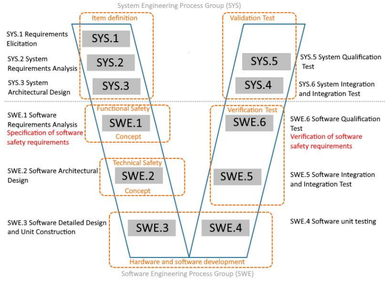
IDENTYFIKACJA ANOMALII WYWOŁUJĄCYCH FAŁSZYWIE NEGATYWNE WYNIKI W PROCESIE TESTOWANIA OPROGRAMOWANIA WBUDOWANEGO

Anna Gnacy-Gajdzik[1,2,3], Piotr Przystałka[1]

[1] Katedra Podstaw Konstrukcji Maszyn, Wydział Mechaniczny Technologiczny, Politechnika Śląska, ul. Konarskiego 18a, 44-100 Gliwice, [2] Szkoła Doktorów, Politechnika Śląska, ul. Akademicka 2a, 44-100 Gliwice , [3] DIP Draexlmaier Engineering Polska Sp. z o.o

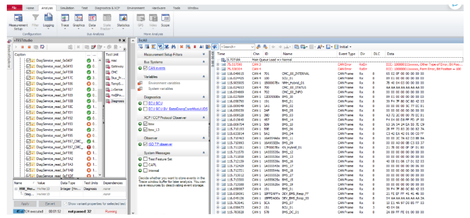
Streszczenie: W niniejszej pracy przeprowadzona została analiza testów dających wyniki migoczące wykonana na podstawie raportów z wykonanych eksperymentów oraz logów zawierających sygnały z magistral komunikacyjnych, wartości wielkości fizycznych mierzonych w symulacji HIL. Miała ona na celu identyfikację i kategoryzację anomalii występujących podczas wykonywania tych testów i wpływających na ich rezultat. Wyniki przedstawionej analizy zostaną wykorzystane w dalszych pracach badawczych, których celem jest opracowanie metodyki automatycznej weryfikacji negatywnych wyników testów układów wbudowanych wykorzystującej algorytmy rozszerzonej inteligencji.

Testy migoczące



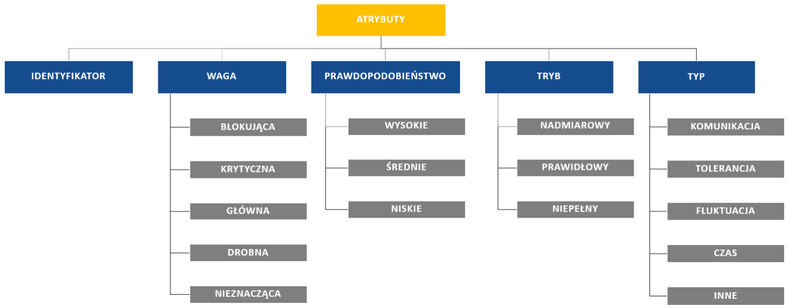
- Dają wyniki pozytywne oraz negatywne przy kilkukrotnym wykonaniu testu na tej samej wersji oprogramowania wbudowanego oraz w tym samym środowisku.
- Zgodnie ze standardem ASPICE, każdy przypadek testowy przed jego pierwszym oficjalnym wykonaniem jest sprawdzony w czasie przeglądu (ang. review). Przegląd dotyczy zarówno specyfikacji przypadku testowego, jego zgodności z wymaganiami, jak i implementacji. W przeglądzie biorą udział specjaliści z różnych dziedzin – inżynierowie systemowi, programiści, inżynierowie jakości oprogramowania wbudowanego, specjaliści ds. bezpieczeństwa funkcjonalnego.
- Również środowisko testowe - symulacje przygotowane przez inżynierów jakości oprogramowania wbudowanego podlegają przeglądom po każdej zmianie wprowadzonej w środowisku.

- Z przeprowadzonych badań wynika, że tendencje do migotania ma ponad 50% przebadanych przypadków testowych na poziomie SWE.6
- Analiza negatywnych wyników testów jest czasochłonna i kosztowna. Celem niniejszych badań była identyfikacja i kategoryzacja anomalii występujących podczas wykonywania testów i wpływających na ich rezultat.
- W dalszej perspektywie wyniki badań zostaną wykorzystane w opracowaniu metodyki tworzenia automatycznych testów systemów wbudowanych umożliwiającej zastosowanie algorytmów rozszerzonej inteligencji do weryfikacji poprawności ich wyników.

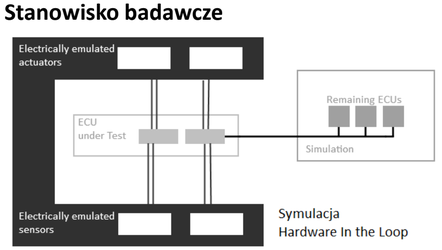


Klasyfikacja anomalii wg normy IEEE Std 1044-2009

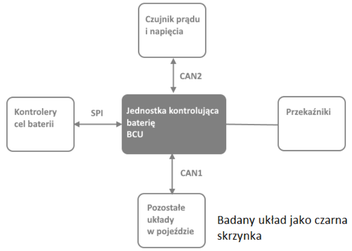
Wartości atrybutów wybrane do klasyfikacji zidentyfikowanych anomalii



Stanowisko badawcze



Symulacja Hardware In the Loop



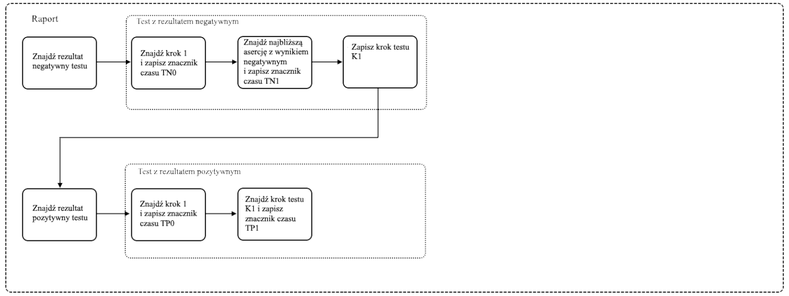
Badany układ jako czarna skrzynka

Metodyka prowadzonych badań

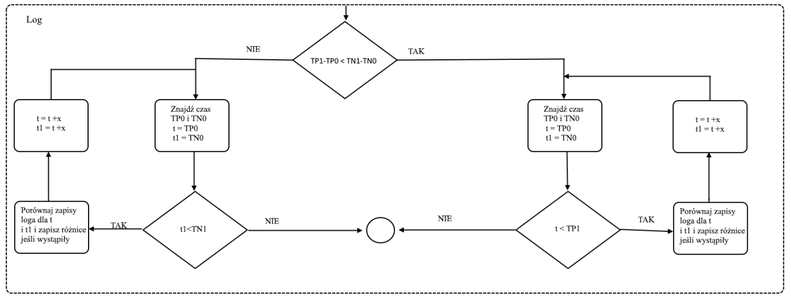
Wstępna faza badań

- Miała na celu wyodrębnienie wyników migoczących przeznaczonych do dalszej analizy.
- Wykonane zostało dziesięciokrotnie 216 przypadków testowych podzielonych na 18 modułów funkcjonalnych
- Testy w ramach modułu wykonywane były w losowej kolejności w celu uniezależnienia otrzymanych wyników od poprzednich.
- Otrzymano 74 wyniki zawsze pozytywne (co stanowi 34 %), 15 zawsze negatywne (co stanowi 7%) oraz 128 migoczących (59%).
- Do dalszych badań spośród 128 testów migoczących wybrano losowo 25.

Etap I – analiza raportów z wykonania testów



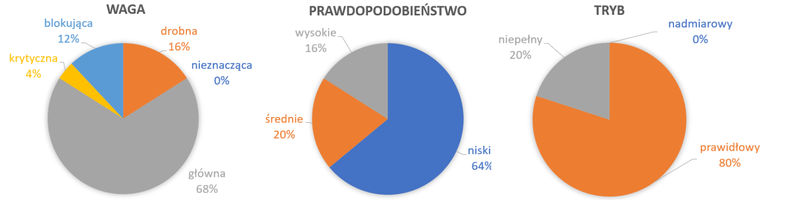
Etap II – analiza zapisu logów z wykonania testów



Rezultaty

Etap I – analiza raportów z wykonania testów

- Przeanalizowane zostały raporty z dziesięciokrotnego wykonania losowo wybranych 25 testów migoczących.
- Testom nadano identyfikatory – kolejne liczby całkowite.
- Określone zostały waga, prawdopodobieństwo oraz tryb.

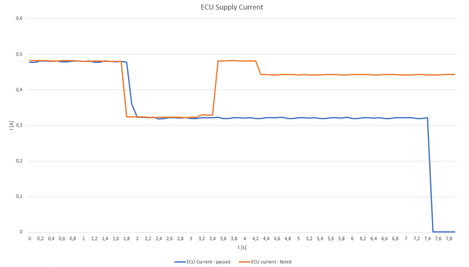


Etap II – analiza zapisu logów z wykonania testów

- Najbardziej czasochłonny oraz najtrudniejszy element analizy – określenie typu anomalii wpływającego na wynik negatywny testu
- Przykładowy test ID=12, jest to test z wysokim prawdopodobieństwem wystąpienia negatywnego rezultatu, wagą określoną jako główna (występująca anomalia nie pozwoliła na przetestowanie funkcji realizowanej przez ECU) – rozmiar pliku loga ~700MB, zawiera ponad 5 milionów linii z danymi zebranymi z symulacji dla 160 różnych wielkości fizycznych
- Dla wspomnianego testu określone w etapie I punkty TN0= 7.685262, TN1= 107.863262, TP0= 11.080579 i TP1= 18.631135, wyodrębniono odpowiednie fragmenty pliku do dalszej analizy.

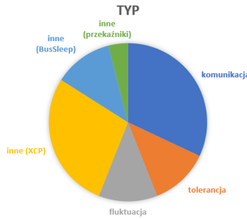
Wynik porównania prądu zasilania ECU dla testu ID=12

17.530564	HIL::fBCUSupplyCurrent	0.321118	0.441916
17.630561	HIL::fBCUSupplyCurrent	0.322425	0.442276
17.730558	HIL::fBCUSupplyCurrent	0.318917	0.442687
17.830555	HIL::fBCUSupplyCurrent	0.318923	0.443042
17.930552	HIL::fBCUSupplyCurrent	0.321828	0.442667
18.030548	HIL::fBCUSupplyCurrent	0.32173	0.442365
18.130545	HIL::fBCUSupplyCurrent	0.321333	0.441982
18.230542	HIL::fBCUSupplyCurrent	0.322124	0.442468
18.330539	HIL::fBCUSupplyCurrent	0.319906	0.442743
18.430536	HIL::fBCUSupplyCurrent	0.319991	0.44303
18.530533	HIL::fBCUSupplyCurrent	0.322106	0.442445
18.630530	HIL::fBCUSupplyCurrent	0.001029	0.442366



Wyniki analizy testów migoczących

Identyfikator	Opis	Waga	Prawdopodobieństwo	Tryb	Typ
1	ASF_006_CalM_Coding_Ident	drobna	niskie	prawidłowy	czas / komunikacja
2	SF_001_DiagServ_ContactorDiag	główna	wysokie	prawidłowy	tolerancja
3	SF_014_ErrHandl_CellVoltSafeLock	krytyczna	średnie	prawidłowy	komunikacja
4	SF_013_Obsrv_st_overcurr_AC_00	drobna	niskie	prawidłowy	komunikacja
5	SF_012_LvSense_KL30C_EqVerif	główna	niskie	prawidłowy	fluktuacja
6	SF_002_VehCan_NVEM02_Timeout	blokująca	niskie	niepełny	inne (XCP)
7	SF_003_ConIf_eConPickup	drobna	średnie	prawidłowy	Inne (BusSleep)
8	SF_001_DiagServ_CmcMeasurement	główna	niskie	prawidłowy	fluktuacja
9	SF_003_ConIf_PickUpVoltageEval	główna	wysokie	prawidłowy	Inne (BusSleep)
10	SF_004_BCK_ChargeMeasurement	główna	niskie	prawidłowy	fluktuacja
11	SF_002_VehCan__Airbag_Deact	główna	niskie	prawidłowy	inne (XCP)
12	SF_005_RTC_WakeUp_BalanOff	główna	wysokie	prawidłowy	inne (BusSleep)
13	SF_013_Obsrv_st_overcurr_DC_00	główna	niskie	prawidłowy	inne (przekaźniki)
14	SF_013_Obsrv_st_overcurr_DC_01	główna	niskie	prawidłowy	tolerancja
15	SF_001_DiagServ_BatteryMode	główna	średnie	niepełny	inne (XCP)
16	SF_001_DiagServ_Temp2_Meas	główna	niskie	niepełny	inne (XCP)
17	SF_001_DiagServ_CalibIdent	drobna	niskie	prawidłowy	inne (XCP)
18	SF_001_DiagServ_VehicleMeas	główna	niskie	niepełny	inne (XCP)
19	SF_001_LAD_CALID_Measurement	główna	niskie	niepełny	inne (XCP)
20	SF_007_Diag_SeccAcc_Download	główna	niskie	prawidłowy	komunikacja
21	SF_007_Diag_CheckProgramPrecon	blokująca	średnie	prawidłowy	komunikacja
22	SF_007_Diag_EraseFlash_NegResp	blokująca	niskie	prawidłowy	komunikacja
23	SF_008_DTCS_PosRespOn	główna	niskie	prawidłowy	komunikacja
24	SF_008_DTCS_BitTP_BlockSize	główna	wysokie	prawidłowy	komunikacja
25	SF_009_IsoMon_IsolationResistance	główna	średnie	prawidłowy	tolerancja



Wnioski: Analiza negatywnych rezultatów testów układów wbudowanych jest procesem skomplikowanym i czasochłonnym. Automatyzacja tego procesu, obejmująca przygotowanie plików logów do dalszej analizy przy pomocy skryptów napisanych np. w Pythonie jest pomocna, ale nie wystarczająca. W przypadku wspomnianego wyżej testu, pośród 160 wielkości występujących w pliku loga są napięcia i prądy, których fluktuacje zostaną wykazane jako różnice w przebiegu wykonania testu, natomiast w tym konkretnych przypadku nie mają wpływu na jego wynik. Zasadne wydaje się zastosowanie algorytmów rozszerzonej inteligencji w procesie dalszej analizy, kolejne badania będą prowadzone w kierunku optymalizacji tego procesu z ich zastosowaniem.

Politechnika Śląska
UCZELNIA BADAWCZA
INICJATYWA DOSKONAŁOŚCI

*Projekt współfinansowany przez Ministerstwo Edukacji i Nauki w ramach dotacji nr DWD/4/55/2020

XXX Sympozjon PKM OPOLE 2021