RTS NOTES

Paper clippings

September 8, 2016

Contents

1 On the effective use of Fault injection for the assessment of AUTOSAR safety mechanism

1.1 Key Idea

AUTOSAR safety standard ISO26262 strongly recommends usage of the fault injection standards, but no definite mechanism exists for enforcing the same. Representation of the faults using the standard fault models e.g. bit flips and data type based corruption are not sufficient to model real time behavior. The existing Fault Injection framework GRINDER is extended to support AUTOSAR and an assessment is provided.

1.2 Width and scope

Provide open source and ready to use framework to do Fault Injection. Dependability assessment on existing AUTOSAR timing monitor safety mechanism for identifying deficiencies and guidelines for derivation of special fault models, injection mechanisms and locations based on abstract AUTOSAR and ISO26262 fault models. Eariler approach mentioned include: Lanigan et.al. and Baugarten et al. First approach being based on VECTOR CaNoe and second approach used anotated SWC Component to introduce fault ports. Another approach is pre implementation testing using UML or AADL, in this failure level are assumed and the effect on model analysed. Vedder et. al extended property based testing to AUTOSAR. Here automated test cases were generated from a pre specified property files.

Hardware based FI: Modeling hardware error such as CAN bus failure, or NVRAM failure through corrupted CRC. Hardware based FI fails to handle component interaction and dependability property. Software based FI: e.g. BeSafe framework to intercept SWC calls and fuzzing error models to check resilience of SWC. GOOFI-2 to evaluate bit flip cases and MODIFI to check model at Simulink level.

Note: Simultaneous fault models with multiple points of failure completely missing from AUTOSAR standards.

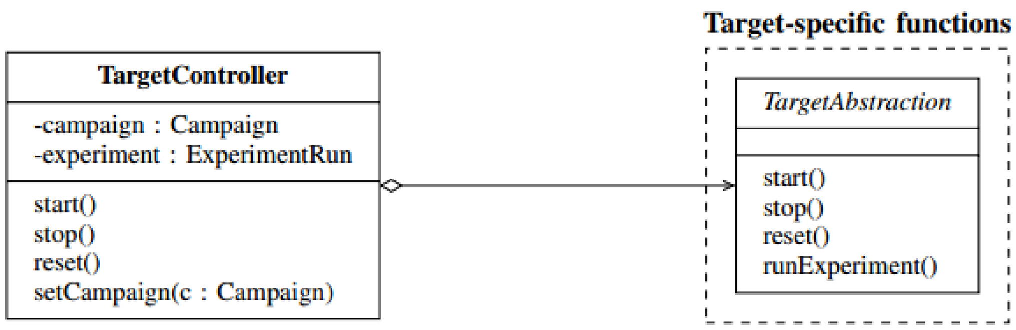


Figure 1: Abstract Target representation of GRINDER

1.3 Experimental approach

Configuration: Target is instrumented with injectors and detectors,. **Execution:** The target system is executed till the injection of fault and the perturbation data is successfully completed. **Evaluation:** The logs and traces are collected. GRINDER extended to AUTOSAR by providing a target specific implementation of TargetAbstraction Class with which GRINDER interacts during FI.

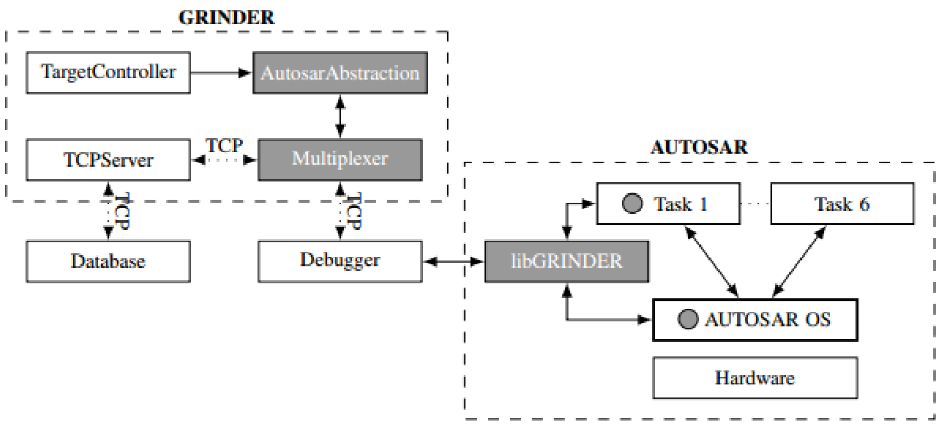


Figure 2: GRINDER new arch.

A case study is further done on Adaptive Cruise Control module.

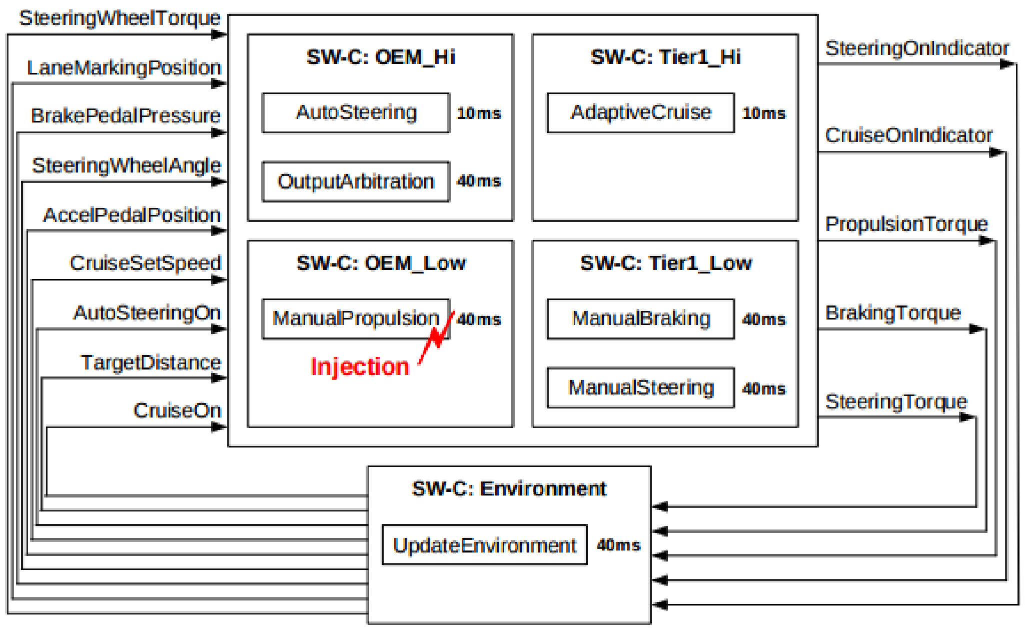
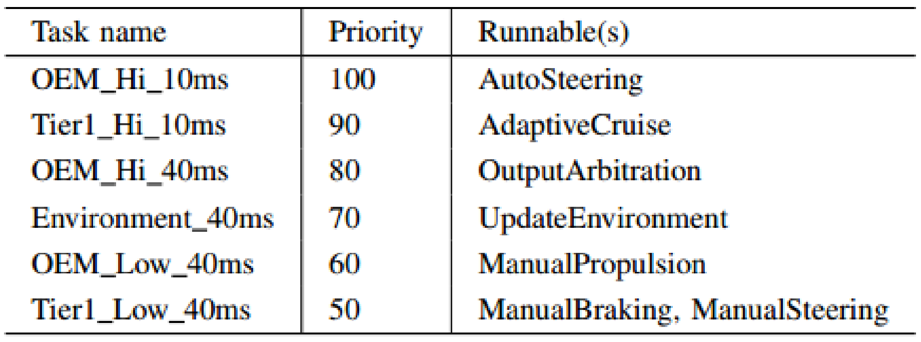


Figure 3: ABS Module architecture

Four different scenarios tested:

- Task timing error: Assess the correctness of the error detection and error mitigation of execution time monitoring. Analyze the propagation of the error with and without the presence of the timing errors.
- Interaction between the execution time monitoring and resource lock timing monitoring. Assess correctness and robustness of the mechanisms.

The task selection for monitoring ABS is shown below:



Task name	Priority	Runnable(s)
OEM_Hi_10ms	100	AutoSteering
Tier1_Hi_10ms	90	AdaptiveCruise
OEM_Hi_40ms	80	OutputArbitration
Environment_40ms	70	UpdateEnvironment
OEM_Low_40ms	60	ManualPropulsion
Tier1_Low_40ms	50	ManualBraking, ManualSteering

Figure 4: ACC task setup

1.4 conclusion

Detailed description of scenarios, A good framework to test the mechanism of fault injection and different failure scenarios. Can be used with possible change to JTAG interface to FTDI interface. Can be implemented as part of the eclipse plugin as as schedulability and testing mechanism.

1.5 Links

http://www1.deeds.informatik.tu-darmstadt.de/External/PublicationData/1/edcc-2015.pdf

2 Adaptive Runtime Shaping for Mixed-Criticality Systems

2.1 Key Idea

Presents an approach to adaptively shape the inflow workload of low-critical tasks based on actual demand of high critical tasks on runtime. Compared to the shaping of the offline bound low-critical tasks event delay is reduced and system utilization is improved.

2.2 Width and Scope

Main contributions are:

- An adaptive scheme for shaping the low critical workload.
- A light weight mechanism with complexity of *O(m.log(n))* to refine the shaping bound.
- Experimental results to show the efficiency.

This work build on three main approaches:

- Real-Time interface analysis: Connecting real time interface design and calculus.

- Workload prediction: Real time calculus models task activation as event, arrival curves originating from network calculus provide an upper and lower bound on number of arrival events. Prediction method based on historical arrival data. Arrival curve predicted by several stair case functions for tighter prediction.

- Runtime shaping: Shapers often used in regulating packets in network. Greedy mechanism for shaping in real time systems. FPGA based shaping mechanism.

2.3 Experimental Approach

Task generation using UUnifast mechanism. Experiments done on MATLAB Based on RTC Tool box(Theele et.al), mainly three different shaping mechanisms are compared.

- Shaping by offline computed bound.
- Shaping by backward derivation online.
- Shaping by proposed lightweight scheme.

2.4 Conclusion

TMD

2.5 Links

http://www6.in.tum.de/Main/Publications/Biao2015EMSOFT.pdf

3 Deadline Analysis of AUTOSAR OS Periodic Tasks in the Presence of Interrupts

3.1 Key Idea

Provide an abstract framework to determine if the given periodic task will miss its deadline in the presence of interrupts. Rather than bounding interrupts for given time. Proposes a mechanism to bind the timing calculation to maximum number of interruption allowed to a given task from interrupts.(e.g. The task will meet deadline as long as the number of interrupts is at most n.)

3.2 Width and Scope

Provides a mean to abstract representation of the maximum number of interruption allowed to a task. Assumptions part of the approach:

- Tasks are periodic and are implicit in nature with deadline equal to period.
- Any task activated by ISR2 will execute and terminate before the end of the ISR.
- Resource locking and blocking to the tasks due to it are not considered.

Mentions two independent healthiness concept: Task that is not interrupted and tasks that are interrupted.

$$\mathbf{TTI}(T_j, l) =_{df} \begin{cases} \lfloor \frac{l}{De(T_j)} \rfloor \times (ET(T_j) + IT(T_j)) + (ET(T_j) + IT(T_j)) \\ \qquad \text{if } \ l \bmod De(T_j) \geq (ET(T_j) + IT(T_j)) \\ \lfloor \frac{l}{De(T_j)} \rfloor \times (ET(T_j) + IT(T_j)) + l \bmod De(T_j) \\ \qquad \text{if } \ l \bmod De(T_j) < (ET(T_j) + IT(T_j)) \end{cases}$$

Figure 5: Timing Bound formula

3.3 Experimental Approach

Experiments done on Mathematica. Total slack available then split to interrupt times.

3.4 Conclusion

Calculate the slack, split it in terms of ISR time.That's main idea.

3.5 Link

http://haslab.uminho.pt/jff/files/2013-deadlineanalysisautosar_os.pdf

4 Mitigating Timing Error Propagation in Mixed-Criticality Automotive Systems

4.1 Key Idea

ISO26262 stipulates freedom from interferences, i.e, error should not propagate from low to high criticality tasks. Different from indirect protection of the critical tasks, approach provides direct low overhead protection to high critical tasks by introducing the concept of preemption budget.

4.2 Width and Scope

Introduces the concept of Preemption Budget(PB) and its specifies the maximum amount of time for which a critical task can be preempted. This is very much similar to the concept of Adaptive Mixed Criticality scheduling with deferred Preemption(AMC-DP), which specifies a minimum amount of time for which the task has to be run without any preemption. Timer based approach, For critical task that are active in the run queue timer is started to limit the earliest expiring preemption time. Once the preemption time is depleted the task is moved up the run queue and made to execute for rest of its budget. Handles Transient error runthrough (A mechanism similar to tolerence, where tasks are allowed to stretch the budget). Transient error run through is allowed under PB due to the slack available.

4.3 Experiments

Study of Transient and Permanent fault cases for ACC tasks. PBM Budget monitoring implemented under AUTOSAR, measurement made for static code size increase due to the implementation and the overhead incurred due to the budget monitoring.

4.4 Conclusion

An alternate take on AMC-DP and Tolerance limit based approach implemented and evaluated to show low overhead.

References