiently check processor functions. Note that most processor faults lead to a total system failure causing an engine halt.

3.2 Process Monitoring

Process monitoring deals with the complete function of the ETC system. This is done by executing all safety-critical computation twice. To guarantee the efficiency of this checking strategy, the system has to be redundant ("two channels"). The processors get input values from different sensors, they have own A/D converters, and they use diverse software. By these measures, the appearing of identical faults is avoided.

3.3 Plausibility Monitoring and Signal Comparison

All the input values are checked for their plausibility. Short circuits and wire breaks can be discovered as leaving the allowed value range. All safety-critical input values are measured by two sensors, and compared between each other. Sensors often use different transfer functions.

These checks deal with the complete input stream, including sensors, signal wires, and A/D converters.

4 Safety Reactions

In order to guarantee the safety of the ETC system, a variety of safety reactions with different severity grades were implemented. The overall goal was to keep the system running as long as possible, e.g. to be able to maintain the heating function of the motor in very cold regions.

Thus, the following *system modes* were implemented (in rising severity):

- **intact:** Although an error occured, the system reacts according to its specification. In this mode, the built-in redundancy avoids a system error. This state is possible as long as no safety-critical fault occors.
- **extended limp home:** In this mode, the car still reacts on the pedal request, but acceleration and torque are reduced, in order to avoid endangering. Thus, the driver is able to react within reasonable time.
- **limp home:** The engine runs a slightly higher idle number-of-revolutions. The goal is to maintain heating and servo functions. Possibly, the car still can be moved.
- **safety stop:** In this mode, the car is stopped. This mode is selected, if no other means are able to maintain safety.

The safety of the system modes increases in the sense, that the probability of a *runaway* decreases. These modes are implemented as *internal states*. The selection of the appropriate state as a fault reaction is an important part of the safety-concept.

For the modeling approach is important, that this system has an *internal memory*. The ECU reaction depends on the current mode, i.e. on the errors that occured in the past.

5 Modeling Technique

We took an "extended fault tree approach" to model this system. To motivate this, let us shortly consider some features of standard fault trees:

- Fault trees are a system function that maps sets of component faults onto a specified undesired event.
- *Stochastic dependencies* have to be taken into account explicitly. In many fault tree tools, extensions exist to decribe this. A similar problem is raised by common-mode failures.
- Using highly redundant or re-configurable systems, the fault *sequence* sometimes plays a role: It is a difference, if a component fails first, or its checking function (cf. silent errors!). This kind of faults need the introduction of *states*. Unfortunately, fault trees only consist of (combinatorial) system functions. Furthermore, fault trees are usually defined over sets of faults, not over ordered sets.

- For the system analysis, only *relevant* fault sequences have to be considered. Thus, we have to "cut" the evaluation, when the system is coming into the *safety stop* mode.
- Fault rates may be *state-dependent*. A typical example is warm redundancy.

Note that this obstacles need not cause problems if they are properly taken into account. But on the other hand, for complex systems it can be rather difficult to model this using fault trees. Furthermore, significant errors can be caused by a naive evaluation of redundant systems.

Generalised Stochastic Petri Nets (GSPNs, [1]) are able to cope with the aspects mentioned above. This formalism uses a Petri Net extension for the description of the system failure behavior and its interdependencies, and maps this net to Markov chains for an evaluation, as pointed out in [1]. Thus, concurrent subsystems, synchronisation, system states, and exponential failure rates can be modeled consistently.

5.1 The TEFT Approach

For modeling the ETC system, we used our Markovian technique TEFT (Time-Extended Fault Trees). In this approach, the system function is modeled using fault trees, and the temporal behavior is modeled by GSPNs. The state transitions caused by faults are described by fault trees in a functional way. This comprises component faults, distinguished by the state in which they may occur, and the sequential state. TEFT can only be used for acyclic Markov chains. This usage is motivated by the *repairing strategy of car-based systems*: As soon as a fault occurs, cars are expected to be repaired in a way they behave like new ones, i.e. the Markov chain is reset to its initial state.

The evaluation is done by state space exploration: All fault sequences are generated, until a given depth, i.e. number of errors, is reached. This reachability graph with the exponential fault rates of the components, spans a Markov chain. The evaluation of this Markov chain gives the probability of the considered failures. For the performance of the Markov chain solver it is very important, that the reachability graph does not need to be held within memory, thus avoiding the main obstacle of state space explosion. The TEFT evaluation algorithm evaluates 200,000 states within one minute, so that one billion states can be reached within reasonable computing time. Furthermore, time-consuming iteration is avoided. The solution can be given as closed-form expression.

This TEFT approach enables the modeler to put very detailed questions to the system by computing several fault sequences, e.g. to measure how long the system can be used after the first fault.

5.2 System Components

The first modeling step is to break the system down into its "components" or "logical blocks". This dissection is determined by the appropriate abstraction level.

The ETC system was broken down into 40 logical blocks, comprising sensors, actuators, processors, and software components. If several logical blocks appear to be a serial system, they can be aggregated to one single logical block, to ease modeling and reduce the evaluation complexity. For each component the failure modes are listed; this work is done on the base of a FMEA. Note that these failure modes only describe the faults of a single component. Here, several failure modes are possible, i.e. the pedal faults can be described in "Runaway/Middle/Idle" rather than in a Boolean "Ok/Faulty". For ETC, we considered 72 failure modes. All the failure modes get (exponential) fault rates, expressed in FIT (Failure In Time, i.e. within 10^9 hours).

5.3 Functional Part

The *fault effects*, i.e. the system behavior that can be watched from outside, were distinguished like this:

- **intact:** The car reacts in the specified way.
- **accelaration drift:** Slight undesired accelerations are noted by the driver. The driver usually is able to compensate this system reactions; they compare to effects like sudden wind changes.
- **limited performance:** The car still reacts on the pedal request, but acceleration and torque are reduced. This reaction appears in system mode *extended limp home.*
- **idle performance:** The engine runs at slightly increased idling speed. This reaction appears in system mode *limp home.*
- **engine stop:** The engine is turned off.
- **runaway:** Undesired high engine performance appears, e.g. high accelerations.

These effects are considered like "top events" in faults trees, i.e. undesired resulting events. Each combination of faults falls into exactly one of these categories; this circumstance can be used for model consistency checks. The modeling has to be done manually. In our case, it required an extensive modeling specification; in this work package, we and our customers had many useful discussions and got a lot of very relevant knowledge that was represented in the resulting model. In the modeling phase, the *multi-value* property of faults and the availability of system *states*, which are implemented in TEFT, turned out to be extremely useful.

5.4 Temporal Behavior

Time appeared in two aspects:

1. Some faults were only possible after primary faults, i.e. not in every case and particularly not in the initial state of the system. TEFT describes this by using state-dependend activation conditions.
2. The internal states, i.e. the system memory, have to be modeled. TEFT implements this by a reference on more than one state.

6 Model Validation

A big advantage of TEFT is, that every fault sequence and its effect can be listed. This list may be compared to minimal cut sets of fault trees, but representing ordered sets. It can also be sorted by the resulting FIT rate. This representation may remind to a FMECA analysis, but the criticality is not estimated, but computed. (There are also some formal differences, so we will avoid naive comparison.)

This list can also be used as a base for generating test cases: Then, it can be used to compare the real system directly with the modeled one. And it can be compared systematically with every system that leaves a production line.

The length of the list produced by TEFT is not always a pure advantage: Several 100,000s of list entries can no longer handled with standard table calculation tools. Furthermore, to order the results may become difficult. On the other hand, it really reflects the system complexity, and by using postprocessors the handling can become very easy, without losing information. We often do this by writing a "question sensitive postprocessor". This makes the captured results very valuable.

The evaluation speed makes it also possible to vary parameters, and to enable the modeler to study the effect on the complete system. This can be used both for the variation of component values, and for components, whose FIT rates are unknown, e.g. very new components.

7 Modeling results

We modeled and evaluated several system alternatives, to study several architectural changes that were in discussion.

All systems have in common, that redundant pedal and throttle sensors are used. System differences are:

- **System #1:** This is a one-processor solution. The processor does process checking. We modeled this system to compare the availability of the redundant solution to a – non-existing – naive one.

- **System #2a:** This system uses two processors, which do processor checks. No process checking is done.
- **System #2b:** The only difference to #2 is the process checking.
- **System #3:** This is a kind of true 2-channel system. The processors do processor checks; in addition the 2nd processor does process checking. Both processors have A/D converters, and the redundant pedal and throttle signals are delivered to both processors.

The following table gives an overview of the main system differences [see Table 1]. Note: "p1" means processor #1.

	A/D converters	**Comparing redundant signals**	**Processor check p1**	**Processor check p2**	**Process checking**
System #1	p1	p1	*none*	*irrelevant*	p1
System #2a	p1	p1	p2	p1	*none*
System #2b	p1	2x p1	p2	p1	p1
System #3	p1 and p2	p1 and p2	p2	p1	p2

Table 1: Significant differences of the considered system alternatives

7.1 Single Faults

Since cars are expected to be maintained regularly, single faults are expected to play the most important role in the system. All the systems turned out to be rather similar; exception was the safety-critical *runaway* case. For the *runaway* probability we got the following values [see Fig. 2]:

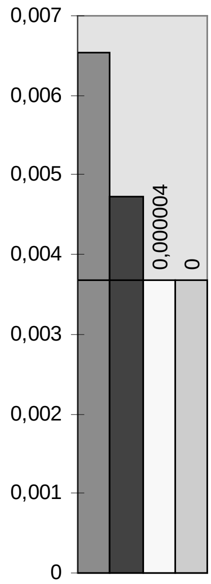

Fig. 2: Results for single faults (scale is linear, using arbitrary units)

Much less than 1% of the single faults lead to a *runaway*. The fault cases are:

- The greatest part of the resulting probability is caused by faulty torque requests via the CAN bus. Note that in such cases the ETC system has no possibility to recognize these torque requests as faulty. These requests can be induced by several car components; examples are gear switch signal at automatic transmissions, the so-called deceleration slip control, or faulty reference voltage at the pedal sensor. Moreover, at system #3 these torque requests are the only single faults that lead to the *runaway* case.
- Faults in the A/D converters may cause a runaway, if both pedal sensor values are wrong. This is possible, if both analogue signal from the pedal are multiplexed to a single A/D converter; only at system #3 this fault constellation is avoided. The A/D converter errors can be recognized with a rather high probability, so that the resulting effect is rather small. But this is the only single fault that leads to the *runaway* case system #2b.
- Systems without process check – e.g. system #2b – may trigger the *runaway* case by faulty torque computings.
- Systems without processor check – e.g. system #1 – may trigger the *runaway* case by a processor error.

The modeling result was that only the systems #3 and #2b are safe enough for the real-world implementation. Another result was to have a closer view to the external torque requests: Their plausibility cannot be checked by ETC-immanent means. Due to this, these comfort (!) signals have to be considered as being safety-critical.

7.2 Double Faults

The main goal for the double fault examination was to make sure the single fault results. Another goal was to have statements, if it is critical to run the car after some single faults. Fortunately double faults turned out to lead mainly to *idle performance* or *engine stop*. This result emphasises the relevance of single faults as the main safety criterion [see Fig. 3].

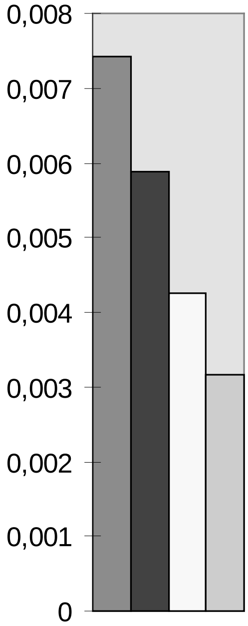

Fig. 3: Results for double faults (scale is linear, using arbitrary units)

7.3 Common-Mode Failures

Following common-mode failures lead to a *runaway*.

- Common pedal value error, e.g. if the pedal breaks.
- Common throttle value error. But this fault can be recognized by using the air mass sensor, maybe by other engine parameters.
- Common A/D converter error. The most likely cause is a faulty reference voltage.

- Common processor errors. They are mainly caused by software faults.

These failures can be avoided by a diversity approach.

8 Conclusions

The TEFT modeling method turned out to be able to scrutinize even complex systems, in order to make the systems safer by targeted architecture optimisation. This guided optimisation is a sound base to design systems efficiently and to save costs for unnecessary components, which can be an additional failure cause.

References

[1] Ajmone Marsan, M.; Balbo, G.; Conte, G.: "Performance Models of Multiprocessor Systems"; MIT-Press 1986

[2] DIN 25424: "Fehlerbaumanalyse. Methode und Bildzeichen". 1981 (part 1) and 1990 (part 2)

[3] DIN 25448: "Ausfalleffektanalyse (Fehler-Möglichkeits- und -Einfluß-Analyse)". 1990

[4] Jurgen, R.: "Automotive Electronics Handbook". McGraw-Hill 1995