SOFTWARECPR®
CRISIS PREVENTION AND RECOVERY, LLC

IVD INSTRUMENT TRAINING EXAMPLE

This is not a template or cookbook.

It has pros and cons and is partial and to be used for training discussions only.

It includes comments for consideration.

It shows the evolution of some items as the lifecycle progresses.

It is a vertical slice partial example not a document set.

It has been constructed specifically for our Software Standards Training Course by Raffaele Caliri of SoftwareCPR®

1. Introduction

This example is using an IVD application to illustrate the relationship between different technical documents and their relevance to the IEC 62304 standard.
The documents we are addressing span from construction phase to system test phase of a project.

To limit the scope of this example to a manageable size we shall take into consideration a specific slice of the project. It is not the purpose of this example to fully develop this slice, only to give an idea of what is required in terms of documentation and what could be the process. For this reason, in this example we shall proceed by exemplifying portions of the documents, without any pretence to be complete.

We shall show how risk analysis is identifying Risk Control Measures that add to or modify the original software requirements

We shall show how testing is applied to the critical requirements derived from risk control measures.

We shall address the role of traceability to demonstrate coverage of all risk control measures.

2. Description of the application

The application of this example is a Blood Gas Analyzer, measuring pH (Hydrogen ion activity), pO_2 (Oxygen partial pressure) and pCO_2 (Carbon Dioxide partial pressure) on whole blood. These parameters are used to monitor the efficiency of the oxygen exchange at pulmonary level and are of vital importance to keep under control a patient under ventilation.
It is an instrument meant to be used in a wide range of situations and environments: ER, ICU, Laboratory, etc.
The tests performed are not routine tests. They are critical care tests, the blood needs to be analyzed immediately after extraction and the results are often needed as soon as possible.
In addition to that, the blood used for these tests is usually arterial blood, which means that the extraction is not very easy and could in some cases be slightly traumatic.
This originates the following main requirements for the instrument:
- Being always ready for use: all samples are STAT samples.
- Being very reliable to avoid the need to repeat the sample collection.
- Having a pretty fast response time, in the order of few minutes worst case.

The instrument does not perform any direct diagnostic or therapeutic intervention, but the physician is basing diagnostic and therapeutic decisions, that in some cases could be life threatening, on the results of the analysis carried out on this instrument.

The basic responsibilities of software in this instrument are:
- To control the fluidic cycles to aspirate the blood and bring it in contact with the electrochemical sensors.
- To control the acquisition and conversion of the signal from the sensors.
- To calculate the quantitative results from the acquired signals.
- To present the results to the user.

Therefore a software misbehavior could cause erroneous results to be presented to the user and erroneous results could lead the physician to apply a wrong therapy that could potentially result in a major injury or death. In our view this behavior would lead to classify the software of this device as **Class C**, according to the indications of **clause 4.3 of IEC 62304**.

Note however that the software of this type of device is usually classified as "moderate level of concern" in FDA submissions. This is probably due to the fact that usually a physician does not base his/her therapeutic decisions exclusively on the results of Blood Gas Analysis, and this somewhat mitigates the risk of life threatening therapeutic decisions based on wrong results generated by software misbehavior.

If we want to be conservative, we should classify this software **Class C**.

3. Description of the slice

The Concept Specification or Marketing Requirement Specification shall indicate for each measured parameter requirements related to the measure range and the precision and accuracy of the results within the measure range.
It could be something like:

Analytical Specifications
(at 90 % confidence level)

Parameter	Range	Max Inaccuracy	Max Imprecision
pH	from 6.900 to 7.700	± 15 mpH	< 5.0 mpH SD
pCO2	from 8.0 to 29.9 mmHg	± 2.0 mmHg	< 2.0 % CV
	from 30.0 to 59.9 mmHg	± 2.0 mmHg	< 1.5 % CV
	from 60.0 to 90.0 mmHg	± 3.0 % of theor. value	< 1.5 % CV
pO2	from 10 to 29 mmHg	± 1.5 mmHg	< 4.0 % CV
	from 30 to 99 mmHg	± 2.0 mmHg	< 2.0 % CV
	from 100 to 199 mmHg	± 3.0 % of theor. value	< 1.5 % CV
	from 200 to 399 mmHg	± 3.0 % of theor. value	< 2.0 % CV
	from 400 to 700 mmHg	± 4.0 % of theor. value	< 2.0 % CV

Let's assume that the measure sensors for each of the parameters have a linear response in the specified measure range or a response that, despite not being fully linear, could be made sufficiently close to linear in the specified measure range by applying some linear correction algorithms empirically determined by the observation of the behavior of a sufficient number of selected sensors.

In this case a linear calibration algorithm based on the acquisition of the values of two calibration standards and the equation of a straight line could be used. The calculated calibration parameters will be the offset and slope of the straight line used during analysis to calculate the quantitative values of the blood sample measures.

The vertical slice of this example shall focus on the software required to support the calibration of the instrument.
Since a misbehavior of the calibration software could affect the results presented to the user, this software component, according to **section 4.3(d) of IEC 62304**, inherit the same class of the main software (**class C**).

4. Requirements related to calibration.

In this section some examples of requirements related to the calibration functionality are presented. It does not intend to be a complete set of requirements, but just some hints to the different types of requirements that need to be written to define this critical portion of the software project.

System Level Requirements:
In the System Requirement Specification, document that is outside the scope of IEC 62304, the need for a calibration should be covered with requirements of the type
Before the instrument is ready for use, a two points calibration shall be performed on all measured parameters by presenting two standards of known composition and calculating the offset and the slope of the calibration curve.

Software Level Requirements:
The following software requirements are just examples developed to show compliance with **clauses 5.2.1 and 5.2.2 of IEC 62304**. They are largely incomplete and are mostly functional requirements, although we can identify among them also input and output, software driven alarms, usability requirements, data definition requirements.

User Interface.
- [UISRS111] When the instrument is not calibrated (after switch on or after a calibration failure), the main screen shall present the indication "NOT READY: UNCALIBRATED" and a "CALIBRATE" soft key.
- [UISRS123] Pressing the "CALIBRATE" soft key shall start the Calibration Process. During calibration, the main screen shall present the indication "NOT READY: CALIBRATION IN PROGRESS" and a "STOP" soft key.

Comment [RC1]: Note that the system requirement here is not complete and left pretty generic because it is outside the scope of this example. Additional requirements could be present such as the required standard composition, the sequence of operation, the level of accuracy, etc.

Comment [RC2]: The use of tags helps identifying requirements for tracing purpose. Tag numbers used here are irrelevant: they just show that each requirement needs to be identified by an unique tag. Also the use of letters to identify different functional areas in the spec is totally arbitrary.

Comment [RC3]: Note that this requirement could require more details related to the interface between the User Interface and the Process Control

Comment [RC4]: Note that these requirements are pretty basic and of course the user interface could be much more complicated and imply driving the operator to execute whatever action is required during the calibration (e.g. presenting the calibrants, starting the aspiration etc. in case the calibration is not automatic with calibrants on-board.)

- [UISRS124] Pressing the “STOP” soft key shall abort the calibration and the main screen shall revert to present “NOT READY: UNCALIBRATED”.
- [UISRS132] If the Calibration Process is successfully completed, the main screen shall present the indication “READY” and a “SAMPLE” soft key.
- [UISRS133] Pressing the “SAMPLE” soft key shall start the Sampling Process to measure a Patient or Control sample.
- [UISRSxxx] ……..

Process Control.

- [PCSRS099] The process control software shall drive the peristaltic pump motor at speed XXXX for YYY seconds to aspirate the calibrant and bring it in contact with the sensors.
- [PCSRS100] After ZZZ seconds from the end of the aspiration, the process control software shall start the acquisition of the calibrant values by activating the reading from the sensors.
- [PCSRS111] When reading is complete, the process control software shall drive the peristaltic pump motor at speed AAAA for BBB seconds to remove the calibrant and flush the lines.
- [PCSRSxxx] ……..

Data Acquisition.

- [DASRS011] The data acquisition software shall read the XX bits A/D converter every YYY seconds.
- [DASRS012] To increase resolution and remove eventual spikes, a digital filtering of order Z shall be applied to the acquired data. The final resolution shall be 3 decimal digits.
- [DASRS055] When start acquisition command is received, a data ready algorithm shall be applied to the acquired (filtered) values to ensure that a stable plateau is reached: the acquisition result shall be returned when the difference among N subsequent acquisition is constantly <MMM.
- [DASRS077] In case the acquired data do not reach the stability (data ready) condition within SSS seconds from start of acquisition, the data acquisition software shall stop the acquisition and return an “equilibration error”, that shall cause an “END POINT” alarm to be presented to the user.
- [DASRSxxx] …….

Calibration.

- [CALSRS033] For each parameters, two calibration standards shall be measured and the acquired data shall be used to calculate the calibration coefficients. The resolution for this calculation shall be at least N decimal digits….
- [CALSRS034] The slope calibration coefficient shall be calculated as

  m = (Ref 2 - Ref 1) / (X 2 - X 1)

  where Ref1 and Ref2 are the title values respectively for Calibration standard 1 and Calibration Standard 2 and Y1 and Y2 are the acquired values for the two standards.
  The figure below illustrate the meaning of the above algorithm (the orientation

Comment [RC5]: Note that some hardware interface requirements could be needed to specify how the motor is driven

Comment [RC6]: Additional requirements could be needed in case liquid sensors are used instead of or in addition to time to stop aspiration.

Comment [RC7]: Additional requirements could be needed here to specify the interface between the process control and the data acquisition.

Comment [RC8]: Note that some hardware interface requirements could be needed to specify how the A/D converter is managed.

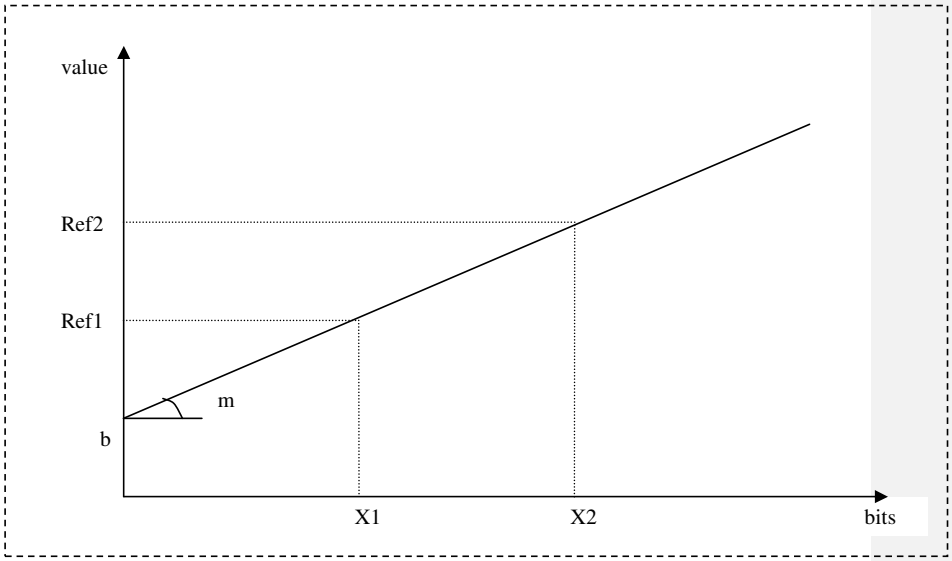
Comment [RC9]: The details of the filter could be left to design, but some characteristics, such as final resolution of the data should be specified as requirement because they directly affect the accuracy of the final results and therefore are connected to the analytical specification.

Comment [RC10]: This is a pretty generic way to specify a validity criterion for the result. It could be differently specified, with more or less detail. It is important that it is specified what is needed to achieve a final accuracy compatible with the analytical specifications.

Comment [RC11]: This is just an example of a possible error handling requirement: more details could be added to fully define the handling.

Comment [RC12]: It is necessary to define requirements allowing to meet the analytical spec.

of the line is purely indicative, the slope could be positive or negative):



- [CALSRS035] The offset calibration coefficient shall be calculated as
  b = Ref1 – mX1
- [CALSRS011] The results of any patient or QC sample measured after calibration shall be obtained applying the linear equation
  Y = mX + b
  and then applying the following linearity corrections:
  - ……
  - ……
  - ……
- [CALSRS050] The following checks shall be applied during calibration:
  - The X1 acquired value shall be in the range aaa ≤ X1 ≤ bbb
  - The X2 acquired value shall be in the range ccc ≤ X2 ≤ ddd
  - The calculated slope coefficient shall be in the range eee ≤ m ≤ fff
  - The calculated offset coefficient shall be in the range ggg ≤ b ≤ iii
  - ……..

  In case the calibration does not pass the checks, the CAL ERROR alarm shall be raised and the instrument shall revert to the "not ready: uncalibrated" status.
- [CALSRSxxx] ……..

Comment [RC13]: Linearity corrections will be required if the response of the sensor is not fully linear or not linear in the entire measure range.

Comment [RC14]: This is just a simple example, the requirements on checks and alarm could be more detailed and the subsequent actions could be more complicated: for example, multiple trials could be specified before raising the alarm.

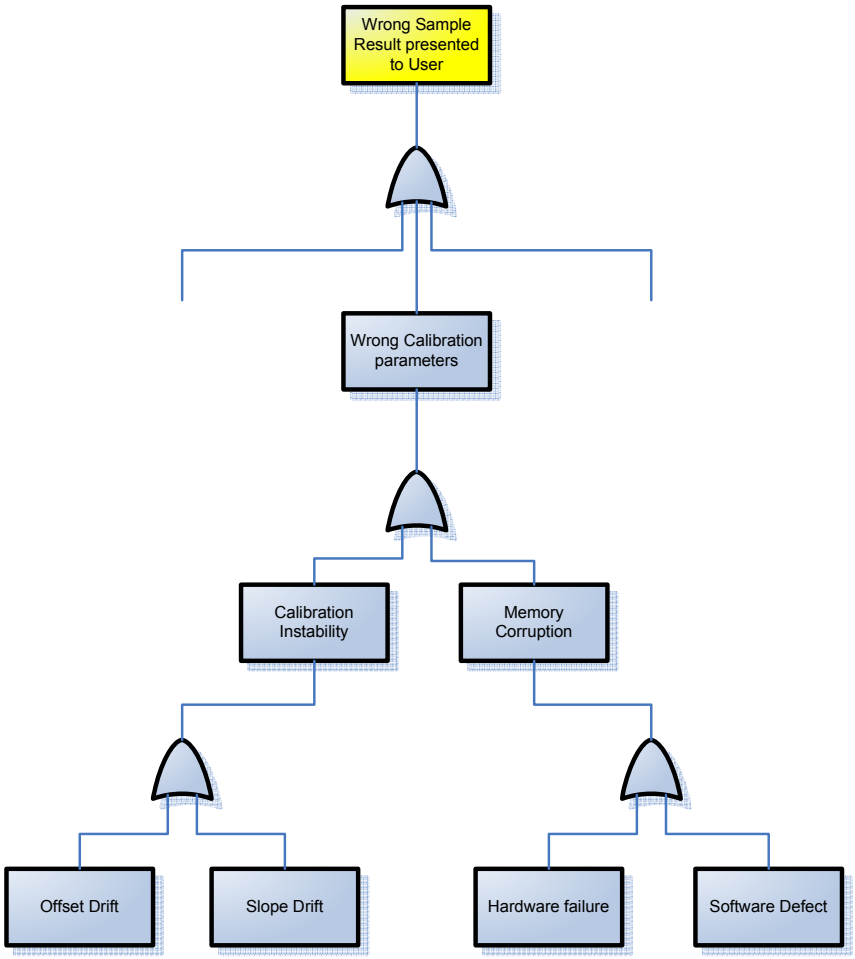
It is important to note that the Analytical Specification defined in the Concept or Marketing spec are driving many of the requirements related to data acquisition, data reduction and calibration: from the resolution in the acquisition to the resolution in the calculations, from the selection of the calibration points to the definition of the linearity corrections, from the definition of the limits for data stability to the definition of the limits for calibration range.

It is necessary to define all these detailed requirements in a way to fulfill the budget of variation allowed by the accuracy and precision requirements of the Analytical Specifications.

5. Risk analysis examples.

Of the several risks that could affect the calibration functionality, this section shall identify two examples: the first example refers to an application related risk with risk control measures implemented in software, the second one to possible hardware or software defects requiring risk control measures implemented in software. This allows to show application of **clauses 7.1.1, 7.1.2 and 7.2 of IEC62304**.

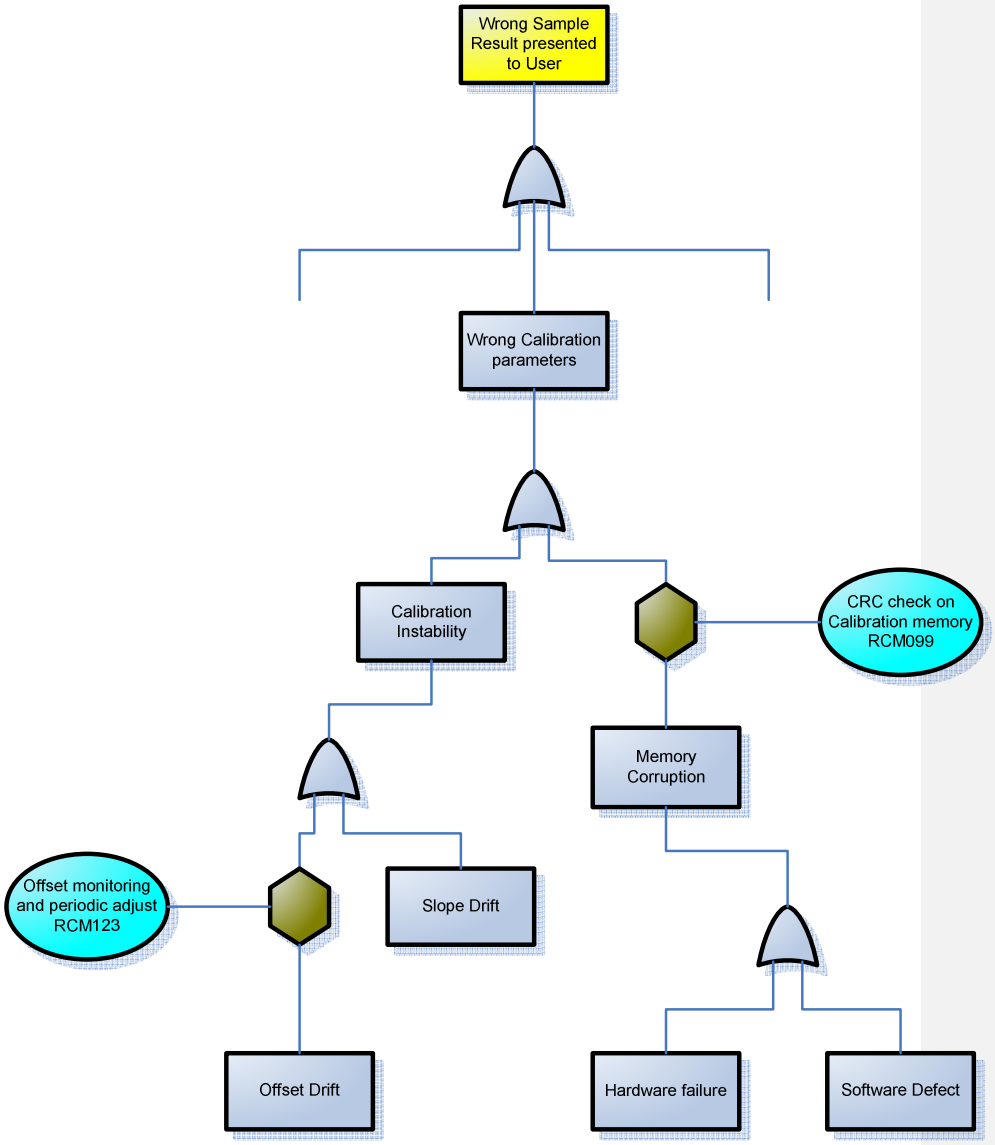
The following partial and simplified FTA diagram illustrates these examples.



Example 1: calibration stability.
Let's assume that the initial calibration is not stable over time because of a change of the sensor characteristics. This is a pretty common behavior, and is usually mitigated with the requirement to repeat the calibration once in while, where the frequency of the repetition is related to the expected behavior of the sensor.
But let's assume that the risk of a drift on the offset over time is considerably high and much higher than the risk of a drift on the slope.
The risk analysis performed has indicated that the severity of this risk is high (a drift in calibration, if not addressed, could mean wrong or highly inaccurate results and, as consequence wrong diagnosis and wrong therapy) and that the probability of this to happen is also significantly high, therefore a mitigation is needed.
As **Risk Control Measure** we can define a frequent check of the offset by repeated reading of one of the calibration standards and, when the drift reaches a certain level, trigger the execution of one point recalibration to update the offset coefficient.

Example 2: calibration parameters corruption.
Let's assume that the calibration parameters are kept in memory, where there is the risk of corruption caused by a software defect (stack overflow? Not properly handled memory allocation? ….) or by a hardware failure.
Also in this case the severity of this risk is high (wrong calibration coefficients means wrong results) and, since the cause could be a software defect, probability is high.
As **Risk Control Measure** we can put a CRC on the section of memory used to store the calibration coefficients. The CRC is updated every time data are written in that section and it is checked frequently. In case of CRC error, the execution of a two point calibration is triggered.

The FTA diagram is modified to include the risk control measures.



6. Additional calibration requirements related to RCMs.

The implementation of the risk control measures identified for the two examples above implies additional requirements to be added to the set of calibration requirements exemplified in section 4, as requested by **clauses 7.2.2 and 5.2.3 of IEC 62304**.

In particular, at system level the need of automatic recalibration shall impose a system design that keeps the calibration standards on-board and allows to bring them in contact with the measure sensors without operator's intervention. This shall have impacts on the mechanical/fluidic hardware (e.g. having calibration standard containers on board, valves to aspirate from different lines, possibility to automatically rinse, etc.), as well as on electronics and, of course, on software. Some examples of additional software requirements made necessary by the RCMs are following.

Example 1: RCM123.

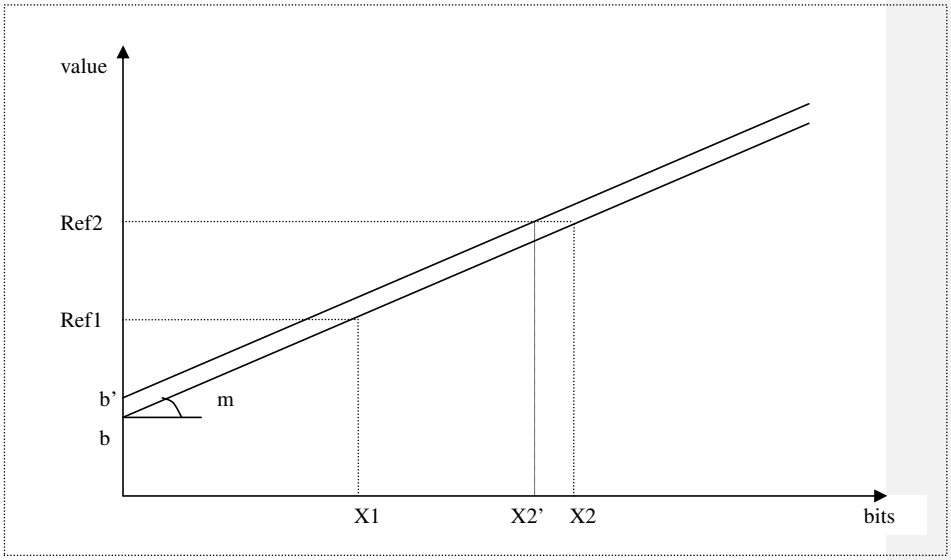
- [PCSRS200] After the 2 points calibration is completed, the process control software shall keep the second calibration standard in contact with the measure sensors.
- [PCSRS201] Every XXX minutes, the process control software shall drive the peristaltic pump motor at speed YYYY for ZZZ seconds to renew the standard in contact with the sensor.
- [DASRS100] After 2 Points Calibration is completed, every AAA seconds the data acquisition software shall perform a new acquisition and check the acquired value for evaluating the drift from the value used during previous calibration. If the drift is > XXX, a 1 point offset calibration update shall be requested.
- [CALSRS123] When 1 point calibration is requested, a new offset shall be calculated based on the value of the second calibration standard. It is equivalent of a translation of the calibration straight line parallel to itself as illustrated from the figure below.
  The offset calibration coefficient shall be updated as follows:

  b' = Ref2 – mX2'

Comment [RC15]: This requirement is replacing the requirement to remove the standard at the end of calibration.

Comment [RC16]: This could be needed if the standard is expected to modify its property over time when kept in the measure chamber.

Comment [RC17]: The drift could checked as absolute value or as percentage change.



Example 2: RCM099.

- [EHSRS111] Every time the calibration coefficients are calculated and stored, a CRC shall be calculated and appended according to the following CRC algorithm: ……..
- [EHSRS121] Whenever the calibration coefficients are used, a check shall be performed on the CRC. In case of error a WRONG CALIBRATION alarm shall be raised.
- [EHSRS122] Every XXX seconds, the software shall verify the CRC of calibration parameters. In case of a CRC error, the software shall:
    - Raise a WRONG CALIBRATION alarm
    - Trigger the automatic execution of a 2 points calibration.

Now it is important to note that there are cases where the implementation of a risk control measure, while on one side is reducing a risk and allowing to keep it under control, on the other side could originate a different risk, requiring the implementation of additional risk control measures.

Let's use the Example 1 to illustrate this. We have identified a risk of drift on the calibration offset and implemented a risk control measure that corrects for that drift when it occurs. However repeatedly correcting for the drift of the offset could mask other problems of calibration: for example, what if the change in the reading of the standard is not totally due to a drift of the offset, but there is also a component that is modifying the slope? Correcting for the offset as indicated above, in this case we are only reaching a sufficient accuracy for patient samples that are in a range close to the value Ref2 of the second calibration standard, but could generate pretty bad results for patient samples that are far away from that value.
Let's assume that the probability of such behavior increases with the increase of the perceived change in the offset. A risk control measure, in this case, would be to set limits to the ability to correct the calibration with one point offset correction and raise a calibration error alarm and/or automatically execute a two points calibration in case these limits are reached.

Additional requirements to implement the additional RCM could be as follows.

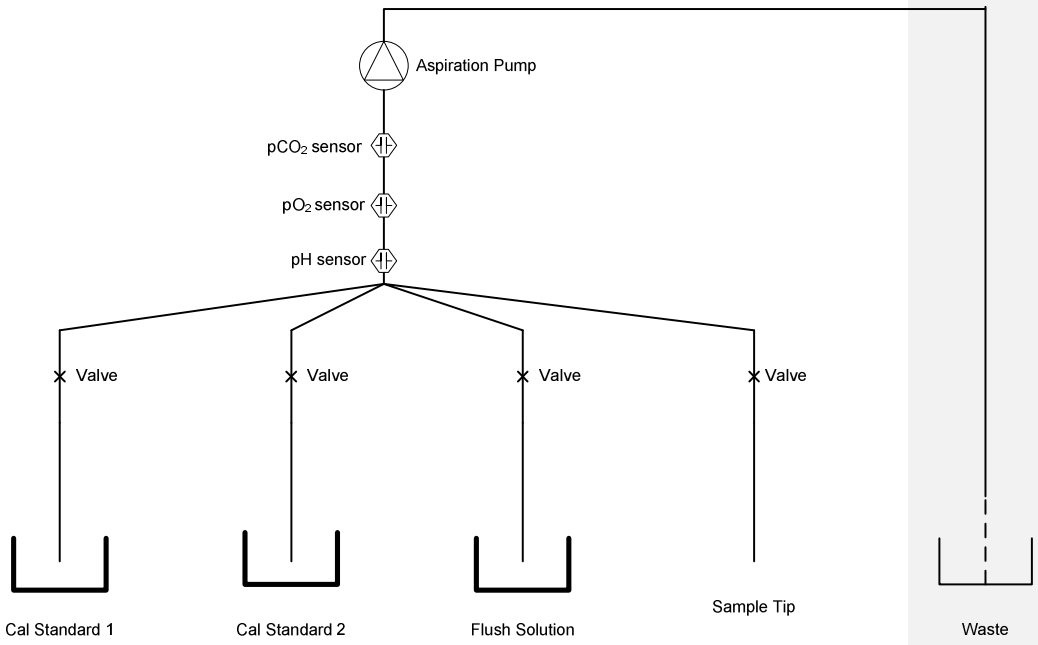
- [CALSRS155] Every time the calibration offset is updated with 1 pt cal, the calibration software shall check that the change from the original calibration offset calculated during 2 pt cal is < XXX. In case it is ≥ XXX, the software shall:
    - Raise a WRONG CALIBRATION alarm
    - Trigger the automatic execution of a 2 points calibration.

Comment [RC18]: The change could checked as absolute value or as percentage change.

This shows compliance with **clauses 5.2.4 and 7.3.2 of IEC 62304**.

7. Design.

The following drawing illustrates a simplified schematic of instrument's fluidic path in the case the calibration standards need to be present on-board.
The real structure could be more complicated than the one here depicted, especially when we consider the need to be able to fully wash and decontaminate the fluidic path to avoid carry over between samples or with the calibration standards, but for sake of simplicity let's just use, as illustrated in the drawing, a single peristaltic pump to aspirate either one of the calibration standard or the patient/QC sample.
The fluidic valves on the lines allow to selectively aspirate what is required in the different phases of the instrument operation.



The software shall need to command the pump and the valves to aspirate the required solution until it is in contact with the sensors, activate the reading from the sensors, elaborate the data read from the sensors to present the results to the user, then command the pump and the valves to wash the lines with flush solution.

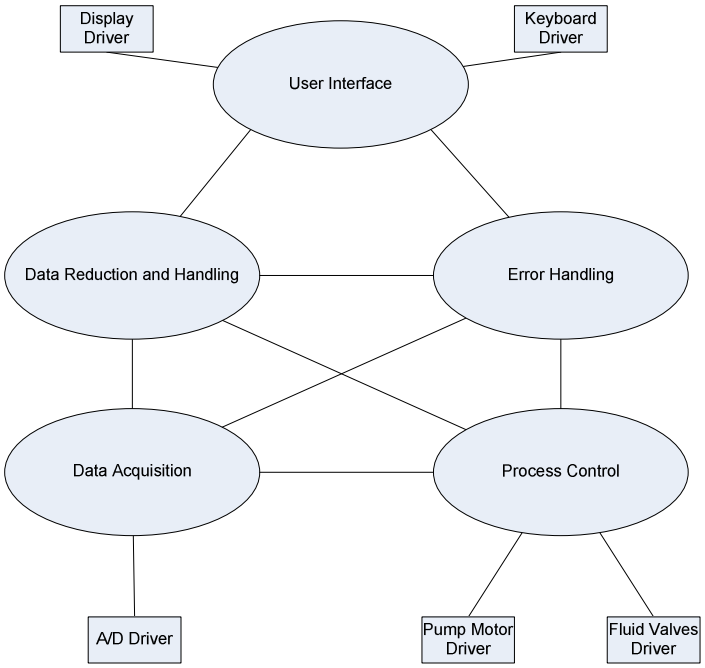
From the software architecture point of view (see **clause 5.3 of IEC 62304**), the simplified diagram below shows what could be the main components for the slice that we are considering in this example.
In this decomposition:

- The Data Acquisition process is responsible for selecting the parameter to acquire, reading the data from the A/D Converter, applying filtering, checking data stability, returning the filtered stable result to the Data Reduction process.

Comment [RC19]: This does not mean to be complete (of course the instrument shall have other components) nor to imply that this would be the best architectural solution for this software project.. It is just an example of decomposition that could satisfy the needs of the project.

- The Process Control process is responsible for controlling the sequence of operations of the machine cycles and controlling the mechanical and fluidic hardware.
- The Data Reduction and Handling process is responsible for executing the calculations involved in the calibration and in patient results and for appropriately storing the results of this handling.
- The Error Handling process is responsible for managing the checks on process and data, for starting recovery actions and for raising alarms when needed.
- The User Interface process is responsible for getting input from the operator and for displaying results, status and error messages.
- The interfaces between the different processes could be through message queues to obtain decoupling of the different processes.

Comment [RC20]: A different design solution could have two different processes



Each of the processes shall have further architectural decomposition and detailed design for each of the components, as requested by **clause 5.4 of IEC 62304**, but detailed design of each component is outside the scope of this example.
For example, for the Data Acquisition process we could identify the following components:
- A/D component:
    - Selects a parameter.
    - Requests the A/D driver for an acquisition

Comment [RC21]: The driver itself could manage all the acquisition timing.

- Checks the validity of the converted data (e.g. we could raise underflow or overflow error and discard the data, if the value is too close to 0 or to the full scale of the A/D converter).

- Filter component:
  - Accumulates acquired data in 3 different memory buffers (one for each parameter).
  - Performs filtering as requested: for example, a simple filtering could be a moving average of xx acquisition points, discarding the highest and lowest value in the series before averaging.
  - Returns the filtered results for each parameter.
- Data ready component:
  - For each parameter applies a specific algorithm on the filtered results to determine when the result is stable and can be used for further processing.
  - Stops the acquisition on each parameter, when it is stable.
- Interface component:
  - Receives the command to start acquisition from the Process Control process.
  - Sends the results to the Data Reduction process, when all parameters are stable.
  - Informs Process Control that acquisition is complete, when all parameters are stable.
  - Interfaces to Error Handling process for alarms processing.

Comment [RC22]: The check on validity could be eventually performed by the driver, which in such case would return the result plus a validity flag.

Comment [RC23]: Further detailed design would describe buffers handling, e.g. usage of pointers for circular buffers.

Comment [RC24]: Filtering need depends of the expected noise and could be as simple as the described averaging algorithm or as complex as a FFT algorithm.

Comment [RC25]: Algorithms could be different for different parameters. Examples of algorithms could be:
-Plateau algorithm: the result is considered stable when first derivative is close to 0.
-Peak algorithm: the result is considered stable when there is an inversion in the sign of first derivative.

Comment [RC26]: A data ready timeout, to address the problem of results that never get stable, could be managed internally to this process or externally by other processes (EH or Process Control)

8. Testing.

In order to fully test the calibration and especially the RCMs discussed in this example, it is necessary to create error conditions that it is difficult to generate at system level:

- to cause and verify the offset correction of 1 point calibration, it is necessary to cause a drift in the acquisition data
- to cause an error on calibration data CRC it is necessary to generate a corruption in the data stored in memory.

For this reason the calibration functionality cannot be fully tested at system level, but it needs to have some of the testing performed white box or grey box with the use of tools apt to create the error conditions.

To act on the acquisition data to generate drift or other error conditions we could conceive a hardware tool that acts on the analog signal at the input of the A/D converter or a software simulation tool that acts on the acquired data. The first one has the advantage of allowing to test the entire chain, including the A/D driver, but the second one is probably easier to implement and control and would not need the use of external hardware. To allow to test the most of the software in the data acquisition/data reduction chain, the simulation tool should act at the lowest level in the chain, replacing with simulated data the data acquired at A/D level in the A/D driver. The tool could remain on-board in the final instrument software and be possibly activated via password protected service program.

Comment [RC27]: It is outside the scope of this example to fully define the capabilities and the design of this tool.

Comment [RC28]: Note that a simulator on-board, besides the use for testing, could be of use also to marketing, during presentations, to allow to run the instrument in known conditions without the use of calibration standards and of real samples.

To allow verifying the results of testing, a password protected service program should be developed to access and display and/or print data that are normally internal and not shown to the user:
- Calibration coefficients.
- Acquisition data at different stages in the data acquisition/data reduction chain
- Timers
- Error flags

To allow forcing a CRC error, a password protected service program should be developed, allowing to write into memory.

Testing needs to be applied at different stages.

Static testing (design review and code inspections) are essential to verify that the design and the implementation fulfill the requirements. For example, the requirement about the resolution in the calculation of the calibration coefficients is easy to check through a code inspection, could be more difficult to test dynamically. Code inspections are a way to implement software unit acceptance as requested by **clause 5.5.3 of IEC 62304**, while design reviews implement verification of architecture and detailed design as requested by **clauses 5.3.6 and 5.4.4. of IEC 62304**.

Algorithms, such as the filtering algorithm, the data ready algorithm, the calibration coefficients calculations should be tested at unit level in order to be able to verify a wide variety of situations and of combinations of data, including limit conditions that would be best identified if you know the details of the algorithm.
For example, let's imagine that the plateau algorithm, instead of using the first derivative, has been implemented by checking that at least N consecutive acquisitions are within ±xx%.
Unit testing could be used to verify if and how this algorithm fulfills the requirements at different level of final value: when the plateau is very low (few A/D points) or very high (close to the full scale of the A/D).
Unit testing could be used to verify fast reaching and slow reaching of the plateau, to verify limit situations where the data at reaching the plateau are not increasing or decreasing regularly, but are jumping up and down but still within the range fulfilling the plateau criteria, situations where the acquired data start low and grow until the plateau is reached or vice versa start high and decrease.
Unit testing, together with code inspection, is part of the software unit verification requested by **clauses 5.5.2 through 5.5.5 of IEC 62304**.

Integration testing could be applied in this example to verify the entire chain of data acquisition and data reduction, or to verify integration of the Data Acquisition process with the Process Control process and the Error Handling process by checking for example a condition where a plateau is not reached in the specified time and an END POINT error is generated.
Integration testing is requested by **clauses 5.6.3 through 5.6.7 of IEC 62304**.

Software system testing is then applied to the fully integrated system as requested by **clause 5.7 of IEC 62304**.
The following is an example of a system level test case that could make use of the described tools to verify some of the RCMs related to calibration as requested by **clause 7.3.1 of IEC 62304**.

BG raf / Test Case – CALxxx

VERSION:	**xx.x**	**DATE CREATED:**	**15 September 2008**
TEST TYPE:	SW System	**CREATED BY:**	**John Brown**
TEST TITLE:	**Calibration Drift Check**		
TEST OVERVIEW:	This test case verifies the following calibration drift handling: • If difference in acquisition data on second calibrant is within the limits, the calibration coefficients are not modified. • If difference in acquisition data on second calibrant is outside the limits, the offset calibration coefficients are updated. • If the drift on the calibration offset, reaches the limit, a calibration error is generated and a 2 points calibration is triggered.		

DEPENDENCIES All actions required to place the system in the proper state prior to executing the test.	
Hardware Preparation:	Standard instrument. No need of calibrants on-board.
Software Preparation:	Software version xxx.x, including the acquisition simulation tool.
Other / Set-up:	The instrument must be on, warm up completed, uncalibrated.

SUPPORTING LINKS	
Document References:	**SRS xxxx, RA yyyy, ……**
Automated Scripts:	**None**
SRS Ids	**CALSRSxxx, CALSRSyyy, UISRSzzz, ……**
Change Ids:	
Test Data Sets:	**Simulation sets:** **CAL1STD_normal: simulates a normal data sequence for CAL1** **CAL2STD_normal: simulates a normal data sequence for CAL2** **CAL2STD_fixed: maintains indefinitely normal CAL2 value** **CAL2STD_drift_low: simulates a drift on CAL2 within the limits** **CAL2STD_drift_high: simulates a drift on CAL2 exceeding limits** **CAL2STD_drift_err: simulates a drift on CAL2 causing calibration repetition**

TEST Inputs and Expected Results			
Step	**Actions**	**Expected Results**	**Reference**
1.	Activate the service program to display in a window internal information.	The debug window appears in the specified section of the screen.	SRVSRSxxx

Comment [RC29]: More detail is normally needed to specify actions and results, but this is just an example

Comment [RC30]: Traceability information to the spec., note that wherever possible we use the specific tags we have used in the spec examples to show how traceability works.

2.	Select to display the acquired data at the stage they are passed to the data reduction and the calibration coefficient.	The debug window shows blank acquisition data (no activity is in progress) and default calibration coefficients.	
3.	Activate acquisition simulation and select the simulation set **CAL1STD_normal**		
4.	Press the "CALIBRATE" soft key	The main window displays "NOT READY: CALIBRATION IN PROGRESS" with a "STOP" soft key. The CAL STANDARD 1 is aspirated according to fluidic cycle.	UISRS123, CALSRSyyy, PCSRS099
5.	Observe the debug window.	After xx seconds from the end of aspiration, data acquisition is started. The acquired data on pH grow from aaa to bbb, then remain fixed. The acquired data on pO_2 grow from ccc to ddd, then remain fixed. The acquired data on pCO_2 grow from eee to fff, then remain fixed.	PCSRS100, DASRS055,
6.	When data are fixed on all 3 parameters, select the simulation set **CAL2STD_normal**	The CAL STANDARD 2 is aspirated according to fluidic cycle.	CALSRSxxx, PCSRS099
7.	Observe the debug window.	After xx seconds from the end of aspiration, data acquisition is started. The acquired data on pH grow from ggg to hhh, then remain fixed. The acquired data on pO_2 grow from iii to jjj, then remain fixed. The acquired data on pCO_2 grow from kkk to lll, then remain fixed.	PCSRS100, DASRS055,
8.	When data are fixed on all 3 parameters, select the simulation set **CAL2STD_fixed**	The main window displays "READY" with a "SAMPLE" soft key. The pH calibration coefficients are set: offset = qqq, slope = mmm. The pO_2 calibration coefficients are set: offset = rrr, slope = nnn. The pCO_2 calibration coefficients are set: offset = sss, slope = ooo. The acquired data remain fixed with pH = hhh, pO_2 = jjj and pCO_2 = lll	CALSRS034, CALSRS035, UISRS132,

9.	Select the simulation set **CAL2STD_drift_low**	The pH acquired data is increased to ttt. The pO2 acquired data is decreased to uuu. The pCO2 acquired data is increased to vvv. These values are close to, but within the limits, therefore no correction is done to the calibration coefficients. They remain the same of step 8	DASRS100,
10.	Select the simulation set **CAL2STD_drift_high**	The pH acquired data is increased to tt1. The pO2 acquired data is decreased to uu1. The pCO2 acquired data is increased to vv1. These values are just outside the limits for offset adjustment. The offset calibration coefficients are recalculated, while the slope coefficients remain the same. pH offset = qq1, slope = mmm. pO_2 offset = rr1, slope = nnn. pCO2 offset = ss1, slope = ooo.	DASRS100, CALSRS123
11.	Select the simulation set **CAL2STD_drift_err**	The pH acquired data is increased to tt2. The pO2 acquired data is decreased to uu2. The pCO2 acquired data is increased to vv2. This brings the recalculated offsets outside the limits of allowed offset drift recovered by 1 point calibration. A WRONG CALIBRATION alarm is presented. A 2 points recalibration is started. The main window displays "NOT READY: CALIBRATION IN PROGRESS" with a "STOP" soft key.	CALSRS155, UISRS123,
12.			

Comment [RC31]: This is just one example of testing limit conditions. Several sets of data could be used. This test case does not mean to be complete. Same considerations apply to the following steps.

Comment [RC32]: In the real world the testing should be more detailed. We could have just one parameter outside the limits, triggering a 2 points recalibration. We could test one parameter at a time.

Comment [RC33]: Additional steps could require verifying the recalibration and involve the use of other simulation sets.

9. Traceability.

Traceability can be achieved in different ways, through tags in the document text as we have shown in this example or by using documentation management tools.
In this example we need to trace the RCMs to their testing through the requirement specification of the RCM to fulfill the requirements of **clause 7.3.3 of IEC 62304**.

For example the RCM123 "Offset monitoring and periodic adjust" shall trace to the requirements:

- [PCSRS200] After the 2 points calibration is completed, the process control software shall keep the second calibration standard in contact with the measure sensors. Every XXX minutes, the the process control software shall drive the peristaltic pump motor at speed YYYY for ZZZ seconds to renew the standard in contact with the sensor.
- [DASRS100] After 2 Points Calibration is completed, every AAA seconds the data acquisition software shall perform a new acquisition and check the acquired value for evaluating the drift from the value used during previous calibration. If the drift is > XXX, a 1 point offset calibration update shall be requested.
- [CALSRS123] When 1 point calibration is requested, a new offset shall be calculated based on the value of the second calibration standard. It is equivalent of a translation of the calibration straight line parallel to itself as illustrated from the figure below.
  The offset calibration coefficient shall be updated as follows:

b' = Ref2 – mX2'

The requirement [CALSRS123] shall trace to step 10 of the system test example presented in previous section. Of course it shall trace also to other steps or to other test cases, because step 10 is only covering some conditions.
It shall also trace to calibration design, wherever the design defines the details of the calibration algorithm.
Traceability needs to be established this way for all the requirements to demonstrate test coverage and especially test coverage of all critical requirements and Risk Control Measures.

10. Appendix: IEC 62304 Cross Reference.

The following table shows what clauses of the standard are addressed or referenced in this example. The first column lists the clauses of the standard, the second column indicates in what section of the example they are referenced (left blank if not referenced in the example, in brackets if partially or indirectly addressed), the comments in the third column explain in more detail at what level they are addressed.

ANSI/AAMI/IEC 62304	Referenced in Section	Comments
4.1 Quality management system		Not addressed in this example
4.2 Risk Management	(5)	The example only address some aspects, not the entire process.
4.3 Software safety classification	2, 3	The example mainly addresses sub-clauses a) and d)
5 Software Development Process		
5.1 Software development planning		Planning not addressed in this example
5.2 Software Requirements Analysis	4, 6	
5.2.1 Define and document software requirements from System Requirements.	4	
5.2.2 Software requirements content	4	Examples of requirements of different types are given, but not all the types are covered. Some are indicated as necessary but not fully developed
5.2.3 Include risk control measures in software requirements.	6	
5.2.4 Re-evaluate medical device risk analysis	6	
5.2.5 Update system requirements		Not addressed in this example
5.2.6 Verify software requirements		Not addressed in this example
5.3 Software Architectural Design	7	
5.3.1 Transform software requirements into an Architecture.	7	Only limited to the slice described in the example
5.3.2 Develop an architecture for the interfaces of the software items	(7)	In the example we only give an hint, interfacing not fully developed.
5.3.3 Specify functional and performance requirements for SOUP items.		SOUP is not addressed in the example
5.3.4 Specify system hardware and software required by SOUP items.		SOUP is not addressed in the example
5.3.5 Identify segregation necessary for risk control.		Segregation not addressed in the example
5.3.6 Verify Software Architecture	(8)	Only hints given
5.4. Software Detailed Design	(7)	Discussed but not developed
5.4.1 Refine software architecture into software units.	(7)	Only initial example given
5.4.2 Develop detailed design for each software unit.	(7)	Only initial example given
5.4.3 Develop detailed design for interfaces		Not addressed in this example
5.4.4. Verify detailed design	(8)	Only hints given
5.5 Software Unit Implementation and Verification	(8)	Not specifically addressed in the example.
5.5.1 Implement each software unit		Implied, not developed

ANSI/AAMI/IEC 62304	Referenced in Section	Comments
5.5.2 Establish Software Unit Verification Process	(8)	Some hints given.
5.5.3 Software unit acceptance criteria	(8)	Some hints given.
5.5.4 Additional Software unit acceptance criteria	(8)	Some hints given.
5.5.5 Software unit verification	(8)	Some hints given.
5.6 Software integration and integration testing	(8)	
5.6.1 Integrate software units		Implied, not developed
5.6.2 Verify software integration		Implied, not developed
5.6.3 Test integrated software	(8)	Some hints given.
5.6.4 Integration testing content	(8)	Some hints given.
5.6.5 Verify integration test procedures	(8)	Some hints given.
5.6.6 Conducts regression test	(8)	Some hints given.
5.6.7 Integration test records contents	(8)	Some hints given.
5.6.8 Use software problem resolution process		Not addressed in this example
5.7 Software System Testing	(8)	
5.7.1 Establish tests for software requirements.	8	An example given
5.7.2 Use software problem resolution process		Not addressed in this example
5.7.3 Retest after changes		Not addressed in this example
5.7.4 Verify software system testing	(8)	Some hints given.
5.7.5 Software system test record content	(8)	Some hints given.
5.8 Software Release		Release not addressed in this example
6 Software Maintenance Process		Maintenance not addressed in this example
7 Software Risk Management Process		
7.1 Analysis of software contributing to hazardous situations	(5)	
7.1.1 Identify software items that could contribute to a hazardous situation	5	
7.1.2 Identify potential causes of contribution to a hazardous situation	5	
7.1.3 Evaluate published SOUP anomaly lists		SOUP is not addressed in the example
7.1.4 Document potential causes	(5)	
7.1.5 Document sequences of events	(5)	
7.2 Risk Control Measures	5	
7.2.1 Define risk control measures	5	
7.2.2 Risk control measures implemented in software	5	
7.3 Verification of Risk Control Measures	6,8,9	
7.3.1 Verify risk control measures	8	
7.3.2 Document any new sequence of events	6	
7.3.3 Document traceability	9	
7.4 Risk Management of Software Changes		Software Changes not addressed in this example
8 Software Configuration Management Process		Software Configuration Management not addressed in this example
9 Software Problem Resolution Process		Problem Resolution Process not addressed in this example