Hochschule Ruhr West
Institut für Naturwissenschaften
Sicherheitstechnik - Funktionale Sicherheit
i.A. von Prof. Dr.-Ing. David Schepers

Project „Safe Temperature Monitoring “

Software system test plan

Modification history:

Version	Date	Modification	Creator	Auditor
1.0	2020-09-28	First Version	SaS	DS

following document: SW-Integration test plan

Content

1 Software – System Test Plan

1.1 Verification of safety function „Safe Temperature Monitoring “

1.1.1 Test Management und Automation Tools [SW-STP1]

The procedure for test management and automation tools is recommended for SIL 1 or higher levels, especially for SIL 3 or higher. If the procedure is not implemented for SIL 3 or higher, a plausible explanation must be given.

The procedure prescribes the use of appropriate and supporting tools.

Tool No.	Type of the tool	Provider	Tool Name	Version	Classification
1	UT tool	QA-Systems	Cantata	2019-12	

"Classification": Definition, if the tool correspond to T3, T2 or T1 (IEC 61508-3, 7.4.4)

- Reason for not implementing: The Unit testing of the safety function “Safe Temperature monitoring” is not tested with test automation methods since the project scope is not huge and manual testing method is still convenient to develop tests.

1.1.2 Statistical Tests [SW-STP2]

The statistical test procedure is recommended for SIL 2 or higher. If the procedure is not implemented, a plausible explanation must be provided.

The statistical tests are used to estimate the software reliability. For this purpose, x randomly generated test cases are generated once within the limits and once with values outside the limits. These software runs through all defined tests and then an output of success or failure of the tests is generated for each test.

Tests within the limits: Test case “Celsius”

Number of tests:	3
Random variables:	TempS1, TempS2
If necessary: Definition of maximum permissible delta between random variables	
Upper limit:	1000°C
Lower limit:	-273.15°C

Tests with values outside the limits:

Number of tests:	3
Random variables:	TempS1, TempS2
If necessary: Definition of maximum permissible delta between random variables:	
Upper limit:	1000°C
Lower limit:	-273.15°C

Tests within the limits: Test Case “Fahrenheit”

Number of tests:	3
Random variables:	TempS1, TempS2
If necessary: Definition of maximum permissible delta between random variables:	
upper limit:	1832°F
lower limit:	-459°F

Tests with values outside the limits:

Number of tests:	3
Random variables:	TempS1, TempS2
If necessary: Definition of maximum permissible delta between random variables:	
Upper limit:	1832°F
Lower limit:	-459°F

1.1.3 Dynamic analysis and tests

The procedure of dynamic analysis and tests is recommended for SIL 1or higher, especially for SIL 2 or higher. If the procedure is not implemented for SIL 2 or, a plausible explanation must be given.

The dynamic analysis and tests include several test cases:

1.1.3.1 Performing of boundary tests (boundary value analysis) **[SW-STP3]**

Performing boundary tests is recommended for SIL 1 or higher, especially for SIL 2 or higher. If the procedure is not implemented for SIL 2 or higher, a plausible justification must be given.

After determining the boundary values, test cases can be generated using a truth tables.

Boundary values:		Celsius	
Upper limit:	1000°C	**Lower limit:**	-273.15°C

Lower limit: Test Case „Celsius“

Truth table			
Function or fault insertion test	**Expectation**	**Test Result**	**Acceptance**
monitorTemp_T001	Function error		✓
monitorTemp_T002	Function error		×
monitorTemp_T003	Function error		✓

Upper limit: Test Case „Celsius“

Truth table			
Function or fault insertion test	**Expectation**	**Test Result**	**Acceptance**
monitorTemp_T004	Function error		✓
monitorTemp_T005	Function error		×
monitorTemp_T006	Function error		✓

Boundary values:		Fahrenheit	
Upper limit:	1832°F	**Lower limit:**	-459°F

Lower limit:

Truth table			
Function or fault insertion test	**Expectation**	**Test Result**	**Acceptance**
monitorTemp_T007	Function error		✓
monitorTemp_T008	Function error		×
monitorTemp_T009	Function error		✓

Upper limit:

Truth table			
Function or fault installation test	**Expectation**	**Test Result**	**Acceptance**
monitorTemp_T010	Function error		✓
monitorTemp_T011	Function error		×
monitorTemp_T012	Function error		✓

1.1.3.2 Performing test cases based on fault expectations **[SW-STP4]**

The execution of test cases based on fault expectations is recommended for SIL 1 or higher. If the procedure is not implemented, a plausible justification must be given.

To generate test cases from fault expectations, pathological cases and the expected operator's curiosity are used to generate exceptional test cases using a truth table.

Truth table			
Function or fault insertion tests	**Expectation**	**Test Result**	**Acceptance**
monitorTemp_T013	Function error		✓
monitorTemp_T014	Function error		×

1.1.3.3 Performing fault insertion tests **[SW-STP5]**

The performance of fault insertion tests is recommended for SIL 2 or higher. If the procedure is not implemented, a plausible justification must be given.

To generate the test, known or possible failures are introduced in the software and the behavior is evaluated using a truth table.

Celsius

Truth table			
Function or fault insertion tests	**Expectation**	**Test Result**	**Acceptance**
monitorTemp_T015	Function error		✓
monitorTemp_T016	Function error		×

Fahrenheit

Truth table			
Function or fault insertion tests	**Expectation**	**Test Result**	**Acceptance**
monitorTemp_T017	Function error		✓
monitorTemp_T018	Function error		×

1.1.4 Structure-based testing (code coverage testing) [SW-STP6]

Performing the structure-dependent tests is recommended for SIL 1 or higher, especially for SIL 2 or higher. If the procedure is not implemented for SIL 2 or higher, a plausible explanation must be provided.

1.1.4.1 Entry point (call graph) coverage (100%)

Performing the structure-based test of entry point coverage is particularly recommended for SIL 1 or higher. If the procedure is not implemented, a plausible explanation must be given.

Each subroutine of the software has to be called at least once.

- Reason for not implementing:

1.1.4.2 Statement coverage (100%)

Performing the structure-based test of statement covering is recommended for SIL 1 or higher, especially for SIL 2 or higher. If the procedure is not implemented for SIL 2 or higher, a plausible explanation must be given.

Each statement in the source code has to be called at least once.

1.1.4.3 Branch coverage (100%)

It is recommended to perform a structure-based test for branch coverage for SIL 1 or higher, especially for SIL 3. If the procedure is not implemented for SIL 3 or higher, a plausible explanation must be provided.

Both sides of every branch should be checked. (This may be impractical for some types of defensive code).

1.1.4.4 Condition coverage (100%)

It is recommended to perform a structure-based test for SIL 1 or higher, especially for SIL 4. If the procedure is not implemented for SIL 4, a plausible explanation must be provided.

Every condition in a compound conditional branch (i.e. linked by

AND/OR) is exercised.

1.1.5 Functional/White Box and Black-Box-Testing

Performing the functional/white box testing and black box testing is recommended for SIL 1 or higher, especially for SIL 2 or higher. If the procedure is not implemented for SIL 2, a plausible explanation must be given.

[SW-STP7]

During functional/white box testing, one or more test cases are generated from the properties defined in the specification of the (sub-)function (e.g. limit values).

Function:

Properties of the function:

Test cases:

Truth table			
Function or fault insertion tests	**Expectation**	**Test Result**	**Acceptance**
			✓(for copying)
			×(for copying)

Function:

Properties of the function:

Test cases:

Truth table			
Function or fault insertion tests	**Expectation**	**Test Result**	**Acceptance**
			✓(for copying)
			×(for copying)

- Reason for not implementing: White box testing is not required for this project

[SW-STP8]

In black box testing, the function is executed in its specified environment with specified test data.

Environmental conditions:

Test data:

Test with permissible range

Expectation:

Parameters:	format	min_temp	max_temp	max_delta	temp_sensor1	temp_sensor2	Result
Test values	C	-200	800	25	56	90	3
Test values	C	-100	600	100	70	170	7
Test values	F	-300	350	154	97	500	7
Test values	F	-420	500	187	150	155	3

Test with improper range

Expectation:

Parameters:	format	min_temp	max_temp	max_delta	temp_sensor1	temp_sensor2	Result
Test values	C	-1000	8000	25	56	90	5
Test values	C	-5000	1000	100	70	2000	5
Test values	F	-3500	3500	154	97	500	5
Test values	F	-4420	500	187	1500	155	5

Test at range limits

expectation:

Parameters:	format	min_temp	max_temp	max_delta	temp_sensor1	temp_sensor2	Result
Test values	C	-273.15	1000	0	1000	1000	7
Test values	F	-459	1832	0	1832	1832	7

Test at range limits:

Expectation:

Parameters:	format	min_temp	max_temp	max_delta	temp_sensor1	temp_sensor2	Result
Test values	C	-273.15	1000	0	1000	1000	7
Test values	F	-459	1832	0	1832	1832	7

Test with exceeding delta:

Expectation:

Variables:	format	min_temp	max_temp	max_delta	temp_sensor1	temp_sensor2	Result
Test values	C	-273.15	1000	0	1001	1000	3
Test values	F	-459	1832	0	1833	1832	3

1.1.6 Performance testing [SW-STP9]

Performance testing is recommended for SIL 1 or higher, especially for SIL 3 or higher. If the procedure is not implemented for SIL 3 or higher, a plausible explanation must be provided.

1.1.6.1 Stress testing

Performing stress testing is recommended for SIL 1 or higher, especially for SIL 3 or higher. If the procedure is not implemented for SIL 3 or higher, a plausible explanation must be provided.

To burden the test object with an exceptionally high workload in order to show that the test object would stand normal workloads easily. There are a variety of test conditions which can be applied for avalanche/stress testing. Some of these test conditions are:

- if working in a polling mode then the test object gets much more input changes per time unit as under normal conditions;
- if working on demands then the number of demands per time unit to the test object is increased beyond normal conditions;
- if the size of a database plays an important role then it is increased beyond normal conditions;
- influential devices are tuned to their maximum speed or lowest speed respectively;
- for the extreme cases, all influential factors, as far as is possible, are put to the boundary conditions at the same time.

Under these test conditions, the time behaviour of the test object can be evaluated. The influence of load changes can be observed. The correct dimension of internal buffers or dynamic variables, stacks, etc. can be checked.

- Reason for not implementing: Stress testing is not applicable for this project.

1.1.6.2 Response timing and memory constraints

The procedure of response timing and memory constraints is particularly recommended for SIL 1 or higher. If the procedure is not implemented, a plausible explanation must be described.

By this procedure, it shall be ensured that the system will meet its temporal and memory requirements. The requirements specification for the system and the software includes memory and response requirements for specific functions, perhaps combined with constraints on the use of total system resources.

- Reason for not implementing: Memory constraint tests is not applicable for this project

1.1.7 Model based testing [SW-STP10]

Performing model-based testing is recommended for SIL 1 or higher, especially from SIL 3. If the procedure is not implemented for SIL 3 or higher, a plausible explanation must be provided.

In model-based testing, a test model is created from the requirements of the specification. During the process, expectations are generated, for example with the help of decision tables or finite state machines, and compared with the real outputs of the software.

- Reason for not implementing: Model based testing is not required for this project since the implementation is done in C language.

1.1.8 Interface testing

Performing interface testing is recommended for SIL 1 or higher, especially for SIL 3 or higher. If the procedure is not implemented for SIL 3 or higher, a plausible explanation must be provided.

Shall be performed with the help of the condition checking template and all interfaces of the software.

- Reason for not implementing: Interface testing is not required for this project since it the safety function is implemented as a hardware independent library.

1.1.9 Data recording and analysis

The procedure of data recording and analysis is particularly recommended for SIL 1 or higher. If the procedure is not implemented, a plausible explanation must be provided.

Documentation of all data, decisions and rationale in the software project to allow for easier verification, validation, assessment and maintenance.

Detailed documentation is maintained during a project, which could include

– testing performed on each software module;

– decisions and their rationale;

– problems and their solutions.

During and at the conclusion of the project this documentation can be analyzed to establish a wide variety of information. In particular, data recording is very important for the maintenance of computer systems as the rationale for certain decisions made during the development project is not always known by the maintenance engineers.

Results of the test:

- Reason for not implementing: Interface testing is not required for this project since it the safety function is implemented as a hardware independent library.
-

1.1.10 Traceability

Applying traceability is recommended for SIL 1 or higher, especially for SIL 3. If the procedure is not implemented for SIL 3, a plausible justification must be provided.

In this procedure, all requirements are checked by forward and backward traceability throughout all documents.

Result:

1.1.11 Formal verification

The formal verification procedure is particularly recommended for SIL 3 or higher. If the procedure is not implemented, a plausible justification must be given.

In the formal verification procedure, all operators are replaced by mathematical representations to check the defined behavior. For this purpose, an abstract model of the source code is needed.

2.0 Templates for test plans

2.1 Truth table

Truth table			
Function or fault insertion tests	**Expectation**	**Test Result**	**Acceptance**
			✓
			×

If the result of the test does not match the expectation, a "procedure in case of not passed" must be specified.

2.2 Checklist

Checklist	
Measures	**Acceptance**
	✓
	×

If the measures have not been met, a "procedure in case of not passed" must be specified.

2.3 Comparison

Comparison (target value > real value)		
Target value	**Real value**	**Δ**

If the real value is higher than target value, a „procedure in case of not passed“ must be specified.

Comparison (real value > target value)		
Target value	**Real value**	**Δ**

If the target value is higher than the real value, a „ procedure in case of not passed “ must be specified.

2.4 Procedure in case of not passed

- Analyze the deviation by a suitable method.
- Determine the severity of the deviation.
- Check whether the value to be achieved is realistic.
- Derive measure and carry out software modification if necessary.
- The complete procedure must be documented in detail and attached to the documents.

3.0 Note

This template is **not considered complete, but applicable**. The template contains the requirements of IEC 61508 for the development of software functions. The template is to be regarded as alive, i.e. during the development process, possibly issues may be discovered, which are not contained in the template. These should be added to the template at the correct place, in order to be able to consider them during next use. The more diverse software is created by means of this template, the more completely and exactly can the development process for software be realized in the future.