securityandtrust.lu

FAQAS Framework - DAMAt
Verification Report
ESAIL-ADCS Case Study

O. Cornejo, F. Pastore, E. Viganò

Interdisciplinary Centre for Security, Reliability and Trust

University of Luxembourg

ITT-1-9873-ESA-FAQAS-VR-ESAIL-AD

Issue 1, Rev. 1

November 11, 2021

UNIVERSITÉ DU LUXEMBOURG

EUROPEAN SPACE AGENCY. CONTRACT REPORT.
The work described in this report was done under ESA contract. Responsibility for the contents resides in the author or organisation that prepared it.

Contents

1 Executive Summary **3**

1.1 Terms, definitions and abbreviated terms 3

2 Configuration of the toolset **4**

2.1 Probe insertion 4

2.2 Fault Model 5

3 Results **9**

3.1 Metrics 9

3.2 Uncovered Fault Models 10

3.3 Uncovered Mutants 10

3.4 Live Mutants 10

Chapter 1

Executive Summary

This document is a report on the issues identified via Data-driven Mutation Analysis, on the ESAIL-ADCS case study using the *DAMAt* tool, which has been developed in the context of the FAQAS-framework.

DAMAt is a data-driven mutation analysis tool. It generates mutants by modiying the data contained in the SUT buffers throught the insertion in the source code of Mutation Probes. The mutation probes generate mutants following mutation operators defined in a Fault Model.

The code we analyzed was the ESAIL SVF test suite, which evaluetes the ESAIL OBSW and in particular the test cases regarding the interaction between the ADCS (Attitude Determination and Control System) component and the OBSW (On Board Software).

1.1 Terms, definitions and abbreviated terms

- FAQAS: activity ITT-1-9873-ESA
- FAQAS-framework: software system to be released at the end of WP4 of FAQAS
- SUT: Software under test, i.e, the software that should be mutated by means of mutation testing.

Chapter 2

Configuration of the toolset

2.1 Probe insertion

In the *ESAIL-ADCS case study* the mutation probes were inserterted in the function of the *ADCS_IF_SW* that manages the communication between the ADCS and the OBC, i.e., *ObcRecvBlockCb*. The function is implemented in the file *AdcsIf.c*.

ObcRecvBlockCb mainly consists of a switch command that generates a response for the OBC after invoking a data generation method selected according to the request received on the data link. For example, method *GetIfStatus* prepares a response packet containing the information about the ADCS status.

Each data generation method receive as input an object of type *std::vector* that will be used to store the data to be sent to the OBC. The vector is called *newBlock* and acts as a buffer and it contains elements of type UInt8.

Each invocation of a data generation method generates a response that may either contain the desired result or an error code. The response generated in the first case is referred to as nominal response message, the response generated in the second case is an error response message.

In both cases a *Mutation Probe* has been inserted to mutate the data contained in the buffer.

An example of this probe insertion strategy is reported in Listing 2.1.

```
1
2     if(Status->ADRD || ((cmdId == 1) && (subcmdId < 3)))
3     {
4         switch(cmdId)
5         {
6         case 1:
7         {
8             switch(subcmdId)
9             {
10            case 0:
11            {
12                cr = GetIfStatus(newBlock);
13
14                //MANUALLY INSERTED PROBES
15                if(cr != CR_Failure){
16
17                  mutate_FM_IfStatus(&newBlock);
18
19                }
20                //END PROBES
21
```

```
22              }
23              break;
```

Listing 2.1: Probe insertion Strategy

2.2 Fault Model

The Fault Model for the *ESAIL-ADCS case study* was defined in the *.csv* file format. It is reported in Table 2.1.

Table 2.1: Fault model for the *ESAIL-ADCS case study*

FaultModel	DataItem	Span	Type	FaultClass	Min	Max	Threshold	Delta	State	Value
IfStatus	0	1	BIN	BF	3	3	NA	NA	-1	1
IfStatus	0	1	BIN	BF	4	4	NA	NA	-1	1
IfStatus	0	1	BIN	BF	5	7	NA	NA	-1	1
IfStatus	1	1	BIN	BF	0	4	NA	NA	-1	1
IfStatus	4	1	BIN	BF	0	2	NA	NA	-1	1
IfStatus	4	1	BIN	BF	2	4	NA	NA	-1	1
IfStatus	4	1	BIN	BF	5	7	NA	NA	-1	1
IfStatus	5	1	BIN	BF	0	1	NA	NA	-1	1
IfStatus	5	1	BIN	BF	2	7	NA	NA	-1	1
IfHK	12	2	DOUBLE	VAT	NA	NA	3.6	0.1	NA	NA
IfHK	12	2	DOUBLE	FVAT	NA	NA	3.6	0.1	NA	NA
IfHK	14	2	DOUBLE	VAT	NA	NA	33.53	0.01	NA	NA
IfHK	14	2	DOUBLE	FVAT	NA	NA	33.53	0.01	NA	NA
IfHK	14	2	DOUBLE	VBT	NA	NA	24	1	NA	NA
IfHK	14	2	DOUBLE	FVBT	NA	NA	24	1	NA	NA
IfHK	24	2	DOUBLE	VAT	NA	NA	6	1	NA	NA
IfHK	24	2	DOUBLE	FVAT	NA	NA	6	1	NA	NA
IfHK	28	2	DOUBLE	VOR	-20	50	NA	1	NA	NA
IfHK	28	2	DOUBLE	FVOR	-20	50	NA	1	NA	NA
IfHK	30	2	DOUBLE	VOR	-20	50	NA	1	NA	NA
IfHK	30	2	DOUBLE	FVOR	-20	50	NA	1	NA	NA
IfHK	32	2	DOUBLE	VOR	-20	50	NA	1	NA	NA
IfHK	32	2	DOUBLE	FVOR	-20	50	NA	1	NA	NA
IfHK	34	2	DOUBLE	VOR	-20	50	NA	1	NA	NA
IfHK	34	2	DOUBLE	FVOR	-20	50	NA	1	NA	NA
IfHK	36	2	DOUBLE	VOR	-20	50	NA	1	NA	NA
IfHK	36	2	DOUBLE	FVOR	-20	50	NA	1	NA	NA
GYTM	0	1	BIN	BF	0	0	NA	NA	-1	1
GYTMFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x51
GYTMFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x52
GYTMFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x53
GYTMFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x54
GYTMFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x56
S.SensorTM	0	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	0	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	2	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	2	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	4	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	4	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	6	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	6	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	8	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	10	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	10	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	12	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	12	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA

S.SensorTM	14	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	16	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	16	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	18	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	18	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	20	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	20	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	22	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	22	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	24	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	24	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	26	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	26	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	28	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	28	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	30	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	30	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	32	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	32	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	34	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	34	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	36	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	36	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	38	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	38	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	40	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	40	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	42	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	42	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	44	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	44	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	46	2	DOUBLE	VAT	NA	NA	2.6	0.1	NA	NA
S.SensorTM	46	2	DOUBLE	FVAT	NA	NA	2.6	0.1	NA	NA
S.SensorTMFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x51
S.SensorTMFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x54
S.SensorTMFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x56
SSTP	0	2	DOUBLE	VOR	-70	100	NA	1	NA	NA
SSTP	0	2	DOUBLE	FVOR	-70	100	NA	1	NA	NA
SSTP	2	2	DOUBLE	VOR	-70	100	NA	1	NA	NA
SSTP	2	2	DOUBLE	FVOR	-70	100	NA	1	NA	NA
SSTP	4	2	DOUBLE	VOR	-70	100	NA	1	NA	NA
SSTP	4	2	DOUBLE	FVOR	-70	100	NA	1	NA	NA
SSTP	6	2	DOUBLE	VOR	-70	100	NA	1	NA	NA
SSTP	6	2	DOUBLE	FVOR	-70	100	NA	1	NA	NA
SSTP	8	2	DOUBLE	VOR	-70	100	NA	1	NA	NA
SSTP	8	2	DOUBLE	FVOR	-70	100	NA	1	NA	NA
SSTP	10	2	DOUBLE	VOR	-70	100	NA	1	NA	NA
SSTP	10	2	DOUBLE	FVOR	-70	100	NA	1	NA	NA
SSTPFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x51
SSTPFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x54
SSTPFail.	0	1	HEX	IV	NA	NA	NA	NA	NA	0x56
SpaceCraftHK	0	2	DOUBLE	VAT	NA	NA	3.3	0.1	NA	NA
SpaceCraftHK	0	2	DOUBLE	FVAT	NA	NA	3.3	0.1	NA	NA
SpaceCraftHK	0	2	DOUBLE	VBT	NA	NA	0	0.1	NA	NA
SpaceCraftHK	0	2	DOUBLE	FVBT	NA	NA	0	0.1	NA	NA
SpaceCraftHK	2	2	DOUBLE	VOR	0.5	2.75	NA	0.01	NA	NA
SpaceCraftHK	2	2	DOUBLE	FVOR	0.5	2.75	NA	0.01	NA	NA
SpaceCraftHK	4	2	DOUBLE	VOR	0	50	NA	1	NA	NA
SpaceCraftHK	4	2	DOUBLE	FVOR	0	50	NA	1	NA	NA
SpaceCraftHK	6	2	DOUBLE	VOR	0	50	NA	1	NA	NA
SpaceCraftHK	6	2	DOUBLE	FVOR	0	50	NA	1	NA	NA
SpaceCraftHK	8	2	DOUBLE	VOR	0	50	NA	1	NA	NA

SpaceCraftHK	8	2	DOUBLE	FVOR	0	50	NA	1	NA	NA
SpaceCraftHK	10	2	DOUBLE	VOR	0	50	NA	1	NA	NA
SpaceCraftHK	10	2	DOUBLE	FVOR	0	50	NA	1	NA	NA
SpaceCraftHK	12	2	DOUBLE	VOR	9.9253	29.9979	NA	0.0001	NA	NA
SpaceCraftHK	12	2	DOUBLE	FVOR	9.9253	29.9979	NA	0.0001	NA	NA
SpaceCraftHK	14	2	DOUBLE	VOR	9.9253	29.9979	NA	0.0001	NA	NA
SpaceCraftHK	14	2	DOUBLE	FVOR	9.9253	29.9979	NA	0.0001	NA	NA
M.SetPWMRSP	0	1	BIN	BF	0	0	NA	NA	0	1
M.SetPWMRSP	16	2	DOUBLE	VOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	16	2	DOUBLE	FVOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	18	2	DOUBLE	VOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	18	2	DOUBLE	FVOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	20	2	DOUBLE	VOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	20	2	DOUBLE	FVOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	22	2	DOUBLE	VOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	22	2	DOUBLE	FVOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	24	2	DOUBLE	VOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	24	2	DOUBLE	FVOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	26	2	DOUBLE	VOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	26	2	DOUBLE	FVOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	28	2	DOUBLE	VOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	28	2	DOUBLE	FVOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	30	2	DOUBLE	VOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	30	2	DOUBLE	FVOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	32	2	DOUBLE	VOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	32	2	DOUBLE	FVOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	34	2	DOUBLE	VOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	34	2	DOUBLE	FVOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	36	2	DOUBLE	VOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	36	2	DOUBLE	FVOR	0.14	0.21	NA	0.01	NA	NA
M.SetPWMRSP	38	2	DOUBLE	VOR	0	0.2	NA	0.1	NA	NA
M.SetPWMRSP	38	2	DOUBLE	FVOR	0	0.2	NA	0.1	NA	NA

Every line of the file represents a mutation operator, while every column represents a configuration parameter for that operator.

- Column *FaultModel* contains the name of the Fault Model containing the operator. Typically the user shall define a fault model for every different kind of message exchanged through the buffer.
- Column *DataItem* refers to the index of the first elemente of the targeted data item in the array representing the buffer.
- Column *Span* reports the number of array elements that make up the data item target by the mutation.
- Column *Type* reportts about the type of data targeted by the mutation: `INT`, `LONG`, `FLOAT`, `DOUBLE`, `BIN` or `HEX`.
- Column *FaultClass* contains the type of fault that the mutation will emulate, depending on the chosen mutation operator. A summary of the mutation operators can be found in Table 2.2.
- The other columns represent configuration parameters and assume different meanings depending on the mutation operator they refer to. More details on the data-driven mutation operators and their configuration can be found in Table 2.2.

A mutation operator can generate one or more mutants performing a *Mutation Operation*.

Table 2.2: Data-driven mutation operators

Fault Class	Types	Parameters	Description
Value above threshold (VAT)	I,L,F,D,H	T: threshold Δ: delta, difference with respect to threshold	Replaces the current value with a value above the threshold T for a delta (Δ). It simulates a value that is out of the nominal case and shall trigger a response from the system that shall be verified by the test case (e.g., the system may continue working but an alarm shall be triggered). Not applied if the value is already above the threshold. ***Data mutation procedure:*** $v' = (T + \Delta)(if v \leq T); v' = v(otherwise);$
Value below threshold (VBT)	I,L,F,D,H	T: threshold Δ: delta, difference with respect to threshold	Replaces the current value with a value below the threshold T for a delta (Δ). It simulates a value that is out of the nominal case and shall trigger a response from the system that shall be verified by the test case (e.g., the system may continue working but an alarm shall be triggered). Not applied if the value is already below the threshold. ***Data mutation procedure:*** $v' = (T - \Delta)(if v \geq T); v' = v(otherwise)$
Value out of range (VOR)	I,L,F,D,H	MIN: minimum valid value MAX: maximum valid value Δ: delta, difference with respect to minimum/maximum valid value	Replaces the current value with a value out of the range $[MIN; MAX]$. It simulates a value that is out of the nominal range and shall trigger a response from the system that shall be verified by the test case (e.g., the system may continue working but an alarm shall be triggered). Not applied if the value is already out of range. ***Data mutation procedure 1:*** $v' = (MIN - \Delta)(if MIN \leq v \leq MAX); v' = v(otherwise)$ ***Data mutation procedure 2:*** $v' = (MAX + \Delta)(if MIN \leq v \leq MAX); v' = v(otherwise)$
Bit flip (BF)	B	MIN: lower bit MAX: higher bit STATE: mutate only if the bit is in the given state (i.e., 0 or 1). VALUE: number of bits to mutate	A number of bits randomly chosen in the positions between MIN and MAX (included) are flipped. If STATE is specified, the mutation is applied only if the bit is in the specified state; the value -1 indicates that any state shall be considered for mutation. Parameter VALUE specifies the number of bits to mutate. ***Data mutation procedure:*** the operator flips VALUE randomly selected bit if they are in the specified state.
Invalid numeric value (INV)	I,L,F,D,H	MIN: lower valid value MAX: higher valid value	Replace the current value with a mutated value that is legal (i.e., in the specified range) but different than current value. It simulates the exchange of data that is not consistent with the state of the system. ***Data mutation procedure:*** Replace the current value with a different value randomly sampled in the specified range.
Illegal Value (IV)	I,L,F,D,H	VALUE: illegal value that is observed	Replace the current value with a value that is equal to the parameter *VALUE*. ***Data mutation procedure:*** $v' = VALUE(if v \neq VALUE); v' = v(otherwise)$
Anomalous Signal Amplitude (ASA)	I,L,F,D,H	T: change point Δ: delta, value to add/remove VALUE: value to multiply	The mutated value is derived by amplifying the observed value by a factor *V* and by adding/removing a constant value Δ from it. It is used to either amplify or reduce a signal in a constant manner to simulate unusual signals. The parameter *T* indicates the observed value below which instead of adding we subtract . ***Data mutation procedure:*** $v' = T + ((v - T) * VALUE) + \Delta(if\ v \geq T); v' = T - ((T - v) * VALUE) - \Delta(if\ v < T);$
Signal Shift (SS)	I,L,F,D,H	Δ: delta, value by which the signal should be shifted	The mutated value is derived by adding a value Δ to the observed value. It simulates an anomalous shift in the signal. ***Data mutation procedure:*** $v' = v + \Delta$
Hold Value (HV)	I,L,F,D,H	V: number of times to repeat the same value	This operator keeps repeating an observed value for V times. It emulates a constant signal replacing a signal supposed to vary. ***Data mutation procedure:*** $v' = previous\ v'(if\ counter \leq V); v' = votherwise$
Fix value above threshold (FVAT)	I,L,F,D,H	T: threshold Δ: delta, difference with respect to threshold	It is the complement of VAT and implements the same mutation procedure as VBT but we named it differently because it has a different purpose. Indeed, it is used to verify that test cases exercising exceptional cases are verified correctly. In the presence of a value above the threshold, it replaces the current value with a value below the threshold T for a delta Δ. ***Data mutation procedure:*** $v' = v(if v > T); V' = (T - \Delta)(otherwise)$
Fix value below threshold (FVBT)	I,L,F,D,H	T: threshold Δ: delta, difference with respect to threshold	It is the counterpart of FVAT for the operator VBT. ***Data mutation procedure:*** $v' = v(if v < T); v' = (T + \Delta)(otherwise)$
Fix value out of range (FVOR)	I,L,F,D,H	MIN: minimum valid value MAX: maximum valid value	It is the complement of VOR and implements the same mutation procedure as INV but we named it differently because it has a different purpose. Indeed, it is used to verify that test cases exercising exceptional cases are verified correctly. ***Data mutation procedure:*** $v' = v(if MIN \leq v \leq MAX); v' = random(MIN, MAX)(otherwise)$

Legend: I: INT, L: LONG INT, F: FLOAT, D: DOUBLE, B: BIN, H: HEX

Chapter 3

Results

3.1 Metrics

Starting from the Fault Model represented in Table 2.1, *DAMAt* generated 666 mutants. The mutated version of the program were executed against the *ESAIL-ADCS case study* test suite.

The results were expressed with the following three metrics:

1. Fault model coverage, the percentage of fault models covered by the test suite.

2. Mutation operation coverage, the percentage of data items that have been mutated at least once, considering only those that belong to the data buffers covered by the test suite.

3. Mutation Score, the percentage of mutants killed by the test suite (i.e., leading to at least one test case failure) among the mutants that target a fault model and for which at least one mutation operation was successfully performed.

A low score in one of the metrics indicate one of following scenarios, repsectively:

1. The message type targeted by a fault model is never exercised.

2. The message type is covered by the test suite, but it is not possible to perform some of the mutation operations. It depends on the fact that not all the input partitions are exercised by the test suite.

3. The mutation is performed but the test suite does not fail. It may depend on two reasons: (1) the test oracles are imprecise (e.g., they do not verify all the state variables), (2) the system is not brought into a state where the effect of the mutation is notices (i.e., the scenarios exercised are insufficient).

The results for the *ESAIL-ADCS case study* are reported in Table

3.2 Uncovered Fault Models

3.3 Uncovered Mutants

3.4 Live Mutants