– **single channel with periodic self-test and monitoring** (H.2.16.7);
– **dual channel (homogenous) with comparison** (H.2.16.3);
– **dual channel (diverse) with comparison** (H.2.16.2).

NOTE Comparison between **dual channel** structures can be performed:

– by the use of a **comparator** (H.2.18.3) or

– by **reciprocal comparison** (H.2.18.15).

H.11.12.1.2.2 Control functions with software class B shall have one of the following structures:

– **single channel with functional test** (H.2.16.5);
– **single channel with periodic self-test** (H.2.16.6);
– **dual channel** without comparison (H.2.16.1).

A software class C structure is also acceptable for a software class B structure.

H.11.12.1.3 Other structures are permitted if they can be shown to provide an equivalent level of safety to those in H.11.12.1.2.

H.11.12.2 Measures to control faults/errors

H.11.12.2.1 When **redundant memory with comparison** is provided on two areas of the same component, the data in one area shall be stored in a different format from that in the other area (see **software diversity**).

H.11.12.2.2 Controls with software class C using **dual channel** structures with comparison shall have additional **fault**/error detection means (such as periodic functional tests, periodic self-tests, or **independent** monitoring) for any **fault**/errors not detected by the comparison.

H.11.12.2.3 For **controls** with software class B or C, means shall be provided for the recognition and control of errors in **transmissions** to external safety-related data paths. Such means shall take into account errors in data, addressing, **transmission** timing and sequence of protocol.

H.11.12.2.4 For **control** with software class B or C, the manufacturer shall provide, within the control, measures to address the **fault**/errors in safety-related segments and data indicated in Table H.1 and identified in Table 1, requirement 68.

Table H.1 (H.11.12.7 of edition 3) – Acceptable measures to address fault/errors [a] *(1 of 6)*

Component [b]	Fault/error	Software class		Example of acceptable measures [c d e]	Definitions
		B	C		
1. CPU 1.1 Registers	Stuck at	rq		Functional test, or periodic self-test using either: – **static memory test**, or – **word protection with single bit redundancy**	H.2.16.5 H.2.16.6 H.2.19.6 H.2.19.8.2
	DC **fault**		rq	Comparison of redundant CPUs by either: – **reciprocal comparison** – **independent** hardware **comparator**, or **internal error detection**, or **redundant memory with comparison**, or periodic self-tests using either – **walkpat memory test** – **Abraham test** – **transparent GALPAT test**; or **word protection with multi-bit redundancy**, or **static memory test** and word protection with single bit redundancy	 H.2.18.15 H.2.18.3 H.2.18.9 H.2.19.5 H.2.19.7 H.2.19.1 H.2.19.2.1 H.2.19.8.1 H.2.19.6 H.2.19.8.2
1.2 Instruction decoding and execution	Wrong decoding and execution		rq	Comparison of redundant CPUs by either: – **reciprocal comparison** – **independent** hardware **comparator**, or **internal error detection**, or periodic self-test using **equivalence class test**	 H.2.18.15 H.2.18.3 H.2.18.9 H.2.18.5
1.3 Programme counter	Stuck at	rq		Functional test, or periodic self-test, or **independent time-slot monitoring of the program sequence**, or **logical monitoring of the programme sequence**	H.2.16.5 H.2.16.6 H.2.18.10.4 H.2.18.10.2
	DC **fault**		rq	Periodic self-test and monitoring using either: **– independent time-slot and logical monitoring** – **internal error detection**, or comparison of redundant functional channels by either: – **reciprocal comparison** – **independent** hardware **comparator**	H.2.16.7 H.2.18.10.3 H.2.18.9 H.2.18.15 H.2.18.3

Table H.1 *(2 of 6)*

Component [b]	Fault/error	Software class		Example of acceptable measures [c d e]	Definitions
		B	C		
1.4 Addressing	DC **fault**		rq	Comparison of redundant CPUs by either: – **reciprocal comparison** – **independent** hardware **comparator**; or **Internal error detection**; or periodic self-test using a **testing pattern** of the address lines; or **full bus redundancy**, or **multi-bit bus parity**	 H.2.18.15 H.2.18.3 H.2.18.9 H.2.16.7 H.2.18.22 H.2.18.1.1 H.2.18.1.2
1.5 Data paths instruction decoding	DC **fault** and execution		rq	Comparison of redundant CPUs by either: **reciprocal comparison**, or **independent** hardware **comparator**, or **Internal error detection**, or periodic self-test using a **testing pattern**, or **data redundancy**, or **multi-bit bus parity**	 H.2.18.15 H.2.18.3 H.2.18.9 H.2.16.7 H.2.18.2.1 H.2.18.1.2
2. Interrupt handling and execution	No interrupt or too frequent interrupt	rq		Functional test; or time-slot monitoring	H.2.16.5 H.2.18.10.4
	No interrupt or too frequent interrupt related to different sources		rq	Comparison of redundant functional channels by either **reciprocal comparison**, **independent** hardware **comparator**, or **Independent time-slot and logical monitoring**	 H.2.18.15 H.2.18.3 H.2.18.10.3

Table H.1 *(3 of 6)*

Component [b]	**Fault/error**	**Software class**		**Example of acceptable measures** [c d e]	**Definitions**
		B	**C**		
3. Clock		rq		**Frequency monitoring**, or time slot monitoring	H.2.18.10.1 H.2.18.10.4
	Wrong frequency (for quartz synchronized clock: harmonics/ subharmonics only)		rq	**Frequency monitoring**, or time-slot monitoring, or comparison of redundant functional channels by either: – **reciprocal comparison** – **independent** hardware **comparator**	H.2.18.10.1 H.2.18.10.4 H.2.18.15 H.2.18.3
4. Memory 4.1 **Invariable memory**	All single bit **faults**	rq		Periodic **modified checksum**; or **multiple checksum**, or **word protection with single bit redundancy**	H.2.19.3.1 H.2.19.3.2 H.2.19.8.2
	99,6 % coverage of all information errors		rq	Comparison of redundant CPUs by either: – **reciprocal comparison** – **independent** hardware **comparator**, or **redundant memory with comparison**, or periodic cyclic redundancy check, either – single word – double word, or **word protection with multi-bit redundancy**	 H.2.18.15 H.2.18.3 H.2.19.5 H.2.19.4.1 H.2.19.4.2 H.2.19.8.1
4.2 Variable memory	DC **fault**	rq		Periodic **static memory test**, or **word protection with single bit redundancy**	H.2.19.6 H.2.19.8.2
	DC **fault** and dynamic cross links		rq	Comparison of redundant CPUs by either: – **reciprocal comparison** – **independent** hardware **comparator**, or **redundant memory with comparison**, or periodic self-tests using either: – **walkpat memory test** – **Abraham test** – **transparent GALPAT test**, or **word protection with multi-bit redundancy**	 H.2.18.15 H.2.18.3 H.2.19.5 H.2.19.7 H.2.19.1 H.2.19.2.1 H.2.19.8.1

Table H.1 *(4 of 6)*

Component [b]	Fault/error	Software class		Example of acceptable measures [c d e]	Definitions
		B	C		
4.3 Addressing (relevant to **variable memory** and **invariable memory**)	Stuck at DC **fault**	rq	rq	**Word protection with single bit redundancy** including the address, or comparison of redundant CPUs by either: – **reciprocal comparison**, or – **independent** hardware **comparator**, or **full bus redundancy** **Testing pattern**, or periodic cyclic redundancy check, either: – single word – double word, or **word protection with multi-bit redundancy** including the address	H.2.19.18.2 H.2.18.15 H.2.18.3 H.2.18.1.1 H.2.18.22 H.2.19.4.1 H.2.19.4.2 H.2.19.8.1
5. Internal data path 5.1 Data	Stuck at DC **fault**	rq	rq	**Word protection with single bit redundancy** Comparison of redundant CPUs by either: – **reciprocal comparison** – **independent** hardware **comparator**, or **word protection with multi-bit redundancy** including the address, or **data redundancy**, or **testing pattern**, or **protocol test**	H.2.19.8.2 H.2.18.15 H.2.18.3 H.2.19.8.1 H.2.18.2.1 H.2.18.22 H.2.18.14
5.2 Addressing	Wrong address Wrong address and multiple addressing	rq	rq	**Word protection with single bit redundancy** including the address Comparison of redundant CPUs by: – **reciprocal comparison** – **independent** hardware **comparator**, or **word protection with multi-bit redundancy**, including the address, or **full bus redundancy**; or **testing pattern** including the address	H.2.19.8.2 H.2.18.15 H.2.18.3 H.2.19.8.1 H.2.18.1.1 H.2.18.22
6 External communication	**Hamming distance** 3	rq		**Word protection with multi-bit redundancy**, or **CRC – single word** , or **transfer redundancy**, or **protocol test**	H.2.19.8.1 H.2.19.4.1 H.2.18.2.2 H.2.18.14

Table H.1 *(5 of 6)*

Component [b]	**Fault/error**	**Software class**		**Example of acceptable measures** [c d e]	**Definitions**
		B	**C**		
6.1 Data	**Hamming distance** 4		rq	**CRC – double word**, or	H.2.19.4.2
				data redundancy or comparison of redundant functional channels by either:	H.2.18.2.1
				– **reciprocal comparison**	H.2.18.15
				– **independent** hardware **comparator**	H.2.18.3
6.2 Addressing	Wrong address	rq		**Word protection with multi-bit redundancy**, including the address, or **CRC – single word** including the addresses, or	H.2.19.8.1 H.2.19.4.1
				transfer redundancy or	H.2.18.2.2
				protocol test	H.2.18.14
	Wrong and multiple addressing		rq	**CRC – double word**, including the address, or	H.2.19.4.2
				full bus redundancy of data and address, or	H.2.18.1.1
				comparison of redundant communication channels by either:	
				– **reciprocal comparison**	H.2.18.15
				– **independent** hardware **comparator**	H.2.18.3
6.3 Timing	Wrong point in time	rq		Time-slot monitoring, or **scheduled transmission**	H.2.18.10.4 H.2.18.18
			rq	**Time-slot and logical monitoring**, or	H.2.18.10.3
				comparison of redundant communication channels by either:	
				– **reciprocal comparison**	H.2.18.15
				– **independent** hardware **comparator**	H.2.18.3
	Wrong sequence	rq		Logical monitoring, or	H.2.18.10.2
				time-slot monitoring, or	H.2.18.10.4
				scheduled transmission	H.2.18.18
			rq	(same options as for wrong point in time)	
7. Input/output periphery	**Fault** conditions specified in Clause H.27	rq		**Plausibility check**	H.2.18.13
			rq	Comparison of redundant CPUs by either:	
				– **reciprocal comparison**	H.2.18.15
				– **independent** hardware **comparator**, or	H.2.18.3
7.1 Digital I/O				**input comparison**, or	H.2.18.8
				multiple parallel outputs; or	H.2.18.11
				output verification, or	H.2.18.12
				testing pattern, or	H.2.18.22
				code safety	H.2.18.2

Table H.1 *(6 of 6)*

Component [b]	Fault/error	Software class		Example of acceptable measures [c d e]	Definitions
		B	C		
7.2 Analog I/O 7.2.1 A/D- and D/A- convertor	**Fault** conditions specified in Clause H.27	rq 	 rq	**Plausibility check** Comparison of redundant CPUs by either: – **reciprocal comparison** – **independent** hardware **comparator**, or **input comparison**, or **multiple parallel outputs**, or **output verification**, or **testing pattern**	H.2.18.13 H.2.18.15 H.2.18.3 H.2.18.8 H.2.18.11 H.2.18.12 H.2.18.22
7.2.2 Analog multiplexer	Wrong addressing	rq	 rq	**Plausibility check** Comparison of redundant CPUs by either: – **reciprocal comparison** – **independent** hardware **comparator**, or **input comparison** or **testing pattern**	H.2.18.13 H.2.18.15 H.2.18.3 H.2.18.8 H.2.18.22
8. Monitoring devices and **comparators**	Any output outside the static and dynamic functional specification		rq	**Tested monitoring**, or **redundant monitoring** and comparison, or **error recognizing means**	H.2.18.21 H.2.18.17 H.2.18.6
9. Custom chips [f] for example, ASIC, GAL, Gate array	Any output outside the static and dynamic functional specification	rq	 rq	Periodic self-test Periodic self-test and monitoring, or **dual channel (diverse) with comparison**, or **error recognizing means**	H.2.16.6 H.2.16.7 H.2.16.2 H.2.18.6

CPU: Central programmation unit

rq: Coverage of the **fault** is required for the indicated software class.

[a] Table H.1 is applied according to the requirements of H.11.12 to H.11.12.2.12 inclusive.

[b] For **fault**/error assessment, some components are divided into their subfunctions.

[c] For each subfunction in the table, the software class C measure will cover the software class B **fault**/error.

[d] It is recognized that some of the acceptable measures provide a higher level of assurance than is required by this standard.

[e] Where more than one measure is given for a subfunction, these are alternatives.

[f] To be divided as necessary by the manufacturer into subfunctions.

H.11.12.2.5 Measures others than those specified in H.11.12.2.4 are permitted if they can be shown to satisfy the requirements listed in Table H.1.

H.11.12.2.6 Software **fault**/error detection shall occur not later than the time declared in requirement 71 of Table 1. The acceptability of the declared time(s) is evaluated during the **fault** analysis of the **control**.

Part 2 standards may limit this declaration.

H.11.12.2.7 For **controls** with functions, classified as Class B or C, detection of a **fault**/error shall result in the response declared in Table 1, requirement 72. For **controls** with functions declared as class C, **independent** means capable of performing this response shall be provided.

H.11.12.2.8 The loss of **dual channel** capability is deemed to be an error in a **control** function using a **dual channel** structure with software class C.

H.11.12.2.9 The software shall be referenced to relevant parts of the **operating sequence** and the associated hardware functions.

H.11.12.2.10 Where labels are used for memory locations, these labels shall be unique.

H.11.12.2.11 The software shall be protected from **user** alteration of safety-related segments and data.

H.11.12.2.12 The software and safety-related hardware under its control shall be initialized to, and terminate at, a declared state as indicated in Table 1, requirement 66.

H.11.12.3 Measures to avoid errors

Void.

H.11.12.3.1 General

For **controls** with software class B or C the measures shown in Figure H.1 to avoid systematic **faults** shall be applied.

Measures used for software class C are inherently acceptable for software class B.

The content of this is extracted from IEC 61508-3 and adapted to the needs of this standard.

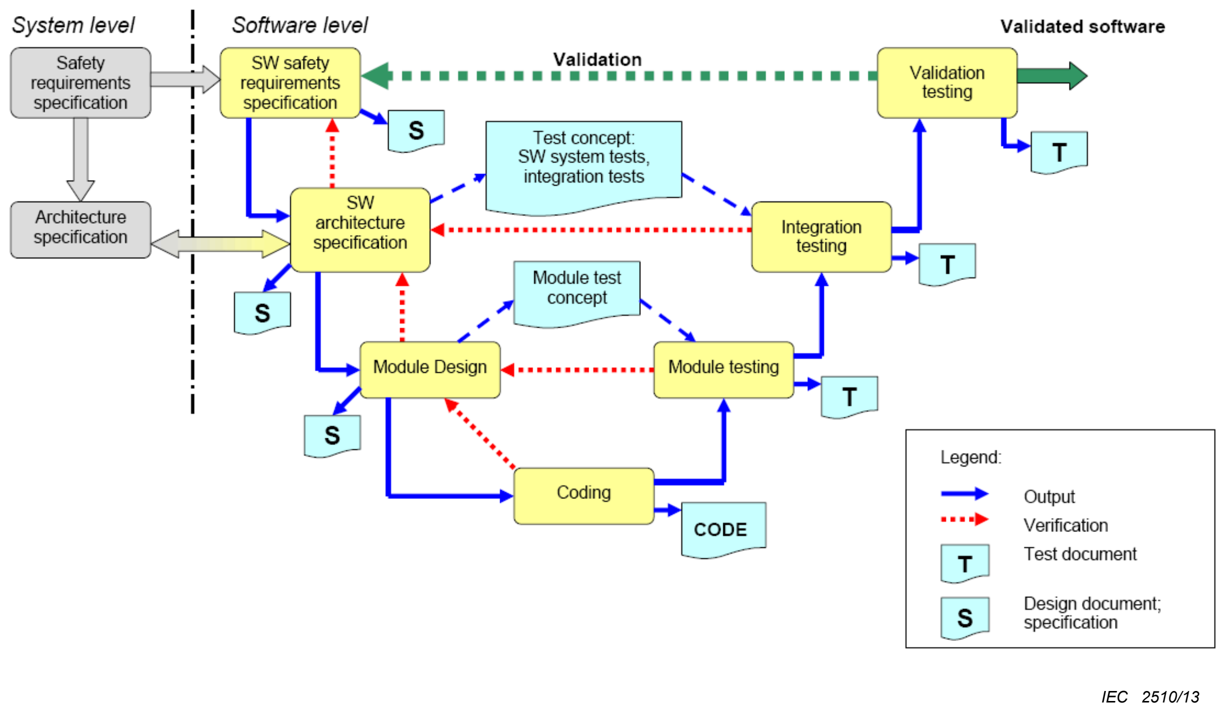
System level
Software level
Validated software
Safety requirements specification
SW safety requirements specification
Validation
Validation testing
S
T
Test concept: SW system tests, integration tests
Architecture specification
SW architecture specification
Integration testing
S
T
Module test concept
Module Design
Module testing
S
T
Coding
CODE
Legend:
Output
Verification
Test document
Design document; specification
IEC 2510/13

Figure H.1 – V-Model for the software life cycle

Other methods are possible if they incorporate disciplined and structured processes including design and test phases.

H.11.12.3.2 Specification

H.11.12.3.2.1 Software safety requirements

H.11.12.3.2.1.1 The specification of the software safety requirements shall include:

– a description of each safety related function to be implemented, including its response time(s):
  - functions related to the application including their related software classes;
  - functions related to the detection, annunciation and management of software or hardware **faults**;

– a description of interfaces between software and hardware;

– a description of interfaces between any safety and non-safety related functions.

Examples of techniques/measures can be found in Table H.2.

Table H.2 – Semi-formal methods

Technique/Measure	**References (informative)**
Standards identification	
Semi-formal methods – Logical/functional block diagrams – Sequence diagrams – Finite state machines/state transition diagrams – Decision/truth tables	B.2.3.2 of IEC 61508-7:2010 C.6.1 of IEC 61508-7:2010

Other methods to comply with the requirements can be applied.

H.11.12.3.2.2 Software architecture

H.11.12.3.2.2.1 The description of software architecture shall include the following aspects:

– techniques and measures to control software **faults**/errors (refer to H.11.12.2);
– interactions between hardware and software;
– partitioning into modules and their allocation to the specified safety functions;
– hierarchy and call structure of the modules (**control** flow);
– interrupt handling;
– data flow and restrictions on data access;
– architecture and storage of data;
– time based dependencies of sequences and data.

Examples of techniques/measures can be found in Table H.3.

Table H.3 – Software architecture specification

Technique/Measure	**References (informative)**
Fault detection and diagnosis	C.3.1 of IEC 61508-7:2010
Semi-formal methods:	
– Logic/function block diagrams	
– Sequence diagrams	
– Finite state machines/state transition diagrams	B.2.3.2 of IEC 61508-7:2010
– Data flow diagrams	C.2.2 of IEC 61508-7:2010

H.11.12.3.2.2.2 The architecture specification shall be verified against the specification of the software safety requirements by static analysis.

NOTE Acceptable methods for **static analysis** are:
– **control** flow analysis;
– data flow analysis;
– **walk-throughs**/design reviews.

H.11.12.3.2.3 Module design and coding

NOTE 1 The use of computer aided design tools is accepted.

NOTE 2 For Defensive Programming (for example, range checks, check for division by 0, **plausibility checks**), see C.2.5 of IEC 61508-7:2010.

H.11.12.3.2.3.1 Based on the architecture design, software shall be suitably refined into modules. Software module design and coding shall be implemented in a way that is traceable to the software architecture and requirements.

The module design shall specify:

– function(s),
– interfaces to other modules,
– data.

Examples of techniques/measures can be found in Table H.4.